Network Architectures
and Services
NET 2012-07-3

Dissertation

Bilateral and Multilateral Negotiation for Agreement Discovery and Formation

Dipl.-Inf. Andreas Klenk

Network Architectures and Services
Department of Computer Science
Technische Universität München

TECHNISCHE UNIVERSITÄT MÜNCHEN
Institut für Informatik
Lehrstuhl für Netzarchitekturen und Netzdienste

Bilateral and Multilateral Negotiation for Agreement Discovery and Formation

Andreas Klenk

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Florian Matthes
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. Georg Carle
	2.	Univ.-Prof. Dr. Dr. Alexander Schill Technische Universität Dresden

Die Dissertation wurde am 02.03.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 22.06.2012 angenommen.

Cataloging-in-Publication Data
Andreas Klenk
Bilateral and Multilateral Negotiation for Agreement Discovery and Formation
Dissertation, July 2012
Network Architectures and Services, Department of Computer Science
Technische Universität München

ISBN 3-937201-32-7
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)
DOI: 10.2313/NET-2012-07-3
Network Architectures and Services NET-2012-07-3
Series Editor: Georg Carle, Technische Universität München, Germany

Abstract Negotiation is one of the most important tools for the coordination of collaborations and for conducting trade in market economies. Collaborations are becoming short-lived, task-driven, and spontaneous. Electronic auction protocols are in wide spread use in the Internet and create large markets for commodities. However, auction protocols are restricted to negotiations where typically price is the only variable. Auctions provide no means to negotiate about the terms of an agreement. This asymmetric trading relationship puts the buyer at a disadvantage as the buyer either accepts the terms under which an item is for sale or there is no deal at all.

This dissertation focuses on agreement negotiation protocols that allow for the negotiation about the terms of an agreement. Policy intersection protocols are efficient in identifying mutually compatible parameters. The involved stakeholders formulate policies with permissible options. This thesis demonstrates how single-round policy intersection serves as a tool to establish mutually compatible security parameters and how to use the agreement to establish secure end-to-end communication channels in a layer independent manner. Agreements reached with policy intersection protocols are restricted to the available options set forth in the policies. Policy intersection is a suitable method if the negotiating parties know the possible parameter space beforehand and if they can formulate their policies accordingly. Policy intersection is no tool to discover new and unforeseen agreements.

Iterative requirements-driven agreement negotiation with the novel VersaNeg protocol overcomes these limitations. It allows each party to state what it can offer and what it wants in turn. An iterative exchange of requirements and offers leads to a rooted tree structure of alternative agreement options. The agreement negotiation protocol becomes a tool to discover new agreement options. Only few related negotiation protocols are capable of multilateral negotiations. The VersaNeg agreement negotiation protocol works equally well for bilateral and multilateral negotiations. In multilateral negotiations with VersaNeg, the involved parties discover collaborations on the fly and establish comprehensive agreements that govern the terms of the collaboration. Comprehensive agreements can be interpreted without any knowledge about the message exchange during the negotiation. The comprehensive agreement alone is sufficient to define all relevant statements made during the negotiation.

Electronic negotiations can only reach a widespread deployment if the security of the negotiation can be guaranteed and if the negotiators can proof the outcome of the negotiation to a third party. Iterative signatures with message state reduction is a novel security technique that protects bilateral and multilateral agreements against forgery. It assures the integrity and authenticity of messages during the negotiation and allows negotiators to detect manipulations of negotiation messages. After an agreement has been reached, a third party can verify the authenticity of each statement in the agreement.

Zusammenfassung Die Aushandlung von Übereinkünften ist eines der wichtigsten Werkzeuge um die Zusammenarbeit verschiedener Parteien zu koordinieren und um Handel zu treiben. Kollaborationen sind zunehmend kurzlebig, projektgetrieben, und werden spontan gebildet. Protokolle für elektronische Auktionen sind im Internet weit verbreitet und schaffen große Märkte für standardisierte Güter. Auktionsprotokolle sind jedoch zumeist auf Verhandlungen beschränkt, bei denen der Preis die einzige Variable ist. Auktionen sind nicht dafür gedacht über die Bedingungen einer Übereinkunft zu verhandeln. Diese asymmetrische Beziehung benachteiligt potentielle Käufer, weil der Käufer entweder die Bedingungen akzeptiert, unter denen ein Gut zum Verkauf steht und ein Gebot abgibt, oder kein Handel zustande kommt.

Diese Dissertation hat ihren Schwerpunkt auf Aushandlungsprotokollen, die es erlauben über die Bedingungen einer Übereinkunft zu verhandeln. Policy Intersection Protokolle sind effizient um für alle Parteien akzeptable Parameter zu finden, und um allen beteiligten Akteuren Kontrolle über die Menge der möglichen Übereinkünfte zu geben. Diese Arbeit demonstriert wie Policy Intersection eingesetzt werden kann, um gegenseitig akzeptable Sicherheitsparameter für eine Kommunikationsverbindung zu finden. Die Übereinkünfte die man mit Policy Intersection erzielen kann, beschränken sich jedoch auf die Möglichkeiten, die in der statisch definierten Regelbasis jeder beteiligten Partei vorhanden sind.

Das neue VersaNeg Protokoll überwindet mit seiner iterativen anforderungsgetriebenen Aushandlung diese Beschränkungen. Jede Partei drückt aus was sie anbieten kann und was sie als Gegenleistung erwartet. Durch einen iterativen Austausch von Anforderungen und Angeboten entsteht eine Baumstruktur mit alternativen Übereinkünften. Damit wird das Aushandlungsprotokoll zu einem Werkzeug um neue Übereinkünfte zu entdecken. Nur wenige verwandte Ansätze eignen sich auch für die Mehrparteienaushandlung. Das VersaNeg Protokoll unterstützt gleichermaßen bilaterale und multilaterale Aushandlungen. In Mehrparteien Aushandlungen mit VersaNeg werden Kollaborationen während der Verhandlung entdeckt und in vollständigen Übereinkünften festgehalten. Vollständige Übereinkünfte können ohne Wissen um den Nachrichtenaustausch, der während der Verhandlung statt gefunden hat, interpretiert werden. Sie enthalten alle für die Übereinkunft relevanten Informationen, die während der Aushandlung ausgetauscht wurden.

Elektronische Aushandlungsprotokolle können nur eine große Verbreitung finden, wenn die Sicherheit der Verhandlung garantiert werden kann, und wenn die beteiligten Parteien das Ergebnis der Verhandlung einer dritten unbeteiligten Partei gegenüber nachweisen können. Iterative digitale Signaturen mit Reduktion des Nachrichtenzustands ist eine neue Sicherheitstechnik um bilaterale und multilaterale Verhandlungen gegen Manipulationen zu schützen. Sie stellt die Integrität und Authentizität der Verhandlungsnachrichten sicher und erkennt Manipulationen am Nachrichtenzustand. Nachdem eine Übereinkunft erzielt wurde, kann eine dritte Partei die Authentizität aller Teile der Übereinkunft prüfen.

Acknowledgements This research would not have been possible without the guidance and input of many individuals. First, I want to thank my advisor Prof. Dr. Georg Carle for his faith in my ideas and for granting me the opportunity to pursue this research direction with the support of Universität Tübingen and Technische Universität München. Despite his busy schedule, he gave me many opportunities to discuss matters at hand and to give advice on my research. I am very grateful for Prof. Dr. Dr. Alexander Schill to agree to be co-examiner. I also want to thank Prof. Dr. Florian Matthes for kindly organizing the examination process.
Many thanks go to my colleagues who created a friendly and productive working environment where ideas could flourish and constructive feedback was given.
It was a great pleasure to work together with numerous thesis students and HiWis. This thesis would not have produced the same results without the the discussions, the creative input, and the collaboration with Frank Petri, Hannes Angst, Andreas Beck-Greinwald, and Marcus Masekowsky.

The warmest thanks goes to my loving wife which supported my decision to go back to research and who always encouraged me during this thesis. My wife and my two children Sofia and Elias have always been a great source of strength. Without their support, this thesis would be unimaginable.

Contents

List of Figures

List of Tables

List of Listings

1. Introduction

The emergence of the Internet changed the way people and businesses interact. There is no precedent in history for a communication medium that is open to so many people and that allows for such a direct and timely communication[1].
This new flexibility has large impacts on the way businesses can perform their trade. The new communication technologies enable that a company conducts business with an abundance of potential trading partners. According to economic theories [MacD94], the increased competition leads to lower prices and higher quality of products. However, the group of potential trading partners is still limited by the human to human interactions, which are necessary for companies to discover mutually compatible business objectives and opportunities to collaborate.

Electronic agreement negotiation is an important tool to allow a company to reach a much larger group of potential trading partners.
Electronic auctions [KeNT00] have gained a significant market volume. A large body of research on electronic auctions puts its focus on intelligent bidding strategies. The participating agents use comparable simple negotiation protocols to make their bids. The research on auctions is highly relevant to business because it allows for efficient price determination in open markets.
The drawback of auctions is, that the asymmetric trading relationship puts the buyers often at a disadvantage. They have to accept the terms and conditions under which an item is sold or there is no deal at all. Complex collaborations with interdependent partners cannot be established with current auction based marketplaces because negotiations are limited to price only. Instead, companies are still forced to negotiate complex agreements directly between humans with little electronic support.

The challenge to adapt to the objectives and capabilities of a remote party not only exists for business to business interactions, but also on a technical level. There is a large number of special purpose protocols to coordinate actions between remote systems. For example, TLS [DiRe06] uses a multi-round handshake protocol to agree upon cryptographic algorithms and parameters, to establish a secure communication channel. However, each protocol standardizes its own method how to establish an agreement about session parameters.
Negotiation is also an important tool to realize self-organization. Multi-agent systems

[1] 1966 Million Internet Users - Internet Usage and World Population Statistics, June 30, 2010 - http://www.internetworldstats.com

rely on negotiation protocols to coordinate the actions of a group of agents [Krau97]. Negotiation is considered as a core technology to realize self-management in autonomic systems [GaCo03].
Distributed systems must not only reach consensus about purely technical parameters. Interactions can easily involve distinct legal entities and therefore might have legal consequences [Hoer09]. Under these circumstances, negotiation protocols must be able to produce legally binding agreements.

The challenge is to design agreement negotiation protocols that are equally well-suited for reaching a consensus about technical parameters and for establishing legally binding agreements on behalf of human stakeholders.

1.1 Problem Statement

Research on negotiation protocols that caters for more than just price is still in its infancy. In order to become general purpose tools for defining terms and conditions of agreements, it is highly desirable that electronic agreement negotiation protocols realize these functionalities:

1. *Agreement negotiation* can be divided into two functions
   (a) *Agreement discovery* is an iterative process to identify how negotiators can reach an agreement. Does the protocol discover new agreement options that are compatible with the objectives of the negotiating parties? Share the parties the same understanding of the agreement under negotiation? Do all parties have the same ability to state their requirements and offers during the negotiation? Are the interrelationships of the negotiation messages easy to understand and automatically verifiable?
   (b) *Agreement formation* leads to a definition of the agreement that has to be met. Is the reached agreement clearly defined? Does the agreement unambiguously state what each party has to fulfill and under what dependencies?
2. *Comprehensive agreements* can be interpreted without any knowledge about the message exchange during the negotiation. The comprehensive agreement alone is sufficient to define all relevant statements made during the negotiation. Does the negotiation protocol produce comprehensive agreements?
   (a) With comprehensive agreements, everybody knows what to do and under which circumstances.
   (b) In front of court, comprehensive agreements are much easier to interpret because they contain the disputed provisions. The court does not have to understand all interactions during the negotiation process, but can rely instead on the information in the comprehensive agreement.
3. An agreement defines the dependencies of obligations, permissions and prohibitions between two or more parties. Does the protocol produce agreements for more than two parties?
   (a) *Bilateral agreements* are between two parties. Even if the negotiation involves more negotiators, the final agreement involves only two parties.

   (b) *Multilateral agreements* define the interdependencies between a group of parties. Even in multilateral settings, bilateral contracts are predominant nowadays.

4. *Agreement languages* standardize syntax and semantics how to interpret agreements. Can the protocol leverage existing language constructs for defining the content of the agreement?
5. Electronic negotiations can only become a general purpose tool if a *high level of security* can be guaranteed. Does the negotiation protocol guarantee a high level of security, even for multilateral negotiations?
   (a) Authenticity and integrity are necessary to clearly state who made what statement during the negotiation.
   (b) Non-repudiation is also known as accountability. A party in a dispute cannot refute the validity of a statement made during the negotiation.
   (c) Confidentiality assures that only legitimate receivers can decrypt and access sensitive information during the negotiation.

The problem is that even though protocols exist that have some of the properties stated above, none of the existing protocols comes close to covering all these desirable features. The research objective of this thesis is *to develop a secure negotiation protocol for agreement discovery and formation of comprehensive bilateral and multilateral agreements.*

1.2 Research Methodologies

This thesis proposes different protocols for realizing the declared objectives. The following methodologies have been applied at different stages of the research.

1. **Model Methodology** The first step in this research deals with the definition of a generic negotiation model. From this model of negotiation systems, one can derive properties of real implementations. The generic model also allows for a separation of the negotiation protocol from complementary technologies, such as the integration into business processes or the interaction with some decision intelligence.
2. **Formal Methodology** The formal modeling of the negotiation protocols allows for unambiguous protocol specifications. The formal model is the prerequisite for the analysis of time and space complexity of the proposed algorithms.
3. **Build Methodology** The build methodology is very important in computer science to demonstrate that a proposed technology not only works under some overly simplistic assumptions of the corresponding formal models. The peril is that the formal model only partially captures the constraints experienced in real world scenarios and fails to predict real world behavior. The build methodology proofs that a given technology is doable and is an important step towards the realization of software products that benefit from the realization of the thesis objectives.
4. **Experimental Methodology** This methodology is divided into an exploratory phase for defining what questions should be answered by the experiments and for identifying the relevant parameters for these experiments. The evaluation phase draws conclusions from the test results. The experimental phase of this thesis followed a process with stringent record keeping of tests and test parameters.

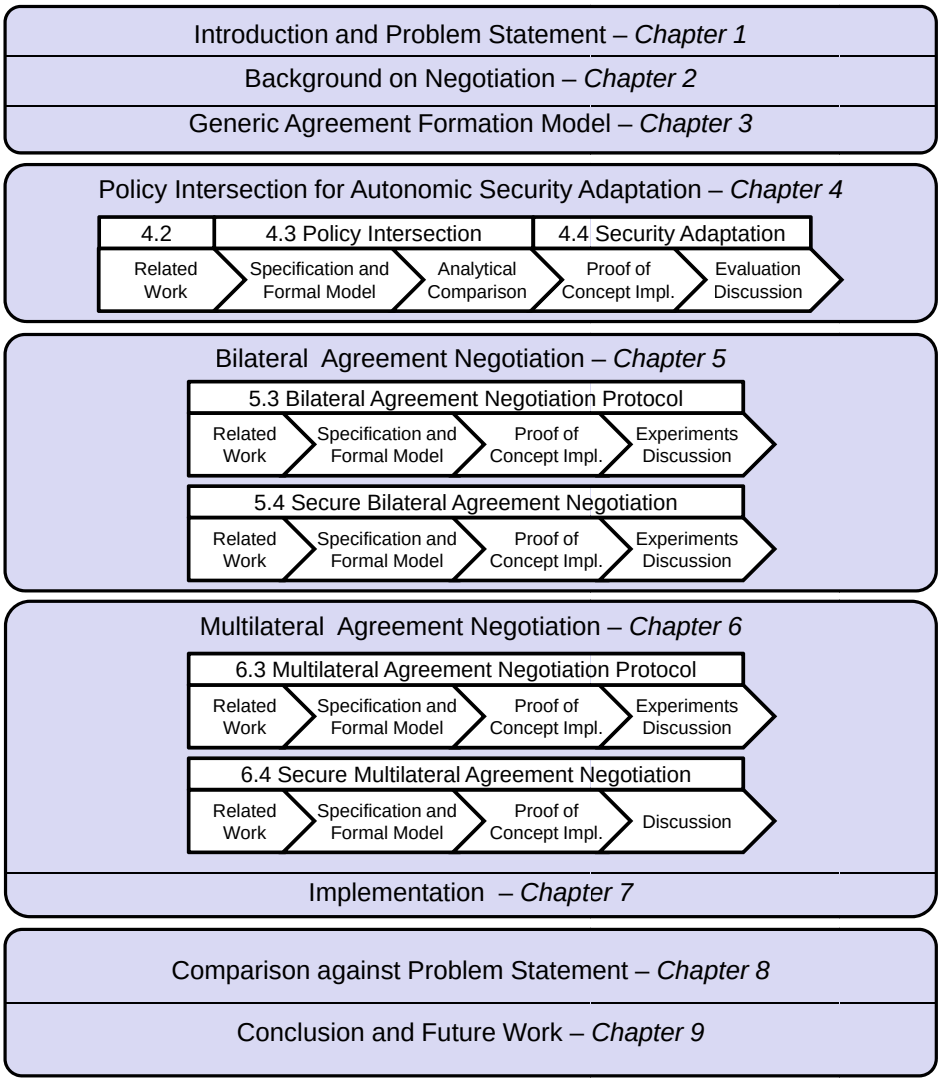

Figure 1.1: *Thesis Organization*

1.3 Thesis Organization

The following chapters will introduce different protocols to address the problem statement from above. The analysis of the juridical and economic background on negotiations in Chapter 2 helps identifying the underlying concepts for legally binding agreement negotiation. Additionally, this chapter gives an overview of different research communities that deal with electronic negotiation, but none of these communities has a focus on establishing general purpose agreement negotiation protocols. Chapter 3 derives the generic agreement formation model from the market transaction model [ScLi98]. The generic agreement model helps to define different activities for handling agreements and allows to clearly state what a negotiation protocol can achieve and what tasks other systems have in the overall agreement formation process.
Chapter 4 deals with policy intersection as an efficient and secure method to establish mutually compatible parameters between two parties. An analytical comparison shows that single-round policy intersection is more efficient than iterative policy intersection in the Internet. The second focus of this Chapter is to demonstrate with a fairly complete system how an agreement negotiation protocol can be used for the layer independent establishment of secure communication channels. The autonomic security adaptation

with the novel Extensible Security Adaptation Framework (ESAF) shows how policies are defined, how policy intersection between a local and a remote system identifies compatible session parameters, how to automatically choose one option with a utility function, and how to establish secure communication channels automatically. Policy intersection is at the heart of this approach to establish mutually compatible security parameters. It allows full control for all stakeholders in the involved systems. Policy intersection is easy to implement and can cover many well defined use cases. However, it can not discover any agreement outside the pre-defined policies. Policy intersection is therefore not suitable for electronic contracting, but it is highly relevant for protocol handshakes.

The iterative agreement negotiation in Chapter 5 overcomes the limitations of policy intersection. First comes an analysis of related work to identify capabilities and limitations of these negotiation approaches. The idea of requirements-driven iterative agreement negotiation with the novel VersaNeg negotiation system is presented next. The protocol relies on the iterative matching of requirements of one party with offers by another party. Offers are turned into obligations when a certain agreement option will be turned into the final agreement. The protocol forms comprehensive agreements in an iterative message exchange. Because the negotiation state grows at each iteration, the protection of negotiation states does not work with existing security algorithms. Existing algorithms are unable to separate authorized changes from the immutable parts of a negotiation state during a processing step. This thesis introduces alternating signatures with message state reduction to allow each negotiator to verify the integrity and authenticity of a negotiation state for the whole negotiation. Experiments on the protocol performance with and without security give a good indication how this protocol behaves for different scenarios and under various negotiation loads.

There are few protocols capable of establishing multilateral agreements through an agreement negotiation. Chapter 6 extends the VersaNeg approach to form multi-party agreements and to discover collaborations with the negotiation protocol. VersaNeg uses comprehensive negotiation states that contain all information that has been exchanged during the negotiation up to this point in the negotiation. The comprehensive negotiation states are now used in a ring protocol where each party can receive the negotiation state, inspect all relevant requirements and make own offers. The ring protocol assures that each party gets to know all relevant changes to the negotiation state. Analytical comparisons and experimental performance analysis show the how the protocol behaves in different environments and scenarios. Security is an integral part of the negotiation protocol for agreement negotiation in multilateral settings. This chapter extends the alternating signatures with message state reduction to the ring protocol with multiple parties.

Next comes Chapter 7 that explains further details of the VersaNeg implementation and protocol. For instance, this chapter describes failure handling in case one party drops out of the negotiation, and it introduces the detailed XML representation of negotiation states.

Chapter 8 puts the agreement negotiation approaches from this thesis into comparison with popular agreement standards. A systematic analysis states the distinguishing features of the protocols and it details the trade-offs and limitations of using the different approaches. The comparison shows, that VersaNeg has some unique capabilities that no other analyzed protocol offers, but also that the use case determines which agreement approach is the most favorable under the given circumstances. The Montreal Taxonomy for Electronic Negotiations is applied to VersaNeg and ESAF to further characterize the protocol behavior.

Finally, this thesis concludes in Chapter 9 with a list of contributions of this thesis and an outlook on future work that has to be done before the novel agreement negotiation approaches are ready for use in productive environments.

For a tree to become tall it must grow
tough roots among the rocks

Friedrich Nietzsche

2. Background on Negotiation

Before we outline the major research areas on negotiation, we will first define what negotiation is. According to the Oxford English Dictionary [SiWe89] negotiation is *"a discussion or process of treaty with another (or others) aimed at reaching an agreement about a particular issue, problem, etc"*. According to the same source an agreement can express *accordance in sentiment, opinion, action, or purpose; harmony, concord; absence of dissension.*

The following definition of the term negotiation stresses additionally that negotiation is an integrative process for parties with diverting interests and captures very well the motivations of this thesis:

> Negotiation is a dialogue intended to resolve disputes, to produce an agreement upon courses of action, to bargain for individual or collective advantage, or to craft outcomes to satisfy various interests. [Wiki09]

2.1 The Legal Perspective on Contracts

Negotiation leads to agreements which might have legal implications. However, there is no one single legal interpretation of contracts on a global scale. It depends on the legislation what meaning a contract has and what legal obligations arise from a contract. According to Beale et al. [Zimm96] "Contract law has many *purposes*, but the central one is to support and to control the millions of agreements that collectively make up the *market economy*". Contracts are formed every day: for the sale or purchase of goods, for the provisioning and consumption of services, for lending or investing money, and for labor and wages.

This Section presents historic background on contracts, the interpretation under the common law which is prevalent in most Anglophone countries, and the interpretation under German contract law. This Section presents relevant background for electronic contractual negotiation and does not intend to give a comprehensive legal overview of contracts.

2.1.1 Contract and Obligation in Historical Context

Negotiation is at the heart of most business activities. The etymology of negotiation goes back to the Latin noun *negotiatio* and means *"to carry on business"*. The legal

background for the interpretation of negotiation and for establishing binding contracts evolved during the centuries in different legislations.
Many relevant concepts of contracts haven already been established in Roman Law [Zimm96]. The *obligation* is a fundamental concept in Roman Law. An obligation is a two-ended relationship that appears from one end as a right to claim and from the other end as a duty to render performance. The party that is bound to deliver the promised performance is called the *debtor* whilst the other party is the *creditor*. Before the legal system of obligations was introduced and enforced by the State authority, people had to enforce agreements by sheer power. As it was understood in ancient societies, a party had the right of vengeance if another party inflicted harm against the body or the property of this party. The legal concept of enforceable obligations by legal means superseded the right of vengeance. The victim's right of vengeance was redeemable by the wrongdoer. The state standardized the amount of compensation for various delicts. If the wrongdoer was unable to compensate the victim, the victim had the right to inflict harm on the wrongdoer. This mechanism put a large pressure on obligated parties to stick to their obligations.
Contracts rely heavily on the concept of obligations. Zimmermann [Zimm96] describes how parties use obligations to form contractual liabilities: "if one party wanted to obligate another to make a specific performance, he would ask the latter to subject himself to the power of seizure in case he failed to perform." Historically, this construct was in wide spread use for lending money. The debtor was liable to the creditor if he did not redeem himself by paying a specific sum of money back in time.

Following these concepts, Prisacariu and Schneider [PrSc07] use deontic logic to model contracts as obligations, permissions, and prohibitions of one party towards another. A contract may include the following statements that party p_2 makes towards party p_1:

- **Obligation:** Party p_2 has to take some course of action, which benefits p_1.
- **Permission:** Party p_2 has to grant rights to party p_1.
- **Prohibition:** Party p_2 will refrain from some action.

2.1.2 English Contract Law

English contract law is the basis of the legal regulation not only in Great Britain, but also in most countries of the Commonwealth and in the United States. It is important to know that English law is case law. The courts make new interpretations, by using inductive reasoning to apply former rulings to new areas. Case law mainly derives legal rules from such precedents, instead of statutory law where statutes and codes are enacted by the legislation. The precedents and to a smaller extent the legislation together form the English contract law.
An agreement in contract law is a voluntary "meeting of minds". The involved parties interact during the formation of a contract until they reach a common understanding about the contract. An agreement constitutes a contract if all involved parties agree about the same thing and they intend the agreement should be legally binding. Legally binding means here, that the parties bound by a contract have rights and obligations, which can be enforced by the courts, if necessary. In simple words, a contract contains promises that are recognized by law and that can be enforced through the state.
Under English contract law, a contract has the following seven main characteristics [Tilb06]:

1. **Offer and Acceptance**: one party makes an offer to another party. The contract is formed by the acceptance of the offer.

2. **Intention** to create legal relations. The parties must intend to form a legally enforceable contract. Parties may form agreements that are not contracts, for instance, a group of friends agrees to go to cinema together, which is obviously not intended to be a contract.

3. **Capacity**: each party must have legal capacity to make the contract. Small children, for instance, are not bound to bargains they made.

4. **Consent** must be genuine (no fraud or duress, mistake or misrepresentation). The declarations of the involved parties must match and clearly express consent.

5. **Consideration** must be present. The basic idea is a "quid pro quo", something in exchange for something else. Gratuitous promises cannot be enforced by law.

6. **Legality**: the object of the contract must not be one of which the law disapproves.

7. **Possibility** of performance. The law does not enforce impossible promises, for instance, to paint the moon blue.

The contract relies on a declaration of consent between the involved parties about a specific performance. The scope of a contract is not confined to specifically recognized transactions. A valid contract does not require compliance with any formalities. Hence, the drafting of contracts is also a process that is not limited to specific methods. Negotiation is a process that serves as the tool for drafting agreements and for expressing consensus about the agreement.

It is important to understand when a contract has been formed. One party makes an offer for a bargain and the other party can make a counter-offer, or may accept the original offer. This iterative process intends to produce a "meeting of minds" about the contract through an iterative exchange of offer and counter-offer. A typical example is bargaining about price by iteratively stating prices until both parties agree upon one price. In English contract law, a party can withdraw an offer if the other party did not accept the offer, yet. After the other party accepted the offer, both parties are bound to the contract. The same applies to multiple parties, only after the acceptance of an offer by all involved parties, the contract has been formed.

Invitation to treat is another term from contract law and stems from the Latin phrase *"invitatio ad offerendum"*. It means to invite an offer. The difference to an offer is that the invitation to treat is the invitation to exchange offers during a negotiation. The invitation to treat can not be directly turned into a contract through acceptance by another party. An example is advertisements that display a price. Such an advertisement can not be directly turned into a contract. Instead, the advertisement is the invitation to exchange offers about the good, which can now lead to a contract. This gives the seller the freedom, not to sell to a specific buyer, for instance, if there are more buyers than goods in stock.

Another special case of an invitation to treat is an *invitation for tenders*. An example here is one party that sells goods and invites other parties to bid on the goods. The sellers are not obligated to choose the highest bid, instead they can choose to accept any offer by the bidders to close the deal and form the contract.

Depending on the circumstances, *Auctions* can be either perceived as an exchange of

offers or as an invitation to treat. The legal interpretation mainly depends on the conditions stated at the start of the auction. For instance, if an auction has no reserve price, all bids are interpreted as offers. Before the auction is closed, any party may retract its bid. The auctioneer accepts the highest offer by closing the auction and thereby forms the contract.

2.1.3 German Contract Law

Civil law is the legal tradition in many countries of the world. In contrast to case law, civil codes are codified laws. Civil law aspires to use legal science to create a comprehensive system of law [ApDe95]. The German civil code is called *Bürgerliches Gesetzbuch* or abbreviated *BGB*, and has been influenced by the Napoleonic code from 1804. The BGB has been the basis for drafting civil codes in many other civil law jurisdictions, such as Portugal, China, Japan, Thailand, South Korea, Taiwan, Greece and the Ukraine. German contract law forms a general law of obligations. This section introduces the German contract law, notwithstanding that similar rules apply to other countries under civil law jurisdiction, for instance, within the European Union [ApDe98]. The following Section states the German legal terms in parenthesis and provides English translations of the terms. This Section relies on the English translation of the BGB available online at [Muss09].

The principles of German contract law are similar to English contract law. In German contract law, there also exists the concept of offer (Angebot) and acceptance (Annahme) to form a contract (Vertrag) see §145, §147 BGB. The contract consists of mutual declarations of intent (übereinstimmende Willenserklärung) of at least two parties. If the declarations are inconsistent, there is no valid contract. An inconsistent declaration of intent is treated as a new offer. German law does not require consideration but infers the intent of the parties for the interpretation of the contract. Every human has the right to draft contracts within the legal bounds of the law. Offers, acceptance and contracts need not follow a particular form, except some special cases, for instance, in employment law. However, German law requires that the *"essentialia negotii"*, the essential aspects of a contract are sufficiently defined. For instance, if someone wants to sell a book, this person must define price, identify the book and name the buyer. The *"accidentialia negotii"* name contract clauses that are inessential to the contract (§154 BGB). There might be no consent about the "accidentialia negotii" and still there is a valid contract (§155 BGB). The law assumes that the involved parties would still have agreed upon the contract, even if this detail is missing. For instance, if there is no consent about the payment details, and the money has been transferred timely and in a reasonable manner, the contract still holds.

One important difference to English contract law is, that offers are binding §145 BGB: "Any person who offers to another to enter into a contract is bound by the offer, unless he has excluded being bound by it.". Because offers are binding, German law regulates the validity period of an offer in §147(2) BGB: "An offer made to a person who is absent may be accepted only until the time when the offeror may expect to receive the answer under ordinary circumstances." Parties may define time constraints for the acceptance of offer §148 BGB. If an offer is accepted too late, the acceptance is considered to be a new offer under §150(2) BGB.
The concept of *"Invitatio ad offerendum"* also exists in German law. Advertisements

and the product listings in an online shop are considered as an invitation to treat. Bids in online auctions are considered as binding offers [Hoer09].

The German law has provisions for dependencies within contracts. This regulation of dependencies is very important for the legal interpretation of bilateral agreements (see Section 5) and multilateral agreements (see Section 6):

- §158 BGB(1): If a legal transaction is entered into subject to a condition precedent, the legal transaction that is subject to the condition comes into effect when the condition is satisfied.
- §158 BGB(2): If a legal transaction is entered into subject to a condition subsequent, the effect of the legal transaction ends when the condition is satisfied; at this moment the previous legal situation is restored.
- §159 BGB: If, under the terms of a legal transaction, the consequences linked to the satisfaction of the condition are to become effective from an earlier time, then when the condition is satisfied the parties are under a duty to render each other the performance that they would have rendered if the consequences had occurred at the earlier time.

2.1.4 Electronic Signatures

The previous sections highlighted how a contract is formed and how it is interpreted under English contract law and German contract law. Parties do not always abide to contracts. The legal system provides means to enforce contracts by bringing them before civil courts. The court rules then if a breach of contract has occurred and the court may decide to award compensation for damages, or impose an injunction to prevent an act or to compel an act.
However, the court can only base its decisions on verifiable claims. The party that demands remedy must be able to proof its claims. This can be difficult for the contracting party. Without being able to proof a contract, a party cannot enforce the contract.
For this reason, real world contracts are often written documents that include handwritten signatures of the contracting parties. Forensic document analysis determines if a document is genuine and it can expose document forgery. There are different types of documents examination, for instance, an analysis of the authenticity of the handwritten signature, an examination of the age of the ink, and also if the same type of paper has been used for all pages. The forensic document analysis is very important to treat documents as proof before the court.

Electronic contracting faces the challenge to give electronic contracts the same legal significance as written contracts. Only electronic contracts that can be verified by the court can be an alternative to written contracts. With the advent of the E-Commerce, lawmakers around the world drafted new laws to give electronic contracts the same validity as written contracts. The European Commission was on the forefront of the endeavor and created the “DIRECTIVE 1999/93/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 13 December 1999 on a Community framework for electronic signatures”. This directive does not define technology but requirements for digital signatures that allows for the creation of electronic contracts, and has been implemented by the member states of the European Union.

The directive distinguished three kinds of electronic signatures. Here is the wording of the implementation of the directive in German law with the "Law Governing Framework Conditions for Electronic Signatures (Signatures Law - SigG)" [Bund01]:

- §2 SigG(1): **Electronic signature** shall be data in electronic form that are attached to other electronic data or logically linked to them and used for authentication
- §2 SigG(2): **Advanced electronic signature** shall be electronic signatures as in (1) above that
  (a) are exclusively assigned to the owner of the signature code
  (b) enable the owner of the signature code to be identified
  (c) are produced with means which the owner of the signature code can keep under his sole control and
  (d) are so linked to the data to which they refer that any subsequent alteration of such data may be detected
- §2 SigG(3): **Qualified electronic signatures** shall be electronic signatures as in (2) above that
  (a) are based on a qualified certificate valid at the time of their creation and
  (b) have been produced with a secure signature-creation device

These three definitions of signatures have different juridical value. The *electronic signature* is a weak electronic signature. This definition is very broad. Putting a name under an email even is an *electronic signature* under this definition. Consequently, *electronic signature* without additional supporting evidence is of no value for a lawsuit.

The *advanced electronic signature* is much better suited as evidence. A document with such a signature is evidence that the document is authentic, that it has been signed on behalf of a unique identity, and it even guarantees the integrity of the electronic document (that is the document has not been modified after applying the signature). *Advanced electronic signatures* are considered as evidence at court. These signatures can be realized through authentication with asymmetric cryptography, for instance, by employing RSA [RiSA78]. *Advanced electronic signature* requires additionally that the signature is linked to an identity. The latter can be achieved through different means.

Qualified electronic signatures define a framework, of how to link an identity to the signature, and require a secure environment for the creation of these signatures. The framework relies on *qualified certificates* that include data about the owner of the certificate, the key for verifying the signature (usually public key), a validity period, the signature of the certification-service provider that issued this certificate, and additional information about the certification-service provider. The law defines a complex infrastructure and legal requirements for issuing *qualified certificates*. The big advantage over the other signature types is that *qualified electronic signatures* are standardized and can be easily used without establishing identities or exchanging keying material between contracting parties. The *secure signature-creation device* assures that no malicious third party (e.g. by using malware) can sign documents in the name of the victim.

Qualified electronic signatures enjoy the highest level of trust for the consideration of evidence at court. They can even be employed where the law requires signatures in written form.

2.1.5 The role of Jurisdiction for International Contracts

Business interactions are increasingly international, and so are contracts. Especially for negotiation via the Internet, cooperations easily span different countries with potentially conflicting legislation. For instance, an offer in Germany is legally binding whilst an offer under English contract law can always be withdrawn, as long the other party did not yet declare acceptance of the offer. It is obvious that in case there is a breach of contract, the jurisdiction is of prime importance for the enforcement of the contract.

A seemingly easy answer to this problem is to define the jurisdiction as part of the contract. Switzerland, for instance, is a popular jurisdiction for transatlantic business to business agreements. However, it depends on many factors if the definition of the jurisdiction is legally sound. For instance in the Europe Union (EU), contracts via the Internet between a consumer and a company assume the jurisdiction of the consumer, for enforcing consumer protection laws (EG Nr. 44/2001). Business to business agreements in the Europe Union are free to define the jurisdiction as part of the contract (§23 EuGVO).

Another problem is to enforce the contract in case one party did not fulfill its promises. Even if one party can sue the other party under a given jurisdiction, it is not clear if the verdict of the court can be enforced [Hoer09]. The legal framework in countries of the Europe Union allows enforcing contracts within the EU. The ability to enforce international contracts outside the EU mainly depends on "Hague Choice of Court Convention" and bilateral agreements under international law.

2.1.6 Requirements for Legally Binding Agreement Negotiations

For the specification and realization of agreement negotiation protocols, the English contract law and the German contract law are equally important. This section can only present ideas and requirements for electronic agreement negotiation protocols. It cannot make any definitive assertions how such agreements will be interpreted by the courts under different legislations.
Nevertheless, one can learn about requirements and constraints that influence protocol design: What are the requirements for an agreement negotiation protocol that works across different jurisdictions? What must be realized as part of the protocol specification and what must be acknowledged by humans that determine the behavior and inputs of the negotiation process?

- The protocol implementation must be able to determine the jurisdiction of a contract. This could be inferred from the participants and context of the negotiation. Another method could be to define the jurisdiction as part of the contract. A party may choose not to participate in a negotiation if it determines that it cannot enforce a contract under the given jurisdiction.
- An offer under German law must be upheld for a reasonable time, in contrast to an offer under English contract law, where one party may withdraw its offer.

The requirement for an agreement protocol is that it must provide mechanisms to withdraw an offer. From this follows an additional requirement that the protocol implementation must be able to determine if the withdrawal of an offer is permissible under the given jurisdiction.

- The protocol must lead to agreements that express the consent of the parties about the contract. The protocol should either arrive at a state where consent is reached, or the parties recognize that there is dissent.
- The protocol should allow one or more parties in the negotiation to make offers and to declare the acceptance of the agreement.
- The protocol should explicitly state when an agreement has been reached.
- The contract must contain an agreement that unambiguously defines the consent that has been reached.
- It is highly desirable that the negotiation protocol produces agreements that are considered evidence under the jurisdiction of the contract. Therefore, the protocol should allow for the integration with legally binding digital signatures.
- The protocol implementation should verify the compliance of the agreement with the law. The implementation should verify the intention to form a legal relation, the capacity of each party to make a contract, the legality (potentially under the jurisdiction of each party), and the possibility of performance. Under jurisdictions following English contract law, additionally consideration must be present in the agreement for each party.
- For multilateral agreements, the negotiation should produce an atomic outcome. Either all involved parties in an agreement are legally bound, or no party is bound by the agreement.

These are just the requirements to align the negotiation protocol with the legal frameworks of many countries of the world. Of course other requirements, such as business requirements, are also very important for the design of an agreement negotiation protocol.

2.2 The Economic Perspective on Negotiation

The market economy is underlying principles of almost all economic systems in the world. A popular definition of the term market economy is [Altv99]: “A market economy is economy based on the power of division of labor in which the prices of goods and services are determined in a free price system set by supply and demand”. The good a supplier sells on the market satisfies the demand of a customer.
Negotiation plays a crucial role in the market economy for establishing terms and prices for the exchange of goods and services. People negotiate because they seek to do something together what they cannot do alone. During a negotiation a party usually gives something away to receive something else in return. Successful negotiations in market environments are a compromise between two conflicting objectives of a party: the desire to maximize what a party gains through a negotiation and to minimize what a party has to give away.

2.2.1 Utility Function

Many economic theories rely on the concept of *utility* to model the preferences of an agent. They assume that one can map the preferences of an agent to a utility function that yields numerical values. Let $X = \{C_1, C_2, \ldots, C_n\}$ be a consumption set of distinct goods C_i. The utility function $u : X \rightarrow \mathbb{R}$ ranks different consumptions sets. Most utility functions in literature are monotonic, continuous, and quasi-concave. If there are two consumption sets $\{x_1, x_2\}$ and $\{y_1, y_2\}$ where $u(x_1, x_2) > u(y_1, y_2)$, the consumer strictly prefers $\{x_1, x_2\}$ to $\{y_1, y_2\}$. If $u(x_1, x_2) = u(y_1, y_2)$ the agent is indifferent to the two consumption sets.
The concept of a utility function is very popular for modeling preferences because it allows for a comparison between different consumption bundles. If there are conflicting goals where only some goals can be reached at the expense of other goals, the utility function models this trade-off. Economic theories often simply assume that a utility function exists and do not define how the utility function is implemented and thereby avoid the complexity of analyzing preferences together with the available information.

For example, a person has different utilities for food. This person has a utility of 100 for one unit of pizza, 60 for one unit of noodle soup and 25 for one unit of salad. If we assume a utility function $u_f(X) = \sum_{d \in X} u_f(d)$ where the overall utility of food X is the sum of the utilities of each dish d, this person would prefer two units of noodle soup over one unit of pizza.
This example highlights that overly simplistic utility functions do not map to real world behavior. One person can neither live solely on noodle soup, neither on pizza. Much complexity of defining utility function stems from the fact that an action to choose one good, changes the outcome of the utility function for choosing the same good again in the future.

The objective of an agent is to maximize its utility. Under this assumption an agent always chooses actions in such a way that it maximizes its utility. One method for calculating the expected utility EU for an action a is by using the following function:

$$EU(a) = \sum_{s'} P(s'|s, a) \cdot u(s') \tag{2.1}$$

In this function $P(s'|s, a)$ is the probability of reaching the state s' from the state s by performing action a in s. This function calculates the utility of each possible resulting state with the probability of reaching this state and summing up the terms.
Let A be the set of all possible actions. The agent can maximize its utility by performing the action a^m where $max_{a \in A}(EU(a)) = EU(a^m)$

2.2.2 Price Formation

The price of a good or service is the single most determining factor for making a purchasing decision and is the incentive for the seller to offer its good. A large body of economic research addresses price formation.

The *efficient market hypothesis* [Bach00] states that in financial markets that are informationally efficient, no information or analysis can be expected to result in outperformance of an appropriate benchmark. The market as a whole is always right even if the

individual participants of the market overreact or underreact to new information. All information that becomes available will directly reflect in the expectations of the price. Prices on an efficient financial market are always in equilibrium because all available information is reflected in the total capital market pricing of the stock. Agents acting in efficient markets have rational expectations.

In reality, financial markets are neither perfectly efficient nor completely inefficient. The paradox of efficient markets is, that if an investor believes that market is efficient it will refrain from doing an own analysis of the securities and the stock price. If the investor does not make the analysis, the market is not efficient anymore.
Another problem is that the efficient market hypothesis assumes that information is available to all market participants on an equal basis. Insider trading is a prominent example where information asymmetries lead to markets that are not efficient.

2.3 Research Communities dealing with Electronic Negotiation

The term *negotiation* in computer science is used in very different scenarios. It is not obvious what role negotiation has in different research communities. One can use the google scholar search[1] to judge the amount research that involves negotiation. By using the search terms shown in Figure 2.1, scholar gives the number of papers and one can identify the first arguably relevant paper. The Figure 2.1 puts the number of papers in a research area in comparison with the number of papers in this area that mention the use of negotiation.

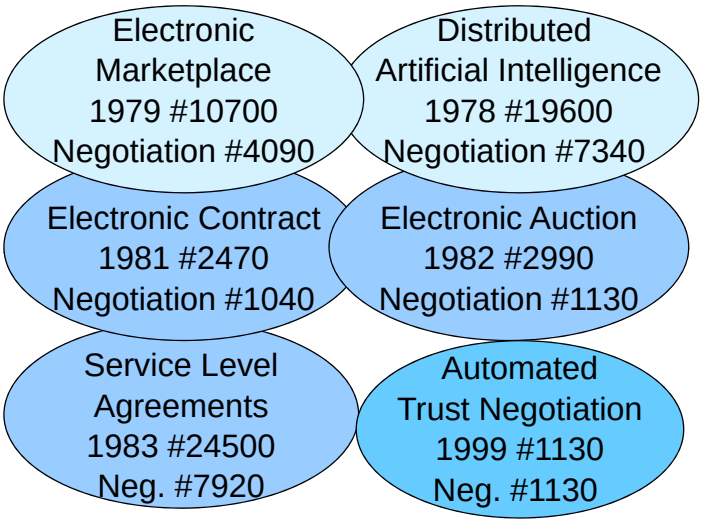

Figure 2.1: *Research Communities addressing Electronic Negotiation Year of First Arguably Relevant Publication, Number of Papers with Search Term, Number of Papers with Negotiation*

The idea of an "Electronic Marketplace" dates back to the 70s and was one of the first big research topics where negotiation played was a crucial component. This research community realized the great potential impact that the emerging communication networks,

[1] http://scholar.google.com

such as the world wide Internet had for business interactions. According to [Laco00] *"electronic commerce (also e-commerce) is generally understood to span the whole range of business situations that are at least partially supported by a communication network such as the Internet."* Of special interest was the introduction of computer support for all different stages of business activities, such as signing contracts, transferring funds, executing contracts, and the integration with business processes in a legally sound way. What blurred the focus of this research area was that many technologies such as digital signatures, secure communication, the World Wide Web [BLCG92] were still in their infancy. Hence, much effort was put into the evolution and adaptation of these generic technologies and sometimes the focus on the research problems that are unique for electronic commerce was lost.

The "Distributed Artificial Intelligence" (DAI) community also started early and produced many results on distributed negotiation in the course of the years. The research on Artificial Intelligence deals with single agents that can exhibit intelligent behavior. Distributed Artificial Intelligence specializes on the distributed nature of agents in networks and deals with coordinated, concurrent action and problem solving [BoGa88]. Research on "Distributed Problem Solving" divides and shares knowledge within a group of agents to develop a solution. "Multi-Agent Systems" coordinate the actions of distributed agents to reach a common goal or individual interdependent goals. Distributed Artificial Intelligence developed a large number of communication methods to coordinate distributed agents. These communication primitives give a large freedom for the implementation of artificial intelligence methods, but most of these approaches are not suitable as agreement negotiation protocol.

"Electronic Auctions" have a tremendous success in the Internet. eBay[2] alone had a gross merchandise volume of $59.7 billion in 2008. Business to business auctions are nowadays an established tool for electronic trade. Transporeon[3], a transport logistic auction platform, had a gross merchandise volume of €8.5 billion in 2009. Most of these auctions are only partially automated; the buyers must interpret the description of the item for sale and decide about their maximum bid. The underlying communication protocols for auctions are now in everyday use. Recent research mainly focuses on automated intelligent bidding strategies. A fundamental limitation of auctions is that the bidders are willing to take the good as it is and cannot negotiate about the terms and conditions under which the good is sold. The terms and conditions are set forth before the auction starts. For instance, during a car auction, a bidder cannot state that she makes the bid only under the condition that she receives an extended warranty. These restrictions exclude auctions from the more complex scenarios described in this thesis.

During the 80s work on "Service Level Agreements" and "Electronic Contracts" gained momentum. The research on Service Level Agreements [Verm99, TGRS+04] deals with the definition of interdependent terms and conditions for service delivery and support. Price and quantities are just some properties amongst many others, which can be defined by Service Level Agreements (SLA). An SLA can be used to describe the services to be delivered, performance guarantees, priorities, responsibilities, and warranties. One main objective of this research direction is to make SLAs machine processable to automatically monitor and enforce SLA compliance. SLA languages [KeLu03] are expressive

[2] http://www.ebay.com
[3] http://www.transporeon.de

and can cover many real world scenarios. WSLA [KeLu03] is a specification for Service Level Agreements in Web Service environments. RBSLA [Pasc05] is a declarative SLA language that allows for rule based service level management. However, most research on SLA deals with the processing of already established SLAs and does not address how SLAs are formed.

Since the late 90s, the research on *Trust Negotiation (TN)* [WiLi02] [SmSJ04] [BeFS04] deals with automatically establishing mutual trust between strangers by an iterative credential exchange. "Trust Negotiation" systems use a policy driven iterative negotiation process to reach an agreement between two parties that need not have a prior trust relationship. The main focus is on the protection of sensitive information (credentials and policies) and the definition of policy languages for the negotiation process.
Amongst the discussed research areas in this section, Trust Negotiation is the domain that has the strongest focus on iterative negotiation protocols. That is one reason why many concepts and ideas in this thesis have roots in Trust Negotiation.

Start out with an ideal
and end up with a deal.

Karl Albrecht

3. Generic Agreement Formation Model

This chapter will first introduce different negotiation types before it introduces a generic model for agreement formation.

3.1 Types of Negotiation

Distributive and integrative negotiations are the two most important types of negotiation processes [fHum05]. The *distributive* negotiation is about making concessions until an agreement has been reached or the involved parties decide that no agreement is possible. An example is a seller of a car that negotiates about the price. If a seller lowers the price, the buyer has the gain that she can buy the car cheaper. The gain of the buyer comes at the expense of the seller. This kind of negotiation is also referred to as win-lose negotiation. Distributive negotiations are very common for price negotiation about standardized commodities.
The weakness of distributive negotiations is that price is only one of many factors. Business relationships are becoming more and more complex because market transactions usually consist of more than the exchange of goods and money. Suppliers do not only deliver a good but also provide support for a defined period. Supply chains must react flexible to changes and still operate reliably.

Integrative negotiations find solutions by forming cooperations where price is only one factor. Both sides try to increase the value of the agreement by searching for mutually profitable alternatives through a negotiation about multiple issues simultaneously. The parties in integrative negotiations are ready to search actively for agreement options that are more attractive to the involved parties then the starting point of the negotiation. Integrative negotiations go beyond the win-lose characteristics of purely distributive negotiations by discovering alternative agreement options that increase the utility of the involved parties and increase the likeliness of an agreement. One can extend the example from above. The seller of the car that is not willing to lower the price any further. Instead, the seller includes winter tires at the current price, to make the deal more attractive.

Negotiation is not only about multiple quantifiable issues but also about the specification of the terms of an agreement. These terms define what good or service has to be delivered upon conclusion of the contract. It depends on these terms if the agreement

makes sense to the negotiating parties. For instance, for buying tires, not only the price is important, but also that the tires are compatible with the car. Discovering and matching these requirements with available products can be very complex and time consuming for agreement negotiation. The *integrative negotiation of terms and conditions* often involves a trade-off between different options with varying performance characteristics. An example of an *integrative negotiation of terms and conditions* from the technical world are handshake-protocols in network protocols where only an agreement about the session parameters allows for the establishment of the communication session.

Another important factor is the motivation of the negotiating agents [DFJN97]. *Self-interested* agents want to maximize their own local utility through negotiations. This behavior leads to competition about scarce resources where the overall utility may suffer. For instance, the *free rider problem* in game theory [Rabi93] describes how individual agents increase their local utility by not contributing a fair share to the cost of a public resource. The result could be that the public good may not be available in the future because in a population of rational self-interested agents, no agent would contribute to the cost of the public resource. Self-interested agents neglect the utility of the system as a whole.
Altruistic agents in contrast rate the benefit of the system higher then their individual benefit. This behavior is common in computer systems that belong to one organization where the utility of the whole system is of prime importance. Negotiation between altruistic agents is primarily for the coordination and for forming cooperations. Hence, negotiations are mainly integrative negotiations about terms and conditions.
A group of agents belonging to one organization might be altruistic within the group and the whole group may act self-interested towards agents belonging to other organizations.

An ideal negotiation protocol should work for all negotiators independent of the intrinsic motivation (self-interested versus altruistic) and support distributive negotiations as well as integrative negotiation of terms and conditions.

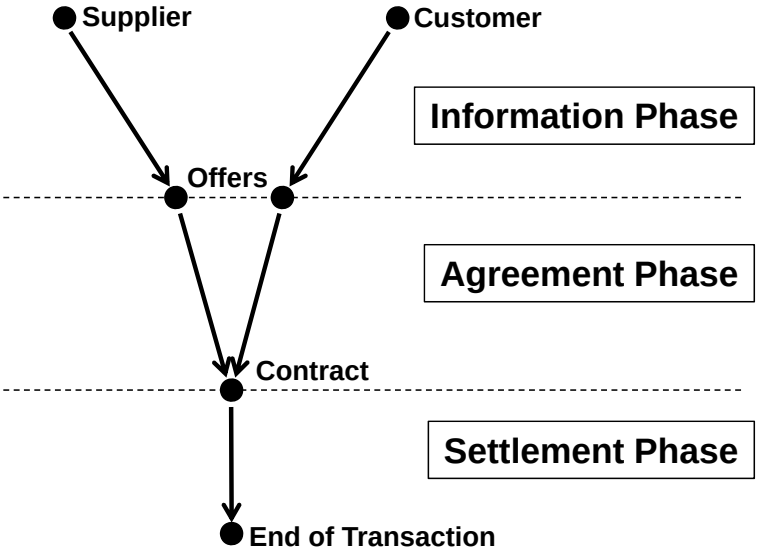

Figure 3.1: *The three phases of Market Transactions*

3.2 Three Phases of Market Transactions

In a market, everything connected to the exchange of goods and services is part of a so called *market transaction*. According to Schmid and Lindemann [ScLi98], market transactions can be divided into three phases as shown in Figure 3.1:

1. **Information phase**: During the *information phase*, the participants gain an overview of the market and identify potential trading partners. The *information phase* allows suppliers and customers to define own needs and potential offers. They discover their market environment that includes suppliers, competitors and potential customers. Especially the collaboration with suppliers introduces many additional factors that go beyond price, for instance, available technologies, interoperability, compatibility of business plans, liquidity and reliability of partners.

2. **Agreement phase**: When the market partners begin an iterative information exchange with the objective of carrying out a market transaction, they are in the *agreement phase*. During this phase, the partners try to establish consensus about all parameters necessary for defining the market transaction, for instance, quantities, price, quality and delivery dates. The offers made during the negotiation might be binding, for instance, during auctions. In most negotiations, the parties have the choice to abort the negotiation during the agreement phase if they reach the conclusion that the agreement under the current terms is not beneficial. Negotiation is the primary tool for reaching agreements during this phase. The agreement is usually a contract and can be reached through bargaining, auctions or complex iterative negotiations. After an agreement about the market transaction has been reached, the parties enter the *settlement phase*.

3. **Settlement phase**: The partners must now fulfill their contracts. The contractors deliver what they promised and monitor if they receive what has been promised to them. Depending on the agreement, entering this phase triggers complex business processes that can contain anything from procurement, manufacturing, to logistics. The parties control if the obligations made towards them are fulfilled. If some obligations are not fulfilled, the party might decide to enforce the contract through court and demand remedy. After a successful completion of this phase the contract is fulfilled.

In contrast to detailed negotiation models in literature [Gull79], the three phase model of market transactions is generic. It does not make any assumptions about the behavior and strategies of the participants during the market transaction. The model even applies to more than two parties.
Auctions can serve as an example of how to apply the three phase market transaction model. During the information phase, the seller identifies an auctioneer and arranges that a certain good should be sold. A number of parties discover the auction for this good and decide to bid. The auction itself can be perceived as the agreement phase where the parties announce their bids. When the auctioneer awards the good to the highest bidder, the settlement phase starts. The settlement phase includes the exchange of money and the transfer of ownership of the good. If one party thinks that the other party did not adhere to its part of the deal, it might decide to enforce the result of the auction by law.

3.3 Generic Negotiation Model for Electronic Negotiations

An agreement can be more than a contract; it can be the means to establish knowledge and to coordinate actions. Agreement formation also happens for the coordination between altruistic agents. For instance, services can coordinate themselves with the help of agreement negotiation. Agreement negotiation can be a tool for exchanging knowledge between different agents and for establishing coherent knowledge about the subject at hand. The three phase model 3.2 of market transactions is not restricted to e-commerce as originally proposed by Schmid and Lindemann [ScLi98]. With small extensions, the three phase model can be applied to agreement negotiation in general.

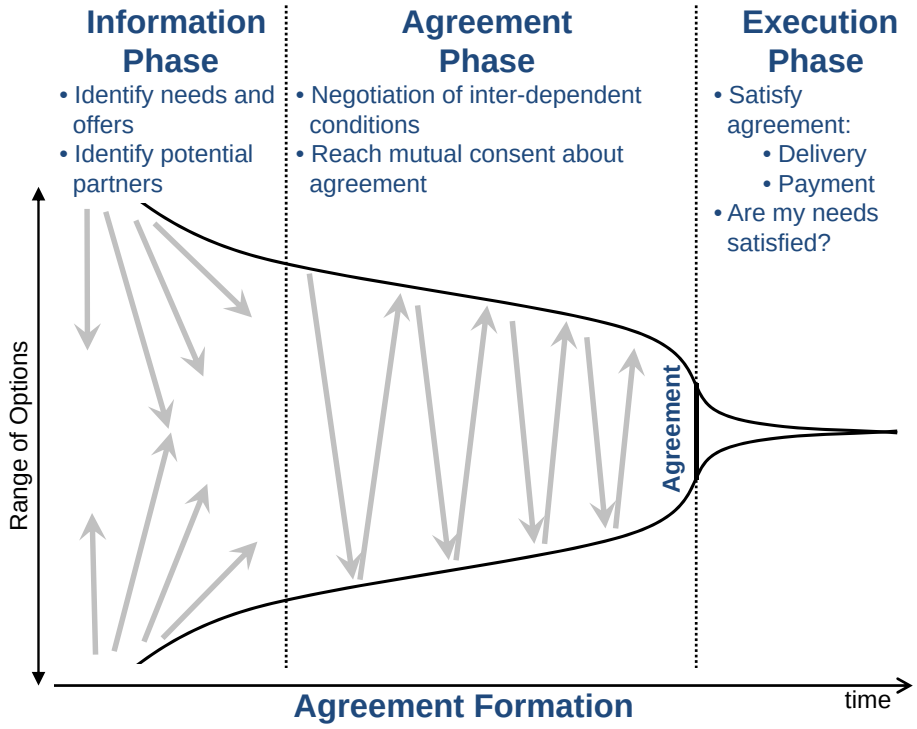

Figure 3.2: *Generalized Three Phase Model of Agreement Formation*

Figure 3.2 shows the more generic agreement formation model. The first phase is again the *information phase*. Two parties start an agreement formation process to achieve together what they cannot do alone. Each party has its objectives and knows what it can offer. The parties search for partners to reach an agreement. Typically, the positions of the parties, of what they want, are still far from each other. During this process, the parties evaluate possible options to form an agreement and already exclude positions, for instance, in the case that no partner has the capability to work together for this particular option. The parties set the agenda for a possible negotiation and define their goals.

After the parties decide to evaluate prospects to form an agreement with a number of potential partners, they enter the *agreement phase*. Negotiation during the agreement phase narrows the differences between the parties. The parties announce what they want and what they offer. The negotiation process continues until they reach consent and thereby establish an agreement, or if at least one party decides to abort the negotiation.

The options for reaching the goals of the involved parties are clearly defined by the agreement. The parties still have some freedom of how to interpret and implement the

agreement. The *execution phase* is where the parties act based upon the agreement. They deliver now what they promised and they monitor that they receive what has been promised to them. If the agreement does not work out as intended, different strategies are possible. In the case of self-interested agents where the agreement is a legally binding contract, this contract may be enforced by law. Altruistic agents might try to uphold as much of the agreement as possible, or they might renegotiate to adapt to the new situation. After no one has any more claims from the agreement, the execution is finished. The positions of the parties finally converge when the execution ends and the agreement has turned into reality.

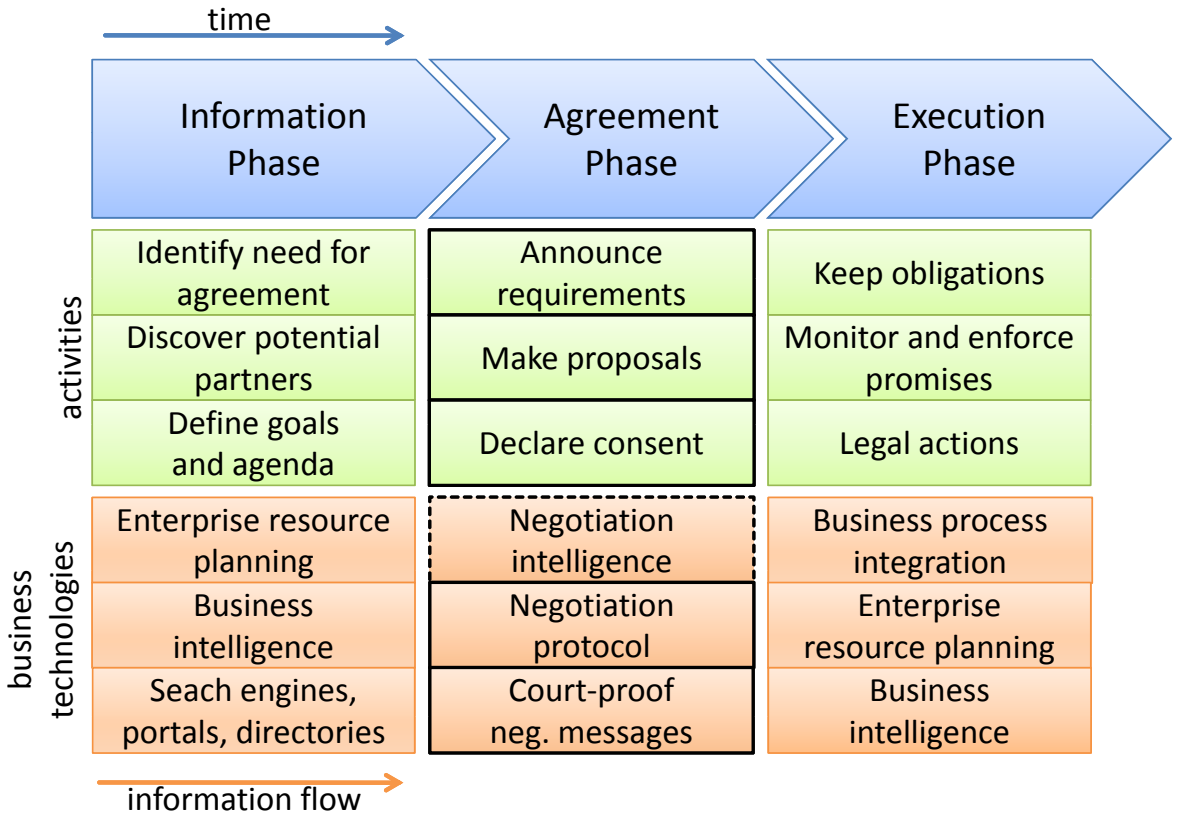

Figure 3.3: *Components of Agreement Formation Model*

The technological environment differs, depending on the motivation of the involved systems. The Figure 3.3 shows the environment for business services which usually have a self-interested motivation when they engage in a negotiation with other organizations. On the other hand, altruistic systems are very common in computer networks. Negotiations are carried out to achieve interoperability and to establish cooperations. The next two sections give a non-exhaustive overview how negotiation works together with core technologies in these environments.

3.3.1 Business Environment

In a business environment, negotiation is an important function to engage with other organizations. Negotiation happens within the scope of complex business processes. *Enterprise resource planning* (ERP) systems control many aspects, such as, finance, production, supply chain management and logistics. Negotiation can serve as a tool for an ERP system to interact with systems outside of its business entity. The ERP

system defines the goals and the agenda for the negotiation. For instance, when an ERP system receives an order, it determines which parts it needs to fulfill this request. The ERP system may discover that it needs to order these parts from suppliers. Companies often know their potential trading partners very well. To discover additional partners to negotiate with, *search engines*, electronic *portals*, and business *directories* can be used. *Business intelligence* gathers knowledge about possible partners, customers and suppliers. It feeds this knowledge into the negotiation. The business intelligence is important to decide which partners to negotiate with, and to predict the reliability of the partner to keep the obligations of a potential agreement.

There are three important technological domains for negotiation in business environments. The *negotiation protocol* (also called interaction protocol) is how the systems communicate during the agreement negotiation. The capabilities of this protocol limit what a negotiation can be about. There exists a large body of research [KeNT00] and standards for auctions protocols [FIPA02]. Auction protocols are however distributive and are by nature limited to very specific negotiations, for instance, about price and quantities. The *negotiation intelligence* that defines negotiation tactics and strategies to obtain the best utility, is often an integral part of research on auction protocols. Integrative negotiation protocols are more powerful, as they allow complex statements about the agreement. Due to the higher complexity, work on integrative negotiation separates the negotiation intelligence from the negotiation protocol. Another important aspect in the business environment is that agreements must be enforceable by law. The messaging of the negotiation protocol has to produce court-proof evidence of the outcome of a negotiation. Work on electronic signatures (see Section 2.1.4) and fair contract signing [PaGä99, GaMa99] deals with these issues.

After an agreement has been concluded, the agreement enters the execution phase and will proceed within the business process. The agreement is stored persistently and will be interpreted by the ERP systems. The behavior of the parties during the negotiation and the compliance of the parties with the agreement in the execution phase will enter the knowledge base of the business intelligence. This knowledge allows for the adaptation of the strategy for future negotiations.

3.3.2 Cooperation of Altruistic Services

Most network interactions between services can be characterized as altruistic. These services negotiate to ensure the operation of the distributed system itself. They are usually only self-interested in the sense that they want to preserve their individual operability. An important motivation is to coordinate services that have different specializations and to gather distributed knowledge. Depending on the environment of the services, advanced negotiation protocols are employed.
Autonomic systems [GaCo03] rely on negotiation as an important tool to realize self-organization and self-management of distributed services. Altruistic *multi agent systems* employ strategies to coordinate distributed agents and to maximize the overall system utility. Service oriented architectures rely on negotiation to achieve interoperability in heterogeneous environments [VOHH+07] and to orchestrate service interactions. Even Internet protocols rely on basic negotiation mechanisms during the protocol handshake to establish the communication parameters [DiRe06].

In these environments, negotiation protocols must provide the functionality for the use case at hand and produce agreements efficiently. Negotiation intelligence is usually

much simpler. Policy based systems are popular for steering negotiations [VOHH+07]. Preferences define the goals of the system and simple matching strategies are employed for the selection [Holl09]. Security of the negotiation is equally important as with business environments because the functioning of the systems can be severely impaired, if an attacker manages to manipulate the communication. Negotiations do usually not have to produce court-proof message trails.

After the negotiation terminates either with an agreement or with a failure, the systems take the information from the agreement to adapt their behavior to this new knowledge.

Make every bargain clear and plain,
that none may afterwards complain.

Greek Proverb

4. XML Policy Intersection for Autonomic Security Adaptation

Distributed computer systems have different capabilities, for instance, with regard to protocol versions and available security algorithms. The computer systems must agree on one set of compatible options to work together. Handshake protocols are used to negotiate one set of session parameters from a range of possible options. The handshake happens usually when a connection shall be established. The involved parties must agree on one set of common parameters to achieve interoperability.
Policy intersection is a simple, yet fundamental technique for consensus building during protocol handshakes. The policy intersection yields a set of all mutually compatible parameters for the given policies.
The chapter will first introduce the fundamentals of policy intersection protocols. The second part deals with a concrete use case for a policy intersection protocol for the autonomic configuration of secure communication. The use case will show how different services and components interact during all phases of the generic agreement formation model (see Figure 4.1) to reach an autonomic configuration of security protocols and security parameters.

4.1 Contributions of Chapter on XML Policy Intersection

This chapter introduces a policy intersection protocol to demonstrate the capabilities and limitations of set intersection for agreement negotiation. Additional this chapter helps to differentiate the basic agreement policy intersection negotiation methods from the more powerful iterative agreement negotiation, introduced in Section 5 and Section 6.

1. Introduction to policy intersection as an agreement negotiation primitive. Discussion of capabilities and limitations of policy intersection.

2. Comparison of single-round protocol for policy intersection with iterative policy intersection.

3. Practical use of policy intersection protocol to achieve autonomic cross-layer configuration of secure communication channels. The ESAF framework demonstrates

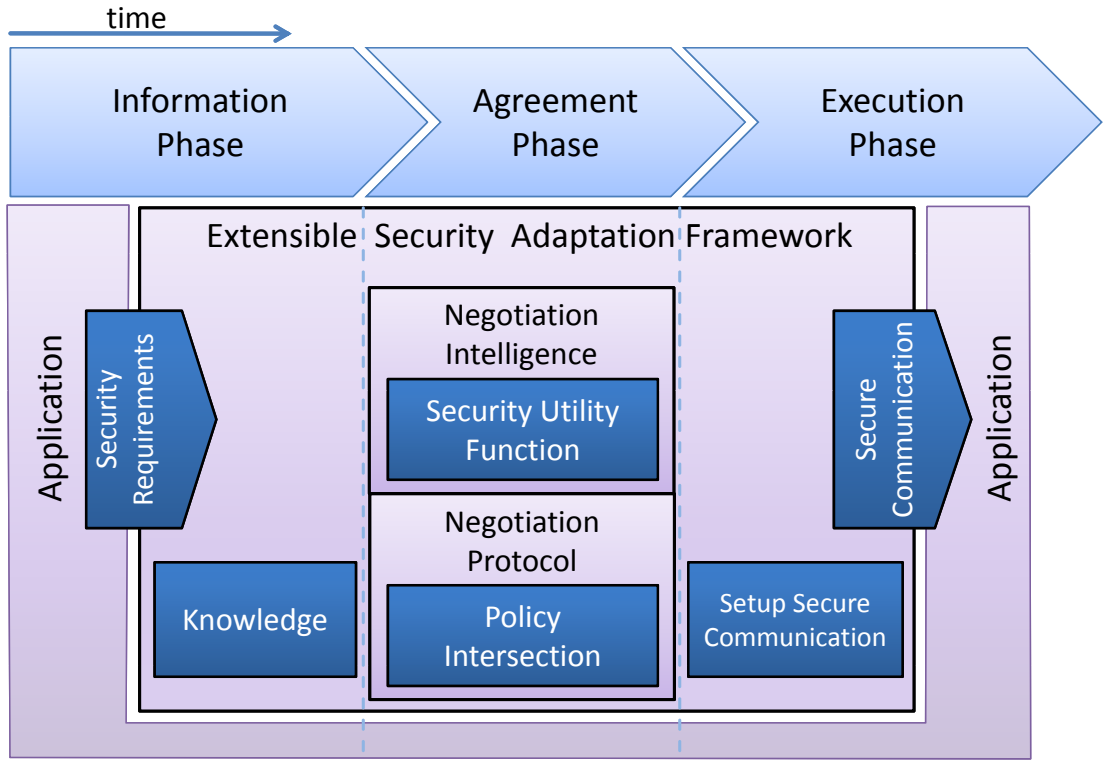

Figure 4.1: *Bilateral Policy Intersection Protocol for Autonomic Security Adaptation*

a system that works fully automated through agreement negotiation and highlights how the activities in all three phases of the generic agreement formation model 3 relate.

4.2 Related Work

Policy intersection is the underlying mechanism of the WS-Policy standard [VOHH+07]. The WS-Policy standard introduces XML policy intersection for Web Services. The standard defines how to define policies and how to perform policy intersection. WS-Policy is intended to identify compatible parameters that are common to all policies in the intersection. WS-Policy does not define how to join complementary policies. For instance, complementary policies that define a configuration at different levels of abstractions must first be joined to one comprehensive policy, before they can be used for policy intersection during a protocol handshake (see Section 4.4.3). Kolovski and Parsia [KoPa06] argue how to use the Web Ontology Language (OWL) for semantic reasoning about policy intersection. The drawback is that the results of semantic reasoning are not always intuitive for an administrator that defines the policies.

Hollunder [Holl09] introduces a semantic extension to domain-specific semantic policy intersection with WS-Policy. The Section 4.4.3) describes how to deal with the semantics of security policies for the autonomic configuration of secure communication channels.

The authors in [CCZY09] established a formal algebra for the combination of access control policies. The basic constructs are also used in this section for defining the policy intersection algorithms. This paper does not specify how to use policy intersection for protocol handshakes.

4.3 Policy Intersection

Policy intersection is a popular technique for combining different local policies into one single policy. It allows different stakeholders in the system to all take control of an application through individual policies.
Policy intersection protocols go one step further and establish a joint policy with a remote party that is agreeable to all parties.
This Section explains the fundamentals of XML policy intersection and introduces how policy intersection protocols work.

4.3.1 Operators for Combining Policies

Policy intersection is arguably the most popular method for handshake protocols to establish knowledge of compatible parameters during session setup. Intersection of two sets A and B in set theory denotes that all elements that belong to A and B at the same time, are part of the result set: $A \cap B := \{x \in A \text{ and } x \in B\}$.

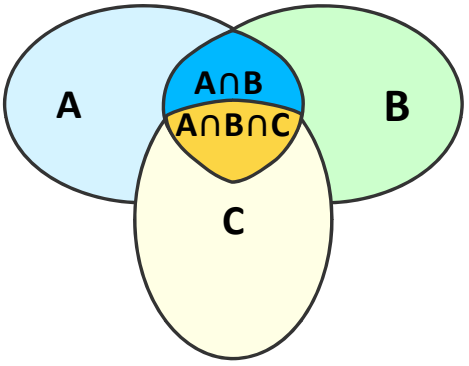

Figure 4.2: *Policy Intersection Between Three Parties*

Policy intersection is analogous with set intersection but it does not imply the semantics of formal set intersection. Policy intersection applies intersection to policies that describe the available options. Compatible parameters for two parties are elements that are present in both policies. The policy intersection yields a set of all mutually compatible parameters. The WS-Policy [VOHH+07] standard relies on policy intersection to establish compatible parameters between Web Services.
Policy intersection is commutative and associative. Joining three policies A, B, C (see Figure 4.2) can happen in arbitrary order and will yield the same result: $(A \cap B) \cap C = A \cap (B \cap C)$.

Sometimes policies are merged. Let us assume two policies A and B that should be merged. Elements can come from policy A, or from policy B, or the element can be present in both policies. The analogous operation in set theory is the *union* of two sets: $A \cup B = \{x : x \in A \text{ or } x \in B\}$.
Another important operation is *join*, which is also used extensively in relational databases. A join operation is used to combine two policies by applying a logical *expression*. This expression determines if a combination of elements (one element from each policy) should become part of the result set: $A \bowtie_{expression} B := \{a \cup b | a \in A \wedge b \in B \wedge expression\}$.
The *left outer join* operation computes the join operation of two policies and adds the values from the left policy set that did not match the join expression.

4.3.2 XML Policy Intersection

The Extensible Markup Language (XML) [BPSME+04] is a popular language for policies. It allows for an extensible definition of structured data that can be processed via standardized APIs, for instance, via DOM and SAX compliant parsers. XML is a format for storing and processing data and is the foundation of many Web Service communication protocols.

XML is attractive for policy intersection. XML provides large degrees of freedom to define human readable data formats. XML Schema validation can be used to ensure that policies are in accordance with the policy specifications. Policies can directly be edited by administrators and are at the same time machine processable text.
Different implementations of an application type might lead slight variations of the policies themselves. Policy intersection algorithms might exclude logically equivalent options just because they have slightly varying representations. The XML canonicalization [Boye01] standard can address differences in the representation of XML. Two XML documents that have the same canonical form, after applying XML canonicalization, are logically equivalent. XML canonicalization is an important step to bring logically equivalent XML entities into the same representation. It facilitates policy intersection even for documents that differ in their XML representations.

A simple recursive algorithm that performs XML policy intersection is given in Listing 4.1. It takes a copy of the local policy `target` that will become the result by eliminating entities that do not exist in the other policy `compareTo`. The algorithm starts at the root nodes of both documents.
The `policyIntersection` function iterates over all children of the two nodes it has been invoked with. For each child of the `target` node, it will try to find an equivalent node in `compareTo`. If it cannot find this node, it will remove this node from the target document. The presented algorithm considers different element orders of the same elements as equivalent. For each node that can be found as a child of `compareTo`, the algorithm will call itself for this child. This recursive algorithm eliminates all elements in the target document, which cannot be found in the document to compare to.
When are two XML elements equivalent? This example relies on the following implementation of `compareNodes`. The qualified name is an identifier that is subject to namespace interpretation. Two XML elements are equivalent if the qualified names match, all attributes match and if the elements are text nodes, the text value matches. Figure 4.3 is an example that shows the result of a policy intersection with the described algorithm.

```
void policyIntersection (Node* target, Node* compareTo) {
  NodeList targetList = target->get_children();
  NodeList compareList = compareTo->get_children();

  // iterate over nodes in targetList
  for (target=targetList.begin();target!=targetList.end();target++){
    matchFound = false;

    // iterate over nodes in compareList
    for (comp=compareList.begin();comp!=compareList.end();comp++) {

      // compare selected nodes in targetList and compareList
      if (compareNodes(*target, *comp)) {
```

```
            // nodes are equal, continue recursion for children
            policyIntersection(*target, *comp);

            matchFound = true;
            break;
          }
        }

        // no matching node was found in compareList
        // remove this node (and all children) from targetList
        if (!matchFound) {
          target->get_parent()->remove_child(target);
        }
      }
    }
```

Listing 4.1: *XML Policy Intersection Algorithm*

```
<AAA>
  <BBB>
  </BBB>
  <EEE>
    <FFF>
    </FFF>
  </EEE>
  <GGG>
    <HHH>
    </HHH>
  </GGG>
</AAA>
```

```
<AAA>
  <EEE>
    <FFF>
    </FFF>
  </EEE>
  <BBB>
    <CCC>
    </CCC>
  </BBB>
  <HHH>
  </HHH>
</AAA>
```

```
<AAA>
  <BBB>
  </BBB>
  <EEE>
    <FFF>
    </FFF>
  </EEE>
</AAA>
```

Figure 4.3: *Policy Intersection that tolerates arbitrary order:*
a) Policy of Party A (left) b) Policy of Party B (middle)
c) Result of Policy Intersection (right)

Depending on the application, other methods for combining policies are also reasonable, for instance, one policy intersection algorithm might enforce a strict element order, another algorithm might only accept one element if all its children are also present in the other document.

A Left Outer Join can be used for combining complementary policies. Logical expression that defines under what conditions elements can become part of the solution, for instance, to combine two policies based on equal element names (see Example 4.4). This Left Outer Join can be useful if one policy contains a high level description and the other policy contains detailed parameters that can enrich the high level descriptions with additional details.

4.3.3 Single-Round Policy Intersection for Protocol Handshakes

This behavior is useful because it allows different policies to be present on one system, where only the intersection of all these policies is a permissible parameter set. For instance, the computer wide policy restricts the set of possible parameters to *Host A* and the application specific policy has different constraints and only allows for *Host B*.

```
<AAA>
  <BBB>
  </BBB>
</AAA>
<DDD>
</DDD>
```

```
<BBB>
  <CCC>
  </CCC>
</BBB>
<EEE>
</EEE>
```

```
<AAA>
  <BBB>
    <CCC>
    </CCC>
  </BBB>
</AAA>
<DDD>
</DDD>
```

Figure 4.4: *Left Outer Join: a) Policy of Party A (left) b) Policy of Party B (middle) c) Result of Left Outer Join of the two policies (right)*

When a remote computer performs a connection handshake, $A \cap B \cap C$ yields the possible parameters. This behavior guarantees that only parameters that are permissible for all three policies can be chosen.
The associative policy intersection can also be used to identify one possible parameter set for a group of parties. This application is less common for protocol handshakes.

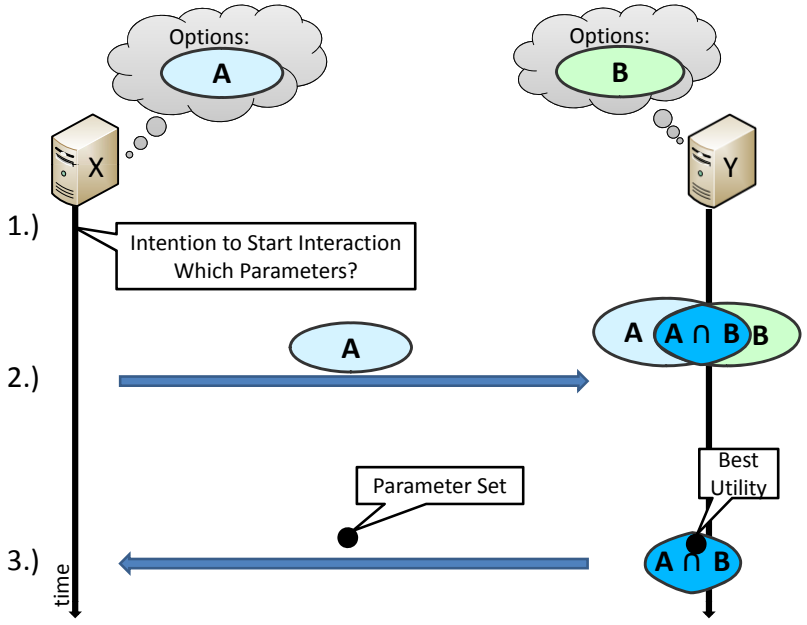

Figure 4.5: *Handshake Protocol with Policy Intersection*

The Figure 4.5 shows exemplarily how to agree upon one parameter set with a single-round handshake protocol through policy intersection.

1. An application triggers the session establishment with another system. It triggers the protocol handshake to establish one agreed upon set of parameters.

2. The party X transmits its available parameters to a party Y which performs the policy intersection locally, then. In most implementations, party Y selects one parameter set from the intersection. Local configuration policies and simple *utility functions* (see Section 2.2.1) determine how this party chooses an option.

3. The chosen parameter set will be transmitted to the remote party X and the session can be established.

4.3.4 Comparing Iterative Policy Intersection with single-round Policy Intersection Protocols

Many protocols rely on an *iterative policy intersection* approach. This strategy allows for an iterative refinement, starting with high level information (for instance, compatible protocol versions) and ending up with detailed configuration parameters for the session.
Iterative policy intersection protocols are easier to implement because most protocol messages contain the pure data values in a predefined format and the implementer must not bother about syntactic and semantic definitions of data fields of single-round policy intersection.

A single-round policy intersection protocol discloses all information from one party at once. The iterative policy intersection saves bandwidth, because it only transmits relevant information at each step. During the iterative refinement, the protocol avoids sending information that is irrelevant, because it has already been excluded through a choice at an earlier stage in the protocol.
This behavior introduces an interesting trade-off between a single-round policy intersection protocol and an iterative policy intersection protocol. How long do both protocols need for negotiations and what are the dominant factors?

Many factors contribute to the time that policy intersection protocols consume (see Figure 4.6 and Figure 4.7). This section models the behavior of both protocol variants to determine the trade-off between the number of messages for iterative policy intersection and the message size of single-round policy intersection. The model only wants to give rough estimates and makes some simplifying assumptions.

The model relies on three basic parameters. The serialization delay σ is the time to serialize the message m_i for transmission over the network. The serialization delay σ can be calculated from the bandwidth of the link and size of the data to be sent: $sigma = packetsize(bits)/transmissionrate(bps)$. The bandwidth is usually constrained by the access network of client and server and usually not by the bandwidth of the core network.
The network delay φ is influenced by factors such as queuing and forwarding at routers, packet loss, effects of multi-path routing and most important, and the distance of the link. The distance of the link determines the propagation delay at roughly $5\mu s$ per km. For this model we assume φ to be equal for all messages. The time for a queued message to arrive at its destination is therefore $\sigma + \varphi$.
The processing time α is the time for processing the message by the protocol implementation and the application, after it has been fully received by the host.

The iterative policy intersection protocol transmits a number of small messages that contain few data. This model assumes that the messages are roughly the same size and that the serialization delay σ_1 is equal for all small messages. The processing time α_1 can also be expected to be small for these small messages. The overall performance of

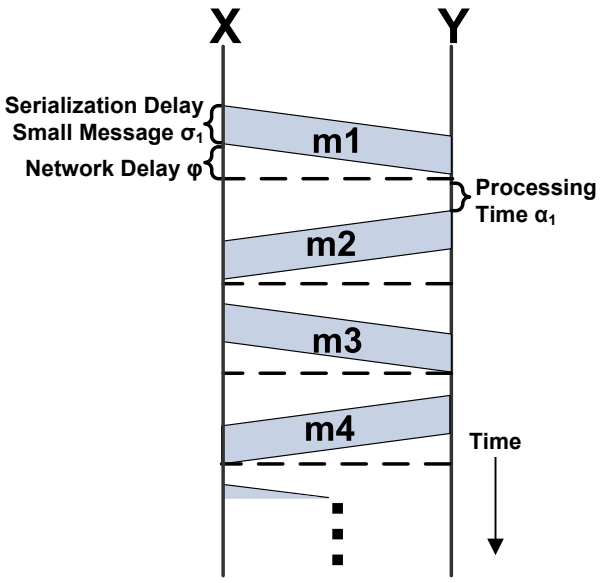

Figure 4.6: *Handshake Protocol with Iterative Policy Intersection*

iterative policy intersection with n messages can be approximated as:

$$T_{iterative} \approx n \cdot (\alpha_1 + \sigma_1 + \varphi) \tag{4.1}$$

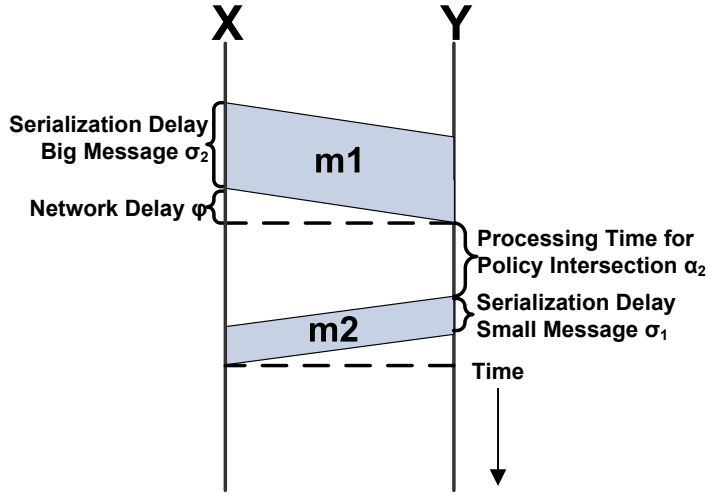

Figure 4.7: *Handshake Protocol with single-round Policy Intersection*

In the single-round policy intersection, party X sends one comparable big message $m1$ to Y. Serialization takes a longer time σ_2 and the processing is also more time consuming with α_2. The response message, however, is comparable small. This model assumes that the response message that already contains the chosen parameter set, can be roughly processed at α_1. The overall performance of the single-round policy intersection protocol can be approximated as:

$$T_{complete} \approx \alpha_1 + \sigma_1 + \alpha_2 + \sigma_2 + 2 \cdot \varphi \tag{4.2}$$

These models tend to underestimate the total delay, because the model neither considers packet loss and fragmentation, nor the effects of multi path routing which all contribute to the delay variance in real networks. Refer to [MFRa02] for an analysis on the behavior of TCP over large distance links in the face of packet loss.

The models can be used to calculate how the two protocol modes compete for a given scenario. For instance, imagine the following scenario: The uplink is the bottleneck for subscriber lines and determines the serialization delay σ. Nowadays a bandwidth of 1 Mbps is the uplink speed of many ADSL connections. Let us assume 100bytes for the message sizes of the small messages and 10kbytes for the single message that contains the whole options for party X. The number of offer-answer messages for the iterative negotiation could be 10 messages in total for this imaginary scenario. The network delay does not depend on the packet size. In the following text, the one-way network delay φ is assumed to be 10 milliseconds, if not stated otherwise. The processing time α_1 for small messages is assumed to be 10ms and a conservative estimation of α_2 for policy intersection with large messages is 100ms.

Network Delay	single-round Policy Intersection	Iterative Policy Intersection
0.010	0.13	0.20
0.025	0.16	0.35
0.050	0.21	0.60
0.100	0.31	1.10
0.200	0.51	2.10

Table 4.1: *Impact of one-way Network Delay (all values in seconds)*

The Table in 4.1 shows estimates for this scenario. The network delays in this table reflect typical round-trip times one can determine with the ping tool by sending "Echo Request" and "Echo Reply" packets with the Internet Control Message Protocol (ICMP). The delays in the table are 10ms for optimal conditions within Germany 25ms for accessing systems in Europe and 100ms for accessing systems in other areas in the world. Besides very small one-way network delays of 10ms, the iterative approach is significantly slower than the approach with the single-round protocol. These results were to be expected, as the iterative protocol is sensitive to network delay due to the large number of messages.
An additional analysis of this scenario showed that the available bandwidth of the access network does not make a big difference for the performance of any of the two protocols. The messages are comparable small and the impact of serialization is negligible in comparison with the other factors.

The results showed that the iterative protocol can hardly compete with 10 messages. How many messages can the iterative protocol use, to stay below the time for the single-round policy intersection protocol?
The Figure 4.8 shows a plot for varying message numbers of the iterative policy intersection protocol. This plot assumes a scenario where all parameters except the message number are as stated above. The parameters of the single-round policy intersection protocol were constant.

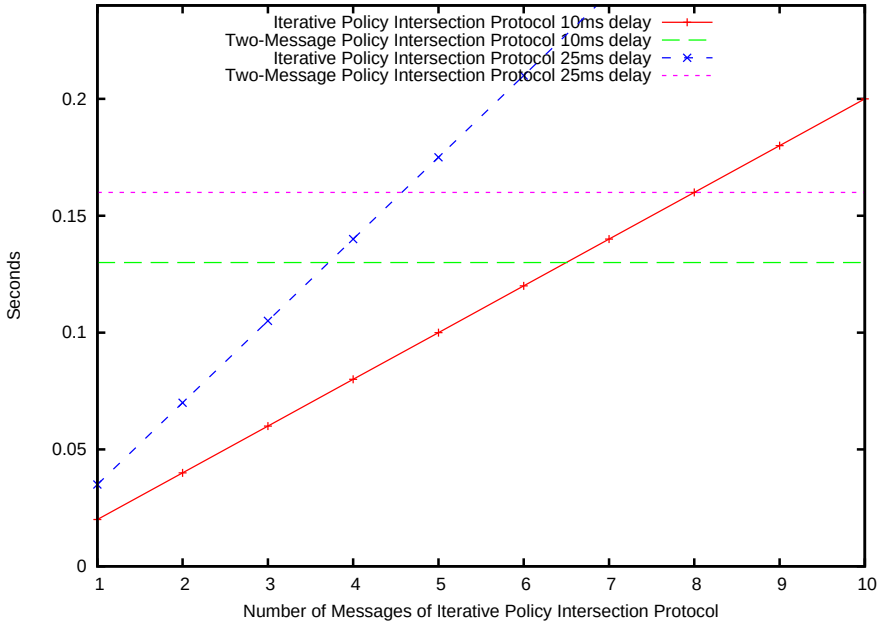

Figure 4.8: *Time for Protocol Runs depending on Number of Messages for Iterative Policy Intersection Protocol and on Network Delay σ*

The plot shows that for small message numbers, and for a low delay of $\sigma = 10ms$, the iterative policy intersection protocol performs better for up to to three offers and respective answers. A TLS handshake with mutual authentication, for instance, can be performed with five messages. The plot for a network delay $\sigma = 25ms$ shows that for a handshake protocol with five messages, single-round policy intersection performs slightly better.

In the ideal case, both protocol variants are comparable in performance for this scenario. However, negative effects that impact delay worsen the situation for iterative policy intersection protocols. Delay can increase in the face of bad network conditions. Packet loss increases the delay due to packet retransmissions. Overloaded servers might need a long time to process messages. Iterative policy intersection protocols worsen the situation under these circumstances because they are sensitive to delay. One may even argue that iterative policy intersection protocols increase the load at servers because the server must perform context switches for small processing steps and thereby introduces a comparable big overhead in the server and in the operating system.

The numbers show that the single-round policy intersection performs better in difficult network conditions with high delays. Consequently, this chapter will focus on a single-round policy intersection protocol.

4.3.5 Decision Intelligence

Policy intersection typically works with policies that contain different alternatives for performing the same task. The policy intersection process leads to mutually acceptable

policy options. The problem remains how to choose one option amongst a number of viable policy options.
Utility functions (see Section 2.2.1) are a popular means to choose policy options automatically. Refer for the use of utility functions in autonomic systems to [TeKe04].
Utility functions can be used to assign a utility for each policy option. One policy option is a subset of the policy. It contains all required parameters to perform the objectives of the application. The application logic or XML Schema definitions can define the constraints of viable policy options. The application extracts the set of all policy options. Each policy option set P_O consists of parameters e. There are many different imaginable implementations for utility functions. One simple utility function can be defined as follows:

$$u_{policyOption}(P_O) = \sum_{e \in P_O} u_{parameter}(e) \tag{4.3}$$

The function $u_{policyOption}$ takes one policy option P_O and aggregates all individual utilities $u_{parameter}$ of each parameter e.
This utility function is very simple and easy to implement as it only requires a utility function to map a given parameter to one utility. This left-total relationship is very easy to implement and can be realized as a lookup table where the different parameter values map to exactly one utility.
An adequate policy option can be chosen automatically by selecting the policy option that has maximum utility.

4.3.6 Merits and Drawbacks of Policy Intersection

Configuration policies are the means of an administrator to define the behavior of software. Policy intersection is highly popular for reaching agreements over session parameters that are in accordance with the local policies.
Policy intersection is straight forward to implement. The logic for discovering mutual acceptable subsets is easy, particularly for iterative policy intersection where only few values have to be compared. The implementations have a low complexity and are efficient.
By performing handshake protocols over secure channels, the risk introduced by policy intersection is low. Each party can test if the session parameters S are in accordance with the local policy *Host A* by verifying that $S = A \cap S$. In other words, only permissible options can be chosen, unauthorized options can be discovered easily.

Policy intersection with single-rounds reveals all possible parameters of the party that initiates the handshake. Iterative policy intersection protocols do not reveal all possible parameters. However, this behavior is no real security gain, because an adversary can perform repeated handshakes and probe all options systematically.
As the previous Section 4.3.4 already showed, iterative policy intersection can have better performance under ideal network conditions. However, bad network conditions are reinforced by the back and forth of small messages. The conclusion for this research is to rely on single-round policy intersection to make the protocol more robust to bad network conditions.

The biggest drawback of policy intersection is that it unable to discover new or unforeseen solutions. No option outside of the policies can become part of a solution,

even if this additional option would be acceptable to all parties. This restriction excludes policy intersection from many use cases that have requirements that go beyond performing a session handshake, for instance, contract negotiation or trust negotiation. See Section 5 and Section 6 for iterative agreement negotiation protocols that do not have these limitations.

4.4 Policy Intersection as a Handshake Protocol to establish Secure Communication Channels

A natural use of policy intersection protocols is for the session handshake during connection establishment. This Section demonstrates how *policy intersection* can be applied for building consensus about security protocols and configuration parameters to establish a secure end-to-end communication channel. Policy controlled configuration of secure communication channels is advantageous because it decouples applications from the underlying cryptographic protocol. This leads to a clear separation of the requirements for secure communication and the technical configuration details for the establishment of a secure communication channels. Consequently, cryptographic protocols for secure communication become interchangeable, even across different protocol layers. Policy intersection is the method to establish an agreement about the cryptographic protocol to use and the required configuration parameters. The policy controlled security use case, discussed in the following sections, demonstrates how to realize all relevant activities of the generic agreement formation model, see Figure 4.1.

4.4.1 Approach to Autonomic Configuration of Secure Communication

Current security technologies are either integrated into the system (like IPSec) or into the application (like SSL/TLS).
An example for security configuration at system level is the management of security policies or associations for IPSec. The configuration, especially of IPSec, is awkward. It is up to the administrator to configure the installation of IPSec as well as to provide device-specific security policies. IPSec uses a system wide configuration to store security policies in a security policy database (SPD). These policies determine if and how to secure a particular packet. The mayor drawback of a system wide security configuration is that applications cannot adapt the security and cryptographic algorithms to their requirements. On resource constrained devices, for instance, an application might rely on a weaker protection to have enough free resources for video streaming.

In contrast to the system-wide configuration, security at application-level must be supported already at the time of implementation. Applications which deal with the configuration of security protocols on their own, are therefore forced to support the particularities of the protocol interfaces. Such applications break if old protocols become unavailable even if new security protocols are introduced into the system as a replacement. At the time of the development unforeseen security configurations cannot be used.
If an application is closely coupled with a security policy, the administrator can only influence the behavior through application specific configuration options and cannot enforce the security policies at a single point in the system.

Policy controlled security separates the security requirements from the complex configuration of cryptographic protocols. The different stakeholders in the system formulate *security requirement policies* and do not have to bother how to configure cryptographic protocols that satisfy the security requirements.

The novel ***E**xtensible **S**ecurity **A**daptation **F**ramework (ESAF)* [KMNG05, KNMC06, Mase05] takes *security requirement policies*, derives an acceptable policy for all stakeholders and negotiates the parameters for secure communication with a remote host. *Policy intersection* is used as a tool to give different stakeholders in the system full control of the security parameters to establish secure communication channels. Policy intersection is used locally to take security requirements from different sources as an input and generate one *security requirements policy* as an output that is acceptable to all local stakeholders. The *policy intersection protocol* establishes compatible security options between the local host and the remote host. The ESAF automatically chooses one configuration option with the help of a *security utility function* and establishes the secure communication channel.

ESAF is designed to make configuration more flexible and to avoid protocol-dependent application development. Among its features are the seamless integration of new protocols, exchangeability of corrupt protocols and utilization of the best protocol available for communication. This choice is based upon security policies specified by the administration, the user, and the applications. The security support is end-to-end and layer-independent. High-level policies provide a fine-grained support for defining requirement levels. Requirements are therefore not binary, but scalar. Policy intersection is the primary means for handling these different requirements and for reaching consensus about connection parameters.

4.4.2 Related Work

Levine was the first to discuss the effects of Quality of Security Service [IrLa02] in depth and to describe how to adapt the system to defined requirements. Her system offers security choices to reflect the different behavior of the mechanisms. The decision is derived from security ranges and performance ranges specified in form of policies. The core argument behind her reasoning is: If a user decides on a minimum security level for an application, would she ever agree to more security if that increases her costs?
The Celestial Project[FILX00] was developed at the North Carolina State University and is an application-aware and protocol-oriented security policy management system. The core component is the Security Management Agent(SMA), which performs the security protocols configuration and exchanges configuration messages with other Celestial nodes for connection setup. SMNAs reside at middleboxes on the data path and influence the connection setup by adding their own security policies to the message. The system does not support an end-to-end negotiation about security parameters.
Yarvis proposed the Conductor framework[(MPG00] for coordinated distributed adaptation inside the networks. Conductor intercepts the communication and redirects it to the framework. Agents inside the network adapt the data streams according to a plan made by the Conductor framework at the data source. This approach requires an intelligent agent at the data source to make decisions. Policy intersection, in contrast, is easier to implement because it clearly defines how to identify compatible policy options and how to apply a utility function to make an automatic choice.
The Generic Security Service Application Program Interface(GSS-API) [Linn97] is standardized by the IETF and aims at providing a generic interface to use end-to-end security, independent of the security mechanism and the communication protocol. The

GSS uses tokens for the establishment of the security context and the protection of the communication. It does not support single-round policy intersection.
Ganz introduced the security broker[GANZ98] architecture for WLANs. This framework chooses security services upon security requirements specified by the user, available network performance and the performance of security routines. There exist three security classes to choose from, each with a different level of protection and different performance. The security broker does not allow to negotiate about security parameters with a remote party.
Neither of the approaches presented above uses a single-round policy intersection protocol. One approach that does use single-round policy intersection is the WS-Security Policy standard [VOHH+09]. However, this standard is limited to the security parameters of Web Services and does not address security at transport layer and network layer. Another restrictions is that the standard cannot deal with security policies at different abstraction levels. For instance, the ESAF use case describes how to combine an abstract *security requirement policy* with a concrete *system capabilities policy* to identify the available policy options for a session.

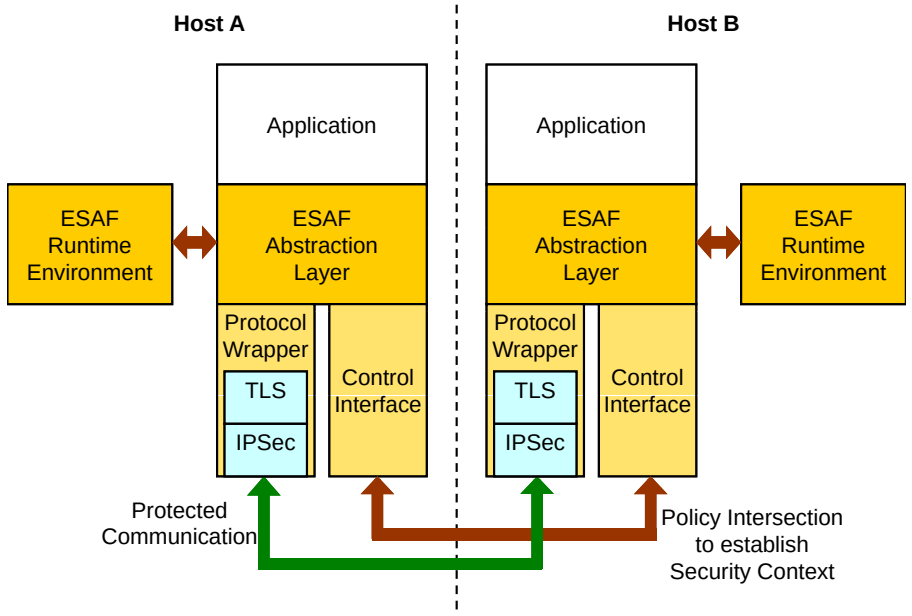

Figure 4.9: *Extensible Security Adaptation Framework*

4.4.3 Extensible Security Adaptation Framework

ESAF assists the applications with the establishment of secure connections. A security context negotiation is performed during connection setup to determine the requirements of the communication partners. The configuration offer will be sent and the intersection with the configuration offer of the server leads to a list of supported and required protocols for the connection. A choice can be made then, what the "best" protocol for this session will be.

The basic idea of ESAF is to use policies to formulate security requirements for communication channels and to define the parameters for the communication setup. There are two policies. One to define what a stakeholder wants and another policy to define what the system is capable of:

1. *Security requirement policies*: These policies describe in an abstract form the required security and communication parameters.

2. *System capabilities policies* contain the available cryptographic communication primitives on a host and define which security requirements can be satisfied by a protocol.

The framework generates a number of policies during the single-round policy intersection:

1. *Configuration offer* contains available secure communication options of the host that wants to establish a secure communication. The configuration offer is derived from the *system capabilities policy*, but only contains configuration options that satisfy the security requirements.

2. *Session policy*: One host that receives a *configuration offer* makes a choice from a number of alternative configuration options. The resulting parameters are defined in the *session policy*. The*session policy* can be used directly by ESAF to establish the secure communication channel.

This type of policy controlled security requires an abstraction of the protocols to make them exchangeable. Abstraction of communication protocols offers a compelling approach to solve two problems at once. It allows applications to use secure communication protocols at different layers without any change to the application itself and it makes it easy to introduce new protocols to the system.

4.4.3.1 Security Context Negotiation

When a connection has to be established, it is necessary to perform a *security context negotiation*. The security context negotiation is a single-round policy intersection protocol. The participants identify a set of mutual acceptable protocol configuration options.

Figure 4.10 shows the sequence of the negotiation.
In step 1.) *Host A* first determines the requirements by forming a union of all applicable *security requirement policies*. The result set after the union contains the most demanding security requirements from all policies. ESAF then joins the *security requirement policy* with the *system capabilities policy* to find all possible configuration options on this system. Thereby it establishes a set of protocols and configurations that meet the requirements. Only the entries which possess a security rating of equal or better then the minimum requirement, specified by the *security requirement policy*, will be included and form a special policy, the *configuration offer*.
The generated *configuration offer* is now sent to *Host B* within the connection request in step 1). Optionally, the security requirement policy can also be included with the configuration offer to inform *Host B* about the preferences for *Host A*.

In step 2.) *Host B* processes the received *configuration offer*. First, it must evaluate its own *security requirement policies* and join them with its *system capabilities policy* to get the available *local configuration options*.
Then, the algorithm starts to determine the adequate configuration by performing a

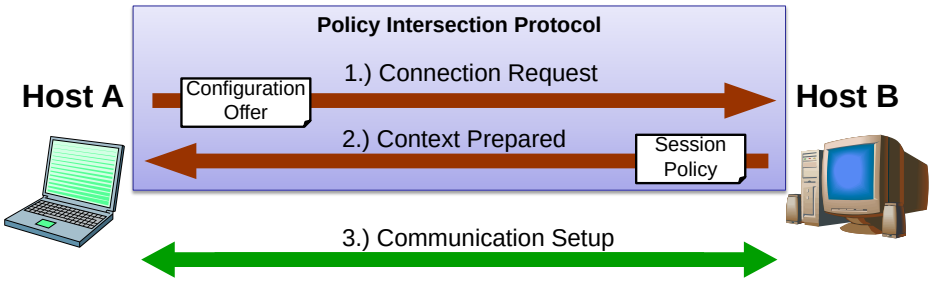

Figure 4.10: *Security Context Negotiation Sequence*

policy intersection of the *configuration offer* with the *local configuration options* to obtain a policy that contains all mutually acceptable configuration options.
Host B can now use its utility function to choose one configuration option. After one configuration has been selected, the connection is prepared and a *context prepared message* is sent to *Host A* in step 2), containing the *Session Policy*.
In step 3.) *Host A* receives the *session policy*. The ESAF middleware on *Host A* verifies that the selected configuration is consistent with the request it sent previously in step 1). ESAF can now establish the secure connection with the exchanged configuration information.

Caching of the locally available configuration options can speed up the negotiation process. For hosts which must serve many similar requests, it can save resources to implement a caching scheme at this stage.
In case the policy intersection of *Host B* does not yield any mutually acceptable configuration option, it will send an *Agreement Failed* message back to *Host A* and attach its own security requirement policy and system capabilities policy. In the failure case, *Host A* can try to adapt its policies to find a workable solution to communicate with *Host B*. This modification of the security requirements requires human approval because it could lead to degradation of the security level.

4.4.3.2 ESAF Policies

This Section describes the different XML policies of the ESAF framework (see Figure 4.11). The public communication interface of ESAF accepts *security requirement policies* whereas internally a *system capabilities policy* is used to describe the available options with the installed protocols.

4.4.3.3 Security Requirement Policies

The *security requirement policies* help the stakeholders to express their requirements and preferences. It is important that *security requirement policies* are truly protocol and configuration independent and describe the full range of requirements in a general manner. For such a policy language it is important to identify a set of language constructs and keywords that are able to express the full range of communication requirements.
The security of a communication link is usually judged based on the degree it provides the following characteristics: *authentication*, *integrity*, *confidentiality* and *non-repudiation*. Two more parameters of high importance for secure communication are

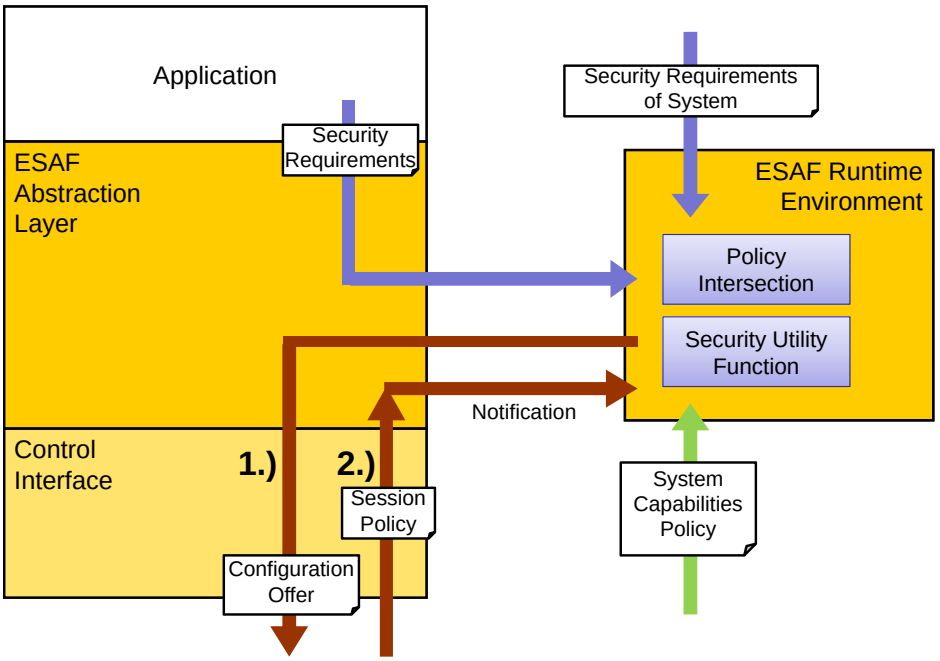

Figure 4.11: *Intersection of Policies from different Stakeholders within ESAF*

reliability and *performance*.

Security protocols are not equally optimized for all identified parameters. The level of security varies depending on key lengths and utilized encryption algorithms. The performance of the algorithm may also be an important factor, imagine a resource constrained device like a handheld computer. This trade-off between security and performance is also termed Quality of Protection(QoP) [IrLa02]. Each service requirement has a scalar value attached, to express the importance of the parameter on a scale between 0 and 10. The value 0 would mean "no importance" while 10 would give the parameter the highest priority. To differentiate even further we introduced the notion of *minimum* as a knock out barrier and *ideal* as the desired configuration value.

Security requirement policies, which reside at the same system, can be joined by ESAF. The application specifies an application dependent *security requirement policy* as well as the administrator can specify a system policy. These policies can easily be unified because they refer to the same system capabilities policy. The union algorithm takes the higher (=more secure) scalar value for each matching element.

```
<?xml version="1.0" encoding="UTF-8"?>
<security_requirement_policy>
  <security_requirements>
    <message_authentication>
      <minimum>5</minimum>
      <ideal>7</ideal>
    </message_authentication>
    <data_integrity>
      <minimum>7</minimum>
      <ideal>10</ideal>
    </data_integrity>
    <confidentiality>
```

```
    ...
    </confidentiality>
    <non_repudiation>
    ...
    </non_repudiation>
    ...
    ...
  </security_requirements>

  <communication_requirements>
    <connection_type>connection-oriented</connection_type>
    < reliability >reliable</ reliability >
    <sequencing>yes</sequencing>
    <error_control>yes</error_control>
    <performance>
      <minimum>5</minimum>
      <ideal>10</ideal>
    </performance>
  </communication_requirements>

</security_requirement_policy>
```

Listing 4.2: *Security Requirement Policy*

4.4.3.4 Protocol Descriptions Policy

All messages exchanged during a policy intersection contain detailed protocol descriptions. The security requirements are subjective in nature, and consequently, are only used locally. The XML format for the *protocol descriptions policy* is shown in Listing 4.3. It contains all details necessary to configure a secure connection for the protocol alternatives in the message.
The *system capabilities policy*, the *configuration offer*, and the *session policy* are all instrances of the *protocol descriptions policy* format.
The basic elements of this format are the security protocols and the communication protocols. These elements contain the descriptions what a protocol can do and how it must be configured. If critical exploits in a security protocol are disclosed, the administrator can easily disable the corresponding entries in the *system capabilities policy* or degrade the security level. This will allow that the applications to use more secure protocols for their connections. The user will not even notice the change and the application does not have to bother.

```
<?xml version="1.0" encoding="UTF-8"?>
<protocol_descriptions>
  <supported_security_protocols>
    <!-- alternative security protocols are defined here -->

    <security_protocol id="ipsec">
      <supported_security_services>
        <confidentiality>
          <encr_algorithm id="aes128-cbc">
            <key_length>128</key_length>
            <security_level>9</security_level>
            <performance>6</performance>
          </encr_algorithm>
```

```
            <encr_algorithm id="aes256-cbc">
              <key_length>256</key_length>
              <security_level>10</security_level>
              <performance>4</performance>
            </encr_algorithm>
            <encr_algorithm id="twofish128-cbc">
              ...
            </encr_algorithm>
            ...
          </confidentiality>
          <message_authentication>
            ...
          </message_authentication>
          <non_repudiation>
            ...
          </non_repudiation>
          ...

        </supported_security_services>
        <required_communication_protocols>
          <!--reference transport protocols which can be used with this security protocol -->
        </required_communication_protocols>
      </security_protocol>

      <security_protocol id="TLS">
        ...
      </security_protocol>
      ...
    </supported_security_protocols>


    <supported_communication_protocols>
      <transport_layer>
        ...
      </transport_layer>
      <network_layer>
        ...
      </network_layer>
    </supported_communication_protocols>

</protocol_descriptions>
```

Listing 4.3: *Excerpt of Protocol Descriptions Policy*

The *system capabilities policy* describes in detail the possible configuration options for each security protocol and a system local security rating. These elements correlate directly with the elements of the *security requirement policy*. It is now possible to determine all possible encryption algorithms in the system which can provide a certain security service, for example, confidentiality.

4.4.3.5 Security Utility Function with trade-off for Quality of Security Services

The security utility function of ESAF follows the Quality of Security Service [IrLa02] approach. The security utility function can be defined different on each host. The security utility function presented in this Section was devised to make automated decisions between different possible parameter sets and to make a trade-off between performance and security requirements.

Without knowledge about the performance of different security algorithms and the respective protocols, one cannot make this trade-off. We performed extensive experiments [NiKC06] to determine the runtime performance of a broad range of security algorithms.

Algorithm	Confidentiality	Performance
AES-128	8	8
AES-192	9	7
DES	2	4
3DES	8	2
Blowfish-128	9	6
Blowfish-192	10	6
Algorithm	**Authentication**	**Performance**
HMAC-MD5	2	9
HMAC-SHA-1	5	6
HMAC-SHA-2-256	8	3

Table 4.2: *Possible values for ESAF performance ratings*

ESAF security utility function requires a rating of the available security services on a host. These ratings are host local. Varying these ratings is an option to tune security algorithms for resource constrained devices. It is important to keep in mind that such ratings are subjective in nature and can only represent some usage scenarios. Hence ESAF considers only locally available ratings to determine which protocols might be acceptable. There is no exchange of the ratings during the negotiation of the communication context for a connection. Instead, the rating helps to determine which protocol configuration options fulfill the security requirements for a given connection to form *configuration offer* and *session policy*.
The results from the measurement study [IrLa02] are useful to derive the performance levels of encryption and authentication services as shown in Table 4.2. The other scalar value expresses a subjective rating about the assumed security for each algorithm. These values are part of the *protocol descriptions policy* with the *security_level* and the *performance* entities.

Let P be a *protocol descriptions policy*, $S \in P$ is a security protocol, and $a \in S$ is one property of the security protocol. Let R be the *security requirement policy*. The function $\sigma(a)$ returns for the protocol property a the host local performance rating. Function $\beta(r, S)$ retrieves the security rating of property r in S or in R. The utility u of one security protocol S can be calculated as follows:

$$u(S,R) = \sum_{r \in R} \begin{cases} \sigma(r) & if \beta(r,S) \geq \beta(r,R) \text{ //compliant} \\ -1000 \cdot (\beta(r,R) - \beta(r,S)) + \sigma(r) & if \beta(r,S) < \beta(r,R) \text{ //incompliant} \end{cases} \tag{4.4}$$

The function $u(S, R)$ rates each security protocol. Protocols that fully satisfy the security requirement have a positive value. Applying $u(S, R)$ to each $S \in P$ yields an ordered list of ratings, where bigger values indicate a larger degree of compliance with the security and performance requirements. Notice here that this utility function al-

ways gives precedence to the fastest available security algorithm, if this algorithm can provide the required security level.
Let us consider an example. An application wants to establish a communication link with a set of security requirements. One requirement is that the minimal authentication security level is at least 4. The HMAC-SHA-1 would be chosen in our example, because its security rating is sufficient and it is faster than HMAC-SHA-2-256. If the HMAC-MD5 would possess a better security rating, say 4, it would be selected due to the better performance value.
An alternative definition of the security utility function could give precedence to the most secure algorithm that satisfies the performance requirements in R and that satisfies all security requirements.

4.4.4 Summary of Use Case

The use case demonstrated how to implement policy intersection during all phases of the generic agreement formation model. ESAF covers most interactions and interfaces that are required for integrating policy intersection protocols with real world applications.
ESAF allows for the autonomic configuration of end-to-end communication links. This Section described the concept of requirement driven security configuration and how abstraction of the underlying mechanisms is achieved. Policy intersection is the enabling technology for the policy-driven autonomic security control.
The systems can act autonomously during communication largely because of the strict separation between requirements and configuration. Applications can leverage the abstraction to introduce new communication mechanisms and for hassle free utilization of well proven security services. The separation between requirements and particular configuration rules grants the flexibility to choose the "best" configuration option at a time. The selection depends on the requirements specified in form of policies and capabilities of the communication partners. The context of the link is defined during the negotiation phase. The first priority is fulfillment of security requirements; quality of service aspects are considered next. An ESAF application is unaware of the utilized mechanisms. Hence applications remain operable whilst new services are introduced into the system and deprecated ones are removed.

4.5 Discussion of Policy Intersection

Policy intersection protocols have a number of desirable properties. They are easy to implement and allow for complete automation of the negotiation process.

Such protocols have a low risk for the involved parties. It is easy to assess the risk of the local policies. The own policy specifications contain every possible solution of the negotiation process. That makes it easy to analyze the impact of different policy options.
The negotiation is single-round and produces a parameter set that is acceptable to both parties. Each party can easily verify that this parameter set is part of the exchanged policies and can reassure that the chosen parameter set is compatible with the local policy.
Bilateral policy intersection protocols can be secured by employing standardized protocol for secure end-to-end communication. Typical security for bilateral negotiation protocols protects against the malicious third parties. End-to-end security guarantees

the integrity and authenticity of received messages for the applications.

Policy intersection is powerful because it not only serves as a consensus building method for parameter sets, but because it also allows for a local intersection of different policies. The local policy intersection allows that different stakeholders formulate policies to cover different aspects of the organizational objectives. For instance, a system administrator could formulate a policy that describes the available security capabilities of an individual system whereas the maintainer of the application writes a policy that is geared towards high performance of the application, as demonstrated with *ESAF* in Section 4.4. Policy intersection of both policies on the local system leads to one policy that serves as an input to the negotiation.

Policy intersection protocols can be automated because they allow for the integration of utility functions to make an automatic choice amongst different options. *ESAF* in Section 4.4 showed how to integrate a utility function to make a trade-off between the system load from the security algorithms and the requested security levels.

Misbehavior of one party during the negotiation is usually handled via solely technical means. A constantly misbehaving party might be blocked from future negotiations, for instance, block party that makes a Denial-of-Service attack on the protocol handshake.

Policy intersection also has its drawbacks.
Policies must match to a large degree to allow for policy intersection. Policy intersection as presented above is a syntactic set intersection. The syntax and content of parameters options between two policies must match exactly, for the parameters to be present in the set of mutual acceptable options. Small syntactic variations might already exclude certain options from the mutually acceptable options, even if the interpretation of these options matches for both policies.
To overcome these restrictions, the negotiators that employ a policy intersection protocol must use one single syntactic format to specify policy options and must have a consensus about the interpretation of the policies. For instance, WS-SecurityPolicy [VOHH+09] extends the generic WS-Policy [VOHH+07] specification with additional security related elements and defines the semantic of these elements.

The fundamental weakness of policy intersection protocols is that they cannot discover any solution outside the predefined policies. Policy intersection protocols are therefore unfit to discover previously unknown solutions. A policy intersection protocol cannot be used to discover new solutions to a given problem.
Human to human negotiations, in contrast, are negotiation processes that often start with a vague idea how to approach a problem and might end with a previously unforeseen solution that fits the interest of the involved parties' best. A human to human negotiation is a process where both parties learn wishes and offers from each other and can adapt their negotiation strategy accordingly to discover new options to reach a mutual beneficial agreement. The next Chapters 5 and 6 introduce an iterative agreement negotiation to discover previously unknown solutions through the negotiation protocol. Consequently, policy intersection protocols are mainly used as handshake protocols about technological parameters and do not intend to produce legal binding agreements. Most policy intersection protocols are employed between two parties. Multi party scenarios are better handled through iterative requirements driven agreement negotiation

as presented in Chapter 6.
Please refer to Chapter 8 for a detailed comparison of policy intersection with iterative agreement negotiation protocols for bilateral and multilateral negotiation.

To draw a conclusion, if policy intersection protocols are suitable for a given use case, they are the preferable negotiation method because they are fast, they are easy to implement and to automate. If the use case requires a negotiation protocol that discovers previously unknown solution and defines collaborations of the fly, the negotiation protocol presented in Chapter 5 and Chapter 6 is better suited for the task.

4.6 Summary

Policy intersection is an important technique to identify compatible policy alternatives that are acceptable to all parties. Each stakeholder defines her own policy and can be sure that only options defined in this policy can be chosen. Policy intersection can be applied locally to transform a number of local policies into one common policy, for instance, to bring system policy and application policy into accordance.
Policy intersection protocols use this approach to build consensus about compatible policy alternatives with a remote party. A utility function can be applied to select one parameter set from the different mutually compatible options.
Policy intersection is easy to implement and also straight forward to supervise for an administrator. The good performance of policy intersection protocols makes them the preferable choice for session handshakes within communication protocols. Utility functions with simple algorithms allow for fully automated negotiations.
For these reasons, we also applied policy intersection for the coordination of distributed NAT traversal services in [MüKC08, MüKC09]. For the NAT traversal use case, policy intersection yields compatible NAT Traversal methods in heterogeneous environments.
Policy intersection is an efficient tool with low implementation complexity. However, policy intersection is limited to scenarios where the solution space of policy options can be defined a priori. Policy intersection cannot discover new solutions. Agreement negotiation in more demanding scenarios, for instance, electronic contracting, requires agreement negotiation mechanisms that can discover new solutions and that can define collaborations dynamically. The next Chapter 5 will introduce a bilateral agreement negotiation protocol that can discover new agreement options through iterative negotiation.

The bargain that yields mutual satisfaction is the only one that is apt to be repeated.

B. C. Forbes

5. Bilateral Agreement Negotiation

Negotiation is an iterative Decision Making process [BeVe08] to reach an agreement that is acceptable to all involved parties. Agreement formation between human stakeholders of their organization's interests is a tedious and error prone task. Automated negotiation is an area that receives constant attention from different research communities [JFLP+01] to reduce negotiation time, increase the volume of non-uniform transactions, reach higher quality agreements and allow for a steep integration into business processes. Refer to Section 2.3 for a discussion of approaches from different research communities to automated negotiations.
Automated negotiation reduces the time for drafting agreements. Even in scenarios where human intervention for making critical decisions is required, the automation of large parts of the negotiation still allows for an increase of the volume of non-uniform transactions, higher quality agreements and integration into business processes.
Much research has been devoted to auctions [KeNT00]. However, only small parts of real world contracts define price. Much complexity stems from the specification of dependencies of what a party gives and what a party expects from the opponent.
An agreement can be anything between an agreed upon set of configuration parameters to foster the interaction of systems and a formal contract between different parties.There are many areas where agreement negotiation can be applied. Three possible areas are:

- E-Contracting: The objective of E-Contracting is to reach legally binding agreements with electronic means. It allows for automated creation and supervision of contracts which allows for better scalability, faster business transactions and reduced cost.

- Self-Organization for Autonomic Computing systems: Autonomic computing research aims for facilitating administration of complex infrastructures by introducing self-management capabilities [GaCo03] into networks and devices. Autonomic computing environments rely on devices that are able to make intelligent decisions without human supervision. Agreement negotiation serves as the tool to build common knowledge about the environment and for reaching consensus about future coordinated actions.

- Automated Trust Negotiation: Automated Trust Negotiation supports the cooperation of devices with no prior trust relationship. They can reach an agreement by iteratively exchanging credentials during a negotiation process. These credentials may carry information that becomes a parameter of the further service usage or they can serve as authorization tokens. A careful negotiation strategy helps in protecting sensitive credentials that must only be available to authorized entities. This thesis expands upon the work on stateless trust negotiation in [KPRS+07, Petr05] to reach comprehensive agreements.

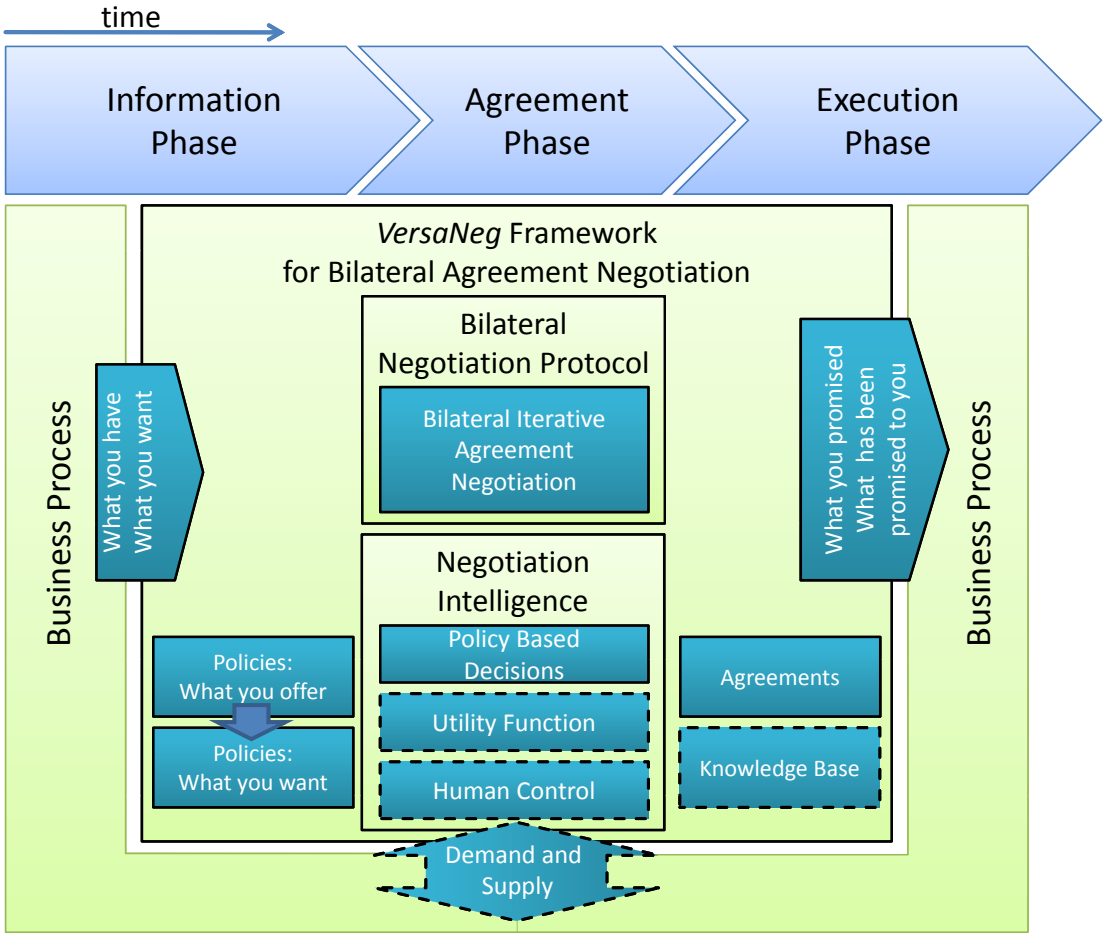

Figure 5.1: *Negotiation Framework during the three phases of the Generic Agreement Formation Model*

This thesis introduces the novel *VersaNeg*[1] agreement formation protocol, language, and framework.
Figure 5.1 shows the *VersaNeg* framework for bilateral agreement negotiation and its activities through all three phases of the generic agreement formation model. Agreement negotiation is just one activity in the scope of a wider business process. The business process triggers the negotiation by providing input on the negotiation objectives. The business process defines what should be achieved by the negotiation and what can be offered. The implementation of the framework relies on local policies that define what the system demands and what the system may offer.

[1] *VersaNeg*: Versatile agreement negotiation about a variety of subjects and tasks

The *VersaNeg* negotiation framework has two main components: The negotiation protocol component is responsible for interpreting negotiation messages and for sending the response. The Negotiation Intelligence provides the critical decisions during the negotiation. These decisions include the Offers to make and the Requirements to state, to choose an agreement or to abort the negotiation. The Negotiation Intelligence receives input from the business process about what to request and what to offer. Depending on the use case, the business process might reserve resources for Offers made during the negotiation. The boxes with solid contours are realized in *VersaNeg* and will be discussed in the following Chapters. The boxes with dashed lines should be present in real world system, but are out of scope for this thesis, as a large body of research exists, for instance, on making smart choices with utility functions (see Section 2.2.1), or for intuitive human machine interfaces [DeFP97, MiBe05].
After an agreement has been formed, the agreement is stored persistently and the agreement is executed by the business process. The framework might interpret the negotiation strategy of the opponent in the negotiation and store this interpretation into its knowledge base.

5.1 Alternative Approaches to Bilateral Agreement Negotiation

Much research exists on negotiations that presents valuable ideas on the language constructs for E-Negotiation, on negotiation protocols, and on the integration of negotiation in business processes.
Kersten et al. [KeNT00] asked Are all "E-Commerce Negotiations Auctions?". They found that existing research overwhelmingly deals with the determination of values (e.g. price), but ignores other aspects of bilateral and multilateral agreements such as the complexity of business models, the challenge to establish collaborations, and the relationships between different parties.
Research on service level agreements (SLA) has a focus on the definition of languages for expressing comprehensive agreements that cover aspects, such as, resource definitions, availability, quality of service, price and penalties [Verm99]. Keller and Ludwig [KeLu03] define a SLA language and a framework that incorporates compliance monitoring of different metrics.
The paper in [Cami00] gives an overview of different modeling approaches and proposes petri-nets and a dialog system for reaching bilateral agreements. Gasmelseid proposed another agent based framework in [XuJG05] and analyzes the integration of human and autonomous agents. The authors in [MaMi01] present a formal model of E-Contracts and the use of deontic logic for imposing temporal constraints and for capturing dependencies between actions specified in the agreement. Automated enactment of E-Contracts is the next logical step after automated agreement negotiation, with the help of dynamically composed services as described in [LuHo03].

There are different web service standards that provide negotiation mechanisms. WS-Policy [VOHH+07] allows for an agreement about technical parameters by providing policy intersection and policy merge operations. The ebXML CPA [ACCF+02] standard defines a language for agreements. The non-normative part of the document suggests policy intersection as the negotiation method. Different research approaches exist for bilateral negotiation with ebXML [VZMB+04, ReTT04]. The WS-Agreement [ACDK+07] standard uses a template-based offer-answer negotiation protocol to establish agree-

ments that incorporate detailed descriptions of services, constraints on possible agreements, and business values, such as rewards and penalties. WS-Agreement [ACDK+07] and the Job Submission Description Language [ABDF+05] are promising standards for negotiating agreements and are suitable for establishing SLAs. It is currently the most promising approach to agreement negotiation for web services.
The lack of WS-Agreement to negotiate agreements iteratively was recognized and work started on a standard draft for WS-Agreement Negotiation [ACKL+07], but did not produce any final results. The authors in [PWZW08] propose an offer and counter-offer extension for WS-Agreement that gives both negotiating parties the ability to insert own constraints and service descriptions in the agreement template. The paper in [BMLH07] presents a privacy agreement model as an extension to WS-Agreement. The VersaNeg approach, in contrast, wants to protect sensitive data by releasing only the absolute minimum set of information for an agreement.

Most agreement negotiation approaches focus exclusively on price and ignore that real world contracts also consist of complex dependencies between the involved parties. Policy intersection and policy merge with WS-Policy or ebXML CPA requires that the structures of the policies match to a large degree. No agreements outside of the predefined templates can be discovered. WS-Agreement goes beyond policy intersection because it allows one party to append own statements to the agreements. It is similar to a standardform contract. A standardform contract is a contract that is pre-defined in most areas and where there are some blanks that can be filled in, for instance, price, date, and quantities. The options for discovering agreements with WS-Agreement are still limited. WS-Agreement can therefore form agreements that are close to the original agreement template, however, the protocol is not fit to discover agreement options that go significantly beyond the choices defined in the agreement template.
Research in the area of next generation service platforms is driven by a need to cope with increasingly dynamic service relationships. It becomes difficult to derive trust from static knowledge about identities and services. A growing body of research targets *Trust Negotiation (TN)* as a method for establishing trust on the fly (see [WiLi02, SmSJ04, BeFS04]). TN systems use a policy-driven iterative negotiation process to reach an agreement between two parties that need not have a prior trust relationship. Trust establishment works through an iterative exchange of digital credentials. Credentials can act as authorization tokens to obtain access to services or they can contain verifiable information linked to a negotiating party. Trusted third parties can certify the correctness of the information in the credentials. Credentials for frequent flyer programs, credit card information, or affiliations to organizations are just a few examples. The main focus of TN research is on the protection of credentials with sensitive information [ElES04]. Access control policies define under which conditions credentials can be released. The TN protocols ensure that only a minimal set of credentials is exchanged and that all preconditions in the access control policy are met before releasing a particular credential.
Trust negotiation introduces many relevant concepts, such as dependency trees between mutually required credentials. However, Trust Negotiation is not a general tool for Agreement Discovery and formation.

5.2 Contributions to Bilateral Agreement Negotiation

The proposed iterative bilateral agreement negotiation protocol in this thesis, centers on the pivotal declarations human-to-human negotiations typically consist of:

- The *Offer* to perform some deed or give something away and
- the inseparable *Requirement* to receive something in return.
- After choosing one agreement option, the corresponding Offers are turned into *Obligations*.

In this novel iterative agreement negotiation protocol, the Offer is a proposal to take a certain Obligation if associated Requirements are satisfied. In the approach presented here, Offer and Requirement belong together. Only if my Requirement is satisfied, I am willing to fulfill my Offer. Hence, Offer and Requirement together constitute a *Proposal*. An iterative exchange of Offer and Requirements declarations leads to dependency trees that describe alternative agreement options. This negotiation process leads to different agreement options, where the Requirements and dependencies of both parties are satisfied for each agreement option. After one agreement option is chosen, the agreement is formed and the mutual Obligations are clearly defined.

The two negotiating parties only have to model the Obligations they are willing to take and can derive Offers and Requirements. The parties do not have to predict the complete agreement structure before the negotiation even started, as with existing approaches. This approach does neither assume perfect knowledge about possible Requirements and matching Offers, nor does it require that the parties know beforehand how a complete possible agreement could be. Instead, the approach, presented in this thesis discovers dependencies during the negotiation and is able to establish novel agreements on-the-fly.

This novel agreement negotiation protocol captures all dependencies between the negotiating parties in one single agreement. The contributions of this protocol include the following areas:

1. A generic negotiation protocol for bilateral agreement negotiation. This protocol discovers different options to reach an agreement between the two parties and eventually turns one agreement option into commitments. Successful negotiations automatically lead to a comprehensive agreement document. Such a comprehensive agreement document contains all details about the relationships between what parties promised each other and under what conditions a party is willing to fulfill its promises.

2. An XML-based message format for requirements-driven agreement negotiation. It can easily incorporate language constructs from related standards. It allows each party to specify its Requirements and different alternative options to satisfy these Requirements. Parties make Offers and take Obligations. Agreements are a set of unambiguous commitments that satisfy all Requirements relevant for one particular agreement option.

3. The protocol uses a careful exchange strategy for the disclosure of sensitive information. The protocol only discloses the minimal set of sensitive information, required for a chosen agreement option. Sensitive information will only be exchanged if it becomes part of the final agreement. Information pertaining alternative agreement options will not be disclosed. By adapting techniques from Trust Negotiation [WiLi00] only details relevant for an agreement will be exchanged, sensitive information that is part of an agreement option that will not be part of the final chosen agreement will not be exchanged.

4. Performance evaluations with the prototype, a generic performance model and analytical results demonstrate the scalability and the feasibility of the novel protocol for real world scenarios.

So far, only the functional contributions of the protocol were presented. Section 5.4 introduces the novel *alternating signatures with message state reduction* approach to make the agreement protocol secure. Here are the contributions of the security extension:

5. Protection not only against malicious third parties but also against a malicious opponent in the negotiation that wants to gain an advantage by forging negotiation states.

6. Non-repudiation of comprehensive negotiation messages that contain the complete negotiation state, allows the negotiator to proof the agreement to an outsider, for instance, to enforce the agreement by legal action.

7. Discussion and evaluation of the trade-offs involved in using different cryptographic primitives.

5.3 Bilateral Iterative Agreement Negotiation with Comprehensive Negotiation States

This Section presents the VersaNeg negotiation protocol and the XML message format. The Section 5.3.1 gives an overview of basic concepts of iterative agreement negotiation and introduces the different phases of the negotiation with VersaNeg. Next comes Section 5.3.2 that explains how Requirements and Offers can be matched automatically by relying on XML technologies. Section 5.3.3 defines the dependency tree of Requirements, Offers and states how agreement options can be identified in the dependency tree. Finally, this Section explains how to perform an iterative requirements-driven negotiation between two parties. Section 5.3.4 goes into more details of the challenges the agreement negotiation faces in real world scenarios. Experiments with the VersaNeg implementation in Section 5.3.5 demonstrate the feasibility and scalability of the approach.

5.3.1 Approach to Bilateral Agreement Negotiation

The basic idea of this novel agreement negotiation protocol is the concept of Requirements and matching Offers. The *Offer* is the declaration from one party to perform some deed or give something away. A party can state various *Requirements* to receive something in return for an Offer. An Offer that has been chosen as part of an agreement is turned into an *Obligation*.
This agreement negotiation protocol centers around an iterative bilateral negotiation process to learn about Requirements from the opponent in the negotiation, to make matching Offers and to state own Requirements. The protocol leads to a complex dependency tree of Requirements and corresponding Offers. One branch of the tree, where each Requirement has a matching Offer, is called agreement option. One agreement option can be turned into an agreement.

5.3.1.1 Agreement Negotiation in Four Phases

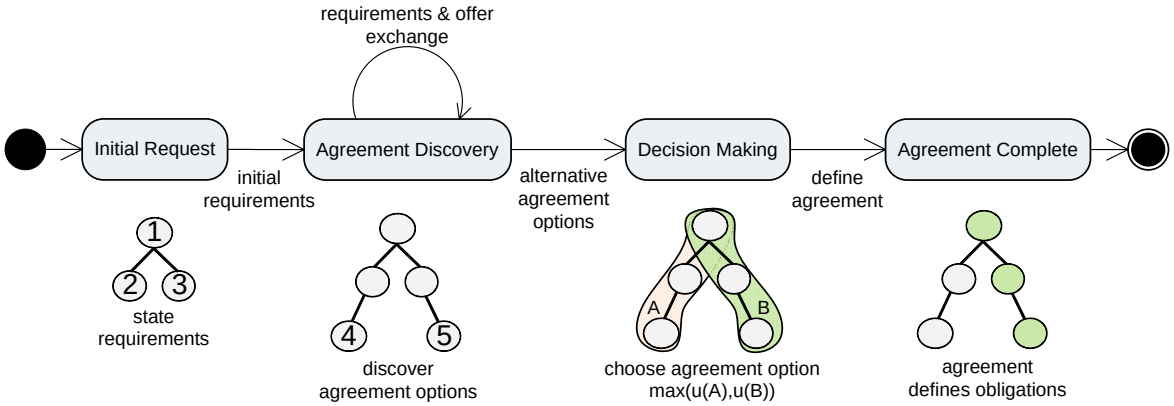

Figure 5.2: *Basic Negotiation Phases*

The party that starts a negotiation is called *Negotiation Initiator*. The party that responds to a negotiation is called *Negotiation Responder* or alternatively *Negotiation Opponent*. A simple method for choosing one agreement option is through a dedicated *Deciding Party*. The Deciding Party is usually the Negotiation Initiator that probably

pays the bill in the end. Alternative consensus building methods that involve both parties can also be integrated with the bilateral agreement negotiation protocol.
The basic agreement negotiation process can be divided into four phases as depicted in Figure 5.5:

1. **Initial Request**: The negotiation is started by sending the initial Requirements to the Negotiation Opponent. These initial statements are the only Requirements that are not connected to an Offer.

2. **Agreement Discovery**: The objective of this phase is the discovery of alternative agreement options. Each negotiating party learns about the Requirements and Offers of her counterpart through an iterative exchange of a dependency tree. Each negotiator inspects the dependency tree and makes Offers and states corresponding Requirements by appending these statements to the leaves of the tree. The dependency tree grows during this process. Branches where each Requirement has at least one matching Offer are called agreement options. After at least one agreement option has been discovered the Deciding Party can enter the next phase.

3. **Decision Making**: The Deciding Party weights the alternative agreement options, for instance, with a utility function, and chooses one agreement option. In the simple form of the protocol, the deciding party concludes the agreement by listing all Requirements and Offers that are part of the agreement. The dependency tree defines the relationship between these Requirements and Offers. This protocol mode is called *forthcoming disclosure mode* because it discloses all information already during Agreement Discovery. The *cautious disclosure mode* avoids the disclosure of information that is not required for the agreement, but this mode introduces additional complexity to the protocol.

4. **Agreement Complete**: Is the phase where an agreement is completely defined. This agreement can be fed into a business process to execute the agreement.

The iterative exchange of negotiation messages happens during the Agreement Discovery phase. The other phases are characterized by their distinct functionalities and are comparable short.

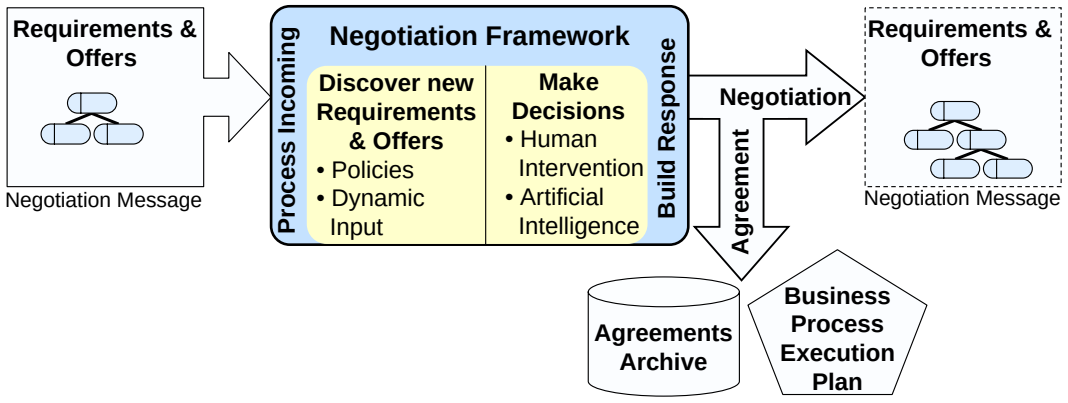

Figure 5.3: *Framework for Agreement Negotiation*

5.3.1.2 Agreement Negotiation Framework

The basic architecture for requirements-driven agreement negotiation is shown in Figure 5.3. Negotiation messages are comprehensive that means they contain the whole state that has been exchanged during the negotiation so far. The negotiation framework receives messages, advances the Negotiation State and sends the resulting negotiation message back. This process continues iteratively until an agreement has been reached or one party decides to abort the negotiation.

After a negotiation message has been received, the Negotiation Logic processes the message. During the Agreement Discovery phase, the Negotiation Logic advances the Negotiation State by appending new Offers to existing Requirements and by appending own Requirements for Offers made (see Figure 5.4). The protocol gives much freedom on how to realize the Negotiation Logic. The implementation in this thesis uses a policy driven rule-based system for realizing automated agreement negotiation. The Negotiation Logic uses policies that define the Offers a party is willing to make and the Requirements that are attached to this Offer. More dynamic strategies for implementing the Negotiation Logic are also possible.
The resulting message is then sent to the opponent in the negotiation after the Negotiation Logic finishes.

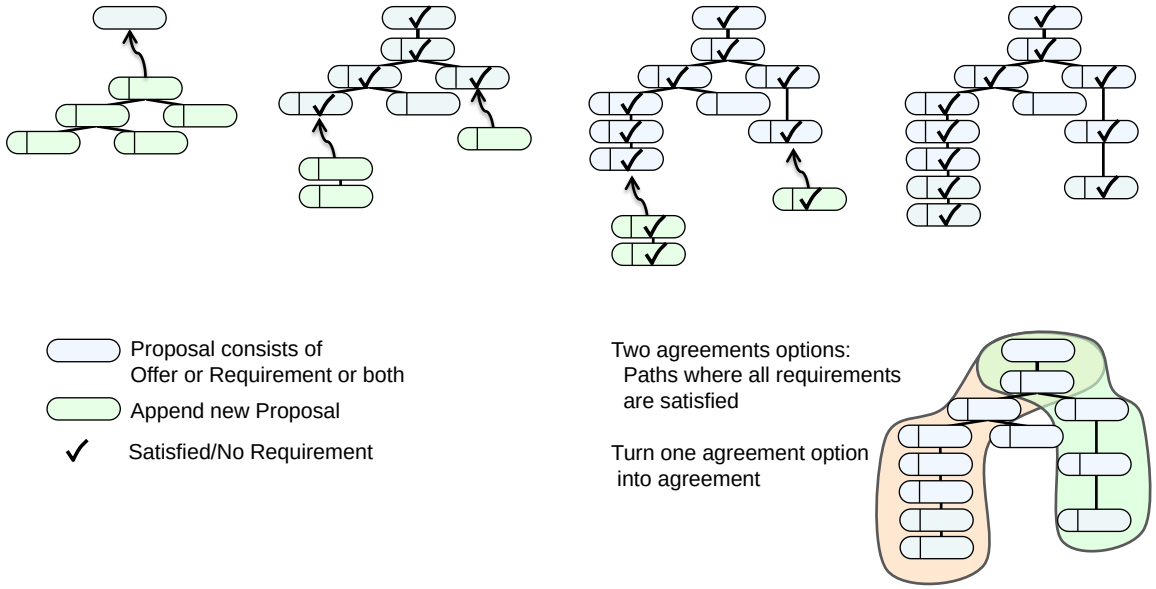

Figure 5.4: *Iterative Growth of the Dependency Tree by appending Proposals (Offers and/or Requirements)*

There are many decisions to make during the different phases of the negotiation. The business intelligence of a company can provide dynamic input, for instance, capacity planning provides availability of resources for fulfilling Requirements in the negotiation. The decisions determine if an Offer should be made, what Requirements are connected to an Offer, and it serves to choose amongst different agreement options. These decisions can be reached automatically, for instance, through a policy-driven system or they can involve a sophisticated negotiation intelligence. The realization of the negotiation intelligence depends of the risk of a decision. Risk-free decisions can be easily automated, for instance, to provide the contact address of the company; high-risk deci-

sions typically involve human supervision. As the negotiation protocols are the focus of this thesis and not the development of artificial intelligence methods, a simple negotiation intelligence is sufficient to demonstrate the viability of the protocol for automated negotiations.

After an agreement has been reached, the agreement is fed into the Agreement Execution phase. Agreements are typically made persistent if they are legally binding. The different commitments made during the negotiation must now be fed into the business process. For instance, one implementation might automatically transform the agreement into a business process execution plan that gets instantly executed. The fact that agreements are comprehensive facilitates this transformation; the relevant knowledge for the implementation of the business process is part of the agreement.

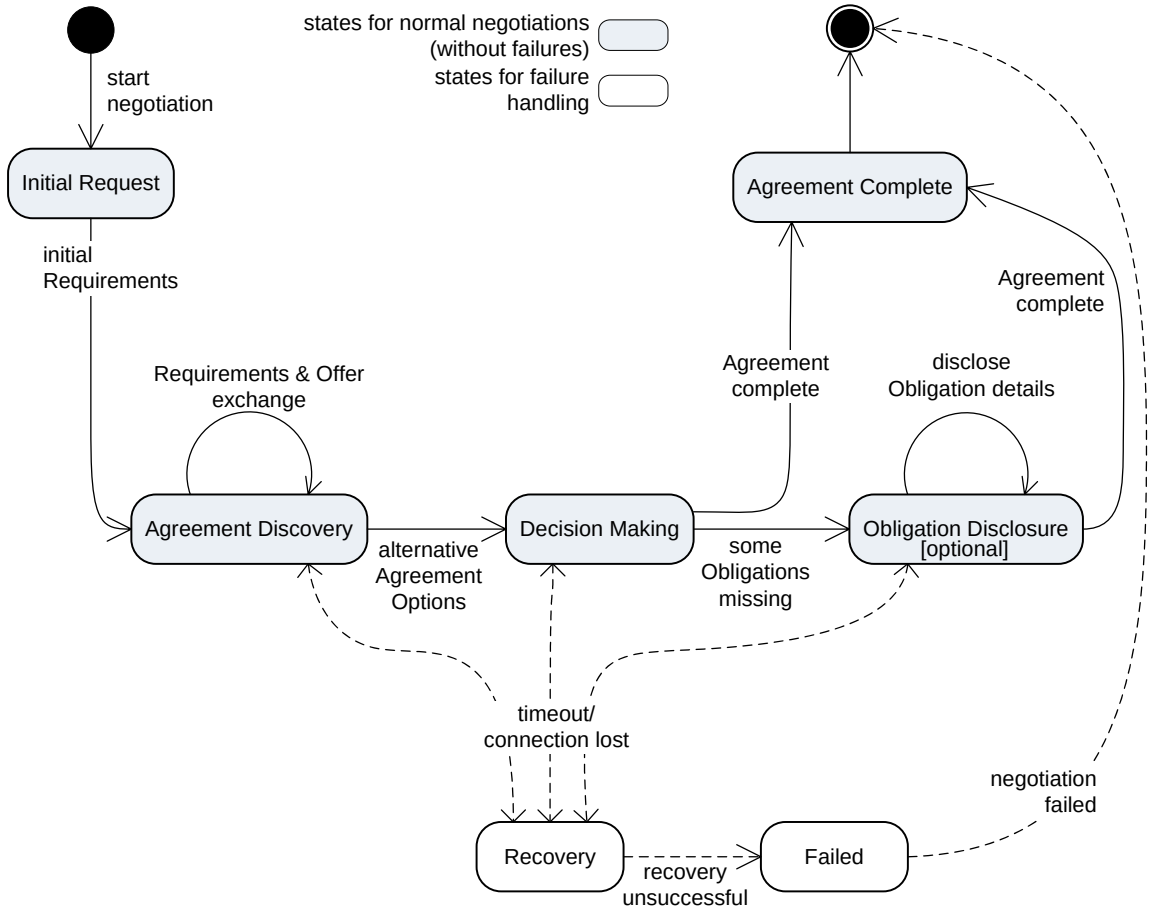

Figure 5.5: *State Diagram of the Negotiation Process*

5.3.1.3 Adapting Trust Negotiation to protect Sensitive Information

The previous Section 5.3.1.1 introduced the four basic phases for agreement negotiation. The standard there is the *forthcoming disclosure mode* where each party directly inserts the relevant information requested by a Requirement, for instance, the credit card information. This behavior leads instantly to complete agreements after one agreement option is chosen.

The problem of the forthcoming disclosure mode is that it forces the negotiating parties to reveal all details about the different agreement options during the Agreement

Discovery phase. However, only one agreement option will become part of the final agreement. The forthcoming disclosure mode also discloses confidential information that is part of agreement options that will not become part of the final agreement. For instance, consider a negotiation that gives a choice to pay either by credit card or by money transfer. If one agreement option contains the Offer to pay by credit card, this party will only want to reveal the credit card information if this Offer is part of the final agreement.
Another problem is that the information in an Offer can be quite extensive, which would waste much bandwidth during the negotiation.

Therefore, this protocol has the *cautious disclosure mode* for protecting sensitive information during agreement negotiation. This facultative mode divides the exchange of Proposals during the agreement negotiation in two phases: *Agreement Discovery* for the exchange of Offers and *Obligation Disclosure* for the exchange of all details that make up the Obligation (see Figure 5.5).

The *Agreement Discovery* comprises the start of the negotiation and the iterative exchange of Requirements as described above. The *Obligation Disclosure* phase will transmit the information that belongs to a specific agreement option. Each party is committed to the chosen agreement option and has to provide all necessary information and must of course satisfy the agreement. This division assures that only information that is relevant for a particular agreement option will be disclosed.

The division in two phases is already a good protection for information that will not become part of the final agreement. The order of the disclosure of sensitive information can also be relevant. For instance, an online music store only allows downloading a song after it received the credit card information to withdraw money. This order of information disclosure can be enforced through a disclosure strategy that obeys the sequence of interdependent Requirements and Offers.
The disclosure of information in a particular order can by achieved by applying strategies from the research on Trust Negotiation [ZhWi08]. The path in the dependency tree that constitutes the chosen agreement option can be treated as a safe disclosure sequence. A safe disclosure sequence enforces that information is iteratively transmitted in the order specified by the branch of the dependency tree.

A negotiation is not supposed to fail during the Obligation Disclosure phase, but if it does the safe disclosure sequence assures that as few as possible information is transferred. In case a negotiation fails under these circumstances, one party may choose not to negotiate with the responsible party in the future. In case the path of the agreement defines Obligations to act immediately, the sequence enforces that actions happen in the order of the Requirements, for instance, grant service access first before money will be transferred.

The following Sections always discuss the cautious disclosure mode. The cautious disclosure mode just introduces the additional Obligation Disclosure phase, and thereby also explains all phases of the forthcoming disclosure mode.

5.3.2 Matching Requirements with Offers and Obligations

The negotiation framework must be able to match automatically if a set of possible Offers and Obligations might satisfy a Requirement. This is a prerequisite for automated negotiation, but is also required for human assisted negotiations. In the latter case, the negotiation framework should show the user a list of all matching Offer for a Requirement. The human user can make a choice then and can ignore Offers that do not match Requirements.

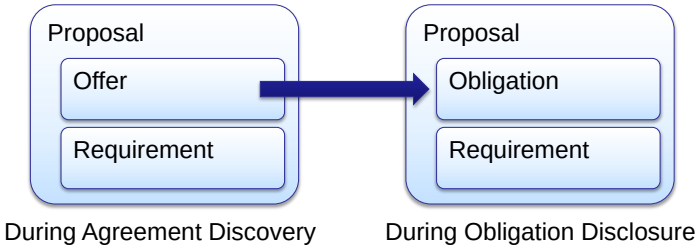

Figure 5.6: *Proposal with Offer during Agreement Discovery and with Obligation after Obligation Disclosure*

5.3.2.1 Requirements, Offers and Obligations

A *Requirement* is a declaration that one party wants something. An *Offer* in our protocol is the proposition of another party to satisfy the Requirement. The Offer has to give enough information to distinguish agreement options and allow for a well-thought choice.

The Proposal structure contains the Offer one party makes and the associated Requirement that describes what this party wants in turn for its Offer (see Figure 5.6). The Proposal structure is the basic entity that one party inserts into the dependency tree. It serves as a container to group what a party wants and what a party can give. Of course it is possible to make Proposals without stating a Requirement. Another special case are the initial Proposals that start a negotiation. These Proposals contain Requirements but no Offers. All Proposals during the Agreement Discovery phase, except the ones in the initial request must contain an Offer.

The *Obligation* extends the Offer by adding all details for satisfying the Requirement. Offers can be perceived as subsets of Obligations where sensitive details may be missing. This separation of Offer and Obligation protects confidential information for Obligations that are not chosen to become part of the final agreement. For instance, a *Requirement* to choose a payment method could be answered by an *Offer* to pay by credit card and later on by an *Obligation* that contains the credit card number. The advantage is that the credit card number is only revealed if the Offer will become part of the final agreement.
Offers are firm, in the sense, that if an Offer will be chosen to become part of the final agreement, the party that made the Offer is obligated to fulfill the Requirement in accordance with its Offer and provide an Obligation that is consistent with the Offer. In other words, this party has to turn its Offer into an Obligation that must validate against the Requirements it promised to fulfill.

By making an Offer, the party is allowed to state one or more own Requirements. Thereby, the protocol discovers dependencies between Requirements and Offers. These

dependencies will eventually result in a list of different agreement options that define what each party wants, what each party promises and how these statements relate, as in a contract.

The protocol does not define the content of Requirements, Offers and Obligations. As long as the party that stated a Requirement and the party that made and Offer share the same understanding about which Offer satisfies which Requirement and they associate the same meaning with these statements However, one Requirement statement must contain all relevant details that another party needs for formulating an appropriate Offer and providing a complete Obligation later on. Our protocol is generic in the sense that it can incorporate all kinds of Requirement specifications as long as the specification

1. is unambiguous
2. fully defines the expected Offer and Obligation
3. all parties have a common understanding about the meaning of Requirements, matching Offers and Obligations
4. Obligations are automatically verifiable if they satisfy the Requirement

The meaning of Offer and Obligation depends on a common understanding of the party that stated the Requirement and the party that made the Offer, that is, which legal obligations, permissions, and prohibitions result from a given VersaNeg Obligation and how it reflects in the business process of the involved parties. Section 5.3.4 discusses how negotiators on a shared marketplace arrive at the same understanding of matching Requirements, Offers, and Obligations. An Obligation could have the same function as a ServiceDescriptionTerm in WS-Agreement [ACDK+07], constitute a commitment as in [WeXu03], or act as a credential for Trust Negotiation [ZhWi08]. Section 7.2 describes in more detail how to use WS-Agreement as the language for expressing Requirements, Offers, and Obligations.

5.3.2.2 Automated Conformance Check of Requirement and Obligation

The purpose of Offers is to support the decision amongst different agreement options. An Offer should only be made if the offeror knows that it can provide an Obligation that validates against the Requirement. The Requirement specifies the required data fields for Obligation and Offer.

For an automated or semi-automated protocol execution, it is crucial to have an automated conformance check to validate if Obligations/Offers match a given Requirement:

- Before any party makes an Offer, it must first check if it is able to produce an Obligation that can be verified against the Requirement. It can simply match all its available Obligations against the Requirements in the Negotiation State.
- A party can verify that the Offers/Obligations from other parties match her Requirements. The latter verification should always succeed for well-behaving partners.

Formally speaking, a function $\mathcal{V}(Requirement, Obligation) = True, False$ returns either $True$ if a *Requirement* is satisfiable by an *Obligation*, $False$ otherwise.

The VersaNeg framework uses XML to encode Requirements, Offers, and Obligations. The automated verification works through a combination of 1.) unique identifiers, 2.) XML Schema [TBMM04], and 3.) creation constraints as in WS-Agreement.

Right now $\mathcal{V}$ is implemented in a way that a unique identifier has to match between Requirement and Offer, the XML schema specified in the Requirement has to validate the XML of the Offer, and the WS-Agreement creation constraints must also be kept. The unique identifiers assure that Requirements are unambiguously named and all concerned parties connect the same meaning of Obligation and Requirement with this identifier. The XML Schema allows for the specification of data and syntactic definitions. Thereby, a Requirement can express a set of rules to define the structure of XML Obligations, to assure the presence of XML elements, and to validate the type and range of data values.
These XML Schema rules are complemented by WS-Agreement style creation constraints. Creation constraints define additional dynamic restrictions on values as part of the Negotiation State. The intention is that the Requirement defines static parts of an Obligation with the XML Schema and the dynamic constraints on values are defined via WS-Agreement style creation constraints. Hence, Requirements and Obligations can be realized as static templates with a few variable fields.

With these three techniques, the framework can determine automatically if the current party can make an Offer for a Requirement, and it can assure that the Obligations provided by the other parties really comply with the own Requirements. The framework works by simply matching Requirements and Offers and can work fully automatic then. It might even be possible to extend this mechanism in the future, to allow for Offers that only partially satisfy a Requirement by rating Offers based on how close they come to satisfying the Requirement.

5.3.2.3 Example of Requirement and Offer

There are no restrictions how to encode Requirements, Offers and Obligations. This Section introduces an example how to encode Requirement and Obligation and how to perform the conformance check by verifying unique Identifiers, by applying XML Schema Validation and by verifying WS-Agreement style creation constraints.

Imagine a scenario where a data storage service in the Internet offers different options to store data persistently. One client wants to use this service to make long-term backups of local data. The client starts a negotiation about the required storage capacity and about bandwidth to access the remote storage.

The XML Requirement fragment is shown in Listing 5.1. It contains a unique identifier with the URN `urn:data.com:res:21431`, an XML Schema to define the expected format of the Obligation `schemaLocation= http://www.data.com/services/backup http://www.data.com/services/backup/backupServiceSLA.xsd` and a creation constraint to request a minimal bandwidth per instance of 1MBit/s through the element at `//Constraint/restriction/mininclusive`.
The implementation automatically verifies the compliance of Obligations for such a Requirement statement. Note that one Requirement element can contain any number of `Constraint` elements for imposing restrictions on the Offer. The elements within Constraint behave like creation constraints in WS-Agreement [ACDK+07]. The resource

```
<Proposal xmlns:xsi=.. xsi:jsdl=.. xmlns:jsdlposix=..>
  <Owner>Party01</Owner>
  <Requirement>
    <Identifier>urn:data.com:res:21431</Identifier>
    <ApplySchema xmlns:db="http://www.data.com/services/backup"/>
    <Constraint>
      <Location>//jsdl:TotalDiskSpace</Location>
      <xs:restriction base="xs:integer">
        <xs:mininclusive value="1073741824.0"/>
      </xs:restriction>
    </Constraint>
    <Constraint>
      <Location>//jsdl:IndividualNetworkBandwidth[@dir="both"]</Location>
      <xs:restriction base="xs:integer">
        <xs:mininclusive value="1048576"/>
      </xs:restriction>
    </Constraint>
    ...
  </Requirement>
  ...
</Proposal>
```

Listing 5.1: *Example of Proposal with Requirement*

description relies on the elements defined by the Job Submission Description Language standard JSDL [ABDF+05].

The storage service provider now parses the Requirement for the constraints and verifies that it has enough capacity for 10GB of reliable storage for the backups and that it allows up to 2MBit/s connectivity.
It constructs the Obligation via a template by using at least the quantities as requested and by making own choices for the remaining values. Because the template-based approach can cover many use cases, there is usually no need to create Requirements, Offers and Obligations from scratch. Because the Obligation does not contain sensitive details, the storage service provider chooses to insert the complete Obligation as an Offer (that is the Offer already contains all information of a future Obligation).

Obligations and Requirements can be realized as templates where only a few data fields have to be adapted for a specific request. The Requester of the cloud service could use a template where only the value of `//jsdl:IndividualNetworkBandwidth` and `//jsdl:TotalDiskSpace` must be adapted for each request. The remaining fields could be static.

The URN `urn:data.com:res:21431` of the Requirement can easily be matched against Obligations a party might be willing to take. The framework employs XML Schema `http://www.data.com/services/backup` to validate that the Obligation satisfies the Requirement. The XML Schema definition assures that all required data fields for the storage service are present. The Constraint elements verify that the dynamic parameters of the request are met. The whole verification with $\mathcal{V}$ happens automatically and allows for the automation of the whole framework.

```
<Offer xmlns:db="http://www.data.com/services/backup" xmlns:jsdl=.. xmlns:jsdlposix=..>
  <Identifier>urn:data.com:res:21431</Identifier>
  <jsdl:JobDefinition>
    <jsdl:JobDescription>
      < jsdl:JobIdentification  >... </ jsdl:JobIdentification >
      <jsdl:Resources>
        <jsdl:FileSystem name="REDUNDANT">
          <jsdl:Description>Reliable Disk Space for Remote Backups</jsdl:Description>
          <jsdl:MountPoint>/home/customer/8907234/backup</jsdl:MountPoint>
          <jsdl:TotalDiskSpace>
            <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
          </jsdl:TotalDiskSpace>
          <jsdl:FileSystemType>normal</jsdl:FileSystemType>
        </jsdl:FileSystem>
        <jsdlposix:IndividualNetworkBandwidth dir="both">
          2097152
        </jsdlposix:IndividualNetworkBandwidth>
        ...
      </jsdl:Resources>
      ...
    </jsdl:JobDescription>
  </jsdl:JobDefinition>
</Offer>
```

Listing 5.2: *Example for Backup Service Offer*

Additional Standards for the Specification of Requirements, Offers and Obligations

Other standards define already extensively how to describe services and requests for a service. The VersaNeg agreement negotiation protocol can leverage the definitions of existing standards and only defines new XML elements where absolutely necessary.
Use Uniform Resource Identifies (URI), EXtensible Resource Identifier (XRI), or Globally Unique Identifier (GUI) as agreed upon identifiers, which unambiguously name the same type of resource.
XML Schema [TBMM04] can complement these identifiers by giving a precise definition of the expected Offer details.
Rely on OWL-S [MPMB+04] or RDF [BeMc04] for requesting a priori unknown Obligations through semantic properties.
The WS-Agreement [ACDK+07] standard defines other relevant elements which might also become part of a Proposal, for instance, Business Values for rewards/penalties and Guarantee Terms that must be kept by a service. JSDL [ABDF+05] and WSLA [KeLu03] are highly suitable for defining service properties typically found in SLAs.

5.3.3 Requirements-Driven Bilateral Negotiation

The negotiation protocol evolves around an exchange of *Requirements*, *Offers* and *Obligations*. One party states its Requirements and the opponent can answer the Requirement by making Offers. Eventually the negotiation arrives at different agreement options. One agreement option is chosen and Offers are turned into Obligations.

5.3.3.1 Dependency Tree of Proposals

The objective of the negotiation protocol is to discover dependencies between *Requirements*, *Offers*, and *Obligations*. A rooted tree structure tracks these dependencies. A node is called *Proposal* if it contains a Requirement, an Offer, or both elements. This also reflects the close bond between an Offer and the corresponding Requirement. During Obligation Disclosure the Proposal may contain additionally an Obligation. The dependencies between Proposals define which Obligations might become part of an agreement. A party may only append Proposals if it makes at least one Offer to the Requirement of the Proposal it wants to append to.

The notation $P_1 \leftarrow P_2$ expresses that Requirement of Proposal P_1 and Requirement of Proposal P_2 both have to be satisfied. Note that each Proposal might have a Requirement and an Offer attached.
If there are multiple Requirements for one Offer, the notation $R_1 \wedge (R_2 \vee R_3)$ denotes that the Requirement R_1 and either the Requirements R_2 or R_3 must be satisfied. The formula $R_1 \wedge (R_2 \vee (R_3 \wedge R_4))$ expresses that either the Requirements $\{R_1,R_2\}$ or $\{R_1,R_3,R_4\}$ have to be satisfied. Prohibitions are special Requirements $\neg P_p$ that state what is not permitted. Sometimes a party wants nothing in return for an Offer. It can append a Proposal that only has an Offer but does not have a Requirement, in this case.

These dependencies between Requirements form a rooted tree structure τ of interdependent Proposals. A node w is called a descendant of a node v, if v is on the unique path from the root to w; v is called the ancestor of w then. All Offers that could possibly satisfy a given Requirement are descendants of the Proposal node that contains the Requirement.

5.3.3.2 XML Message Format for Negotiation States

The XML structure of a negotiation message is shown in Figure 6.12. The use of XML for the negotiation messages allows for an easy integration of complementary language constructs from related standards [ACDK+07][KeLu03].
The Appendix D contains the complete XML Schema definition in Listing D.1 that defines the permissible negotiation messages of the *VersaNeg* protocol. All negotiation messages must validate against this schema. Each party in the negotiation is obliged to validate received negotiation messages with this Schema.
Messages that do not pass the Schema validation and that cannot be transferred into a compliant state through a recovery protocol, will lead to an Abort of the negotiation. The party that sent the corrupt message is obviously responsible for the Abort and must be prepared to face consequences, for instance, a decrease of the rating or in case of binding offers, legal consequences.

The message has a *Context* section that contains information about the negotiating parties, the method for making a decision between available agreement options, and legal statements referring to some contractual framework on how to interpret the agreement. The *Proposal* section contains the initial *Requirement* of the party that initiated the negotiation. The next element is the `ExactlyOne` element that is the parent of a number of Proposal elements. The `ExactlyOne` element states that one of its child Proposal elements must be satisfiable, to become an agreement option. A party can specify dependencies between its own Requirements by providing nested Proposals.

The Negotiation State is the root node of the dependency tree and contains the initial Requirement.
After choosing one agreement option, the *Commitments* element contains references to all Proposals belonging to the agreement and defines which Obligations must be provided to satisfy which Requirements. This element is only present for the Obligation Disclosure and Agreement Complete message states.

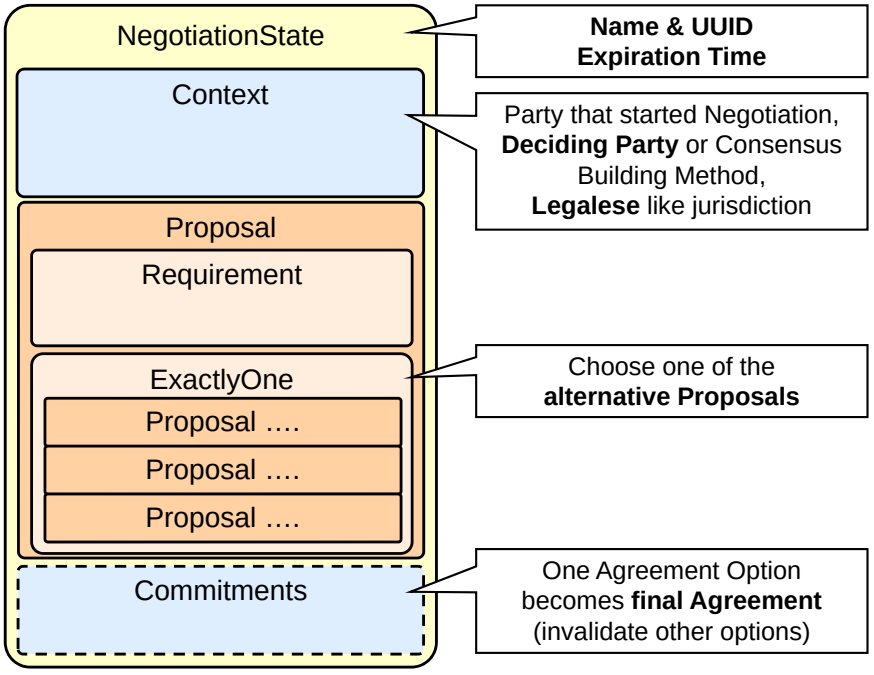

Figure 5.7: *XML structure of Negotiation State*

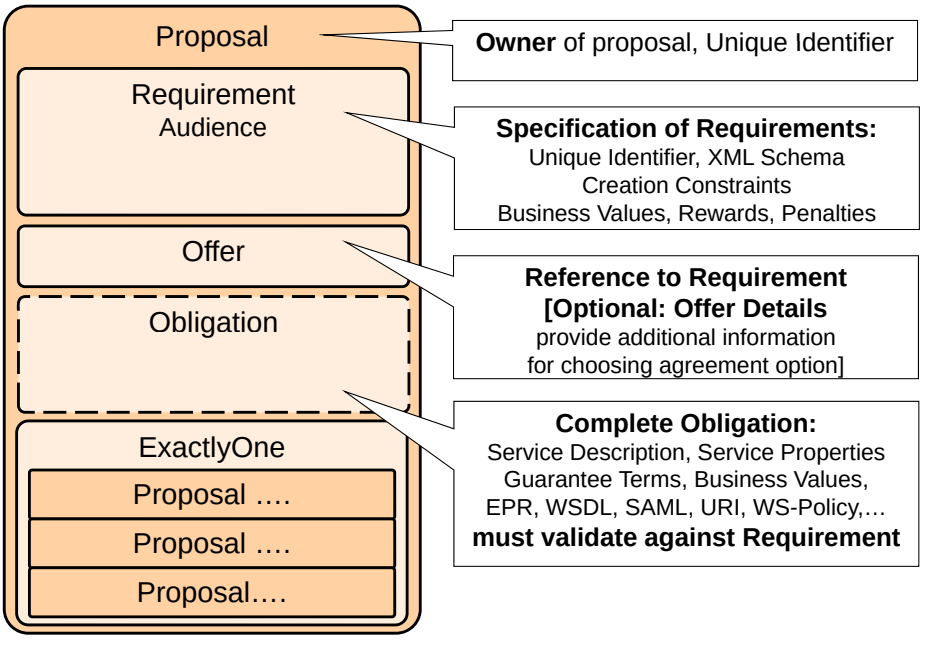

Figure 5.8: *Nested Proposals with Requirement, Offer, and Obligation*

Figure 6.13 shows the recursive structure of the *Proposal* element. It may include one *Requirement*, one *Offer*, one *Obligation*, and must contain the `ExactlyOne` element

Offer	R/S	Description
P_A	O_{backup}	Offer backup service
	$R_{identity} \wedge (R_{bestEffort} \vee (R_{reliable} \wedge (R_{goldStatus} \vee R_{debitCard})))$	
	$R_{identity}$	requires proof of identity of the requester of the service
	$R_{bestEffort}$	requires a choice to either use best effort service ...
	$R_{reliable}$	...or to use service with reliability and availability guarantees
	$R_{goldStatus}$	requires proof that client has gold status subscription
	$R_{debitCard}$	requires permission to withdraw specified amount of money from debit card
P_B	$O_{identity}$	offer customer identity, contact information, email address,...
	$R_{secrecy}$	requires guarantees to use private data only for processing this request
P_C	$O_{reliable}$	indicate willingness to use reliable storage
	$R_{secrecy}$	requires guarantees to use private data only for processing this request
P_D	$O_{goldStatus}$	asserts that the requester has a gold status subscription that entitles to use the service under the terms of the subscription
	$R_{support}$	requires details to contact 24/7 support
P_E	$O_{debitCard}$	allow for direct debit of the given amount from the specified debit card
P_F	$O_{bestEffort}$	indicate willingness to accept best effort guarantees for storage under the general terms
	$R_{secrecy}$	requires guarantees to use private data only for processing this request
P_G	$O_{secrecy}$	promise of the service to keep the entrusted data and the identity of the customer private
P_H	$O_{support}$	provide 24/7 access to level-1 support through the specified contact channels
	$R_{goldStatus}$	require proof that client has gold status subscription
P_I	$O_{secrecy}$	promise of the service to keep the entrusted data and the identity of the customer private

Table 5.1: *Offers of the bilateral reference example, R:Requester/S:Service*

which may have several nested *Proposal* elements. This recursive Proposal structure allows representing the dependency tree in XML.
Examples of the XML format of Requirements and Obligations have already been presented in Section 5.3.2.3. The Offer is a subset of an Obligation. The Offer element has a reference to the Proposal that contains the Requirement, the Offer is intended for. If one party wants to make Proposals during Agreement Discovery it must also make one or more Offers to Requirements.

5.3.3.3 Storage Service Reference Example

It helps to study an example to understand how Proposals relate with Requirements and Offers. A client wants to use an online data storage service to make reliable backups via the Internet. The storage service targets new customers and existing customers with a broad spectrum of service options. The storage service relies on the bilateral agreement negotiation protocol to announce different alternatives to access the service. The storage service requires a legally responsible entity that can be held accountable for misuse of the service. The service offers two modes of operation: Reliable storage with

regular backups and availability guarantees, but only for customers, or free unreliable storage. The requester requires secrecy guarantees about the handling of its data from the storage service.

The Table 5.1 shows different Offers that are relevant for the negotiation. Each party is able to take the corresponding Obligation for an Offer. For instance, the storage service can make the Offer OA:backup to indicate the willingness to serve as a backup service. The Requirement is also part of this table and states for each offer, what a party wants in turn. The implementation of the agreement negotiation framework relies on these definitions of dependencies between Obligation and Requirement to perform automated negotiations.

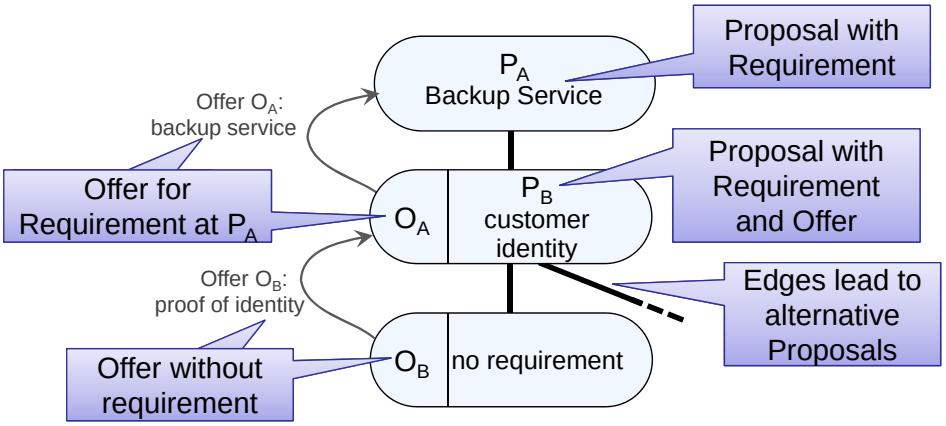

Figure 5.9: *Legend for Graphical Representation of Dependency Tree*

The diagrams used in the following Chapters, show dependencies between Proposals and allow the reader to derive different agreement options (see Figure 5.9). Proposals are depicted as nodes. The Offers are shown on the left side of the node. The Requirement is at the center of the node. The straight edges between nodes state that one Proposal depends on another Proposal. If one node has multiple outgoing edges, these edges indicate alternative Proposal options. The arrows pointing upwards show which Offer promises to satisfy which Requirement. Multiple Proposals by one party may constitute one joint Offer for the same Requirement.

The storage service makes the Offer $O_{A:backup}$ and states a number of Requirements $P_{B:identity} \leftarrow (P_{F:bestEffort}, (P_{C:reliable} \leftarrow (P_{D:goldStatus}, P_{E:debitCard})))$. The meaning of this statement is as follows: "I offer the backup service $O_{A:backup}$ if the requester can satisfy one set of requirements. I require a proof of identity ($P_{B:identity}$) for legal purposes. I give the choice to use my service as best effort with no reliability and performance guarantees ($P_{F:bestEffort}$) or to use the service under certain QoS guarantees ($P_{C:reliable}$). For the reliable service I demand either a proof of subscription with an associated bank account to withdraw money from ($P_{D:goldStatus}$) or I require a debit card ($P_{E:debitCard}$) to withdraw money at a slightly higher rate".
Figure 5.10 shows the tree representation after the storage service added its requirement for offering a backup service. The tree shows that there are currently three paths from the root to the leafs in the tree. Each single path could become part of the final agreement. By following the paths, one can quickly identify the different agreement options. Hence, these are all agreement options.

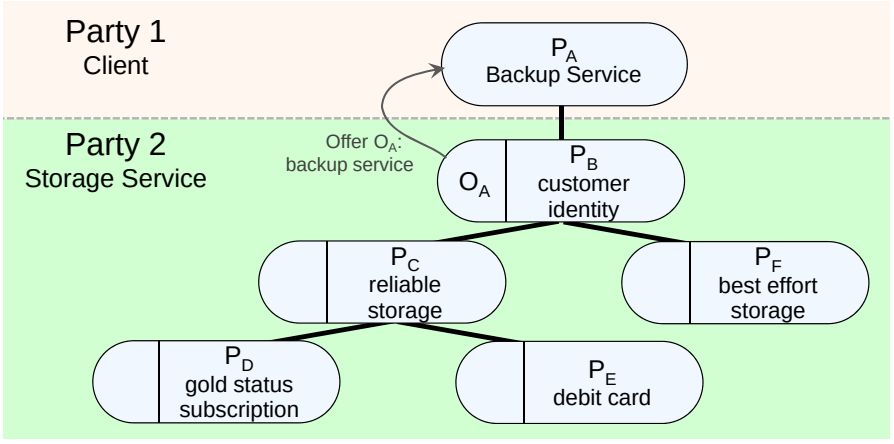

Figure 5.10: *Dependency Tree of Proposals with Initial Requirements after two Messages*

5.3.3.4 Agreement Options

What agreement options are present at this state in the negotiation? We call the set of Proposals α satisfiable if the function $\mathcal{F}(\alpha, \tau)$ returns $True$. If $\mathcal{F}(\alpha, \tau)$ is $True$ all Proposals are satisfiable, either because there is a corresponding Offer for each Requirement in α or because the remaining Proposals do not have a Requirement:

$$\mathcal{F}(\alpha,\tau) = \begin{cases} True & \text{if } \forall P \in \alpha : \\ & (\exists d \in \text{ descendants of } P \text{ in } \tau \wedge \\ & \ d \text{ is Offer for Requirement of } P) \\ & \vee (P \text{ has no Requirement}) \\ False & \text{otherwise} \end{cases} \tag{5.1}$$

An agreement option is a set of Proposals β that has been constructed according to the conditionals where $\mathcal{F}(\beta)$ is $True$. An agreement option β is called a *minimal complete agreement option* when $\mathcal{F}(\beta) = True$ and no $\lambda \subset \beta$ exists with $\mathcal{F}(\lambda) = True$.
For each Proposal father node, only one Proposal child element can become part of one set β. Hence, this relationship leads to a number of minimal complete agreement options that equals the number of its children.

5.3.3.5 Searching for Agreement Options

This section describes how to search minimal complete agreement options in a tree of interdependent Proposals. We present an algorithm for discovering all agreement options that allows humans or artificial intelligence methods to make decisions about agreement options without understanding the message format and the negotiation protocol.

The objective of the following algorithm is to discover all minimal complete agreement options. The algorithm relies on a depth first search [Tarj71] to discover the agreement options. It descends until a leaf node in the Proposal tree is reached. After reaching the leaf nodes, the algorithm ascends and collects satisfiable paths s from leaf to the root of the document where $\mathcal{F}(p) = True$. Let S be the set of all possible satisfiable paths. Let $children(P_x)$ be the function that returns all Proposals that are a child of the conditional in Proposal node P_x. The recursive algorithm that returns the set of minimal complete agreement options is as follows:

```
1 AgreementOptions(P):
2
3     S = {AgreementOptions(child)|∀child ∈ children(P)}
4
5     RETURN SatisfiablePaths(P, S)
```

Listing 5.3: *Depth first search for agreement options*

The function $AgreementOptions(P)$ descends from Proposal node P to the leaf nodes. After it reaches the leaf nodes, it starts the ascent. It collects all satisfiable paths during the ascent. The set S is a set of path sets. Each path set is an ordered sequence of Proposals that are all satisfied.

The function $SatisfiablePaths(P, S)$ takes the current node P during the ascent and the set of satisfiable paths S contributed by its children. It combines these paths with the node P, but only if after this combination, the paths are still satisfiable. The function discards paths that are not satisfiable after this combination. The reasoning is that a Proposal on the path that is not be satisfied a descendant on the path will remain unsatisfied. Offers satisfy at least one ancestor, Requirements can only be satisfied by a descendant. An ancestor can never satisfy a descendant.
If P is the root node of the tree, all paths that $AgreementOptions(P)$ yields are satisfiable agreement options.

```
1 SatisfiablePaths(P, S):
2     if children(P)=∅ ∧ F({P})=False}
3         RETURN ∅
4     else if children(P)=∅ ∧ F({P})=True
5         RETURN {P}
6     else
7         RETURN {e | ∀s ∈ S, e := s + {P} : s ≠ ∅ ∧ F(e) = True}
```

Listing 5.4: *Combine P with $\mathcal{S}$ and form satisfiable paths*

The function $SatisfiablePaths(P, S)$ distinguishes three cases: If this function is called on a leaf node, that is $children(P) = \emptyset$, it adds the current Proposal P to the path, but only if this node can be satisfied by a descendant of P, it returns an empty set otherwise. If P is not a leaf node, the equation in line 8 of Listing 5.4 combines P with all paths in $\mathcal{S}$. This equation adds the current node P to each path in S, verifies that the resulting paths are still satisfiable and returns this set of satisfiable paths.

Under these rules, a Proposal tree contributes one agreement option for each satisfiable path from leaf node to root node. The example shown in Figure 5.10 would have no agreement option because the Proposals P_D, P_E, P_F have requirements that are not satisfiable with the current dependency tree. If we assume for the moment that all Requirements in this tree are satisfiable, the $AgreementOptions(P_A)$ function would return three paths: $\{\{P_A, P_B, P_C, P_D\}, \{P_A, P_B, P_C, P_E\}, \{P_A, P_B, P_F\}\}$.

5.3.3.6 Growing the Dependency Tree during Agreement Discovery

After introducing the dependency tree and the message format, we can now describe the negotiation protocol for the Agreement Discovery phase. An *agreement negotiation*

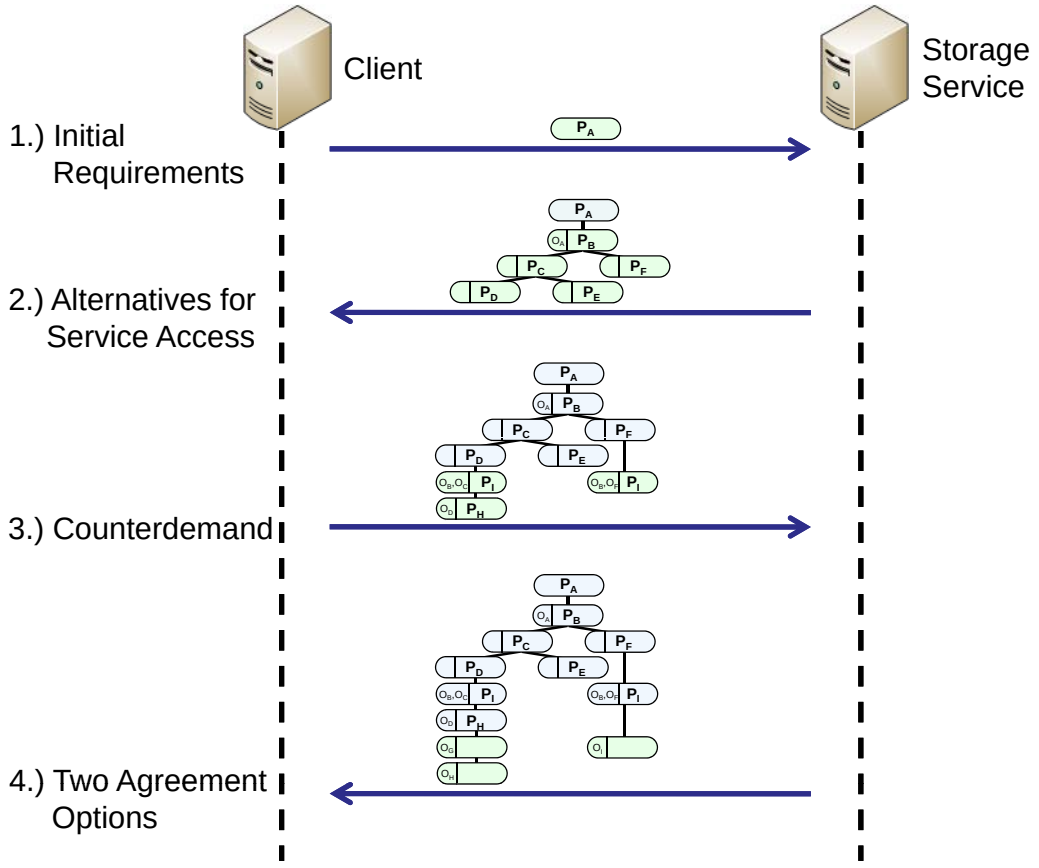

Figure 5.11: *Dependency Tree of Proposals grows during Agreement Discovery Phase*

is an exchange of interdependent Proposals that contain Requirements and Offers. The agreement negotiation can be perceived as advancing a Negotiation State S_i to S_{i+1} by responding to Requirements. This leads to an iterative exchange of Negotiation States where one party receives S_i, makes its Proposals and extends the state to S_{i+1}, which will be sent back to the opponent in the negotiation (see Figure 5.11).
One party starts by appending Proposals that contain its initial Requirements, for instance, the request for the storage service. The opponent in the negotiation determines if it is willing to satisfy one or more Requirements and can now add its own Proposals to the Negotiation State. This party only inspects the Requirements that have been added since it processed the Negotiation State in the previous round.

Proposals can only be appended below the `ExactlyOne` conditional elements (see Section 5.3.3.2). This conditional element states that all Proposals attached below `ExactlyOne`, are alternatives where only one Proposal might become part of the final agreement.

The protocol further restricts where a party may append Proposals to the dependency tree. If $P_o^{p_1} \in S_i$ is the node the party wants to make a number of interdependent Proposals for, this party may append its Proposals to leaf nodes of the biggest subtree of S_i that has $P_o^{p_1}$ as its root node and that only contains nodes that belong to the owner p_1 of $P_o^{p_1}$. If this party already added some Proposals to the leaf nodes of this subtree, whilst processing S_i, it may append its additional Proposals to these own Proposals nodes to form dependencies between own Proposals.

Let $owner(P_x)$ be the function that returns the name of the party which inserted the node P_x. The party p_2 wants to make an Offer $P_o^{p_2}$ for the Requirement in $P_r^{p_1}$. The following algorithm takes the node $P_r^{p_1}$ as an input and returns all nodes this party is allowed to append its Proposals to. Party p_2 may choose any subset of these possible nodes to append to.

```
1  CandidatesToAppendTo(P_n):
2     if children(P_n)=∅
3        RETURN P_n
4     else
5        foreach k ∈ children(P_n)
6           E=E ∪ CandidatesToAppendTo(k)
7           if k ∈ E ∧ owner(k) = self ∧ owner(P_n) = self
8             E=E ∪ P_n
9
10       RETURN E
```

Listing 5.5: *Find all possible nodes to append Proposals to*

This recursive algorithm performs a depth first search [Tarj71] to discover leaf nodes that are descendants of the input node $P_r^{p_1}$. Each leaf node that is a descendant, is a potential node to append Proposals to.
There are some more nodes one could append to. If during the processing of the current negotiation state, a party already stated other Proposals, this party may append its additional Proposal $P_o^{p_2}$ to any previously added proposal that is a descendant of $P_r^{p_1}$. In simple terms, one party might add Proposals inside a subtree containing newly added own Proposals.

The property that one party must not insert nodes within a subtree of nodes that belongs to the owner $P_r^{p_1}$, allows the party p_1 to specify Proposals with Requirements that have all to be satisfied. p_1 may insert a node $P_X^{p_1}$ as the sole child of another node $P_Y^{p_1}$ to force the other party to satisfy both Requirements of $\{P_X^{p_1}, P_Y^{p_1}\}$, otherwise this path can not become part of an agreement option. Hence, the father-child relationship is an implicit propositional *AND* condition that can be depicted as $P_X^{p_1} \leftarrow P_Y^{p_1}$ (see Section 5.3.3.1).

Let us consider an example how to apply these rules. How did the negotiation presented above, arrive at the dependency tree introduced previously in Fig. 5.10?

The customer p_1 stated its initial Requirement $P_{A:backup}^{p_1}$. The storage service p_2 made alternative Offers for $P_{A:backup}^{p_1}$. The storage service made the following Proposals: $P_{B:identity}^{p_2} \leftarrow (P_{F:bestEffort}^{p_2}, (P_{C:reliable}^{p_2} \leftarrow (P_{D:goldStatus}^{p_2}, P_{E:debitCard}^{p_2})))$. There is only one option to append the Proposals to: as child of the `ExactlyOne` node of the Proposal $P_{A:backup}^{p_1}$.

The negotiation continues now as shown in Figure 5.12, with the Proposals and Requirements introduced in Table 5.1. The client p_2 inspects the Requirements by the Storage Service p_1 and must now decide if it wants to make Offers to the Requirements of p_1.

The algorithm inspects all Proposal nodes that the storage service appended to the dependency tree. The client inspects the Requirement $P_{B:identity}$ and makes according

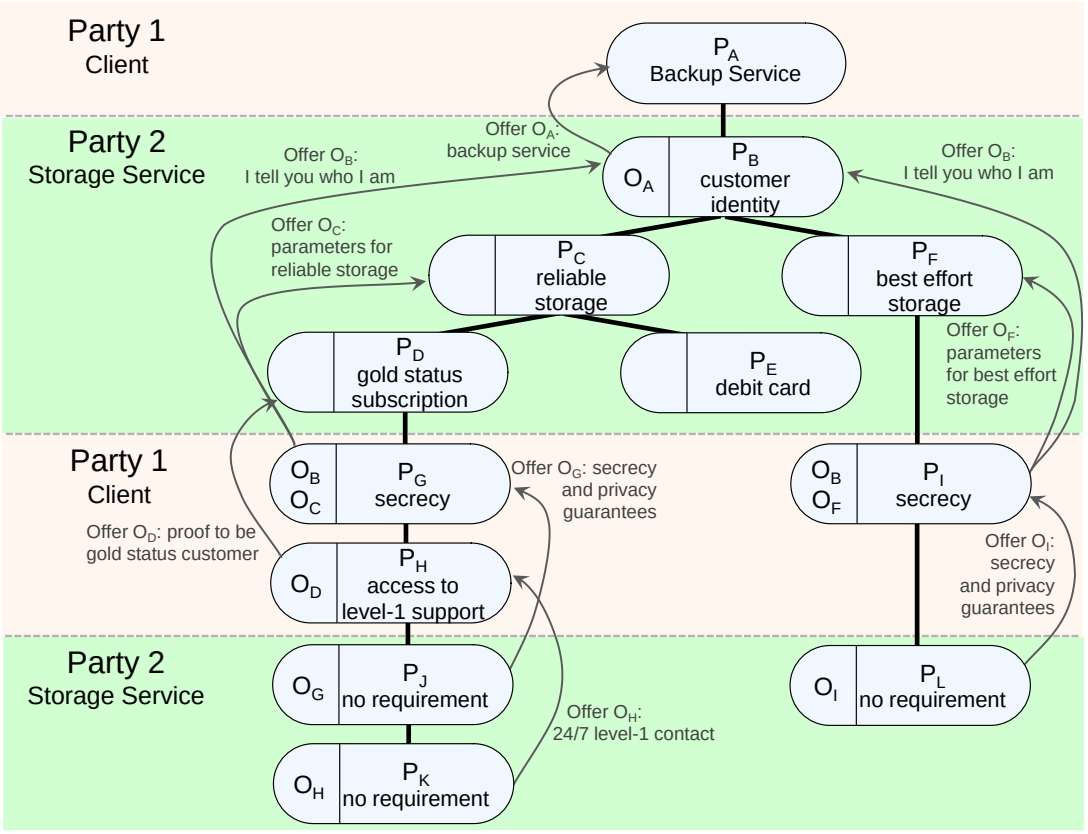

Figure 5.12: *Example Negotiation State*

offers. The client appends the Offers to leaf nodes that are descendants of $P_{B:identity}$: For any proposal to make, there are three leaf nodes where the client might append own Proposals to: $P_D^{p_2}, P_E^{p_2}, P_F^{p_2}$. The client can exclude $P_E^{p_2}$ because it has a policy $O_{E:debitCard} \leftarrow false$ that prohibits that the client offers its debit card number. Hence, for this client, there are only two Proposal nodes to append to: $P_D^{p_2}, P_F^{p_2}$.
The client appends two equivalent Offers $P_G^{p_1}, P_I^{p_1}$, one at each leaf node. The Proposals do not only include an Offer to proof the identity later on, they also include one Requirement: A party that wants to know the identity of the client must make privacy and secrecy guarantees about handling of the identity but also of any data exchanged with the service. The client continues to inspect the Requirements by p_2 and arrives at the two service options: $P^{p_2}_{F:bestEffort}, P^{p_2}_{C:reliable}$. These Requirements demand that the client indicates the willingness to accept the options and that the client specifies the parameters for each storage mode. The client wants to accept both options, but only if the storage service gives privacy and secrecy guarantees for the entrusted data. Because the Requirement for secrecy and privacy is the same as for the Offer to reveal the identity, the protocol allows these identical Requirements to be consolidated into one Proposal. This Proposal $P_G^{p_1}$ has two Offers and one Requirement then. This allows the storage service later on to make only one Offer.
The client continues and inspects the last Requirement at Proposal $P^{p_1}_{D:goldStatus}$ that demands proof of a gold status subscription. The client has such a subscription and can make an according Offer O_D with Proposal $P_H^{p_1}$. The terms of the gold status subscription state that the client is entitled to level-1 support. The client states a Requirement at $P_H^{p_1}$ to demand the contact details for 24/7 access to administrators and technical staff.

After all these Proposals have been added to the Negotiation State, the client sends the state back to the storage service. The storage service inspects the new Proposals and Requirements. It has policies to automatically give the privacy and secrecy guarantees O_G, O_I and that provide the contact details for level-1 support with O_H.

The negotiation has now two agreement options:
$\{\{P_A, P_B, P_C, P_G, P_H, P_J, P_K\}, \{P_A, P_b, P_F, P_I, P_L\}\}$.
The Requirements of each of these two paths are satisfiable through corresponding Offers in the agreement option. The Negotiation Intelligence at the client rates the utility of both options and makes a choice. The client puts the agreement option with the references to $\{P_A, P_B, P_C, P_G, P_H, P_J, P_K\}$ in the Commitments section of the XML message (see Figure 5.8) and sends the message to the storage service.
The negotiation enters the Obligation Disclosure phase (refer to Section 5.3.3.7).

Real world negotiations can be expected to be much more complex as this simple scenario. The example given above would probably also contain payment details, choices pertaining SLA, and define rewards and penalties.

5.3.3.7 Completing the Agreement during Obligation Disclosure

Eventually one agreement option has been chosen and all parties are committed to their Offers. The chosen agreement option defines which Obligations must be present in the final agreement. There are two modes on how the protocol continues after the Decision Making phase:

1. In the *forthcoming disclosure mode*, the agreement is already complete. There is no distinction between Obligation and Offer. All Offers that belong to an agreement are interpreted as Obligations that must validate with function $\mathcal{V}$ against the Requirements.

2. In the *cautious disclosure mode*, the agreement is not complete. The Offers of an agreement define the expected Obligations. The Obligations with all details necessary for the verification with function $\mathcal{V}$ must be provided during the *Obligation Disclosure* phase.

Forthcoming Disclosure Mode

In the optional *forthcoming disclosure mode*, the Offers already contain complete Obligations. Each party only needs to verify if the Offers it has received with function $\mathcal{V}$ (see Section 5.3.2.2). After a successful verification of Obligations, the negotiation enters the Agreement Complete state.

So far we did not elaborate how one party inserts the Obligations for its Offer, which corresponds with a Requirement. The standard option is that each party directly inserts the relevant information requested by a Requirement, for instance, the credit card information. This behavior leads instantly to complete agreements after one agreement option is chosen.

However, the forthcoming disclosure mode is usually not desirable because much sensitive information would be disclosed which will not become part of the final agreement because only one agreement option will be chosen. Another problem is that the information in an Obligation can be quite extensive, which would waste much bandwidth during the negotiation for the unneeded Obligations.

Cautious Disclosure Mode

Therefore, an alternative protocol mode protects sensitive information during agreement negotiation (see Figure 6.12). This mode divides the agreement negotiation in two phases:
Agreement Discovery and *Obligations Disclosure*. The *Agreement Discovery* comprises the start of the negotiation and the iterative exchange of Requirements as described above. The *Obligations Disclosure* phase will transmit the information that belongs to a specific agreement option. Each party is committed to the chosen agreement option and has to provide all necessary information and must of course satisfy the agreement.

The standard *cautious disclosure mode* employs a careful exchange strategy for sensitive information.
The research on agreement negotiation [ZhWi08] introduced safe disclosure sequences. In trust negotiation protocols, credentials are only transmitted if all associated access control policies for this credential are satisfied.
This concept can be applied to the exchange of Obligations during the Obligation Disclosure phase. One can simply consider the chosen agreement option as a safe disclosure sequence. Before a party transmits an Obligation for its Offer, it assures that the corresponding Requirement has a matching Obligation of another party that can be verified with $\mathcal{V}$. Informally speaking, the party only releases its Obligation when the party has access to the full Obligation details it requested in return for the Offer. Only then, it will insert the Obligation for the Offer of the Proposal.

This strategy guarantees that a party is always in a state where all its Requirements, up to the current node in the sequence, are satisfied before it inserts any Obligation. The agreement option defines a sequence for the disclosure of Obligations where this verification always succeeds. To verify that Offers are consistent with Obligations, we assume that Offers are subsets of Obligations where $\mathit{Obligation} \wedge \mathit{Offer} = \mathit{Offer}$.

A negotiation is not supposed to fail during Obligation Disclosure, but if it does, the safe disclosure sequence assures that as few as possible information is transferred. In case a negotiation fails under these circumstances, one party may choose not to negotiate with the responsible party in the future.

The information in an Obligation can either be necessary to fulfill the agreement later on (e.g. credit card information to withdraw money) or the proof that a party has already performed an action (e.g. resource reservation and provisioning of an access token for the cloud service). In case the path of the agreement defines Obligations to act immediately, the sequence enforces that actions happen in the order of the Requirements, for instance, grant service access first before money will be transferred.

5.3.3.8 Duplicates and Cycles

The same Requirement can be present multiple times in different Proposals for one agreement option. Sometimes it is necessary to satisfy each redundant Requirement. It is more likely that it is sufficient to satisfy the Requirement only once. The language of the negotiation protocol allows the party that appends the Requirement to specify that either each redundant occurrence of the Requirement has to be satisfied, or only the first Requirement in an agreement option. For instance, it is probably enough to provide address information one time even if multiple Requirements request the address.

The protocol is guaranteed to terminate eventually by introducing a counter that decreases after each round. If this counter reaches zero the negotiation enters the Decision Pending state. In the expected scenarios for the protocol, this event is not very likely due to the finite number of distinct Offers an organization is able to make.
Even if the Agreement Discovery aborts, there still might be complete agreement options present in the document.

More difficult to handle are cyclic dependencies between Requirements, which could lead to infinite growth of the dependency tree.
Take two parties p_1, p_2, for example, where p_1 makes the Proposals $P_X^{p_1}$ that contains an Offer A and a Requirement B. The party p_2 makes a Proposal $P_Y^{p_2}$ that contains an Offer for B but requires A. This example could lead to the cycle $P_X^{p_1} \leftarrow P_Y^{p_2} \leftarrow P_X^{p_1} \leftarrow P_Y^{p_2} \ldots$
The algorithm keeps track of the relationship between Proposals, by providing a function $\beta(P_m) := P_n$ where P_n is the Requirement that can be satisfied by the Offer contained in P_m. Let $specification(P_X, P_Y)$ be the function that takes two Proposals and returns $true$ if these Proposals are semantically equivalent, $false$ otherwise. A cyclic dependency is present, if for a path p there are two Proposals $P_X, P_Y \in p$ at different locations on the path where the following is true: $specification(P_X, P_Y) \wedge (owner(P_X) = owner(P_Y)) \wedge (\beta(P_X) = \beta(P_Y))$.
One option for treating a cyclic dependency is to mark the affected path as unsolvable. Another option would be that one party allows that the Requirement can be satisfied by an Obligation at a later point in time during the negotiation. We call such a Requirement a *Requirement with a weak timely constraint*. For the example above, p_2 could satisfy P_A^{p1} before it expects p_1 to satisfy P_B^{p2}. If it is acceptable for one party to resort to a Requirement with a weak timely constraint, cyclic dependencies can be avoided. A third option would be to resort to a trusted third party for the exchange of P_A and P_B.

5.3.4 Marketplace, Legal Interpretation, and Failure Handling

Real world usage of negotiation protocols requires more than just a protocol definition. The legal interpretation should be obvious to all parties. There must shared knowledge about matching Requirements, Offers, and Obligations between all parties. The marketplace itself can be the driver to establish this shared knowledge. The protocol must operate in unreliable environments. It must be clear how to take control of the negotiation process, either through human supervision or some kind of decision intelligence.

5.3.4.1 Legally Binding Bilateral Agreements

Before one can answer the question when a legally binding contract is formed, the legal interpretations under different legal systems must be understood first:

- Choosing one agreement option through on single Deciding Party is legally compliant under German contract law (see Section 2.1.3) because offers are binding offers. One party that makes an offer, states that it enters the contract under the terms of the offer if the other party (here the Deciding Party) accepts this offer. Legally binding offers map very well to the protocol implementation with a Deciding Party.

- Offers under English Contract law can be withdrawn at any time (see Section 2.1.2). This could be a problem for this protocol as it would allow parties to make offers just for the sake of determining the limits and options of an automated negotiation system. A legal framework agreement between both negotiators can define that offers under this protocol are legally binding for a reasonable time span. However, the protocol could also allow that one party withdraws offers or even cancels the negotiation at any time.

- In case of international cross-border contracts it is advisable to agree upon a place of jurisdiction (see Section 2.1.5) for the interpretation of the agreements. The place of jurisdiction can be defined through a framework agreement.

When is a legally binding contract formed? One could hold the opinion that only at the Agreement Complete state, the contract is concluded. Both parties made Offers and demonstrated through the Obligations Disclosure the consent with the agreement. An alternative interpretation, especially under German contract law, is also thinkable: Offers are legally binding. It is up to the Deciding party to pick one Offer and form the contract. Hence, already during the Decision Making phase, the contract is formed. The Obligation Disclosure phase would already be part of the contract execution. The parties only deliver the Obligation details (which cannot contradict the Offers made during the Agreement Disclosure phase).
Under both legal interpretations, the negotiating parties still have much power to decide which agreement options are feasible. Only agreement options where all Requirements are satisfiable by a corresponding Offer can become an agreement. Agreement options that have unsatisfied Requirements can not become an agreement.
Bilateral consensus building is also highly relevant for forming legally binding agreements. The negotiating party uses the different agreement options as input to a consensus building process where both parties agree upon one agreement option. The agreement protocol has a supporting role in the contract formation process, then. The legal contract is formed outside the protocol, for instance, through the declaration of mutual acceptance of one agreement option.

5.3.4.2 Shared Marketplace

The protocol is intended for markets with repetitive similar negotiations. A common understanding of matching Requirements, Offers and Obligations is crucial for the execution of the agreement negotiation.

Market Drivers establish a Shared Understanding of Requirements/ Offers/Obligations

Companies are well aware of the potential customers, suppliers, and potential partners in their business domain. Human stakeholders understand very well what their respective companies can Offer and what they require. These stakeholders initially establish a mutual understanding about Requirements and matching Obligations between a group of companies to automate future negotiations. A framework agreement between human stakeholders can be used to define the meaning and all legal liabilities associated with certain Requirements, Offers, and Obligations. With this knowledge the stakeholders can manually formulate templates and rules for the generation of Requirements, Offers and Obligations and thereby enable automated negotiation. For the initial adoption, the framework could also run human-assisted to discover Requirements and formulate

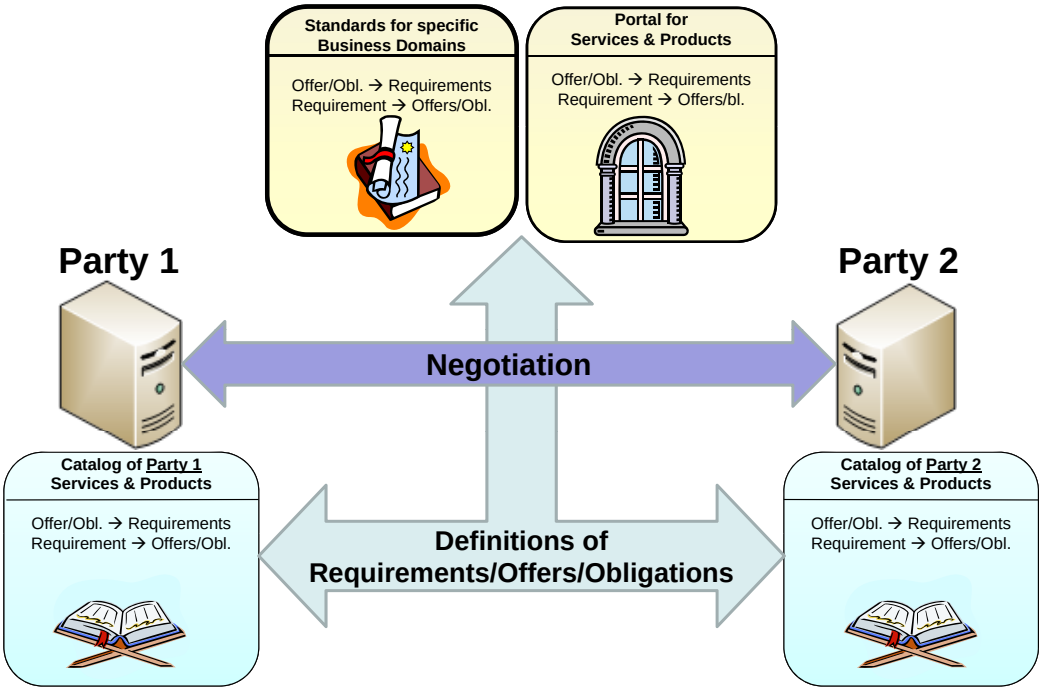

Figure 5.13: *Retrieving Requirement/Offer/Obligation Definitions*

Offers and Obligations manually for later re-use. They agree upon a set of Requirements and matching Offers to automate the negotiations.

Becoming a supplier of a big company usually means to adopt the standards set forth by the customer. For realizing demand-driven supply networks, big companies join organizations such as the Supplier Excellence Alliance [2] to certify suppliers and to standardize the interactions within the supply chain. These organizations might also standardize catalogs of matching Requirements, Offers and Obligations for their business domains. A large-scale deployment of the negotiation protocol could also be driven by service providers and vendors. Each company could publish Requirements and matching Obligations thereby establish de-facto standards. A client can simply customize a Requirement, for requesting a service or for joining a negotiation. The big player might be dominant service providers which offer services to a large number of customers and define how to request services.

Conveying the Interpretation of Offers/Requirements/Obligations

The meaning of Offers, Requirements and Obligations can be established on a shared marketplace. The marketplace helps in defining the interpretation and sharing this knowledge between the potential negotiators. Different options exist to establish a shared understanding:

- **Catalogs**: Catalogs contain basic descriptions of what products a company can offer and what a company may demand, for instance, payment via credit card. Companies announce their catalogs that contain triples of Requirements and matching Offers/Obligations. These human readable descriptions explain how to interpret

[2] http://www.seaonline.org

a Requirement and what kind of legal commitment arises when one party makes an according Offer and takes an Obligation. Two companies can learn about each others promises by inspecting the respective catalogs and by formulating according Requirements. This allows for a bilateral negotiation without prior knowledge.

- **Portals**: Portals are similar to catalogs as they also explain how to interpret Requirement, Offer and Obligation. The main difference is that portals are hosted by a third party that is not directly involved in a negotiation. Portals are very important to build consensus on a market place about typical Requirements, Offers and Obligations. Only this shared knowledge allows one party to state Requirements in a negotiation without inspecting all the options in the catalog of the opponent. It allows both parties to focus on the own Requirements and Offers. Negotiation strategies can be defined based on this knowledge without the need to inspect the catalog of the opponent. The drawback of portals is that the negotiating parties must agree to accept the definitions by specific portals, relevant for the negotiation.
- **Standards**: Standardization bodies might eventually acknowledge the need for a large number of parties to negotiate based upon a shared understanding of Offers, Requirements and Obligations. Standards for different business domains, such as automotive supply chains or logistics, allow any member in the respective business domain to negotiate. The impact of standards is usually much bigger than the impact a portal can achieve. Whole business domains can rely on the standards for their interactions.

For the success of the agreement negotiation protocol, one does not have to assume that standards must exist right from the start. Instead, the market will probably evolve from catalogs over portals towards standards.
Catalogs are the primary means when the protocol is introduced. However, the automation of negotiations is limited with catalogs, because the catalog requires many human decisions about Requirements to choose and Offers to formulate. Each party must examine the options of her counterpart carefully and define at each step of the negotiation how to act. The bilateral agreement negotiation protocol has a supportive role with catalogs; important parts of the negotiation process still rely on human supervision. The advantage is that the protocol structures the negotiation process and allows for the automation of similar negotiations in the future, for instance, for a well established supplier customer relationship.
After many similar negotiations have been concluded, one can expect that portals start to appear that provide definitions of recurring negotiation elements. This step is very important for the generalization and automation of negotiations. Each negotiator can now define a strategy and predict possible agreements based on this set of definitions. The definitions are now generic and allow for negotiations between parties that do not need to interpret the catalogs of their respective counterpart.
Finally standardization takes the role of portals and widens the applicability scope of the definitions. Standardization can be expected to develop a coherent set of unambiguous Requirements, Offers and Obligations. Coherence and unambiguity of the definitions is not necessarily to be expected from portals.

5.3.4.3 Reliability and Failure Handling

Sometimes negotiation fails due to reasons that are out of control of the negotiating parties. For example, communication breaks down because the access network has prob-

lems or critical servers fail and prohibit a negotiator to make decisions. Other problems are transient in nature, for instance, bad connectivity leads to high packet loss and temporary unavailability. Malicious parties can intentionally try to break negotiations. How does bilateral trust negotiation deal with failures?

Problems that require failure handling at a protocol level can arise during the Agreement Discovery, Decision Making, and Obligation Disclosure phases (see Figure 5.5). A failure during the Initial Request basically means that the negotiation did not yet produce any progress and can simply be restarted. After a negotiation reached the Agreement Complete phase, failures might still occur, but these failures must be handled as part of the Agreement Execution within the business process and not by the negotiation protocol.

For the remaining phases, there is a number of common strategies for failure handling on the protocol level. The proposed bilateral agreement negotiation protocol works with comprehensive Negotiation States. This allows fail-over systems to step in when a negotiation server fails. Such a fail-over system can simply reprocess the last message again. Because the message contains the complete state, the fail-over system can continue the negotiation at the point where the original negotiation server failed.
These are two fundamental strategies for failure handling:

- **Resend and reprocess**: A negotiation system that detects a broken connection or a timeout, can resend the last message to the opponent. This last message must contain an additional header that marks this message as a copy of the original message and contains a number that must be incremented for each fork of the original negotiation message.
- **Abort negotiation**: A negotiation can be aborted if it does not produce any viable agreement option. Another reason are negotiations that cannot be recovered. This can be detected if a sufficiently large timeout expires or when all recovery attempts to resend and reprocess fail. A party might decide under these circumstances to abort the negotiation. This party must include all steps taken to recover the negotiation in the abort message. If the abort message cannot be delivered, this party must respond to all messages belonging to this negotiation that arrive from this time on, with this abort message.

This fail-over protocol might lead to duplicate Negotiation States for the same negotiation. The *oldest* Negotiation State always takes precedent over younger states; younger states are discarded in this case. The oldest Negotiation State is either the original state or the copy with the smallest fork number.

Abort messages are undesirable because both parties already revealed sensitive information during the negotiation process. However, Abort messages are always possible and can even be provoked by a party that wants to get out of a negotiation. What is the impact of an Abort message?
During the Agreement Discovery phase, no agreement has been concluded, yet. Hence, an Abort means that maybe too much information has been revealed during the negotiation about Requirements and Offers. As the details of the Obligations are still missing with the cautious disclosure mode, only little information is revealed.

It depends on the legal interpretation of the Agreement protocol at what time (Decision Making/Agreement Complete), a legally binding contract is formed (see next Section 5.3.4.1 above). If the Contract is formed only when the Agreement Complete state is reached, an Abort leads to a terminated negotiation in any of the Agreement Discovery phase, Decision Making phase, and even during the Obligation Disclosure phase. Only with the last disclosure of an Obligation, the agreement would be complete under this interpretation.
In contrast, when a decision during the Decision Making phase has been reached, and this can already be considered to be a legally binding contract, an Abort can only be accepted before the decision is reached. This semantic gives the Deciding Party much power to decide if a negotiation should be terminated or if an agreement has been reached. To avoid unfair behavior, the Deciding Party should be a trusted third party. Distributed consensus building would also lead to fairness during the Decision Making process.

5.3.4.4 Negotiation Intelligence for Steering the Negotiation Process

Due to the impact and risk associated with an agreement, care has to be taken how to engage in a negotiation. Some decisions might be highly critical and humans might have the final word, other decisions are less risky and artificial intelligence might be employed to reach a high degree of automation.
The focus in this section is not the specification of artificial intelligence methods, but rather the definition of interfaces for intelligent Decision Making modules to take control of the negotiation.

```
makeOffer(Requirement, [available_Offers],
          negotiation_state, knowledge_base):
    RETURN [new_Offers_and_Requirements]

chooseFinalAgreement([agreement_options],decision_method,
                     negotiation_state, knowledge_base):
    RETURN agreement_option
```

Listing 5.6: *Pseudocode Interface for Negotiation Intelligence*

During the *Agreement Discovery* phase, parties already have the choice if and how to make Offers for Requirements. The `makeOffer` interface takes the complete Negotiation State up to this point. It will be invoked for each new Requirement node that is applicable for this party and returns a set of new Offers and Requirements to append to the tree.
The implementation of `makeOffer` in this thesis relies on a fully automated system with disclosure policies. Each Offer has an associated disclosure policy which will be invoked for each matching new Requirement in the Negotiation State. This method copies the Requirements defined by the disclosure policy automatically into the Negotiation State for each Offer to make.

The `chooseFinalAgreement` method is for making a choice between available satisfiable agreement options. The input is a set of all agreement options the output is the chosen agreement option.
The chosen method for the implementation for this thesis is to have one designated party that makes the decision. This deciding party would presumably be the party which made

the initial request and which will probably pay the bill in the end. Another method could be to build consensus with all parties about the most favorable option [ZMFA04]. Reverse auctions could help in determining the fair price of each agreement option.

The *Obligations Disclosure* phase does not give as much freedom of choice, because the specification of the agreement is probably already quite detailed. Decisions have to be taken mainly for satisfying agreements, for instance, to determine the most efficient strategy to assign resources to fulfill the agreement.

After the agreement is complete, the parties can rate the compliance of the other parties with the agreement to derive trust for future interactions and add it to the `knowledge_base`.

5.3.5 Experimental Results

The performance of iterative agreement negotiation with comprehensive Negotiation States, determines the applicability for automated negotiations. The time for complete negotiations determines the applicability for automated processes. The faster the negotiation is, the better is it suited as part of time critical business processes. The other issue is the scalability. The number of negotiations a sever can handle in a time interval determines the number of negotiation servers an organization needs for peak demand.

The two party experiments used three PCs on the same LAN. Each PC had an Intel Core 2 CPU 6600 at 2.40GHz and with 2.6 GB of RAM. The Operating System was a standard Debian GNU/Linux 5.0 installation with Linux kernel 2.6.26. Two machines were the negotiators. One machine served as the control PC that issued scripted commands to the two negotiators. The machines were all connected to the same Gigabit Ethernet (1000Mbit/sec) switch. There was no other network traffic on this LAN except for these three PCs. The round-trip time of ICMP echo request and echo response was smaller than 1 millisecond in this LAN. The libxml2 (http://www.xmlsoft.org) library was used for parsing the XML document into a DOM representation.

The reference example from Section 5.3.3.1 serves as the test case for the following experiments. The negotiation consists of 8 messages: 4 for the *Agreement Discovery* and 4 for the *Obligation Disclosure*. The implementation verifies that Requirement and Obligation match and also tests if the creations constraints are kept. The aggregated size of all negotiation messages in this experiment is 100kB.

The first experiment determines the duration of a single negotiation. That is the time starting with the first negotiation message until the complete agreement is returned after 8 messages. Each negotiation was run 120 times with a separation of 4 seconds between each negotiation. The individual negotiations did not overlap. There were no resource consuming background jobs in the operating systems that could compete for resources. The negotiations in this experiment only used one of the two available CPU cores.

The Table 5.2 shows the results from the experiment to determine the duration of a single negotiation. The table presents the mean and the quantiles. The evaluated negotiation finishes in about 50 milliseconds. Nearly 56% of the time is spend on the processing of the Negotiation States. Processing is the time for the evaluation of a received Negotiation State, and the time for preparing the answer. Processing also contains the time spent inside the negotiation server for assigning a message to a thread and for passing Negotiation States to the internal functions. The DOM parser and

	Negotiation Duration					
	Mean	1%Qt.	5%Qt.	Median	95%Qt.	99%Qt.
Total Negotiation Time	0.046	0.031	0.036	0.045	0.049	0.052
Processing	0.026	0.024	0.025	0.025	0.028	0.028
DOM Parser	0.009	0.007	0.007	0.009	0.009	0.009
Communication	0.011	0.010	0.011	0.011	0.014	0.016

Table 5.2: *Negotiation Duration (sec) with Mean and Quantiles*

the generation of the string representation of the message took 20% of the time. The communication amounts to 24% consists of the time spent on the wire for transmission of the negotiation messages, the time in the TCP/IP stack in the operating system, and the time in the multi-threaded server until processing starts. The experiment was performed in a network topology with low RTTs, typically only found within the LAN of an organization. The time for communication will grow for larger RTTs, for instance, for negotiations via the Internet. The numbers show that negotiations are reasonable fast and indicate that integration into time critical business processes is feasible.

The next experiment was conducted to determine how well a negotiation server handles a high load of negotiations. What is the maximum negotiation rate of completed negotiations per second in the setup stated above? Both negotiators had the same processing resources. One PC started negotiations. This PC is called negotiation initiator. The PC that reacts to the negotiation request is termed negotiation responder. The negotiation rate was determined at the negotiation responder by counting the number of successful negotiations that terminate in one time interval. The initial message that triggers the negotiation is cheap because it only contains a resource identifier for the negotiation. This allows the negotiation initiator to overload the negotiation responder with negotiation requests without spending too much resources on its own. The resource consumption for processing of the following negotiation messages leads to roughly equal load at the negotiation initiator and the negotiation responder. Some small tests confirmed that increasing the number of negotiation initiators does not lead to a higher overall negotiation rate at the negotiation responder. The number of negotiation requests generated per second is called the offered negotiation rate.
The negotiation initiator increased the offered negotiation rate in 10 increments. The red line in Figure 5.14 shows the results. The vertical axis shows the negotiation rate that has been achieved for the given offered negotiation rate at the horizontal axis. Each point shows the negotiation rate of complete negotiations at the negotiation responder. The error bars indicate the 5% and the 95% quantiles of the negotiation rate per second. The negotiation responder can cope with the offered negotiation rate up to 44 negotiations per second. Because each negotiation consists of 8 messages, 352 negotiation messages were processed per second in this experiment. An additional increase above this negotiation rate cannot be handled by the negotiation responder and the initial messages start to queue.

The Intel Core 2 CPU 6600 is a dual core CPU. The VersaNeg implementation can utilize both cores because negotiation messages are handled by posix threads which can run on both CPU cores in parallel. This leads to another experiment for determining to what extent the VersaNeg implementation can benefit from the multi core architecture. One CPU core at the negotiation responder was disabled. The negotiation initiator still

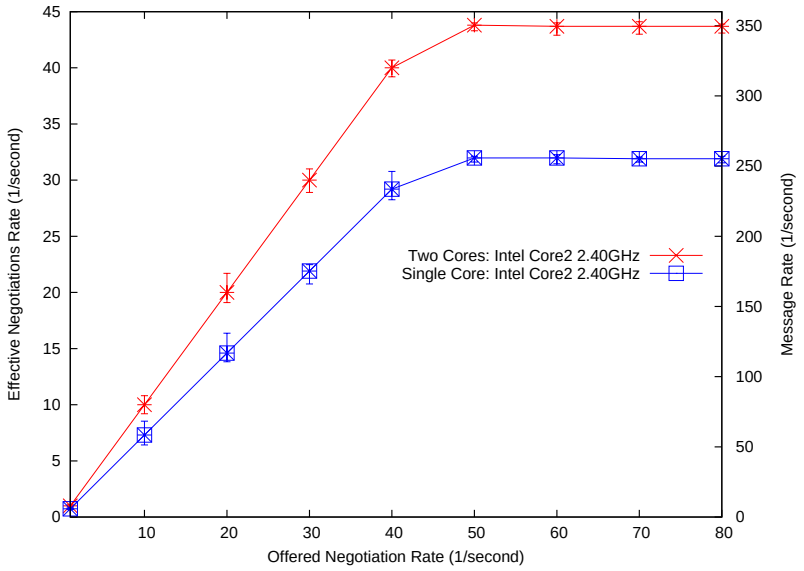

Figure 5.14: *Size dependent Scalability of Trust Negotiations*

ran with two CPU cores. The blue line in Figure 5.14 shows the result for one CPU core. The negotiation responder handles a negotiation rate of 32 per second with one CPU. This is a decrease of the negotiation rate by 27%. This number indicates that the VersaNeg implementation does not utilize the full processing capacity of the second CPU. In an ideal case, each CPU would provide 50% processing capacity in this setup. That is not the case because even though the CPUs run in parallel, both CPUs still compete for access to memory via the shared memory bus, they must cope with the synchronization of CPU caches, and compete for access to the network stack. These factors limit the benefit of the additional CPU core for the VersaNeg implementation.

One obvious factor that determines the negotiation rate is the size of the Negotiation States. As the VersaNeg protocol has comprehensive Negotiation States that grow at each iteration, the message size might become a determining factor for the scalability of the system of the negotiator. The Negotiation States during Agreement Discovery phase are comparable small. Negotiation States can grow significantly during Obligation Disclosure as there is no limit on the Obligation size. What is the negotiation rate for growing Negotiation States?
The experiment setup is as stated above. The negotiation initiator increases the Negotiation State by inserting a large obligation into the first message during obligation disclosure. That means that there are four large messages in the negotiation. The Figure 5.15 shows the achievable negotiation rate for growing negotiation sizes. The horizontal axis shows the aggregated size of all Negotiation States during the individual negotiation. The vertical axis shows the achieved negotiation rate for each aggregated size. One can see that the negotiation rate is subject to exponential decline because it decreases at a rate that is proportional to the aggregated size. A slightly smaller negotiation with an aggregated size of 50kB achieves a rate of 47 negotiations per second. A negotiation with an aggregated size of the Negotiation States of 5000kB, only achieves a negotiation rate of 6 negotiations per second. The total consumed network bandwidth is 240Mbit per second. Hence, the network is not the bottleneck for large negotiation messages but the XML DOM parser and the processing is.

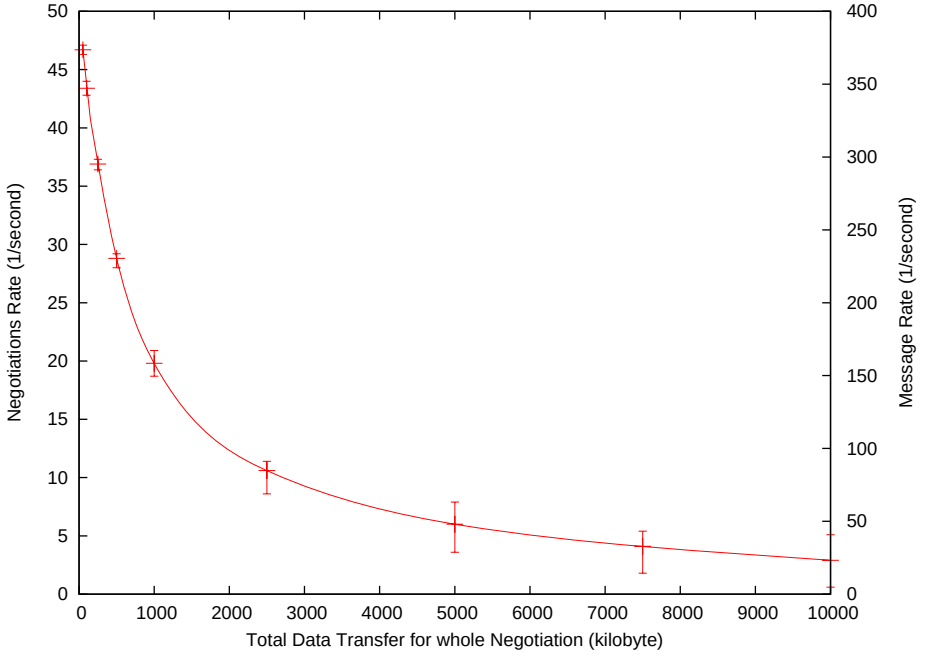

Figure 5.15: *Size dependent Performance of Agreement Negotiations*

5.3.6 Discussion of Requirements-Driven Bilateral Negotiation

This Section introduced the VersaNeg protocol for bilateral iterative requirements-driven agreement negotiation. This novel approach can discover unforeseen agreements on the fly. The parties must only model their individual Requirements, Offers and Obligations. The protocol discovers the dependencies and clearly states alternative options to reach an agreement. The requirements-driven negotiation enables iterative agreement negotiation.

However, this approach only works with a shared understanding of Requirements, Offers and Obligations. Section 5.3.4.2 discussed different options for how to establish knowledge about Requirements and matching Offers/Obligations. One can use the VersaNeg framework as a tool for human negotiators. A human operator of VersaNeg can learn about previously unknown Requirements and create Offers/Obligations on the fly. This enables the VersaNeg approach to start without any shared knowledge about Requirements, Offers, and Obligations and still arrive at a situation where there is a clear definition of these entities between pairs of parties. One can expect that market-dominating companies and standardization bodies play an important role in establishing a shared understanding in the market to enable largely automated negotiations. This would largely eliminate the need for a human operator that defines matching Requirements and Offers/Obligations.

Another important issue is the language for Requirements, Offers, and Obligations. The VersaNeg protocol clearly defines the dependencies between these three fundamental constructs. It does not put any constraints on the language for Requirements, Offers, and Obligations. As long as the language can be encoded as XML, it is compliant with the protocol. Therefore, it is even possible to use binary formats. WS-Agreement is an XML standard that defines a bilateral agreement. In the VersaNeg approach, WS-Agreement templates can be used as Requirements,Offers and WS-Agreement agreements as Obligations. The advantage of using WS-Agreement is to have a language

that is flexible to describe the service terms under negotiation, and to clearly state what service a party expects and which constraints apply. WS-Agreement links service descriptions to rewards and penalties, based on measurable service level objectives. By using WS-Agreement with VersaNeg, all research that extends WS-Agreement with semantic technologies, such as OWL [OVSH06], also becomes applicable for VersaNeg negotiations.

The policy driven VersaNeg implementation with a XML message format allows for a large degree of automation without sacrificing control. Requirements-driven negotiation gives more freedom to discover unforeseen agreements. Obviously, this introduces the risk that the agreement discovery arrives at a state where some potential agreement options are undesirable for a party. Still, each party has full control of the outcome. It can always choose not to react to a Requirement. This path in the tree would automatically be excluded from the set of possible agreement options. It is therefore sufficient for each party to check that the resulting agreement options are acceptable to this party, before it appends Offers to a Requirement.

The stateless nature of the agreement negotiation system is advantageous in open environments due to their ability to recover from failures by simply processing the last message again. Furthermore, the stateless negotiation reduces resource consumption at the negotiating parties for long lasting negotiations. Especially, when humans are involved in the negotiation process, and negotiations can even take days instead of seconds, this property becomes advantageous. The experiments show that the VersaNeg approach can operate reliably, scale easily, and work efficiently with common service hosting topologies.

5.4 The Security Challenge in Stateless Bilateral Agreement Negotiation

Security is not just a challenge in the presence of malicious third parties. Without protection, comprehensive Negotiation States, as introduced in the previous Section, can easily be manipulated by a malicious opponent in the negotiation. This Section will introduce security to protect against malicious third parties and malicious opponents in the negotiation.

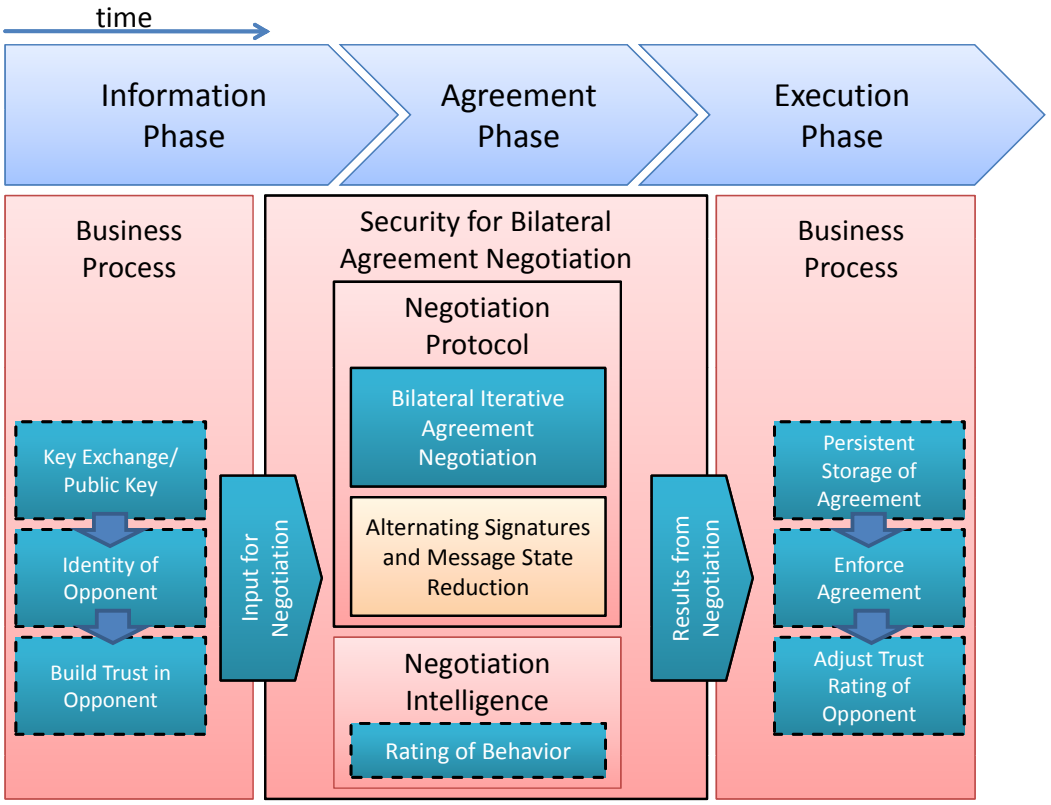

Figure 5.16: *Message State Reduction and Verification with Alternating Signatures*

The security enhancement with *alternating signatures and message state reduction* is an important part of a comprehensive security strategy that must be implemented to use bilateral agreement negotiation in real world scenarios. Figure 5.16 shows different activities that should complement the secure bilateral agreement protocol.
One of the first activities is to identify potential opponents for a negotiation. E-Negotiation requires that the negotiation systems can proof the identity via electronics means. Public key cryptography together with a public key infrastructure is a popular method for authentication in the Internet. For instance, digital X.509 certificates, issued by a Public Key Infrastructure (PKI) authenticate the negotiators.
Trust cannot be derived solely by applying technology. Ultimately trust is established through human action, for instance, through initial contracts between human stakeholders of the respective organizations. Legal framework agreements can be formed between the humans to allow for automated agreement negotiation in the future.

The realization of these activities is covered by a broad range of research, for instance, on identity management, reputation based trust, and enforcement of agreement via commitments. Consequently, this Section has a focus on how to make the bilateral

iterative agreement negotiation protocol secure, a topic where no solution exists in research, yet.

5.4.1 Security Aspects of the Negotiation

Security is especially challenging in agreement negotiation with comprehensive Negotiation States, due to the large potential negative impact and the legal dimension of the negotiation.

5.4.1.1 Attacker Model for Stateless Negotiation

There are two types of adversaries for stateless negotiation:

1. A *malicious third party* that is located on the message path between two negotiating parties and is able to intercept, manipulate and replay negotiation messages.

2. The *opponent in the negotiation is a malicious adversary* that tries to cheat during the negotiation. End-to-end security is no protection if the other party in the negotiation wants to take unfair advantage by forging the Negotiation State in parts or as a whole. Each peer wants to protect the Negotiation State from malicious modifications because this document states Obligations that the peers must fulfill. If an adversary succeeds in manipulating the Negotiation State, it can generate agreements that favor itself and are very unfavorable for the victim. Iterative stateless agreement negotiation protocols with comprehensive negotiation states are especially vulnerable to this kind of adversary because the complete history of the negotiation including all Offers, commitments, and promises made in previous rounds could be falsified.

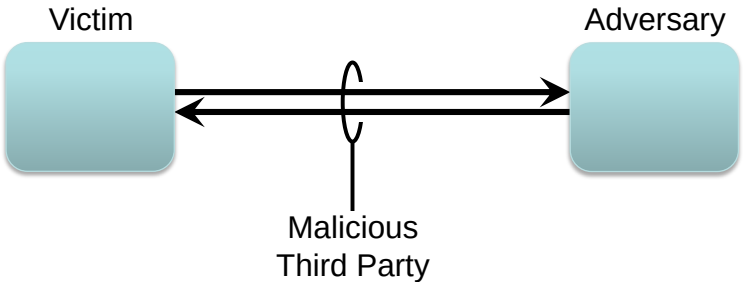

Figure 5.17: *Attacker Model for Stateless Agreement Negotiation with Comprehensive Negotiation States*

5.4.1.2 Discussion of Security Aspects

Both parties in a negotiation can protect the integrity and confidentiality against a malicious third party by using asymmetric cryptography and digital signatures with cryptographic protocols, like TLS/SSL [DiRe06] or WS-Security [ws-s04]. It is more difficult to protect the negotiation against manipulations of the opponent in the negotiating. Stateful implementations must check if negotiation messages are consistent with the local state. For negotiations with complex dependencies and much semantic information, conformance checks against local state might be difficult.

The VersaNeg bilateral agreement negotiation framework, presented in this thesis, relies solely on the received comprehensive *Negotiation State* to decide about the correctness of the Negotiation State. Our envisaged strategy is to sign the Negotiation State each time before it is sent and to use a reduce-verify approach to track back changes [KCRS09]. The other party will make changes to the document by appending Proposals and Obligations, but it will not modify the existing parts of the tree. When the state is received back, the *Negotiation State* can be reduced to the priorly signed state and be evaluated against the digital signature. However, our implementation does not yet support this check.

Denial-of-Service attacks on agreement negotiation systems are critical even for a system that must not keep state. However, Denial-of-Service attacks on stateful agreement negotiation systems have a higher negative impact because attackers can exhaust resources of individual agreement negotiation service entities more easily than resources of a cluster of agreement negotiation service entities working in parallel and being able to share load without restrictions. If applicable, it is a safety precaution to only allow negotiations from authorized domains. Nonces in the negotiation message and soft-state about the Nonces of recent messages can help to conquer replay attacks. Listing D.1 in Appendix D shows the specification of the Nonce in the negotiation message.

Attacks on an agreement negotiation system impair the establishment of new relations between services in different domains. However, Denial-of-Service attacks on stateful agreement negotiation systems are more devastating, because an attacker can exhaust the resources of a single resource more easily in contrast to a cluster of agreement negotiation services. If applicable, it is a safety precaution to only allow negotiations from authorized domains. The processing of negotiation messages is usually as resource expensive for the client as for the server. An attacker could try to repeatedly send the same message at a high rate. Nonces in the negotiation message and soft-state about recent messages can help to discard an abnormal amount of duplicate messages at the server.

5.4.2 Approach to Secure Bilateral Agreement Negotiation

Agreement negotiation is a challenge for services in distributed open environments. The agreement negotiation framework in this thesis relies on a requirements-driven method for establishing bilateral comprehensive agreements. Protocols for *stateless* agreement negotiation use messages which contain the whole Negotiation State. The drawback of stateless negotiation systems is that the complete Negotiation State is prone to forgery by the other party in the negotiation. It could remove preconditions, modify dependencies, and replace Proposals to obtain an agreement that favors the adversary. Stateless negotiation cannot be used if it does not address these vulnerabilities.
End-to-end security protocols (e.g. TLS, IPSEC) protect only against manipulations by malicious third parties. Only with an effective protection against forgery, stateless agreement negotiation protocols become feasible. The challenge is to distinguish legitimate modifications of the Negotiation State, which are in accordance with the negotiation protocol, from forgery of all other portions of the Negotiation State. The preconditions, dependencies and Proposals that have been exchanged in previous rounds must not be modified.

VersaNeg can be extended with a security extension to achieve secure stateless negotiations. The novel alternating signature protocol detects manipulations of the Negotiation State and introduces non-repudiation to agreements. Stateless agreement negotiation extended with the alternating signature protocol can be a viable alternative to stateful negotiation approaches especially for long lasting negotiations in unreliable environments.
The approach to secure bilateral agreement negotiation with comprehensive Negotiation States and stateless negotiator services is as follows:
A novel alternating signature protocol for the protection of XML negotiation messages to detect forgery of Negotiation States. The application of XML Signature [BBFL$^+$02] to sign portions of the messages for verifying the integrity of the negotiation. However, XML Signature alone is not sufficient, because it only protects the *old* parts of the Negotiation State against changes. It cannot determine if the *new* parts of the Negotiation State, which have been appended by the other party, comply with the specification of the negotiation protocol and are in accordance with the policies that steer the negotiation.
This novel security algorithm reverses the last negotiation step of the other party by first removing all legitimate changes that are in accordance with the negotiation protocol and by verifying that the remaining state has not been modified. A generic model of XML Negotiation States allows formalizing the protocol messages during stateless agreement negotiation. This Section uses this model to demonstrate how the alternating signature protocol extracts the parts of the Negotiation State that should have been immutable at the current iteration of the negotiation.
Different simulated attacks will show how the implementation of the alternating signature protocol detects manipulations. Experimental results from measurements with the prototype support a careful choice of cryptographic algorithms to improve the negotiation performance.

5.4.3 Model of XML Negotiation States

The Negotiation algorithm works through an exchange of XML documents that contain the complete Negotiation State (also called comprehensive Negotiation State). The endpoints do not need to preserve state across multiple iterations of the negotiation; instead they can reestablish all information about a negotiation by parsing the current negotiation message.
Documents with comprehensive Negotiation State grow at each iteration because the processing party appends its Requirements, Offers, and Obligations to the document. All data remains in the document until the negotiation terminates.

There are various ways for describing an XML document formally. One can define the data model and the operations on the data in the context of XML Negotiation States. An XML document is formed by a hierarchy of XML tags and can be represented by a directed graph $G_{negState}$. Adjacency matrices can be used to represent finite directed graphs. However, common definitions of adjacency matrices do not allow preserving the order of the children of a node, which is important for XML InfoSets and XML canonicalization.
This model uses a slightly different definition and maps the directed graph $G_{negState}$ that has n vertices to a matrix $M_d^{n\times n}$ where an entry $a_{i,j} > 0, a_{i,j} \in \mathbb{Q}^+, ij \in \mathbb{N}, i \leq n, j \leq n$ denotes the presence of one edge from the vertex v_i to v_j. The child $a_{i,x}$ of v_i precedes child $a_{i,y}$ if $a_{i,x} < a_{i,y}$. The node labeling function $\sigma_d : V \mapsto \sum$ provides values from the

alphabet $\sum$ for entity names, and $\gamma_d : V \mapsto \sum$ provides a set of attributes for a vertex and if present the XML data value. The whole XML Negotiation State is represented by the tuple $d =< r_d, M_d, \sigma_d, \gamma_d >$, where r_d is the root element.

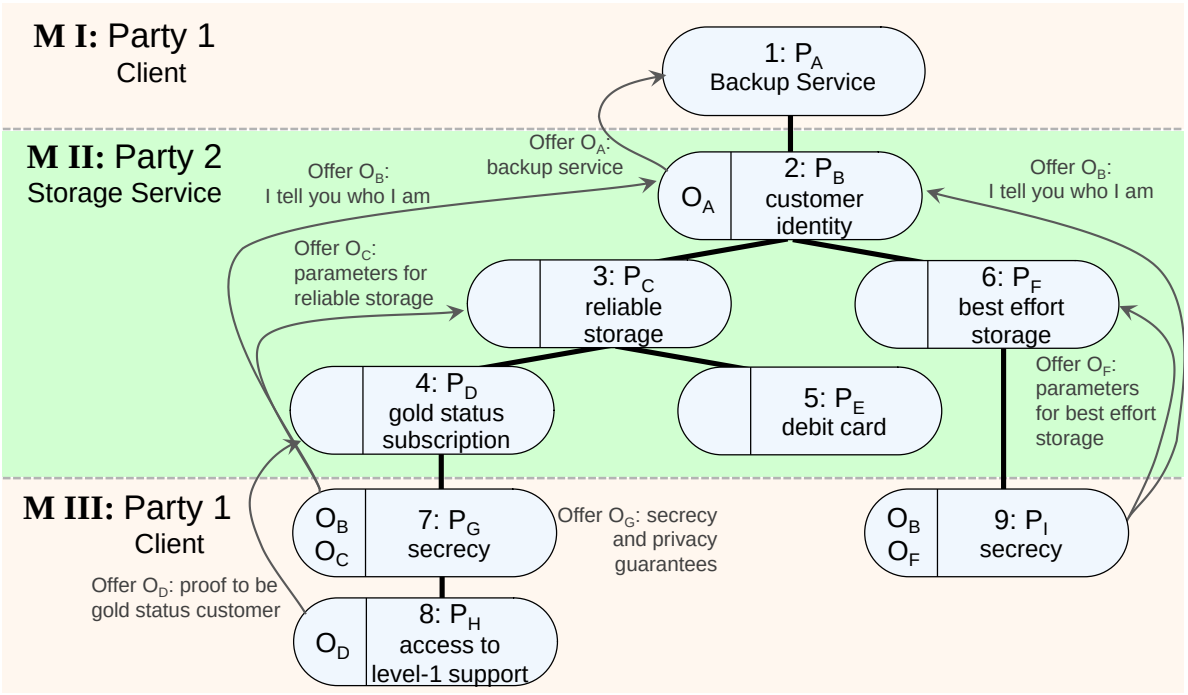

Figure 5.18: *Dependency Tree after three Messages, Nodes are Numbered in Sequence*

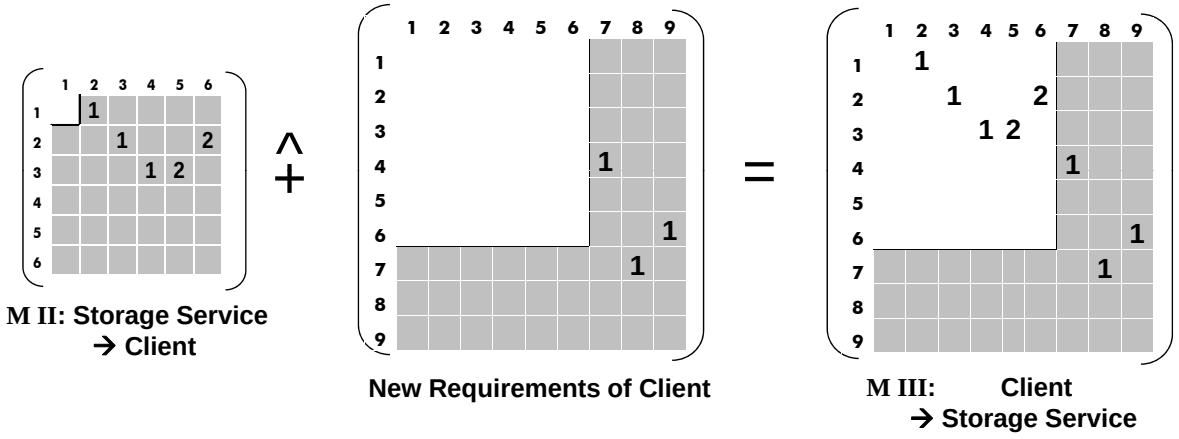

Figure 5.19: *Matrix Representation of Dependency Tree, Transition from Message in Step II to Message in Step III*

5.4.3.1 Agreement Discovery Phase

The Negotiation Phase is where each party states its Proposals. The negotiation starts when a Requester R sends a resource request to a Service S. The service has various Requirements regarding its Offer that would allow R to access the service. For example, it could require either a permanent subscription or the willingness to pay for the service. By processing these Proposals the requester learns about the terms under which this service is available.

The requester has probably its own Requirements concerning the service usage, for instance, quality of service and privacy. The requester now appends its Proposals to the Negotiation State.
The result is a tree structure (see Figure 5.18) of interdependent Proposals. Father-child relationships denote conjunctive conditions where all Proposals must be fulfilled. A sibling relationship denotes a disjunctive condition where only one condition must be satisfied.

The Agreement Discovery phase can be modeled as a sequence of matrix additions. We define a matrix addition between matrices with unequal dimensions $A^{n\times m}, B^{x\times y}, x \leq n, y \leq m$ in our context as follows:

$$(A^{n\times m}\hat{+}B^{x\times y})_{i,j} = \begin{cases} a_{i,j} + b_{i,j} & \text{for } \begin{matrix} a_{i,j} \in A, b_{i,j} \in B, \\ i \leq x \wedge j \leq y \end{matrix} \\ a_{i,j} & \text{for } \begin{matrix} a_{i,j} \in A, x < i \leq m \\ \vee y < j \leq n \end{matrix} \end{cases} \tag{5.2}$$

The requester at iteration j appends its new Proposals with the Requirements and Offers, described by matrix M_{j+1}^{req}, and creates the Negotiation State $M_{j+1} = M_j \hat{+} M_{j+1}^{req}$. This algorithm repeats for all $j \in \{1...n-1\}$, where n is the number of iterations during the negotiation phase. Figure 5.19 shows the matrix representation of the message M_{II} from Figure 5.18 and how the message M_{III} is constructed.

5.4.3.2 Obligation Disclosure Phase

The purpose of the Agreement Discovery phase is to discover one viable solution for both parties. However, it is still possible that one party does not deliver the promised Offers. The *Obligation Disclosure* phase ensures that one Obligation is released only, after all preconditions stated in its Requirements have been fulfilled.
The order of the Proposals in the path from leaf to the root in the negotiation tree has the property that it is a safe disclosure sequence. The Obligations $M_j^{obligations}$ are only released after all associated Requirements are satisfied.
The protocol will iteratively exchange the Obligations by appending them to the Negotiation State $M_{j+1} = M_j \hat{+} M_j^{obligations}$. The negotiation has reached the Complete Agreement state after all Requirements have been satisfied by corresponding Obligations.

5.4.4 Alternating Signatures and Message State Reduction

The algorithm with alternating signatures and message state reduction [KCRS09] verifies that previous messages have not been manipulated. The purpose of the algorithm is to ensure that the other party did only append to the Negotiation State and did not change any preexisting parts of the message.

Before a message M_i is sent, the current peer will apply an XML Signature [BBFL$^+$02] covering the whole Negotiation State. This signature includes the history of all Proposals exchanged up to that point. The peer will append the signature to the message and will send it to the opponent in the negotiation.

The negotiation messages have one `Integrity` element (see Schema definition in Listing D.1 in Appendix D). For bilateral negotiations, it contains two `Signature` elements,

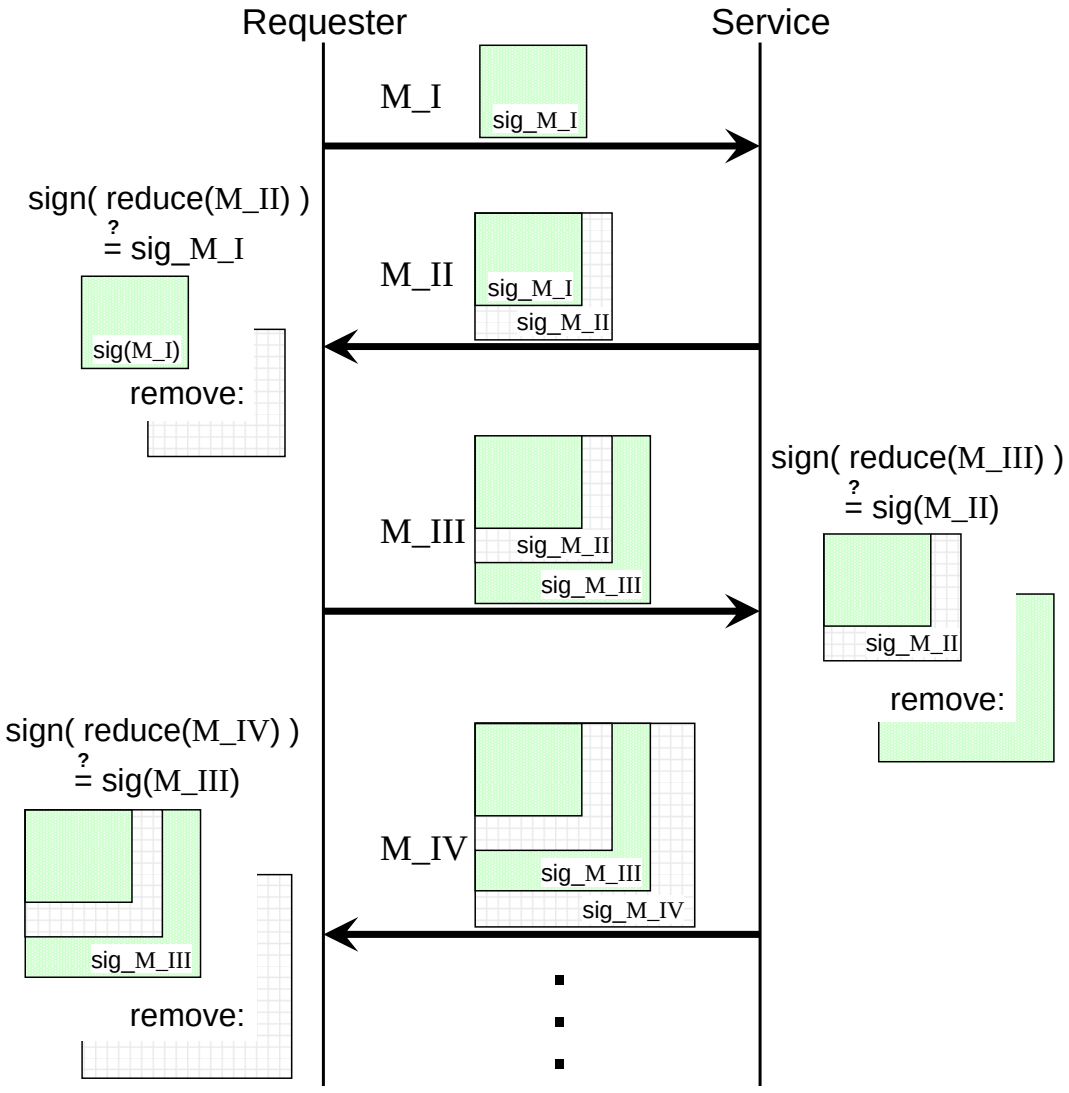

Figure 5.20: *Message Reduction and Verification with Alternating Signatures*

one for each party. Listing D.2 includes the additional elements, necessary to integrate XML Signature into the negotiation messages.

The opponent will process the message and send a reply M_{j+1} with an extended Negotiation State. When the first peer receives the answer to its original message, it must first verify the integrity of message M_{j+1}, before it can start processing.
Without the message state reduction, the signatures of the two messages do obviously not match $sign(M_{j+1}) \neq sign(M_j)$ because the message M_j has been extended with additional state, by appending Proposals.

Upon reception of the message, the receiving peer must remove all nodes from the XML document that the other party appended. In the model of XML Negotiation States, the dimensions of the matrix have grown to accommodate the new nodes. In terms of this model, one must reduce the dimension of the received message $M_{j+1}^{b \times b}$ to obtain the

representation of the original message $M_j'^{a\times a}$with $b > a$.
The matrix $I^{m\times n}$ can be defined as follows:

$$(I^{m\times n})_{i,j} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases} \text{ with } i \in 1,\ldots,m, j \in 1,\ldots,n \tag{5.3}$$

The matrix multiplication $M^{l\times m} \times I^{m\times n} = R^{l\times n}$ preserves elements that are within the new dimensions of $R^{l\times n}$. By using two matrix multiplications, we obtain the message $M_j'^{a\times a}$:

$$I^{a\times b} \cdot M_{j+1}^{b\times b} \cdot I^{b\times a} = M_j'^{a\times a} \tag{5.4}$$

The implementation of the protocol uses a *new nodes list* to indicate which new nodes have been appended to the document by the other party. One peer must first remove the new nodes before it can verify the signature. Be aware that the resulting document might still not fully match the original document because of different representations of equivalent documents due to syntactic freedoms of XML.
One XML standard is very helpful in obtaining a valid signature after the document has been modified two times: XML Canonicalization from [Boye01] ensures that XML documents, which can have varying representations, but are logically equivalent in a given application context, will be transformed in an unambiguous XML representation.

The digital signature of the reduced document must match the previous signature of this peer. The digital signatures of both peers are contained in the messages. The signatures will only match if the resulting document after the reduction of M_{j+1} is equivalent with the document M_j.
Non-matching signatures indicate that the previous Negotiation State has been tampered with and the negotiation fails. If the new nodes list is corrupted, the algorithm will remove an incorrect set of nodes and the signatures will also not match. After successful signature verification, the rest of the message is handled by the negotiation algorithm as described in Section 5.3.3. It ensures that the new parts of the message comply with the semantics of the negotiation protocol and the access control policies.
The Figure 5.20 shows how both negotiating parties apply this algorithm. The Requester includes $sig_{MI} = sign(M_I, key_{Requester})$ in its message M_I. When it receives M_{II} it removes the changes that the Service applied and verifies that $sig_{MI} = sign(reduce(M_{II}, changedNodes), key_{Requester})$.
The Service acts also according to this algorithm. The algorithm will be applied successively for each message until the negotiation ends. The complete history of Proposals and choices made is thereby protected.
If the signatures do not match for one message, the negotiation is terminated and the incident will be logged.

5.4.4.1 Different Options for Cryptographic Signatures

The VersaNeg framework uses the XML Signature standard [BBFL$^+$02] to protect the integrity of the complete Negotiation State and includes two signatures in the document: one for the Requester and one for the Service. One option is to use RSA [RiSA78] public key cryptography to calculate the digital signature of a Negotiation State before the

message is sent. Public-key cryptography allows any party that receives the plain XML negotiation message and that knows the public-key to verify the signature.

The use of RSA makes fail-over and load-balancing easier because other backup servers can verify the digital signature with the public key of the failed server. An XML firewall could integrate the logic of the alternating signature protocol to verify the signatures and drop manipulated messages. Such a firewall can only operate of course, if it has access to the unencrypted XML messages.
Another important property of using public key cryptography together with digital certificates is non-repudiation. Both parties sign the whole agreement successively by applying this algorithm. A third party can verify the integrity and the origin of the agreement by checking the signatures.

An alternative option is to use the keyed-hash message authentication code (HMAC) from [KrBC97]. HMAC uses a secret key and a cryptographic hash function to calculate the message authentication code of a Negotiation State. The secret keys in our protocol are exclusive for Service and Requester each. Only the peer that used a key for its signature must be able to verify the signature. If one negotiator wants to use load-balancing between multiple servers internally, the secret key must be shared with all other servers that can act as backup. However, compromise of one server would impair the security and trustworthiness of all other servers.

There is one attack that digital signatures do not protect against: Replay of previous negotiation messages. To counter this threat, the Negotiation State includes a nonce, a sequence number, and a timestamp to limit the number of negotiations the service accepts. A service stores the nonce and sequence number of each message it has processed already.
The negotiation service discards negotiation messages that have an unreasonable timestamp or which can be identified as a replay via nonce and sequence number.

5.4.5 Experimental Verification of Security

This Section serves two purposes: It demonstrates how the message state reduction and verify approach protects against a broad range of attacks from a malicious opponent in the negotiation and it shows the trade-offs between different cryptographic algorithms on the runtime performance of the protocol.

5.4.5.1 Simulation of a Malicious Adversary

This experiment assumes that a suitable cryptographic protocol for end-to-end security (e.g. TLS or IPSEC) is in place to ensure *confidentiality* and to protect against attacks by *malicious third parties*.
Hence, this security analysis focuses on the malicious opponent in the negotiation that forges Negotiation States. One party was augmented with a software module for each experiment, to simulate different attacks on the negotiation as depicted in Figure 5.18. In these experiments, the Negotiation State belonging to the service was manipulated by the Requester and vice versa. The following list shows the attacks that have been simulated and their impact on the agreement:

1) *Attack* Adversary removes nodes
a) *Test* Remove $P_{goldStatus}$ and $P_{debitCard}$
Impact Agreement to access service without paying or subscription
b) *Test* Remove Proposal $P_{reliable}$
Impact Agreement without reliability guarantees
2) *Attack* Adversary modifies node content
Test Modify $P_{debitCard}$
Impact Require less money to access service
3) *Attack* Adversary moves nodes
Test Swap $P_{reliable}$ and $P_{bestEffort}$
Impact Obtain service with reliability guarantees for free
4) *Attack* Adversary adds nodes
Test Sneak in additional Proposal with no Requirement for reliable storage $P_{reliableAndFree}$
Impact Obtain service with reliability guarantees for free
5) *Attack* Adversary manipulates new nodes list
a) *Test* Do not state all nodes that were added
Impact Exclude from processing by negotiation algorithm
b) *Test* Try to hide added $P_{reliableAndFree}$ from attack 4)
Impact Cover forgery of Negotiation States

The Negotiation State forgery of attacks 1),2),3) are detected by verifying the digital signatures of the document portions that should have been immutable.
The attacks 4) and 5) cannot be detected by XML Security alone, but require the message state reduction that obeys the protocol specification. All nodes that follow the protocol rules will be removed from the document before the signature is verified. Any XML node that does not comply with the algorithm will stay in the test document and will lead to non-matching signatures.
Thwarting the attack 5b) also relies on the negotiation algorithm that enforces that nodes in the changed nodes list belong to the remote party. The new Proposal $P_{reliableAndFree}$ would remain in the document during message state reduction, because it is missing in the new nodes list. As a consequence, the signature would fail.

5.4.5.2 Performance Evaluation

It is important to understand the performance impact of using different cryptographic algorithms on the protocol performance to choose an appropriate cryptographic algorithm. Alternating signatures with message state reduction protects the integrity and authenticity of messages. It does not protect confidentiality. HTTPS/TLS or IPSEC assure confidentiality. Because the performance of HTTPS/TLS and IPSEC [KNMC06] is well understood, this section focuses on the evaluation of the alternating signatures with message state reduction algorithm.

As stated previously, HMAC is expected to perform better than RSA. RSA on the other hand, relies on public key cryptography which allows each party to proof the data origin of the signed negotiation message to a third party. This property is important if such a document becomes part of a legal conflict where the involved parties must present a proof of the electronic agreement in front of court. The question is how much security costs at all and what the price of public key cryptography is.
There are related publications [vEZh08, MTIN04] on the performance of WS-Security

with HMAC and RSA. Engelen et al [vEZh08] use gSOAP with openssl [3]. In their experiments with small messages, HMAC needs less than 5% of the CPU time of RSA with SHA1. The underlying HTTPS with TLS needs nearly the same time as the HMAC processing.

This section presents experiments that show how the security impacts the duration of the negotiation and the negotiation rate a server can handle.
One algorithm under evaluation, was the keyed-hash message authentication code with a secret key (HMAC) from [KrBC97]. The secret keys used with symmetric cryptographic algorithms must not be shared. No remote party needs to verify the signature of another party, because one party verifies if the other party modified the document, by checking its own signature.
Public key cryptography with the RSA [RiSA78] algorithm was also part of the experiments.

The measurements throughout the experiments were performed with the same setup as described in Section 5.3.5. It consisted of 2 PC that served as negotiation initiator and negotiation responder. The third PC was in control of the experiment and issued scripted commands to the two negotiators. The libxml2 (http://www.xmlsoft.org) library was used for parsing the XML document into a DOM representation and for applying Canonical XML [Boye01] to obtain a canonical XML representation.
Negotiation initiator and negotiation responder applied XML Signature with the XMLSec library [4]. The XMLSec library relies also on the cryptographic routines from openssl.

The first experiment measured the time for a complete negotiation with different cryptographic algorithms. The duration of a single negotiation is the time between the first message that has been sent and the last complete negotiation message received by the negotiation initiator. The time between each test was more than 3 seconds and was big enough to avoid any overlap between negotiations. Each test was run 120 times for each cryptographic algorithm.

Figure 5.21 shows the performance of only the security algorithm, protected by HMAC and RSA. The time for the security algorithm is the time for verifying the signature of a received message and the time for applying a signature to an outgoing message. The processing for the alternating signature with message state reduction algorithm happens for the verification of received messages.
The Figure contains error bars for the 1% and 99% quantile. The box is from the 5% quantile to the 95% quantile and shows the median as a black line. The time spent for the cryptographic algorithms had little variantions, with some outliers that took more time. The HMAC protected negotiations need 0.022 seconds for the alternating signature with message state reduction. In contrast, the same algorithm with RSA takes circa 0.031 seconds.

The Table 5.3 shows the time for the complete negotiation together with the time that is spent inside the alternating signatures and message state reduction algorithm. One can see that the time for complete negotiations that relies on RSA is around 0.096 seconds, whereas the time for the HMAC variants is circa 0.084 seconds. These numbers can be put into perspective with the numbers from the experiments without security, shown in

[3] http://www.openssl.org
[4] http://www.aleksey.com/xmlsec/

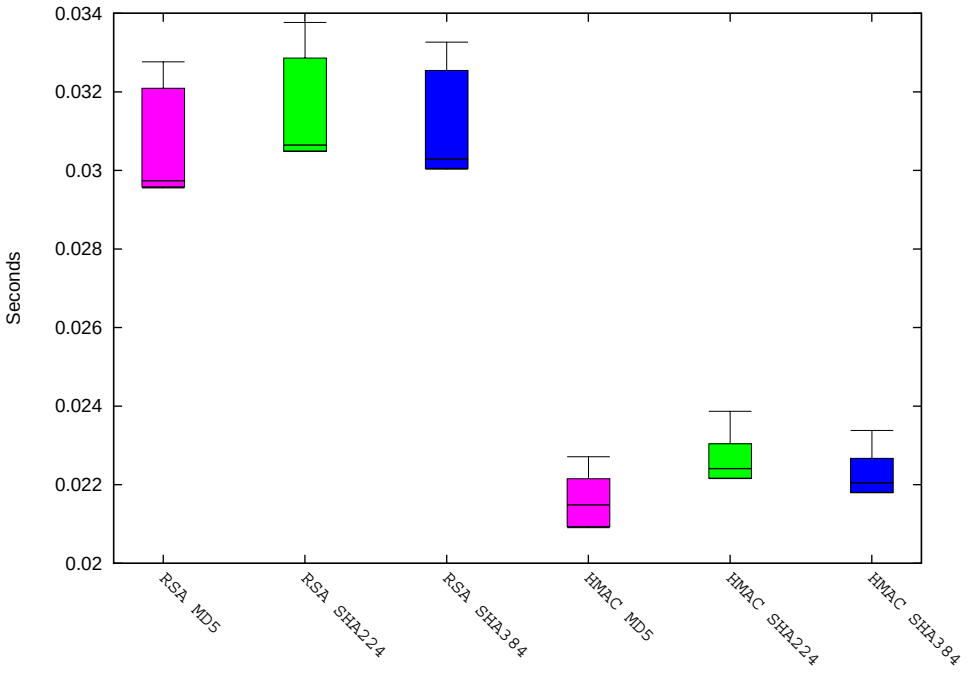

Figure 5.21: *Time for Alternating Signatures with Message State Reduction for Cryptographic Algorithms*

	Security only				Compl. Neg.
	Mean	5%Qt.	Median	95%Qt.	Mean
RSA MD5	0.0298	0.0296	0.0297	0.032	0.0946
RSA SHA224	0.0310	0.0304	0.0306	0.0328	0.0963
RSA SHA384	0.0313	0.0301	0.0303	0.0325	0.0957
HMAC MD5	0.0219	0.0209	0.0214	0.0222	0.0821
HMAC SHA224	0.0226	0.0222	0.0224	0.0230	0.0845
HMAC SHA384	0.0223	0.0218	0.0220	0.0227	0.0839

Table 5.3: *Time (sec) for Alternating Signatures with Message State Reduction for Cryptographic Algorithms and for Complete Negotiation*

Table 5.2. The same experiment without security takes 0.046 seconds. Security takes nearly the double time of a non-secure negotiation. That means that the overhead introduced by the security is more than the creation and verification of digital signatures during the alternating signature with message state reduction algorithm.

The Figure 5.22 shows the breakdown of different activities that are performed during the negotiation. The bars show the median and the error bars show the 5% and 95% quantiles. Processing is again the largest part followed by the time for communication and the time for the DOM XML parser. The time for the security can be divided into the time for applying the digital signature, the time for the message state reduction and the time for the verification of the signature. The message state reduction takes nearly 0.015 seconds for this experiment. The message state reduction has to copy the DOM tree, remove all previously applied changes, and transform the current XML negotia-

tion message for the digital signature validation. This effort is the same, regardeless which cryptogarphic algorithm is applied for the digital signature. The results show how HMAC is faster both for signing and verifying digital signatures. However, it is interesting to note that the signature validation with RSA is faster than RSA signature creation, whereas HMAC has the opposite behavior. The penalty for using RSA over HMAC is small in comparison with the complete duration of a secure negotiation, making RSA the preferable option due to its functional benefits.
Another observation is that security also increases the time for processing, XML DOM parser, and for communication. This can be explained, because the Negotiation States grow by integrating the XML Digital Signature element for both parties. The first message grows by 1.3kB for including one digital signature for security. All other messages carry 2.5kB for two digital signatures. The security overhead for this negotiation is a total of 18.4kB that must be parsed and handled by the negotiation server which explains why the effort for the other non-security related activities grows.

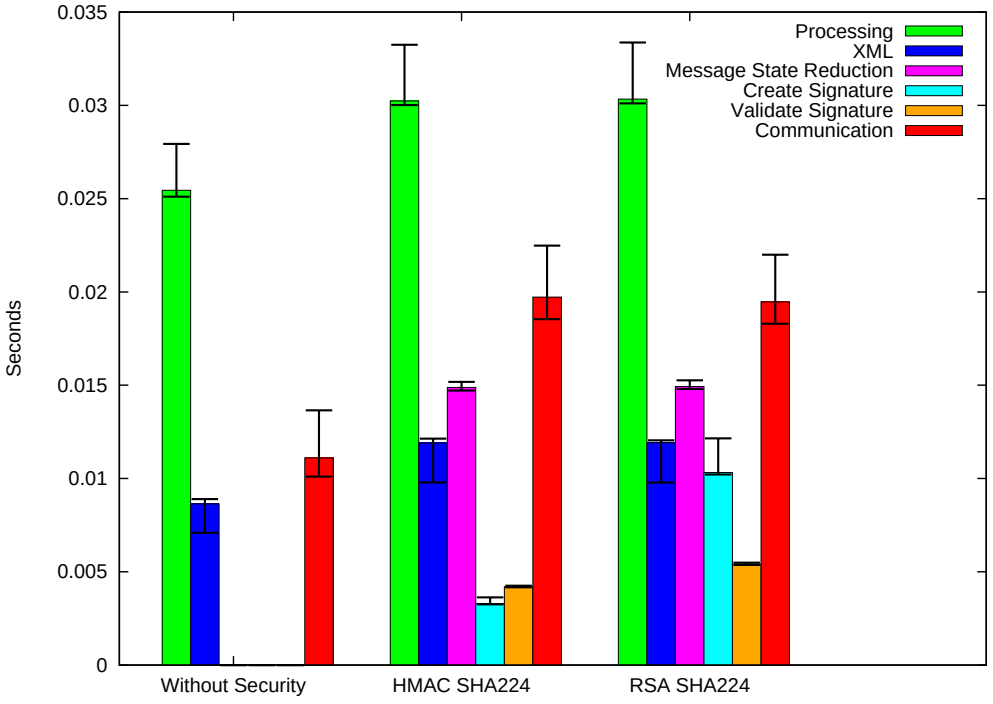

Figure 5.22: *Breakdown of Time spent for different Activities during Negotiation*

The next experiment determined the impact of message sizes on cryptography and scalability. The experiment used different Obligations to produce Negotiation States with growing sizes. Each combination of negotiation size and cryptographic algorithm was measured for 300 seconds.
The negotiation rate is the number of complete negotiations per time interval. The size of all messages in a negotiation was summed up to determine the impact of the message size on the effective negotiation rate. The symmetric nature of the protocol implies that the computational effort of negotiation initiator and negotiation responder is roughly equal. The message to start the negotiation is cheaper which allows the negotiation initiator to easily overload the negotiation responder.
Figure 5.23 shows that the size of the negotiation messages determines the processing capacity. Each point is one test run in the experiment and includes the 5% and 95%

quantiles as error bars. The use of security has a significant impact on the scalability of the negotiation systems. The achievable negotiation rate with security is nearly half the negotiation rate without security. The secure negotiation with HMAC reaches a rate of 22.1 negotiations per second (176 messages per second) and the RSA reaches 19.3 negotiations per second for an aggregated negotiation size of 100kB. The difference between HMAC SHA224 and RSA SHA224 is rather small. HMAC achieves a 20% better negotiation rate than RSA for small messages. The performance of HMAC SHA224 and RSA SHA224 converges for large messages.

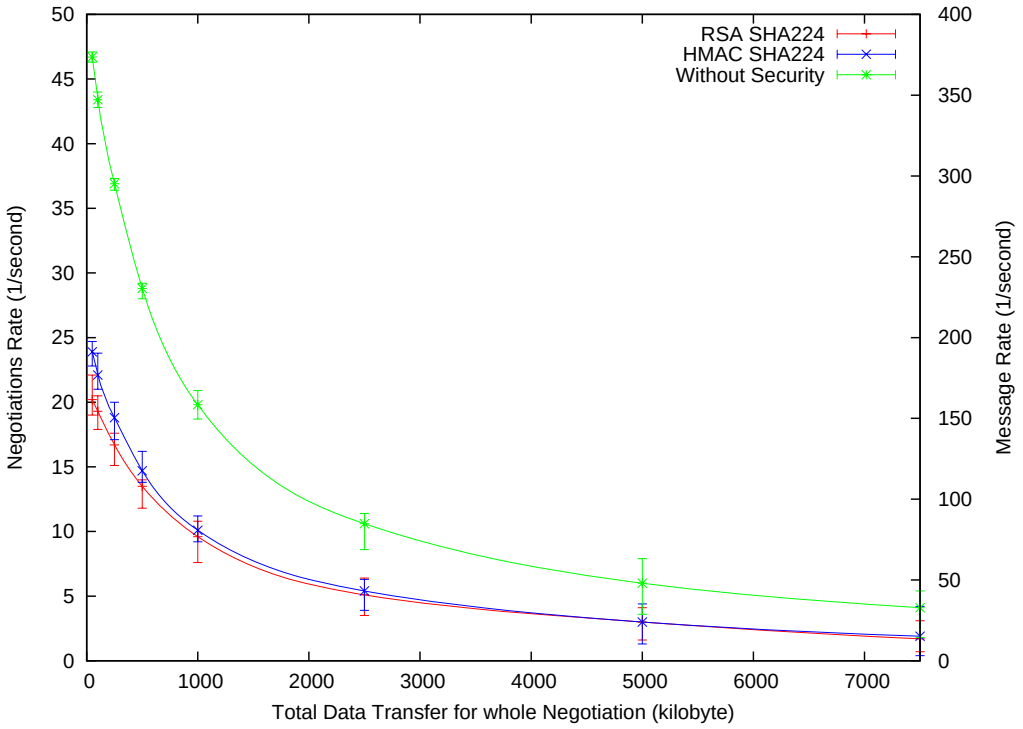

Figure 5.23: *Size dependent Negotiation Rate for Cryptographic Algorithms*

How does the cryptographic hash algorithm influence the negotiation performance? The same experiment was also conducted for RSA MD5 and HMAC MD5. However, the MD5 cryptographic hash "should be considered cryptographically broken and unsuitable for further use"[5]. Nevertheless it is one of the fastest cryptographic hash algorithms, which is used in many publications as a reference. Therefore, it makes sense to perform the experiments with MD5 to determine to what degree a faster cryptographic hash algorithm improves the achievable negotiation rate and because it allows to put the results into perspective with related work.

The Figure 5.24 shows the results of SHA224 and MD5 together. For negotiations that are smaller than 100kB, the choice between HMAC and RSA is the determining factor. The performance difference of applying SHA224 and MD5 is negligible. Negotiation with an aggregated sizes of bigger than 5000kB lead to the opposite behavior. The performance is determined by SHA224/MD5 and the HMAC/RSA algorithm make only a small difference. The point where RSA MD5 has the same performance as HMAC SHA224 is around 1200kB.

[5]US-CERT Vulnerability Note VU#836068: http://www.kb.cert.org/vuls/id/836068

This behavior is to be expected as the SHA224 and MD5 are applied to the whole message and are therefore size dependent. The RSA and HMAC algorithm are only applied to the XML Signature container that contains the digest that was calculated by the cryptographic hash. Therefore, the RSA and HMAC are always used for a constant sizes, whereas the cryptographic hash algorithm is size dependent.
For negotiations smaller than 1200kB the use of HMAC can bring a small performance benefit. For larger negotiations, RSA is the preferable option due to its enhanced security features. The MD5 algorithm would be faster for larger negotiations, but the use of MD5 is discouraged.

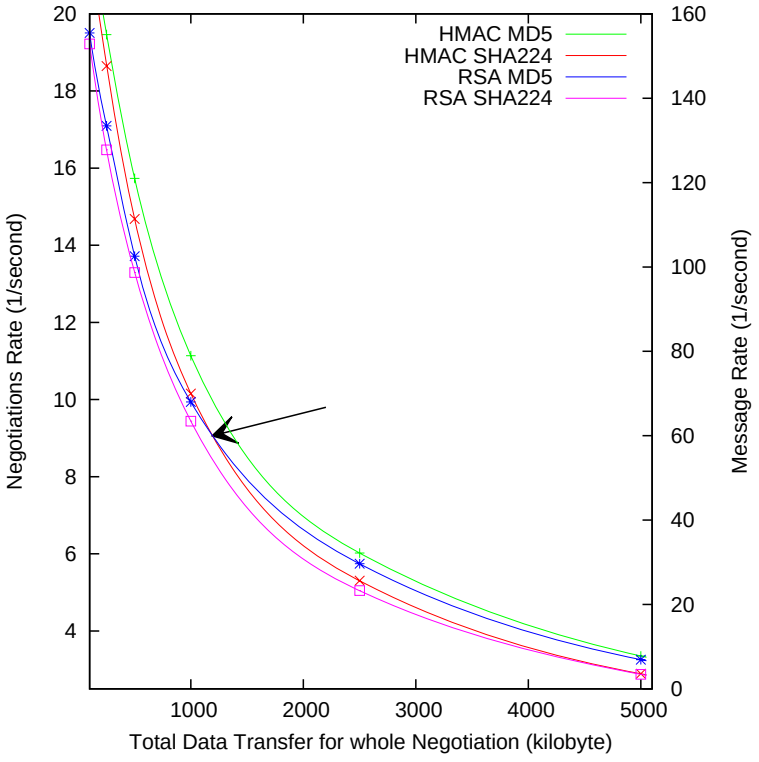

Figure 5.24: *Size dependent Negotiation Rate for RSA/HMAC with SHA224/MD5*

The results also show that the choice of the cryptographic hash function has a small impact on performance. The SHA224 algorithm produces a SHA256 hash that is truncated internally. Therefore SHA224 and SHA256 performance is expected to be nearly identical. It is interesting to put the results into perspective with the highly tuned IPSEC AH [KNMC06] on a similar testbed with older Intel Pentium IV PCs. These IPSEC AH experiments allowed for an MD5 throughput of circa 70MB/sec and a SHA256 throughput of circa 23MB/sec. In contrast, the highest throughput of alternating signatures with message state reduction was circa 20MB/sec for MD5 and 11MB/sec for SHA224 on more capable PCs. Hence, the performance of the cryptographic hash functions is still important for the throughput of negotiations, but the additional message state reduction tasks and the general negotiation logic are even more expensive.

The obtained results show that negotiations with the private key HMAC are faster than RSA with public-key cryptography for small negotiation sizes. The drawback of

utilizing the HMAC based algorithm is that it does not allow for non-repudiation, as the same secret key is used for signing and verifying the messages.
Future work includes the examination of a hybrid strategy that uses HMAC for the negotiation and public-key cryptography only for signing the final agreement. Incremental signatures as presented in [BeGG94] are also a promising direction for improving the negotiation performance, as large parts of the negotiation message are immutable.

5.4.6 Discussion of Alternating Signatures and Message State Reduction

Stateless agreement negotiation is well suited for long-lasting negotiations and negotiations in unreliable environments. However, the stateless design of the negotiation systems introduces additional security challenges. This Section presented a secure stateless agreement negotiation.
Traditional end-to-end security can establish communication channels to guarantee the confidentiality, integrity and authenticity of the negotiation in the presence of malicious third parties. It does not protect against malicious negotiators that manipulate the Negotiation State.
The novel alternating signatures and message state reduction algorithm protects the integrity of XML Negotiation States against forgery by malicious third parties. Different simulated attacks were carried out to demonstrate how the algorithm detects forgery of Negotiation States. The experimental results showed the performance impact of various XML security algorithms. Whilst this Section described how alternating signatures protect the novel stateless agreement negotiation algorithm of this thesis, this approach can be extended to other stateless XML negotiation systems as well.
Agreements under VersaNeg can constitute legally binding contracts. In this case, it is important to have the property of non-repudiation to be able to proof the outcome of the negotiation to third parties. To achieve this goal, one has to create legally binding electronic signatures. The alternating signatures and message state reduction algorithm allows for the integration of electronic signatures, refer to Section 2.1.4 for the legal background. Advanced electronic signature can be used to sign each individual message in the negotiation, for instance, by using public key cryptography. The court has some freedom to decide if advanced electronic signatures are legally binding. Qualified electronic signatures have the same credibility in front of court as real-world signatures on paper. However, these signatures must be applied by a human during the negotiation with a secure signature-creation device. Therefore, qualified signatures shall rather complement public-key cryptography in certain phases. For instance, the parties might sign the Negotiation State with a qualified electronic signature, either when entering the Decision Pending state, or when entering Agreement Complete state. This would allow each party to proof the agreement in front of court and enjoy the same level of trust in the signature as with written contracts.

5.5 Summary

Bilateral agreement negotiation with *VersaNeg* is attractive for scenarios where the potential dependencies between two parties in an agreement are unknown beforehand. The agreement negotiation is the tool for establishing these dependencies.
The bilateral agreement negotiation framework relies on comprehensive Negotiation States that contain all information that has been exchanged, to conclude the agreement. Thus, these comprehensive Negotiation States are well suited to serve as legally binding contracts for E-Contracting scenarios because they fully define the result of the agreement negotiation. All obligations, permissions, prohibitions of an agreement and the dependencies between those statements are preserved by the agreement.
The comprehensive Negotiation States greatly facilitate linear scalability by adding standard hardware and also help to improve failure resilience through easy fail-over handling. Comprehensive Negotiation States reduce the runtime state of negotiation servers for long lasting negotiations, for instance, when humans are involved for making decisions. The alternating signatures with message state reduction not only protects the integrity of the Negotiation States, but also introduces non-repudiation for agreements. Non-repudiation is an important property to proof agreements in front of court.
Bilateral agreement negotiation with *VersaNeg* has an advantage over related approaches because it can discover initially unforeseen agreements. Competing standards require extensive a-priori knowledge about the expected agreement, which drastically impedes the capability of these protocols to discover initially unforeseen agreements. The capability of *VersaNeg* to perform iterative negotiations based on an exchange of Requirements and Offers facilitates the discovery of previously unknown agreement options.

The whole is more than
the sum of its parts.

Aristotle, Metaphysica

6. Multilateral Agreement Negotiation

Inter-domain collaborations between multiple parties suffer not only from technological obstacles that hinder interoperability, but also from diverting business objectives of the involved domains. Nowadays, hand-crafted contracts define the terms and conditions for service interactions, but do not scale for dynamic scenarios where service providers join ad hoc services collaborations. Electronic negotiation can serve as the enabler of inter-domain collaborations by providing a large degree of freedom for the automation of agreement formation and electronic contracting.

Bilateral agreement negotiation is a powerful tool to discover agreement options between a pair of parties and to form a bilateral agreement. However, bilateral agreement negotiation is limited in collaborative scenarios, where a number of distinct parties interact to reach their goals through cooperation. The formation of complex collaborations with bilateral agreements is limited, because dependencies that go beyond a single bilateral agreement can hardly be captured.
Parties that rely on bilateral agreements, face difficulties to assess the risk of collaborations. The negotiators find it hard to obtain a comprehensive overview of all dependencies involved in a collaboration. Without the knowledge about critical dependencies, no accurate risk assessment is possible.

Multilateral agreements are capable of capturing complex dependencies between a group of parties to foster collaboration. They make complex dependencies visible and allow for an accurate rating of different alternative collaboration options.
Multilateral agreement negotiation also improves control for all involved parties. Only the individual parties decide to make Proposals or to refrain from Proposals. The distributed negotiation protocol allows the parties to behave according to their strategic interests.
Iterative requirements-driven multilateral agreement negotiation is a tool for discovering the unknown. New and unforeseen agreement options can be discovered at runtime. Multilateral negotiation is more apt to shift tasks and responsibilities between the involved parties in comparison with a set of bilateral negotiations that already assume a division of labor before the negotiation even starts. This flexible division of tasks is important to achieve higher overall efficiency of complex collaborations. With multilateral negotiation, the tasks are divided during the agreement discovery process, leading to

agreements that can better adjust to the capabilities and offers of the involved parties. Multilateral agreement negotiation leads to a higher volume of negotiations and increases the competition to find effective solutions at an attractive price.

This Chapter extends the *VersaNeg* framework to support complex multi-party agreement discovery and formation. With few extensions to the negotiation language and the underlying negotiation model, the *VersaNeg* framework is capable of performing iterative multilateral agreement negotiations.
The multilateral agreement negotiation allows for the discovery of collaborations and for the formation of multi-party agreements. The protocol works through an iterative exchange of Requirements and Offers. A major benefit over existing bilateral negotiation protocols is that *VersaNeg* is capable of discovering potential collaborations between different parties to achieve common goals and that it will leave each negotiating party with a complete agreement document after a successful negotiation. This comprehensive agreement document defines the interdependent Obligations between all parties and is due to its similarity with real world contracts well suited for E-Contracting. An important property of the protocol is the protection of sensitive information that belongs to agreement options that will not become part of the final agreement. This feature is particularly important when agreement negotiation leads to a high number of possible agreement options, where only a small subset will eventually become a complete agreement.

Figure 6.1 shows the activities of multilateral negotiation that go beyond the activities for bilateral negotiation, as introduced previously in Chapter 5 with Figure 5.1. Multilateral agreement negotiation deals with groups of parties. The agreement negotiation should also be flexible with regard to the parties that take part in a multilateral negotiation. The business process must supply the framework with potential partners. The agreement negotiation must be flexible enough to negotiate with unforeseen partners. Either the partners in a negotiation are established as a fixed set during the Information Phase, or the agreement negotiation may be extended dynamically with additional partners. In the latter case the negotiation framework requests additional potential collaborators from the business process and adds these dynamically to the negotiation. The Negotiation Intelligence must also be more capable to deal with complex multilateral agreements. It must be able to assess the risk of different agreement options and it must learn from past negotiations. Potential approaches to for realizing a powerful Negotiation Intelligence are complex [BiDK89, XiLS04] and the application of these techniques is future work that goes beyond this thesis.

Security is another important focus of this Chapter. The alternating signature algorithm with message state reduction can be extended to multilateral negotiation to protect against message forgery. Besides the protection of the message integrity, multilateral negotiation introduces additional security challenges. Information in the negotiation state shall not be equally accessible to all parties.

The first part of the Chapter explains the extensions to *VersaNeg* for multilateral negotiations. The second part of the Chapter has a focus on the security related functionalities.

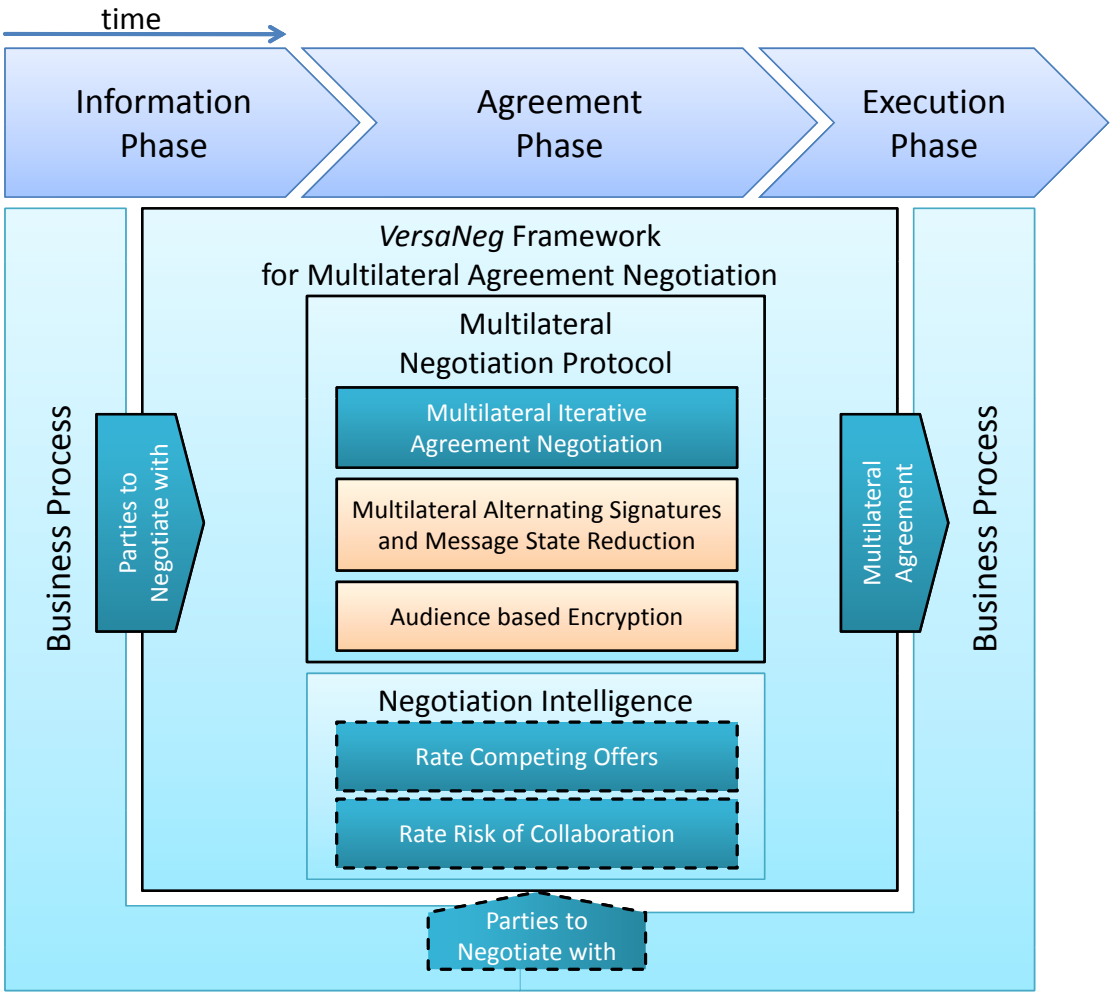

Figure 6.1: *Specific Activities for Multilateral Agreement Negotiation*

6.1 Alternative Approaches to Multilateral Negotiation

The papers presented on agreement negotiation in Section 5.1 dealt only with bilateral negotiations. Work on multilateral negotiations comes from different research directions.

Griffel et al. [GTMM+97] were among the first to introduce the concept of multi-party E-Negotiation. They already introduced a framework for multi-party negotiation and presented the idea of exchanging contract data objects among the group of contractors to build a common contract. However, they do not specify any details of the negotiation protocol.

WS-Agreement [ACDK+07] and Job Submission Description Language [ABDF+05] are current standards that allow for the formation and management of bilateral agreements. There exists work on WS-Agreement for modify existing agreements through re-negotiation [DMRTV07]. However, this standard cannot discover agreements that are outside the predefined agreement template. Policy intersection and policy merge with WS-Policy or ebXML CPA requires that the structures of the policies match to a large degree. The template-based approach of WS-Agreement works well for bilateral negotiations, but is a dead end for multi-party agreement negotiation. It is infeasible to capture all possible interrelations and dependencies between different parties and

formulate one single multi-party agreement template beforehand.

Zhang and Winslett introduce distributed authorization by multi-party trust negotiation in [ZhWi08]. Their approach evolves around a careful exchange of credentials to establish trust between different parties that protect credentials and resources with so called disclosure policies. Their approach is similar to ours with respect to the protection of sensitive information but differs because it cannot form comprehensive agreements and consequently does not address E-Contracting scenarios.
The approach presented in [CWZL05] relies on an asynchronous weak-commitment search for solving a constraint satisfaction problem to coordinate services collaborations. This method does not yield agreements but relies on informal local decisions to collaborate on tasks with other agents. The different agreement options are highly distributed and no entity has a complete overview of the decisions made in the system. The system cannot produce one comprehensive contract that defines the collaboration.
Kim and Segev argued in [KiSe03] that negotiation is often interleaved with business processes and proposed a framework that allows for the coordination of concurrent auctions. They support generic requirements but do not account for any dependencies between requirements of different parties.
The paper in [ZMFA04] presents a multi-party agreement protocol that relies on majority rule. However, the multi-party agreement consists of a set of bilateral agreements and does not capture complex dependencies between multiple parties.

According to Weigand and Xu [WeXu03] a "contract is an agreement between two or more parties that is binding on those parties and that is based on mutual commitments". The formal construct of conditional commitments are the basis of a number of publications.
Execution and modeling of multi-party contracts with commitments was the focus of Xu et al. [XuJG05]. They provide algorithms for detecting violations of the agreement and for identifying violators. They do not give an algorithm for how to obtain agreements through negotiation, but presented a valuable technique for the Agreement Execution phase.
The paper in [Wini07] deals with the coordination of agents through conditional commitments. Wan and Singh also present a commitment based formalization of conditional multi-party agreements in [WaSi05] and propose different algorithms for deadlock handling. Bargatti et al. [BBMM+07] apply a decentralized two phase commit protocol to reach consensus between distributed parties on an agreement.
The presented commitment based approaches have in common that they assume the presence of some sort of agreement containing the commitments, but do not specify how such an agreement is formed. Nevertheless the idea of mutual commitments as the foundation of contracts is highly relevant to this thesis.

6.2 Contributions to Multilateral Agreement Negotiation

The novel multilateral agreement negotiation protocol works through a few extensions to the bilateral agreement negotiation protocol. The novel contributions of this Chapter include the following areas:

1. Protocol, system and framework for multilateral agreement negotiation. With only a few extensions to the bilateral agreement negotiation protocol, presented in Chapter 5, complex multilateral agreement negotiations become feasible.

2. The dependencies between collaborating parties are an important factor for the risk that a collaboration does not produce the desired results. The protocol makes these dependencies visible and allows for a better risk assessment of collaborations.

3. A generic negotiation protocol for multi-party agreement negotiation with ring traversal. This protocol discovers different options to reach an agreement between all parties and eventually turns one agreement option into commitments. Successful negotiations automatically lead to a comprehensive agreement document. Such an agreement document contains all details and relationships between the stated requirements and promises made during the negotiation.

4. Performance evaluations with the prototype, a generic performance model and analytical results show the scalability and the feasibility of the VersaNeg protocol for real world scenarios.

6.3 Multilateral Iterative Agreement Negotiation with Comprehensive Negotiation States

This Section first motivates the approach to multilateral agreement negotiation before it introduces the extensions to VersaNeg to allow for iterative multi-party agreement negotiation. Experiments and an analytical evaluation of the protocol behavior conclude this Section.

6.3.1 Approach to Multilateral Agreement Negotiation

Comprehensive negotiation states as presented in Chapter 5 are a prerequisite to realize multilateral agreement negotiation with VersaNeg. The statements made during the negotiations not only address one party, but must reach a group of parties instead. Comprehensive negotiation states retain all information during the negotiation and are a natural choice for realizing agreement negotiation.
An extension to bilateral agreement negotiation must include the following features:

1. Suitable communication pattern to convey statements and information during the negotiation to a group of parties.
2. Agreement Discovery and Agreement Formation must capture interdependencies between the Proposals of the involved parties.
3. Agreement negotiation must facilitate collaboration between parties to satisfy one Requirement jointly.
4. Agreements must include all dependencies between the statements made during the negotiation, to interpret the agreement as a legally binding contract.
5. Protection of the integrity of negotiation states during the group communication.
6. Confidential information within a negotiation state must only be accessible to a defined audience.

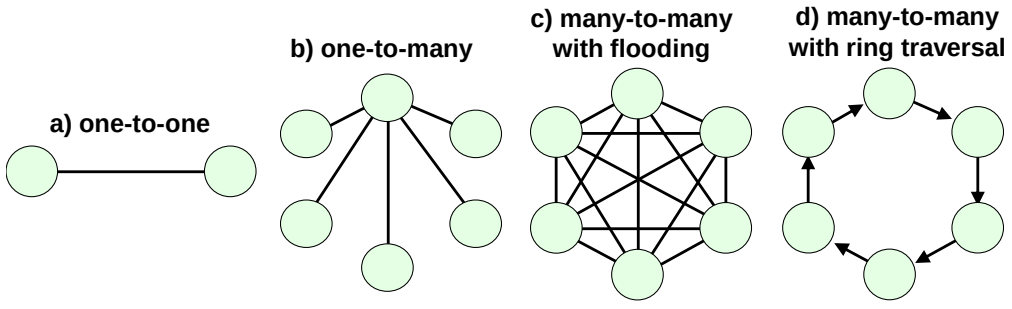

Figure 6.2: *Negotiation Communication Patterns*

6.3.1.1 Traditional Communication Patterns

Basic agreement negotiation is *one-to-one* and takes place between solely two parties (see Figure 6.2a). This pattern is found in many communication protocols that rely on policy intersection (see Chapter 4). FIPA [FIPA02] is one standard that defines simple primitives for contractual negotiations between two agents, especially about price.

Frameworks relying on the WS-Agreement [ACDK+07] standard can reach complex bilateral agreements. It relies on XML templates for defining the Requirements for resource consumption, corresponding guarantees, and even penalties if certain guarantees are violated.

The *one-to-many* communication pattern (see Figure 6.2b) relies on a special party that coordinates between concurrent agreement negotiations. This type of communication is the underlying pattern of auctions which have been extensively studied in the past [KeNT00]. Many agreement negotiations following the *one-to-one* communication pattern can be extended to *one-to-many* by introducing a capable coordinator, as demonstrated for instance in [KiSe03]. These *one-to-many* communications are popular to negotiate with a number of competing parties about a set of simple issues. Usually, only one agreement will result from this negotiation.
This scheme can easily be extended to establish a set of distinct bilateral agreements that together form the multi-party agreement [ZMFA04]. This approach has drawbacks for multilateral negotiation. All information passes through the central negotiator, which must be trustworthy. Besides reliability problems, due to the single point of failure in *one-to-many* schemes, the ability of the coordinator to understand and act upon information received in agreement messages, limits the construction of agreements with multiple dependencies between the parties. Relationships between the bilateral agreements may be unclear or completely missing.

The *many-to-many with flooding* communication pattern (see Figure 6.2c) allows for direct negotiations between all parties and removes the need for a central coordinator. One party starts the negotiation by sending a set of initial requirements to the other parties which in turn react with new requirements. Each party directly sends its requirements and promises to each potential recipient, for instance, as in asynchronous weak-commitment search [CWZL05]. The result is a distributed agreement where each party is responsible for keeping track of its requirements and promises.
One drawback of this method is that it is difficult to obtain a coherent view of an agreement to account for all inter-party dependencies which are scattered across the network. Risk analysis of the dependencies is difficult. This is a challenge for E-Contracting where these dependencies become relevant when the agreement is violated. It would be difficult then to identify which party did not fulfill its obligations and whom to demand compensation from. Even in multi-party consensus building, *many-to-many with flooding* becomes difficult to supervise and debug, because the state is highly distributed.

6.3.1.2 Many-to-many with Ring Traversal

To address these limitations, the new communication pattern *many-to-many with ring traversal* (see Figure 6.2d) relies on comprehensive negotiation states. Every information a party adds, stays in the negotiation state for the remainder of the negotiation. The immutability of negotiation states is later on important to integrate the security. All negotiation messages traverse the ring of parties in a predefined order. Parties are only allowed to add information to a negotiation state. The message with an updated negotiation state traverses the ring for at least one additional round to give each party the chance to respond to the new state. Each party thereby learns about Requirements from the other parties and can make Offers and may state own Requirements in turn.

The protocol automatically leads to comprehensive agreements. We call a negotiation message *comprehensive*, if all Requirements, Offers, and Obligations that have been part of the agreement negotiation up to this point in time, are now also contained in the representation of the negotiation message.

The *many-to-many with ring traversal* has several advantages in comparison to the above mentioned communication patterns. It introduces comprehensive agreement documents that are well suited for E-Contracting, but also enables a straightforward supervision and troubleshooting of negotiations. This pattern significantly reduces the number of messages during a negotiation in contrast to *many-to-many with flooding* and avoids a coordinator that could become a bottleneck.

Load balancing is also much easier with comprehensive negotiation states. A party can rely on the state in the negotiation messages and operates locally with soft state only. This property allows for a linear scalability of negotiation capacity by adding new servers. Negotiation servers need almost no state, which is desirable if humans are involved in the Decision Making process and agreement negotiations last for a comparable long time.

Failure Handling

Failure handling relies on two techniques. If a server goes down and a TCP connection timeout occurs, fail-over works by simply redirecting the message to a working negotiation service of the same party. The negotiation continues normally then.
More challenging is the case when a negotiating party fails completely. A negotiation timeout triggers a recovery protocol. One party, for instance, the initiator of the negotiation, is chosen to recover the negotiation. This party obtains the latest negotiation state from all remaining parties. It can thereby determine the most current negotiation state and decides if the negotiation should be aborted or the negotiation can continue without the failed party. The parties involved in an aborted negotiation are not obliged to the negotiation anymore. An aborted negotiation can be restarted without the failed party. It is advisable to integrate some kind of reputation system to keep track of parties that repeatedly fail during negotiations. Such unreliable parties might be excluded from future negotiations.
For a detailed description of the failure handling protocols, refer to Section 7.1.

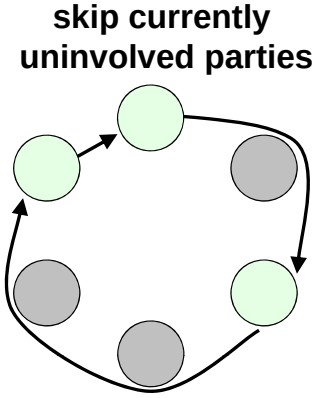

Figure 6.3: *Optimization of Agreement Negotiation with Ring Traversal*

Optimization

In the basic form, *many-to-many with ring-traversal* is a communication pattern that always requires all parties to process a message during one round. This is not as bad as

one might assume, because most of the times there are changes to the document that require that every party inspects the negotiation state. However, this communication pattern can be optimized.
If one assumes that each party makes its Proposals as soon as it learns about Requirements from another party, one can optimize the communication during Agreement Discovery. Sometimes the new Requirements during one round are irrelevant for a given party. The *VersaNeg* framework detects that a party is not in the Audience of any of the newly made Requirements and consequently can be skipped (see Figure 6.3). This algorithm improves the performance during Agreement Discovery.
Obligation Disclosure does not follow the many-to-many with ring-traversal communication pattern. All information is exchanged in a sequence from the leaf nodes in direction of the root node. The message exchange here is solely determined by sequence defined by the Agreement Option that was tuned into a set of commitments.

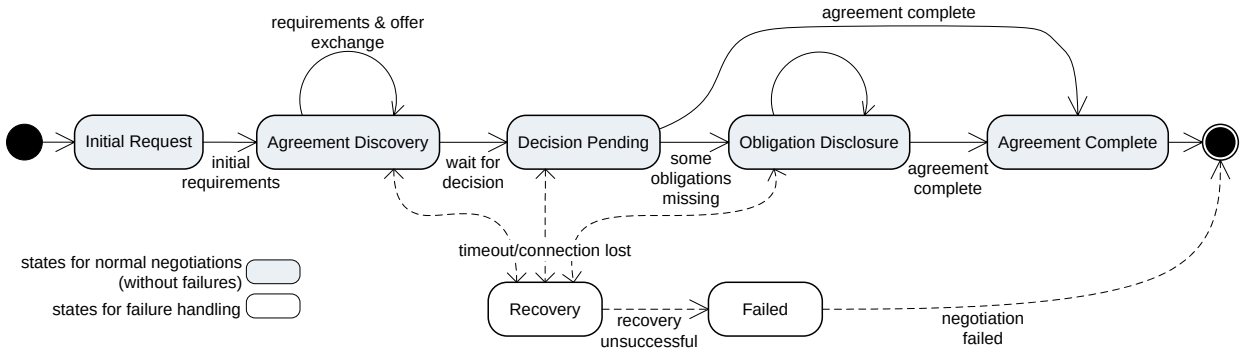

Figure 6.4: *Message States during Multilateral Negotiation*

6.3.1.3 Message States for Multilateral Agreement Negotiation

The many-to-many with ring traversal communication pattern uses the same message state transitions as bilateral negotiation (see Figure 6.4 in Chapter 5).
One party in the ring starts a negotiation by building the *Initial Request* containing its Requirements. The next party receives this message and enters the *Agreement Discovery* state. This phase discovers dependencies between the parties and leads to a number of different agreement options. Each agreement option is acceptable for all involved parties. Of course, a party does not accept everything in an agreement option, but every Requirement of the involved parties in this agreement option is satisfied.
The party that receives the message extracts the negotiation state which contains different Requirements, some of which might be satisfiable by the current party. This party could now make an Offer to some Requirements and specify its own Requirements. The negotiation document circulates a number of rounds during the *Agreement Discovery* until eventually no new agreement options can be discovered.
The message is now in the *Decision Pending* state and one agreement option is chosen then, based on the specified decision method. There are various approaches to decision making in distributed systems that also apply here. One might simply argue that the party that initiated the multilateral negotiation pays the bill in the end and will consequently also be the party that takes a decision. Of course consensus building methods can also be applied here, as well as approaches to the social choice problem [PeFP06].
After a choice is made, all parties are now committed to this agreement option, all other options are invalidated.

The Offers are now turned into Obligations. Some Offer might already contain all required information whilst other Offers might not be complete, yet. The *Obligation Disclosure* phase assures a careful exchange of Obligations and of their sensitive information. After all Obligations are complete, the message is now in the *Agreement Complete* state and each party receives a copy of this message.

This message constitutes a legally binding agreement between all parties and can directly be fed into the business process. The negotiation now enters the Agreement Execution phase and each party stores the agreement in a database and translates it into a local execution plan to steer the collaborative business process. The details within the Obligations provide the input parameters for the process. The business process handles the agreement and verifies that the party receives everything that has been promised and that it fulfills its Obligations.

6.3.2 Extensions to Negotiation Language and Framework for Multilateral Agreement Negotiation

The comprehensive negotiation states from bilateral agreement negotiation with *VersaNeg* (see Section 5) only require few extensions to handle complex multi-party agreement negotiations. The basic concepts stay the same: The negotiation protocol evolves around an exchange of *Requirements*, *Offers* and *Obligations*.
One party states its Requirements and one or more parties compete to make Offers for satisfying the Requirements. The initiator of a negotiation can invite a closed group to join the negotiation. The initiator might also permit that each party in the negotiation invites other parties even during the negotiation. Eventually the negotiation arrives at different agreement options. One agreement option is chosen and Offers are turned into Obligations.
The following Sections will introduce a reference example for multilateral agreement negotiation, to explain the different extensions to the negotiation protocol and the framework.

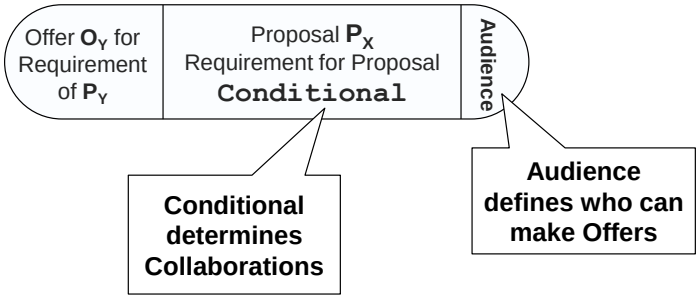

Figure 6.5: *Extended Graphical Representation of Multilateral Dependency Tree*

6.3.2.1 Multilateral Dependency Tree of Proposals

The objective of the multilateral negotiation protocol is to discover dependencies between Requirements, Offers, and Obligations between multiple parties.
Each Requirement a party states can either address all parties or target a specific audience. The *audience* element defines which parties are allowed to make an Offer for a Requirement. This can either be a list of parties or an "Any" specifier to allow all parties to make Offers.
Sometimes, one negotiating party alone is not capable to satisfy a given Requirement.

The protocol allows negotiators to cooperate to satisfy one Requirement together.
The graphical representation of Proposals for multilateral negotiations in Figure 6.5 contains the Audience in the right-hand side. On the bottom of the Proposal node is a *Conditional* which defines how parties might cooperate to satisfy a Requirement through cooperation. The `ExactlyOne` Conditional states that only one child of this node may become part of the final agreement. Hence, the `ExactlyOne` expresses the same choice as introduced previously for bilateral agreement negotiation in Section 5.3.3.1. One can think of bilateral dependency trees, as dependency trees where all Conditionals are `ExactlyOne`. The atomic relationship between one Proposal that contains a Requirement and one Proposal that contains a matching Offer is still one-to-one. The dependency tree as a whole describes multilateral dependencies that a formed by interdependent pair wise dependencies. A formal definition of Conditionals for different collaboration options follows in Section 6.3.2.4.

6.3.2.2 Reference Scenario for Multilateral Negotiation

Let us consider an example to show how Proposals with their Requirements and Offers relate in a multilateral setting.
Consider a cloud computing example where a client wants to start a highly scalable new service and relies on the collaboration of specialized service providers. The client p_1 wants Software as a Service and specifies a minimum number of users the system should be able to support concurrently. The Proposal $P_A^{p_1}$ contains the Requirement with the specifications of the service and the Proposal $P_B^{p_1}$ defines lower bounds for certain quality of service and reliability metrics. The party p_2 makes two Proposal $P_C^{p_2}, P_D^{p_2}$ to indicate that it wants to provide the service as requested by party p_1. Another service provider p_3 makes the Offers $P_E^{p_3}, P_F^{p_3}$ but has one Requirement. At this stage in the negotiation there are two parties competing for providing the service for p_1.
The XML *Requirement* fragment for the Requirement in $P_A^{p_1}$ is shown in Listing B.1 in the Appendix. The corresponding Offer from $P_D^{p_2}$ is shown in Listing B.2.

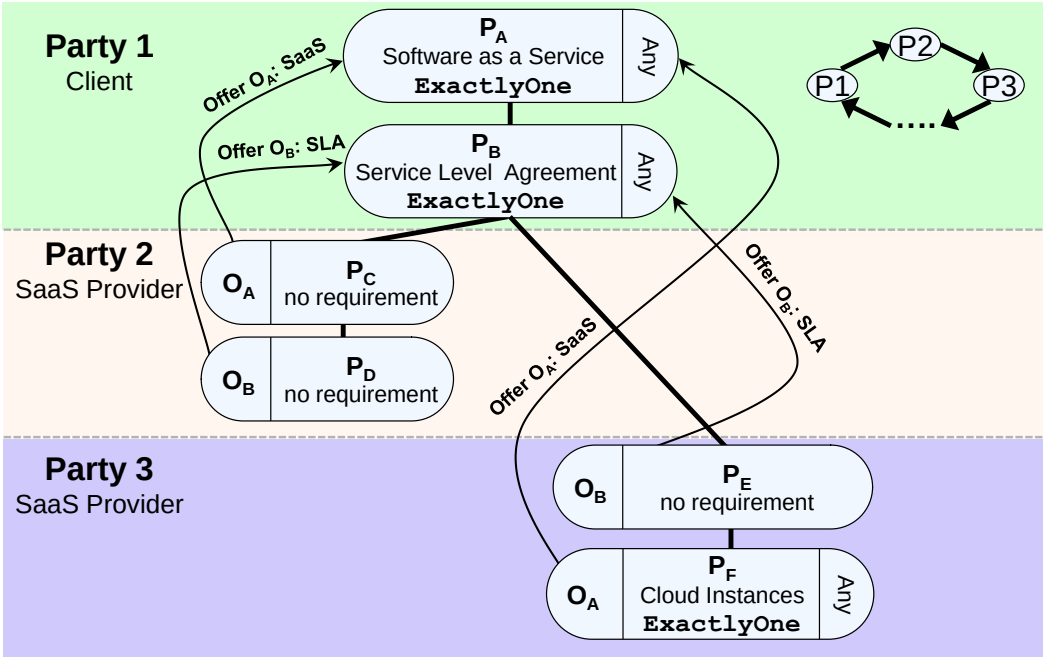

Figure 6.6: *Multilateral Dependency Tree of Proposals after Two Messages*

Proposal	Who	Offer/ Req.	Description
P_A	P_1	R_{Video}	Content provider wants to make its video catalog available for on-demand video streaming and searches for platform provider.
P_B	P_1	R_{SLA}	Network access characteristics are defined in separate SLA. Violator must pay penalty for not meeting requirements.
P_C	P_2	O_{Video}	Video platform provider, specialized in life streaming on behalf of customers, offers to host content.
P_D	P_2	O_{SLA}	Video platform provider guarantees that network access under the specified metrics and constraints.
P_E	P_3	O_{Video}	SaaS provider of streaming software application stack offers to host content if R_{Cloud} is satisfied.
		R_{Cloud}	SaaS provider only maintains software stack and relies on cloud platform to host service.
P_F	P_4	O_{SLA}	SaaS provider guarantees that network access under the specified metrics and constraints.
P_H	P_5	O_{Cloud}	Cloud platform provider offers to host service under the specified network access constraints.
P_G	P_6	O_{Cloud}	Cloud platform provider offers to host service under the specified network access constraints.

Table 6.1: *Proposals with Offers O and Requirements R in the multilateral reference example*

For an `ExactlyOne` Conditional node, only one Proposal child element can become part of the final agreement. Hence, an `ExactlyOne` Conditional could lead to a number of minimal complete agreement options that equals the number of its children. In our example presented above, the `ExactlyOne` Conditional at Proposal $P_B^{p_1}$ states that only one of its children can be chosen. The Proposal $P_F^{p_3}$ of party p_3 has a Requirement which does not have an Offer, yet.
Hence, at the current state of the negotiation only the Proposal of party p_2 leads to an agreement option $\{P_A^{p_1}, P_B^{p_1}, P_C^{p_2}, P_D^{p_2}\}$.

6.3.2.3 Growing the Multilateral Dependency Tree during Agreement Discovery

The basic communication follows the *many-to-many with ring traversal* communication pattern as explained in Sec. 6.3.1. The multilateral dependency tree grows when the individual parties append their Proposals.

The *initiating party* starts the negotiation by appending Proposals that contain its initial Requirements, for instance, the request for Software as a Service. The initiating party is also responsible for defining which parties will be part of the negotiation, for the order of the parties in the ring and to state how to choose one agreement option. The protocol requires that each Proposal in the negotiation state must arrive at all parties that might be interested in this Proposal during one round. If one Requirement has Any as audience, the negotiation message reaches each party in the ring. It is possible to skip parties in the optimized ring traversal protocol mode (see Section 6.3.1.2). If the Audience of the newly added Proposals only address a subset of parties, only these

parties will be involved.
The next party in the ring that receives a negotiation state, determines if it is willing to satisfy one or more Requirements and can now add its own Proposals to the negotiation state. This party only inspects the Requirements that have been added since it processed the negotiation state in the previous round.

The initiator of a negotiation specifies if a negotiation runs until either no more Proposals are made or until a defined number of satisfactory agreement options have been discovered. The initiator also defines a timeout to trigger the negotiation recovery and a larger timeout to abort the negotiation. Many different methods are possible to decide which agreement option should be used. Interfaces towards a decision intelligence have already been introduced in Section 5.3.4.4.
After one agreement option is chosen the parties involved in this option are committed to provide Obligations that are consistent with their Offers. Conversely the parties are no longer committed to Offers made for other agreement options.

The rules that defined where one party might append its Proposals for a Requirement to, must be extended for the multilateral case. The bilateral rules in Listing 5.5 do not address that different parties compete for making the same Offer. Each Offer must be appended as a sibling of the competing Offers.

Append Proposals to Multilateral Dependency Tree

The protocol defines where a party may append Proposals to the dependency tree. Let S_x be the negotiation state in negotiation step x (the message number x in the negotiation). If $P_o^{p_i} \in S_x$ is the node the party i wants to make a number of interdependent Proposals for, this party may append its Proposals to leaf nodes of the biggest subtree of S_x that has $P_o^{p_i}$ as its root node and that only contains nodes that belong to the owner p_i of $P_o^{p_i}$. The function $SubtreeLeafNodes(P_o^{p_i})$ returns such nodes:

```
1 SubtreeLeafNodes(P_n):
2   K={P_c ∈ children(P_n) | owner(P_c)=owner(P_n)}
3   if K=∅
4     RETURN {P_n}
5   else
6     RETURN ⋃_{k∈K} SubtreeLeafNodes(k)
```

Listing 6.1: *Find Leaf Nodes in the Subtree where all Nodes belong to Owner of P_n*

If this party already added some Proposals to the leaf nodes of this subtree, whilst processing S_x, it may append its additional Proposals to these Proposals nodes to form dependencies between own Proposals.
Let E_x be the set of nodes that party i already appended during the current processing step x. Let the function $descendants(P_o^{p_i})$ return all descendants of $P_o^{p_i}$. The following algorithm takes the node $P_o^{p_i}$ and E_x as input and returns all nodes where this party is allowed to append its Proposals to. The current party may choose any subset of these possible nodes to append to. The function $CandidatesToAppendTo(P_o^{p_i}, E_x)$ returns all nodes that a party may append its Proposals to.

```
1 CandidatesToAppendTo(P_o^{p_i}, E_x):
2     D={P_d ∈ E_x | P_d ∈ descendants(P_o^{p_i})}
3     L=SubtreeLeafNodes(P_o^{p_i})
4     RETURN D ∪ L
```

Listing 6.2: *Find all possible Nodes to append Proposals to*

With this algorithm, all other parties are forced to append to leaf nodes of a subtree of S_x that contains only nodes that belong to the owner of P_o^{pi} and that contain P_o^{pi} or to nodes that a party inserted during the current processing step.
The property that one party must not insert nodes within a subtree of nodes that belongs to the owner $P_o^{p_i}$, allows the party p_i to specify Proposals with Requirements that have all to be satisfied. p_i may insert a node $P_X^{p_i}$ as the sole child of another node $P_Y^{p_i}$ to force other parties to satisfy both Requirements of $\{P_X^{p_i}, P_Y^{p_i}\}$; otherwise this path can not become part of an agreement option.

Let us consider some examples how to apply these rules. How did the negotiation presented above, arrive at the dependency tree as shown in Fig. 6.6? Where in the tree is a party allowed to append its new Proposal?

Party p_1 stated its initial Requirements $P_A^{p_1} \leftarrow P_B^{p_1}$. Party p_2 made Offers for $P_A^{p_1}$ and $P_B^{p_1}$. According to the rule stated above, p_2 could only append its new Proposal $P_C^{p_2} \leftarrow P_D^{p_2}$ as a child of $P_B^{p_1}$. The reasoning behind this rule is that one party p_m could avoid satisfying a Requirement by adding a sibling $P_X^{p_m}$ for $P_A^{p_1}$ within the subtree that belongs to party p_1. Doing so would lead to the dependencies $P_A^{p_1} \leftarrow P_B^{p_1}, P_X^{p_m}$ and would introduce an additional agreement option $\{P_A^{p_1}, P_X^{p_m}\}$. This additional agreement option is probably not what party p_1 intended because $P_B^{p_1}$ would not be satisfied.
The rule additionally states that another party p_3 could also only append its new Proposals $P_E^{p_3} \leftarrow P_F^{p_3}$ for the Requirements of $P_A^{p_1} \leftarrow P_B^{p_1}$ to $P_B^{p_1}$. Hence $P_C^{p_2}$ and $P_E^{p_3}$ are siblings. The rule prohibits that a party p_m appends a Proposal $P_Y^{p_m}$ to $P_D^{p_2}$. If p_m could append its Proposal there, the Proposal $P_D^{p_2}$ could only become part of an agreement option together with $P_Y^{p_m}$ and only if $P_Y^{p_m}$ also receives an Offer.

The negotiation continues now as shown in Figure 6.7.
The party p_3 only offers and administrates the software but does not host the SaaS itself. It relies on a cloud provider to host the service and uses the Requirement in $P_F^{p_3}$ to request that cloud providers should host a number of machine instances. This Requirement is shown in Listing B.1 in the Appendix.
The Proposal of the SaaS provider can only become part of an agreement option if at least one cloud provider wants to collaborate on hosting the machine instances. The two cloud providers p_4 and p_5 make the Proposals $P_G^{p_4}$ and $P_H^{p_5}$ respectively and do not state further Requirements. The party p_4 makes an Offer as shown in Listing B.2 in the Appendix.

The negotiation enters now the Decision Pending phase. There are three agreement options: p_2 could provide the service on its own. p_3 introduces two agreement options, because it could either collaborate with p_4 or p_5. Notice how the *VersaNeg* protocol makes the dependency of cloud provider and SaaS provider visible for p_1. This knowledge allows p_1 to influence the decision which cloud provider should be chosen if p_3

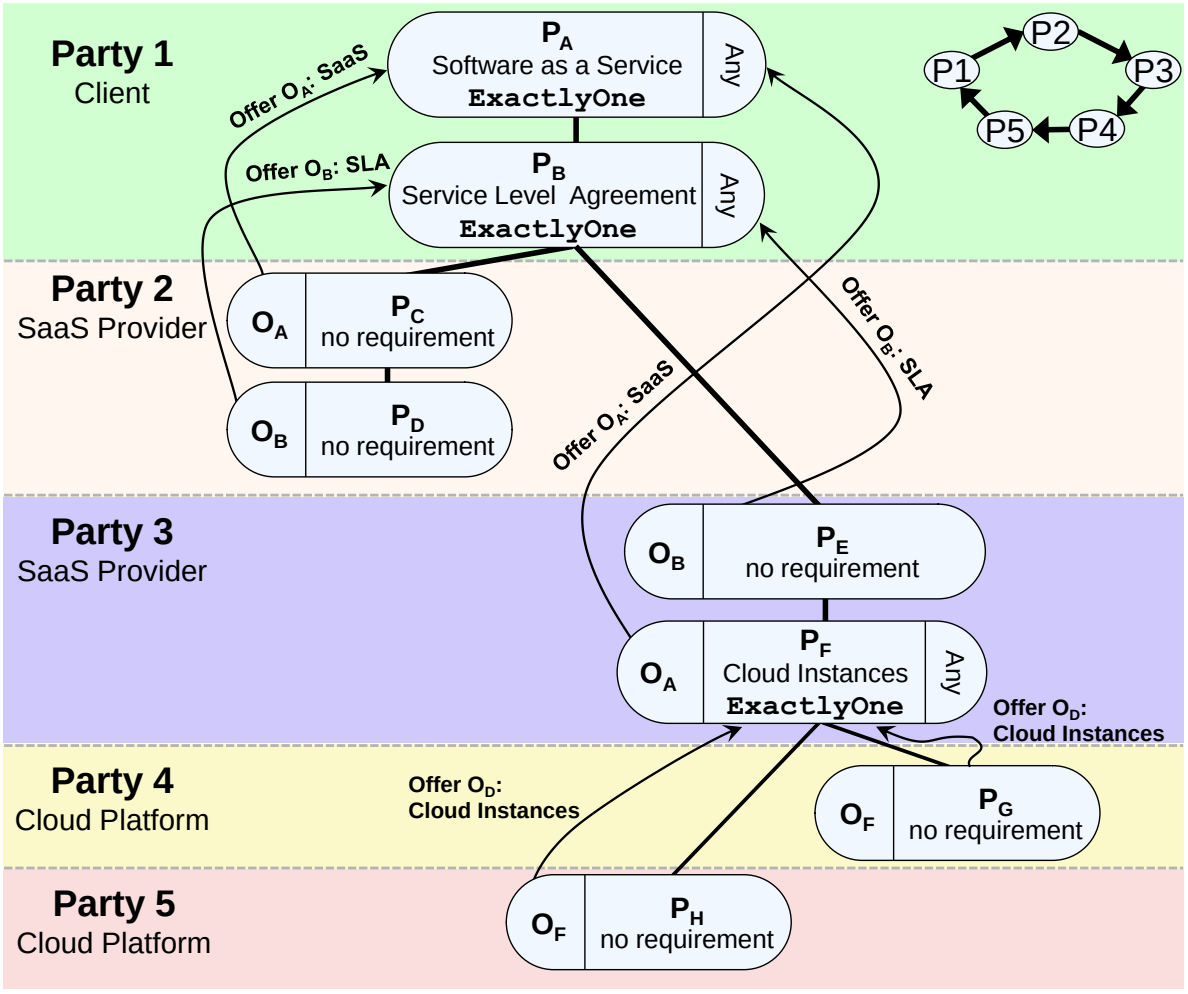

Figure 6.7: *Dependency Tree of Proposals after Four Messages*

would Offer the service.

Real world negotiations are probably much more complex. One party could make different Offers for the same Requirement to give a choice, for instance, to distinguish between basic support and level 3 support which is available 24/7. Offers containing prices can be encrypted for the party that stated the Requirement. This can even be used to integrate sealed-bid auctions with *VersaNeg*. Each party encrypts its bid within an Offer. One party decrypts all bids and determines the lowest/highest bidder.

6.3.2.4 Collaborations and Agreement Options

This section describes how to search minimal complete agreement options in a tree of interdependent Proposals with different Conditionals during the Decision Pending phase. The objective of the following algorithm is to discover all minimal complete agreement options without lexicographical permutations. The algorithm relies on a depth first search [Tarj71] to discover the agreement options. It descends until a leaf node in the Proposal tree is reached. After reaching the leaf nodes, the algorithm ascends and collects satisfiable paths S from leaf to the root of the document where $\mathcal{F}(p) = True$. Let S be the set of all possible satisfiable paths. The recursive algorithm that returns the set of minimal complete agreement options is as follows:

MultilateralAgreementOptions(P):

$S = \{$**MultilateralAgreementOptions**$(child) | \forall child \in children(P)\}$

RETURN **SatisfiablePaths**(P, S)

Listing 6.3: *Depth first search for agreement options*

The function $SatisfiablePaths(P, S)$ takes the current node P during the ascent and the set of satisfiable paths S contributed by its children to construct the paths that result from this node P. The function $\sigma_d(P)$ returns the Conditional as described below. It depends on the Conditional how satisfiable paths from the children are combined. If P is the root node of the tree, all paths that $MultilateralAgreementOptions(P)$ yields are satisfiable agreement options.

$$SatisfiablePaths(P, S) = \begin{cases} \emptyset & \text{if } children(P){=}\emptyset \wedge \mathcal{F}(\{P\}){=}False, \\ \{P\} & \text{else if } children(P){=}\emptyset \wedge \mathcal{F}(\{P\}){=}True, \\ SatisfiablePaths_{ExactlyOne}(P, S) & \text{else if } \sigma_d(P){=}ExactlyOne, \\ SatisfiablePaths_{All}(P, S) & \text{else if } \sigma_d(P){=}All, \\ SatisfiablePaths_{OneOrMore}(P, S) & \text{else if } \sigma_d(P){=}OneOrMore. \end{cases} \tag{6.1}$$

One party can state that it offers alternatives for satisfying its Requirements. The expression $P_A \xleftarrow{ExactlyOne} P_B, P_C$ implies that there are two satisfiable paths $\{P_B, P_A\}$ and $\{P_C, P_A\}$. For an `ExactlyOne` node, the deciding party has the choice here to select exactly one path that was contributed by the children of this node. To discover all viable options for an `ExactlyOne` node, the Equation 6.2 adds the current node P to each path from the children S, verifies that the resulting paths are still satisfiable and returns this set of satisfiable paths.

$$SatisfiablePaths_{ExactlyOne}(P, S) = \{e \,|\, \forall s \in S, e := s + \{P\} : s \neq \emptyset \ \wedge \ \mathcal{F}(e) = True\} \tag{6.2}$$

A Proposal tree that only consists of `ExactlyOne` nodes contributes one agreement option for each satisfiable path from leaf node to root node. Bilateral dependency trees (see Section 5.3.3.5) only consist of `ExactlyOne` conditionals. The tree in Figure 6.8 shows a dependency tree. There would be three agreement options if the Conditional at P_A is `ExactlyOne`: $\{\{P_B, P_A\}, \{P_D, P_C, P_A\}, \{P_E, P_C, P_A\}\}$.

The drawback of `ExactlyOne` nodes is that parties cannot collaborate to satisfy **one** Requirement. Imagine a cloud computing scenario, where no single party can satisfy a request for CPU time of 500 hours at at least 100 instances in parallel, but two parties that collaborate could satisfy the Requirement. With `ExactlyOne` there can only be alternative solutions by choosing either one party or the other to satisfy the Requirement.

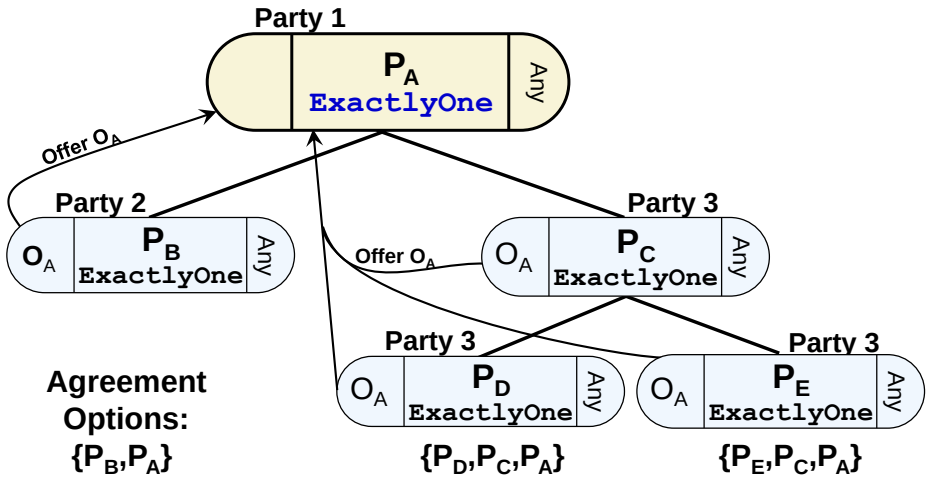

Figure 6.8: *`ExactlyOne/All/OneOrMore` Conditional in P_A*

The `All` node allows multiple nodes to collaborate on satisfying one Requirement. Each party that appends Proposals to `All` promises to satisfy the Requirement, if the `All` node becomes part of the chosen agreement option.

$$SatisfiablePaths_{All}(P,S) = \begin{cases} \emptyset & \text{if } \exists s \in S, e := s + \{P\}, \\ & s = \emptyset \ \vee \ \mathcal{F}(e) = False \\ \bigcup\limits_{s \in S} s + \{P\} & \text{otherwise} \end{cases} \tag{6.3}$$

An example would be that the Conditional in the root node of Figure 6.9 is `All` and that the Requirement demands that each party has to specify its postal address. The example has two agreement options, due to the `ExactlyOne` node P_B. Note how P_B is joined with the two satisfiable paths $\{\{P_D, P_C\}, \{P_E, P_C\}\}$ contributed by P_C to result in $\{\{P_B, P_D, P_C, P_A\}, \{P_B, P_E, P_C, P_A\}\}$.

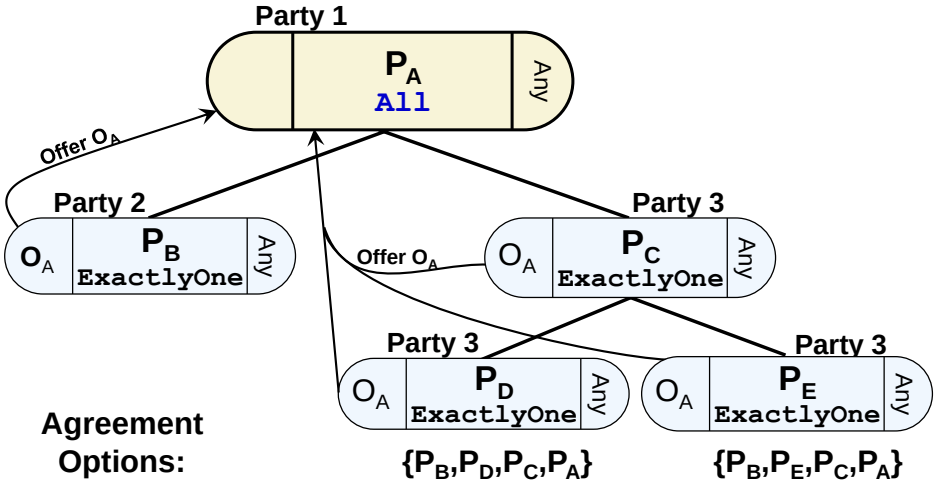

Figure 6.9: *`All` Requirement*

Sometimes one party can not specify a priori how many parties should make Offers to its Proposals and the contained Requirements. The `OneOrMore` node states that the

owner of the Requirement wants that any subset of the paths from the children can become part of the agreement option. This means the owner of the Requirement allows for collaborations of a subset of different parties that made an Offer to satisfy one Requirement. Hence `OneOrMore` introduces a set of combinations of all viable options that satisfy its Requirement.
To compute these combinations the function $\mathcal{P}(s)$ returns the power set of the set of paths s. The power set of s is a set of all subsets of s containing 2^n sets of paths, where n is the number of paths in s. Each satisfiable subset of these paths could become part of the agreement. If one set in $\mathcal{P}(s)$ contains more than one path, these paths are joined with $SatisfiablePaths_{All}$.

$$SatisfiablePaths_{OneOrMore}(P,S) = \{SatisfiablePaths_{All}(P,s) | \forall s \in \mathcal{S}(S), s \neq \emptyset\} \tag{6.4}$$

There are five agreement options if the Conditional in Figure 6.8 is `OneOrMore`: $\{\{P_B, P_A\}, \{P_D, P_C, P_A\}, \{P_E, P_C, P_A\}, \{P_B, P_D, P_C, P_A\}, \{P_B, P_E, P_C, P_A\}\}$. The `OneOrMore` node is an important element because it allows multiple parties to collaborate and make a number of Offers to satisfy one Requirement together. `OneOrMore` enables the deciding party to choose the subset of all Proposals that satisfy above mentioned Requirements at the most favorable terms.

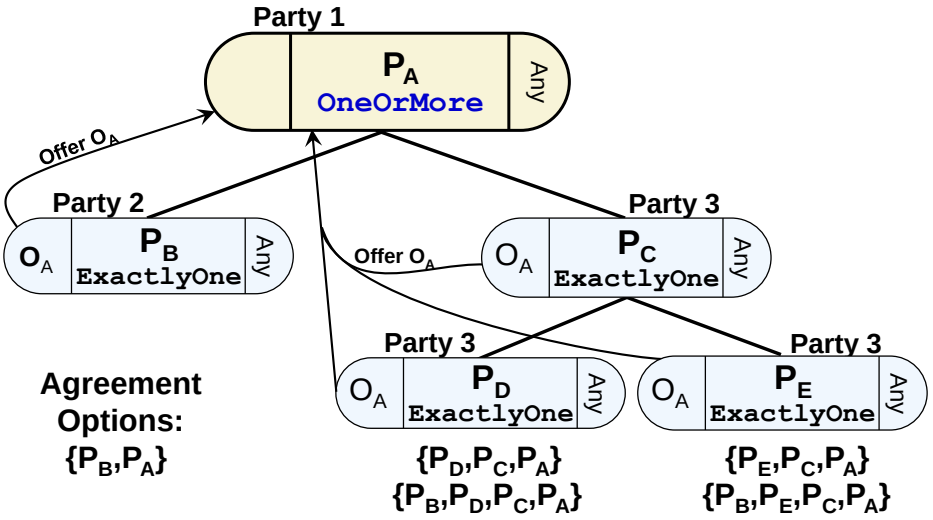

Figure 6.10: *`OneOrMore` Requirement*

6.3.2.5 Extended XML Message Format

The Appendix D contains the comprehensive XML Schema definition in Listing D.1 that defines permissible negotiation messages of the *VersaNeg* protocol for bilateral and multilateral negotiations.

The `ExactlyOne/All/OneOrMore` Conditionals section can take the *Proposals* from the same party or from other parties.

Combination	Agreement Options
ExactlyOne	$P_B P_A$
	$P_D P_C P_A$
	$P_E P_C P_A$
All	$P_B P_D P_C P_A$
	$P_B P_E P_C P_A$
OneOrMore	$P_B P_A$
	$P_D P_C P_A$
	$P_E P_C P_A$
	$P_B P_D P_C P_A$
	$P_B P_E P_C P_A$

Table 6.2: *Agreement options*

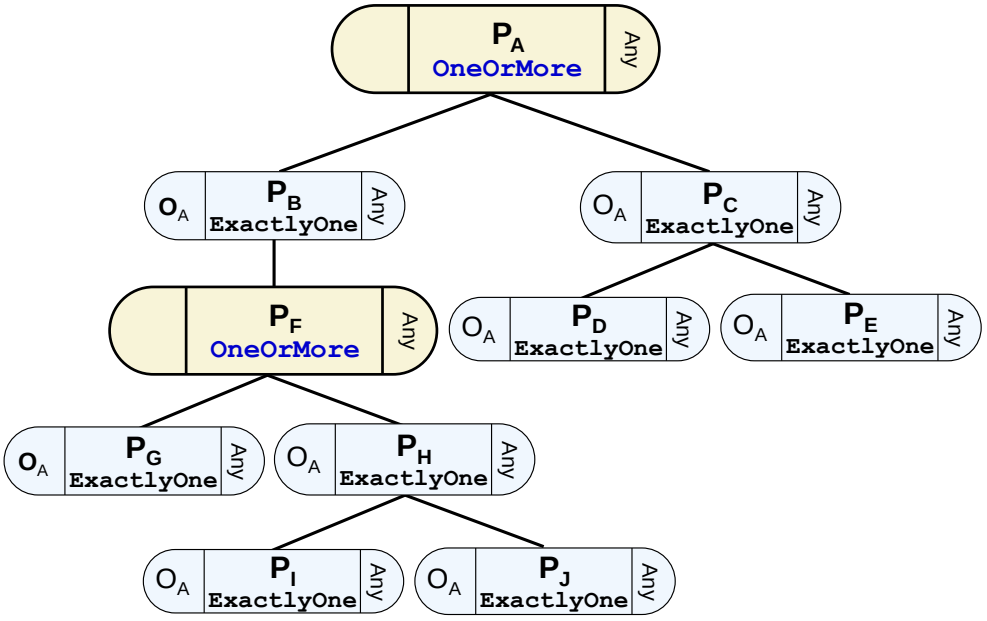

Figure 6.11: *Nested OneOrMore*

6.3.3 Analytical Evaluation and Experiments

The negotiation language allows for many choices about which agreement option to pursue and it allows for the collaboration of different parties. Sinz et al. [SKFG06] showed that it is feasible to decompose complex product specifications into small fragments that can dynamically form an unambiguous tree of propositional conditions and constraints which can be checked automatically for completeness and consistency. Our approach relies on a similar tree structure to express conditions and could therefore be enhanced with this type of automatic model checking to assist humans in formulation of the Requirements and the supervision of negotiations.
Hence, this section has a different focus and will provide an analysis of how many possible solutions a given negotiation can have. Next comes an experimental evaluation of the performance of our protocol implementation, from which we will derive a generic performance model. Finally we will present an analytical comparison with hypothetical stateful multi-party negotiation approaches.

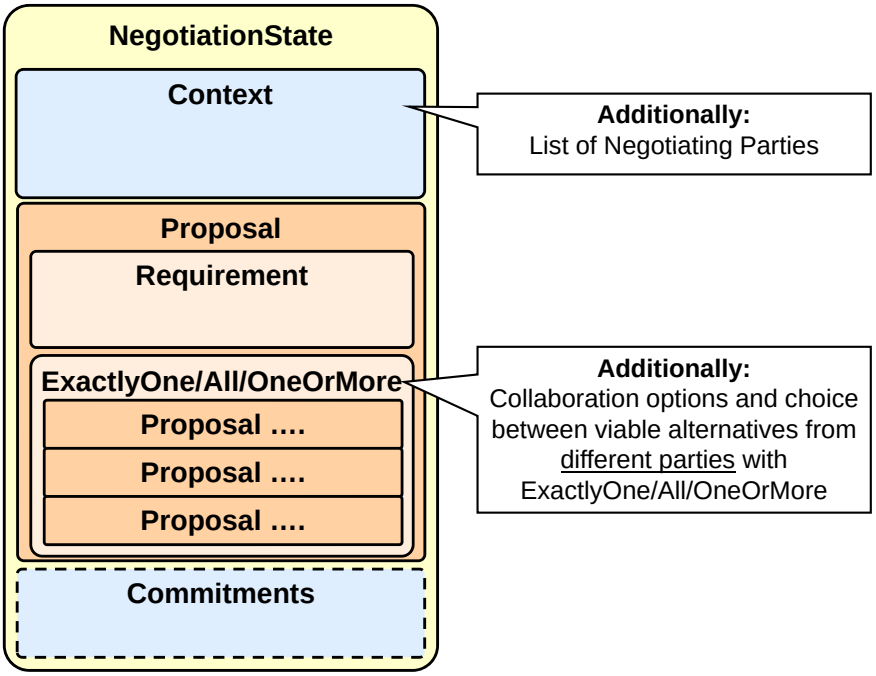

Figure 6.12: *Enhancements of XML Structure for Multilateral Negotiations*

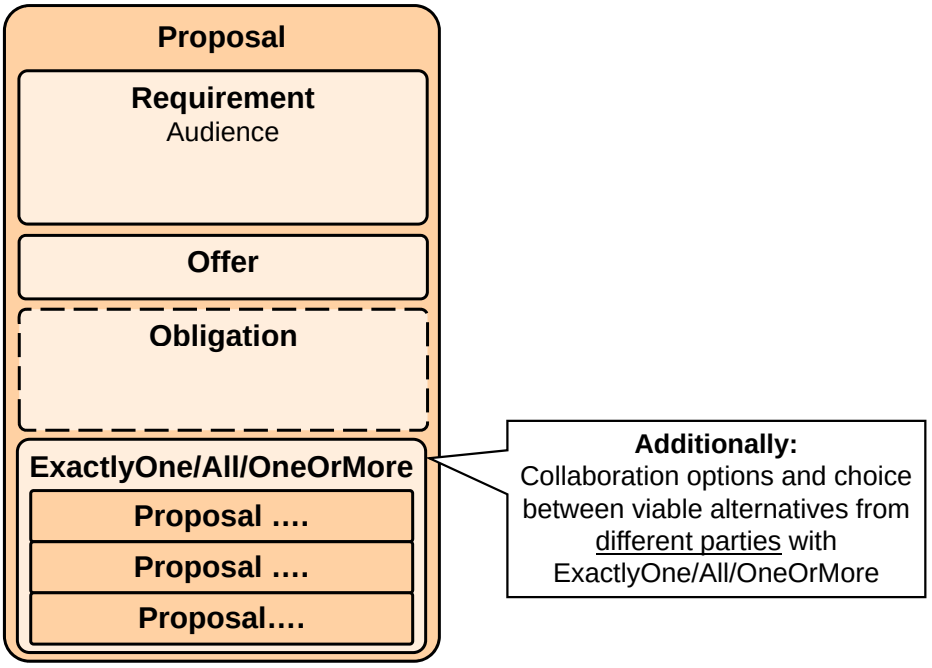

Figure 6.13: *Enhancements of Proposal for Multilateral Negotiations*

6.3.3.1 Quantifying the Solution Space of Agreement Options

One can give a recursive algorithm that calculates the number of agreement options for a dependency tree that has been stripped of all unsolvable paths. Most of the times, the unsolvable paths in a tree are quite obvious, for instance, all paths with leaf nodes that are *Conditional Offers* are unsolvable and can be removed. Hence, this algorithm can provide the number of agreement options in a more efficient manner than searching for all possible agreements $SatisfiablePaths(...)$. The function $N(v, C)$ takes the current

node v under investigation and all children C of node v. The function $N(v, C)$ starts at the root node of the agreement tree.

$$N(v, C) = \begin{cases} 1 & \text{if } C = \emptyset, \\ N_{ExactlyOne}(C) & \text{if } \sigma_d(v) = ExactlyOne, \\ N_{All}(C) & \text{if } \sigma_d(v) = All, \\ N_{OneOrMore}(C) & \text{if } \sigma_d(v) = OneOrMore. \end{cases} \tag{6.5}$$

If a leaf node is reached, it contributes one potential option. For the `ExactlyOne` node, the number of potential options equals the sum of the contributed paths from all children:

$$N_{ExactlyOne}(C) = \sum_{a \in C} N(a, children(a)) \tag{6.6}$$

The `All` vertex may not only contribute new solutions but might even reduce the number of solutions. Each child must have at least one solution to satisfy the `All` node. If, for instance, there is only one solution contributed by each child, the `All` yields only one solution. If the children have multiple solutions, the different possible permutations increase the number of the solutions.

$$N_{All}(C) = \prod_{a \in C} N(a, children(a)) \tag{6.7}$$

The `OneOrMore` node allows for different combinations of solutions from the children. Hence, the number of solutions can be summed over the elements of the power set $\mathcal{P}$.

$$N_{OneOrMore}(C) = \sum_{s \in \mathcal{P}(C)} N_{All}(s) \tag{6.8}$$

6.3.3.2 Performance Evaluation

The performance and scalability of the negotiation protocol determines the scenarios the multi-party negotiation protocol is suitable for. The protocol performance is not a big obstacle if humans are involved in the process, where negotiations can easily last for hours or even for weeks. However, the protocol performance can be a determining factor for fully automated concurrent negotiations that compete for the same limited resources.

The evaluation of the bilateral negotiation performance in Section 5.3.5 and Section 5.4.5.2 show the results for VersaNeg with the same implementation as used for the multilateral experiments in this section. The bilateral results show the rate of messages that a negotiation server can handle. These numbers also apply to the message rate that a negotiation server can handle for multilateral negotiations. In the end, a negotiation server receives a messages, processes the messages, and sends a new message, regardless if it is a bilateral negotiation or a multilateral negotiation. The main determinant of the message rate is the message size. Therefore, this section will focus on the duration of multilateral negotiations.

The C++ implementation of *VersaNeg* includes the negotiation logic for handling XML message states and growing the dependency tree, basic tests for protocol compliance of

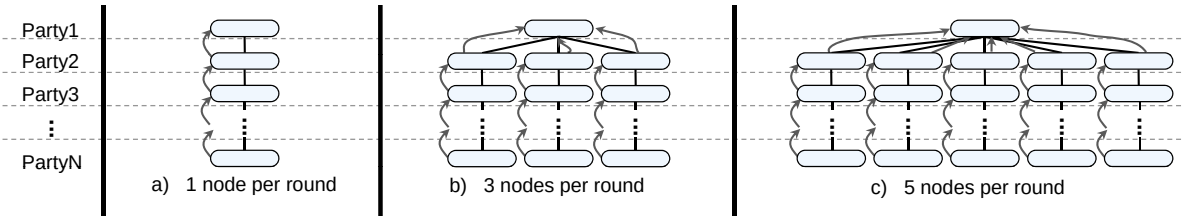

Figure 6.14: *Test cases with 1/3/5/7/9 Requirements per party led to Proposal trees with 1/3/5/7/9 agreement options*

messages, the algorithm to derive all agreement options from a negotiation state and the automated validation of Obligations against Requirements. *VersaNeg* used libxml2[1] for all XML operations. The framework was executed within the *VersaNeg* multi-threaded server and the messages were sent via TCP. No artificial intelligence methods were used for making choices, instead disclosure policies were used for automatically matching Requirements with Offers and for providing matching Obligations. A random agreement option was chosen during Decision Pending. After providing the Initial Request for a test case, the negotiation runs fully automated.

The performance numbers stem from experiments in a testbed of 4 PCs (Linux Kernel 2.6.26, Intel Core 2 Duo 6600@2.40GHz, 2.6 GB RAM) interconnected by a LAN with RTTs of less than 1ms. One additional control PC issued scripted commands to control the experiments. Each test was run at least 50 times with several seconds between each test. The variances experienced during the measurements were low. The result plots contain error bars for the 1% and 99% quantiles.
The implementation contains one optimization in comparison with the *many-to-many with ring traversal* presented in Section 6.3.1. It allows skipping parties that have nothing to do. This is the case if they are not in the audience of any newly inserted Requirement or if it is not their turn to provide Obligations. The security algorithm is aware of this optimization and can still determine if parties are skipped maliciously, i.e. parties are skipped that could contribute to the negotiation. The cloud example presented in Figure 6.7 disclosed 3 Obligations and took 37 ms on average in this environment.

Several test cases were generated to measure the influence of 1.) Obligation size, 2.) Proposal count and 3.) network delays on total duration for negotiations using the protocol. The generated test cases led to dependency trees with 1, 3, 5, 7 and 9 branches (see Figure 6.14). PartyN is one of the 4 parties that receives the last message in Agreement Discovery. The difference between these test cases is the ratio of how many Obligations are exchanged to the number of Requirements and Offers. For the test case with five branches this ratio is 1:5. The framework only processes nodes that have been added or have been changed in one round. Therefore, the protocol performance only depends on the number of changed nodes and not on the interdependencies between the Proposals.

The basic test case consists of 4 parties with one round of Agreement Discovery and one round of Obligation Disclosure with one branch in the dependency tree as shown in Figure 6.14 a). The test case has therefore 8 messages in total. The default Obligation size is 270 kB and no traffic shaper was used on the LAN.

The first experiment used the parameters as described above and made variations to the Obligation size. The communication time and the time for the XML parser is a linear function of the Obligation size as shown in Figure 6.15. The negotiation logic and the

[1]The XML C parser and toolkit: http://www.xmlsoft.org/

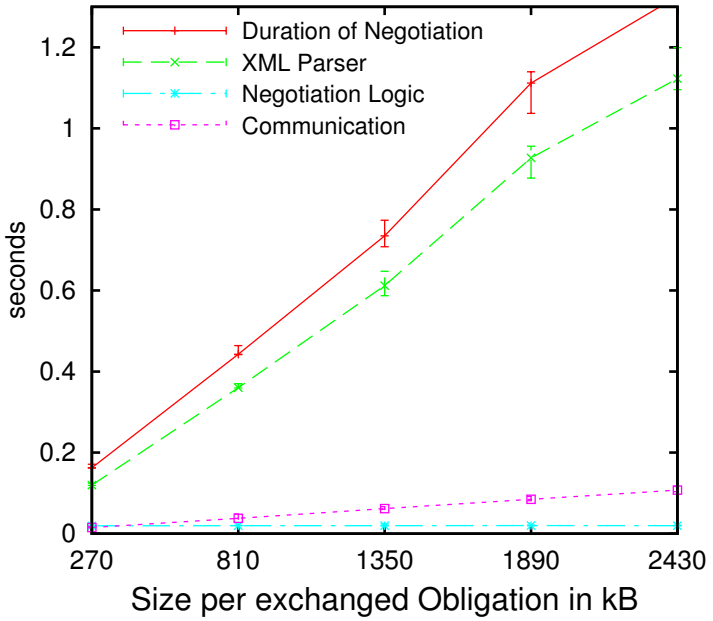

Figure 6.15: *Duration of complete negotiation depending on size of negotiation document and Obligations*

retrieval of disclosure policies are comparably cheap, because the implementation only needs to inspect the small number of Proposals that have been added during a round. The time of the negotiation logic is constant here, because the Obligation size does not influence the dependencies within the Proposal tree. The XML parser is responsible for parsing the constantly growing XML messages (up to 11 MB) from a string into a DOM tree, performing the XML schema validation of Obligations, and after the negotiation logic finishes, the XML Parser translates the DOM tree into a string representation. The duration of negotiation in the diagram is the time for one complete negotiation. The time for the negotiation grows linearly with the size of the Obligations. The bilateral experiments in Section 5.3.5 showed that the growing negotiation states will lead to an exponential decline for the negotiation rate a server can achieve.

The next experiment focused on the number of Proposals that have to be processed by each party. The number of Proposals a party appends at each round was increased. The tree has either 1,3,5,7, or 9 branches (see Figure 6.14) and each party makes this number of Proposals for each message it receives during Agreement Discovery. The depth of the tree determines the number of Obligations and is independent from the number of branches. Four Obligations were exchanged for each test. The resulting XML documents had nearly the same size, because the overhead of the additional Requirement and Offer nodes is relatively small in comparison with the Obligation size. The XML Parser and Communication showed nearly constant runtimes as expected, but the negotiation logic was now under higher load (Figure 6.16). The negotiation logic still exhibited good performance properties as it needed only 0.1 seconds for processing 9 Proposals per party.

The third experiment determined the impact of network delay with a tree as in Figure 6.14 a). The experiment relied on Obligations with 10 kB and used the Linux Advanced Routing & Traffic Control[2] to introduce a delay on the network connections.

[2]Linux Advanced Routing & Traffic Control: http://lartc.org

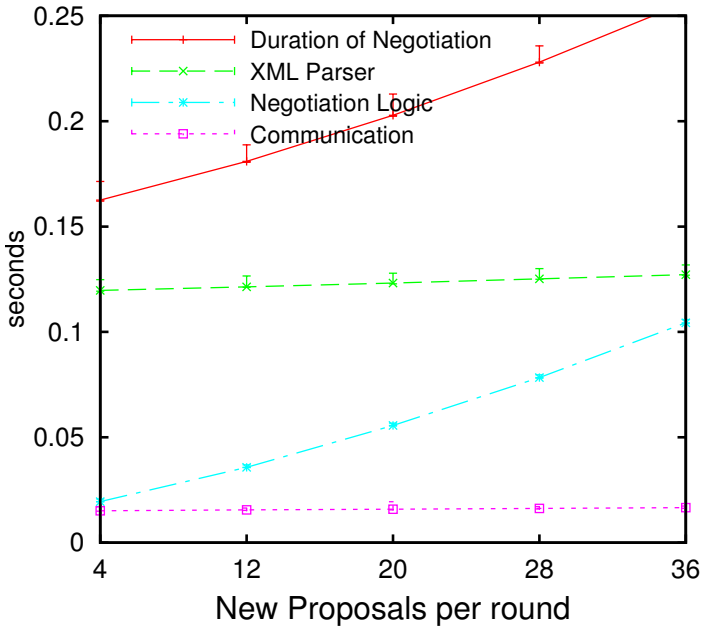

Figure 6.16: *Duration of complete negotiation depending on number of processed Requirements and Offers per round*

Each message is sent over a new TCP connection, requiring a full TCP connection handshake and forces all messages to go through TCP slow start. These delay settings were measured for 1, 2, 4 and 6 rounds of *Agreement Discovery*, this corresponds to 8, 16, 32 and 48 messages in the whole negotiation. One can see from the results in Figure 6.17 that the protocol scales linearly with latency. For 4 rounds it stays below 10 seconds with reasonable delays of 40ms. Even under difficult conditions with very long network delays and six rounds, the time for a complete negotiation still is around 40 seconds. The dominant factor that determines the protocol performance is the number of messages and the network delay. Still these numbers could be improved significantly by employing a web server with persistent connections, for instance, by using HTTP 1.1 and thus avoiding repeated TCP slow starts.

A method from statistical regression analysis helped to identify a number of linear dependencies between the presented parameters and the total negotiation time. By performing linear regression analysis, one can determine general models for the negotiation performance. The sum of squared distances between the fitted values and the observed values was minimized with GNU R[3] to devise a model for the negotiation performance. All the individual data points from each test presented above were fed into GNU R. This helped to identify the models which were able to explain the results with a minimum set of input parameters. The coefficient of determination R^2 is defined as the proportion of explained variance to the total variance of a dependent variable. The coefficient of determination R^2 is a good measure of how well future results will be predicted by the model.

Let p be the number of parties, d the normal distributed network delay, e the num-

[3]GNU R: http://www.r-project.org/

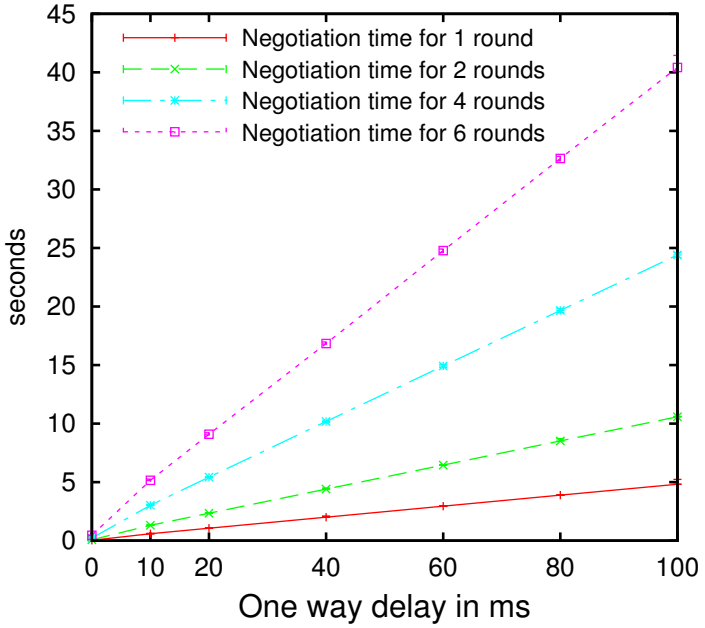

Figure 6.17: *Duration of complete negotiation depending on network latency and number of rounds*

ber of Requirements during the negotiation, r the number of rounds, s the size of the Obligations, α the intercept and $\beta_{i,j}$ are the regression coefficients.

$$\begin{aligned} NegotiationLogic =& \alpha_1 + \beta_{1,1} \cdot r + \beta_{1,2} \cdot p + \beta_{1,3} \cdot e \\ XMLParser =& \alpha_2 + \beta_{2,1} \cdot r + \beta_{2,2} \cdot p + \beta_{2,3} \cdot s \\ Communication =& \alpha_3 + \beta_{3,1} \cdot r + \beta_{3,2} \cdot p + \beta_{3,3} \cdot d \cdot r \cdot p \end{aligned} \tag{6.9}$$

The processing phases of the algorithm depend only on a subset of all experiment parameters as one can see in 6.9. The models manage to predict the performance accurately as also indicated by the values for R^2 which were all above 95% for the experiments. It is interesting to note the dependencies between the parameters that define the negotiation and how these parameters impact the performance.

$$Negotiation = \alpha_4 + \beta_{4,1} \cdot r + \beta_{4,2} \cdot p + \beta_{4,3} \cdot s + \beta_{4,4} \cdot d \cdot r \cdot p \tag{6.10}$$

The overall performance of the negotiation can be modeled as in 6.10 with R^2 of 99.6%.

6.3.3.3 Potential Performance Benefits of Introducing State

This section will give estimates of possible bandwidth reductions for multi-party negotiation by introducing negotiation state at the negotiating parties. The estimates are for the number of messages and for the total bandwidth consumption during the negotiation for negotiations with increasing number of rounds.

The multi-party negotiation protocol with comprehensive negotiation states (*Stateless VersaNeg*) has a number of desirable functional properties, such as the ease to turn agreements into complete contracts, the linear scalability and effective failure handling. However, it introduces redundancy in the messages during a negotiation.

Rounds	Negotiation Protocol	Agreement Discovery #msg	Agreement Discovery size (kB)	Obligation Disclosure #msg	Obligation Disclosure size (kB)
1	Stateless VersaNeg	7	66	6	528
	Stateful VersaNeg	7	41	6	273
	Separate Message	162	29	9	270/90
3	Stateless VersaNeg	15	419	14	3617
	Stateful VersaNeg	15	145	14	997
	Separate Message	594	107	69	990/330
6	Stateless VersaNeg	27	1449	26	12850
	Stateful VersaNeg	27	301	26	2084
	Separate Message	1242	224	69	2070/690

Table 6.3: *Number of Messages and estimated Bandwidth Consumption for Negotiation Protocols. 4 parties, 3 Requirements at each round, 10kB per Obligation*

One approach to mitigate the bandwidth consumption of Stateless VersaNeg is to optimize the protocol by introducing local state about the messages that have been exchanged in previous rounds during the negotiation. The idea is to turn the stateless protocol into *Stateful VersaNeg*.
Any data in Stateful VersaNeg must only be present for one round to notify all other parties. The complete negotiation can be reconstructed with the local state and the current message.
The results are shown in Table 6.3. One can see that the Stateful VersaNeg protocol has a definite advantage for a large number of rounds. The numbers for the analyzed example show that the stateful approach reduces the bandwidth consumption by nearly 60% for Agreement Discovery and Obligation Disclosure.
However, the functional drawbacks of the stateful approach are that the negotiation states are not comprehensive anymore and do not lead to single contractual agreement documents. Error handling becomes also much more complex. A party that wants to recover a negotiation would need to rely on other parties to reconstruct the current negotiation state.

It is helpful to understand the most extreme opposite approach to put these numbers into perspective. One could imagine that a hypothetical protocol, similar to the one presented in [CWZL05], uses individual small messages for each Requirement, Offer and Obligation. Searching for Offers to Requirements results in flooding of the request to all negotiating parties. This hypothetical protocol is referred to as *Separate Message* protocol. The advantage would be the inherent parallelism through the concurrent protocol exchange. One can assume for this protocol that each party has to have complete information about the agreement at the end.
As one can see in Table 6.3 for Separate Message, the messages themselves are much smaller than with Stateless MPN, but the drawback is the large number of messages. The bandwidth for the Obligation Disclosure is the same for Separate Message and Stateful VersaNeg, because the optimized VersaNeg protocol would also only transmit Obligations to the parties that stated the Requirement. The disadvantage of Separate Message negotiation is the large number of messages and the difficulties to track the dependencies between different messages and to discover how these statements relate. Handling the large number of messages is challenging in case of delayed packet delivery, packet loss, or packet corruption.

Functional arguments are in favor for Stateless VersaNeg, but if the functional drawbacks of introducing Stateful VersaNeg are acceptable, bandwidth consumption can be improved. Stateful VersaNeg is a good compromise between Separate Message and Stateless VersaNeg negotiation, because it does not introduce the very big number of messages of the former protocol and significantly reduces the bandwidth for the latter protocol.

6.3.4 Discussion of Multilateral Agreement Negotiation

Multilateral iterative agreement negotiation is a research field where only few approaches exist. This Section introduced a novel approach that can use agreement negotiation as a tool to discover collaborations on the fly.
Multilateral agreement negotiation allows a negotiation initiator to state goals without the need to anticipate the structure of the agreement beforehand. The negotiation process will search for possible collaborations to achieve these goals. The requirements-driven negotiation paradigm allows each party to model the individual capabilities and demands without the need to understand the whole negotiation. Any multilateral agreement under this protocol can be decomposed into a set of bilateral relationships between Offer/Obligation and Requirement. From a legal perspective, each party must only bother about the bilateral relationships where it is either obligated in some way, or where it expects to receive something. Each party has full power to decide if it enters such a bilateral relationship during the negotiation, or not. This intrinsic property of the VersaNeg approach allows parties to negotiate without understanding all possible dependencies in a multilateral agreement, and still have certainty that the agreement is in its own interest.
Many arguments from the discussion of bilateral VersaNeg agreement negotiation in Section 5.3.6 also apply here. Using the WS-Agreement standard for Requirements, Offers and Obligations is also applicable for multilateral agreements. There is always a surjective relationship that maps an Offer/Obligation to one Requirement within an agreement option. This surjective relationship allows for the representation of the whole VersaNeg agreement as a set of WS-Agreement agreements. The VersaNeg agreement defines the dependencies and the collaborations, whilst the WS-Agreement defines the contents of the agreement.

Collaborations tend to become more risky as the number of involved parties grows. Nowadays, collaborations are often governed by a set of bilateral agreements that together make up the framework for the collaboration. The drawback is that these bilateral agreements are often only known to the two parties. This leads to a situation where no single party has an overview of the hidden dependencies between the collaborators. These hidden dependencies can be a risk that can jeopardize the whole collaboration. The VersaNeg approach makes these dependencies visible. The deciding party in particular has a vested interest to control the risk of the collaboration. An agreement with VersaNeg contains all dependencies and allows the deciding party to analyze how critical the dependencies are for the success of the collaboration. The deciding party can choose an agreement option that has the best ration between risk and utility.

The implementation of VersaNeg shows that this approach is feasible and that bilateral requirements-driven negotiation can be adapted with comparable few changes to multilateral negotiations. It is even possible that parties participate with the same policies in bilateral and multilateral negotiations. For instance, a supplier does not bother during procurement, if it delivers an engine to a car manufacturer as part of a large supplier network, or if it sends the engine directly to a mechanic for repairs. The most important

variable is probably the price that the supplier charges for the engine. The experiments with the prototype show that the implementation scales to large scenarios with many rounds. The ring protocol introduces some redundancy with the comprehensive negotiation states. The additional bandwidth consumption is bearable in an era where subscriber lines in the mass market easily exceed 16MBit/s. The functional advantage of having a comprehensive agreement that fully defines all dependencies outweighs the bandwidth cost.

An open issue in multilateral agreement negotiation in VersaNeg is the point in time until when a legally binding agreement has been reached. After this point, the parties can no longer withdraw from the negotiation. First analysis indicates that under German contract law, parties are already obligated after the deciding party announced its decision, whilst under English contract law parties can withdraw until the agreement is complete. From a network protocol perspective, VersaNeg handles both scenarios equally well. The important difference is the time until when an Abort message will be accepted. Either until the Decision Pending state has been reached, or the Agreement Complete state. The legal interpretation is pending further analysis. In any case, it is advisable to specify the jurisdiction as a Requirement that must be accepted by all parties.

6.4 Secure Multilateral Agreement Negotiation

Achieving secure multilateral negotiations is even more challenging than with bilateral negotiations. In bilateral negotiations one has to assure that the opposite party does not manipulate the negotiation state. Both parties still have a high incentive to find a solution together. In multilateral negotiation, competitors are often also part of the negotiation. This introduces the additional threat that one negotiator forges the negotiation state of its competitor to its own advantage. This section discusses the specific security challenges in multilateral negotiation and describes how to enhance the protocol to make multilateral iterative agreement negotiation secure.

6.4.1 Security Considerations

In a bilateral negotiation, the basic assumption of VersaNeg (see Chapter 5) is that one has to verify the correctness of the previously sent own negotiation state and one has to assure that all changes by the remote negotiator are in compliance with the negotiation protocol. In multi-party negotiations, a negotiation state undergoes many changes by different parties during the ring-traversal. That makes it challenging to distinguish legitimate changes from forgery.

The attacker model for multilateral security includes three main attacker scenarios:

a) Malicious third party that intercepts end-to-end communication between legitimate negotiators in the ring.
b) Malicious negotiator that forges my parts of negotiation state.
c) Malicious negotiator that forges negotiation state that belongs to competitors.

The cases a) and b) are also imminent for bilateral negotiations. The case c) is an additional threat for multilateral negotiation.

Here is a non-exhaustive list of what could happen if there is no security in place for the multilateral negotiation:

1. Negotiator modifies own statements made during previous rounds.
   (a) Remove Offers made during previous rounds.
   (b) Modify attributes of Offer, for instance, insert lower price for Offer.
   (c) Add additional Requirements that make it unlikely that this agreement option can become an agreement.
   (d) Modify conditions of Requirement, for instance, demand a higher price.
2. Negotiator modifies statements of competitors.
   (a) Remove Offers of competitors to have only one agreement option.
   (b) Add Offers with unfavorable terms in name of competitor.
   (c) Add Requirements that cannot be fulfilled for Offer of competitor.
   (d) Remove Requirements to allow for a “cheap” Offer.
   (e) Remove competitor completely from negotiation.
   (f) Add uninvited parties to negotiation.
3. Negotiator can evaluate the Offers of competitors.

   (a) Underbid the competitor by a small margin.
   (b) Learn about cost and conditions from Offers and Requirements of competitor.
4. Negotiator modifies agreement after it has been concluded.
   (a) Modify terms of Offer, for instance, demand a higher price, deliver less than originally promised, deliver at a later time.
   (b) Remove Offers to reduce what a party has promised. Malicious party probably also removes corresponding Requirement.
   (c) Add Offers with unfavorable terms in name of competitor. Malicious party might add a Requirement to justify existence of Offer.

One can see from the list that without security, multilateral iterative agreement negotiation won't happen.

The challenge in multilateral secure negotiation is to verify the correctness and protocol compliance of the negotiation state. It is therefore important to assure the integrity and data origin authentication. Integrity must be guaranteed to assure that the negotiation state has not been manipulated. Additionally, one must be able to identify for each part of the legitimate negotiation state, which made the given change. Non-repudiation of statements made during the negotiation allows the negotiators to proof the outcome of the negotiation to a third party, for instance, to produce evidence of the agreement in front of court.
Multilateral negotiations face the additional challenge to protect the confidentiality during the negotiation. In a negotiation between two parties, there is no need to encrypt data as long as secure end-to-end communication channels are in place. In multilateral negotiations, it must be possible to restrict access to information to a certain audience. For instance, if one party makes a bid with a price for a certain agreement option, it might specify that only the negotiation initiator might see the price, but not the competitors.

6.4.2 Model for Multilateral Negotiation

The model of XML negotiation states, introduced in Section 5.4.3, also applies for multilateral negotiations. A multilateral negotiation also has the property that a negotiation state always grows and information is never removed. One can model a multilateral negotiation as a sequence of matrix additions.
Let the matrix $M_1^{n_1 \times m_1}$ be the Initial Request. Let the matrices $M_j^{n_j \times m_j}$ be the representation of the negotiation state in step j, with $j \in \mathbb{N}^+, n_j \geq n_{j-1}$ and $m_j \geq m_{j-1}$.
Let $R = A, B, C, D, ...$ be the alphabet that names the parties in the negotiation and let $x := \psi(i, j), x \in R$ be the function that returns for each entry in the matrix, whom the entry belongs to. The ownership of entries should is constant during a negotiation as all information is immutable, after it has been added to the negotiation state.
The negotiator that processes the negotiation state at step j describes its new Proposals with the Requirements and Offers with the matrix $N_{j,x}, j \in \mathbb{N}^+, x \in R$. The negotiation state advances from step $j-1$ to state j by $M_j = M_{j-1} \hat{+} N_{j,x}$.

The Figure 6.18 shows an example of one round in a multi-party negotiation. The party A sends the Initial Request $M_1 = N_{1,A}$ to party B. Party B adds some Offers and Requirements $M_2 = M_1 \hat{+} N_{2,B}$ and sends the extended negotiation state to party C. Party C does not make any changes and sends the negotiation state to party D, which again adds some Offers and Requirements $M_4 = M_3 \hat{+} N_{4,D}$. The negotiation state

arrives again at party A and contains all changes made during this round: $M_4 = N_{1,A} \hat{+} N_{2,B} \hat{+} N_{4,D}$. The negotiation continues after this round, but for the discussion of the security, one round is sufficient to explain the concept of iterative message state reduction in multilateral negotiations.

6.4.3 Multilateral Signatures and Iterative Message State Reduction

The Section 5.4.4 introduced the protection of bilateral negotiations with alternating signatures. This mechanism can be extended to cover also multilateral negotiations. The protection of multilateral negotiations works through a novel algorithm where each party applies signatures and uses an iterative message state reduction to verify the correctness of the negotiation state. The basic idea of iterative message state reduction is to rollback the changes of each negotiator step-by-step and a) to verify the data origin and integrity of each portion of the negotiation state and b) to verify the protocol compliance of all additions to the negotiation state. This security algorithm runs at each party in the negotiation for every message that is received.

Let us first consider how signatures are applied. Before a message M_i is sent, the current peer will apply an XML Signature [BBFL+02] covering the whole Negotiation State. This signature includes the history of all Proposals exchanged up to that point. Let $sign(x, sk)$ be the function that computes a public key signature of the negotiation state x with the XML Signature standard and the secret key sk. The peer will append the signature to the message and will send it to the next party in the ring. This party appends its Proposals and signs now the complete message M_{i+1}.
The multilateral security assumes that a Public Key Infrastructure (PKI) is in place that issues cryptographic public keys to the involved parties and that allows each party to verify the identity of the party that applied the signature. PKIs are common nowadays and are no technical obstacle. However, if there is no mutually trusted PKI available, the parties must establish public keys through a secure channel before the negotiation

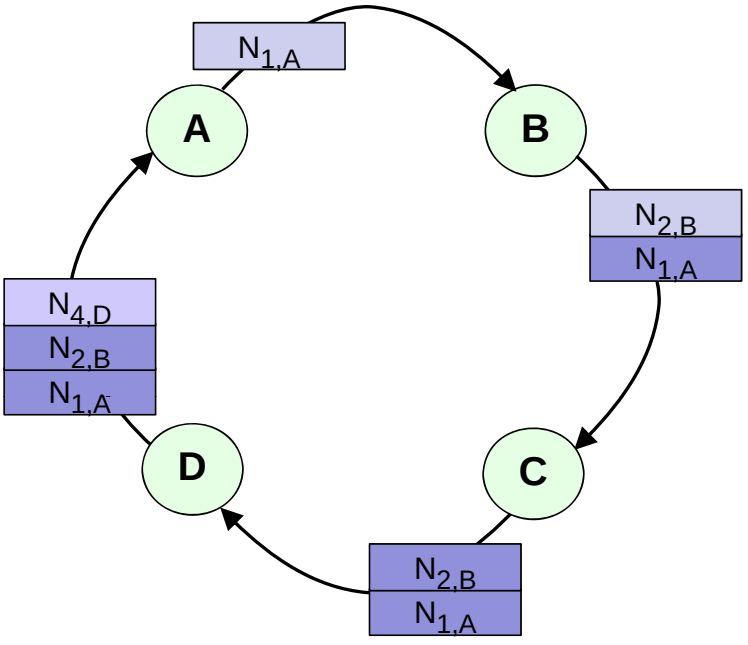

Figure 6.18: *Negotiation States grows in Multilateral Negotiation*

starts.

Figure 6.19 shows how signatures are applied in the example from above. Party A creates $sA := sign(M_1, sk_A)$ and party B creates $sB := sign(M_2, sk_B)$. The signatures are added to an XML structure that contains the sequence of XML Signatures that have been generated during the negotiation. The $sign(...)$ function does not cover the XML structure that contains the signatures. Party C does not add any Proposals, however, even as it only forwards the message it signs the negotiation state $sC := sign(M_2, sk_C)$ and allows other parties to verify that party C has not been skipped. This guarantees that party C had the chance to respond to the negotiation state and it documents that party C chose not to react to the current state. Party D adds its Proposals and signs the resulting message state $sD := sign(M_4, sk_D)$ before it arrives at party A.

So far, we only discussed how digital signatures are created. How does the verification of negotiation states work? The function $VerifyNegotiationState(M_n)$ takes the current negotiation state and returns $true$ if the negotiation state has not been manipulated, $false$ otherwise. The listing in 6.4 shows the algorithm for $VerifyNegotiationState(M_n)$ in pseudo code.

Let $getSignature(M_n, k)$ be the function that takes a negotiation state M_n and returns the signature applied in step k.

The negotiation state contains a numbered list of changed nodes, where each party must declare its additions to the negotiation state. The function $maxStep(M_n)$ returns the number of the last negotiation step. The function $getNewNodes(M_n, i)$ returns a forest of trees that contains all changes that have been declared for step i. The root of each tree is the node that is attached directly to a foreign node. The function $reduceNegState(M_{i+1}, newNodes)$ removes all declared changes $newNodes$ from the negotiation state and thereby undoes the changes from the last negotiation step.

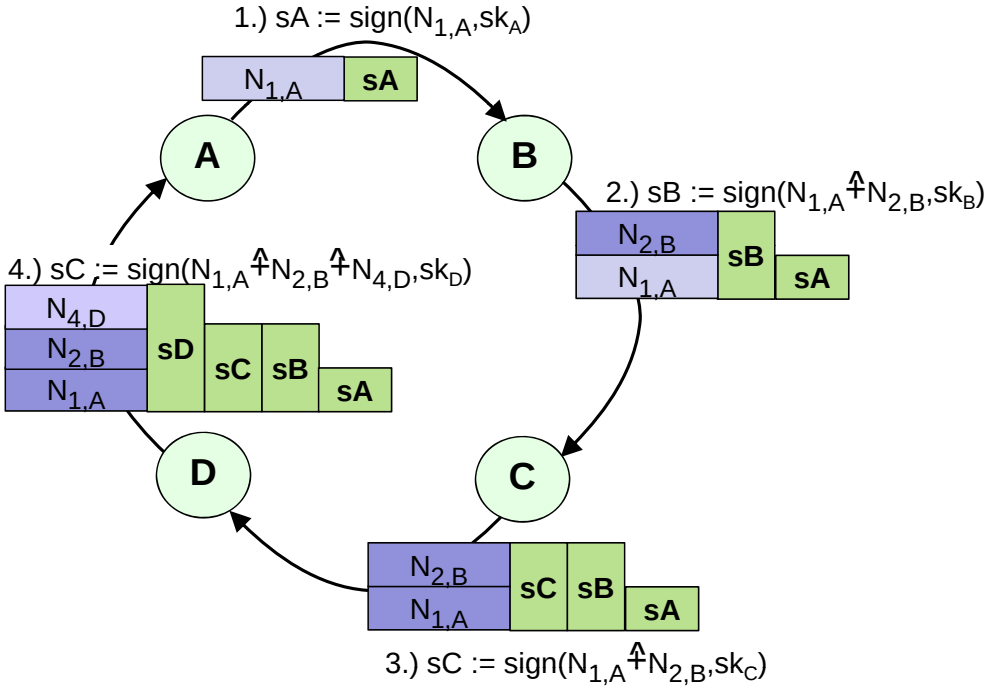

Figure 6.19: *Each Negotiator applies its Signature to the whole Negotiation State*

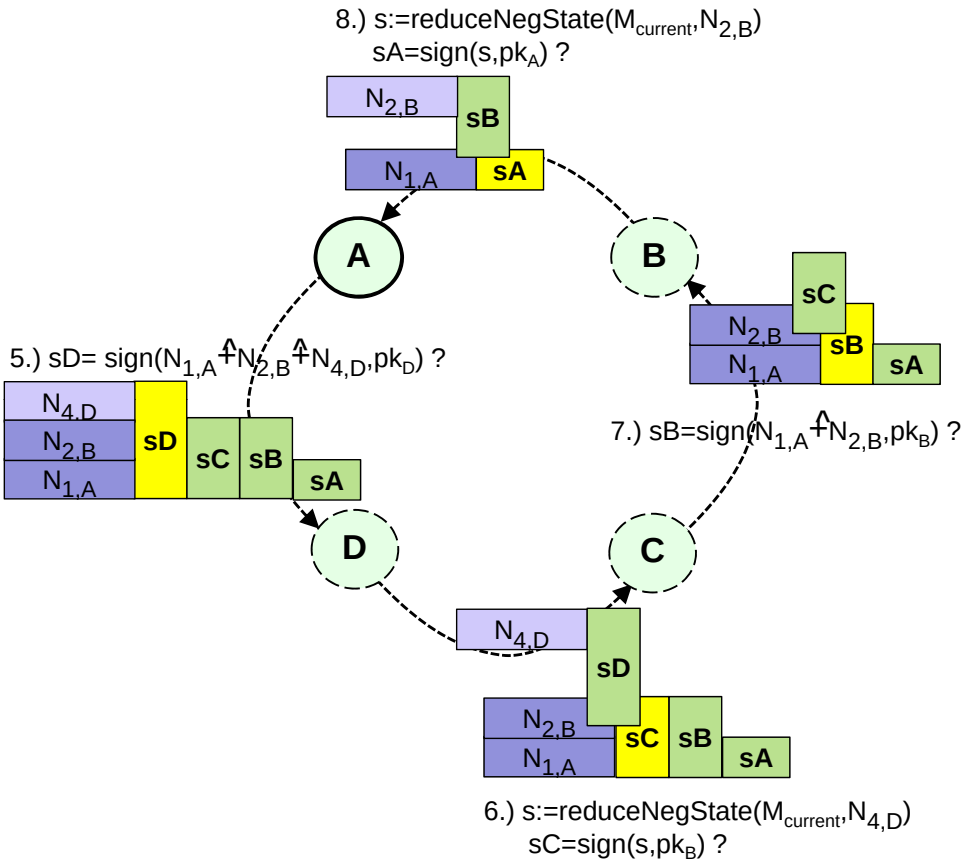

Figure 6.20: *Negotiator A uses Reduce and Verify approach to authorize all Changes made during current Round*

The function $validateProtocolCompliance(M_{current}, M_{previous}, newNodes)$ is crucial to the security. It validates that all declared changes in the document are in accordance with the protocol rules, as described in 6.3. For instance, the algorithm verifies that each Requirement corresponds to an Offer, that Offers satisfy their Requirements, that Audience of Requirements is obeyed, that only permissible parties take part in the negotiation, that each root node of a tree in $newNodes$ is in $CandidatesToAppendTo(M_{previous})$, that the state transitions occur at the right points in the negotiation, and compliance with the XSD Schema definition of the protocol. The $validateProtocolCompliance$ function only returns *true* if all changes by foreign parties are in compliance with the protocol.

```
1  VerifyNegotiationState(M_n):
2      //start with the newest signature
3      M_current := M_n
4      for i := maxStep(M_n) − 1 to 1 do {
5          //get nodes that have been added in last step
6          newNodes := getNewNodes(M_current)
7
8          //check signature of current step and that
9          //signature matches ownership of new nodes
10         if ( (getSignature(M_current, i) != sign(M_current, pk_owner(M_current)))
11               or (owner(getSignature(M_current, i)) != owner(newNodes)) ) {
12             handleInvalidSignature (...) ; //abort negotiation
13             RETURN false
14         }
```

```
15
16        //optimization: may exit verification if my signature is found
17        if (owner(M_current = myself) {
18            //trust chain: all other steps have been verified in previous rounds
19            RETURN true
20        }
21
22        //reduce negotiation state to state of previous step
23        if (i > 1) {
24            M_previous := reduceNegState(M_current, newNodes)
25
26            //verify that all changes in newNodes are protocol compliant
27            if not validateProtocolCompliance(M_current, M_previous, newNodes) {
28               handleNonCompliance(...); //abort negotiation
29               RETURN false
30            }
31            M_current := M_previous
32        }
33    }
34
35    //reached initial request and all verifications were successful
36    RETURN true
```

Listing 6.4: *Algorithm for iterative message state reduction and signature verification*

The algorithm from listing 6.4 goes through all signatures until it either arrives at the Initial Request or until it finds its own signature. An own signature is only present if this party verified the message successfully in a previous round.
The algorithm first verifies the digital signature with the public key of the party that processed the negotiation state in this step. It calculates the signature of each XML representation of a current step and verifies that it is the same as the signature that has been attached for this step. Additionally, this party assures that each $newNode$ from this step has been signed by the owner of the signature. This check assures the integrity and data origin of the changes in this negotiation state. By verifying the digital signature one knows that the complete negotiation state is the one that the negotiator has signed and has sent in step i.

One must now reduce the negotiation state to obtain M_{i-1}. Each change that has not been declared in $newNodes$ will not be removed by $reduceNegState()$. Consequently, the signature validation of M_{i-1} will fail. The $reduceNegState()$ together with the consecutive signature validation enforces that each new node must be declared.
The final verification for this step is now to check with $validateProtocolCompliance()$ that all $newNodes$ have been added in compliance with the protocol. For example, to assure that an Offer has a corresponding Requirement.

The Figure 6.20 shows how the reduce and verify approach applies to the example from above. Let us consider party A that receives the negotiation state M_4 after one round. It first verifies that the signature sD from party D covers the whole negotiation state M_4. Only after this check succeeds, the algorithm examines the changes $N_{4,D}$ from party D. It reduces the negotiation state to M_3 by removing all declared changes $N_{4,D}$ and verifies the protocol compliance of all changes in $N_{4,D}$.
Now party A knows that the negotiation state from party D is correct and it can continue to check the negotiation state from party C. Party C did not make any changes to the negotiation state, hence one only has to verify the digital signature of M_3.

Next the algorithm verifies the digital signature sB of M_2 and verifies protocol compliance and reduces the negotiation state to M_1. For M_1 it is sufficient to verify only the own signature SA as it is the negotiation state that party A has previously sent.

Keep in mind that these verifications have all been carried out by party A. By doing so, this party backtracked the negotiation until it arrived at a negotiation state that could be considered as trustworthy by party A. Each party in the negotiation executes this algorithm to verify the negotiation state. Party D, for instance, would check the negotiation states M_3, M_2, M_1.
If one party manipulates the negotiation state and all parties execute $VerifyNegotiationState$, the forgery is detected by the next party after the malicious party in the ring. This algorithm is robust even if multiple parties collaborate to forge the document. For instance, if party B and party D collaborate to skip party C maliciously, the missing signature would be detected by party A. As soon as a manipulated negotiation state arrives at a sincere party, this party can detect the compromise of the negotiation state.
The iterative message state reduction guarantees that for all changes to the negotiation state, the a) integrity b) data origin authentication, and c) protocol compliance is assured. The algorithm thereby detects a compromise of the negotiation state. The $VerifyNegotiationState$ algorithm has been implemented to verify the viability of the reduce and verify approach. As a rule of thumb, the digital signature nearly doubles the negotiation time in the experiments in this thesis. However, it cannot proof who is guilty of a forgery, it can only proof who declared and signed which nodes of a valid negotiation state.

6.4.4 Discussion of Multilateral Security with Iterative Message State Reduction

This section introduced iterative message state reduction to secure negotiation states during the negotiation in the ring. The security algorithm detects if information in negotiation states has been altered or removed and it discovers if one party has been skipped. Parties that do not comply with the protocol can be excluded from future negotiations.
Each party in the negotiation can be sure that only authentic statements are part of the negotiation. A third party that was not part of the negotiation, can verify the statements and the interrelationship of Requirements and Offers in an agreement. The order of the parties in the ring assures that the security algorithm discovers if one party has been left out. This security check enforces that each party receives the current negotiation state and has the chance to learn about new Proposals.
The implementation of the algorithm proofs the viability and performance of the approach. XML Digital Signature with public key cryptography is the underlying cryptographic mechanism. XML Digital Signature relies on XML Canonicalization [Boye01]. This standard transforms the memory document object model into a canonical form that allows to generate and validate signature. The canonical representation of the negotiation state is generated completely anew from the document object model. Only with canonical XML, the reduce algorithm with its extensive modifications of the negotiation state, becomes feasible.
For the security in bilateral negotiation, one can benefit from the better performance of symmetric key cryptography with HMAC (see Section 5.4.5.2). Multilateral security

works only with public key cryptography. The advantage is that public key cryptography allows for an implementation with Advanced Electronic Signatures (see Section 2.1.4) that allows to proof agreements in front of court.

The integration of encryption algorithms with the negotiation protocol is future work, but should integrate easily with the XML negotiation states. There are many options how to integrate encryption into multilateral negotiations. Public key encryption can be applied to restrict access to the details within Requirements, Offers, and Obligations, to a certain party. One encrypts the sensitive information with the public key of the receiver. Group cryptography [RaHu03] can handle cases more efficiently, where more than one party should be able to decrypt sensitive information. Attribute based encryption [GPSW06] protects the information not for individual receivers, but based on properties of the audience. For instance, suppliers might have a cryptographic attribute that designates them as a supplier in this context. In open negotiations, where negotiators might join even during the negotiation, one can specify that suppliers can access certain Requirements, without knowing the identity of the suppliers beforehand.

Another important property of the negotiation is the point when an agreement has been reached. The fundamental idea is that the contracting process produces a "meeting of minds", refer to Sections 2.1.2 and 2.1.3 for further details. The multilateral agreement protocol in this thesis produces a "meeting of minds" through the strict matching of Requirement with Offer. A party states with the Requirement that it is willing to accept any Offer under the conditions of the Requirement. The offeror states the willingness to fulfill this Requirement. In an agreement option, each Requirement is satisfied by at least one Offer, hence there is consent between all involved parties about the agreement. The negotiation protocol has similarities with auctions and stock markets where one is also bound to a bid made as soon as someone accepts the bid.
Sometimes it is desirable to introduce one additional round to re-confirm the willingness of the parties to conclude an agreement. Fair-exchange protocols [GaMa99, Baum01, PaGä99] can achieve fair contract singing in multilateral agreements. These techniques can be integrated as an additional phase to conclude the contract between all parties in the chosen agreement option. A number of properties are relevant for multilateral agreements: a) fairness: if I sign and send a contract, I will receive a signed contract from all other parties b) timeliness: no endless waiting for signed contract c) exclusion freeness: at the end of protocol, any excluded participant can proof that it has been excluded. The drawback is that these protocols need a large number of rounds and messages to converge on one agreement.

6.5 Summary

Multi-party agreement negotiation becomes increasingly important to enable service cooperations across different domains. It allows for the dynamic establishment of Requirements and Obligations that govern inter-domain service interactions. VersaNeg is a novel protocol for requirements-driven multi-party agreement negotiation that can be integrated seamlessly into web service business processes. All that is required is an accurate and verifiable definition of Requirements, Offers, and Obligations shared between the negotiating parties. This proposed protocol can reuse language elements from related standards and enables much more dynamic and scalable negotiations with only a few novel language constructs. The analysis of the proposed language for multi-party agreements showed its ability to express complex agreements and how different parties can collaborate to achieve common goals. The experiments with the prototype showed that the performance of the protocol allows for the integration with real world business processes.

In theory, there is no difference
between theory and practice.
But, in practice, there is.

Jan L. A. van de Snepscheu

7. VersaNeg Implementation

The Objective of this Chapter is to give additional details about the implementation. There are many freedom degrees on how to realize failure handling or the internal software architecture. This Chapter introduces the realization of VersaNeg, alternative architectures are not discussed here.

7.1 Failure Handling

One advantage of our comprehensive negotiation states is the ability to recover from failures by processing the last negotiation state again. Below are diagrams that show the distributed recovery protocol. A simplistic view of the negotiation process is that one party receives a negotiation state, processes the state and sends the new state to the next party. Different failure conditions might occur:

1. Broken connection
2. Corruption of negotiation state
3. Pending negotiation does not terminate

7.1.1 Detection of Failure Conditions

Each of these failure conditions can be detected via different means.
The implementation relies on TCP for the transport and uses TCP keepalive which periodically sends probe packets with no payload and waits for TCP ACKs. A *broken connection* or timeout on connect is signaled by the socket API of the operating system then. A broken connection can be detected by the sender or the receiver of a negotiation message, depending on which party fails.
However, a party may even fail after it received the complete message. It is therefore important to have additional application level acknowledgments that are sent after a party finished the processing of the message. Let us assume a sequence of three parties p_0, p_1, p_2 in the ring. The party p_1 that processes a message, must send an application level acknowledgment back to p_0, after it finished processing and successfully sent its message to the next party p_2. The acknowledgment is required to extend this simple failure detection mechanism to cover also the duration of the message processing.

A *corruption of the negotiation state* can be detected via the integrity verification and the compliance check of the security. Corruptions can stem from different origins, for instance, faulty hardware at sender or receiver.

The first approach to *broken connections* or *corruption of negotiation state* is a bilateral algorithm to recover from the failure (see Section 7.1.2). Only if the failure can not be recovered via the bilateral algorithm, the *deciding party* takes over to either recover the negotiation or to abort the negotiation.
One can expect that the negotiating parties want to complete negotiations and therefore abide to the protocol. Hence, the probability that multiple parties fail at the same time is comparably small and most failures can be handled via the bilateral recovery protocol.

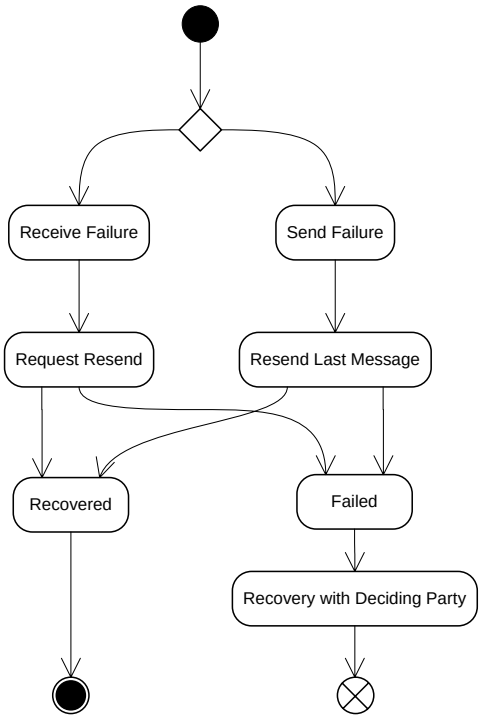

Figure 7.1: *Bilateral Recovery Protocol*

Sometimes a *pending negotiation does not terminate*. This is the case if one party in a negotiation does neither receive negotiation states to continue the negotiation, nor receives a *Failed* or *Final Agreement* negotiation state that concludes a negotiation. Failures of direct neighbors can be handled as the *broken connection* case or the *corruption of negotiation state* case. More difficult are situations where one or more parties fail that are no direct neighbors of the party. In this case, a timeout mechanism detects a *pending negotiation that does not terminate* and initiates a recovery protocol with the *deciding party* of the negotiation (see Section 7.1.3).

Of course malicious parties could try to break ongoing negotiations out of self-interest. Eventually, such a negotiation with an uncooperative party would be handled via the deciding party as a *pending negotiation that does not terminate*. If possible the negotiation would continue without the uncooperative party or the negotiation would have to be restarted.

The recovery protocols assume that each party keeps the last negotiation message in a negotiation until the negotiation terminates. The negotiation messages from the

previous rounds can be discarded. Notice that, according to the definition, the servers can still be considered as *stateless servers* because they do not need these previous messages for the processing of current messages during the negotiation. This soft state is only needed to recover other failed negotiation servers.

7.1.2 Bilateral Recovery Protocol

After one party has failed, the failure will be most probably be detected as a *broken connection*. Sometimes the failed party might be able to recover, for instance, by an automatic restart or by a fail-over server taking over. The VersaNeg protocol tries first a bilateral recovery protocol because it can thereby restore negotiations without any changes to the available agreement options.

The *bilateral recovery protocol* works by reprocessing the last message again that was completely processed. The state diagram of the algorithm is shown in Figure 7.1.
If a sender detects the broken connection, it can simply try to send the last message again a number of times. To avoid overloading the failed party, a random backoff mechanism determines when the last message is.
If a party p_2 was supposed to receive a message, it can contact the party p_0 that is the predecessor of the failed party p_1 to send its last message again.
If the bilateral recovery protocol cannot restore a negotiation after a number of attempts to resend the message, the *recovery protocol with deciding party* takes over.

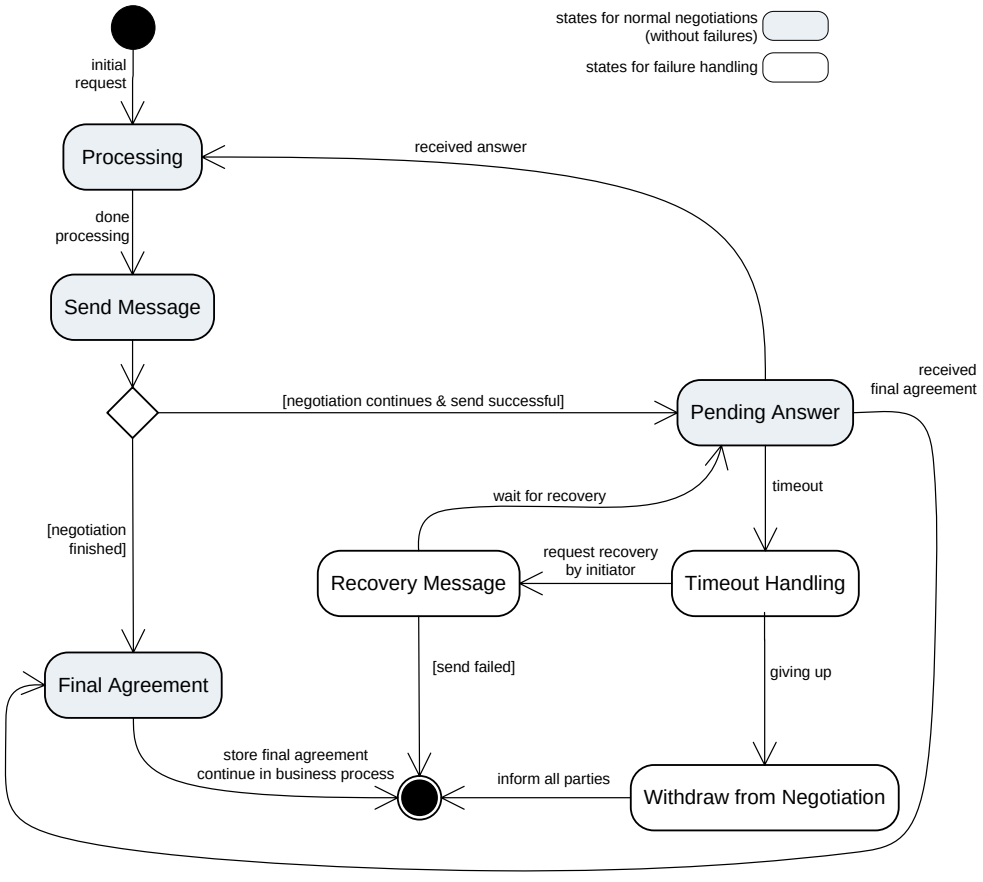

Figure 7.2: *Recovery and Abort Protocol at Peer*

7.1.3 Recovery Protocol with Deciding Party

The *recovery protocol with deciding party* is invoked if one party fails and cannot recover, if more than two parties fail that are neighbors (e.g. network outage), or if a malicious party wants to interrupt the negotiation. This recovery protocol is invoked either after an unsuccessful bilateral recovery or after a *pending negotiation does not terminate*.
Figure 7.2 shows the state machine at a peer with failure detection. The normal processing sequence traverses the states *Processing* for the handling of a received message, *Send Message* for distributing the resulting message and *Pending Answer* for waiting for a continuation of the negotiation. The *Pending Answer* state waits for any negotiation message that advances the negotiation further than the state of the last message that was sent. A successful negotiation arrives eventually at the *Final Agreement* state.

Failure handling now works through a timeout mechanism in the *Pending Answer* state. If there is no answer until the timer expires, the party sends a recovery message to the deciding party. The timeout in the *Pending Answer* state consists of a minimum timeout plus a random value to avoid initiation of the recovery algorithm by multiple parties concurrently.
After a timeout has occurred the party initiates the recovery protocol with deciding party by sending a Recovery Request to the *deciding party*. If the party that detected the failure cannot contact the *deciding party* it goes directly into the *Withdraw from Negotiation* state. It will inform all parties in the negotiation that it has withdrawn and will not continue with the negotiation even if some message belonging to this negotiation arrives. It will also go into the *Withdraw from Negotiation* state if the *deciding party* does not recover the negotiation in a reasonable time.

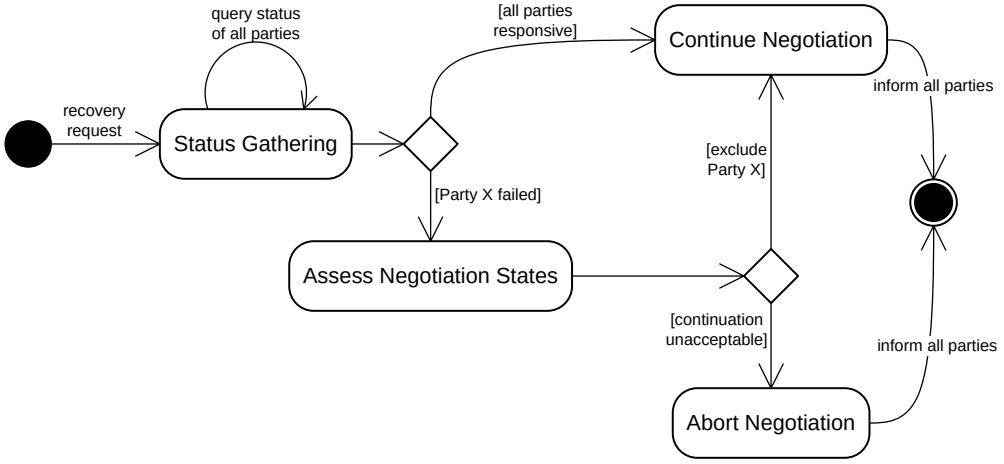

Figure 7.3: *Recovery Protocol at Deciding Party*

When the deciding party receives a recovery request, it has to take a decision if it is reasonable to continue a negotiation without a failed party p_x or if it should terminate the negotiation (see Figure 7.3). If there is no deciding party, one agreed upon failure handling party can take the role of the deciding party for failure recovery. Only the first recovery request will be processed as long as there is another active recovery for this negotiation.

The deciding party requests the last negotiation message from all parties to determine the most current negotiation state and also to discover which parties are still responsive.

It depends on the negotiation state and the available agreement options if it is reasonable to continue with a negotiation. During the *Agreement Discovery* phase, the deciding party determines all agreement options that depend on the failed party p_x and are therefore not possible anymore. It weights these broken paths against the still available agreement options that work without p_x. If p_x is a comparably unimportant party to the negotiation and the negotiation already involved many parties and led to a complex negotiation state, it might be preferable to continue. In this case, the deciding party marks all paths in the agreement state as broken that contain p_x by appending a *broken node* as a leaf. The deciding party will inform all remaining parties that the negotiation continues and will send the negotiation state to the next party in the ring.

If it makes no sense to continue a negotiation, the deciding party informs all parties that the negotiation is aborted. Terminated negotiations might be restarted as a new negotiation by the original negotiation initiator.

7.1.4 Withdraw Protocol with Deciding Party

Sometimes negotiating parties want to withdraw from an ongoing negotiation. A party might only withdraw as long the negotiation is in the *Initial Request* or *Agreement Discovery* state. During these states a negotiating party could otherwise simply state unfulfillable Requirements to assure that it will not become part of the agreement. During the remaining negotiation states the party p_x is already committed to its offers if the associated agreement option gets chosen.

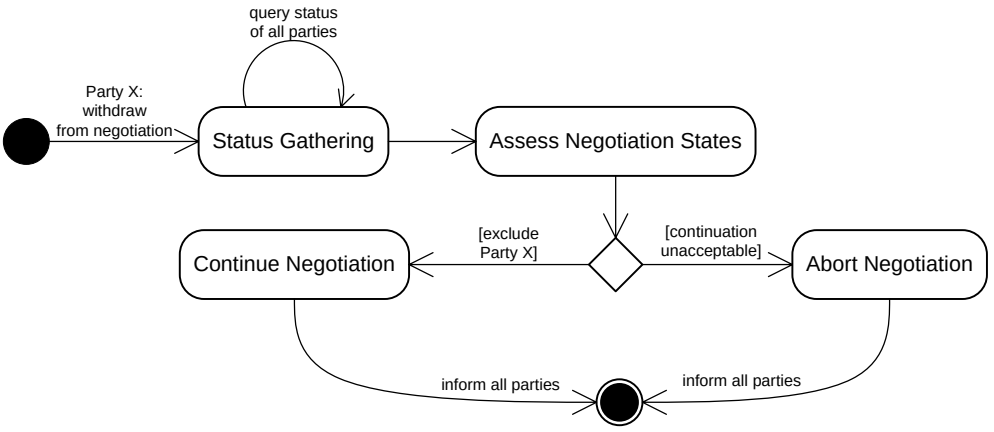

Figure 7.4: *Abort Protocol at Deciding Party*

The *withdraw protocol with deciding party* basically works as the *recovery protocol with deciding party*. Depending on the impact of the withdrawal of p_x on the available agreement options, it might be possible to continue the negotiation or it might be preferable to terminate the negotiation.

7.2 WS-Agreement for VersaNeg Proposals

The WS-Agreement specification [ACDK+07] has been released by the Grid Resource Allocation Agreement Protocol Working Group[1]. WS-Agreement is highly relevant to research and more than 300 scientific publications reference this standard. One of the reasons is that the WS-Agreement specification contains an extensive definition of

[1] http://forge.gridforum.org/projects/graap-wg

language elements that are commonly found in Service Level Agreements (SLA). The WS-Agreement Language is extensible to include related standards, for instance, the Job Submission Description Language (JSDL) [ABDF+05] for the description of job submissions to the grid. This enables research to focus on specific extension to the WS-Agreement standard, instead of defining complete agreement languages anew.
For this thesis, WS-Agreement is very important for several reasons. The WS-Agreement is also driven by the need to form comprehensive agreements that specify not only price but also the complex dependencies that contracts consist of. It is a language that can express many real world agreements. For this thesis, it serves as a reference point to explain the functional differences between iterative requirements-driven agreement negotiation and the template-based negotiation with WS-Agreement. The VersaNeg approach does not exclude using WS-Agreement. In fact, VersaNeg can use WS-Agreement as an agreement language for its Requirements, Offers, and Obligations. Therefore, this thesis does not have to develop an own agreement language, it can embed the WS-Agreement language. This embedding creates the opportunity that research work on WS-Agreement is also applicable for VersaNeg, for instance, on semantic agreements [OVSH06].
The first part of this section describes the language constructs of WS-Agreement. The next part describes how WS-Agreement can serve as the language for Requirements, Offers, and Obligations.

7.2.1 WS-Agreement Language Constructs

The objective of WS-Agreement is to establish agreements between two parties. The WS-Agreement standard relies on a template-based negotiation. One party provides a *Template*, which will be turned into an *AgreementOffer*, and eventually arrives at the *Agreement*. The WS-Agreement standard relies on a common set of language constructs for defining the agreement during all three states.

Figure 7.5 shows the language constructs that are common to Template, AgreementOffer, and Agreement. Each agreement has an optional `Name` element that may be used to describe the agreement in a human-understandable way. The `AgreementContext` contains elements to describe the two parties in the agreement, and it may contain an `ExpirationTime` for the whole agreement. The `TemplateID` must be set if the AgreementOffer is based on a Template. The `ServiceProvider` states which of the two parties is obligated to provide the service.

The agreement `Terms` are at the heart of the standard. The language states: *"A term expresses the defined consensus or obligations of a party"*. There are two different types of Terms: The `ServiceDescriptionTerm` elements contain the information, necessary to instantiate a service. The `GuaranteeTerm` elements define the service levels that may be monitored and enforced by a management system.
The WS-Agreement standard defines a nested tree structure of Terms with the `All`, `OneOrMore`, and `ExactlyOne` elements. This nested tree structure is used to express choices, the same way as with WS-Policy [VOHH+07] and as described in this thesis in Chapters 5 and 6. The important difference to requirements-driven negotiation is that with WS-Agreement, one can only decide between different options in the template. The protocol does not provide a structured mechanism to discover new agreement options. As stated before, the Terms element can contain ServiceDescriptionTerm and GuaranteeTerm elements. The nested structure therefore describes different choices for the service provisioning (ServiceDescriptionTerm) and the consequences and conditions of choosing certain options (GuaranteeTerm).

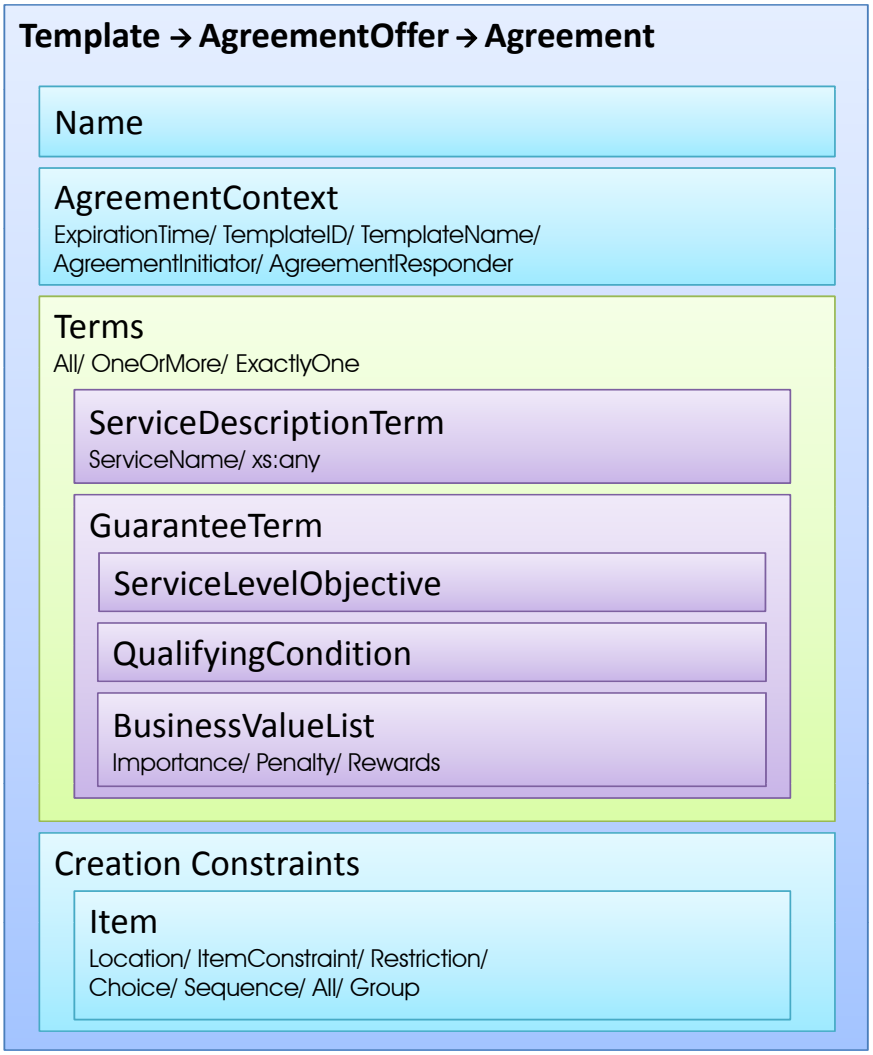

Figure 7.5: *WS-Agreement Language Constructs*

The service description is inside the `ServiceDescriptionTerm`. This service may already exist or this service has to be created as specified here. ServiceDescriptionTerm can embed any language XML construct to describe the service. The standard gives examples how to embed JSDL in ServiceDescriptionTerm elements to describe services in the grid domain.
Another elements for describing the service is the `ServiceReference` to reference an existing service. By providing a ServiceReference, the parties can avoid to describe the service, as WS-Agreement assumes that both parties share the same understanding of this service. The `ServiceProperties` can be used to describe measurable variables that characterize the service.

`GuaranteeTerm` elements define an *"assurance on service quality, associated with the service described by the service definition terms"*. It contains a `ServiceScope` element to reference one or more services that this GuaranteeTerm belongs to. The `Obligated` element defines which party is obligated to fulfill the GuaranteeTerm. The `ServiceLevelObjective` is used to state an *"assertion over service attributes and/or external factors such as date, time"*, most often in the form of Key Performance Indicators, such as response time and completion time. The `QualifyingCondition` is an assertion over external factors, for instance, the service request rate by the client.

The `BusinessValueList` describes the consequences of being compliant of the consequences of violating the GuaranteeTerm. The `Importance` element defines the relative importance of meeting an objective. The `Reward` describes what a party gains by fulfilling an objective. The `Penalty` element describes what compensation a party has to provide in case it does not meet the objective.

The `CreationConstraints` element is very important because it describes the permissible values to fill into the Template to create AgreementOffer and Agreement. The WS-Agreement standard even allows for Agreements have little in common with any Template. However, it is not clear from the standard, how automated systems should deal with Agreements that do not comply with any Template. For this thesis, automated processing of agreements is relevant. Consequently, only AgreementOffers and Agreements that match a Template and the CreationConstraints are of interest.
The CreationConstraints element can have a number of `Item` elements which describe the constraints. Each `Item` uses a `Location` to reference Terms in the Agreement, for instance, with XPATH. The `ItemConstraint` can contain an XML Schema `restriction`. This restriction must be acknowledged, for inserting a value at the specified location. The `Choice`, `Sequence`, `All`, `Group` elements from XML Schema can also be used here, to further define the permissible values, pointed to by Location.

7.2.2 WS-Agreement for Requirements, Offers, and Obligations

As one can see from the previous section, WS-Agreement is an expressive agreement language that can capture many real world agreements. That makes it desirable for VersaNeg to integrate WS-Agreement as an agreement language. VersaNeg is capable of forming multilateral agreements. Even despite WS-Agreement does only handle one-to-one relationships, the WS-Agreement language can be integrated in multilateral agreements. The reason is that the underlying concept of multi-party agreements with VersaNeg is based on the surjective relationships between the Offers/Obligations and their respective Requirement within an agreement option.

VersaNeg has three basic constructs for Proposals: the Requirement describes what a party wants, the Offer what a party promises, and the Obligation what a party is obliged to do. WS-Agreement has a transition from Template, to AgreementOffer, to Agreement. One can embed WS-Agreement in VersaNeg by using Template as Requirement, AgreementOffer as Offer, and Agreement as Obligation. The WS-Agreement is responsible then for defining services and detailed options for a service. Even more important is that WS-Agreement can define the Rewards and Penalties for services. The VersaNeg is responsible for Agreement Discovery and Agreement Formation to establish unambiguous agreements between multiple parties by defining dependencies between Proposals.

The WS-Agreement Template has large similarities with the VersaNeg Requirement. It clearly states what kind of service is requested but it gives the opposite parties choices for how to satisfy the service request. The Template with its Terms and CreationConstraints clearly state what permissible AgreementOffers can be. Parties usually make offers to be entitled to the Rewards contained in the BusinessValueList. The Reward itself might just be discovered during the negotiation, for instance, the price might be set by the service provider as part of the Offer. The party that stated the Requirement can choose the Offer with the lowest price.

The VersaNeg Offer can be mapped to the WS-Agreement AgreementOffer. The Offer is not binding during Agreement Discovery. In fact, even if a single AgreementOffer is valid, it might still be on a path in the VersaNeg dependency tree, where it does not become part of an agreement option, because some other Requirements on this path cannot be satisfied.
VersaNeg requires that that all CreationConstraints that are relevant for an AgreementOffer, are obeyed. This property shall enable VersaNeg to verify AgreementOffers automatically for compliance with the WS-Agreement Template.

When VersaNeg forms the agreement, it turns the Offers into Obligations. This can interpreted as the WS-Agreement transition from AgreementOffer to Agreement. With VersaNeg, the WS-Agreement Agreement might contain additional details that were not present in the AgreementOffer. The VersaNeg requirement is that all relevant CreationConstraints of the original Template are satisfied in the Agreement.

For embedding WS-Agreement into VersaNeg agreements, one has to take care that the WS-Agreement is only valid inside a VersaNeg agreement. It must be obvious from the Template, the AgreementOffer, and the Agreement that the embedded WS-Agreement is only valid if it is part of a VersaNeg agreement. One can use the Context section of WS-Agreement to reference the VersaNeg agreement and to state that only if this WS-Agreement is part of the VersaNeg agreement, that the WS-Agreement is considered valid. Otherwise, no party is bound by the WS-Agreement.
The Appendix C contains some examples of embedded WS-Agreement in a VersaNeg negotiation state. Refer to Listing C.1 for an example on how the WS-Agreement Template serves as Requirement. Listing C.2 shows AgreementOffer as VersaNeg Offer and Listing C.3 shows the WS-Agreement Agreement as VersaNeg Obligation.

7.3 Negotiation Server

This Section describes architectural features and implementation details of the prototypical realization of VersaNeg.

7.3.1 Multi Threaded Negotiation Server

The negotiation is handled by a stand-alone multi-threaded server with XML messaging interfaces. The time sequence diagram in Figure 7.6 shows the processing of an incoming negotiation message. There are three objects that handle incoming messages before the negotiation logic itself starts processing of the message. The *NegotiationHandler* instantiates different objects and passes TCP socket handles to the *Communicator*. The Communicator reads the message from the TCP stream, and uses an XML parser to obtain the DOM representation of the XML document. Next are a number of verification steps that must succeed or the negotiation fails. The *Security* module first verifies the integrity and authenticity of the received message by verifying the XML signatures (see Section 5.4 and Section 6.4). The next test verifies if the XML message complies with the interaction protocol and performs several tests, amongst others, XML schema validation, ownership check of nodes, and verifies logical correctness. After all checks completed successfully, the *NegotiationLogic* processes the message as described in Section 5.3.3 and in Section 6.3.2. After the *NegotiationLogic* completes processing a message, it returns with a negotiation message that will first be signed by the *Security* and that has then to be sent by the *Communicator* to the next party in the negotiation.

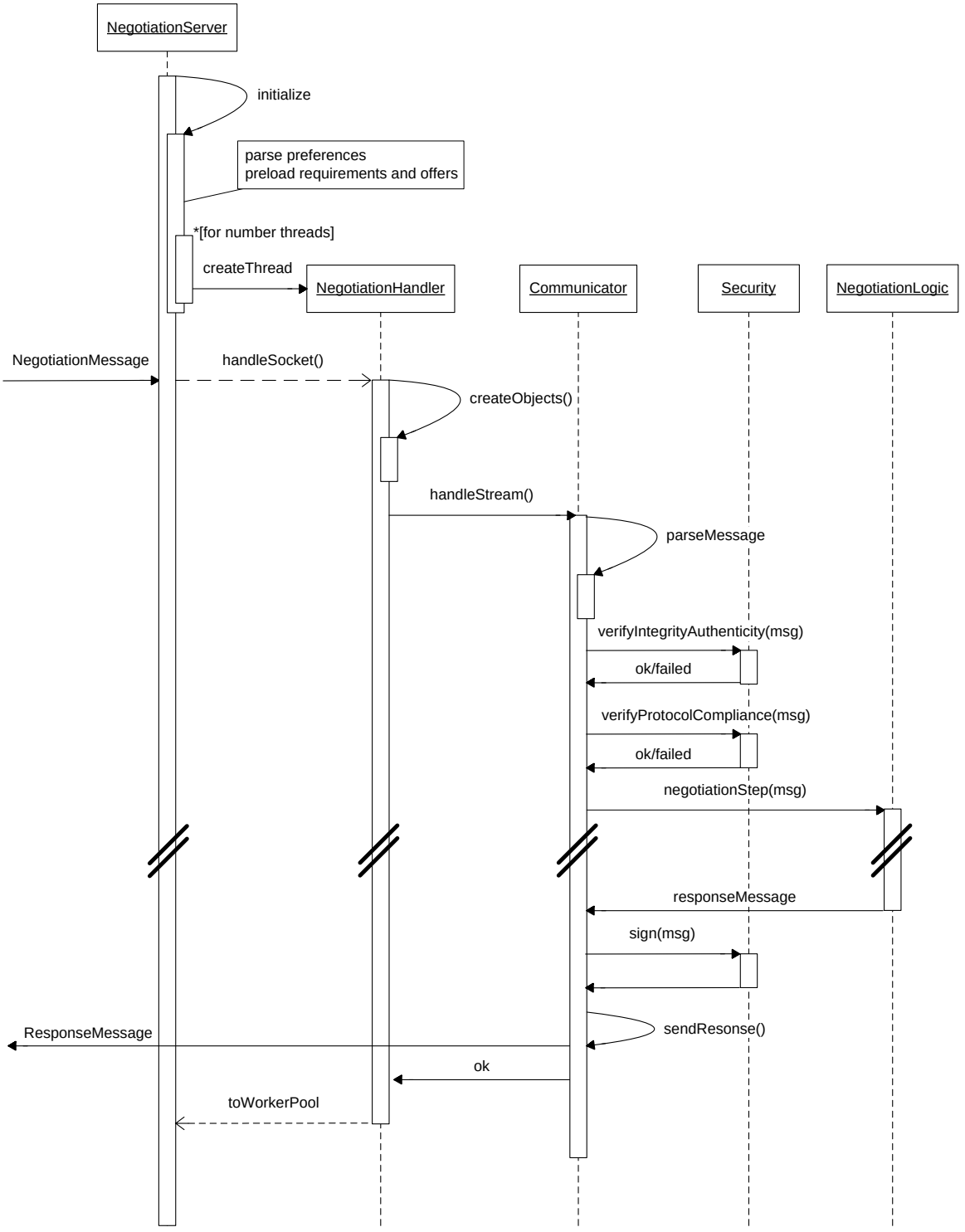

Figure 7.6: *Time Sequence Diagram for handling Negotiation Messages at Server*

The implementation relies on C++, libxmlsec[2], OpenSSL[3] and the Advanced Crypto Software Collection[4]. The code was extensively instrumented for regression testing, logging of program behavior, and performance analysis.

[2]http://www.aleksey.com/xmlsec
[3]http://www.openssl.org
[4]http://acsc.cs.utexas.edu

To save time for allocating new threads, the server pre-allocates a number of threads and uses a synchronized queue to pass incoming connections to the worker threads. The implementation of the bilateral negotiation protocol relied on one persistent TCP connection with one thread for the whole negotiation. This thread would receive a negotiation message, process the message and send the new negotiation message back. For the multilateral negotiation, each message is handled by a separate thread. The reason is that the current status of the negotiation determines who should receive the next message during the negotiation. The party that sent us the message is different from the party that we should send our new message to. With features such as direct send, persistent connections that last for whole negotiations do not work anymore.

7.3.2 XML Processing

The DOM representation of the XML messages allows complex queries and extensive manipulations of XML messages. The XML Schema validation of the protocol messages with the Schema in the Appendix D in Listing lst:mpn:ProtocolSchema assures that the DOM representation can be transferred into an internal tree of objects for the implementation of the *VersaNeg* protocol.

A number of classes encapsulate the functions for XML processing to separate the negotiation logic from the XML API and to provide a uniform logger support.
The *BaseRootNode* is the base class of many derived classes and provides generic accessor functions to the negotiation state. The basic concept of the implementation is a one-to-one correspondence of XML elements and C++ classes. The constructor takes an *XML Element* that the methods of the class work with. All higher level functions access the XML negotiation state through derived classes from the `BaseRootNode`.

The classes that have `RootNode` as part of their name take one XML element and give access to the child elements and descendants. The `MessageRootNode` class takes the DOM root node of the XML message. It provides accessor functions to the classes that represent children of the `AgreementNegotiation` root node.
The `AgreementTermsRootNode` class works on the `Terms` XML element. It is a container for the `ConditionTermNode` tree structure and the `ChangedNodes`. The `ConditionTermNode`s form the dependencies between different requirements and offers and represent `All`, `ExactlyOne`, `OneOrMore`, and `OnlyOne` XML Elements. The `ChangedNodes` keep track of all new requirements and offers in one round.
The `ServiceDescriptionNode` is the container for the `Requirement` and `Obligation` elements. The ServiceDescriptionNode only contains Requirement elements during the Discovery Phase. After the Obligation Details Disclosure Phase, all Requirements are satisfied by at least one Obligation. The Agreement is complete then.

The ConditionTermNode elements, which contain the logical dependencies between requirements, are separated from the ServiceDescriptionNode elements which contain the Requirements and the attached Obligations. This separation has several advantages. The tree of the `ConditionTermNode` focuses on the logical dependencies and provides an easily accessible view on the different agreement options for humans. The details of Requirement specifications and Obligations are stored separately. Another advantage is that different ConditionTermNodes can reference the same ServiceDescriptionNodes.

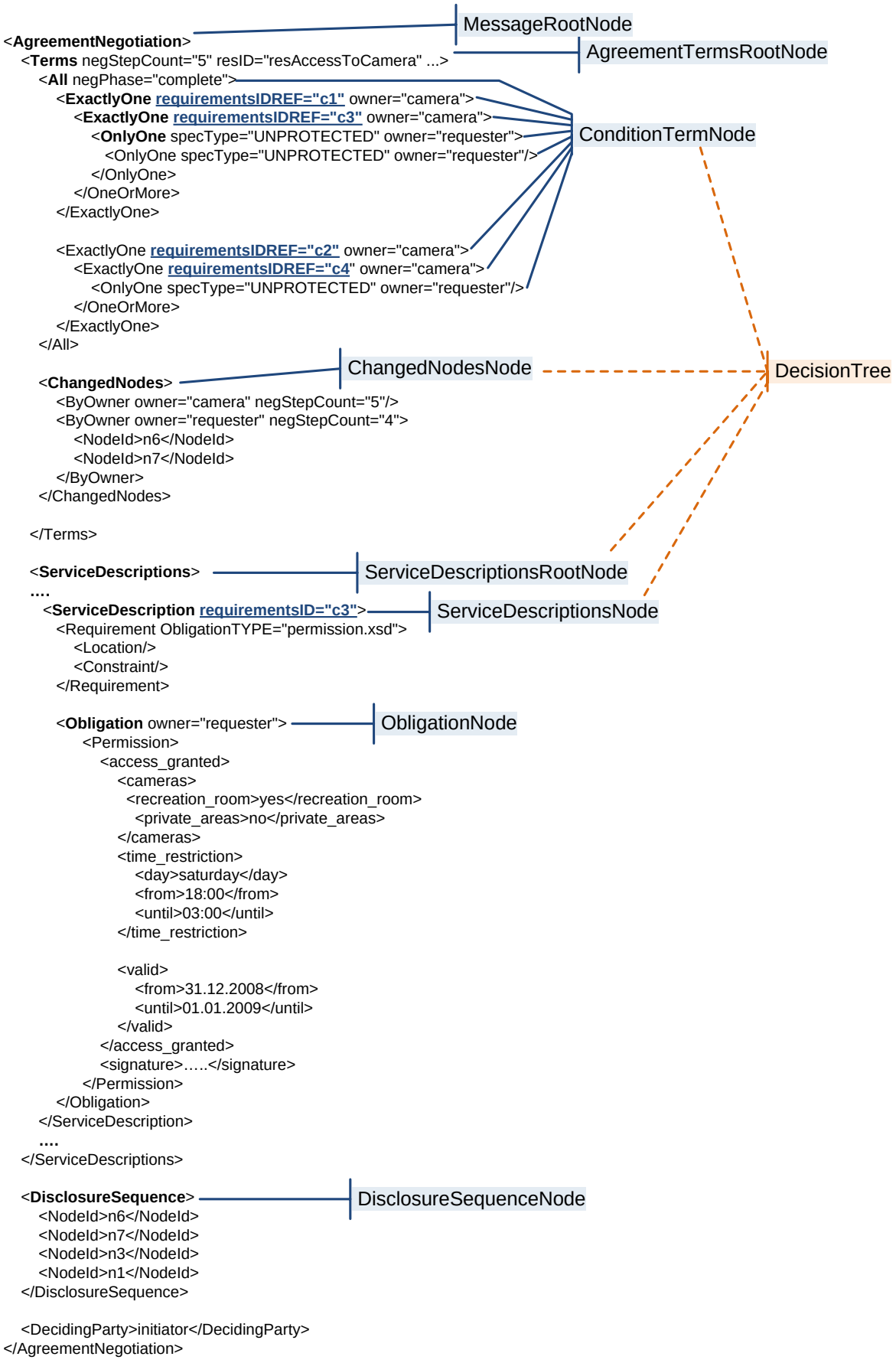

Figure 7.7: *One-to-One Mapping of DOM Tree to Class Instances*

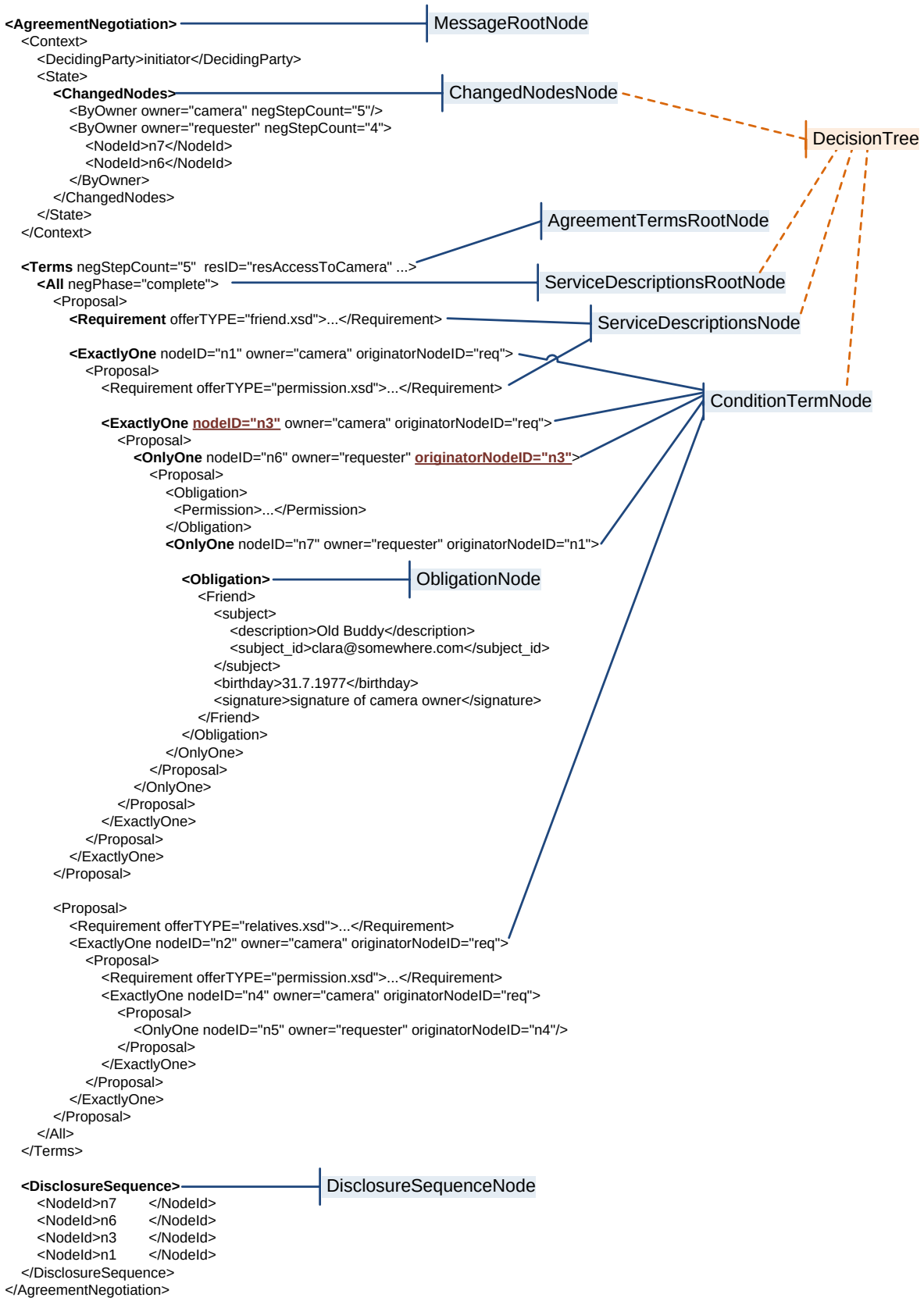

Figure 7.8: *One-to-One Mapping allows for equivalent XML Representations*

The ConditionTermNodes contain the `requirementsIDREF` attribute for referencing ServiceDescriptionNodes through their `requirementsID` attribute. For an easy navigation of the separate XML tree structures the `DecisionTree` class provides a uniform interface to access ConditionTermNode, ServiceDescriptionsNode, and ServiceDescriptionNode. It joins the logical dependencies of the ConditionTermNode with the ServiceDescriptionNode.

The `DisclosureSequenceNode` is a class that keeps track of the sequence in which Offer Details have to be exchanged.
The `ProfileRootNode` handles XML files with preferences and provides a mapping between offerTYPE, available offers and corresponding requirements. The `DisclosurePolicyRootNode` contains all requirements and the dependencies between requirements that must be fulfilled for a given offer.

An alternative messaging format relies on a recursive encapsulation of Requirement and Obligation as shown in Figure 7.8. The tree contains now also Requirement specification and Obligations in addition to the dependencies between ConditionTermNodes. The recursive definition of such a tree is straight forward, but makes it harder for a human to trace the dependencies between requirements.
The XML DOM contains the dependencies between the different XML elements. The one-to-one correspondence of XML elements and class instances, allows the class instances to focus on the negotiation logic and functions for manipulating the negotiation state. This clear separation makes it easy to make fundamental changes to the XML message format, with only small changes in the code. The implementation supports an additional XML messaging interfaces. This messaging interface is nearly identical for the implementation of the negotiation algorithms. Only the accessor functions had to be adapted.
The second Figure shows how Proposals reference Requirements via the `originatorNodeID`. This reference states that the party makes a proposal to satisfy a certain requirement. The `originatorNodeID` are also part of the XML show in Figure 7.7, but were omitted due to space constraints.

Disclosure policies provide the Requirements for an Offer. The `OfferRootNode` class derives from the `BaseRootNode` and provides accessor functions for the requirements.

```
1  class OfferRootNode : public BaseRootNode
2  {
3    public:
4      OfferRootNode(xmlpp::Element* elem, SimpleLogger* logger) : BaseRootNode(elem, logger) { };
5      virtual ~OfferRootNode() {};
6      const Glib::ustring get_offerTYPE() const;
7      const Glib::ustring get_polID() const;
8
9    protected:
10   private:
11 };
```

Listing 7.1: *Root Node for Offer in DOM Tree*

7.4 Reference Scenarios

VersaNeg can be used in many other scenarios, besides e-contracting for cloud computing. This section introduces two additional reference scenarios to explain different aspects of the protocol behavior in more detail. The first example in Section 7.4.2.1 shows how negotiation with VersaNeg can be used as a tool for coordination and authorization. The second example shows in Section 7.4.3 how multilateral negotiation can serve as a tool to discover collaborations in the supply chain of the fashion industry.

7.4.1 Symbols and Notations for Graph Visualization of Negotiation States

Reference scenarios help in understanding the capabilities of the negotiation protocols. The graphs shown in this thesis rely on automated graph layout and visualization with Graphviz[5]. The advantage is that the graphs represent the runtime state of examples that can be generated on the fly by the implementation. The automated graph layout has the drawback that the options for influencing node placement or text style are limited.
The negotiation states are a tree structure of requirements. The father-child relationship between nodes is depicted as solid black lines and convey the dependencies between requirements. The basic idea that a party states requirements for making an offer is preserved in the diagram through references. If a party makes an offer it inserts a node in the tree and references the ancestor node it wants to make an offer for. The node itself contains the requirement. If a party wants to satisfy a requirement without making an offer it marks the node as an unconditional offer, shown as "Offer with No Requirement" in the diagram.

The diagram supports the reader in making decisions about which offers should be made. The online graph generation can even allow human operators to use the MPN framework as a negotiation support system. By inspecting all requirements that have been inserted during one round, the operators can discover which offers can me made and instruct the MPN framework accordingly, for instance, by formulating offer details and corresponding requirements. By doing so the MPN framework can learn new requirement and offers and extend the domain where it can perform automated negotiations.
The diagram also allows the reader to discover agreement options manually. By simply traversing the tree from root to the leafs, one learns about possible paths. With ExactlyOne nodes, each leaf in the tree indicates a potential path that could be an agreement option. Different paths starting at leaf nodes can be combined to satisfy OneOrMore or All conditionals. The next step is to exclude all paths that are not satisfiable. One has to discard all paths that have at least one requirement node which does not have an incoming edge with an offer. Unconditional nodes do not have a requirement and can be ignored for the latter step.

Sometimes one single offer is associated with different requirements. In many scenarios, the conditionals demand that more then one requirement has to be satisfied to gain the offer. If the requirement R_B in Figure 7.10 has an ExactlyOne conditional either R_B, R_C or R_B, R_D have to be satisfied before the party will realize its offer O_A.
To display that a group of requirements, inserted by one party, provide an offer for the same requirement of another party, the virtual node "Same Offer" is introduced. All requirements that belong together and make the same offer, point towards the "Same

[5] http://www.graphviz.org/

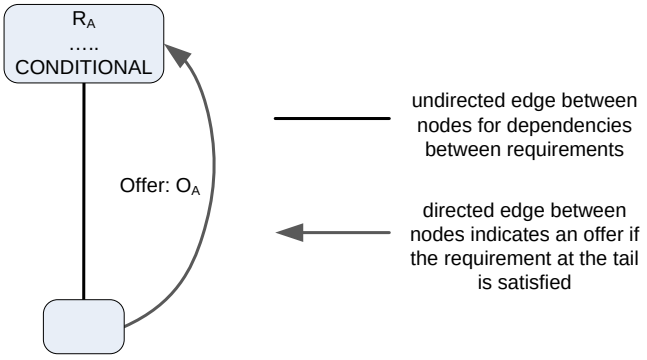

Figure 7.9: *Graph Legend for Negotiation States: Vertices between Nodes*

Offer" node. The outgoing edge of the "Same Offer" node is directed at the requirement that is satisfied by this offer.

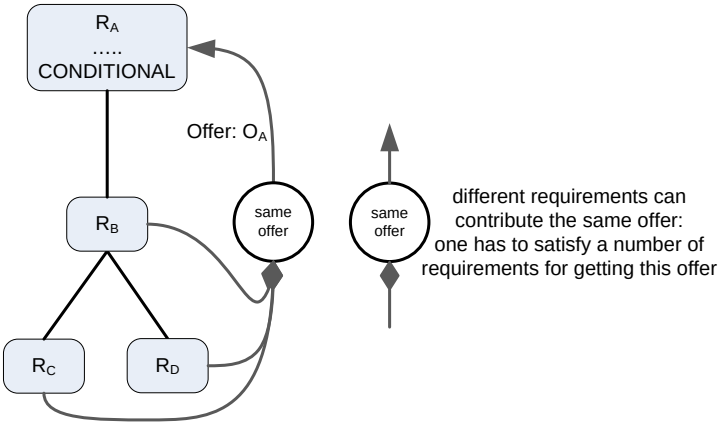

Figure 7.10: *Graph Legend for Negotiation States: Same Offer Indicator*

The graphs use the nodes shown in Figure 7.10 for presenting important details of requirements and offers. Words in upper case letters denote variable fields. The nodes that belong to one party have the same color.
The orange oval is the root node of a negotiation, which contain the current phase of the negotiation (Discovery, Pending, Disclosure, Failed and Complete) and if applicable a scenario identifier for the negotiation.
The boxes contain one requirement each with a unique identifier (UNIQUE REQUIREMENT TYPE). In our examples the identifier is also the name of the XML Schema. Each requirement has an owner field (Owner:PARTY NAME) to indicate who introduced the requirement in the negotiation. The square brackets contain a node identifier [NODE-ID] that is unique for each node in the tree. If the node has been chosen for an agreement option, the bracket contains additionally an identifier to define the sequence for releasing the offer details [NODE-ID, EXCHANGE-SEQUENCE-ID].
On the lower left side is a conditional that applies to the children (ExactlyOne, All, OneOrMore, OnlyOne). The audience states who is allowed to make offers to the Requirement.
A thick line at the border of the box indicates nodes which already have all offer details for their requirement. This is useful to visualize runtime states during offer details disclosure.

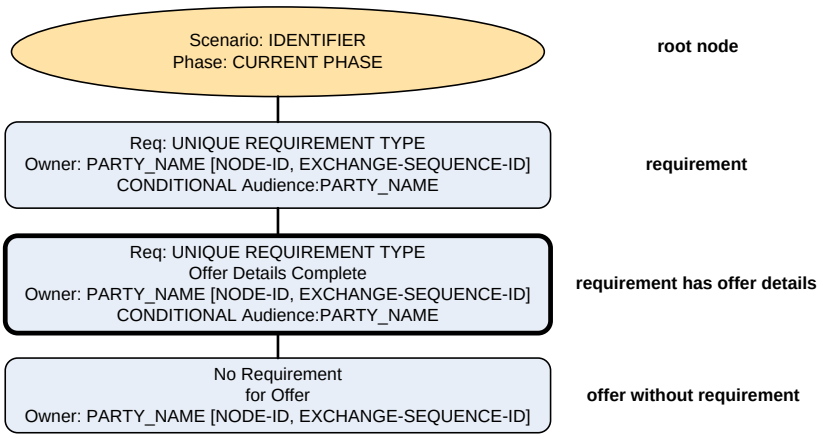

Figure 7.11: *Graph Legend for Negotiation States: Proposal Details*

7.4.2 Surveillance Camera Access

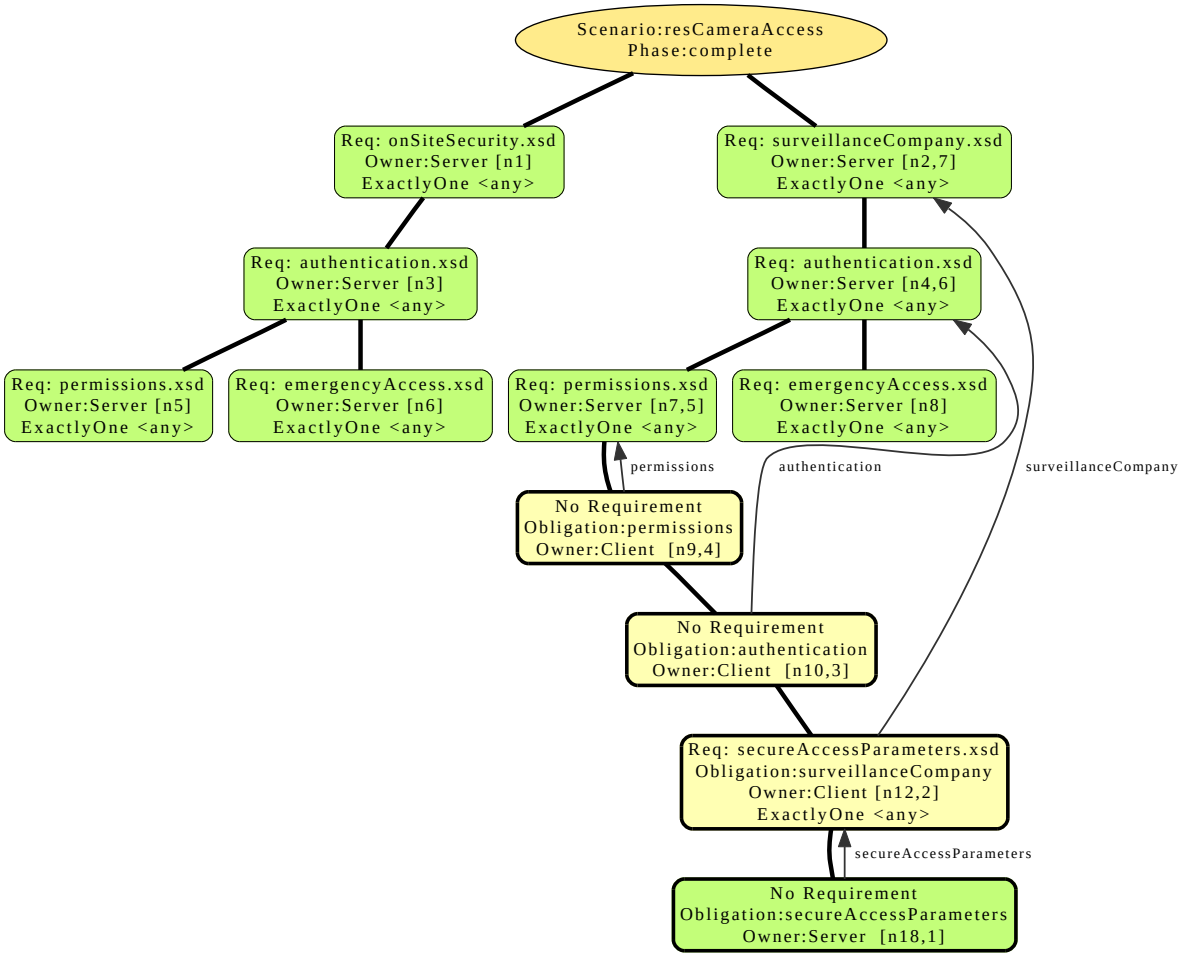

Figure 7.12: *Camera Access*

This basic scenario shows how the generic negotiation framework can realize bilateral Trust Negotiation in a surveillance camera access scenario.
Recall that the research on *Automated Trust Negotiation (ATN)* [WiLi02, BeFS04] deals with automatically establishing formal agreements and the protection of private information during a negotiation process. An important property of ATN is the disclosure

Credential	Who	Type	Description
$C_{authentication}$	S,G	I	proof of authentication issued by trusted third party
$C_{onSiteSecurity}$	G	I	proof of affiliation with guarded company
$C_{surveillanceCompany}$	S	I	proof of affiliation with surveillance company
$C_{permissions}$	S,G	I	encodes which cameras should be accessible for which roles and groups
$C_{emergencyAccess}$	S,G	D	in case of emergency, receive full access
$C_{secureAccess}$	G	I	technical parameters for secure connection with camera

Table 7.1: *Credentials of the Surveillance Camera Access Scenario, I:Information/D:Decision/T:Authorization Token, S:Surveillance Company/G:Guarded Company*

of only the minimal set of credentials and the protection of sensitive information within credentials.
Trust Negotiation governs the access to resources by an attribute based authorization, for instance, to subscribe automatically to an Internet service based on the attributes of the user and not of the user's identity. These attributes are encapsulated in credentials. Trust Negotiation establishes trust by a careful credential exchange. By inspecting credentials that are issued by trustworthy instances, one can derive knowledge about affiliations and properties of the involved parties.
Disclosure policies define logical conditions that must be met before a resource can be accessed or a credential can be released.

7.4.2.1 Surveillance Camera Access Scenario

A typical example from the trust negotiation domain comes next. A company has a number of surveillance cameras on its perimeter. The company has authorized staff to access the cameras and an external security company can also receive video feeds from the cameras. The basic idea is that not each person has access to every camera.
The security company wants to be flexible in allocating human resources for monitoring the cameras. Hence the authentication and authorization is a shared process between security company and the guarded company. The guarded company does not have to administrate user accounts for the security company.
Camera access can be obtained by providing a defined set of credentials. The credentials serve as tokens to identify oneself and other credentials serve as authorization tokens that encode the permissions that a user has. Some credentials are static, others are created dynamically during the negotiation. This scenario assumes that the camera operator distributed different credentials to its staff and the security company.
The security policies of the guarded company state who should be able to access which cameras on the perimeter. An intuitive graphical user interface could help during the creation of these policies, without dealing with the syntax and organization of the policy definitions.

7.4.2.2 Negotiation about Credentials

The video surveillance requires that someone who wants camera access must identify herself. The policy requires additionally an explicit permission (signed credential) that states which cameras should be made accessible for the owner of the permission credential.
The Table 7.1 shows the available credentials for the negotiation. The requirements work as normal. The obligations contain credentials for trust negotiation scenarios. The credential
$C_{authentication}$ is issued on the fly by an identity management system of a trustworthy third party. The $C_{onSiteSecurity}$ security credential states that this person is employee of the guarded company. The surveillance company can issue $C_{surveillanceCompany}$ credentials to state which employees belong to the company and state the security clearance. The $C_{permissions}$ credential is issued by the guarded company and defines which user groups and which security clearances are required to access a certain camera. The $C_{emergencyAccess}$ is a credential that gives access to all cameras. A user that issues a $C_{emergencyAccess}$ during the negotiation will have to justify personally the usage later on at the guarded company. All these requirements are from the system that protects the cameras.
There are also two credentials that the accessor of the camera system demands. The accessor expects the $C_{secureAccess}$ credential that contains technical specifications of the guarded company how to access the camera, for instance, HTTPS/TLS for secure connection, URL to access camera, and camera authenticates via an X.509 certificate.

The negotiation state of a complete agreement document is shown in Figure 7.12. The tree consists of ExactlyOne conditionals. That is, for choosing one agreement option one has to decide for one satisfied branch. The tree has therefor four different potential agreement options. It depends now on the requester which credentials it want to provide. In this example, the surveillance company requested normal access and offered the $C_{permissions}$, $C_{authentication}$, and $C_{surveillanceCompany}$ credentials. It wants in turn the $C_{secureAccess}$ credential from the guarded company to configure its systems to access the camera.
The flat structure for XML negotiation states, as shown in Figure 7.7, has the advantage that for this example the specification of the requirements $R_{authentication}$, $R_{permissions}$, and $R_{emergenyAccess}$ is present only one times in the state, even though the requirement is stated at different locations in the tree.

7.4.2.3 Benefits

Using a trust negotiation for such a scenario has several advantages. The authentication and authorization is now a distributed task that can easily involve different administrative domains, without the need for interoperable communication interfaces and mutual connectivity. It is easier to issue credentials than to adapt complex communication interfaces for interoperability. Authorization policies at the guarded company can be easily adapted for changing security requirements. Access tokens can be revoked by formulating conditions to exclude credentials with certain attributes, for instance, exclude particular user. The guarded company does not need to perform identity management for the surveillance company and thereby reduces administrative expenses for both companies.

We add additional functionality for automated negotiation and storage of established agreements. These components are part of the infrastructure and can be used by any service within a process.
The advantage is that the invocation of the negotiation service could be controlled at the service repository or by modifying partner links in the process definition.

Parameters from the agreement that may pertain the further processing, can also be provided as variables. For these reasons a service proxy should be visible in the process. The details of the negotiation with the business partner are of little relevance. Services might still support the parameters of the agreement for compliance checking. Services can also be implemented unaware of the negotiation service. However, the compliance checking of the agreement has to be deployed as a separate service then.

7.4.3 Multilateral Negotiation in Fashion Supply Chains

Fashion firms develop new products at a high pace that must sell in volumes during short selling seasons. The development and production of fashion articles is a distributed process that happens in short cycles. The fashion retailer Zara is a good example that shows the scale and complexity involved in this rapid moving industry. The international fashion retailer Zara is a pioneer of *fast fashion* [Toka08]. Zara achieves short development cycles, rapid prototyping, small batches and variety of fashion articles. Zara requires a very fast and highly responsive supply chain to achieve the high turnover of new designs and to achieve high delivery rates in short time periods. Zara has circa 400 near-by suppliers in Portugal in Spain and much more suppliers distributed around the globe. Zara evaluates 30000 designs a year from which 11000 designs will be produced.

Fast fashion is an area where multilateral negotiation is an appealing tool to establish collaborations through electronic negotiation. One of the reasons is that fashion is very shorted lived and requires constant collaboration through the supply chain, whilst the processes themselves can be standardized.

7.4.3.1 Fast Fashion Scenario

This section will evaluate how the VesaNeg protocol can be employed to further speed up the development and production process. There is little public information available on how Zara coordinates its supply chain. Simatupang et al [SiSL04] published a detailed description and analysis of a leading fashion retailer in South-East Asia. This retailer defines its seasonal strategy and reaches production 12 weeks later. The items reach the store after 4 more weeks.
The VersaNeg negotiation model of a supply chain will be based on the interactions between the stakeholders described in their publication. There are four types of stakeholders in this scenario:

1. **Fashion Retailer** is in charge of the design, supplier selection, and distribution of the fashion articles.
2. **Designers** are responsible for the creative design. Production designers translate the design into specifications of the required fabrics and processing instructions. The current scenario assumes that a designer carries out both, creative design and production design.
3. **Garment Supplier** is responsible for fabricating the fashion articles in required numbers according to the design specifications.

Obligation	Party	Description
$O_{details\_blazer}$	designer1,garment2	design and specifications of blazer
$O_{details\_delivery}$	garment1,garment2	possible delivery volume and shipping dates
$O_{details\_garment}$	garment1,garment2	description of fashion article
$O_{details\_white\_fabric}$	fabric1,fabric2	specification and images of white fabric
$O_{details\_black\_fabric}$	fabric1	specification and images of black fabric

Table 7.2: *Obligations for Fast Fashion Scenario*

4. **Fabric Supplier** produces the fabrics in the required numbers and according to quality requirements. The fabrics are used by the Garment Supplier to produce the fashion article.

The multilateral negotiation serves as a tool to discover possible collaborations. The exchange of samples in the physical world and the negotiation about prices happens outside the protocol. Consequently, VersaNeg serves in this scenario as a protocol to discover possible collaborations and to establish fundamental parameters for a contract. The contract will be formed later on, via traditional means by human-to-human interactions with the input from the VersaNeg negotiation. A framework contract outlines that the VersaNeg negotiation outcome serves as an input to legal contracts, but it does not constitute a contract on its own.
There is one fashion retailer ($retailer$), one designer ($designer1$), two garment suppliers ($garment1, garment2$), and two fabric suppliers ($fabric1, fabric2$) in this scenario. The Table 7.2 shows the negotiating parties in this scenario and the Obligations they are willing to take. The Figure 7.13 shows the negotiation state for the fast fashion scenario at the end of the Agreement Discovery phase.

The $retailer$ is in control of the process and consequently starts the negotiation. The fashion retailer uses the Requirement R_{blazer} to describe that the retailer wants a blazer design that is in line with a current fashion trend. The Requirement $R_{delivery}$ states that the production of this article must meet certain production numbers, quality metrics, and that the product must be available in a certain time frame. The $retailer$ uses ExactlyOne as a conditional because it wants the same quality and style of this blazer for all stores.
The Requirement R_{blazer} is answered by the $designer1$. The $designer1$ offers a blazer but requires a garment supplier to produce the blazer in required numbers. The $designer1$ gives detailed specification of the blazer with $R_{garment}$ to instruct the garment supplier how to manufacture the blazer. The garment supplier in turn requires R_{white_fabric} and R_{black_fabric} to produce the garment in accordance with the specifications of $designer1$. The garment supplier not only offers $O_{details\_garment}$ to $designer1$ to produce the design, but it also offers to the fashion retailer with $O_{details\_delivery}$ to keep the delivery schedule, quality guarantees, and production numbers.
The Requirement R_{blazer} is also answered by $garment2$. This garment supplier has its own designers and can directly propose a blazer $O_{details\_blazer}$ to the fashion retailer and $O_{details\_delivery}$.
Both garment suppliers require fabric for their garments. There is more flexibility with regards to having slightly different fabrics from various suppliers in the same collection. Therefore, both garment suppliers use OneOrMore to indicate that they are willing to source from different fabric suppliers.
The fabric supplier $fabric1$ offers $O_{details\_white\_fabric}$ and $O_{details\_black\_fabric}$. The other

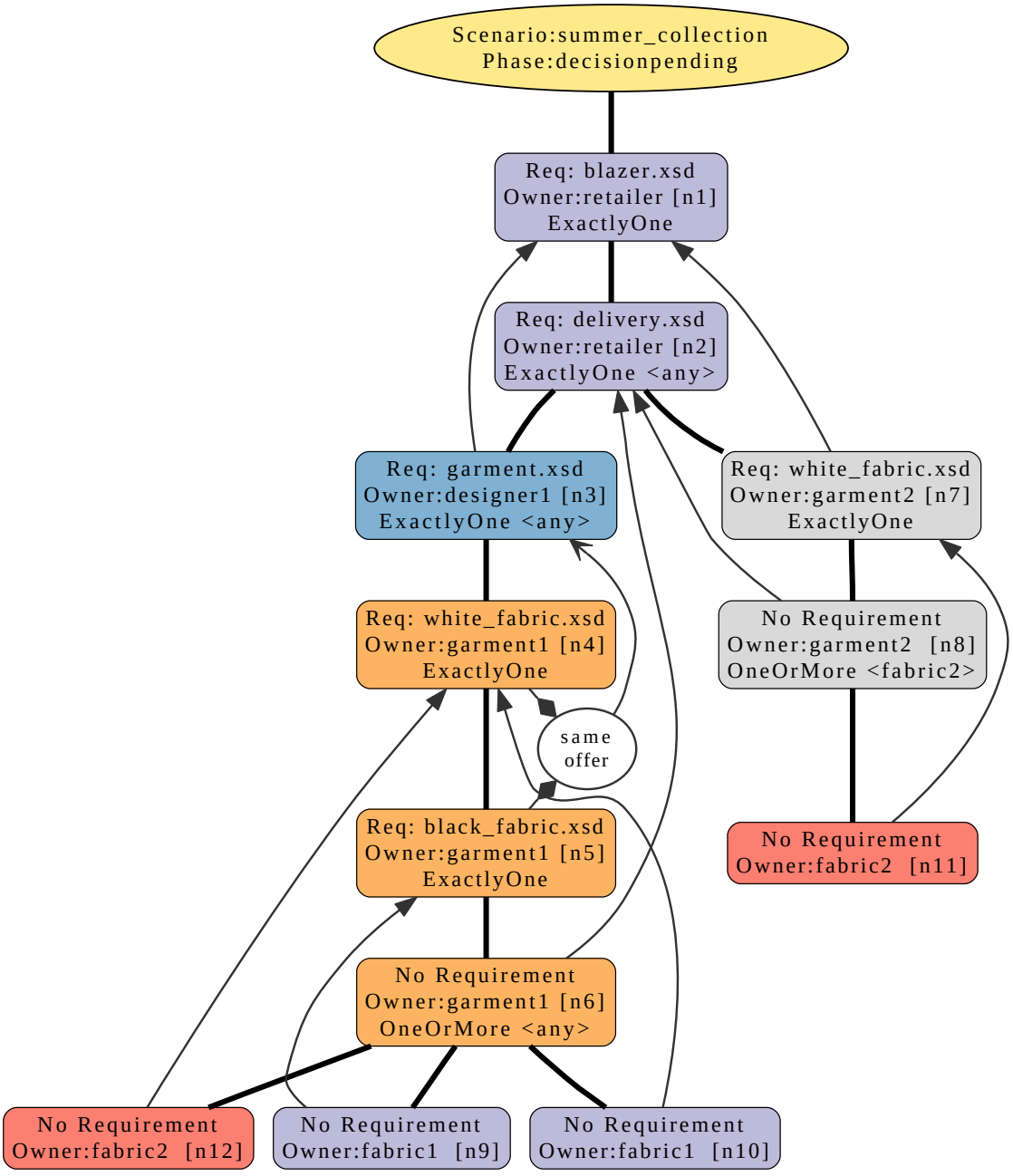

Figure 7.13: *Fast Fashion Scenario Negotiation State*

fabric supplier $fabric2$ can only offer $O_{details\_white\_fabric}$.

There are now 4 alternative agreement options present in this negotiation state. The fashion retailer decides to give both garment suppliers a chance to submit samples. It duplicates the negotiation state and sends one negotiation state to $garment1$, where it chooses the agreement option $\{P_{n10}, P_{n9}, P_{n6}, P_{n5}, P_{n4}, P_{n3}, P_{n2}, P_{n1}\}$. It sends another copy of the negotiation state with the agreement options $\{P_{n11}, P_{n8}, P_{n7}, P_{n2}, P_{n1}\}$ to $garment2$. We now have two concurrent negotiation states where the parties add their Obligations. The Obligations contain detailed statements how the parties plan to fulfill their promises. The VersaNeg protocol arrives at two agreements for the two agreement options.

Human negotiators take the outcome of both concurrent negotiations and request samples from the garment suppliers. The human negotiators can refine the design and enter a price negotiation. The VersaNeg agreement will provide many details that can be filled into a contract later on.

The negotiation state shows some particularities of the VersaNeg protocol. The *same offer* node indicates that *garment*2 answers the same Requirement $R_{garment}$ with two Proposals P_{n4}, P_{n5}. These two Proposals are a joint Offer and will result in only one Obligation $O_{garment}$ during the Obligation Disclosure phase.
The Proposal P_{n6} is an answer to the Requirement $R_{delivery}$ in P_{n2}. The function $CandidatesToAppendTo(P_o^{p_i}, E_x)$ in 6.2 permits that this Proposal can be added directly to P_{n3}. This would not make much sense, however, as *garment*1 would offer to deliver a garment that it does not know about. Therefore, *garment*1 appends its Proposal to P_{n5} which it added during the same processing step. It thereby states that it can deliver the blazer design from *designer*1.

7.4.3.2 Benefits

The VersaNeg protocol is used for a part of the fashion procurement process. It is a structured method to call for fashion designs and the necessary suppliers. The protocol is a fast option to discover alternative designs and manufacturing options for new fashion articles. The human negotiators can now focus on designs that can be realized by the supply chain under defined delivery constraints. The protocol already excludes designs that cannot be produced with the available suppliers. This saves the human negotiators a significant part of the coordination effort.
The multilateral negotiation provides much flexibility to discover different types of collaborations. The protocol allows for a competition between suppliers that take the whole responsibility for the complete product and for all sub-products. In the same time, the protocol allows for work distribution between different suppliers. This flexibility allows many suppliers to make bids, even if they are highly specialized. Thereby, this protocol fosters a quality competition and a price competition.
The negotiation discovers already many important parameters for a contract later on. That saves time for the human negotiators to formulate the contract. Because the agreements are already in a machine processable format, they can be fed into a business process orchestration tool to coordinate the production later on.
Even though a part of the contract formation is still done by human negotiators, the use of VersaNeg saves time and increases competition.

Discussion is just a tool. You have to aim; the final goal must be a decision.

Harri Holkeri

8. Comparison of Protocols for Bilateral and Multilateral Negotiation

This doctoral thesis introduced three approaches to agreement negotiation. None of the three approaches can claim to be the single best solution to all agreement negotiation scenarios. This Chapter gives a brief comparison of the three approaches and of related negotiation protocols from research and standardization.

8.1 Agreement Standards and Negotiation Protocols

The WS-Policy standard [VOHH+07] from the World Wide Web Consortium relies on policy intersection for forming agreements between two web services. Policy intersection is an efficient tool to establish mutually compatible parameters during a protocol handshake. Two messages can already be enough to establish an agreement with policy intersection. Policy intersection protocols are easy to automate because they can work with static pre-defined policies. Standardization of the policy formats assures that both parties have the same mutual understanding of the policy. The risk of automated agreement formation is limited because both parties in a negotiation can assure that each protocol message is compliant with its local policies. The static structure greatly reduces the risk of automated negotiations but also limits how the policy intersection protocols can be used. They are unable to discover unforeseen agreement options outside their policy definitions and they are not intended to create legally binding agreements.

OASIS and UN/CEFACT started the standardization of Electronic Business with the eXtensible Markup Language (ebXML) [CCKH+01] in 1999. The objective of ebXML is to create XML based standards to form business relationships and to conduct electronic trade. The ebXML Collaboration-Protocol Profile and Agreement Specification (CPA) [ACCF+02] defines a language to define agreements. The non-normative part of the document suggests policy intersection as a negotiation method to form ebXML CPA compliant agreements. No details are given how to realize such a negotiation. Consequently, ebXML is a means to define business contracts. However, the ebXML does not define a protocol for Agreement Discovery and Agreement Formation. It is mainly suited to formalize and execute agreements that have been discovered out-

side the ebXML protocol.

The FIPA Contract Net Protocol [FIPA02] is popular for coordinating tasks in multi-agent systems. An agent announces its task and other agents make proposals to perform this task. There exists a large body of research on intelligent bidding strategies during auctions with the Contract Net Protocol. The protocol allows the negotiating parties to state pre-conditions and terms of an agreement. However, the Contract Net Protocol provides only a method to transmit such conditions without defining the specifics of how to negotiate about terms of an agreement. It allows for an exchange of pre-conditions for proposals and for agreements. The interrelationship between the descriptions of the conditions in proposals and agreements must be handled by the negotiation systems that use the Contract Net Protocol. The protocol itself is only a means to transport arbitrary conditions but does not define how the systems should handle these statements. In fact, the terms stated in a proposal can be completely independent from the terms in the request for proposals and even from the final agreement. The Contract Net Protocol gives much freedom for the implementation of the negotiation systems but also means that a large part of the negotiation about terms of an agreement must be handled outside of the protocol. The agreements reached with the Contract Net Protocol are only bilateral.

The OASIS WS-Agreement [ACDK+07] standard uses a template-based offer-answer negotiation protocol to establish detailed bilateral agreements that define the requirements of a service consumer and the assurances by a service provider on service availability and quality of service. Agreements under this standard incorporate detailed descriptions of services, constraints on possible agreements, and business values, such as rewards and penalties. The basic idea of WS-Agreement is that a template largely defines the possible agreements. Only a few fields are variable and must be filled in during the negotiation.
The behavior of WS-Agreement is similar to standard form contracts where also most of the contractual terms are fixed and cannot be negotiated. Only few blank fields can be filled in, for instance, names of the contracted parties, date of the contract, price, and signatures.
Standard form contracts are also referred to as "take it or leave it" contracts because one side has all the bargaining power and can use it to write the contract to her own advantage, whereas the other party has no realistic means to negotiate the terms and conditions of the service. The same is true for WS-Agreement. One party defines the agreement template and the other party fills in values that must obey constraints set forth by the first party. The strength of the WS-Agreement standard for automated agreement formation is that an agreement negotiation system can focus on a few variable fields that are easy to interpret and easy to fill in.
In theory it is possible to establish an agreement that can deviate substantially from the agreement template with WS-Agreement. However, this new agreement could not be established automatically anymore, due to the lack of semantic understanding. Automated systems implementing WS-Agreement cannot cope with substantial changes to the agreement template, because that would require a true understanding of the semantics and dependencies expressed within the agreement.

8.2 Comparison against Problem Statement

This dissertation introduced policy intersection as a robust and efficient method to obtain agreements. Requirements-driven negotiation is a more powerful method for iterative agreement negotiation. It is useful to put these approaches into perspective by comparing them to existing approaches.

The table 8.1 shows prominent standards for Agreement Discovery and Agreement Formation. The symbol ● states that a protocol has a certain property. The symbol ◐ states that the protocol covers a feature at least partially that can also mean that a protocol can be extended with reasonable effort to support a feature. The symbol ○ states that a certain feature is not supported by the protocol.

Agreement Discovery is an important task to identify possible agreements. Agreement Formation is the task to define a mutually accepted agreement and the ability to proof later on the terms of the agreement. All protocols except ebXML CPA are also intended for Agreement Formation. ebXML CPA [ACCF+02] is intended to represent agreements, to describes all the valid visible, and hence enforceable, interactions between the parties and the way these interactions are carried out. ebXML CPA does not standardize the Agreement Formation.

The ability to form comprehensive agreements that contain everything from the negotiation that is relevant for the interpretation of the agreement, is another distinguishing factor. The FIPA Contract Net Protocol [FIPA02] standard gives the negotiating parties much freedom how to use the negotiation primitives and how different statements made during the negotiation relate. Because FIPA Contract Net Protocol is so open, it does no enforce that the final agreement contains everything that is relevant to the agreement. Instead other statement during the FIPA Contract Net negotiation could also be crucial to the interpretation of a contract. All other approaches produce comprehensive agreements.

All approaches target bilateral agreement formation. Only VersaNeg can negotiate about bilateral agreements (see Section 5) and multilateral agreements (see Section 6).

Agreement negotiation protocols should not be limited for use in special domains only. That is probably the most restricting property of WS-Policy [VOHH+07] and ESAF Policy Intersection (see Section 4). These policy intersection approaches work well for establishing mutually compatible technical parameters but they are not suited to negotiate about agreements with business objectives.
The ability to integrate other language constructs to define agreements is crucial to map real world dependencies, as in paper contracts, to the electronic agreement. The policy intersection protocols are limited to define technical parameters. The FIPA Contract Net allows for different agreement languages. The protocol is mainly used together with a number of FIPA content language specifications. The WS-Agreement standard and VersaNeg leverage the flexibility of XML to combine language constructs of different agreement and SLA languages within one single XML document. For instance, WS-Agreement can integrate the Job Service Description Language (JSDL) [ABDF+05] to define job submissions to grid platforms. VersaNeg can integrate WS-Agreement to define Requirements, Offers, and Obligations.

Negotiation systems may find it difficult to verify the interrelationship of statements during a negotiation. For instance, if one party wants to negotiate about a purchase of bananas but it receives only an offer about apples, the apples are most of the time an undesirable offer for the initiator. The more freedom degrees a protocol gives, the more

	Agreement Discovery	Agreement Formation	Comprehensive Agreement	Bilateral Agreement	Multilateral Agreement	Agreement Language Integration	Interrelationship of Statements	Secure Negotiation
WS-Policy	○	◐	●	●	○	○	●	◐
	web service standard for discovery of mutually compatible technical parameters only							
ESAF Policy Intersection	○	◐	●	●	○	○	●	◐
	discovery of mutually compatible technical parameters to establish secure communication channels							
OASIS ebXML CPA	○	○	●	●	○	○	○	◐
	formalization of electronic contracts but does not standardize agreement negotiation							
FIPA Contract Net Protocol	◐	●	○	●	○	●	○	◐
	exchange of proposals and bids for bilateral agreement formation. does not define iterative agreement discovery							
OGF WS-Agreement	◐	●	●	●	○	●	●	◐
	template-based bilateral agreement formation, but is restricted by one offer-answer exchange							
VersaNeg	●	●	●	●	●	●	●	●
	iterative bilateral and multilateral agreement discovery and agreement formation							

●: protocol has property, ◐: partial coverage, or could be enhanced, ○: not supported

Table 8.1: *Comparison of Agreement Standards and Protocols*

difficult it might be to determine automatically if statements during a negotiation really relate. Policy intersection protocols allow for an easy verification; each agreement must also match the local policies. The FIPA Contract Net Protocol allows for agreements and proposals that have nothing in common with the original request for proposals. The burden to verify the interrelationship of negotiation messages is up to the negotiation

systems. WS-Agreement [ACDK+07] can use policy intersection to determine if an agreement offer is compliant with the previously sent template. VersaNeg is the only protocol that allows for an iterative multi-round negotiation where the interrelationship between the statements is always clearly defined.

All protocols except VersaNeg do not explicitly specify how to perform secure negotiations. A negotiation can be considered as secure if the confidentiality, integrity, authenticity, and non-repudiation, are assured and message replay can be detected. The implementations of these standards can integrate secure communication channels with mutual authentication, apply digital signatures and add additional logic to verify the negotiation states. Even though the implementation of the standards can integrate all necessary security measures with these protocols, the fact that there is no comprehensive standardized security concept, makes it more likely that the implementations of these standards have vulnerabilities.

Agreement Discovery

Agreement Discovery is probably the most distinguishing feature for comparing negotiation protocols. The realization of Agreement Discovery largely defines how the final agreement can look like and also determines how much support the protocol itself has for a structured iterative negotiation. This section compares the capabilities of the different protocols during Agreement Discovery.

An agreement protocol must assure that the objectives of the negotiating parties are compatible. For instance, if one party wants to sell a good and another party wants to buy the good, the objectives are compatible. If one party wants to buy a good but the other party does not want to sell the good, the objectives are obviously not compatible. The policy intersection approaches only accept agreements that are compatible with their local policies. Therefore, it is easy to verify that the agreement is compatible with the objectives. The FIPA Contract Net Protocol assumes that the negotiation systems themselves have all the necessary logic to evaluate proposals and decide locally if the proposal is compatible with the objectives of the agent. This assumption gives much freedom for intelligent agents to negotiate, but the protocol itself has little support for enforcing that the objectives are compatible. The WS-Agreement protocol uses a template-based approach that is similar to a form contract. This makes it easy to verify that an offer is compatible with the template. VersaNeg assures the compatibility of the objectives through its strict matching of Requirements with Offers. Only Offers that verifiable match a Requirement are considered valid.

Iterative refinement is a technique during agreement discovery that allows the parties to start with a broad problem statement and refine the possible agreements through consecutive negotiation steps. Iterative refinement is also an important part of human to human negotiations. The policy intersection protocols and the WS-Agreement offer-answer exchange are single-round protocols that do not support this feature. The FIPA Contract Net Protocol supports negotiation with multiple rounds. However, the protocol does not define how the messages relate. Negotiation systems using Contract Net Protocol could define their own negotiation conventions to perform iterative refinement via the Contract Net Protocol.

Agreement Discovery is the phase where the negotiating identifies alternatives to reach an agreement. Obviously, no party can ever be sure that it knows all possible agreement options, before a negotiation starts. It is therefore important that the negotiation

Agreement Discovery	Compatible Objectives	Iterative Refinement	Discover new Options	Establish Collaborations	Price Negotiation	Concession Protocol	Electronic Contracting	Message Exchange Efficiency	Risk of Automated Negotiation
WS-Policy	●	○	○	○	○	○	○	◐	●
ESAF Policy Intersection	●	○	○	○	○	○	○	◐	●
ebXML CP/MA	◐	○	○	○	◐	○	○	○	○
FIPA Contract Net Protocol	○	●	◐	○	●	●	●	◐	○
WS-Agreement	●	○	○	○	◐	○	●	◐	●
VersaNeg	●	●	●	●	◐	○	●	○	◐

●: protocol has property, ◐: partial coverage, or could be enhanced, ○: not supported

Table 8.2: *Comparison of Negotiation Protocols during Agreement Discovery*

protocol supports the discovery of unforeseen agreement options, which might be more favorable than any a-priory envisioned agreement. The policy intersection approaches can only form agreements that are within their local policies. Hence, these protocols cannot discover new agreement options. The FIPA Contract Net Protocol has great freedom to discover unforeseen agreement options, but gives little structure and guidance during this process. The VersaNeg approach gives a structured method to discover unforeseen agreements with an iterative refinement approach. Hence, VersaNeg has less freedom for the discovery of unforeseen agreement options but provides a structured process for the discovery of new agreement options.

Agreements often involve more than two parties. Electronic negotiation protocols can be of high value to form unforeseen multilateral collaborations. Human to human negotiation scales only badly to negotiations where a group of parties must define their collaboration during a multilateral negotiation. Even though, the agreement languages of some of the approaches above allow for agreements that involve multiple parties, only VersaNeg defines a negotiation protocol for multilateral Agreement Discovery.

Price negotiation has been the focus of research on electronic negotiations for a long time. The WS-Agreement and the VersaNeg can also support price as part of their

negotiations; however these protocols are limited in what they can do. For instance, it would be possible to integrate sealed bid auctions with these two protocols, in contrast to open-cry auctions do not fit these negotiation protocols.
Of all approaches presented here, the Contract Net Protocol is the best suitable protocol for multi-round auctions and price negotiations. There exists a large body of research on price negotiation and auctions with the Contract Net Protocol.

Concession protocols start with two opposing views and move towards a compromise by giving up certain Requirements from the own position. Eventually, the parties arrive at a mutually acceptable position. Concession protocols rely on sophisticated artificial intelligence methods to assure fair negotiations. The Contract Net Protocol allows for an iterative exchange of proposals where the parties can give up parts of their original demands and reach a mutually acceptable agreement. Due to its strict Requirement and Offer exchange, VersaNeg does not fit as a concession protocol. With VersaNeg one cannot modify a Requirement that has been stated during a previous round in the negotiation, instead one would have to restart the negotiation.

The policy intersection approaches are not intended for electronic contracting. WS-Agreement, Contract Net Protocol and VersaNeg are suitable for electronic contracting because they can express the business level objectives and requirements. Of course, these approaches must also apply cryptography to assure that agreements can also be considered as evidence in front of court. None of these three approaches can stand on its own, there are still legal framework contracts required that define how to interpret agreements reached under these protocols. This leads to a situation where the static and invariant parts of an agreement that also require human interpretation can be defined in the legal framework contract. The variable parts of the agreement and the parameters that are required for electronic processing can be handled by these protocols.

The message exchange efficiency states how much overhead a protocol introduces during a negotiation for the transmission of the given information in the final agreement. An ideal protocol with regards to efficiency would transmit each unique piece of information only once. The policy intersection protocols introduce redundancy by first sending a complete policy that probably contains too many options, and by returning the set of compatible parameters. The WS-Agreement sends a template that will be returned with a few additionally inserted data values. Negotiation protocols are network delay sensitive. Hence, these three protocols are not that bad, because even though they waste bandwidth with the redundant transmission of information, the single-round characteristic leads to fast negotiation exchange. The Contract Net Protocol can in principal realize an efficient negotiation, however, as the protocol does not encourage efficiency, it is up to the negotiation systems. VersaNeg introduces the most redundancy by transmitting the same data repeatedly during a multilateral multi-round negotiation. The iterative refinement of VersaNeg can be an advantage over the single-round protocols, as the number of possible agreement options can be very large and iterative refinement allows the parties to only transmit the details of a certain option under negotiation; instead of all options simultaneously as with the single-round protocols. In the future, VersaNeg could be extended for incremental message exchange, by transmitting each information only for one round.

A central motivation of research on electronic negotiations is the ability to automate electronic negotiations. Obviously, the ability to automate a negotiation depends on the legal and economic risk that an automated negotiation introduces. The question here is how much of this risk can be already excluded at a protocol level and how much of this risk must be handled by the negotiation systems. Policy intersection protocols

introduce comparable small risk because all possible agreements are already contained in the local policies. Nevertheless negotiation systems using policy intersection must still check if a given combination of options is also acceptable. WS-Agreement can reduce risk significantly by performing strict comparisons of the agreement against the template and by restricting the possible data values already in the template.
The Contract Net Protocol is very open and gives much freedom for the message usage for the negotiation systems. The negotiation systems are responsible for understanding everything under negotiation. One call for proposals can be answered by a completely independent proposal by another party. It is now up to the negotiation system to understand the interrelationship between the call for proposal and the received proposal. Because the protocol has so many freedom degrees, the negotiation system cannot use the protocol to restrict the parameters under negotiation, to a parameter set that is fully understood and where the risk is well-known.
VersaNeg also gives much freedom during the Agreement Discovery. This freedom introduces the risk that the negotiation systems end up with an agreement that does not fit their objectives. The VersaNeg approach has two techniques in place to limit that risk: Requirements largely define already how an Offer can look like. The close link between Requirement and Offer allows the negotiation system to limit possible Offers to a well-known parameter space. Additionally Requirements can be linked to a contractual framework that defines the interpretation of a Requirement and of matching Offers. Iterative refinement is the second technique that also helps in reducing the risk. One negotiation starts with broad Requirements that are amended with more specific Requirements over the course of the negotiation. By stating additional Requirements, the negotiation system can eliminate risk and ambiguity from agreement options. Therefore, VersaNeg has a lower risk than Contract Net and a higher risk than WS-Agreement and policy intersection. However, VersaNeg can discover new and unforeseen agreements whereas WS-Agreement and policy intersection are limited to their predefined policies and templates. Hence, the additional risk of VersaNeg comes also with a higher functionality during the agreement discovery

8.3 Additional Characterizations of ESAF Policy Intersection and VersaNeg Agreement Negotiation

The comparisons above give already a good overview of the properties of the novel negotiation approaches, presented in this thesis. There are some more criteria to characterize the nature of the ESAF policy intersection and the VersaNeg agreement negotiation protocols.

The ESAF policy intersection creates bilateral technical dependencies whereas the VersaNeg can either form bilateral agreements to multilateral agreements for collaborations. It is interesting to note that even though ESAF policy intersection and VersaNeg bilateral agreement negotiation form agreements between two parties, negotiations with these protocols can still involve a group of parties. Indeed much research that claims to address multi-party negotiations has a number of underlying bilateral negotiations where only one agreement is chosen in the end. The outstanding feature of VersaNeg is that it can perform many-to-many negotiations to discover and define multilateral collaborations.

An important question is what common knowledge the negotiation systems require to perform the Agreement Discovery. For ESAF policy intersection, the ESAF already defines for its use case how the policies can look like and how to interpret the poli-

Criteria	`ESAF` Policy Intersection	`VersaNeg` Bilateral Agreement Negotiation	`VersaNeg` Multilateral Agreement Negotiation
Dependencies	bilateral	bilateral	multilateral
Communication Pattern	one-to-one one-to-many	one-to-one one-to-many	many-to-many
Shared Syntax and Semantics	policy language	Requirements/ Offers/Obligations	Requirements/ Offers/Obligations
Individual Decisions	by drafting policy a-priori	make Offers, state Requirements	make Offers, state Requirements
Deciding Party	choose one alternative	choose one alternative	choose one alternative
Termination	after one round	deciding party	deciding party
Sensitive Information	no protection	iterative refinement, safe disclosure sequence	iterative refinement, safe disclosure sequence, attribute encryption

Table 8.3: *Comparison of Functionalities*

cies. The VersaNeg relies on a common understanding about Requirements, Offers, and Obligations. This common knowledge must be established outside the protocol, either through direct human input prior of during the negotiation or by standardization bodies for special business domains.

An interesting question is how negotiation systems and human operators can steer negotiations. With policy intersection, the policies are defined a-priori by the human operator. This limits the possible agreements to the scope the operator defines. The Requirements and Offers exchange allows the operator of a negotiation system to define Requirements and Offers a-priori. It could also be possible that a human operator specifies Requirements and Offers during the negotiation itself. Another choice is which Offer and Requirements should be made during the negotiation.

The ESAF policy intersection and the VersaNeg agreement negotiation both rely on one party that makes the final decision about which agreement option to pursue. Both types of negotiation only contain agreement options that are acceptable to all parties. The deciding party picks one option. This maps to real world use cases where the deciding party is also the party that usually pays the bill in the end. VersaNeg could be extended in the future to a multilateral commit protocol where all parties have to agree to one agreement option. This would give the individual parties more power to influence the agreement formation but it would make it also more difficult to arrive at an agreement.

Another important question is when a negotiation terminates. The policy intersection is guaranteed to reach a decision after one round. The VersaNeg protocol itself does not impose any restrictions on how long a negotiation can last. It is up to the negotiation systems to decide if they see no incentive to continue a negotiation. This is not different from human to human negotiations where each party can always walk out of the negotiation at any time before an agreement has been reached. An individual party that wants to get out of a multilateral negotiation can simply append a special unsatisfiable Requirement to all agreement options where it is involved or it can send

an explicit abort message. The deciding party can terminate the whole negotiation at any time.

The ability to protect sensitive information is also an important negotiation protocol property. It is comparable easy to protect against malicious third parties that eavesdrop on the negotiation by using secure communication channels, for instance, TLS with mutual authentication. The challenge is to limit the amount of information revealed towards the other negotiating parties.
Policy intersection protocols always reveal the whole policy, which might be undesirable. The same is true for WS-Agreement where also always the whole template is revealed. VersaNeg can rely on iterative refinement to limit the disclosure of the possible agreement options. VersaNeg does not reveal all possible Offers and Requirements but only the one that are deemed relevant by the negotiation system for a given negotiation. In multilateral negotiations, VersaNeg can encrypt certain Offers, Requirements, and Obligations so that they can only be accessed by authorized receivers in the negotiation. VersaNeg evolved from research on trust negotiation and still supports safe disclosure sequences to release only the minimum of sensitive information. Safe disclosure sequences in VersaNeg ensure that all Requirements are satisfied with a matching Offer from the remote party, before information about the own Offer is released. The support for safe disclosure sequences in a general purpose agreement negotiation protocol is another unique feature of VersaNeg.

8.4 Applying the Montreal Taxonomy for Electronic Negotiations

The comparison above highlights the most important distinguishing factors between the three approaches to agreement negotiation. The *Montreal Taxonomy for Electronic Negotiations* [MiCh03] allows for a more comprehensive assessment of the protocols. The Table 8.4 below shows the classification of each protocol according to the taxonomy. Each row in the table contains a criterion from the taxonomy and a brief description. The descriptions of the criteria that are relevant to the discussed protocols are taken from the publication [MiCh03].
The Montreal Taxonomy is highly applicable for analyzing auctions, and uses the notion of offers with a different interpretation than in this thesis. Offers can be accepted or answered with a counter offer. Nevertheless, this taxonomy contains a broad set of criteria to differentiate protocols that are also applicable to general purpose negotiation protocols. All Proposals (including the Requirements/Offers/Obligations) under the VersaNeg protocol can be considered as offers under the Montreal Taxonomy.
Distinguishing features of a protocol that give an edge over the other protocols under comparison are printed in bold green letters.

Criteria	Policy Intersection	`VersaNeg` Bilateral Agreement Negotiation	`VersaNeg` Multilateral Agreement Negotiation
1. Roles			
a) Participation	bilateral	bilateral	**multilateral**
	two sides/more than two sides engage in the negotiation		
b) Agents	**multiple agents**	one agent	**multiple agents**

	one agent per side/multiple agents per side can take part in the negotiation		
c) Admission	open	open	open
	no restriction on the admission of agents into the electronic negotiation		
d) Identity	**anonymous**	exposed	exposed
	agents in a negotiation process that are exposed have a unique identity		
e) Collusion	not applicable	not applicable	**approved**
	if collusion is approved, agents can collaborate in order to achieve mutual benefits		
2. Process - Overall Rules			
a) Variation	fixed	**flexible**	**flexible**
	the rules are fixed within the process/ the rules are flexible but the range of possible rules is defined a-priori		
b) Rounds	single-round	**multi-round**	**multi-round**
	in a single-round electronic negotiation the process is passed through only once		
c) Stages	single-staged	single-staged	single-staged
	in a single-staged negotiation the rules are the same from the beginning to the end of the process		
d) Concurrency	multiple-bilateral	multiple-bilateral	**multiple-multilateral**
	with concurrency, one agent could run multiple negotiation sessions at the same time		
3. Process - Offer Specification			
a) Attributes	multiple	**not constrained**	**not constrained**
	the number and kind of attributes might also be not constrained by any rule		
b) Values	multiple	multiple	multiple
	ranges or choices for attribute values can be subject to the negotiation in the multiple values case		
c) Relaxation	fixed	fixed	fixed
	an agent might be allowed to specify values as flexible		
d) Structure	flexible	**dynamic**	**dynamic**
	during the process execution the number and kind of attributes is flexible according to pre-defined rules. the number and kind of attributes can be changed in a dynamic, a priori unknown way during the process		
e) Relation	independent	independent	independent
	an independent electronic negotiation does not allow for any dependencies between agreements		
f) Object	multiple	bundled	bundled
	in the multiple objects case, the offer is bundled, if it allows an agent to specify different subsets of quantities/agents can specify multiple homogeneous or heterogeneous objects in one offer		
4. Process - Offer Submission			
a) Sides	single	multiple	multiple
	all (multiple) sides are allowed to submit and receive offers		

b) Position	single	**multiple**	**multiple**
	in a single-positioned negotiation, agents can assume only one position (or role), such as buyer or seller		
c) Activity	event-based	event-based	event-based
	a process with a restricted event-based activity rule will end if a certain event occurs, such as a period of inactivity		
d) Direction	one direction	**haphazard**	**haphazard**
	in a haphazard negotiation process the direction of offer submission might change within one phase		
5. Process - Offer Analysis			
a) Value	undefined	undefined	undefined
	in an undefined case, there is no rule regarding the value relation of a new offer to other current offers		
b) Threshold	undefined	undefined	undefined
	the analysis regarding the threshold is undefined if the process does not have any predefined thresholds (e.g. reservation price)		
6. Process - Offer Matching			
a) Schedule	triggered	triggered	triggered
	offer matching is triggered by events (e.g. the submission of an offer)		
b) Sorting	satisfying	satisfying	satisfying
	a satisfying sorting agents can define in their offers constraints towards the potential transaction partners, which have to be evaluated for the offers to match		
c) Evaluation	**ranking**	listing	listing
	in a ranking scenario, offers are ordered according to the preferences of an agent in order to find out the "best" offer among the set of matching offers		
d) Resolution	**defined**	forwarding	forwarding
	if no resolution or tie-breaking rules are defined, conflicts are forwarded and have to be resolved by the agents		
7. Process - Offer Allocation			
a) Distribution	discriminatory	discriminatory	discriminatory
	in a negotiation with discriminatory value distribution a distinct value (e.g. price) may be determined for each winning offer		
b) Provision	offer-dependent	offer-dependent	offer-dependent
	through offer-dependent value provision a winning bidder has to pay the price (provide the value) specified in the winning offer		
c) Configuration	open	mediated	mediated
	candidate offers with defined value ranges (e.g. delivery date before 20.10.2010) are resolved to offers with single values		
8. Process - Offer Acceptance			
a) Commitment	indicative	binding	binding
	offers in a process might be binding, meaning that they cannot be retracted, and the agents are forced to execute the transaction according to the agreement		
9. Information			
a) Communication	offer-restricted	**offer-extended**	**offer-extended**

	in an offer-extended case, an offer might be complemented with additional remarks (e.g. comments, inquiries)		
b) Transaction	impl. specific	impl. specific	impl. specific
	the history of past deals resulting from previous processes is available to the agents		
c) Negotiation	impl. specific	impl. specific	impl. specific
	the interaction history of the current process execution is available to the agents		
d) Transparency	protected	protected	**sealed**
	the status information is restricted to the agent to the moment of offer submission, i.e. in the form of submission feedback, and to the third party evaluating the (sealed) offers		
e) Trace	exposed	exposed	exposed
	in a negotiation with exposed trace, available information can be traced back to the associated agent (who still can be anonymous)		
f) Content	unrestricted	selected	selected
	only selected offer or status information (e.g. only quality but no price information) is provided by the electronic negotiation medium		
g) Timing	triggered	triggered	triggered
	information is refreshed if a certain event occurs (triggered)		
10. Strategy			
a) Fees	scenario specific	scenario specific	scenario specific
	negotiations might incur fees for the involved agents		
b) Arbitration	punishing	punishing	punishing
	punishing means that violations of scenario rules result in penalties (e.g. exclusion, behavior restrictions, or payments) for the agents		
c) Ratings	appraisal-based	appraisal-based	appraisal-based
	appraisal-based means that past behavior of the agents results in the assignment of certain properties (e.g. recommendation levels or punishments)		

Table 8.4: *Applying the Montreal Taxonomy for Electronic Negotiations*

8.5 Summary

There is no single best protocol for all negotiation needs. It depends on the nature of the negotiation and the freedom degrees for the definition of the final agreement which protocol is suitable for a given use case. One can derive a number of suggestions from the analysis in this Chapter on when to use which protocol.
Handshake protocols can use policy intersection between two parties as an efficient method to establish agreements that have a low risk. Negotiatiations about agreements that have a business impact should better be carried out by one of the protocols that are suited for electronic contracting. Negotiations in business use cases where there are few options that are known a-priori should rely on WS-Agreement.
ebXML CPA is a well established standard to formalize already existing contracts for the execution in electronic business processes. For iterative agreement negotiation that

is capable of discovering new and unforeseen agreements, choose either the FIPA Contract Net Protocol or VersaNeg. Intelligent negotiation systems that need all freedom degrees for realizing sophisticated negotiation strategies and electronic auctions, are better off with the Contract Net Protocol. For limiting the risk of the negotiation, for iterative refinement, for the integration of safe disclosure sequences, and for obtaining comprehensive agreements use VersaNeg. Negotiations about dynamic collaborations between multiple parties and for defining one comprehensive agreement for the collaboration should also rely on VersaNeg.

9. Conclusion

Protocols for price negotiations are in wide spread use in the Internet. Agreement negotiation protocols that go beyond price, and that are able to define the complex dependencies that real world agreements typically consist of, are still in their infancy. This thesis developed new protocols to discover and form agreements. Policy intersection is a good approach to establish consensus about well-known parameters between two parties. As long as the possible policy options are known before the negotiation starts, it is a low risk approach with limited implementation complexity. Iterative requirements-driven agreement negotiation addresses scenarios where at the start of the negotiation, no one can predict how an optimal final agreement could look like. The agreement negotiation protocol becomes the tool for discovering agreements and for concluding the agreement in a way that each party can proof the outcome. Iterative requirements-driven agreement negotiation is a novel approach that performs bilateral and multilateral agreement negotiation equally well. The following sections will summarize the merits and limitations of the negotiation protocols introduced in this thesis.

9.1 Contributions

This thesis developed and evaluated generic protocols for agreement negotiation to address the challenges described above. The main contributions of this thesis are as follows:

Policy intersection is a simple but powerful negotiation primitive. This thesis presents the use of XML based policy intersection for agreement negotiation. Policy intersection separates the negotiation protocol from the domain specific logic and makes the negotiation protocol generic. The novel ESAF relies on single-round policy intersection for an autonomic and layer independent setup of secure communication channels (see Chapter 4). Single-round policy intersection is more efficient than iterative policy intersection and it is less vulnerable to bad network conditions. ESAF demonstrates how to use policy intersection to establish compatible parameters for secure communication, how to employ a utility function to choose the most favorable parameter set, and how to establish a secure communication channel that satisfies the security requirements from multiple stakeholders.

Policy intersection is a good method if all parties know the possible parameter space beforehand and if they can formulate their policies accordingly. Policy intersection reaches its limits when there is no predefined policy that details all possible agreement options.

Bilateral iterative agreement negotiation allows two parties to define an agreement that satisfies the objectives of both parties. In Chapter 5, this dissertation presents VersaNeg as a generic iterative negotiation protocol and framework to define bilateral agreements through an iterative exchange of Requirements, Offers, and Obligations. The core idea of the protocol is that an Offer states what a party proposes, the Requirement is what a party wants in turn, and the dependencies between Requirements and Offers lead to a rooted tree structure of different agreement options. The negotiation protocol is the tool to discover possible agreement options. After one agreement option has been chosen, the Offers are turned into Obligations.
The VersaNeg framework does not impose any restrictions on how to choose between different agreement options. Depending on the scenario and the involved risk, this choice can either be made by a human, or fully automated through some utility function. The negotiation protocol conveys the commitment to one agreement option and thereby defines an unambiguous agreement. The reached agreement is comprehensive, that is, it contains all terms and conditions agreed upon during the negotiation. For the interpretation of the agreement, a third party does not need any other information about the negotiation and can rely solely on the information contained in the comprehensive agreement. This makes the VersaNeg protocol preferable over other research approaches, where a third party must interpret the whole message exchange to interpret the outcome of the negotiation.

Multilateral iterative agreement negotiation is a research area where only few approaches exist that are able to discover unforeseen agreement options. This dissertation introduces VersaNeg which is equally well suited for bilateral and multilateral agreement negotiation. VersaNeg for multilateral agreement negotiation uses a ring protocol to exchange the negotiation state with the requirements and offers between a group of parties (see Chapter 6). The ring topology assures that each party can learn about all expressed Requirements it might wish to make an offer for. One agreement option is chosen as the final agreement, either by unilateral decision or through consensus between all parties. The chosen agreement fully defines what the parties agreed upon during the negotiation. The result of a multilateral agreement discovery with VersaNeg is a comprehensive agreement state that contains all the dependencies between the different parties for each agreement option.

WS-Agreement integration shows how an existing agreement language can be used to express Requirements, Offers, and Obligations in the VersaNeg protocol. WS-Agreement serves as the language to define guarantee terms, business values, and how to monitor the agreement later on. The WS-Agreement fits well with the multilateral Requirements and Offers/Obligations exchange. The WS-Agreement standard defines the semantic of the WS-Agreement documents. The VersaNeg protocol is responsible for Agreement Discovery and Agreement Formation. VersaNeg defines the dependencies between the bilateral WS-Agreement documents and thereby forms comprehensive multilateral agreements.

Secure agreement negotiation is the prerequisite for the real world use of VersaNeg. Without security measures to protect the authenticity and integrity of negotiation messages, electronic negotiation protocols are not fit for real world use. This thesis presents iterative signatures and message state reduction as the security mechanism to protect bilateral and multilateral agreement negotiations. This security mechanism is the enabler for requirements-driven iterative agreement negotiation to reach legally binding comprehensive agreements. This security extension works well for VersaNeg, and it could also be adapted to other XML based iterative agreement negotiation protocols.

9.2 Future Work

There are three major directions for future work based on this research on iterative requirements-driven agreement negotiation.

Consensus about which Agreement Option to choose, can be reached through different algorithms. So far this thesis assumed that a single party can make the decision between the different agreement options. That maps well to scenarios where one party pays the price of the collaboration and wants full control. However, a distributed decision about possible agreement options could lead to better agreements in terms of risk and quality. Each party knows best how to evaluate Offers for its Requirements and is therefore well suited to make the choice. This could lead to the development of a distributed consensus building protocol where each party only makes the choice amongst the Offers for its own Requirements. Thereby, each party would be responsible for only a part of the choices amongst alternative options in the dependency tree. Distributed voting protocols, as studied for social choice problems, are another option for how to choose between different agreement options.

Intelligent decisions are important at several steps during the agreement negotiation, for instance, to decide which offer to make, which agreement option to choose, and how to implement an agreement. The negotiation protocol does not answer how to make decisions, instead it offers interfaces for a decision intelligence. The negotiation protocol is the tool to discover agreement options, to form agreements, and to commit to agreements. The current implementation of VersaNeg relies on policies and on a rule-based logic for the automation of the negotiations. Humans can take decisions that involve a risk.
This mechanism is limited to simple scenarios. Therefore, the current decision intelligence should be extended to provide automation for more demanding negotiations. With VersaNeg, research on the decision intelligence does not have to bother about inventing new negotiation protocols, but can focus instead on all aspects that are required to automate the negotiation process and limit the risk of automated decisions. Research on decision intelligence should target utility functions to evaluate alternative agreement options, methods to dynamically construct attractive offers for requirements, and artificial intelligence methods to achieve optimal results in competitive environments where a number of parties compete to make offers for the same requirement.

Performance improvements and re-use of negotiation states shall be evaluated. One measure would be to allow for a re-use of already executed negotiations. For instance, one party could restart an existing negotiation and each party would have to indicate if the negotiation state still represents its interests. This would allow for a number of agreements that are based on the same negotiation state.
Another measure would be to speed up the cryptography by employing incremental

cryptography [EkSm99, BeGG94]. As only small portions of the negotiation state are added and most of the negotiation state does not change anymore during the negotiation, it would be sufficient to sign only the new parts of the negotiation state.

Deployment in real world scenarios is the ultimate goal for each approach towards agreement negotiation. The agreement negotiation would have to integrate with real world business processes. Businesses would have to establish a shared understanding of domain specific Requirements, Offers, and Obligations. The protocol assures on a syntactic level that Requirements, Offers, and Obligations match. The individual business domains should define the semantics of a common set of Requirements, Offers, and Obligations to negotiate automatically. Even if this standardization does not exist, the protocol can still work by relying on a human operator for matching Requirements with Offers/Obligations. The situation in the latter case would be not worse than today, as currently human-to-human negotiations also require that the negotiators interpret the semantics of a contract. The advantage would be that the protocol can reuse certain decisions in the future and automate repetitive negotiation steps.
Bilateral agreements are predominant today, even when a number of interdependent parties collaborate. Bilateral agreements have the disadvantage that they only capture the scope of the bilateral dependencies and do not acknowledge the critical dependencies with other trading partners. This behavior makes a risk assessment of multilateral cooperations very difficult because the dependencies between the different partners are hidden.

9.3 Concluding Remarks

Nowadays, agreement negotiation is primarily a bilateral task between human stakeholders. Even in multi-party collaborations, the underlying agreements are predominantly bilateral agreements. If the market place does not change, there is probably only a small incentive to switch to electronic agreement negotiation. Electronic negotiations introduce a new risk to the agreement formation and require computer systems that can deal with the complexity of electronic agreements. It is challenging for human operators to model business objectives and to control electronic negotiations. The enforcement of electronic agreements might be more difficult than with traditional paper contracts. There are few lawyers specialized in enforcing electronic contracts and there is still no established consensus how to interpret all aspects of electronic agreements in front of court.
However, there is a trend towards more complex collaborations between companies. The number of potential trading partners increased tremendously since the emergence of the Internet. Collaborations are becoming short-lived, task-driven, and spontaneous. Human-to-human negotiations cannot cover the full spectrum of possible agreements in a timely manner. Requirements-driven iterative agreement negotiation could be the solution.
There is a trade-off in using electronic agreement negotiations for more than just price negotiation. Are the opportunities of using electronic negotiation big enough to justify the additional risk and effort?
The market might not adopt requirements-driven iterative electronic negotiation protocols, but if it does, it might change the way how businesses interact. The sheer number of potential trading partners could have a significant impact on the business model of many companies. Companies might discover new business opportunities during the negotiation. Especially in multilateral collaborations, completely unforeseen agreements

can be discovered during the negotiation, that might turn out to be better than any manually established collaboration. Bilateral agreements hide dependencies and introduce additional risk to collaborations. Multilateral agreements make these dependencies visible and allow for better risk control. However, the sheer complexity of the interdependencies makes it infeasible for human-to-human negotiations to establish complex multilateral agreements on the fly. Requirements-driven iterative agreement negotiation protocols open up new opportunities to create multilateral agreements dynamically and to obtain better control of multi-party collaborations.

Glossary

Adaptive Security Security mechanisms that can adapt dynamically to external parameters.

Agreement Accordance in sentiment, opinion, action, or purpose; harmony, concord; absence of dissension

Agreement Execution After an agreement has been reached, it must be fulfilled. See *Execution Phase*.

Agreement Negotiation The interactive process for consensus building during the *Agreement Phase*.

Agreement Phase The phase of the generic agreement formation model where parties try to reach consensus. The parties announce what they want and what they offer. The negotiation process continues until they reach consent and thereby establish an agreement, or if at least one party decides to abort the negotiation.

Authenticity The communication/single message is guaranteed to originate from the source it claims to be from.

Autonomic Elements that expose the self-* features are considered autonomic

Autonomous Autonomous elements are entities which are self-directed, self-reliant and possess intelligence with an own reasoning about an environment and act according to own decisions.

Business Intelligence Business Intelligence refers to skills, processes, technologies, applications and practices used to support decision making.

Business Process A set of linked business activities that take one or more inputs and transform them to create an output. Ideally, the transformation that occurs in the process should add value to the input and create an output that is more useful and effective to the recipient either upstream or downstream in the processing chain.

Confidentiality Prevent disclosure of information to unauthorized entities.

Connectionless There is no connection between two protocol messages. For each message a new connection is established and closed. Transparent reuse of connections as in HTTP1.1 can improve the performance without sacrificing the property of being connectionless.

E-Commerce is generally understood to span the whole range of business situations that are at least partially supported by a communication network such as the Internet

Execution Phase The phase of the generic agreement formation model where the parties fulfill what they have promised and monitor that they receive what is defined in the agreement.

Generic Agreement Formation Model A three phase model that separates important but distinct activities during the agreement formation process.

Information Phase The phase of the generic agreement formation model where the parties determine their goals and explore their environment. The parties gather an overview of what they want and what they offer.

Integrity Changes to information will be detected.

Malicious Third Party Third party that is unauthorized to interfere with the communication.

Negotiation Negotiation is a dialogue intended to resolve disputes, to produce an agreement upon courses of action, to bargain for individual or collective advantage, or to craft outcomes to satisfy various interests

Negotiation Intelligence The negotiation intelligence is responsible for taking decisions during all phases of agreement formation. This includes decisions about partner selection, offers to make, and goals to achieve, during a negotiation.

Non-Repudiation Party cannot deny statements it made.

Policy Intersection Protocol Protocol that relies on intersection of policies, usually employed as part of handshake protocols for establishment of session parameters

Risk Possibility of loss or damage.

Self-* Is a synonym for the commonly agreed autonomous properties: self-configuration, self-healing, self-optimization, self-protection.

Self-Configuration Automatic configuration of components.

Self-Healing Automatic discovery, and correction of faults.

Self-Optimization Automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements.

Self-Protection Proactive identification and protection from arbitrary attacks.

SLA Service Level Agreement define terms and conditions for service delivery and support. SLAs can contain performance guarantees, priorities, responsibilities, and warranties.

Stateless Protocol Protocols that allow for the implementation of stateless servers are called stateless. The single messages carry all state that a server needs for processing a message. Notice that stateless protocols may carry payload that references external state.

Stateless Server A stateless server can process protocol messages without knowledge about previous protocol messages. Protocols that allow for the implementation of stateless servers are called stateless protocol. Stateless servers have desirable scalability, load-balancing and fail-over properties.

A. Appendix: Notation for UML Diagrams

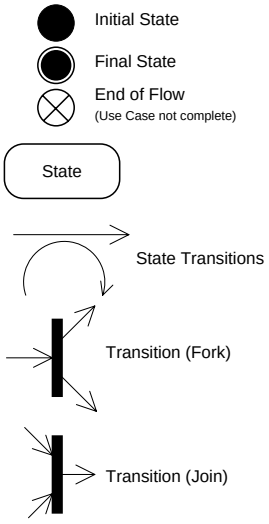

Figure A.1: *Notation of UML State Diagrams*

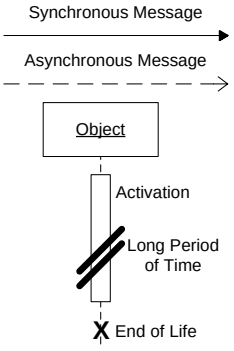

Figure A.2: *Notation of UML Interaction Diagrams*

B. Appendix: Requirement and Offer for Cloud Example

```
<Proposal xmlns:xsi=.. xsi:jsdl=..>
  <Owner>Party02</Owner>
  <Audience>Any</Audience>
  <Requirement>
    <Identifier>urn:cloud.com:res:137181</Identifier>
    <ApplySchema xmlns:ci="http://www.cloud.com/resource/cloud/instance/standards"/>
    <Constraint>
      <Location>//ci:NumberInstances</Location>
      <xs:restriction base="xs:integer">
          <ci:Exact>100</ci:Exact>
      </xs:restriction>
    </Constraint>
    <Constraint>
      <Location>//jsdl:IndividualNetworkBandwidth[@dir="upstream"]</Location>
      <xs:restriction base="xs:integer">
          <xs:mininclusive value="1048576"/>
      </xs:restriction>
    </Constraint>
    ...
  </Requirement>
  ...
</Proposal>
```

Listing B.1: *Example of Proposal with Requirement for Cloud Instances*

```
<Proposal xmlns:xsi=.. xsi:jsdl=..>
  ...
  <Offer xmlns:ci="http://www.cloud.com/resource/cloud/instance/standards" xmlns:jsdl=.. xmlns:jsdlposix=..>
    <jsdl:JobDefinition id="exampleInstanceAllocation">
      <jsdl:JobDescription>
        <jsdl:JobIdentification>...</jsdl:JobIdentification>
        <jsdl:Resources>
        <ci:MachineInstances>
          <ci:ID>urn:cloud.com:res:137181</ci:ID>
          <ci:InstanceCount>100</ci:InstanceCount>
          <ci:Image>
            <ci:Format>http://www.vmware.com/specifications/vmdk.html#sparse</ci:Format>
            <ci:VirtualSystemType>vmx-4</ci:VirtualSystemType>
          </ci:Image>
        </ci:MachineInstances>
        <jsdlposix:IndividualPhysicalMemory>
          4194304 </jsdlposix:IndividualPhysicalMemory>
        <jsdlposix:IndividualNetworkBandwidth
         dir="upstream">
```

```
            2097152
          </jsdlposix:IndividualNetworkBandwidth>
          ...
        </jsdl:Resources>
      </jsdl:JobDescription>
    </jsdl:JobDefinition>
  </Offer>
  ...
</Proposal>
```

Listing B.2: *Cloud Instance Offer example*

C. Appendix: WS-Agreement for Proposals

WS-Agreement can be used to model Requirement, Offer, and Proposal. The example in this chapter shows how the Requirement can use the WS-Agreement Template to largely define the expected services parameters, response times, and a penalty if access time becomes too slow. `NetworkBandwidth` is open to negotiation in this Template. The VersaNeg Offer has now an AgreementOffer that contains now the NetworkBandwidth and a list of IP ranges. The rest of the values from the Template remain the same. The Offer can now be turned into an Obligation with the only change that AgreementOffer becomes now an Agreement as part of the Obligation. Only during the Obligation Disclosure phase, the Obligation contains the credit card details that are requested by the CreationConstraint.

```
<Proposal xmlns:xsi=.. xsi:jsdl=..>
  <Owner>Party02</Owner>
  <Audience>Any</Audience>

  <Requirement requirementsID="c2">
      <Identifier>urn:cloud.com:res:137181</Identifier>

      <Template schemaLocation="http://schemas.ggf.org/graap/2007/03/ws-agreement http://schemas.ggf.org/
        graap/2007/03/ws-agreement">
          <Context>
              <Validity partOf="VersaNegAgreement">
                  <Condition>Only valid as part of this VersaNeg agreement</Condition>
                  <VersaNegAgreement>
                      <Objective>Cloud Resource Allocation</Objective>
                      <Initiator>Party01</Initiator>
                      <StartDate>04.05.2010</StartDate>
                      <GloballyUniqueID>99D0767E-F20E-11DF-AC4D-F934DFD72085
                      </GloballyUniqueID>
                  </VersaNegAgreement>
              </Validity>
          </Context>

          <Terms>
              <All>

                  <ServiceDescriptionTerm Name="ServiceAccessParameters" ServiceName="
                    ServiceAccessParameters">
                      <JobDefinition id="ServiceAccessParameters">
                          <JobDescription>
                              <JobIdentification>
```

```
                        <JobName>network access to service from different places in the world</
                          JobName>
                    </JobIdentification>

                    <Resources>
                        <IndividualNetworkBandwidth></IndividualNetworkBandwidth>
                    </Resources>
                </JobDescription>
            </JobDefinition>
        </ServiceDescriptionTerm>

        <ServiceDescriptionTerm Name="PenaltyPaymentDetails" ServiceName="
          PenaltyPaymentDetails">
        </ServiceDescriptionTerm>


        <GuaranteeTerm Name="AccessTimeGuarantee">
            <QualifyingCondition>applied when current time in week working hours</
              QualifyingCondition>

            <ServiceLevelObjective>
                <CustomServiceLevel></CustomServiceLevel>
                <KPITarget>
                    <KPIName>FastResponseTime</KPIName>
                    <CustomServiceLevel>//Variable/@Name="ResponseTime"<=1s
                    </CustomServiceLevel>
                </KPITarget>
            </ServiceLevelObjective>

            <BusinessValueList>
                <Penalty>
                    <AssessmentInterval>
                        <TimeInterval>P0Y1H</TimeInterval>
                    </AssessmentInterval>
                    <ValueUnit>EUR</ValueUnit>
                    <ValueExpression>25</ValueExpression>
                </Penalty>
            </BusinessValueList>
        </GuaranteeTerm>
    </All>
</Terms>

<CreationConstraints>
    <Item>
      <Location>//jsdl:JobDefinition[@id="ServiceAccessParameters"]/jsdl:JobDescription/jsdl:Resources
        /jsdl-posix:IndividualNetworkBandwidth[@dir="up"]</Location>
        <ItemConstraint>
            <minInclusive value="1048576"></minInclusive>
        </ItemConstraint>
    </Item>

    <Item>
      <Location>//ServiceDescriptionTerm[@id="PenaltyPaymentDetails"]/CreditCard</Location>
        <ItemConstraint>
          <xs:all>
            <xs:element name="Issuer" type="ccIssuersChoice"/>
            <xs:element name="HolderName" type="nameType"/>
            <xs:element name="CardNumber" type="ccCardNumber"/>
            <xs:element name="ValidUntil" type="ccValidUntil"/>
          </xs:all>
        </ItemConstraint>
    </Item>

</CreationConstraints>
</Template>
</Requirement>
...
</Proposal>
```

Listing C.1: *Example of Requirement with WS-Agreement*

```
<Proposal>
  <Owner>Party03</Owner>
  <Offer>
    <AgreementOffer schemaLocation="http://schemas.ggf.org/graap/2007/03/ws-agreement http://schemas.
      ggf.org/graap/2007/03/ws-agreement">
        <Context>
            <ServiceProvider>Party03</ServiceProvider>
            <Validity partOf="VersaNegAgreement">
                <Condition>Only valid as part of this VersaNeg agreement</Condition>
                <VersaNegAgreement>
                    <Objective>Cloud Resource Allocation</Objective>
                    <Initiator>Party01</Initiator>
                    <StartDate>04.05.2010</StartDate>
                    <GloballyUniqueID>99D0767E-F20E-11DF-AC4D-F934DFD72085
                    </GloballyUniqueID>
                </VersaNegAgreement>
            </Validity>
        </Context>

        <Terms>
            <All>
                <ServiceDescriptionTerm Name="ServiceAccessParameters" ServiceName="
                  ServiceAccessParameters">
                    <JobDefinition id="ServiceAccessParameters">
                        <JobDescription>
                            <JobIdentification>
                                <JobName>network access to service from different places in the world
                                </JobName>
                            </JobIdentification>

                            <Resources>
                                <IndividualNetworkBandwidth>1048576</
                                  IndividualNetworkBandwidth>
                            </Resources>
                        </JobDescription>
                    </JobDefinition>
                </ServiceDescriptionTerm>

                <ServiceDescriptionTerm Name="PenaltyPaymentDetails" ServiceName="
                  PenaltyPaymentDetails">
                </ServiceDescriptionTerm>

                <GuaranteeTerm Name="AccessTimeGuarantee">
                    <QualifyingCondition>applied when access time is in working hours of week
                    </QualifyingCondition>
                    <ServiceLevelObjective>
                        <CustomServiceLevel></CustomServiceLevel>
                        <KPITarget>
                            <KPIName>FastResponseTime</KPIName>
                            <CustomServiceLevel>//Variable/@Name="ResponseTime"<=1s
                            </CustomServiceLevel>
                            <CustomServiceLevel type="MonitorDomains">
                              <Monitor>177.12.33.0/24<Monitor>
                              <Monitor>134.12.12.0/16<Monitor>
                               ....
                            </CustomServiceLevel>
                        </KPITarget>
                    </ServiceLevelObjective>

                    <BusinessValueList>
                        <Penalty>
                            <AssessmentInterval>
                                <TimeInterval>P0Y1H</TimeInterval>
                            </AssessmentInterval>
                            <ValueUnit>EUR</ValueUnit>
                            <ValueExpression>25</ValueExpression>
                        </Penalty>
                    </BusinessValueList>
                </GuaranteeTerm>
            </All>
        </Terms>
    </AgreementOffer>
```

```
  </Offer>
  ...
</Proposal>
```

Listing C.2: *Example of Offer with WS-Agreement*

```
<Proposal>
  <Owner>Party03</Owner>
  <Obligation>
    <Agreement schemaLocation="http://schemas.ggf.org/graap/2007/03/ws-agreement http://schemas.ggf.org
      /graap/2007/03/ws-agreement">
        <Context>
            <ServiceProvider>Party03</ServiceProvider>
            <Validity partOf="VersaNegAgreement">
                <Condition>Only valid as part of this VersaNeg agreement</Condition>
                <VersaNegAgreement>
                    <Objective>Cloud Resource Allocation</Objective>
                    <Initiator>Party01</Initiator>
                    <StartDate>04.05.2010</StartDate>
                    <GloballyUniqueID>99D0767E-F20E-11DF-AC4D-F934DFD72085
                    </GloballyUniqueID>
                </VersaNegAgreement>
            </Validity>
        </Context>

        <Terms>
            <All>
                <ServiceDescriptionTerm Name="ServiceAccessParameters" ServiceName="
                  ServiceAccessParameters">
                    <JobDefinition id="ServiceAccessParameters">
                        <JobDescription>
                            <JobIdentification>
                                <JobName>network access to service from different places in the world
                                </JobName>
                            </JobIdentification>

                            <Resources>
                                <IndividualNetworkBandwidth>1048576
                                </IndividualNetworkBandwidth>
                            </Resources>
                        </JobDescription>
                    </JobDefinition>
                </ServiceDescriptionTerm>

                <ServiceDescriptionTerm Name="PenaltyPaymentDetails" ServiceName="PenaltyPaymentDetails
                  ">
                    <CreditCard>
                        <Issuer>VISA</Issuer>
                        <HolderName>Hans, Tester</HolderName>
                        <CardNumber>1232-1234-1212-2112</CardNumber>
                        <ValidUntil>04/2011</ValidUntil>
                    </CreditCard>
                </ServiceDescriptionTerm>

                <GuaranteeTerm Name="AccessTimeGuarantee">
                    <QualifyingCondition>applied when access time is in working hours of week
                    </QualifyingCondition>
                    <ServiceLevelObjective>
                        <CustomServiceLevel></CustomServiceLevel>
                        <KPITarget>
                            <KPIName>FastResponseTime</KPIName>
                            <CustomServiceLevel>//Variable/@Name="ResponseTime"<=1s
                            </CustomServiceLevel>
                            <CustomServiceLevel type="MonitoDomains">
                               <Monitor>177.12.33.0/24<Monitor>
                               <Monitor>134.12.12.0/16<Monitor>
                                ....
                            </CustomServiceLevel>
                        </KPITarget>
                    </ServiceLevelObjective>
```

```
                    <BusinessValueList>
                        <Penalty>
                            <AssessmentInterval>
                                <TimeInterval>P0Y1H</TimeInterval>
                            </AssessmentInterval>
                            <ValueUnit>EUR</ValueUnit>
                            <ValueExpression>25</ValueExpression>
                        </Penalty>
                    </BusinessValueList>
                </GuaranteeTerm>
            </All>
        </Terms>
    </Agreement>
  </Obligation>
  ...
 </Proposal>
</Proposal>
```

Listing C.3: *Example of Obligation with WS-Agreement*

D. Appendix: Schema Definition to validate Protocol Messages

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
      targetNamespace="http://example.org/negotiationProtocolSchema"
      xmlns="http://example.org/negotiationProtocolSchema">


  <!-- basic structure of negotiation messages:
       # AgreementNegotiation(1)
       ## Context(1)
       ## Terms(1)
       ## DisclosureSequence(1)
     numbers in brackets indicate how often element must be present
  -->
  <xs:element name="AgreementNegotiation">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Context"/>
        <xs:element ref="Terms"/>
        <xs:element ref="DisclosureSequence"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Terms">
    <xs:complexType>
      <xs:sequence>
        <!-- root element of dependency tree of proposals -->
        <xs:element ref="All"/>
      </xs:sequence>
      <xs:attribute name="agreementTermsNonce" use="required" type="xs:integer"/>
      <xs:attribute name="negNodeCount" use="required" type="xs:integer"/>
      <xs:attribute name="negStepCount" use="required" type="xs:integer"/>
      <xs:attribute name="resID" use="required" type="xs:NCName"/>
      <xs:attribute name="negPhase" type="xs:NCName"/>
    </xs:complexType>
  </xs:element>


  <!-- Conditionals define how Proposals relate -->
  <xs:element name="All" type="ConditionalType" />
  <xs:element name="ExactlyOne" type="ConditionalType" />
  <xs:element name="OneOrMore" type="ConditionalType" />
  <xs:element name="OnlyOne" type="ConditionalType" />
  <xs:complexType name="ConditionalType">
    <xs:sequence>
      <!-- Recursion: each Proposal has Conditional which might have Proposals ... -->
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="Proposal"/>
    </xs:sequence>
```

```
48      <xs:attribute name="nodeID" use="required" type="xs:NCName"/>
49      <xs:attribute name="originatorNodeID" use="required" type="xs:NCName"/>
50      <xs:attribute name="owner" use="required" type="xs:NCName"/>
51      <xs:attribute name="requirementsIDREF" use="required" type="xs:NCName"/>
52      <xs:attribute name="specType" type="xs:NCName"/>
53    </xs:complexType>
54
55
56    <!-- Proposal consists of:
57           # Requirement(1:n): for Initial Request
58           # Offer(1) and Obligation(0:1) and Requirement(0:1): for any Proposal after Initial Request
59           # Conditional(1)
60    -->
61    <xs:element name="Proposal">
62      <xs:complexType>
63        <xs:sequence>
64          <xs:sequence minOccurs="0" maxOccurs="1">
65            <xs:element ref="Requirement"/>
66            <xs:element name="Audience" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
67          <xs:sequence>
68          <xs:element name="Offer" type="ObligationType" minOccurs="0" maxOccurs="1"/>
69          <xs:element name="Obligation" type="ObligationType" minOccurs="0" maxOccurs="1"/>
70
71          <!-- append Proposals recursively beyond the conditional -->
72          <xs:choice minOccurs="1" maxOccurs="1">
73            <xs:element ref="ExactlyOne"/>
74            <xs:element ref="OneOrMore"/>
75            <xs:element ref="All"/>
76            <xs:element ref="OnlyOne"/>
77          </xs:choice>
78        </xs:sequence>
79      </xs:complexType>
80    </xs:element>
81
82
83    <!-- define what party wants through
84           + requirementsID(1): unique identifier
85           + offerTYPE(1): schema for validation of Offer/Obligation
86           # (Location/Constraint)(1): WS-Agreement style creation constraints based on XPATH in Location and
               restrictions in Constraint
87    -->
88    <xs:element name="Requirement">
89      <xs:complexType>
90        <xs:sequence minOccurs="0" maxOccurs="unbounded">
91          <xs:element name="Location" type="xs:string"/>
92          <xs:element name="Constraint" minOccurs="0" type="xs:string"/>
93        </xs:sequence>
94        <xs:attribute name="offerTYPE" type="xs:NCName"/>
95        <xs:attribute name="requirementsID" use="required" type="xs:NCName"/>
96      </xs:complexType>
97    </xs:element>
98
99    <!-- Obligations are defined through schemas specified in Requirement -->
100   <xs:complexType name="ObligationType">
101     <xs:sequence>
102       <!--- Obligation format not restricted by negotiation protocol -->
103       <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
104     </xs:sequence>
105   </xs:complexType>
106
107   <!-- defines agreement through sequence of nodeIDs -->
108   <xs:element name="DisclosureSequence">
109     <xs:complexType>
110       <xs:sequence>
111         <xs:element name="nodeID" minOccurs="0" maxOccurs="unbounded" type="xs:NCName"/>
112       </xs:sequence>
113     </xs:complexType>
114   </xs:element>
115
116
117   <!-- Context defines how to perform negotiation:
118        # FrameworkContract: reference to contract that defines legal interpretation of agreement
119        # DecidingParty(1): currently only method to choose one agreement option
```

```
120       # Group: definition of participants of negotiation,
121          + closedGroup=false indicates that each party might invite other parties during the negotiation
122       # State(1): information to support processing and for security of protocol
123     −−>
124     <xs:element name="Context">
125       <xs:complexType>
126         <xs:sequence>
127           <xs:element name="FrameworkContract" minOccurs="0" maxOccurs="1" type="xs:NCName"/>
128           <xs:element name="DecidingParty" type="xs:NCName"/>
129           <xs:element name="Group" minOccurs="0" maxOccurs="1">
130           <xs:complexType>
131             <xs:sequence>
132               <xs:element name="Party" type="xs:NCName" minOccurs="2" maxOccurs="unbounded"/>
133             </xs:sequence>
134             <xs:attribute name="closedGroup" use="required" type="xs:boolean"/>
135           </xs:complexType>
136           </xs:element>
137           <xs:element ref="State"/>
138         </xs:sequence>
139       </xs:complexType>
140     </xs:element>
141
142     <!−− helpful state for processing of negotiation message and security,
143           ChangedNodes might change during negotiation −−>
144     <xs:element name="State">
145       <xs:complexType>
146         <xs:sequence>
147           <xs:element ref="ChangedNodes"/>
148           <xs:element ref="Integrity"/>
149           <!−− replay protection −−>
150           <xs:element name="Nonce" type="xs:string"/>
151         </xs:sequence>
152       </xs:complexType>
153     </xs:element>
154     <xs:element name="ChangedNodes">
155       <xs:complexType>
156         <xs:sequence>
157           <xs:element name="ByOwner" type="ByOwnerType" maxOccurs="unbounded" />
158         </xs:sequence>
159       </xs:complexType>
160     </xs:element>
161
162     <!−− integrity protection and authenticity of messages −−>
163     <xs:element name="Integrity">
164       <xs:complexType>
165         <xs:sequence>
166         <xs:element maxOccurs="unbounded" name="Step">
167           <xs:complexType>
168             <xs:sequence>
169               <!−− Digital Signatures as described in Listing below go here
170                       bilateral negotiations: contains exactly 2 signatures
171                       multilateral negotiations: contains one signature for each(!) message
172               −−>
173               <xs:element minOccurs="0" ref="Signature"/>
174               <xs:element ref="ProcessingState"/>
175             </xs:sequence>
176             <xs:attribute name="negStepCount" use="required" type="xs:integer"/>
177             <xs:attribute name="owner" use="required" type="xs:NCName"/>
178           </xs:complexType>
179         </xs:element>
180         </xs:sequence>
181       </xs:complexType>
182     </xs:element>
183
184     <xs:element name="ProcessingState">
185       <xs:complexType>
186         <xs:sequence>
187           <xs:element name="ByOwner" type="ByOwnerType"/>
188         </xs:sequence>
189         <xs:attribute name="negPhase" use="required" type="xs:NCName"/>
190         <xs:attribute name="previousNegStepCount" use="required" type="xs:NMTOKEN"/>
191       </xs:complexType>
192     </xs:element>
```

```
193
194     <xs:complexType name="ByOwnerType">
195       <xs:sequence>
196         <xs:element name="NodeId" minOccurs="0" maxOccurs="unbounded" type="xs:NCName"/>
197       </xs:sequence>
198       <xs:attribute name="negStepCount" use="required" type="xs:integer"/>
199       <xs:attribute name="owner" use="required" type="xs:NCName"/>
200     </xs:complexType>
201
202 </xs:schema>
```

Listing D.1: *Recursive Schema Definition to validate messages of Negotiation Protocol*

```
1    ....
2    <!-- XML Signature as defined by standard -->
3    <xs:element name="Signature">
4      <xs:complexType>
5        <xs:sequence>
6          <xs:element ref="SignedInfo"/>
7          <xs:element name="SignatureValue" type="xs:base64Binary"/>
8          <xs:element name="KeyInfo">
9            <xs:complexType>
10             <xs:sequence>
11               <xs:element name="KeyName" type="xs:string"/>
12             </xs:sequence>
13           </xs:complexType>
14         </xs:element>
15       </xs:sequence>
16     </xs:complexType>
17   </xs:element>
18
19   <xs:element name="SignedInfo">
20     <xs:complexType>
21       <xs:sequence>
22         <xs:element name="CanonicalizationMethod">
23           <xs:complexType>
24             <xs:attribute name="Algorithm" use="required" type="xs:anyURI"/>
25           </xs:complexType>
26         </xs:element>
27         <xs:element name="SignatureMethod">
28           <xs:complexType>
29             <xs:attribute name="Algorithm" use="required" type="xs:anyURI"/>
30           </xs:complexType>
31         </xs:element>
32         <xs:element ref="Reference"/>
33       </xs:sequence>
34     </xs:complexType>
35   </xs:element>
36   <xs:element name="Reference">
37     <xs:complexType>
38       <xs:sequence>
39         <xs:element name="Transforms">
40           <xs:complexType>
41             <xs:sequence>
42               <xs:element name="Transform">
43                 <xs:complexType>
44                   <xs:attribute name="Algorithm" use="required" type="xs:anyURI"/>
45                 </xs:complexType>
46               </xs:element>
47             </xs:sequence>
48           </xs:complexType>
49         </xs:element>
50         <xs:element name="DigestMethod">
51           <xs:complexType>
52             <xs:attribute name="Algorithm" use="required" type="xs:anyURI"/>
53           </xs:complexType>
54         </xs:element>
55         <xs:element name="DigestValue" type="xs:string"/>
56       </xs:sequence>
57     </xs:complexType>
```

```
58    </xs:element>
59    ...
```

Listing D.2: *Extensions for Schema Definition to integrate XML Signatures*

Literature

[ABDF+05] Ali Anjomshoaa, Fred Brisard, Michel Drescher, Donal Fellows, An Ly, Stephen McGough, Darren Pulsipher und Andreas Savva. Job Submission Description Language (JSDL) Specification, Version 1̇0. Specification, GGF, November 2005.

[ACCF+02] Selim Aissi, Arvola Chan, James Bryce Clark, David Fischer, Tony Fletcher, Brian Hayes, Neelakantan Kartha, Kevin Liu, Pallavi Malu, Dale Moberg, Himagiri Mukkamala, Peter Ogden, Marty Sachs, Yukinori Saito, David Smiley, Tony Weida, Pete Wenzel und Jean Zheng. Collaboration-Protocol Profile and Agreement Specification Version 2.0. Specification, September 2002.

[ACDK+07] Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke und Ming Xu. Web Services Agreement Specification (WS-Agreement), 2007.

[ACKL+07] Alain Andrieux, Karl Czajkowski, Asit Dan Kate Keahey, Heiko Ludwig, Jim Pruyne, John Rofrano, Steve Tuecke und Ming Xu. Web Services Agreement Negotiation Specification (WS-AgreementNegotiation). Expired draft, Global Grid Forum, 2007.

[Altv99] Elmar Altvater. *The Future of the Market: An Essay on the Regulation of Money and Nature After the Collapse of Actually Existing Socialism.* Verso, Berlin. 1999.

[ApDe95] James G. Apple und Robert P. Deyling. *A Primer on the Civil-Law System.* LexisNexis UK; 4th Revised edition. 1995.

[ApDe98] James G. Apple und Robert P. Deyling. *The Principles of European Contract Law.* Prepared by the Commission on European Contract Law. 1998.

[Bach00] Louis Bachelier. *Théorie de la spéculation.* Gauthier-Villars. 1900.

[Baum01] Birgit Baum-Waidner. Optimistic Asynchronous Multi-party Contract Signing with Reduced Number of Rounds. In *ICALP '01: Proceedings of the 28th International Colloquium on Automata, Languages and Programming,*, London, UK, 2001. Springer-Verlag, S. 898–911.

[BBFL+02] Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, Ed Simon, Donald Eastlake, Joseph Reagle und David Solo. XML-Signature Syntax and Processing. W3C Recommendation, World Wide Web Consortium, February 2002.

[BBMM+07] Alberto Baragatti, Roberto Bruni, Hernán Melgratti, Ugo Montanari und Giorgio Spagnolo. Prototype Platforms for Distributed Agreements. *Electron. Notes Theor. Comput. Sci.* 180(2), 2007, S. 21–40.

[BeFS04] Elisa Bertino, Elena Ferrari und Anna Cinzia Squicciarini. Trust-X: A Peer-to-Peer Framework for Trust Establishment. *IEEE Transactions on Knowledge and Data Engineering* 16(7), July 2004, S. 827–842.

[BeGG94] Mihir Bellare, Oded Goldreich und Shafi Goldwasser. Incremental Cryptography: The Case of Hashing and Signing. In *CRYPTO '94: Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, London, UK, 1994. Springer-Verlag, S. 216–233.

[BeMc04] Dave Beckett und Brian McBride. RDF/XML Syntax Specification (Revised). Recommendation, February 2004.

[BeVe08] Morad Benyoucef und Marie-Hélène Verrons. Configurable e-Negotiation Systems for Large Scale and Transparent Decision Making. *INFORMS Journal of Group Decision and Negotiation Special Issue on E-Democracy* 17(3), May 2008, S. 211–224.

[BiDK89] Gautam Biswas, Kenneth A. Debelak und Kazuhiko Kawamura. Applications of qualitative modeling to knowledge-based risk assessment studies. In *IEA/AIE '89: Proceedings of the 2nd international conference on Industrial and engineering applications of artificial intelligence and expert systems*, New York, NY, USA, 1989. ACM, S. 92–101.

[BLCG92] Tim Berners-Lee, Robert Cailliau und Jean-François Groff. The World-Wide Web. *Computer Networks and ISDN Systems* 25(4-5), 1992, S. 454–459.

[BMLH07] Salima Benbernou, Hassina Meziane, Yin Hua Li und Mohand-Said Hacid. A Privacy Agreement Model for Web Services. In *2007 IEEE International Conference on Services Computing (SCC 2007)*, Salt Lake City, Utah, USA, July 2007. IEEE Computer Society, S. 196–203.

[BoGa88] A.H. Bond und L. Gasser. Distributed artificial intelligence. *Communication, Co*, 1988.

[Boye01] John Boyer. Canonical XML Version 1.0. W3C Recommendation, World Wide Web Consortium, January 2001.

[BPSME+04] Tim Bray, Jean Paoli, C. M. Sperberg Mcqueen, Eve und Francois Yergeau (Hrsg.). *Extensible Markup Language (XML) 1.0.* W3C Recommendation. W3C. August 2004.

[Bund01] Bundesnetzagentur. *Law Governing Framework Conditions for Electronic Signatures and Amending Other Regulations, inofficial version for industry consultation.* Berlin. 2001.

[Cami00] Martin W. A. Caminada. Towards a formal model for contract execution. In *Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling (LAP 2000)*, Aachen, September 2000.

[CCKH+01] J. Clark, C. Casanave, K. Kanaskie, B. Harvey, N. Smith, J. Yunker und K. Riemer. ebXML Business Process Specification Schema Version 1.01. Technischer Bericht, 2001.

[CCZY09] Xiangran Cheng, Xingyuan Chen, Bin Zhang und Yan Yang. An Algebra for Composing Access Control Policies in Grid. *Computational Intelligence and Security, International Conference on* Band 1, 2009, S. 526–530.

[CWZL05] Jian Cao, Jie Wang, Shensheng Zhang und Minglu Li. A Multi-agent Negotiation Based Service Composition Method for On-demand Service. In *SCC '05: Proceedings of the 2005 IEEE International Conference on Services Computing*, Washington, DC, USA, 2005. IEEE Computer Society, S. 329–332.

[DeFP97] Michael M. Delaney, Abbas Foroughi und William C. Perkins. An empirical study of the efficacy of a computerized negotiation support system (NSS). *Decis. Support Syst.* 20(3), 1997, S. 185–197.

[DFJN97] JE Doran, S. Franklin, NR Jennings und TJ Norman. On cooperation in multi-agent systems. *The Knowledge Engineering Review* 12(03), 1997, S. 309–314.

[DiRe06] T. Dierks und E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, Internet Engineering Task Force, 2006.

[DMRTV07] Giuseppe Di Modica, Valerio Regalbuto, Orazio Tomarchio und Lorenzo Vita. Dynamic re-negotiations of SLA in service composition scenarios. In *EUROMICRO '07: Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications*, Washington, DC, USA, 2007. IEEE Computer Society, S. 359–366.

[EkSm99] Patrik Ekdahl und Ben Smeets. Incremental Authentication of Tree-Structured Documents. In *ICICS '99: Proceedings of the Second International Conference on Information and Communication Security*, London, UK, 1999. Springer-Verlag, S. 275–283.

[ElES04] Bertino Elisa, Ferrari Elena und Anna Cinzia Squicciarini. Trust Negotiations: Concepts, Systems, and Languages. *Computing in Science and Engineering* 06(4), 2004, S. 27–34.

[fHum05] The Society for Human Resource Management. *The Essentials Of Negotiation*. Business Literacy for HR Professionals. Harvard Business School Press. 2005.

[FILX00] XU (C.) andGONG (F.), BALDINE (I.), HAN (L.) und QIN (X.). Building Security-Aware Applications on Celestial Network Security Management Infrastructure. 2000, S. 219–226.

[FIPA02] FIPA. FIPA Iterated Contract Net Interaction Protocol Specification. Technischer Bericht, FIPA TC Communication, December 2002.

[GaCo03] A. G. Ganek und T. A. Corbi. The dawning of the autonomic computing era. *IBM Syst. J.* 42(1), 2003, S. 5–18.

[GaMa99] Juan A. Garay und Philip D. MacKenzie. Abuse-Free Multi-party Contract Signing. In *Proceedings of the 13th International Symposium on Distributed Computing*, London, UK, 1999. Springer-Verlag, S. 151–165.

[GANZ98] GANZ (Z.) GANZ (A.), PARK (H.). Security Broker for Multimedia Wireless LANs: Design, Implementation and Testbed. 1998.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai und Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, New York, NY, USA, 2006. ACM, S. 89–98.

[GTMM+97] Frank Griffel, M. Tuan Tu, Malte Münke, Michael Merz, Winfried Lamersdorf und Miguel Mira da Silva. Electronic contract negotiation as an application niche for mobile agents. In *EDOC '97: Proceedings of the 1st International Conference on Enterprise Distributed Object Computing*, Washington, DC, USA, 1997. IEEE Computer Society, S. 354.

[Gull79] P. H. Gulliver. *Disputes and negotiations.* Studies on law and social control. Academic Press, New York. 1979.

[Hoer09] Thomas Hoeren. *Internetrecht. Stand: Sept. 2009.* Institut für Informations-, Telekommunikations- und Medienrecht, Münster. 2009.

[Holl09] Bernhard Hollunder. Domain-Specific Processing of Policies or: WS-Policy Intersection Revisited. *Web Services, IEEE International Conference on* Band 0, 2009, S. 246–253.

[IrLa02] Cynthia E. Irvine, Timothy E. Levin und Thuy D. Nguyen et al. Overview of a High Assurance Architecture for Distributed Multilevel Security. *Proceedings of the 2002 IEEE Workshop on Information Assurance and Security T1B2 1555 United States Military Academy, West Point, NY*, june 2002.

[JFLP+01] Nicholas R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, Michael Wooldridge und Carles Sierra. Automated Negotiation: Prospects, Methods and Challenges. *Group Decision and Negotiation* 10(2), 2001, S. 199–215.

[KCRS09] Andreas Klenk, Georg Carle, Benoit Radier und Mikael Salaun. Secure Stateless Trust Negotiation. In *IFIP Network and Service Security Conference*, Paris, France, June 2009.

[KeLu03] Alexander Keller und Heiko Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *J. Netw. Syst. Manage.* 11(1), 2003, S. 57–81.

[KeNT00] Gregory E. Kersten, Sunil J. Noronha und Jeffrey Teich. Are All E-Commerce Negotiations Auctions? In *In Proceedings of the 4th International Conference on the Design of Cooperative Systems*, 2000, S. 23–26.

[KiSe03] Jin Baek Kim und Arie Segev. A Framework for Dynamic eBusiness Negotiation Processes. *E-Commerce Technology, IEEE International Conference on* Band 0, 2003, S. 84.

[KMNG05] Andreas Klenk, Marcus Masekowsky, Heiko Niedermayer und GeorgCarle. ESAF - an Extensible Security Adaptation Framework. In *10th Nordic Workshop on Secure IT-systems, Tartu, Estonia*, October 2005.

[KNMC06] Andreas Klenk, Heiko Niedermayer, Marcus Masekowsky und Georg Carle. An Architecture for Autonomic Security Adaptation. *Journal Annals of the Telecommunications* 61(9/10), November 2006.

[KoPa06] Vladimir Kolovski und Bijan Parsia. WS-policy and beyond: application of owl defaults to Web service policies. In *Second International Semantic Web Policy Workshop (SWPW'06) at the 5th International Semantic Web Conference (ISWC)*, 2006.

[KPRS+07] Andreas Klenk, Frank Petri, Benoit Radier, Mikael Salaun und Georg Carle. Automated Trust Negotiation in Autonomic Environments. In *IWSOS*, 2007.

[Krau97] Sarit Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence* 94(1-2), 1997, S. 79 – 97. Economic Principles of Multi-Agent Systems.

[KrBC97] H. Krawczyk, M. Bellare und R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, 1997.

[Laco00] Gérard Lacoste (Hrsg.). *SEMPER - secure electronic marketplace for Europe*. Lecture notes in computer science. Springer, Berlin. 2000.

[Linn97] J. Linn. Generic Security Service Application Program Interface, Version 2. IETF, 1997.

[LuHo03] Heiko Ludwig und Yigal Hoffner. The Role of Contract and Component Semantics in Dynamic E-Contract Enactment Configuration. In *Proceedings of the IFIP TC2/WG2.6 Ninth Working Conference on Database Semantics*, Deventer, The Netherlands, The Netherlands, 2003. Kluwer, B.V., S. 19–33.

[MacD94] J.M. MacDonald. Does import competition force efficient production? *The Review of Economics and Statistics*, 1994, S. 721–727.

[MaMi01] Olivera Marjanovic und Zoran Milosevic. Towards Formal Modeling of e-Contracts. In *EDOC '01: Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing*, Washington, DC, USA, 2001. IEEE Computer Society, S. 59.

[Mase05] Marcus Masekowsky. *An Extensible End-To-End Security Adaptation Framework for Applications, Diploma Thesis*. Universität Tübingen. November 2005.

[MFRa02] Jean-Philippe Martin-Flatin und Sylvain Ravot. TCP Congestion Control in Fast Long-Distance Networks. Technischer Bericht CALT-68-2398, Air Force Electronic Systems Division, California Institute of Technology, USA, July 2002.

[MiBe05] Wolfgang Minker und Samir Bennacef. Speech and Human Machine Dialog. *Computational Linguistics* 31(1), 2005, S. 157–158.

[MiCh03] Ströbel Michael und Weinhardt Christof. The Montreal Taxonomy for Electronic Negotiations. *Group Decision and Negotiation* Band 12, 2003, S. 143–164.

[(MPG00] YARVIS (M.), REIHER (P.) und POPEK (G.). A Reliability Model for Distributed Adaptation, 2000.

[MPMB+04] David Martin, Massimo Paolucci, Sheila Mcilraith, Mark Burstein, Drew Mcdermott, Deborah Mcguinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan und Katia Sycara. Bringing Semantics to Web Services: The OWL-S Approach. In J. Cardoso und A. Sheth (Hrsg.), *SWSWPC 2004*, Band 3387 der *LNCS*. Springer, 2004, S. 26–42.

[MTIN04] Satoshi Makino, Kent Tamura, Takeshi Imamura und Yuichi Nakamura. Implementation and Performance of WS-Security. *Int. J. Web Service Res.* 1(1), 2004, S. 58–72.

[MüKC08] Andreas Müller, Andreas Klenk und Georg Carle. Behavior and classification of NAT devices and implications for NAT traversal. *IEEE Network* 22(5), 2008, S. 14–19.

[MüKC09] Andreas Müller, Andreas Klenk und Georg Carle. ANTS - A Framework for Knowledge based NAT Traversal. In *IEEE Globecom 2009 Next-Generation Networking and Internet Symposium*, Honolulu, Hawaii, USA, November 2009.

[Muss09] Neil Mussett. *German Civil Code BGB*. Bundesministerium der Justiz and Langenscheidt Translation Service, Berlin. 2009.

[NiKC06] Heiko Niedermayer, Andreas Klenk und Georg Carle. The Networking Perspective of Security Performance - a Measurement Study -. In *MMB 2006, Nürnberg, Germany*, March 2006.

[OVSH06] Nicole Oldham, Kunal Verma, Amit Sheth und Farshad Hakimpour. Semantic WS-agreement partner selection. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, New York, NY, USA, 2006. ACM, S. 697–706.

[PaGä99] Henning Pagnia und Felix C. Gärtner. On the impossibility of fair exchange without a trusted third party. Technischer Bericht TUD-BS-1999-02, Darmstadt University of Technology, mar 1999.

[Pasc05] A. Paschke. RBSLA a declarative rule-based service level agreement language based on RuleML. In *International Conference on Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, Band 2, 2005.

[PeFP06] Adrian Petcu, Boi Faltings und David C. Parkes. MDPOP: faithful distributed implementation of efficient social choice problems. In *Proceedings*

of the fifth international joint conference on Autonomous agents and multiagent systems, AAMAS '06, New York, NY, USA, 2006. ACM, S. 1397–1404.

[Petr05] Frank Petri. *Guarantees for Privacy in Automated Trust Negotiation, Diploma Thesis*. Universität Tübingen. December 2005.

[PrSc07] Cristian Prisacariu und Gerardo Schneider. A formal language for electronic contracts. In *In FMOODS 07 LNCS*, Band 4468. Springer, 2007, S. 174–189.

[PWZW08] Antoine Pichot, Oliver Wäldrich, Wolfgang Ziegler und Philipp Wieder. Dynamic SLA Negotiation Based on WS-Agreement. In *WEBIST 2008, Proceedings of the Fourth International Conference on Web Information Systems and Technologies, Volume 1, Funchal, Madeira, Portugal, May 4-7, 2008*. INSTICC Press, May 2008, S. 38–45.

[Rabi93] M. Rabin. Incorporating fairness into game theory and economics. *The American Economic Review* 83(5), 1993, S. 1281–1302.

[RaHu03] Sandro Rafaeli und David Hutchison. A survey of key management for secure group communication. *ACM Comput. Surv.* Band 35, September 2003, S. 309–329.

[ReTT04] Michael Rebstock, Philipp Thun und Omid Amirhamzeh Tafreschi. Supporting Interactive Multi-Attribute Electronic Negotiations with ebXML. In *Group Decision and Negotiation*. Springer Netherlands, 2004, S. 269–286 vol.12, num.4/july.

[RiSA78] R. L. Rivest, A. Shamir und L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21(2), February 1978, S. 120–126.

[ScLi98] Beat F. Schmid und Markus A. Lindemann. Elements of a Reference Model for Electronic Markets. *Hawaii International Conference on System Sciences* Band 4, 1998, S. 0193.

[SiSL04] Togar M. Simatupang, Indah Victoria Sandroto und S.B. Hari Lubis. Supply chain coordination in a fashion firm. *Supply Chain Management: An International Journal* 9(3), 2004, S. 256–268.

[SiWe89] John Simpson und Edmund Weiner. Oxford English Dictionary, 1989.

[SKFG06] Carsten Sinz, Wolfgang Küchlin, Dieter Feichtinger und Georg Görtler. Checking Consistency and Completeness of On-Line Product Manuals. *J. Autom. Reason.* 37(1-2), 2006, S. 45–66.

[SmSJ04] Bryan Smith, Kent E. Seamons und Michael D. Jones. Responding to Policies at Runtime in TrustBuilder. In *POLICY*, 2004, S. 149–158.

[Tarj71] Robert Tarjan. Depth-first search and linear graph algorithms. In *SWAT '71: Proceedings of the 12th Annual Symposium on Switching and Automata Theory (swat 1971)*, Washington, DC, USA, 1971. IEEE Computer Society, S. 114–121.

[TBMM04] Henry S. Thompson, David Beech, Murray Maloney und Noah Mendelsohn. XML Schema Part 1: Structures Second Edition. Recommendation, World Wide Web Consortium, October 2004.

[TeKe04] Gerald Tesauro und Jeffrey O. Kephart. Utility Functions in Autonomic Systems. In *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*, Washington, DC, USA, 2004. IEEE Computer Society, S. 70–77.

[TGRS+04] M. Tian, A. Gramm, H. Ritter, J. Schiller und R. Winter. A survey of current approaches towards specification and management of quality of service for web services. *PIK, vol. 27, no. 3*, 2004.

[Tilb06] Nadine Tilbury. Introduction to English Contract Law. 2006.

[Toka08] Nebahat Tokatli. Global sourcing: insights from the global clothing industry. The case of Zara, a fast fashion retailer. *Journal of Economic Geography* 8(1), January 2008, S. 21–38.

[Verm99] Dinesh C. Verma. Supporting Service Level Agreements on IP Networks. Macmillan Technical Publishing, 1999.

[vEZh08] Robert A. van Engelen und Wei Zhang. An Overview and Evaluation of Web Services Security Performance Optimizations. In *Proceedings of the 2008 IEEE International Conference on Web Services*, Washington, DC, USA, 2008. IEEE Computer Society, S. 137–144.

[VOHH+07] Asir S Vedamuthu, David Orchard, Frederick Hirsch, Maryann Hondo, Prasad Yendluri, Toufic Boubez und Umit Yalcinalp. Web Services Policy 1.5 - Framework. Recommendation, September 2007.

[VOHH+09] Asir S Vedamuthu, David Orchard, Frederick Hirsch, Maryann Hondo, Prasad Yendluri, Toufic Boubez und Ümit Yalçinalp. Web Service Security Policy 1.3. Standard, February 2009.

[VZMB+04] Laurentiu Vasiliu, Michal Zaremba, Matthew Moran, Christoph Bussler und Jim Browne. Web-Service Semantic Enabled Implementation of Machine vs.Machine Business Negotiation. *Web Services, IEEE International Conference on* Band 0, 2004, S. 106.

[WaSi05] Feng Wan und Munindar P. Singh. Formalizing and achieving multiparty agreements via commitments. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, New York, NY, USA, 2005. ACM, S. 770–777.

[WeXu03] Hans Weigand und Lai Xu. Contracts in E-Commerce. In *Proceedings of the IFIP TC2/WG2.6 Ninth Working Conference on Database Semantics*, Deventer, The Netherlands, The Netherlands, 2003. Kluwer, B.V., S. 3–17.

[Wiki09] Wikipedia. Negotiation — Wikipedia, The Free Encyclopedia, 2009. [Online; accessed 19-Sep-2009].

[WiLi00] William H. Winsborough und Ninghui Li. Automated trust negotiation. In *In DARPA Information Survivability Conference and Exposition, volume I*. IEEE Press, 2000, S. 88–102.

[WiLi02] William H. Winsborough und Ninghui Li. Protecting sensitive attributes in automated trust negotiation. In *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, New York, NY, USA, 2002. ACM Press, S. 41–51.

[Wini07] Michael Winikoff. Implementing commitment-based interactions. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, New York, NY, USA, 2007. ACM, S. 1–8.

[ws-s04] WS-Security Specification, March 2004.

[XiLS04] Li Xiong, Ling Liu und Ieee Computer Society. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. *IEEE Transactions on Knowledge and Data Engineering* Band 16, 2004, S. 843–857.

[XuJG05] Lai Xu, Manfred A. Jeusfeld und Paul W. P. J. Grefen. Detection tests for identifying violators of multi-party contracts. *SIGecom Exch.* 5(3), 2005, S. 19–28.

[ZhWi08] Charles C. Zhang und Marianne Winslett. Distributed Authorization by Multiparty Trust Negotiation. In *ESORICS '08: Proceedings of the 13th European Symposium on Research in Computer Security*, Berlin, Heidelberg, 2008. Springer-Verlag, S. 282–299.

[Zimm96] Reinhard Zimmermann. *The Law of Obligations: Roman Foundations of the Civilian Tradition*. Oxford University Press. august 1996.

[ZMFA04] Sheng Zhang, Fillia Makedon, James Ford und Lin Ai. *E-Commerce and Web Technologies*, Band 3182-2004 der *Lecture Notes in Computer Science*, Kapitel A Model for Multi-party Negotiations with Majority Rule, S. 228–237. Springer Berlin Heidelberg. December 2004.

ISBN 3-937201-32-7
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)
DOI: 10.2313/NET-2012-07-3