An Ontology-based Plug-and-Play Approach for In-Vehicle Time-Sensitive Networking (TSN)

Morteza Hashemi Farzaneh and Alois Knoll

***Abstract*— Time-Sensitive Networking (TSN) standards aim to support hard real-time communication with minimum latency based on Ethernet technology. They can also support the automotive domain considering upcoming and challenging application requirements in intelligent vehicles of the future. Despite all advantages, Time-Aware Shaper (TAS), a significant feature of TSN, increases the network configuration overhead. This configuration is required to guarantee deterministic network behavior.**

In this paper, an ontology-based approach is presented that extends TSN capabilities regarding Plug-and-Play and automatic network configuration. The developed ontology meta-models can be used by automotive experts (e.g. network designers or engineers) to design a concrete in-vehicle network based on TSN including knowledge that is required for automatic configuration.

I. Introduction

Time-Sensitive Networking (TSN) [1] is a set of standards in development by Institute of Electrical and Electronics Engineers (IEEE) targeting hard real-time communication with minimal latency based on Ethernet technology. These standards will help to overcome challenging requirements from automotive domain considering upcoming innovative applications such as autonomous driving and infotainment that require e.g. fully deterministic network behavior, high bandwidth, fail-operational and etc. Increasing the number of such applications and increasing number of sensors, actuators and *Electronic Control Units (ECU)* that are involved cause high network engineering overhead for the experts. Considering an in-vehicle network and its communication schedule, modifications can be required when a new application (including hardware and software) has to be integrated in the architecture. In worst-case, an network engineer has to recalculate all the scheduling-related parameters to be sure that all timing or bandwidth requirements are fulfilled.

In spite of all advantages of these standards, a concept for Plug-and-Play in TSN is missing to deal with the mentioned configuration overhead challenge. Time-Aware Shaper (TAS) [1] follows the time-driven scheduling principle and guarantees that high-priority data are not interrupted by low-priority data during the transmission. This requires a static predefined schedule that causes a lot of engineering work for the network configuration. For example, schedule of TAS gate drivers has to be manually configured based on the QoS requirements of the applications and involved data.

To confront this problem, an ontology-based approach is presented in this paper to enhance automatic network configuration of TSN in order to achieve Plug-and-Play abilities. **Contributions** are: (1) an overall Plug-and-Play procedure, (2) a proposal to integrate the approach in a data-centric middleware, (3) modeling details of applications, devices, data, QoS requirements and TSN features specially TAS in an ontology and (4) introducing a simple modeling use case. The rest of the paper is structured as follows: in the next section, the background and related work are discussed and in section III the Plug-and-Play procedure is described. The ontology models are explained in IV and finally in V the conclusion and future work are discussed.

Robotics and Embedded Systems, Technische Universität München { hashemif, knoll} @in.tum.de

II. Background and related work

A. Real-time Ethernet and Time-Sensitive Networking

Standard Switched Ethernet is a wide-spread, successful, low-cost, high-bandwidth and future-oriented infrastructure that is used in home and office networks. Due to the lack of real-time capabilities, standard Ethernet is not able to be used in applications with real-time requirements. The main issue is the best effort strategy of Ethernet which leads to indeterministic packet transmission that is not acceptable for time-critical applications. Based on the advantages of Ethernet, the factory automation domain started to adapt it for real-time communication. Examples for adapted Ethernet technologies such as Ethernet Powerlink, EtherCAT , Time-Triggered Ethernet and Profinet are analyzed and compared in [2]. Each of these technologies has advantages and shortcomings but all of them are similar in one point. All of them are almost proprietary solutions (except openPowerlink) and usually modify the standard Ethernet in hardware or in software stack to achieve real-time guarantees. Ethernet Powerlink for instance, uses obsolete network hubs to achieve the lowest latency and do not use the capabilities of switched Ethernet in duplex mode which is a waste of bandwidth. EtherCAT slave stack is implemented in *(Field Programmable Gate Array) FPGAs* and uses only the physical layer of the standard Ethernet and builds its protocol on top of it. Profinet IRT modifies the Ethernet switches in order to support isochronous hard real-time communication which increases the switch hardware costs significantly.

Based on the mentioned problems, IEEE decided to work on developing standard real-time Ethernet *Audio Video Bridging (AVB)* [3]. AVB proposed solutions in order to achieve real-time guaranties for the highest priority data class A (maximum latency of 2 ms) and Class B (maximum latency of 50 ms). The main use case for AVB was transmission of audio and video data in the entertainment domain. The timing

978-1-5090-0996-1/16/$31.00 ©2016 IEEE

guarantees were possible using the *Credit-based Shaper (CBS)*, *Stream Reservation Protocol (SRP)* and time synchronization. The idea of AVB motivated also other domains such as factory automation domain to contribute in further developing of new features to meet requirements of industrial communication. The motivation was to reduce the maximum latency down to microsecond range and to develop important additional features such as fault-tolerance mechanisms to meet safety-relevant requirements.

Based on this motivation, TSN task group [1] has been initiated. Various standards are still in development to achieve the mentioned objectives.

802.1 Qbv Time-Aware Shaper (TAS) proposes a time-triggered approach which uses gate drivers to prioritize data forwarding of the switches on each switch port. Thus, delay of high-priority and critical Ethernet frames in a switch is minimized. To achieve this, it is mandatory to (1) reserve enough bandwidth for such critical data using *802.1Qca Path Control and Reservation* and *802.1Qcc Stream Reservation Protocol (SRP) Enhancements and Performance Improvements* and (2) to develop a *Frame Preemption* mechanism called *802.1Qbu* to interrupt low-priority frame transmission to the advantage of high-priority frames. In this context, *Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks 802.1AS-Revision* plays a significant role to control the gate drivers. To support fault tolerance features, TSN proposes *802.1CB Frame Replication and Elimination standard*. The content of this paper is based on the current and updated drafts of the standards.

The mentioned standards focus more on fulfilling timing and QoS requirements such as minimum latency or support of fault-tolerance behavior with disjoint paths and not on automatic network configuration and Plug-and-Play functionalities.

B. Plug-and-Play in Real-time Ethernet

Discussed real-time Ethernet technologies suffer all from high manual configuration effort. There are several attempts to enhance Plug-and-Play capabilities of real-time Ethernet. Reinhart et al. [4] propose a five-step-model for reconfiguration of Ethernet Powerlink. However, the approach depends on specific components of the Ethernet Powerlink architecture and it is not clear how to apply it for other real-time Ethernet technologies. Moreover, it is not clear how the recently connected slave and network master know about the application logic and to which other slaves the data should be sent after connection. The approach presented in [5] proposes a Plug-and-Play solution for Profinet based on *Device Profile for Web Services (DPWS)* [6] and further in [7] based on *OPC Unified Architecture* [8]. These proposals also suffer from the generalizability issue and are not applicable for TSN. This issue is identified in [9] and general functionalities required for auto configuration are derived. The result is a four-phase procedure including IP-Connectivity, Device Discovery, Requesting RTE-specific parameters and Cloning engineering tool functionality.

The reviewed approaches focus on field bus systems that usually have similarities regarding network participants e.g. *Programmable Logic Controller (PLC)*, *Bus Coupler*, etc. In such systems, the network configuration and control application logic are implemented and executed on PLCs. The role of network switches are negligible in contrast to TSN where switches and their configuration play a significant role to meet the QoS requirements. Thus, a novel approach is required that focuses on the TSN properties and features (e.g. TAS) improving Plug-and-Play capabilities.

C. Ontology-based Network Management

There are several related works that apply ontologies for network management tasks [10], [11], [12], [13]. In [14] these proposals are analyzed regarding their advantages and shortcomings. The results show, that an ontology-based interoperability framework can help to ease several tasks in the network management. It relieves the network administrators and help to automate the management tasks. In [15] and [16] ontologies are applied to describe the network policies such as firewall policies. Management of complex 5G networks using ontologies is proposed in [17]. Martinez et al. [18] propose an ontology-based information extraction system for bridging the configuration gap in hybrid *Software-Defined Networks (SDN)*. The focus is on formalization of switch and router configuration domain to ease the configuration overhead for administrators.

To the best of our knowledge, there is no solution or proposal that deals with TSN and responds to the question of how to improve Plug-and-Play capabilities and automatic network configuration. It is justifiable because the standardization process of TSN is still in development and first products are expected in 2017.

III. Plug-and-Play Approach for TSN

In order to achieve Plug-and-Play capability with TSN the contributions of this paper are as follows. Firstly, required network services are defined and explained. These services interact with TSN knowledgebase that contains information about the applications, consumed and published data, QoS requirements, networked hardware (network hosts and switches), TSN specific device properties and TSN network configuration. Secondly, a procedure is defined that explains the required steps after a new device joins the network. Thirdly, the Plug-and-Play approach is embedded in a data-centric middleware to use its strengths regarding the higher application layer. Finally, using ontologies, the knowledgebase is explained in details.

A. Required Network Services

Based on the review of the literature, the following network services are defined which have to be available to support Plug-and-Play in TSN:

Device Discovery: is responsible for (1) assigning an adequate network address to the new device, (2) discovery of the device location in the network topology and finally (3) registration of device profiles.

Application Management: explores the device profiles and

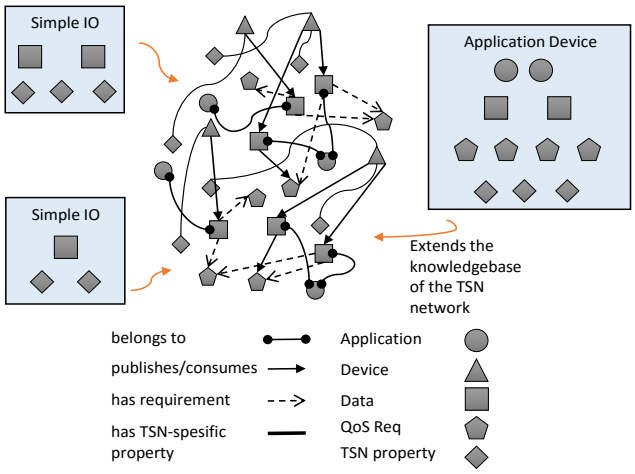

Fig. 1. Network Devices are analyzed by the application management service and the obtained new knowledge extends relevant parts (application, data, device, QoS requirement, device TSN-specific property) of the TSN network knowledgebase.

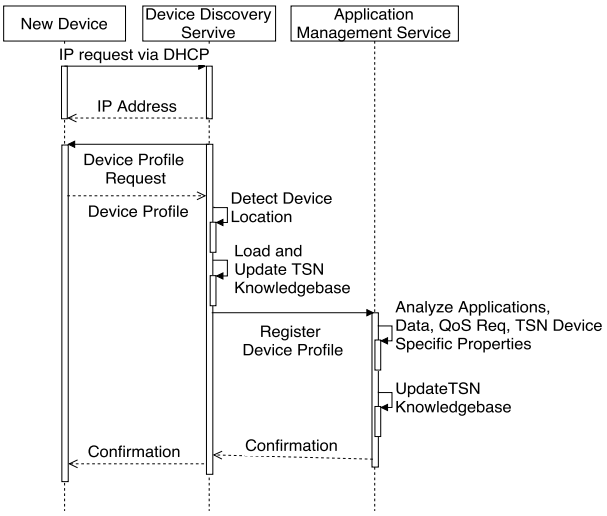

Fig. 2. Sequence diagram of the Plug Phase

updates (extends) the TSN knowledgebase including information about *applications*, published and consumed *data* of the *devices*, *QoS requirements* on those data and *TSN specific properties* of the devices. In the context of this paper, each application is defined by the data which are consumed or published by the application (a similar concept to [19] and [20]). Each application defines a set of QoS requirements on the consumed data. These requirements are exemplified by: Deadline, Durability, Destination Order, Reliability, Transport Priority. For example, reliability means that a data should be replicated on two or more disjoint network paths to guarantee the fail-operational for a specific safety-relevant application.

We define there device types in a TSN Network. *Simple IO devices* such as pure sensors or actuators their data is consumed/published by other applications of other devices, *Application devices* that have their own applications publishing or consuming data and TSN network *switches* transmitting the Ethernet frames in the network. **Figure 1** demonstrates how the device profiles are used by the application management service to update parts (related to applications, data, devices, QoS requirements and device TSN-specific property) of the TSN network knowledgebase.

TSN Plug-and-Play Management: is the core service to support Plug-and-Play. The main functionality of this service is to use reasoning methods on the available TSN network knowledgebase and derive commands for automatic network configuration based on predefined rules. Compared to the application management service which analyzes the relation between applications, devices, QoS requirements,etc., the TSN Plug-and-Play management deals with relation between TSN features and the requirements. It is also responsible to define logical rules so that the relation between chained TSN configuration parameters can be formalized. This formalization in the form of ontology-based knowledge representation allows reasoning to derive commands to configure the TSN devices (specially switches). The reasoning aims to replace step by step the TSN network administrator that has to master complex engineering tasks.

B. Plug-and-Play Procedure

The Plug-and-Play procedure is divided in (1) *Plug Phase* and (2) *Play Phase*. In the plug phase, a new network device, device discovery service, and application management service are involved. A recently joined device sends a *Dynamic Host Configuration Protocol (DHCP)* request to the network and device discovery service assigns an *Internet Protocol (IP)* to the device. Device discovery service sends a device profile request to the new device and it responds with its profile that already has been discussed. Another task of the device discovery service is to find the location of the new device in the current network topology. This information is required when TSN Plug-and-Play management service wants to find out which devices are relevant to send TSN configuration parameters. Here, knowledge about physical connection of the devices is significant. *Link Layer Discovery Protocol (LLDP)* [21] is used to manage this task. The obtained information is used to update the knowledgebase. Finally, the device discovery service sends a registration request including the new device profile to the application management service and it obtains all the new knowledge from the profile and extends the TSN knowledgebase. **Figure 2** demonstrates the sequence diagram of the plug phase.

In the play phase, after a network application is triggered, an execution request is sent to the application management service. It initializes and sends a TSN configuration request to the TSN Plug-and-Play management service. First, the TSN knowledgebase is loaded. Second, the involved TSN devices are extracted applying queries on the knowledgebase about the application and published or consumed data and consequently the involved devices (data is published or consumed by device). Third, based on the QoS requirements, queries are made about the affected TSN features. For example, if there is a deadline requirement, the Time-aware

shaper, frame preemption and stream reservation features are affected. Such relations are predefined in the TSN knowledgebase. Fourth, based on the current TSN configuration parameters (e.g. the event list of the gate driver in the Time-aware shaper feature) and predefined rules new configuration commands are derived and sent to the involved devices and the knowledgebase is updated after modification of configuration parameters. Finally, application management service and the application are notified that the TSN configuration is accomplished successfully. The play phase is demonstrated in **Figure 3** using a sequence diagram.

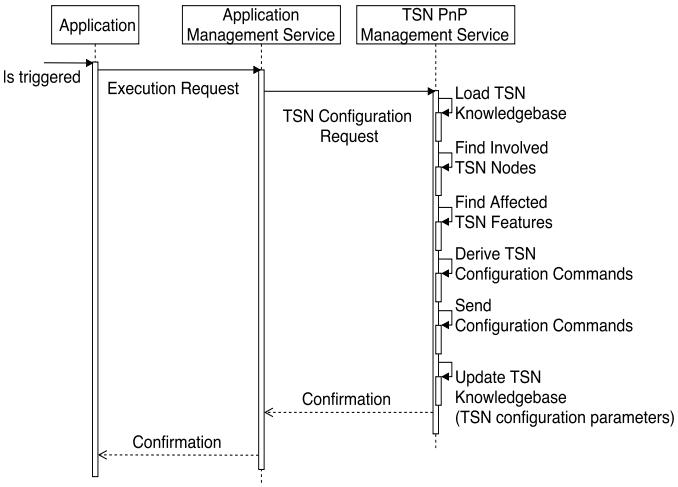

Fig. 3. Sequence diagram of the Play Phase

C. Integration in a Data-centric Middlware

The approach presented in this paper deals with the question how to configure a TSN network automatically so that all QoS requirements of the applications are fulfilled and the network engineering effort is reduced. Considering a TSN network, however there other tasks that are less related to network configuration. For example, application execution, device resource management, data handling, etc. Such tasks are mostly done by a middleware which offers abstractions (from hardware, operating system, etc.) to applications. Data-centric middleware such as *Data Distribution Service (DDS)* [19], [20] and service-oriented middleware such as DPWS and OPC UA (mentioned previously), continuously gain in importance in the communities of computer networks, automation, automotive, robotics and medical. As mentioned before, one of the key points of the presented approach here is data. Data is related to all other elements (see Figure 1). Data is in relation to applications, devices and QoS requirements. TSN automatic configuration is intended to fulfill QoS requirements of a Data with less network configuration effort. In order to have a full Plug-and-Play support in a TSN network, we propose to integrate the automatic configuration approach based on the TSN network knowledgebase into the data-centric middleware CHROMOSOME [20] to extend its capabilities.

The main component of the CHROMOSOME middlware is its run-time environment *XME*. There are the following main components in XME: (1) *Data Handler* that offers an API to the applications to publish and subscribe their data in the network, (2) *Broker* that monitors the availability of the data in the Data Handler and checks if the QoS requirements are fulfilled and (3) *Execution Manager* that executes the applications which are enabled by the broker.

Login Manager and *Login Client* are defined to recognize when a new device joins. The device discovery service of the proposed approach is similar to these components and can directly be integrated. The device discovery service, however has an additional task to find the physical topology of the new device using LLDP as applied in [21]. The middleware also implements *Plug&Play Manager* and *Client*. The Plug&Play client is responsible to announce the pluggable application components to the Plug&Play Manager. The manager calculates the available network bandwidth and changes the XME configuration when all bandwidth requirements can be fulfilled. The information exchange between client and manager is accomplished using manifests. Compared to the solution proposed in this paper, the manifests are the device profiles. The device discovery service extends Login Manager and Client. Also application management service and TSN Plug-and-Play Management service extend Plug&Play Manager and Client of CHROMOSOME. The architectural components are demonstrated in **Figure 4**.

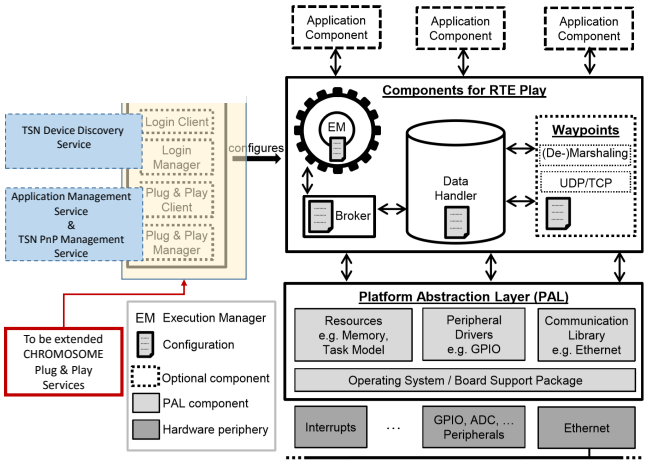

Fig. 4. CHROMOSOME architecture with its components and to be extended Plug-and-Play features. Figure is adapted from Buckel et al. [20]

IV. Ontology-based Modeling of TSN

A. Approach overview

Automatic configuration of TSN components requires acquiring knowledge about the available components and their properties in the network. Additionally, some rules are required to define the relation between those components. Based on the available knowledge, a reasoner is able to infer interesting new knowledge that is used for automatic configuration and enhancing the Plug-and-Play capabilities. As discussed in the related work, ontologies and reasoning

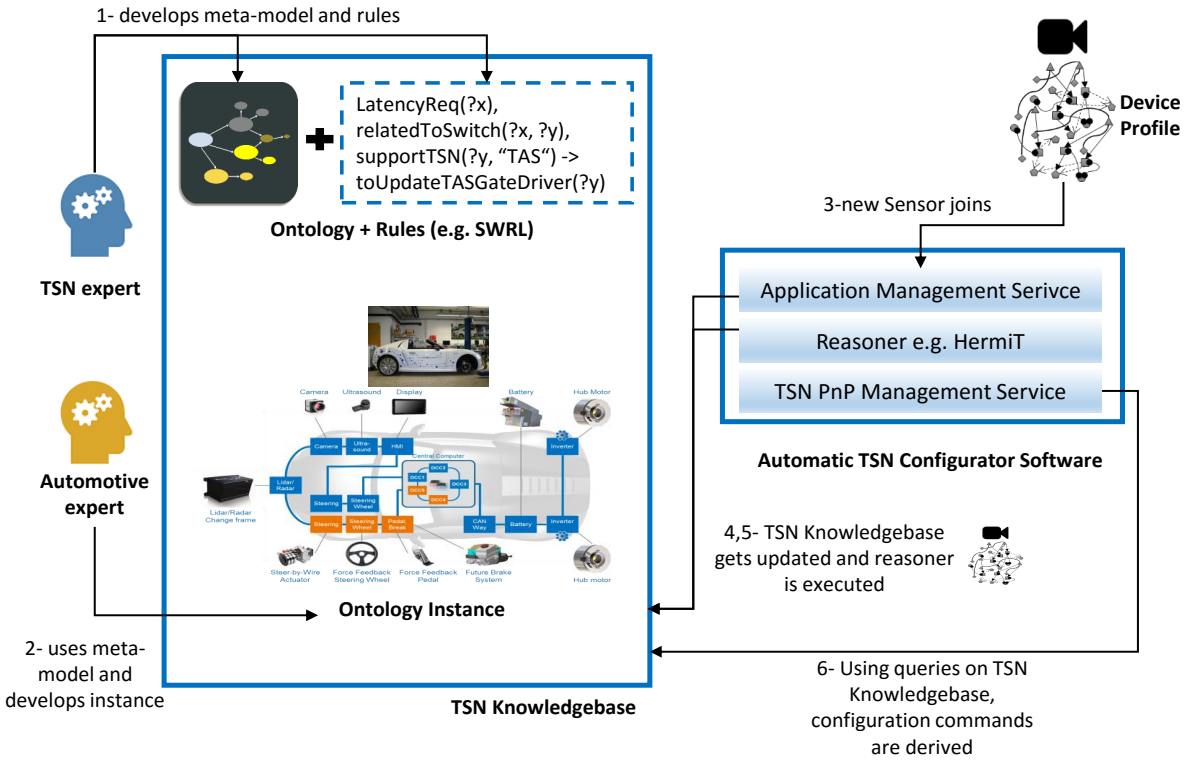

Fig. 5. Ontology-based approach for automatic TSN configuration. The figure uses RACE [22] in-vehicle communication architecture as an example that can be an use case of the developed approach in this paper.

have been successfully used for network management tasks and they can also be applied in TSN automatic configuration procedure. **Figure 5** depicts the details of the proposed procedure.

First, TSN ontology and configuration rules are defined and developed by an *TSN Expert* (this is one the contribution of this paper). Second, a *Domain Expert* (in this case an automotive engineer) uses the defined classes and relations from the TSN ontology as meta-model and develops a TSN network *Instance* (individual) e.g. a concrete in-vehicle network of a specific car. The union of the ontology (including classes, object and data properties), rules and instance is called here *TSN Knowledgebase*. Third, when a new device joins the network, Application Management Service uses the device profile of the recently joined device and updated the TSN knowledgebase and executes the reasoner. Finally, after reasoning, the knowledgebase is updated and TSN Plug-and-Play Management Service do queries on the knowledgebase to derive configuration commands. Except the first and second step, no other human interactions are required to achieve the automatic configuration. Instead of using SWRL as rule definition language we plan to transform the ontology model into a Logic programming (LP) model such as Prolog. Considering the advantages of LP-models regarding expressiveness, the generated facts and rules can be used for verification and configuration use cases in the network.

B. Modeling of devices, topics and requirements

The TSN knowledgebase consists of information about topics, devices, switches, QoS requirements and TSN general features. Using ontologies, these information are modeled so that they can be used as meta-model to initiate individuals. *Topic* class is the super class of *Domain* for existing data domains such as airbag domain which is depicted by *dom_airbag*, the cameras domain represented by *dom_cam* and the media domain for multimedia devices used inside of the car, shown by *dom_media*. The *t_topic* representation is used for different generated traffic streams by each device in the network. *t_topic* is also a subclass of *QoS* and *Topic*.

To each *Device* there is at least one *Port* assigned. The relation between ports and devices is defined as object property *isLinked*. Similarly, device behavior when *publishes* or *consumes t_topic* is defined as object property. A *t_topic* can have *QoS* requirements such as *vlan* showing its priority, *size* of each frame, *Topic_type* for periodic or event-based behavior, *period* of frame arrival and QoS type, which in this scenario appears to be latency, described with the data property *QoS_type*.

These relations are depicted in **Figure 6**. The knowledge about the generated data topics by each device and QoS requirements of data, plays a significant role in the procedure of automatic network configuration. This knowledge is also modeled in the ontology. For instance, QoS requirement *latency* is depicted through data property *Qos_type* within each topic. Similarly, Qos requirements like *publisher* and *con-*

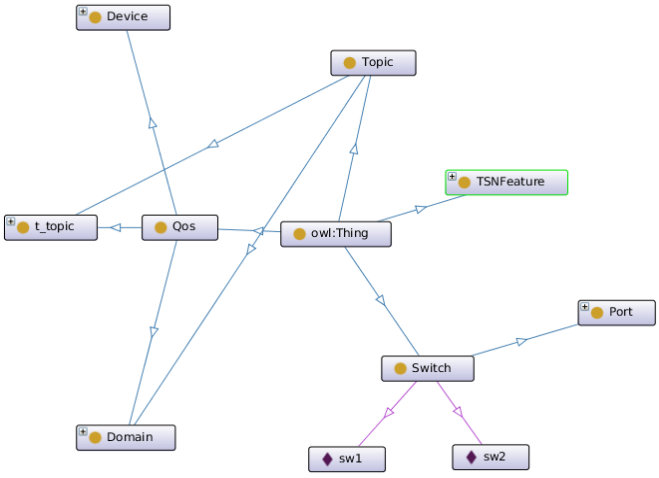

Fig. 6. An excerpt of the developed ontology. Device, Topic, QoS, Switches and etc. are defined as classes and subclasses and the relation between them such as *publishes* is expressed using object properties.

sumer are presented through object properties *publishes* and *consumes*. For example, the *rear_camera* device publishes the *t_rear_camera* topic which afterwards gets consumed by *dacam*.

This means that to fulfill network requirements, at the beginning, this information is required for configuration of e.g TSN switches. In the next steps, it should be clear which data properties have to be modified in the device e.g switch.

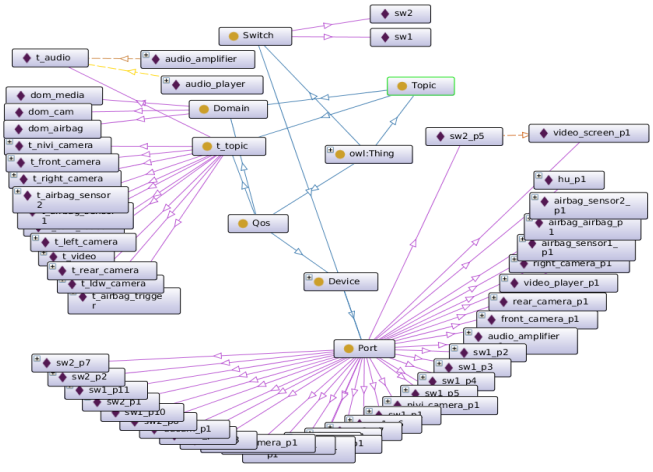

Fig. 7. The detailed relation between QoS requirements and network Topic beside the switches is an important knowledge that is required for automatic network configuration. An excerpt of the ontology modeling this knowledge is presented.

C. Modeling of Time-Aware Shaper

Time-Aware Shaper (TAS) applies a time-driven scheduling approach to manage the link access resulting in highly deterministic communication latency between network devices. Each port has 8 time-aware gates for 8 priority classes. Gate driver opens and closes the gates based on a predefined schedule (event list). As presented in **Figure 8**, for a given time, gates can be either closed (0) or opened (1). The important aspect here is that if a data has the highest priority (i.e. it is a TAS-scheduled data), its transmission must not be interrupted by a non-TAS data. In other words, when a gate is opened for TAS-scheduled data, all other gates are closed. In the given example, at T1 Gate 2 is open for high-priority data and other gates are closed.

To achieve the lowest latency, it is also very important that all TAS nodes in the network are synchronized very precisely, enough bandwidth is reserved and low-priority frames can be preempted. When these conditions are satisfied, a high-priority data can be transmitted with minimum latency through the network.

The mentioned knowledge about TAS (Gate driver, event list, gate status, etc.) are required for correct network configuration. To automate this process, this knowledge is also modeled in the ontology using data properties. **Figure 9** shows an excerpt of the modeled properties.

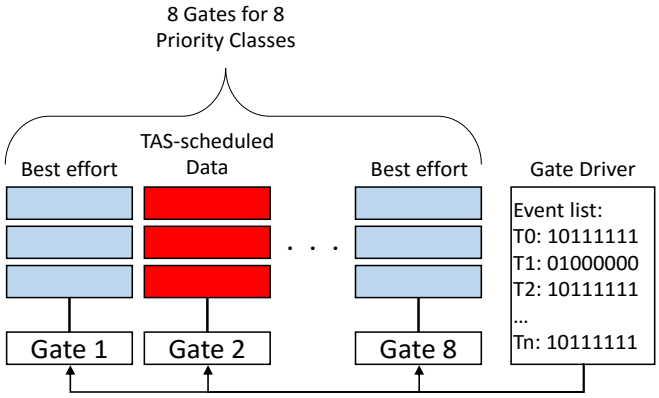

Fig. 8. TAS architecture contains 8 gates for 8 priority classes on each port. The Gate Driver decides which port has to be opened or closed based on a predefined event list. For example, at T1, only Gate 2 is open for transmission of high-priority data (Ethernet frames) without any interruptions by low-priority best effort data.

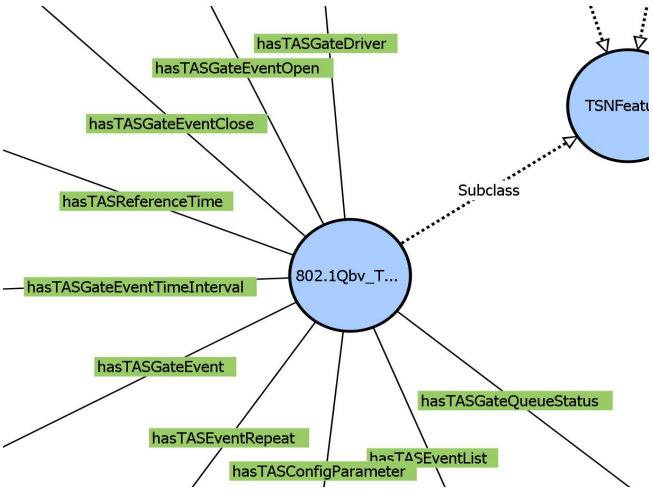

Fig. 9. Ontology data properties are used to model the properties of TAS. They build the most important knowledge for automatic configuration of TAS. An excerpt of the model is presented here.

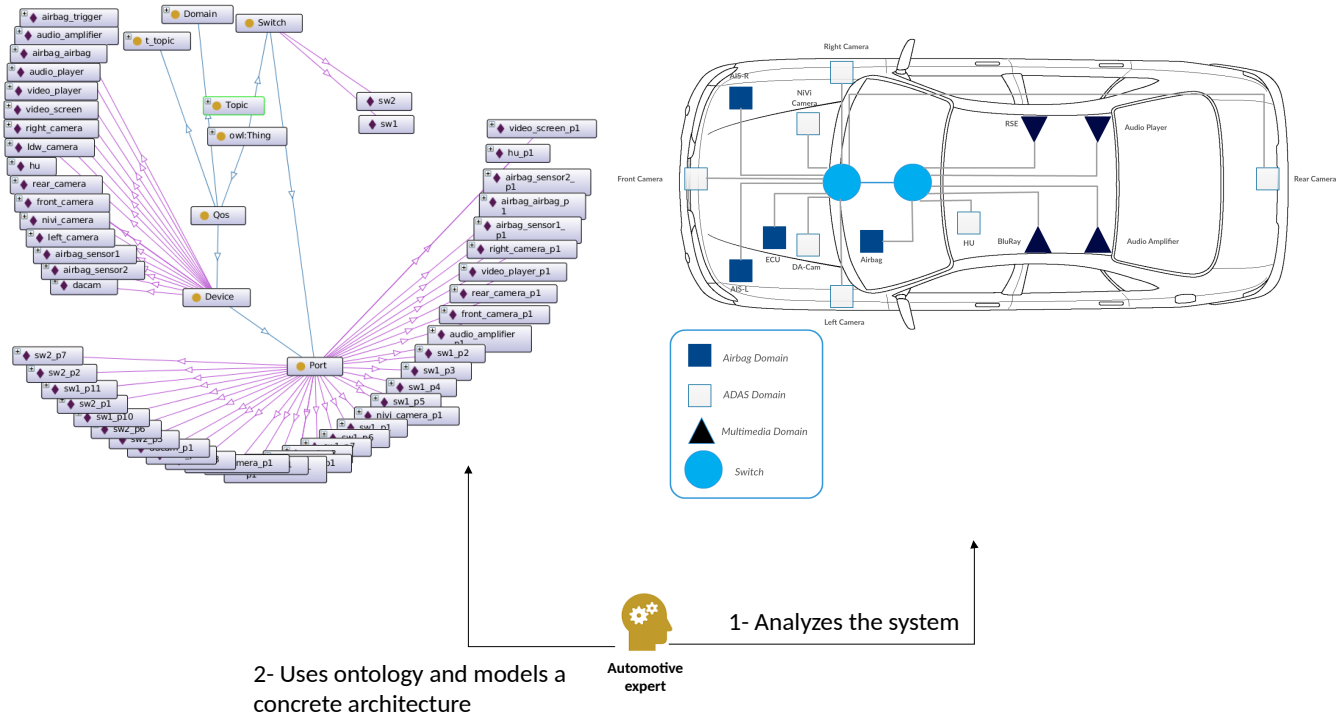

Fig. 10. This example shows how to use ontology classes and object properties to generate instances (individuals). An ADAS TSN network (right) introduced by [23] which is slightly changed with and airbag domain added into and analyzed using the ontology.

D. A simple modeling use case

Following the ontology-based approach, a concrete and simple TSN network is modeled using the ontology classes. There are two TSN switches, one with eleven and the other with seven Ethernet ports (e.g. *sw1_p1*). Two airbag sensors beside a sensor trigger ECU, are connected to switch1 (e.g. *airbag_sensor1*). Four IP cameras, one navigation camera and one Lane Departure Warning/Traffic Sign Recognition (LDW/TSR) camera are connected to the rest of the ports of the switch1 (e.g. *right_camera_p1* is connected to *sw1_p8*). On the other hand, one airbag device, one HU and four other multimedia devices, including a video player, video screen, audio player and audio amplifier are connected to switch2. The link in between switch1 and switch2 is considered to be symmetric. An excerpt of the model is presented in **Figure 10**.

V. Conclusion and Future Work

The advantages of Time-Sensitive Networking (TSN) and its challenges regarding network configuration overhead caused by Time-Aware Shaper are discussed. An ontology-based approach is presented to improve the support of automatic network configuration and Plug-and-Play capabilities in TSN.

Definition of ontology-based configuration rules are very important for the automatic configuration of TSN. In a future work, the expressiveness of ontology rules have to be investigated to decide if they alone are able to describe all required TSN configuration rules or they have to be extended by other tools. We also intend to transform Ontology models to Logic Programming models to increase the expressiveness. The engineer can model the network using the graph-based ontology and transform it automatically into e.g. Prolog facts and rules. These are used for configuration and verification use cases.

Acknowledgment

I would like to thank my student Sina Shafaei for his support during his master thesis.

References

[1] "Time-Sensitive Networking." [Online]. Available: http://www.ieee802.org/1/pages/tsn.html

[2] M. Hashemi Farzaneh, S. Nair, M. A. Nasseri, and A. Knoll, "Reducing Communication-related Complexity in Heterogeneous Networked Medical Systems Considering Non-functional Requirements," *16th International Conference on Advanced Communication Technology*, pp. 547–552, 2014.

[3] "Audio Video Bridging." [Online]. Available: http://www.ieee802.org/1/pages/avbridges.html

[4] G. Reinhart, S. Krug, S. Huettner, Z. Mari, F. Riedelbauch, and M. Schloegel, "Automatic configuration (Plug & Produce) of Industrial Ethernet networks," *9th IEEE/IAS International Conference on Industry Applications, INDUSCON*, 2010.

[5] L. Duerkop, H. Trsek, J. Jasperneite, and L. Wisniewski, "Towards autoconfiguration of industrial automation systems: A case study using Profinet IO," in *Proceedings of 17th International Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, sep 2012, pp. 1–8.

[6] F. Jammes, A. Mensch, and H. Smit, "Service-oriented device communications using the devices profile for web services," in *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*. ACM, 2005, pp. 1–8.

[7] L. Durkop, J. Imtiaz, H. Trsek, L. Wisniewski, and J. Jasperneite, "Using OPC-UA for the autoconfiguration of real-time Ethernet systems," in *International Conference on Industrial Informatics (INDIN)*. IEEE, jul 2013, pp. 248–253.

[8] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC unified architecture*. Springer Science & Business Media, 2009.

[9] L. Durkop, J. Jasperneite, and A. Fay, "An analysis of real-time ethernets with regard to their automatic configuration," in *IEEE World Conference on Factory Communication Systems (WFCS*. IEEE, 2015, pp. 1–8.

[10] A. Guerrero, V. A. Villagrá, J. E. L. De Vergara, and J. Berrocal, "Ontology-based integration of management behaviour and information definitions using swrl and owl," in *Ambient Networks*. Springer, 2005, pp. 12–23.

[11] S. Van der Meer, B. Jennings, D. O'Sullivan, D. Lewis, and N. Agoulmine, "Ontology based policy mobility for pervasive computing," 2005.

[12] J. Keeney, D. Lewis, D. O'Sullivan, A. Roelens, V. Wade, A. Boran, and R. Richardson, "Runtime semantic interoperability for gathering ontology-based network context," in *10th IEEE/IFIP Network Operations and Management Symposium (NOMS)*. IEEE, 2006, pp. 56–65.

[13] P. Ray, N. Parameswaran, J. Strassner, *et al.*, "Ontology mapping for the interoperability problem in network management," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 10, pp. 2058–2068, 2005.

[14] J. E. L. De Vergara, A. Guerrero, V. A. Villagrá, and J. Berrocal, "Ontology-based network management: study cases and lessons learned," *Journal of Network and Systems Management*, vol. 17, no. 3, pp. 234–254, 2009.

[15] A. K. Bandara, A. Kakas, E. C. Lupu, and A. Russo, "Using argumentation logic for firewall policy specification and analysis," in *Large Scale Management of Distributed Systems*. Springer, 2006, pp. 185–196.

[16] T. Klie, F. Gebhard, and S. Fischer, "Towards automatic composition of network management web services," in *10th IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2007, pp. 769–772.

[17] K. Apajalahti, E. Hyvönen, J. Niiranen, and V. Räisänen, "Stare: Statistical reasoning tool for 5g network management," *Submitted to ESWC*, 2016.

[18] A. Martinez, M. Yannuzzi, J. Lopez de Vergara, R. Serral-Gracia, and W. Ramirez, "An ontology-based information extraction system for bridging the configuration gap in hybrid sdn environments," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 441–449.

[19] G. Pardo-Castellote, "Omg data-distribution service: Architectural overview," in *Proceedings. 23rd International Conference on Distributed Computing Systems Workshops*. IEEE, 2003, pp. 200–206.

[20] C. Buckl, M. Geisinger, D. Gulati, F. J. Ruiz-Bertol, and A. Knoll, "Chromosome: a run-time environment for plug & play-capable embedded real-time systems," *ACM SIGBED Review*, vol. 11, no. 3, pp. 36–39, 2014.

[21] I. Schafer and M. Felser, "Topology discovery in profinet," in *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2007, pp. 704–707.

[22] M. Buechel, J. Frtunikj, K. Becker, S. Sommer, C. Buckl, M. Armbruster, A. Marek, A. Zirkler, C. Klein, and A. Knoll, "An automated electric vehicle prototype showing new trends in automotive architectures," in *18th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2015, pp. 1274–1279.

[23] G. Alderisi, G. Iannizzotto, and L. Lo Bello, "Towards ieee 802.1 ethernet avb for advanced driver assistance systems: a preliminary assessment," *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, p. 4, 2012.