Formally Verifiable Modeling of In-Vehicle Time-Sensitive Networks (TSN) Based on Logic Programming

Morteza Hashemi Farzaneh, Sina Shafaei and Alois Knoll
Robotics and Embedded Systems
Technische Universität München
Boltzmannstr. 3, 85748 Garching bei München
Email:{hashemif,shafaei,knoll}@in.tum.de

基于Prolog逻辑编程开发了一种用于配置和验证802.1Qbv的网络模型

***Abstract*—Increasing number of in-vehicle sensors, actuators and controllers involved in novel applications such as autonomous driving, requires new communication technologies to fulfill heterogeneous non-functional requirements such as latency, bandwidth and reliability. Time-Sensitive Networking (TSN) is a set of new standards in development by Institute of Electrical and Electronics Engineers (IEEE) defined to support mixed criticality based on Ethernet technology. This technology has recently raised significant attention of automotive domain. However, the mutual influence of application requirements in relation to TSN standards still remains a complex problem to master. For instance, considering an existing complex automotive network, an engineer has to carefully analyze the possible effects of adding new sensors on other existing critical applications. The network has to be configured such that the fulfilling of all requirements is verified. Targeting this problem, a modeling approach based on Logic Programming (LP) is developed to support more efficient configuration and verification process with focus on in-vehicle TSN networks.**

I. Introduction

Time-Sensitive Networking (TSN) [1] is a set of new standards which are in developing process by Institute of Electrical and Electronics Engineers (IEEE) to support mixed criticality based on Ethernet technology. These standards will help to overcome challenging requirements of 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.

Despite the strengths of TSN, increasing number of involved sensors, actuators and *Electronic Control Units (ECU)* and mutual influence of requirements still cause high network engineering overhead for automotive network engineers. For example, considering an existing in-vehicle network and its applications' requirements, reviewing the whole configuration is required when new critical nodes join the network. After reviewing the TSN standards, some parts of the network configuration may need to be modified and after this reconfiguration one has to formally verify that all requirements are satisfied.

In this paper, a modeling approach based on Logic Programming (LP) is developed for configuration and verification of TSN networks. The major advantage of our approach compared to the other modeling approaches is that the whole model consists of logical facts and rules. Using inference algorithms such as backward chaining (e.g. used in Prolog), the complexity of verification of application requirements is reduced to build the correct queries on the model in order to verify specific network properties including non-functional requirements of the applications. The modeling approach is also used to find out interesting correlations between different requirements which are important for configuration. The main **contributions** are: definition of the logical facts and rules (work in progress) in section III and modeling demonstration using a concrete example in section IV including use case examples for configuration and verification.

II. Related work

An abstract methodology for modeling and verification of Cyber-Physical Systems (CPS) is developed and demonstrated in [2], [3], [4] using logic programming. The focus is to build a bridge between logic programming and hybrid automata as the underlying model with infinite structures and properties. In contrast to these contributions, we additionally deal with mutual influence of different requirements and configuration aspects of In-vehicle TSN. Declarative networking is proposed in [5], based on logic programming language Datalog which is a subset of Prolog. This contribution is extended in [6] in order to apply it for declarative network verification of e.g. routing protocols without discussing the configuration aspects. A practical declarative network management approach based on Datalog is presented in [7]. This paper only describes how to specify QoS requirements such as latency, jitter and bandwidth but it does not explain how to use it for configuration and verification. It has been shown in [8] that Prolog is a programming language which is sufficiently expressive and well-suited for the implementation of distributed protocols. In this paper, Prolog is applied for implementation of the TSN modeling approach. Simulation-based approaches [9], [10], [11], [12] are developed to analyze the performance of Audio Video Bridging (AVB) and TSN. The main disadvantage of these approaches is that not all of the corner cases can be

978-1-5090-5197-7/16/$31.00 ©2016 IEEE

covered by simulations and therefore are not suitable for required formal verification of critical requirements. In contrast, formal analysis methods are developed in [13], [14] to verify the performance of the TSN shapers. These methods however, focus only on timing and latency aspects on the network layer and do not respond to the question of mutual influence of critical requirements.

A. Discussion

The majority of the logic programming-based approaches focus on specification and verification of network protocols. We exploit logic programming for step-by-step modeling TSN features.

III. Modeling Approach

Prolog (SWI implementation) is used as the logical modeling language. It is restricted to Horn clauses and consists of *facts* and *rules*. Each rule has the form $\alpha :- \beta_1, \beta_2, ..., \beta_n$ that is equivalent to $\beta_1 \wedge \beta_2 \wedge ...\beta_n \Rightarrow \alpha$. α is called *head* and $\beta_1, \beta_2, \ldots, \beta_n$ is the *body* of the rule. It is obvious that head is true if the body is true. Each β in the body is a call to a defined *predicate*. These predicates in the body are called *goals* and can be either a fact (a clause with empty body) or a rule.

Each PL-based TSN Model is a *knowledge base* that consists of Prolog facts and rules. Facts are used to describe properties and requirements in the network. The rules are used to describe the relations between facts and other rules. For instance,

```
publishes(front_camera,front_camera_p1,dom_cam
    [t_front_camera]).
```

is a fact describing that the device *front_camera* uses the Ethernet port *front_camera_pl* to send a data topic *t_front_camera* in the camera domain *dom_cam* and

```
firstPort(Dom,T,P) :- publishes(_,P,Dom,TL),
    member(T,TL).
```

is a rule that declares that *P* is the starting Ethernet egress port while transmitting data topic *T* in domain *Dom*, if ($\{\Leftarrow\} \equiv \{:-\}$) a device *publishes* a list of data topics *TL* in domain *Dom* using port *P* and ($and \equiv \wedge \equiv \{,\}$) also *T* is a *member* of *TL*.

In the following, facts and rules are presented that are developed to model a TSN network step-by-step.

A. Facts

The modeling clauses consist of the following facts: *topic*, *qos*, *publishes*, *consumes*, *device*, *switch* and *isLinked*.

Each data topic *T* is either a periodic topic or event-based topic. We define:

$$T_{topic}^{periodic} = (T_{domain}, T_{name}, T_{periodic}, T_{period}^{(\mu s)}, T_{size}^{(Byte)})$$
$$T_{topic}^{event} = (T_{domain}, T_{name}, T_{event}, T_{size}^{(Byte)}) \quad (1)$$

This classification of topics in periodic and event-based is significant regarding the hard real-time quality of service requirements. Data topics with tight timing requirements have to be modeled as periodic in order to calculate a feasible schedule with e.g. the time-aware shaper in IEEE 802.1Qbv. The combination of a topic domain and topic name is assumed to be unique in the whole TSN network. Consider all available topics: $T_{topics} = \{t_1, t_2, \ldots, t_n\}$ and all available domains as $T_{domains} = \{dom_1, dom_2, \ldots, dom_t\}$. It holds that

$$\forall_{T', T'' \in T_{topics}} (T'_{domain} = T''_{domain} \Rightarrow T'_{name} \neq T''_{name}) \quad (2)$$

The origin of the idea of using domains is in data-centric middleware approaches.

TSN nodes are either a *device* or a *switch* that consist of a set of Ethernet ports. Devices and switches use these ports to transmit or forward data topics embedded in Ethernet frames. Device and switch facts are defined as a tuple of name D_{name}, S_{name} and the list of their ports D^{ports}, S^{ports} as a subset of all existing ports in the network defined as $EP = \{p_1, p_1, \ldots, p_n\}$. Formally defined:

$$D_{device} = (D_{name}, D^{ports}), S_{switch} = (S_{name}, S^{ports}),$$
$$D_{devices} = \{D_1, D_2, \ldots, D_m\}, S_{devices} = \{S_1, S_2, \ldots, S_t\},$$
$$\forall D_i \forall S_j (D_i \in D_{devices} \wedge S_j \in S_{devices}),$$
$$\bigcap_{i=1}^{m} D_i^{ports} \bigcap_{j=1}^{t} S_j^{ports} = \emptyset \ ,$$
$$\bigcup_{i=1}^{m} D_i^{ports} \bigcup_{j=1}^{t} S_j^{ports} = EP \quad (3)$$

Each device can publish or consumes a set of topics in a specific domain. The facts are formally defined as:

$$P_{publishes}^{\{T_{topic}, D_{device}\}} = (D_{name}, D_{port}, T_{domain}, P_{topics}^{\{t_1, t_2, \ldots, t_u\}}),$$
$$C_{consumes}^{\{T_{topic}, D_{device}\}} = (D_{name}, D_{port}, T_{domain}, P_{topics}^{\{t_1, t_2, \ldots, t_v\}}),$$
$$\{t_1, t_2, \ldots, t_u\} \subseteq T_{topics}, \{t_1, t_2, \ldots, t_v\} \subseteq T_{topics} \quad (4)$$

The quality of service requirements of a topic among publisher and consumer devices are modeled using the fact *qos* which is defined as:

$$Q_{qos}^{\{T_{topic}, D_{device}\}} = (Q_{type}^{\{latency, reliability, \ldots\}}, T_{domain}, T_{name},$$
$$D_{publisher}, D_{consumer}, Q_{VLAN}^{\{0,1,\ldots,7\}}) \quad (5)$$

Network connections are defined using *isLinked* fact which has to be symmetric in order to describe bi-directional communication. Formally:

$$L_{isLinked}^{\{S_{switch}, D_{device}\}} = (L'_{port}, L''_{port}, L_{bandwidth}^{(\frac{Byte}{s})}),$$
$$\forall L'_{port} \ \forall L''_{port} (\exists D_{device} (L'_{port} \in D^{ports}) \vee$$
$$\exists S_{switch} (L''_{port} \in S^{ports})) \quad (6)$$

B. Rules (WORK IN PROGRESS)

The modeling inference rules are developed step-by-step in order to obtain useful information based on the defined facts. Such information is used for simplification of network verification and configuration. To make the *isLinked* fact symmetric, *link* is defined as:

```
link(P1,P2,BW):- isLinked(P1,P2,BW).
link(P1,P2,BW):- isLinked(P2,P1,BW).
```

To find the involved devices, switches and ports involved in transporting a topic, *path* is defined as :

```
path(Start,End,Path) :- traveling(Start,End,[
    Start],Temp), reverse(Temp,Path).
```

where *traveling* and *reverse* are helper predicates. To find the first and the last port for transmission of a topic, following rules are defined:

```
firstPort(Dom,T,P) :- publishes(_,P,Dom,TL),
    member(T,TL).
lastPort(Dom,T,P) :- consumes(_,P,Dom,TL),
    member(T,TL).
```

These rules are used as goals in *streamPorts* that find all related ports for transmission a topic and classifies ingress and egress ports. For TSN shapers (specially time-aware shaper), the egress ports play a significant role. For example, the most latency-related part of time-aware shaper in IEEE 802.1Qbv standard, depends on adequate schedule of gate drivers of the egress ports. Hence, it is significant to know how many frames with which priority are queued in such a port. Before a device starts to publish a new topic to a consumer, it is important to check and verify the current status of all affected egress ports on the transmission path. The definition of *streamPorts* is:

```
streamPorts(Dom,T,CL2,EP,IP):- firstPort(Dom,T
    ,FirstP),lastPort(Dom,T,LastP), path(
    FirstP,LastP,PL),removeSwitchDevice(PL,CL1
    ),removeDevice(CL1,CL2),portClassifier(CL2
    ,IP,EP).
```

where *removeSwitchDevice*, *removeDevice* and *portClassifier* are helper predicates.

IV. Modeling Example

For the purpose of demonstration, an Advanced Driver Assistance System (ADAS) example scenario [15] is used. It is extended in order to cover the tight real-time requirements for critical applications such as airbag. The extension includes two sensors which are responsible for collecting collision information and an ECU, responsible for processing those data. The last node is an airbag trigger which will act according to the messages it gets from ECU. An excerpt of the LP-based model focusing on airbag domain is explained in this paper.

Figure 1 depicts an excerpt of the related part of the example model. The first *topic* fact is a predicate, describing that *t_airbag_sensor1* is a *periodic* topic and belongs to *dom_airbag* domain with a period value of 250 microseconds and a size of 200 bytes. The first *qos* fact indicates that there is *latency* requirement on topic *t_airbag_sensor1* which is being published by device *airbag_sensor1* and is being consumed by *airbag_trigger* and it has the highest priority indicated by vlan value of *3*. The maximum allowed latency for this topic is initially its period value, described using the topic predicate as mentioned before. The second *publishes* and *consumes* facts describe that *airbag_trigger* ECU device, uses port *airbag_trigger_p1* to publish *t_airbag_trigger* topic which is being consumed by *airbag_airbag* device on the port *airbag_airbag_p1*. The predicate *switch* describes that *sw2* has seven Ethernet ports $[sw2\_p1, sw2\_p2, \ldots, sw2\_p7]$. Similar to switch predicate, the first *device* fact of our modeling example indicates that *airbag_sensor_1* uses *airbag_sensor_p1* for communication. To declare that there is a Gigabyte link between switch port *sw1_p2* and device port *airbag_trigger_p1*, the *isLinked* fact is used.

A. Use case: Configuration

The developed LP-based model is a knowledge basis which helps to reduce configuration effort. The engineer formulates a set of queries and gets response back based on inference mechanisms of Prolog. For instance, a new device *airbag_sensor2* has to be integrated into the network considering the airbag application. Because of the highest crticality level, its data has to be scheduled using time-aware shaper of TSN. The engineer has to find out which egress ports are on the path of the *airbag_sensor2* to its consumer. Using the following query all egress ports can be found on the path between publisher and consumer:

```
?-streamPorts(dom_airbag,t_airbag_sensor2,_,
    EgressPorts,_).
```

Prolog responds with:

```
EgressPorts=[airbag_sensor2_p1,sw1_p2].
```

These ports have to be considered when updating the time-aware shaper' gate driver. Another interesting question but more complicated one is: which data topics have the highest latency priority and go through switch *sw1* and moreover which period value do they have? The appropriate Prolog query is:

```
?-topic(Dom,T,periodic,PeriodVal,_),qos(
    latency,Dom,T,_,_,3),publishes(_,P,Dom,LT)
    ,member(T,LT),link(P,SP,_),switch(sw1,SPL)
    ,member(SP,SPL).
```

Prolog finds three topics, one after another:

```
Dom = dom_airbag,T = t_airbag_sensor1,
    PeriodVal = 250,P = airbag_sensor1_p1,LT=
    [t_airbag_sensor1],SP = sw1_p9, ...
```

B. Use case: Verification

Similar to configuration use case, verification of the network properties is done by adequate queries. Considering that airbag application is very critical the network engineer can verify that there are at least two disjoint paths between *airbag_trigger* ECU and the *airbag_airbag* actuator. The query is formulated as:

```
?-findall(PL,path(airbag_trigger_p1,
    airbag_airbag_p1,PL),Z), length(Z,N), N
    >=2.
```

Prolog responds with:

```
Z = [[airbag_trigger_p1, sw1_p2, sw1, sw1_p1,
    sw2_p1, sw2, sw2_p7, airbag_airbag_p1], [
    airbag_trigger_p1, sw1_p2, sw1, sw1_p12,
    sw2_p8, sw2, sw2_p7|...]],N = 2.
```

```
topic(dom_airbag,t_airbag_sensor1,periodic,250,200).
topic(dom_airbag,t_airbag_trigger,periodic,500,100).
topic(dom_airbag,t_airbag_sensor2,periodic,250,200).

qos(latency,dom_airbag,t_airbag_sensor1,airbag_sensor1,airbag_trigger,3).
qos(latency,dom_airbag,t_airbag_trigger,airbag_trigger,airbag_airbag,3).
qos(latency,dom_airbag,t_airbag_sensor2,airbag_sensor1,airbag_trigger,3).

publishes(airbag_sensor1,airbag_sensor1_p1,dom_airbag,[t_airbag_sensor1]).
publishes(airbag_trigger,airbag_trigger_p1,dom_airbag,[t_airbag_trigger]).
publishes(airbag_sensor2,airbag_sensor2_p1,dom_airbag,[t_airbag_sensor2]).

consumes(airbag_trigger,airbag_trigger_p1,dom_airbag,[t_airbag_sensor1]).
consumes(airbag_airbag,airbag_airbag_p1,dom_airbag,[t_airbag_trigger]).
consumes(airbag_trigger,airbag_trigger_p1,dom_airbag,[t_airbag_sensor2]).

switch(sw2,[sw2_p1,sw2_p2,sw2_p3,sw2_p4,sw2_p5,sw2_p6,sw2_p7]).

device(airbag_sensor1,[airbag_sensor1_p1]).
device(airbag_sensor2,[airbag_sensor2_p1]).
device(airbag_trigger,[airbag_trigger_p1]).

isLinked(sw1_p2,airbag_trigger_p1,1000000000).
isLinked(sw1_p4,airbag_sensor2_p1,1000000000).
isLinked(sw1_p9,airbag_sensor1_p1,1000000000).
isLinked(sw2_p7,airbag_airbag_p1,1000000000).
```

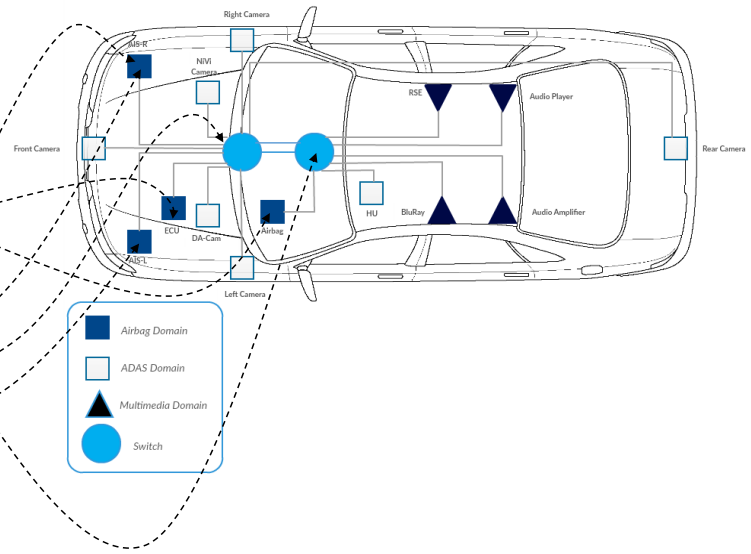

Fig. 1. An excerpt of the LP-based model of in-vehicle network using the formally defined facts. Using inference rules, interesting properties of the network can be verified.

It would return *false* if the number of disjoint paths is increased to $N >= 3$.

The second example deals with the importance of egress port *sw2_p7* for airbag trigger. It has to be verified that there is no data transmission of priority $\{vlan = 3\}$. The verification query in Prolog is:

```
?- streamPorts(_,T,_,EgressPorts,_),member(
    sw2_p7,EgressPorts),qos(_,_,T,_,_,3).
```

Prolog responds with *false* which means that there is definitely no disruptive communication on this port.

V. Conclusion and Future Work

A network modeling approach based on logic programming is presented that afterwards is used to assist automotive network engineers, during the configuration and verification process. The uniqueness of this approach is that the whole network model is based on logical facts and rules which leads to more efficient configuration and verification process that normally costs a lot of engineering effort. This work is still in progress and we are extending the rules to model deep details of TSN with application in automotive domain.

References

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

[2] G. Gupta and E. Pontelli, "A constraint-based approach for specification and verification of real-time systems," in *Proceedings Real-Time Systems Symposium*, Dec 1997, pp. 230–239.

[3] N. Saeedloei and G. Gupta, "A logic-based modeling and verification of cps," *SIGBED Rev.*, vol. 8, no. 2, pp. 31–34, Jun. 2011. [Online]. Available: http://doi.acm.org/10.1145/2000367.2000374

[4] ——, "A methodology for modeling and verification of cyber-physical systems based on logic programming," *SIGBED Rev.*, vol. 13, no. 2, pp. 34–42, Apr. 2016. [Online]. Available: http://doi.acm.org/10.1145/2930957.2930963

[5] B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica, "Declarative networking," *Commun. ACM*, vol. 52, no. 11, pp. 87–95, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1592761.1592785

[6] A. Wang, P. Basu, B. T. Loo, and O. Sokolsky, "Declarative network verification," in *International Symposium on Practical Aspects of Declarative Languages*. Springer, 2009, pp. 61–75.

[7] T. L. Hinrichs, N. S. Gude, M. Casado, J. C. Mitchell, and S. Shenker, "Practical declarative network management," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, ser. WREN '09. New York, NY, USA: ACM, 2009, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/1592681.1592683

[8] N. P. Lopes, J. A. Navarro, A. Rybalchenko, and A. Singh, "Applying prolog to develop distributed systems," *Theory and Practice of Logic Programming*, vol. 10, no. 4-6, pp. 691–707, 2010.

[9] H. T. Lim, D. Herrscher, and F. Chaari, "Performance comparison of ieee 802.1q and ieee 802.1 avb in an ethernet-based in-vehicle network," in *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*, vol. 1, April 2012, pp. 1–6.

[10] G. Alderisi, A. Caltabiano, G. Vasta, G. Iannizzotto, T. Steinbach, and L. L. Bello, "Simulative assessments of ieee 802.1 ethernet avb and time-triggered ethernet for advanced driver assistance systems and in-car infotainment," in *Vehicular Networking Conference (VNC), 2012 IEEE*, Nov 2012, pp. 187–194.

[11] P. Meyer, T. Steinbach, F. Korf, and T. C. Schmidt, "Extending ieee 802.1 avb with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic," in *2013 IEEE Vehicular Networking Conference*, Dec 2013, pp. 47–54.

[12] T. Steinbach, H. T. Lim, F. Korf, T. C. Schmidt, D. Herrscher, and A. Wolisz, "Beware of the hidden! how cross-traffic affects quality assurances of competing real-time ethernet standards for in-car communication," in *Local Computer Networks (LCN), 2015 IEEE 40th Conference on*, Oct 2015, pp. 1–9.

[13] S. Thangamuthu, N. Concer, P. J. L. Cuijpers, and J. J. Lukkien, "Analysis of ethernet-switch traffic shapers for in-vehicle networking applications," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2015, pp. 55–60.

[14] D. Thiele, R. Ernst, and J. Diemer, "Formal worst-case timing analysis of ethernet tsn's time-aware and peristaltic shapers," in *Vehicular Networking Conference (VNC), 2015 IEEE*, Dec 2015, pp. 251–258.

[15] 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.