11th IEEE World Conference on Factory Communication Systems (WFCS), Palma de Mallorca - May 2015
Recipient of "Best Work-in-Progress Paper" award

A Configuration Agent based on the Time-Triggered Paradigm for Real-Time Networks

Marina Gutiérrez Wilfried Steiner
TTTech Computertechnik AG
{marina.gutierrez, wilfried.steiner}@tttech.com

Radu Dobrin Sasikumar Punnekkat
Mälardalen University
{radu.dobrin, sasikumar.punnekkat}@mdh.se

***Abstract*—Distributed cyber-physical systems are growing in size and functionality and deterministic communication is an important requirement for those systems. The existing solutions based on the time-triggered paradigm pose certain limitations in regards to the configuration. Usually configuration is seen as a one-time event during the installation of the network. Future real-time networks need to be able to adapt more easily to changes in the network. Thus, the configuration becomes an ongoing service, e.g., as for network maintenance and re-configuration to add and remove new, respectively old, equipment.**

We postulate that configuration will emerge to a continued service that accompanies a real-time network throughout its different life-cycle phases. In this context of evolving and dynamic networks, we introduce the concept of a *configuration agent* for real-time networks and demonstrate the concept by a realization based on the time triggered paradigm.

I. Introduction

The time-triggered paradigm has been used to provide deterministic communication in systems with high criticality requirements [1][2]. It is based on the idea of planning the moments in time in which messages should be sent. This plan is called the time-triggered schedule for the network.

Obtaining a time-triggered schedule for a network is an NP-hard problem [3]. Typical solutions are discussed in [4] using the YICES SMT Solver [5] and also in [6] using a Tabu Search based technique. However in both methods the schedule is obtained offline and with a major requirement: that all the necessary information from the network is known a priori.

With the current industrial trends in cyber-physical systems, like the Real-Time Internet-of-Things (RT-IoT), that require significantly more capable networks, this approach is no longer valid. These future networks will need to accommodate a much higher number of communication participants, higher communication speeds, and more deterministic end-to-end communication latency and jitter than today's networks.

Furthermore, these networks need to be flexible and network changes will be the rule, not the exception. Especially, in the context of Systems-of-Systems (SoS) there might never be a final phase as the system may continuously evolve and vary its purpose. Therefore, industry is looking into adoption of well-established IT networking technologies like Ethernet and IP networks for distributed cyber-physical systems (e.g., IEEE 802.1 TSN [7] or IETF Deterministic Networking). One of the key challenges in such future networks is configuration and the IEEE and IETF provide protocols for distributed (plug-and-play like) configuration as well as centralized configuration options.

In this context and orthogonal to the well-established configuration possibilities of traditional cyber-physical systems (in critical systems this is often done manually per-device) and the upcoming IEEE/IETF possibilities, this paper looks into the concept of a "configuration agent".

The purpose of this entity is to produce a schedule that guarantees the temporal properties of the messages in a network in which not all the information is available beforehand. For that the configuration agent 1) continuously monitors ongoing traffic in the network, 2) distills traffic parameters from the measured data, and 3) aims to update the configuration of the network to optimize it to specified quality aspects. The configuration agent acts on behalf of the system manager and ultimately as an autonomous mechanism to update and configure the network, thus making the network more capable to adapt to changes.

There are many use cases for configuration agents like that, for example, new network hardware may be installed and the configuration agent (being informed of this update) unlocks new features by re-configuring the network. Another example is a re-prioritization of traffic in cases when new critical traffic enters the network. The commonality in the use cases is that the configuration agent has incomplete knowledge of the network and therefore implements procedures to acquire missing information for optimal configuration.

We continue in Section II with the system model. The details about the configuration agent are explained in Section III. Finally in Section IV conclusions and future work can be found.

II. System Model

The proposed method to obtain a schedule for a previously unsynchronized real-time network would be applied to an existing network. This network is composed by a multitude of nodes, N and switches, S. The nodes execute applications that send messages, F, through the network.

Formally the physical topology of a network is defined by a undirected graph $G(V, E)$. Nodes and switches are vertices v_i such that $\forall v_i \in V$ and $N \cup S = V$. The physical communication links connecting vertices v_i and v_j are edges, so that $(v_i, v_j) \in E$. Fig. 1 depicts an example of a network topology with nine vertices: six nodes and three switches.

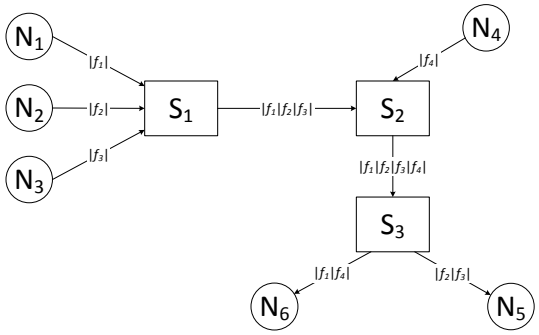

Fig. 1. Network composed by six nodes N_1 to N_6 and three switches, S_1, S_2 and S_3. Nodes N_1, N_2, N_3 and N_4 send messages f_1, f_2, f_3 and f_4 respectively.

The traffic that is sent through the network is composed by messages of different types (periodic, sporadic, bursty, etc) and different criticalities. For this paper we are going to focus just on periodic messages. A message can be characterized by their period, length and priority:

$$\forall f_i \in F : f_i = \{period_i, length_i, priority_i\} \quad (1)$$

Being all messages periodic, the periodic repetitions of the same message are grouped in $flows$. We call $flow_i$ to the set of messages f_i that are sent through the network during a time span L, so that:

$$flow_i = \{f_i(t_1), f_i(t_2), ..., f_i(t_n)\} \quad (2)$$

where $t_1, ..., t_n$ are moments in time that satisfy $\forall t_i, t_j : j > i \Rightarrow t_j > t_i \wedge t_j - t_i < L$.

The traffic in the network is defined by the flows of messages. Since the flows follow different paths through the network, not all the flows / messages are going to be visible in any point in the network. In a given switch s_j we define M_{s_j} as the set of messages that goes through the switch, then the traffic pattern, TP_{s_j} observed in the switch is the set of observed flows:

$$TP_{s_j} = \{flow_i | f_i \in M_{s_j}\} \quad (3)$$

To further describe our system model we have made the following assumptions:

1) The nodes execute applications, which define mandatory requirements and optional desirements on the network.
2) The network is configured in a way to meet all the mandatory requirements (e.g., maximum transmission latency and jitter) and may also satisfy optional desirements (e.g., less latency and jitter).
3) The change in the order of the messages within the flow is not allowed in accordance with the IEEE 802.1 Q [8].

III. Configuration Agent : Three Step Approach

The configuration agent (depicted in Fig. 2) is composed of three elements: the Monitor, the Extractor and the Scheduler. The Monitor observes the network's behavior in order to identify traffic patterns. The Extractor generates some traffic parameters based on the traffic patterns observed by the Monitor and some previous knowledge of the network and applications. Finally, the Scheduler uses the traffic parameters obtained by the Extractor to generate a schedule for the network. This schedule should guarantee that the set of satisfied desirements increases as well as to improve the network performance properties.

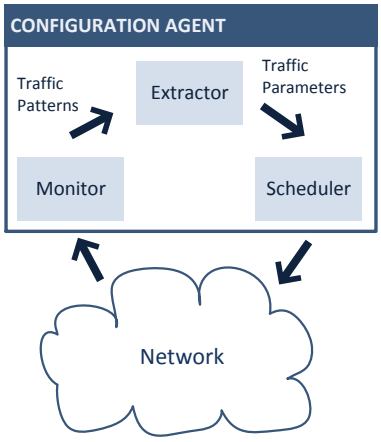

Fig. 2. Configuration Agent Overview

This approach sees the configuration as a potentially endless process. Therefore, after the network has been configured according to the new schedule, the Monitor will start again to gather information from the network because the traffic parameters might have change or will change again.

A. The Monitor : Observing Traffic Patterns

As it is described in the previous section the Monitor observes the traffic in the network to identify the traffic patterns (3). To characterize the traffic patterns quantitatively the monitor needs to gather messages / flows IDs, messages lengths and the points in time in which the messages are received.

Messages / flows IDs and messages lengths are message parameters that doesn't change over time or from a vertex in the network to another. The arrival times of the messages, on the other hand, are different in every vertex in the network. Thus, those measurements always have to be referred to a specific switch in the network and can be expressed as follows:

$$T_{s_k}(n,m) = \begin{pmatrix} t_{f_1^1} & t_{f_1^2} & \cdots & t_{f_1^n} \\ t_{f_2^1} & t_{f_2^2} & \cdots & t_{f_2^n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{f_m^1} & t_{f_m^2} & \cdots & t_{f_m^n} \end{pmatrix} \quad (4)$$

Here f_i^j stands for the j-th message of $flow_i$ and $t_{f_i^j}$ for the point in time in which message f_i^j arrives to switch s_k. Therefore, m is the number of different flows that goes through the switch s_k and n is the number of messages of each flow. If

the measurements are taken during a fixed time span, L then $n = \lfloor L/period \rfloor$, i.e., if periods are different then the matrix (4) will be an "incomplete" matrix, in which not all entries are filled.

B. The Extractor : Obtaining Traffic Parameters

The Extractor obtains relevant traffic parameters from the traffic patterns observed by the Monitor. These traffic parameters are the periods of the messages, their priorities and possible precedence relationships between the messages, such as sensor / actuator kind of communication types.

The process presented in this paper to distill these information consists of three steps. In every step of the process we use more information from the traffic patterns gathered by the Monitor to refine the results.

For the first step we use only the times in which a given message, f_i arrives at the switch $T_{s_k}(n,i) = \{t_{f_i^1}, t_{f_i^2}, ...t_{f_i^n}\}$. With those times we obtain a first value for the period of the message:

$$period_i = \frac{\sum_{j=1}^{n}(t_{f_i^{j+1}} - t_{f_i^j})}{n-1} = \frac{t_{f_i^n} - t_{f_i^1}}{n-1} \tag{5}$$

Although in this paper we are focusing in periodic messages, the inter-arrival times of a messages from the same flow to a given point in the network are not always constantly equal to their period. Sometimes a message from a flow, f_i, has to compete with a message from another flow, f_k, for the use of the network. In these cases the switch has to decide which message is going to be sent first. Thus, if message f_k is sent first, message f_i it is going to be delayed for a certain amount of time. From now on, we refer to this effect as an interference, I, caused by message f_k on message f_i.

To identify these interferences we use $T_{s_k}(n,i)$ again to calculate the inter-arrival times between consecutive messages of the same the flow:

$$\Delta t_i^j = t_{f_i^{j+1}} - t_{f_i^j} \tag{6}$$

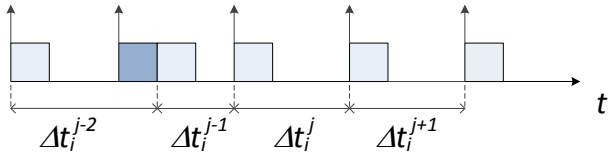

Fig. 3. Interference suffered by message f_i^{j-1}

In Fig. 3 we can see how an interference modifies the inter-arrival times. For each interference we are going to observe a Δt_i longer than the period, Δt_i^{j-2} in the figure, and another shorter than the period, Δt_i^{j-1}. So we can see every $\Delta t_i = period_i \pm C$ where C is the duration of interference. For messages that have not suffered interferences $\Delta t_i = period_i$.

Being able to identify and classify the interferences that a message suffers will help us to detect: 1) whether a message suffers interferences and 2) the duration of those interferences.

We can even go an step further and extract more information from the interferences. Fig. 3 depicts the simplest interference possible: message f_i^{j-1} suffers just one interference from another message. In a general situation a message can suffer interferences caused by different messages. We can classify the interferences into two types as it can be seen in Fig. 4:

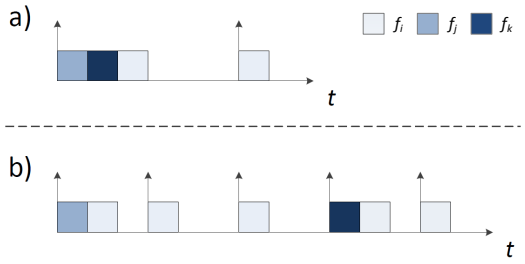

Fig. 4. a) Message f_i suffers an interference because two messages, f_j and f_k, are sent first. b) Message f_i suffers two interferences of the same duration caused by two different messages, f_j and f_k.

a) Message f_i has to wait because several messages are sent first. The duration of the interference would be the sum of all the lengths of the messages that cause the interference and it will occur with a period equal to the least common multiple of the original periods of the messages involved.
b) Message f_i suffers several interferences of the same duration caused by different messages / set of messages.

It is clear now that the total amount of interferences that a message can potentially suffer depends on the number of messages, the length of the messages and the amount of messages of each length. For large networks the exact number is not trivial to obtain and it can also be unrealistically pessimistic. Observing the Δt_i^j as obtained in (6) we can find a more realistic bound for this value. Equation (7) gives the duration of the longest measured interference, I_i^{max}, that affects messages from $flow_i$.

$$I_i^{max} = \frac{1}{2}(max\Delta t_i^j - min\Delta t_i^j) \tag{7}$$

Knowing this value we can discard all the interferences longer than I_i^{max} and therefore discard also the periodic events (other messages in the network) that cause them as a possible source of interference.

The purpose of this last step is to be able to identify the periodic event that is causing each interference which, in time, will help us to determine the priority relationship between messages.

Our expectation is to be able to use the ideas explained before and cross comparing the data from all the messages to learn the necessary parameters that the Scheduler needs to be able to produce a schedule for the network.

C. The Scheduler: Producing an Schedule

Once the Extractor distilled the traffic parameters from the measurements as provided by the Monitor, the Scheduler is in charge to take the traffic parameters as input to generate a

traffic schedule. In order to do so, the Scheduler also needs information about the network topology and the communication path of the messages to be scheduled. There are well-known protocols that map out the topology of a given network, as for example Shortest-Path Bridging (IEEE 802.1aq). Likewise, in an Ethernet network the communication path of a message can be derived from the network topology (as explored by IEEE 802.1Q) and the source and destination address given in the Ethernet header. Alternatively, the (distributed) Monitor may also implement additional functionality to measure the path of messages throughout the network.

An example of the communication schedule of the four messages communicated in network of Fig. 1 is given in Fig. 5. In particular, the figure depicts the points in time of the message transfers as black dots.

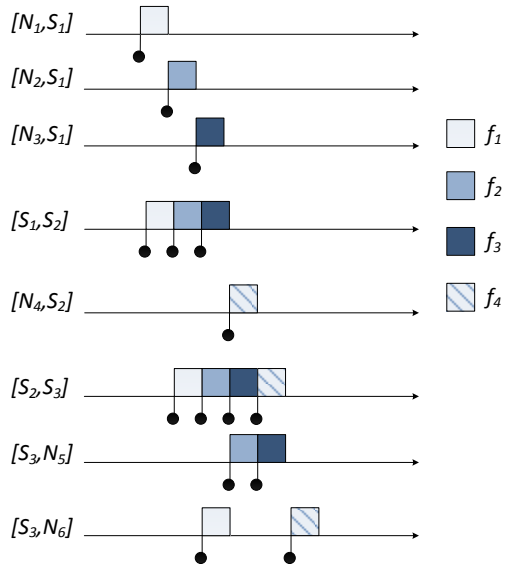

Fig. 5. Example time-triggerd communication schedule for the system of four messages as communicated in the network of Fig. 1

The requirements that such communication schedule needs to meet can be described in terms of constraints [4]. In particular, for time-triggered communication, two core sets of constraints are essential: collision-freedom [9] and maximum end-to-end latency.

Time-triggered scheduling, of course, implies that the end nodes and switches in the system are capable to synchronize their local clocks to each other and are able to execute the communication schedule as produced by the scheduler. However, in legacy systems only some components may have this capabilities built-in. Also in these cases, when only parts of the network are synchronized to each other, at least within these synchronized subsystems the time-triggered paradigm can be applied. Thus, critical messages can exhibit preferred treatment within these subnetworks by transmitting them in accordance with a time-triggered schedule. A schedule in such a case typically foresees several candidate slots per message from which the message will dynamically pick one, thus avoiding to delay a respective message for a whole communication cycle, if the message just arrives after a slot passed.

Once the schedule for the network has been generated, the last step would be the reconfiguration of the network (or subnetwork) according to the schedule. To do so, there are IEEE/IETF configuration standards that can be used. Other option would be for the Scheduler to just produce a report with the configuration agent recommendations for the network, and leave it up to the human manager to decide what to do.

IV. Conclusions and Future Work

In this paper we introduce the concept of a *configuration agent* in order to ease the configuration of real-time / time-triggered networks. Our approach first extracts relevant traffic parameters through the observation of the network's traffic patterns. Then, it creates a time-triggered schedule that guarantees the message transmissions with improved temporal properties, such as latency and jitter, enables determinism and flexibility, as well as satisfies the messages' original timing constraints.

Ongoing efforts primarily focus on testing the feasibility of the approach presented in this paper through network simulations both on small and large scale using OMNeT++ [10]. The results obtained will provide more information that will help us for the future implementation of the configuration agent. Future work will also address the inclusion of other types of traffic (sporadic, bursty...) or other changes in the assumptions made about the system.

Acknowledgment

The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement 607727.

References

[1] S. Poledna, H. Kopetz, and W. Steiner, "Deterministic System Design with Time-Triggered Technology," in *Microelectronics System Symposium 2014 (MESS14)*, 2014.

[2] W. Steiner, G. Bauer, B. Hall, and M. Paulitsch, "Time-Triggered Ethernet," in *Time-Triggered Communication*, R. Obermaisser, Ed. CRC Press, 2011.

[3] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, Jan 2003.

[4] W. Steiner, "An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks," in *Real-Time Systems Symposium (RTSS), 2010 IEEE 31st*. IEEE, 2010, pp. 375–384.

[5] B. Dutertre and L. De Moura, "The Yices SMT solver," *Tool paper at http://yices. csl. sri. com/tool-paper. pdf*, vol. 2, p. 2, 2006.

[6] D. Tamas-Selicean, P. Pop, and W. Steiner, "Synthesis of communication schedules for TTEthernet-based mixed-criticality systems," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. ACM, 2012, pp. 473–482.

[7] "IEEE 802.1 - Time Sensitive Networking Task Group," http://www.ieee802.org/1/pages/tsn.html, 3 November 2014.

[8] "IEEE 802.1Q - Virtual LANs," http://www.ieee802.org/1/pages/802.1Q.html, 13 March 2015.

[9] A. K. Mok and W. Wang, "Window-constrained real-time periodic task scheduling," in *RTSS '01: Proceedings of the 22nd IEEE Real-Time Systems Symposium*. Washington, DC, USA: IEEE Computer Society, 2001, p. 15.

[10] "OMNeT++," http://www.omnetpp.org/, 21 January 2015.