Analysis of Deterministic Ethernet Scheduling for the Industrial Internet of Things

Ramon Serna Oliver Silviu S. Craciunas Georg Stöger
TTTech Computertechnik AG, Vienna, Austria
{rse,scr,gst}@tttech.com

***Abstract*—Scheduling and guaranteeing strict timeliness for real-time communication across large networks, like those envisioned in the Industrial Internet of Things (IoT), in coexistence with non-critical traffic, opens up a remarkable opportunity of application for Deterministic Ethernet. However, the inherently dynamic behavior and flexibility required in IoT systems contrast with the typically static design of scheduled time-triggered networks. This paper explores the feasibility of online incremental synthesis of network schedules that adapt to such dynamic behavior without losing the strict timeliness guarantees. We investigate the computational complexity of the incremental scheduling problem and profile performance metrics enabling the evaluation of potential trade-offs setting the boundaries of online –close to real-time– time-triggered scheduling for the Industrial Internet of Things.**

I. Introduction

The Internet of Things (IoT) [1], particularly in the industrial domain, is a prominent area of application for Deterministic Ethernet. In this paper, we present our findings evaluating the feasibility of scheduled networks for industrial control systems in the context of IoT. Our analysis explores the challenges synthesizing online incremental time-triggered schedules for industrial-sized network topologies and loads, extracting a number of key performance metrics and their impact on the overall schedule efficiency. Our goal is to investigate the problem complexity with sufficient detail to discern the boundaries of online –close to real-time– scheduling.

Deterministic Ethernet is an IEEE 802.1 compliant extension to Quality of Service (QoS), using scheduled message release mechanisms based on a global time instead of dynamic priority schemes. In a Deterministic Ethernet network, regular unconstrained best-effort Ethernet traffic can co-exist with real-time critical traffic flows without altering the guaranteed and deterministic strict delivery timeliness of scheduled traffic flows. Mixed-criticality requirements from classic industrial deployments (e.g. factory floor) can be fulfilled and guaranteed with the support of deterministic scheduled networks and a carefully build distributed communication scheme. However, the rapid conceptual expansion of the IoT introduces a new range of dynamic applications which cannot be handled by a single, static communication scheme. In the context of IoT, adaptability, scalability, and self-configuration are key to success. However, deterministic guarantees are still needed in order to ensure the timely execution of critical applications.

The research leading to these results has received funding from the ARTEMIS JU under grant agreement n° 621429 (Project EMC2) and from the Austrian Ministry for Transport, Innovation and Technology under the funding programme "IKT der Zukunft", grant agreement n° 842567.

The IEEE Time-Sensitive Networking[1] (TSN) Task Group is currently defining key mechanisms for Deterministic Ethernet; once this technology becomes commercially available, the challenge for efficiently using it lies in creating and distributing appropriate message transmission schedules for the network nodes. However, the synthesis of schedules for time-triggered distributed networked systems (e.g. [2]) is a known NP-hard problem typically approached by offline methods. At design time, a model of the network and traffic requirements is created and an exhaustive search over the scheduling domain space – based on exact methods (e.g. [3], [4]) or heuristics (e.g. [5], [6]), is conducted. The high computational complexity of these methods makes them nonviable for online solutions.

Abstract modeling and simulations are common tools for the analysis of network properties and protocol performances, specially when the objects of study belong to new application domains for which no real deployment exists. Examples of modeling and simulation analysis for large IoT networks include [7], [8] and [9].

II. Methodology

Our aim in this analysis is to investigate the viability of online incremental scheduling methods suitable for typical scenarios motivated by the industrial IoT domain. We pursue a deep analysis of the scheduling problem abstracted from the network particularities (e.g. network protocols, schedule distribution mechanisms, security, etc). We consider, nonetheless, the distributed nature of the problem as well as a meaningful characterization of network topologies and traffic load based on realistic industrial scenarios (see Section III). We evaluate our analysis against a set of algorithms exploring alternative scheduling techniques, which range from trivial to medium complexity. Note that our goal is not the implementation of an efficient scheduling algorithm but rather the suitability evaluation of an online incremental approach for the synthesis of time-triggered schedules, as well as profiling a set of metrics and trade-offs having a key impact on feasibility, scalability, and efficiency.

A. Simulation Environment

We have designed a flexible simulation environment for the analysis and evaluation of distributed scheduling algorithms based on the principles of industrial control networks. The main purpose driving the design of this tool is the ability of fast prototyping implementation enabling testing and analyzing different scheduling methods. The set of inputs, which can be stored in files for later reuse, includes a complete description of

[1] IEEE TSN Task Group, online: http://www.ieee802.org/1/pages/tsn.html

a network topology. For our purposes (i.e. message scheduling) we limit the scope of the simulation to new route requests applied to the network switches (from now on *nodes*), as these are the devices which effectively implement the communication schedule. Note that we intentionally leave out the end-system nodes as we are not interested in the production and consumption of the data but merely in schedules for message transportation[2].

We differentiate three levels of nodes organized in a tree/star topology matching common industrial use-cases, and define a taxonomy as illustrated in Figure 1:

- **Edge nodes (e)**, are the bottom level switches typically operating at 1Gbps to which end-system are plugged.
- **Aggregation nodes (a)**, interconnect several edge nodes typically via 10Gbps links forming so called *clusters*.
- **Core nodes (c)**, are top-level nodes connecting clusters (i.e. aggregation nodes) typically via 100Gbps links.

Forwarding delays are set proportionally to typical values of 6μs, 4μs, and 3μs, respectively. Note that this classification as well as the characteristics of each node type is easily configurable within the simulation tool and is not restricted to any combination of node types.

The scheduling simulator core incrementally schedules new periodic communication routes following the selected algorithm. In essence, it tries to reserve a time slices (i.e. transmission window) in the corresponding timeline of each hop belonging to the network path between the source and destination nodes. Note that we assume switches with the capacity of simultaneously forwarding full-duplex traffic through all physical ports, and hence we assign an independent transmission schedule for each directional network link. From the moment the transmission windows at each node are reserved, the end-to-end timeliness is fixed and guaranteed and no other message will be scheduled overlapping the reserved time. New route requests are generated following a configurable traffic pattern, characterized by the frame size (between a configurable minimum a maximum), a communication period, a maximum end-to-end delay (E2E) as a factor of the period (e.g. $E2E = 50\%$, $E2E = 100\%$), and the communication locality (λ). We model the critical network traffic following a locality pattern in which a percentage of communication routes remain within the same cluster (i.e. source and destination edge nodes are connected to a common aggregation node). Typical use-cases in a factory floor set the locality to either extreme, being $\lambda = 90\%$ and $\lambda = 10\%$ values of interest, corresponding to e.g. control loops within a single machine, and communication between a local controller and the main factory floor controller.

B. Test-bed Algorithms

We evaluate our system against four different scheduling algorithms implementing alternative scheduling methods to service new requests:

CSI-1 **(Simple)** – A straightforward implementation dividing the total interval {frame arrival, deadline} into equal sub-intervals for each hop in the route. The algorithm tries to allocate a transmission window on each hop within its assign sub-interval and fails if no free interval is found for at least one hop.

CSI-2 **(Proportional)** – Similar to *CSI-1* but the sub-interval division is proportional to the current link utilization. Therefore, highly utilized links are assigned a larger region to search for an empty slot.

CSI-3 **(Iterative)** – A sequential approach taking the end of the allocated transmission window in link i as the beginning of the search interval for link $i + 1$.

CSI-4 **(Offset)** – A variation of *CSI-3* in which the starting of the transmission window on the first hop defines the offset with respect to the schedule cycle, i.e. the frame is offset with respect to the schedule cycle and can, potentially, wrap over the hyperperiod.

*** Random** – we analyze the impact of randomness in the search of the scheduling slot. For this, we implemented *CSI-2** and *CSI-3** similar as above, but instead of searching for the first free slot within the interval, sub-dividing it into n parts, which are then searched in random order. The actual n depends on the original interval size, such that each part can accommodate at least one maximum length frame. For efficiency, n is bounded to $\{1..32\}$. Our aim is to evaluate the impact of introducing random gaps in the timeline with respect to the chances of scheduling future requests.

Name	HP[i]	$\lvert\Pi\rvert$[ii]	Π[iii]
Π_1^{100}	100 ms	2	$\{100, 50\}$ ms
Π_2^{100}		4	$\{100, 50, 25, 10\}$ ms
Π_3^{100}		6	$\{100, 50, 25, 10, 5, 4\}$ ms
Π_1^{500}	500 ms	2	$\{500, 250\}$ ms
Π_2^{500}		4	$\{500, 250, 100, 50\}$ ms
Π_3^{500}		6	$\{500, 250, 100, 50, 25, 10\}$ ms
Π_1^{1000}	1000 ms	2	$\{1000, 500\}$ ms
Π_2^{1000}		4	$\{1000, 500, 250, 100\}$ ms
Π_3^{1000}		6	$\{1000, 500, 250, 100, 50, 25\}$ ms

[i] Hyperperiod [ii] Number of Periods [iii] Set of Periods

Table I: *Sets of periods and their Hyperperiod.*

III. Analysis

We perform simulations over the set of network topologies $\{T_a, T_b, Tc, T_d\}$ depicted in Figure 1. For each network topology, we generate multiple sets of new requests, each consisting of 250000 requests with frame sizes randomly chosen between $\{64..1518\}$ bytes. We define a set of hyperperiods (hp) $H = \{100, 500, 1000\}$ ms. For each $hp \in H$ we create randomly three sets $(\Pi_1^{hp}, \Pi_2^{hp}, \Pi_3^{hp})$ with cardinality 2, 4, and 6, respectively, $\left|\Pi_1^{hp}\right| = 2, \left|\Pi_2^{hp}\right| = 4, \left|\Pi_3^{hp}\right| = 6$, as summarized in Table I. For each such scenario, we produce two complementary sets with traffic locality $\lambda \in \{10\%, 90\%\}$.

We collect the following statistical data for each run:

- Total number of scheduled requests;
- First request that could not be scheduled;
- Average link utilization when first error occurred;
- Schedule (timeline) for each link;
- Utilization for each scheduled link;
- Memory utilization histogram per node;
- Simulation runtime (for complexity comparison).

[2]For a complete end-to-end scheduling approach including the generation and consumption of data in the end-systems refer to [3].

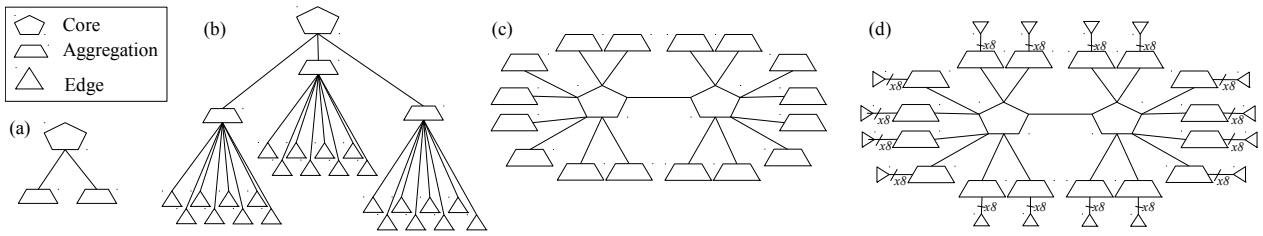


Figure 1: *Network topologies (a) $T_b = 1_c 2_a$, (b) $T_b = 1_c 3_a 8_e$ (c) $T_c = 2_c 8_a$, and (d) $T_d = 1_c 8_a 8_e$.*

A. Average Utilization Analysis

The simulation environment keeps track of the total number of successfully scheduled requests as well as the instance in which the first request could not be satisfied. Note that for the comparison of algorithms the same sequences of requests is used. Figure 2 summarizes the simulation results for the topology $1_c 3_a 8_e$. The simulation runs are grouped vertically by hyperperiod, $hp \in H$, and horizontally by number of periods ($\left|\Pi^{hp}\right|$). We additionally annotate the average utilization achieved after processing the total number of new requests (i.e. 250000). Figure 2(a) refers to results with $E2E = 100\%$, while Figure 2(b) depicts results with $E2E = 50\%$.

Not surprisingly, low utilization runs are successfully scheduled even with the simplest algorithms. However, we can observe an increasing trend on the occurrence of the first error as the average utilization increases. Adding randomness to the scheduling process (CSI-2* and CSI-3*) has different effects compared to the original algorithms. CSI-2* slightly improves schedulability over CSI-2, although CSI-3* performs worse than its counterpart. This may be due to different division of time window during the search of a free interval for a new request. CSI-2 relies on the link utilization but this can be misleading if the timeline is not filled equally for the entire hyperperiod length (e.g. the overall link utilization could be low although the utilization on the exact time-window of interest would be close to 100%). CSI-3 performs an iterative division which adapts better to the runtime conditions, independent of the utilization.

In general, adding random gaps within the timeline seems to serve the purpose of increasing the chances of accommodating future requests, although we observe a negative influence towards the total number of scheduled requests. In essence, future requests are of unknown size and therefore likely to under-utilize the available gaps when they are scheduled. This is not a problem when the overall utilization remains low, but as it increases, a larger number of gaps are left in the timeline, which are too small to fit new requests.

From the same figure, it is already visible that the number of different periods influences the occurrence of the first error. In that sense, independently of the algorithm, the higher the number of periods the sooner a new request results non schedulable. This is related to the repetition of instances for each request up to the hyperperiod. If all instances are exactly scheduled the same number of times (e.g. one single period), the fitting is done without conflicts. However, once the number of repetitions differs, some of the instances must "compete" with frames of different periodicity falling within their scheduling region.

B. Runtime Analysis

Figure 3 depicts the runtime of the previous simulations, showing the proportion of scheduled and not-scheduled requests. Note the logarithmic scale on the time axis. Without surprise, the runtime trend increases for algorithms CSI-1 to CSI-3. Randomness increases the runtime for CSI-2, bringing similar runtime bounds than CSI-3 and CSI-3$*$. However, CSI-3 outperforms the two random variants in most cases. We have observed different variations on other topologies with respect to the random algorithms (not shown due to space limitations).

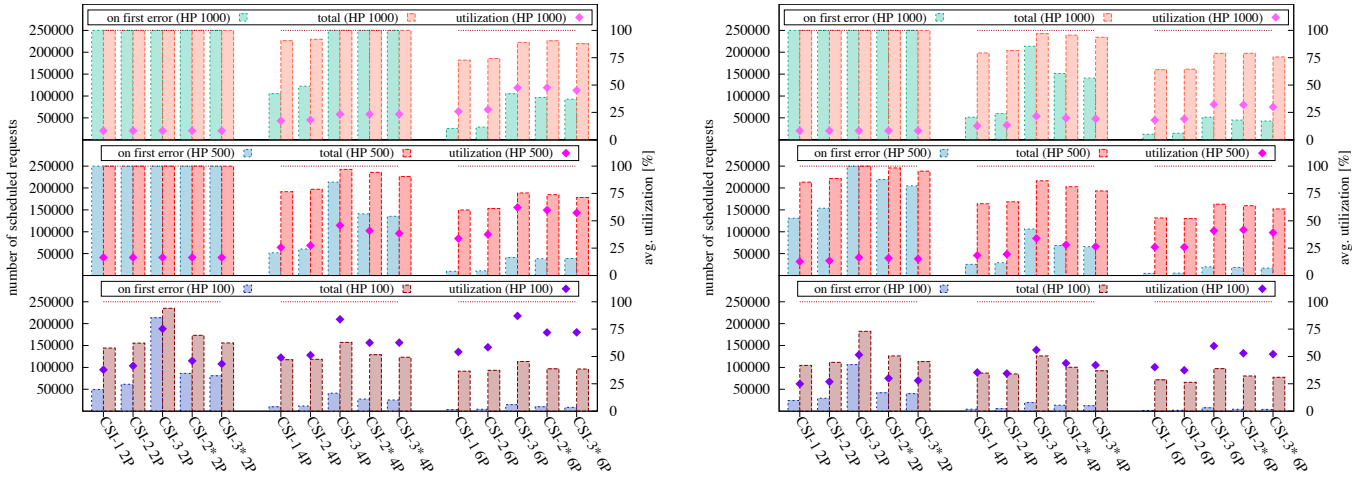
C. Link Utilization Analysis

Figure 4 shows the average utilization on links grouped by their direction of communication. In other words, all directional links are grouped based on the type of the node they belong to as well as the direction –uplink or downlink–. Note that the utilization is shown as a heat-map following the color scale on the right.

Subfigure 4(a) shows the total utilization after 250000 requests with locality $\lambda = 10\%$ have been processed, while Subfigure 4(b) corresponds to the same scenario with $\lambda = 90\%$. Here we can observe how the utilization follows a much more homogeneous distribution in the former case where most communication crosses the cluster boundary. In the latter, in which most of the communication takes place within the same cluster, the utilization is significantly higher on the up- and down-links connecting edge nodes. This analysis shows valuable information for the correct dimensioning of systems based on the topology and communication patterns. It may also help estimating the overall system capacity to support non-scheduled traffic, based on the available capacity at each communication level.

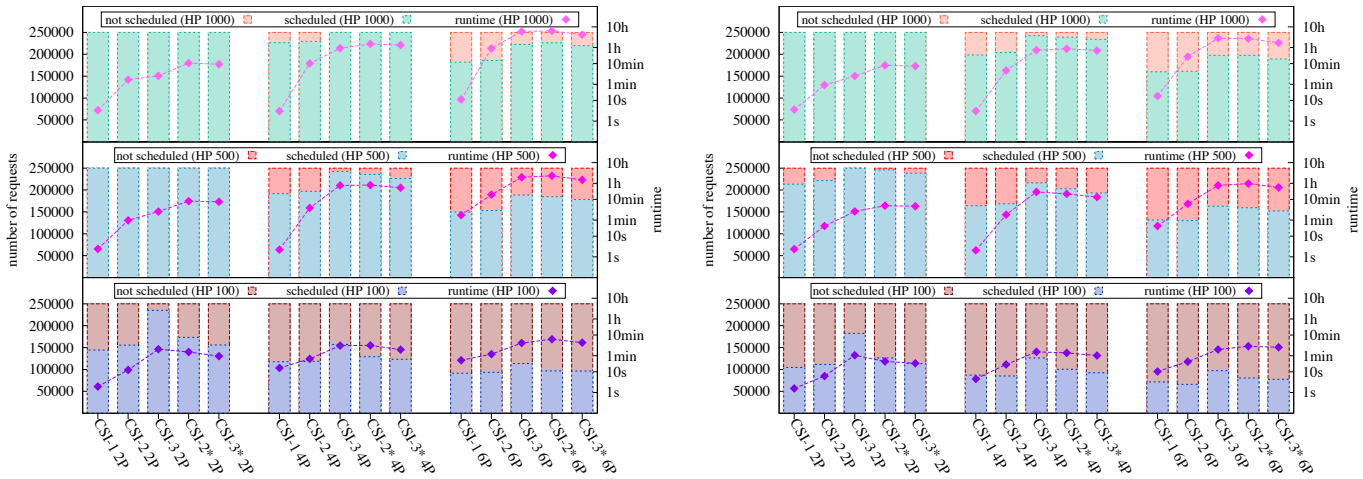
D. Memory Utilization Analysis

At the current stage of investigation we do not consider enforcing maximum bounds on the utilization of physical resources like memory. Our aim, however, is to enable the analysis of resource utilization achieved by the studied algorithms in a way that exposes trade-offs between the demand of physical resources and the schedule efficiency (i.e. cost vs performance). To do this, we run a post-simulation analysis on the memory utilization by adding up the number of frames inside the node's memory at each instant of time. Thanks to the schedule that we have build over time, we can simply keep count of the frames as they ingress and egress along the timeline. Figure 5 shows an example of such analysis for two selected nodes (note the different scale). For completion, we calculate in parallel the exact memory demand in bytes depending on the frame size (in green) as well as the number of frames (red line), assuming that most hardware reserves buffers of a pre-defined maximum frame size.



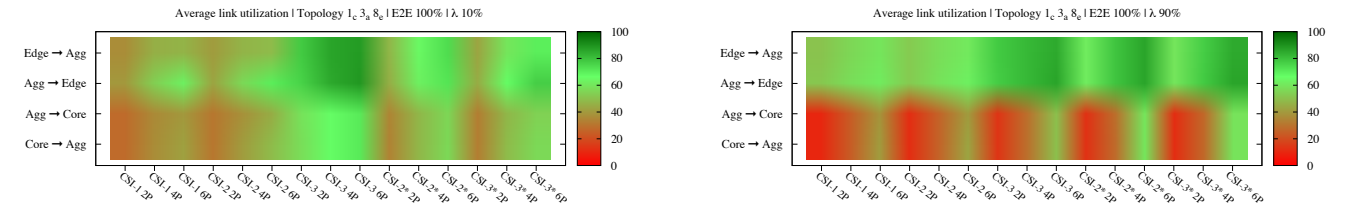
(a) $E2E = 100\%$ (b) $E2E = 50\%$

Figure 2: *Number of scheduled requests (total and first error) vs utilization on* $1_c 3_a 8_e$, $\lambda = 10\%$



(a) $E2E = 100\%$ (b) $E2E = 50\%$

Figure 3: *Runtime vs successful requests for* $1_c 3_a 8_e$, $\lambda = 10\%$



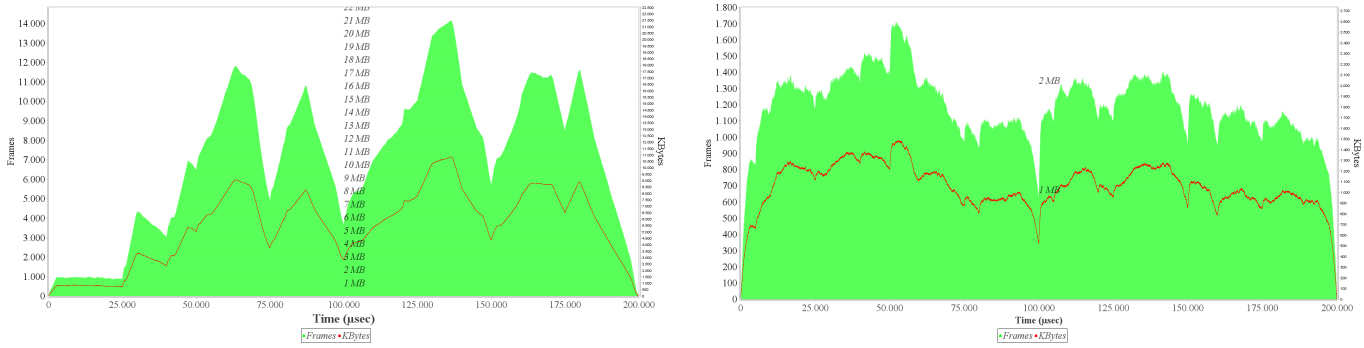
(a) $\lambda = 10\%$ (b) $\lambda = 90\%$

Figure 4: *Link utilization heat-maps for* $1_c 3_a 8_e$, $E2E = 100\%$, $HP = 100$.

E. Metrics and Trade-Off Opportunities

The analysis of an extensive set of simulations with alternative topologies and traffic loads already profiles a number of metrics providing a sense of quantity as well as quality for the analyzed algorithms. This metrics are especially valuable to evaluate quantitatively and qualitatively the performance of new scheduling algorithms, as well as for the identification of trade-off opportunities.

Hyperperiod – We devise a certain relation between the periods of the scheduled requests and the average computation time required to compute new requests. In particular, the relation between the hyperperiod (hp) and each one of the periods in the set of requests defines the amount of instances that need to be scheduled for each periodic request. Larger differences have a strong impact on the required computation time regardless of the scheduling algorithm (i.e. the algorithm



(a) CSI-3 (b) CSI-4

Figure 5: *Memory utilization comparison of two aggregation nodes under different algorithms, with topology* $1_c 3_a 8_e$, $\Pi = \{200, 100, 50, 40, 25\}$ *ms,* $\lambda = 90\%$, $E2E = 100\%$.

needs to schedule more instances for each request). In essence, a trade-off lays in the optionality of choosing arbitrary periods for each request –potentially leading towards large HP relation, against a predefined set of harmonic periods.

Resource scheduling – Additional constrains towards the utilization of resources add up to the overall scheduling complexity. We have used the memory utilization histogram over the scheduling cycle (i.e. hyperperiod) to analyze how the different algorithms perform. Including this and other resources into the scheduling algorithm increases the complexity significantly. We face, in this case, a trade-off between the algorithm complexity –and potentially execution time– and the accurate planning of resources in the network devices.

Raster size – In relation with the hyperperiod length, the raster size –that is, the granularity at which schedule events are possible; infers a huge impact on the algorithm runtime performance. Naturally, the smaller the raster size, the more opportunities of placing each scheduled frame within the timeline. On the other hand, a larger raster size reduces the search space at the expenses of potentially losing valid fittings. The trade-off between the two extremes depends on the specific use-cases and hardware support. If the use-case does not require highly utilized scheduled links, a higher raster size may improve the overall performance significantly. However, maximum utilization may only be achieved if frames are places side-by-side on the lowest time granularity possible. In addition, device-specific properties may set a minimum raster size based on its own hardware design.

End-to-end Latency – The end-to-end scheduling window in which all frames need to be scheduled for each intermediate hop has an obvious impact on the algorithm performance. The wider this window is, the more flexibility is given to the algorithm to find a placement for each frame. E.g. the analysis of simulation runs in which the relation between the period and the maximum allowed end-to-end delay ($E2E$) is 1 show a higher scheduling success than those with $E2E = 2$ (i.e. maximum end-to-end is half of the period). If the allowed end-to-end latency is smaller than the period, the placement of the fist frame on the initial hop determines the scheduling window for all subsequent frames.

Link utilization – The average as well as maximum link utilization are good quality indicators for the scheduling algorithm. In essence, these values reflect how well the algorithm performs finding an empty slot for a new frame. We have also considered the utilization achieved upon the first non schedulable request as a benchmark to reflect the advantages of different scheduling techniques provisioning 'gaps' in the schedule for future requests.

IV. Conclusion

In this paper, we summarize the initial results our systematic analysis, based on dynamic scheduling simulation, targeting deterministic scheduled networks for industrial control systems in the context of IoT. We show how such analysis can provide valuable hints regarding the feasibility and complexity inherent to the synthesis of schedules, online and incrementally, providing deterministic guarantees. Our findings expose relevant variables and design decisions with a significant impact on key performance metrics. We profile a number of trade-offs based on the results obtained through an extensive set of simulations based on industrial-size scenarios.

References

[1] I. Ishaq, D. Carels, G. K. Teklemariam, J. Hoebeke, F. V. d. Abeele, E. D. Poorter, I. Moerman, and P. Demeester, "IETF standardization in the field of the internet of things (IoT): A survey," *Journal of Sensor and Actuator Networks*, vol. 2, 2013.

[2] W. Steiner, "TTEthernet specification," TTA Group, 2008. [Online]. Available: http://www.ttagroup.org

[3] S. S. Craciunas and R. Serna Oliver, "SMT-based task- and network-level static schedule generation for time-triggered networked systems," in *Proc. RTNS*. ACM, 2014.

[4] W. Steiner, "An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks," in *Proc. RTSS*. IEEE, 2010.

[5] D. Tamas-Selicean, P. Pop, and W. Steiner, "Synthesis of communication schedules for TTEthernet-based mixed-criticality systems," in *Proc. CODES+ISSS*. ACM, 2012.

[6] L. Zhang, D. Goswami, R. Schneider, and S. Chakraborty, "Task- and network-level schedule co-synthesis of ethernet-based time-triggered systems," in *Proc. ASP-DAC*. IEEE, 2014.

[7] S. Karnouskos and T. de Holanda, "Simulation of a smart grid city with software agents," in *Proc. EMS*. IEEE Computer Society, 2009.

[8] I. Lanese, L. Bedogni, and M. Di Felice, "Internet of things: A process calculus approach," in *Proc. SAC*. ACM, 2013.

[9] K. V. Jónsson, Y. Vigfússon, and O. Helgason, "Simulating large-scale dynamic random graphs in omnet++," in *Proc. SIMUTOOLS*, 2012.