Exploring Practical Limitations of Joint Routing and Scheduling for TSN with ILP

Jonathan Falk, Frank Dürr, Kurt Rothermel
Institute of Parallel and Distributed Systems
University of Stuttgart
Stuttgart, Germany
Email: {jonathan.falk,frank.duerr,kurt.rothermel}@ipvs.uni-stuttgart.de

***Abstract*—IEEE 802.1Q networks with extensions for time-sensitive networking aim to enable converged networks. Converged networks support hard-real time communication services in addition to the currently supported services classes. Real-time communication in these networks requires routes and schedules for the real-time transmissions. We present a formulation in the integer linear programming (ILP) framework which models the joint routing and scheduling problem for flows of periodic real-time transmissions in converged TSN networks. In the joint routing and scheduling problem, both routes and schedules for real-time transmissions are computed in one step, i.e. we do not schedule over predefined routes. We explore the practical limitations of this approach by evaluating the runtime of problem instances with widely varying parameters with a state-of-the-art ILP solver. The observed solver runtimes indicate the qualitative impact of the number of real-time flows, the size of the network, the transmission frequency of real-time transmissions, and the network topology.**

I. Introduction

The realization of envisioned cyber-physical systems (CPS) for Industry 4.0, Smart Grids and Internet of Things depends heavily on real-time capable communication networks. In many cases, the design of such real-time networks relies on deterministic medium access in combination with a suitable scheduling and resource reservation mechanism to ensure communication with deterministic delay bounds.

The popular Ethernet standard [1] in switched, full-duplex operation complying to IEEE 802.1Q [2] was traditionally lacking in mechanisms to ensure real-time capabilities in case of existing cross-traffic. Currently, the Time-Sensitive Networking (TSN) Task Group is working on extensions to the IEEE 802.1Q standard to solve this problem. One of these extensions is a time-triggered gating mechanism in the switches standardized in IEEE 802.1Qbv [3]. The gating mechanism consists of three major components: a) gates, b) gate control lists, and c) gate drivers. The gates exist per output-port queue per switch and are either "open" or "closed". Frames from the associated output queue are eligible for transmission if the gate is open, and prohibited to be selected for transmission if the gate is closed. The gate control list functions as a schedule for the gate driver, and can be configured per output port. The gate driver iterates over the entries in the gate control list, and changes the states of the gates at the specified time. Consequently, this gating mechanisms allows to enforce cyclic transmission schedules.

Network equipment vendors often stress that the TSN efforts and especially the gating mechanism enable converged networks. The term "converged network" describes the idea of employing one unified Ethernet network at an industrial site, which not only carries the usual business / office data but also replaces traditional proprietary real-time networks, possibly improving vertical business integration. In this scenario, the gating mechanism can be used to separate the office data which is non-real-time traffic, and is generated whenever a business interaction requires it, from the real-time data which we assume to be transmitted isochronously. Real-time data can, e.g., be generated and consumed by distributed control applications where sensor data and control messages have to be exchanged via the network. If the gating mechanism blocks the transmission of non-real-time data when a real-time transmission is expected, real-time data can be transmitted separately and unaffected from non-real-time transmissions.

There remains a problem to solve, though, and a NP-hard one at that: We have to ensure, that real-time transmissions do not interfere with other real-time transmissions. There are two basic methods to achieve this, namely spatial isolation and temporal isolation. Spatial isolation assigns transmissions to disjoint network resources (switch queues, links) to prohibit interference. Spatial isolation leads to a routing problem. Temporal isolation assigns transmissions to disjoint time slots, and leads to a scheduling problem. For real-time communication with very low latency, temporal isolation requires time-synchronization of network elements which can be achieved by synchronization protocols such as PTP [4] or IEEE 802.1AS [5]. Either spatial isolation or temporal isolation can in theory be used alone to enable real-time in Ethernet, but there is one significant drawback: the maximally achievable network utilization is very low due to the restriction to disjoint routes or network-wide schedules. The low network utilization can be alleviated by combining the two methods to a joint routing and scheduling problem.

There are many ways, how one can attempt to solve the joint routing and scheduling problem. One can either try to design a "customized" algorithm, which exploits as much knowledge about the problem (i.e. communication patterns, heuristics for common use-cases, etc.) as possible. In this case, the difficulty

Published in proceedings of the 2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2018) Hakodate, Japan, Hakodate, Japan, August 29-31, 2018, http://dx.doi.org/10.1109/RTCSA.2018.00025

lies not only in showing the correctness of the algorithm, but also in implementing it efficiently. Alternatively, one may express the problem in a generic formal framework such as (integer) linear programming (ILP), SAT, SMT and then use an off-the-shelf solver. In this case, the hope is (provided it is even possible to map the problem at hand to such a generic framework) that many years of research and development of these generic frameworks and solvers make the solving process fast enough. In this paper, we investigate the latter approach and use integer linear programming. Integer linear programming is a well-established, mature approach which can in theory be used to solve all problems that can be modeled with integer decision variables, where the objective function is a linear function of the decision variable, and the decision variables are constrained by a set of linear inequalities.

Our first contribution is to provide an ILP formulation of the joint routing and scheduling problem for arbitrary isochronous communication with zero-queuing. If a feasible solution exists, the routes and schedules can be used to configure IEEE 802.1Qbv-compliant Ethernet networks. Even though the boundaries of the problem sizes that can be tackled by state-of-the-art computers with state-of-the-art ILP solvers are continuously extended either by new hardware or by new optimizations, it is still possible to end up with ILP problem instances that take several days or even weeks to solve. Therefore our second contribution are extensive evaluations of a large variety of problem instances of the joint routing and scheduling problem to explore the scalability of ILP-based formulations.

The remainder of the paper is structured as follows: We give a short overview over existing approaches for solving the joint routing and scheduling problem in communication networks in Section II. We describe our system model and the problem in Section III, and subsequently derive and explain the ILP formulation in Section IV. In Section V, we describe our evaluation scenarios and setup, and evaluate the solver runtime. Finally, we discuss the results and open questions in Section VI.

II. Related Work

While there are many approaches for solving only the scheduling problem for periodic real-time transmissions where routes are computed before searching for schedules [6]–[11], there exists fewer work on the joint-routing and scheduling problem.

In [12], the authors present an approach for scheduling of periodic transmissions in SDN networks. which they extend in [13] to support dynamic addition or removal of flows. They provide ILP-formulations for the scheduling problem for sets of predefined routes, and they also provide an ILP-formulation to determine the routes and schedules jointly. Their approach is limited to uniform transmission lengths, a network-wide base-period, and does not use the gating mechanism, since schedules are network-wide. The ILP formulation which we present supports arbitrary length of reservations for transmissions, arbitrary periods of transmissions, and per-port schedules for TSN gates.

In [14], a 0-1 ILP formulation is presented for the joint message routing and scheduling problem which builds on an existing model of automotive communication networks. The authors do not consider the end-to-end delay of messages for the routing decision, and employ a binary coding which directly relates the granularity of the schedules to the number of variables. The ILP formulation in [14] is targeted at a very specific use-case, since the authors aim to utilize their ILP as component in a multi-objective optimization for automotive communication networks. This is noticeable, e.g., in the choice of variables and encoding. Since we aim for more generic application scenarios, we do not have to restrict ourselves to purely binary variables.

In [15], the authors independently developed an ILP formulation for the joint routing and scheduling problem. The ILP in [15] allows queuing, but per-switch reordering of messages belonging to different flows is not constrained. Thus, the solutions of the ILP in [15] are not guaranteed to be directly applicable to TSN networks in all cases. With our zero-queuing approach, we can guarantee FIFO-queuing for all parameters. Only three network topologies with a vertex degree of maximally three are used for the evaluations in [15] which are focused on the schedulability and optimality of the computed schedules.

III. Problem Statement and System Model

In this section, we will describe our system model and state the problem which we will solve with the ILP in Section IV.

A. System model

We assume a set of applications shall operate in a given, shared ("converged") network. Real-time applications require periodic transmissions with hard real-time guarantees (i.e. an end-to-end-delay lower than a threshold, and bounded jitter) from one source host to another sink host. The repeated real-time transmissions along the path from source to sink are called "flow". All real-time transmissions belonging to a specific flow shall be routed along the same path. Transmission periods can be chosen individually per flow, but have to be finite integer values. We assume that real-time flows need not be synchronized to wall-clock time, nor to other flows.

We model the network topology as a directed graph $G(V, E)$. The vertices in V correspond to identical network elements (switches). Edges in E correspond to an output port plus the attached link of a switch. We assume all edges have identical properties. The transmissions of the real-time applications "originate" (are inserted into the network) at the origin vertex. Equivalently, real-time transmissions are "destined" for a specific destination vertex in the graph, where they are removed from the network. Real-time transmissions have to reach the destination vertex within real-time application-specific delay-bounds. "Origin" and "destination" vertices correspond to the ingress / egress switches where the source host and sink host are attached. Real-time transmissions are

defined by the duration during which they occupy an edge, not by the length of the transmission in data units. Each switch can forward transmissions from incoming edges to any outgoing edge. All effects that add to the per-hop delay in networks (i.e. delays for processing, transmission, and propagation) are reduced to a single constant delay D that is applied when a real-time transmission traverses a vertex (cf. Figure 1).

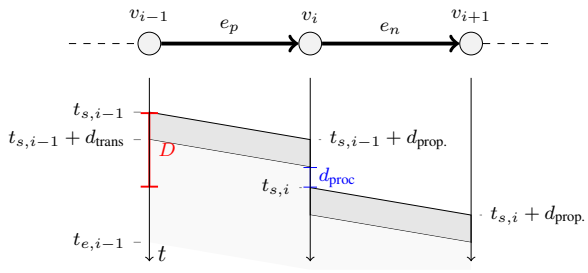

Fig. 1. Relating reservations and switch schedules.

As illustrated in Figure 1, D can be intuitively interpreted as the time between the start of the real-time transmission on edge e_p and the start of the real-time transmission on the next edge e_n adjacent to e_p. For example, in a switched Ethernet network with store-and-forward architecture, $D = d_{\text{trans}} + d_{\text{prop}} + d_{\text{proc}}$, where d_{trans} is the transmission delay for one frame, d_{prop} is the propagation delay for one edge, and d_{proc} is the internal processing delay of the switches. Note, that this requires either uniform frame sizes for all real-time applications. Alternatively, d_{trans} has to be set to the maximal size of frames transmitted by real-time applications. In this case, a frame with smaller frame size (and associated $d_{\text{proc},i}$) would idle in the switch for $d_{\text{proc,max}} - d_{\text{proc},i}$ time units. Heterogeneous frame sizes and switches could be incorporated by splitting D in a flow-specific transmission delay part and a vertex specific delay part. By describing the delay and transmissions via time units (abstract integer numbers, that can be mapped to SI units depending on the scenario), we abstract from the underlying network implementation.

B. Problem

We search for a configuration (routing and scheduling) under which all real-time applications can work properly in the network. The problem will be formulated in Section IV as an ILP: Given a network, and a set of real-time applications with given transmission source and transmission destination, as well as transmission periods, durations of transmissions and end-to-end latency bounds for transmissions (end-to-end deadlines), return routes and schedules for "all" flows, (if existing) which satisfy the following requirements:

- A valid route is a connected sequence of directed edges from origination vertex to destination vertex that crosses every vertex only once (loop-free).
- At every point in time on every edge can be at most one real-time transmission.
- Every transmission has to reach its destination vertex within the flow-specific end-to-end deadline.
- At every vertex, a transmission can only be transmitted on an outgoing edge, if all other transmissions which have arrived earlier at the same vertex, and which have to be transmitted on the same outgoing edge have been transmitted before (FIFO property).
- Every real-time transmission is immediately forwarded (only experiencing the aforementioned constant delay).

In fact, the last two requirements (FIFO and immediate forwarding of transmissions) are targeted at a specific configuration of TSN-capable switches, where 1 queue per output port is exclusively used for real-time transmissions, and every incoming real-time transmission always encounters this queue empty (zero-queuing). With zero-queuing, the lowest possible end-to-end delay is achieved along a route, which we think is a favorable property for real-time transmissions, and it ensures the FIFO property, too. This also means, that the sole function of the end-to-end deadline of transmissions is to restrict the route length. The gating mechanism is used in this configuration to open the gate of the "real-time" queue as soon as the first frame of a real-time transmission has been received while shutting the gates of all queues containing non-real-time traffic until all frames of the real-time transmission have been transmitted (temporal isolation).

Note, that the way we stated the problem, the solution can not just only be mapped to Ethernet networks with support for gating but to any network with output-queued ports without reordering, a deterministic MAC, and either no unsynchronized cross-traffic (cf. TSSDN [12]) or mechanisms to block unsynchronized cross-traffic during real-time transmissions.

IV. Integer Linear Program

A. Notation

We use calligraphic letters (e.g., $\mathcal{V}, \mathcal{E}, \mathcal{F}$) to denote sets, uppercase bold letters for matrices. and lowercase bold letters for vectors. If we want to express in an expression that an index $i \in \mathcal{I}$ may have any possible value from its domain, we use $*$ as a wild card character. We assume a mapping enum_V : graph vertex $\rightarrow v$ exists, which assigns an index $v \in \mathbb{N}$ to every vertex of the graph. Similarly, we assume a mapping enum_E : graph edge $\rightarrow e$ exists, which assigns an index $e \in \mathbb{N}$ to every edge in the graph. A directed edge with index e which departs at vertex with index v_s and enters vertex with index v_t is denoted as $e = (v_s, v_t)$.

B. ILP formulation

In this section, we formulate the problem as an Integer Linear Program. We will use logical operators for conditional constraints (if) and alternative constraints (or) that are not part of the standard or canonical forms of ILPs. The if-operator is used in expression such as

$$\text{if (lin. constraint 1) then (lin. constraint 2)}$$

where the linear constraint 2 only has to be satisfied, if linear constraint 1 is satisfied. The or-operator is used in expressions such as

$$\text{(lin. contraint 1 or lin. constraint 2)},$$

where only one of the constraints needs to apply. Both of these can be transformed into a set of linear (in-)equalities by the "big M"-technique if the involved variables are bounded (which they are in our problem) to get the ILP in standard or canonical form. Using the aforementioned operators not only results in better readability, but state of the art solvers such as CPLEX and Gurobi support indicator constraints [16] to handle these operators without encountering the numerical problems that can occur when using the "big M"-technique. Also, state-of-the-art ILP modeling environments such as zimpl [17], GAMS, AMPL or OPL support modeling problems with these extended operators, and perform the required transformations to a set of linear inequalities that can be processed by the ILP solvers.

With that being said, we next derive graph related parameters and flow related parameters that we use for the formulation of the joint routing and scheduling problem. Graph related parameters are given in Table I.

TABLE I
ILP PARAMETERS DERIVED FROM GRAPH.

$\mathcal{E} = \{0, 1, 2, \ldots\} \subset \mathbb{N}$	set of edges
$\mathcal{V} = \{0, 1, 2, \ldots\} \subset \mathbb{N}$	set of vertices
$\mathbf{A}_{EE} \in \{0, 1\}^{\lvert\mathcal{E}\rvert \times \lvert\mathcal{E}\rvert}$	sparse edge-edge adjacency matrix
$\mathbf{B}_{VE} \in \{-1, 0, 1\}^{\lvert\mathcal{V}\rvert \times \lvert\mathcal{E}\rvert}$	sparse vertex-edge incidence matrix
$D \in \mathbb{N}$	time it takes for crossing over a vertex

The value of an element of the edge-edge adjacency matrix $\mathbf{A}_{EE}$ in row $e_p \in \mathcal{E}$ and column $e_n \in \mathcal{E}$ is given by

$$\mathbf{A}_{EE}[e_p][e_n] = \begin{cases} 1, & \text{if } e_p = (*, v) \text{ and } e_n = (v, *) \\ 0, & \text{otherwise} \end{cases} . \quad (1)$$

That means, if the edge (with index) e_p enters a vertex $v \in \mathcal{V}$, and the edge e_n departs from the same vertex v, then $\mathbf{A}_{EE}[e_p][e_n]$ contains the value 1. The value of an element of the vertex-edge incidence matrix $\mathbf{B}_{VE}$ in row $v \in \mathcal{V}$ and column $e \in \mathcal{V}$ is given by

$$\mathbf{B}_{VE}[v][e] = \begin{cases} 1, & \text{if } e = (v, *) \\ -1, & \text{if } e = (*, v) \\ 0, & \text{otherwise} \end{cases} . \quad (2)$$

That means, $\mathbf{B}_{VE}[v][e] = 1$ if the edge e departs from vertex with index v, and $\mathbf{B}_{VE}[v][e] = -1$ if the edge e enters vertex v. We assume that for most scenarios $\deg(v) \ll |\mathcal{V}|$. In this case, storing only non-zero entries of $\mathbf{B}_{VE}$ and $\mathbf{A}_{EE}$ is more efficient.

The flow related parameters and their meanings are given in Table II. Since ILPs support no modulo-operation in the constraints, we define the hyper-cycle, that is the period of length h with which the reservation pattern repeats from a *global* point of view. We use h in the scheduling constraints to ensure interference-free transmission time in all transmission periods.

After introducing the ILP parameters, we introduce the variables for the ILP. Our ILP formulation follows a "constructive" approach where we can read the solution for the joint routing and scheduling problem directly from the ILP solution. Consequently, we introduce a set of binary decision variables $\mathbf{u} \in \{0, 1\}^{|\mathcal{F}| \times |\mathcal{V}|}$ with the interpretation

TABLE II
ILP PARAMETERS DERIVED FROM FLOW PROPERTIES.

$\mathcal{F} \subset \mathbb{N}$	set of flow indices
$\mathbf{o}_F \in \mathcal{V}^{\lvert\mathcal{F}\rvert}$	vector of flow origins: $\mathbf{o}_F[f] = v$, iff flow $f \in \mathcal{F}$ starts at vertex $v \in \mathcal{V}$
$\mathbf{d}_F \in \mathcal{V}^{\lvert\mathcal{F}\rvert}$	vector of flow destinations: $\mathbf{d}_F[f] = v$, iff flow $f \in \mathcal{F}$ ends at vertex $v \in \mathcal{V}$
$\mathbf{p}_F \in \mathbb{N}^{\lvert\mathcal{F}\rvert}$	vector of flow period: $\mathbf{p}_F[f] = p_f$, iff flow $f \in \mathcal{F}$ has a period of $p_f \in \mathbb{N}$
$\mathbf{r}_F \in \mathbb{N}^{\lvert\mathcal{F}\rvert}$	vector of required transmission durations of flows per period: $\mathbf{r}_F[f] = r_f$, iff flow $f \in \mathcal{F}$ needs a reservation of $r_f \in \mathbb{N}$
$\mathbf{l}_F \in \mathbb{N}^{\lvert\mathcal{F}\rvert}$	vector of maximally allowed end-to-end deadline for flows: $\mathbf{l}_F[f] = t_f$, iff flow $f \in \mathcal{F}$ has a maximal end-to-end deadline of $t_f \in \mathbb{N}$
$h = \mathrm{lcm}(\mathbf{p}_F)$	least common multiple of all individual flow periods

$$\mathbf{u}[f][e] = \begin{cases} 1 \rightarrow \text{flow } f \text{ uses edge } e \\ 0 \rightarrow \text{flow } f \text{ does not use edge } e \end{cases} \quad (3)$$

from which we can derive the routes of the flows after having solved the ILP. Analogously, we introduce a set of bounded variables

$$\mathbf{t} \in \mathbb{N}^{|\mathcal{F}| \times |\mathcal{V}|} \text{ with } 0 \leq \mathbf{t}[f][*] \leq \mathbf{p}_F[f] - \mathbf{r}_F[f] \quad (4)$$

where the value $\mathbf{t}[f][e]$ after solving the ILP is equal to the point in time (modulo the flow period) from which on edge e is reserved for transmission of flow f, if $\mathbf{u}[f][e] = 1$.

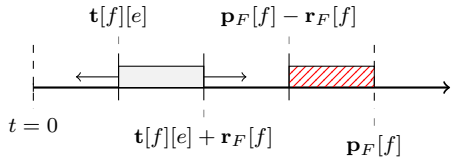

Fig. 2. Upper bound on $\mathbf{t}[f][*]$ from Equation 4: Reserved time has to finish within period.

Setting the upper bound to $\mathbf{p}_F[f] - \mathbf{r}_F[f]$ (illustrated in Figure 2) ensures that on every link used by flow f the reserved time slot ends before the next transmission period starts. From the values of $\mathbf{t}[*][e]$ we can easily construct the schedule per edge, i.e. the output port schedule.

In this paper, any feasible solution for the joint routing and scheduling of the flows is acceptable. Therefore, we introduce a "placeholder" objective function $\min q$ on a bounded, "dummy" variable q that can be replaced in the future with "real" optimization objectives (e.g. distribution of reservations on edges).

We proceed to explain the constraints, that encode the requirements for a transmission pattern that separates transmissions in time (via scheduling) as well as in space (via routing). These constraints can be categorized in three groups: 1) constraints, which ensure per-edge compliance of reserved

time slots, 2) constraints, which ensure loop-free routing from flow origin to flow destination, and 3) constraints, which link the routing and scheduling constraints.

1) Scheduling Constraints: The first scheduling constraint ensures that the reserved time slots of any flows using a particular edge do never overlap, and is given by

$$\forall f_1 \in \mathcal{F}, f_2 \in \mathcal{F} : f_1 \neq f_2, e \in \mathcal{E} :$$
$$\forall a \in \mathcal{A}, b \in \mathcal{B} :$$
$$\text{if } (\mathbf{u}[f_1][e] + \mathbf{u}[f_2][e] \geq 2) \text{ then} \tag{5}$$
$$(\mathbf{t}[f_1][e] + a \cdot \mathbf{p}_F[f_1] \geq \mathbf{t}_F[f_2][e] + b \cdot \mathbf{p}_F[f_2] + \mathbf{r}_F[f_2] \tag{6}$$
$$\text{or } \mathbf{t}[f_2][e] + b \cdot \mathbf{p}_F[f_2] \geq \mathbf{t}_F[f_1][e] + a \cdot \mathbf{p}_F[f_1] + \mathbf{r}_F[f_1]) \tag{7}$$

with

$$\mathcal{A} = \left\{ a \in \mathbb{N} : 0 \leq a \leq \frac{h}{\mathbf{p}_F[f_1]} \right\}, \tag{8}$$
$$\mathcal{B} = \left\{ b \in \mathbb{N} : 0 \leq b \leq \frac{h}{\mathbf{p}_F[f_2]} \right\}. \tag{9}$$

It is not sufficient to restrict the starting times of the reservations such that they do not overlap in one single period, e.g., see Figure 3, where the collision occurs in the 3rd period (from the perspective of flow f_1), respectively, the 2nd period (from the perspective of flow f_2). Instead, we have to prohibit that any overlap occurs for all future periods. For this purpose, we introduce auxiliary sets $\mathcal{A}$ and $\mathcal{B}$, where $a \cdot \mathbf{p}_F[f_1][e]$ denotes the start of the ath period of flow f_1, and $b \cdot \mathbf{p}_F[f_1][e]$ denotes the start of the bth period of flow f_1, respectively, in the hyper-cycle.

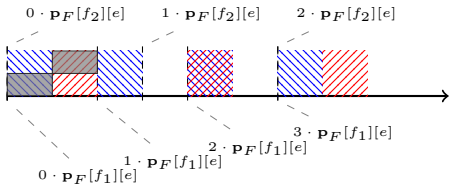

Fig. 3. Scheduling Constraint from Inequality 6 and Inequality 7: Reserved time slots of flow f_1 in iteration 2 overlaps with reserved time slot of flow f_2 in iteration 1 with respect to the hyper-cycle.

We check that for all combinations of $a \in \mathcal{A}$ and $b \in \mathcal{B}$ no overlap occurs. Note, that we need the or-conjunction that links the constraints in Inequality 6 and Inequality 7 since either the reservation of flow f_1 starts after the reservation of flow f_2 has ended, or vice versa.

2) Routing Constraints: The routing constraint

$$\forall f \in \mathcal{F} : \sum_{e \in \mathcal{E}} \mathbf{B}_{VE}\left[\mathbf{o}_F[f]\right][e] \cdot \mathbf{u}[f][e] = -1 \tag{10}$$

achieves that one departing edge from the vertex where flow f originates is used. Likewise, the constraint

$$\forall f \in \mathcal{F} : \sum_{e \in \mathcal{E}} \mathbf{B}_{VE}\left[\mathbf{d}_F[f]\right][e] \cdot \mathbf{u}[f][e] = 1 \tag{11}$$

enforces that one incoming edge at the destination vertex of flow f is used. The constraint

$$\forall f \in \mathcal{F} : \sum_{\mathbf{B}_{VE}[v][e]=1} \mathbf{B}_{VE}[v][e] \cdot \mathbf{u}[f][e]$$
$$= - \sum_{\mathbf{B}_{VE}[v][e]=-1} \mathbf{B}_{VE}[v][e] \cdot \mathbf{u}[f][e] \tag{12}$$
$$\text{with} \quad v \in \mathcal{V} \backslash \{\mathbf{o}_F[f], \mathbf{d}_F[f]\}, e \in \mathcal{E} \tag{13}$$

restricts the number of ingoing edges used by flow f to the number of outgoing edges used by flow f at each vertex. In combination with the constraints in Equality 10 and Equality 11, the assignment of $\mathbf{u}[f][*]$ has to be such, that a loop free path from flow origin vertex to flow destination vertex is created.

3) Constraints Linking Routing and Scheduling Constraints: So far, the routing and scheduling constraints restrict either the values of $\mathbf{u}$, or the values of $\mathbf{t}$. The constraint given by

$$\forall e_p \in \mathcal{E}, \forall e_n \in \mathcal{E} : e_p \neq e_n \text{ and } \mathbf{A}_{EE}[e_p][e_n] = 1, \forall f \in \mathcal{F} :$$
$$\text{if} (\mathbf{u}[f][e_p] + \mathbf{u}[f][e_n] \geq 2) \text{ then} \tag{14}$$
$$(\mathbf{t}[f][e_n] = \mathbf{t}[f][e_p] + D \tag{15}$$
$$\text{or } \mathbf{t}[f][e_n] + \mathbf{p}_F[f] = \mathbf{t}[f][e_p] + D) \tag{16}$$

specifies that the reserved time slot on the outgoing edge e_n at a vertex v on the path of a flow f is shifted by D compared to the incoming edge e_p on the path (cf. Figure 4). There are

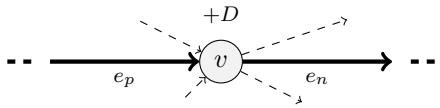

Fig. 4. The constraints from Equality 14-16 relate the reservations along the edges of the path of flow f.

two different cases, how to determine the reservation on the upstream edge e_n. In variant 1 (cf. Figure 5), the reservation on the incoming edge e_n is early enough, so that the reservation on the outgoing edge is in the same period on both edges.

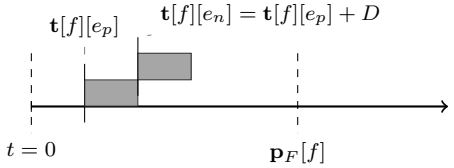

Fig. 5. Illustration of the constraint from Equality 15 where the reservation on the departing edge e_n is shifted by D compared to the reservation on the incoming edge e_p at vertex v. The position of the reservations relative to the start and end of the period are the same from the perspectives of the respective edges and from a global perspective.

Variant 2 (cf. Figure 6) can occur the reservation on the incoming edge e_p is "too close" to the end of the period on edge e_p. Then the transmission in the reserved time slot on the incoming edge e_p in the period with number n will be continued on the outgoing edge e_n in the next period

with number $n+1$. From an edge local perspective, the corresponding reservation on the outgoing edge e_n therefore is earlier in the period compared to the reservation on the incoming edge e_p.

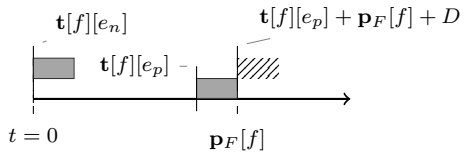

Fig. 6. Illustration of the constraint from Equality 16 where the periodicity of the reservation has to be accounted for. The position of the reservations relative to the start and end of the period from the perspectives of the respective edges are filled in gray. The striped area right of $\mathbf{p}_F[f]$ is the position of the reservation on e_n from a global perspective.

To incorporate upper bounds on the per-flow end-to-end delay, we introduce the constraint

$$\forall f \in \sum_{e \in \mathcal{E}} D \cdot \mathbf{u}[f][e] \leq \mathbf{l}_F[f]. \tag{17}$$

This constraint limits the number of edges in the path of a flow f by limiting the "delay" that can be accumulated along the edges.

C. Complexity

The joint routing and scheduling problem is NP-complete. We sketch the proof by reducing Bin-Packing [18] to the joint routing and scheduling problem. For any given Bin-Packing instance b with $B \in \mathbb{N}$, finite set $\mathcal{U}$ ($\forall u \in \mathcal{U} : s(u) \in \mathbb{N}$), and $K \in \mathbb{N}$ we create a special instance j of the joint routing and scheduling problem. The network of j consists of start node v_o and destination node v_d, K intermediate nodes v_i and $2K$ directed edges (v_o, v_i) and (v_i, v_d) with $i \in \{1 \dots K\}$ and w.l.o.g. $D = 1$, For each $u \in \mathcal{U}$ there is a flow in j from v_o to v_d with reservation $s(u)$ and transmission period B. Iff j has a solution, then $\mathcal{U}$ can be partitioned into K disjoint sets $\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_K$ s.t. $\forall \mathcal{U}_i : \sum_{u \in \mathcal{U}_i} s(u) \leq B$.

V. Evaluation

In this section, we will evaluate the performance of the ILP formulation introduced in Section IV by solving synthetic joint routing and scheduling problem instances. A *problem instance* consists of a graph and a set of flows defined by their parameters (flow endpoints, transmission period, transmission duration and end-to-end deadline). The evaluation setup (including how we generate the joint routing and scheduling problem instances) is explained, before we evaluate the solving performance.

A. Problem Generation

We generate the joint routing and scheduling problem instances in a two-step process. Firstly, we generate a (random) graph. The graph generation is parameterized by the number of vertices and the graph model (line, ring, scale-free, random). Those network models are of practical relevance (line graphs, ring graphs), and belong to fundamentally different classes of topologies (random scale-free graphs (Barabási-Albert network model), and random graphs (Erdős-Rényi network model)). Secondly, we generate a set of flows.

We use graph generators provided by the Python graph-tool [19] library to generate graphs according to the mentioned network models. For all topologies, we define the number of vertices $|\mathcal{V}|$, whereas the number of edges $|\mathcal{E}|$ depends on the graph model. Networks with a line or ring topology are often found in industrial scenarios, and the number of routes is strictly limited. The network topologies following the Barabási-Albert model [20], [21] have scale-free, power-law distributed vertex connectivity with a tail of the degree distribution given by $P_k \propto k^{-(3+0.99)}$. In these "tree-like looking" graphs (cf. Figure 7), there exists only one route between any two vertices, since in the iterative generation process in each step a new vertex is connected to one existing vertex depending on the vertex degree of the existing vertex.

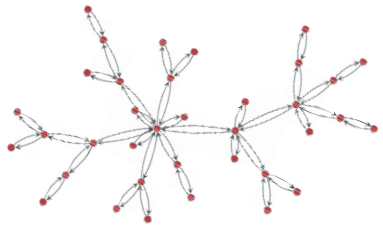

Fig. 7. Example of a scale-free graph with $|\mathcal{V}| = 36$.

The graph-generator we use for network topologies conforming to the $G(V, E)$ Erdős-Rényi model does not guarantee to produce a connected random graph . If the graph is partitioned, we proceed with the largest component. Since networks with this topology are most likely meshed, determining the number of routes is not trivial anymore.

The graph generators produce undirected graphs, which we post-process by replacing each undirected edge by 2 directed edges: a forward edge, as well as a backward edge, to account for the duplex properties of the targeted Ethernet networks. Besides this, the remaining network-related parameters are derived from a $1\,\mathrm{Gbit\,s^{-1}}$ Ethernet network. For $1\,\mathrm{Gbit\,s^{-1}}$ Ethernet networks, it is a reasonable choice to set 1 time-unit in the ILP-formulation to $1\,\mu\mathrm{s}$. The transmission delay for "reasonably-sized" frames in Gigabit Ethernet is given by $d_{\text{trans}} = 368\,\mathrm{B}/1\,\mathrm{GB\,s^{-1}} = 2.944\,\mu\mathrm{s}$. Regarding the applications, we consider a "reasonably-sized" frame to be big enough to hold several timestamps and numerical values (i.e. sensor-data). To compute the propagation delay, we set the link length to $10\,\mathrm{m}$. The propagation delay is given by $d_{\text{prop}} = 10\,\mathrm{m}/\left(\frac{2}{3} \cdot c_{\text{light}}\right) = 0.05\,\mu\mathrm{s}$. A propagation speed of $\frac{2}{3} \cdot c_{\text{light}}$ is in the range of what CAT6-cables provide. The processing delay of the switches is chosen as $d_{\text{proc}} = 5\,\mu\mathrm{s}$. This processing delay can be expected from state-of-the-art Ethernet switches (c.f. measurements in [22]). Consequently, the value of D which we use in our evaluations is set to $D = 8\,\mu\mathrm{s} \approx d_{\text{trans}} + d_{\text{prop}} + d_{\text{proc}}$.

After creating the network topology, we create the sets of parameters for the flows for which we want to compute

the routes and schedules in the network. For each flow f in the set, the index of origin vertex $\mathbf{o}_F[f]$ and the index of the destination vertex $\mathbf{d}_F[f]$ is created by picking 2 values from the set of vertex indices $\mathcal{V}$ with uniform probability. We assume real-time applications to request reservations based on the time it takes to transmit the application specific number of packets / frames. Thus, $\mathbf{r}_F$ is equal to a multiple of t_{trans} in our evaluations. To be more exact, for the reservation $\mathbf{r}_F[f]$ of each flow f, we randomly draw one value from $\{3, 6, 9\}$ with uniform probability. For $\mathbf{p}_F$, the transmission frequencies are in the range from $125\,\text{Hz} = 1$ transmission$/8000\,\mu\text{s}$ to $50\,\text{kHz} = 1$ transmission$/50\,\mu\text{s}$. The end-to-end delay is set depending on the network topology to $\mathbf{l}_F = a \cdot D + 1$ with $a \in \{1, \ldots, |\mathcal{V}|\}$. If not stated differently, we use $\mathbf{l}_F = |\mathcal{V}| \cdot D + 1$, which allows every vertex to be traversed once.

B. Setup

We generate the problem instances for our evaluations with our own Python implementation which relies on the graph-tool library. Our Python code is preprocessing the problem instances, such that they can be digested by the ILP modeling tool. The ILP modeling tool "compiles" the description of the ILP in the respective modeling language and the parameters of a specific problem instance and writes the ILP to a text file in LP-format. Next, this .lp-file is given to the solver which writes the ILP solution (if it exists) to another text file We want to stress, that while all the "data plumbing" is done with our Python tool, we consider only the time of solving process for the respective ILP. The solver is invoked via the vendor-provided binary and the ILP is passed as an .lp-file. We parse the runtime of the solver from the CPLEX log files after executing the solver. This way, we can prepare and post-process the results on different machines without affecting the measurements. For the measurements in this paper, we used the Linux version of IBM ILOG CPLEX Optimization Studio 12.8.0 [23], with OPL as our modeling language (using oplrun to convert it to .mps and .lp) and CPLEX as solver. The "compilation" and solving of the problem instances were performed on a 4-socket compute node equipped with four Intel Xeon E7-4850 v4 CPUs with a nominal clock speed of 2.1 GHz and a total amount of RAM of 1.057 TB. The operating system of the compute node is Linux with kernel version 4.15.15. Even though our compute node can be considered "high-end" with regards to the specifications, we do not expect this to impede the qualitative applicability of our results due to slower machines.

C. Results

We evaluate the effects of various problem properties (graph size, number of flows, topology, transmission frequency, and end-to-end deadline) on the runtime of the solver in the following part.

1) Influence of the Graph Size: From the ILP constraints, it is expected that the number of vertices should influence the runtime of the solver, since the number of constraints depends (via $\mathbf{B}_{VE}$ and $\mathbf{A}_{EE}$) on the size of the graph. We want explore the quantitative relation of solver runtime and graph size, therefore we created sets of problem instances with varying number of vertices, a fixed number of flows $|\mathcal{F}| = 7$, and transmission periods $\mathbf{p}_F[f] \in \{1000, 2000, 4000, 8000\}$. In Figure 8 and Figure 9, we depict the runtimes for these problem instances. For the problem instances in Figure 8, the solving process always completed within the runtime limit $30\,\text{min}$ that was set for the solver. For Figure 9 the runtime

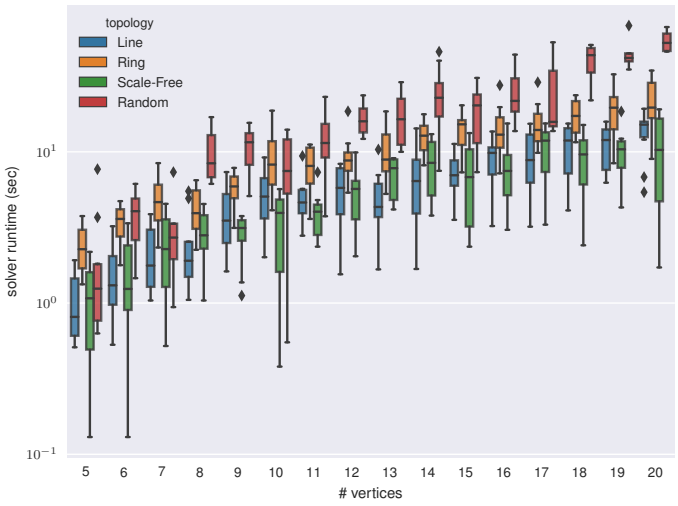

Fig. 8. Box plot of solver runtimes for problem instances with varying number of vertices, fixed $|\mathcal{F}| = 7$, transmission periods $\in \{1000, 2000, 4000, 8000\}$ and runtime limit of $30\,\text{min}$.

limit for the solver was set to $60\,\text{min}$. As can be seen in Table IV, there are a few cases for the random topology, where the solving process did not finish within the time limit.

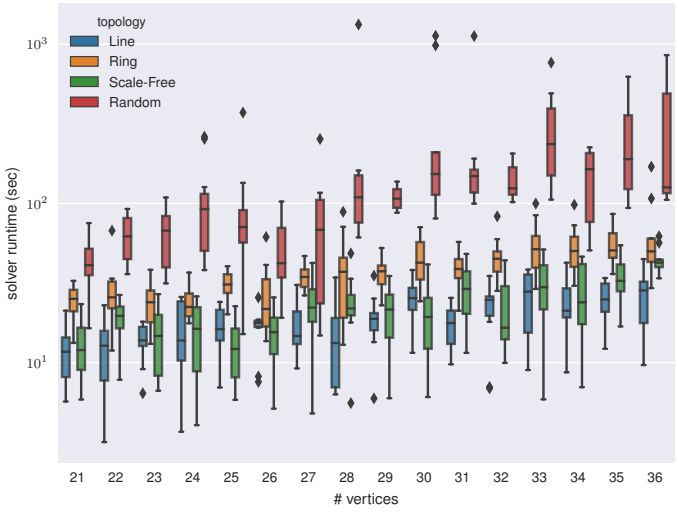

Fig. 9. Box plot of solver runtimes for problem instances with varying number of vertices; parameters are the same in Figure 8, except for number of vertices, and the runtime limit is set to $60\,\text{min}$.

For reference and further comparison, we provide the aggregated runtime results and the number of experiments for all measurements with varying number of vertices in Table III and Table IV.

TABLE III
VARYING NUMBER OF VERTICES, $|\mathcal{F}| = 7$, TRANSMISSION PERIODS $\in \{50, 100, 200, 400, 800\}$ AND $\in \{1000, 2000, 4000, 8000\}$, AND RUNTIME LIMIT OF 30 min.

	Line					Ring					Scale-Free					Random				
Vertices	mean	std	fin	infs	tot	mean	std	fin	infs	tot	mean	std	fin	infs	tot	mean	std	fin	infs	tot
5	1.09	0.52	20	0	20	3.74	2.40	20	0	20	1.37	0.82	20	0	20	2.89	3.15	18	0	18
6	1.58	0.91	20	0	20	5.62	3.21	20	0	20	1.87	1.16	20	0	20	4.82	2.56	24	0	24
7	2.10	0.96	20	0	20	6.37	2.82	20	0	20	2.38	1.38	20	0	20	4.66	4.78	10	0	10
8	2.40	1.53	20	0	20	7.10	4.69	20	0	20	2.66	1.12	20	0	20	10.16	4.33	23	0	23
9	3.72	1.86	20	0	20	8.81	3.91	20	0	20	3.53	1.60	20	0	20	13.39	6.41	25	0	25
10	4.09	2.68	20	0	20	11.18	5.04	20	0	20	3.67	2.04	20	0	20	10.85	6.49	33	0	33
11	4.36	1.77	20	0	20	12.00	6.78	20	0	20	5.10	3.56	20	0	20	12.87	7.44	18	0	18
12	4.86	2.74	20	2	20	13.92	7.20	20	0	20	5.82	2.52	20	0	20	19.09	7.09	19	0	19
13	6.15	3.31	20	0	20	14.22	6.80	20	0	20	6.22	2.64	20	0	20	17.96	7.64	18	0	18
14	7.15	3.70	20	1	20	18.03	9.64	20	0	20	7.42	3.77	20	0	20	25.28	10.42	30	0	30
15	8.47	4.03	20	3	20	17.84	6.93	20	1	20	11.16	6.37	20	0	20	20.61	12.17	24	0	24
16	8.60	3.44	20	1	20	16.30	7.11	20	1	20	8.78	4.06	20	0	20	28.18	15.47	19	1	20
17	10.54	5.02	20	3	20	20.48	9.67	20	1	20	11.65	5.37	20	0	20	29.97	20.96	7	0	7
18	9.07	4.16	20	1	20	22.73	8.69	20	1	20	12.89	9.04	20	0	20	40.43	13.94	20	0	20
19	12.27	6.75	20	2	20	21.89	8.50	20	2	20	13.89	9.79	20	0	20	173.72	359.83	12	0	12
20	14.14	7.91	20	5	20	23.02	8.43	20	1	20	12.48	8.11	20	0	20	222.44	421.66	6	0	6

Each table contains the average runtime (column "mean"), and the standard deviation of the the runtimes (column "std"). The columns labeled "fin" contain the number of problem instances, where the solver finished in time, i.e. either the solver found a solution or declared infeasibility of the problem within the allowed time limit. The columns labeled "infs" contain the number of problem instances, which were infeasible, i.e. where no solution exists. The remaining columns labeled "tot" contains the total number of problem instances that the solver attempted to solve. Subtracting the number of finished problem instances from the number of total problem instances yields the number of unsolved problem instances.

TABLE IV
VARYING NUMBER OF VERTICES, $|\mathcal{F}| = 7$, TRANSMISSION PERIODS $\in \{1000, 2000, 4000, 8000\}$, AND RUNTIME LIMIT OF 60 min.

	Line					Ring					Scale-Free					Random				
Vertices	mean	std	fin	infs	tot	mean	std	fin	infs	tot	mean	std	fin	infs	tot	mean	std	fin	infs	tot
21	11.88	5.10	10	0	10	24.47	5.67	10	0	10	13.16	5.74	10	0	10	41.93	16.31	12	0	12
22	11.84	6.32	10	0	10	29.31	14.83	10	0	10	19.09	5.37	10	0	10	62.66	20.97	8	0	8
23	13.81	3.83	10	0	10	23.97	7.74	10	0	10	15.01	7.25	10	0	10	64.87	29.10	7	0	7
24	15.70	8.27	10	0	10	24.02	6.03	10	0	10	15.68	7.86	10	0	10	102.07	70.18	15	0	15
25	16.72	5.64	10	0	10	31.57	6.42	10	0	10	12.94	5.74	10	0	10	94.52	88.72	13	0	13
26	16.59	5.24	10	0	10	27.50	14.98	10	0	10	15.04	6.79	10	0	10	52.22	25.84	10	0	11
27	17.38	7.09	10	0	10	35.31	7.13	10	0	10	23.08	10.05	10	0	10	80.31	72.07	10	0	10
28	15.70	10.20	10	0	10	39.21	24.98	10	0	10	24.13	11.18	10	0	10	258.12	434.34	8	0	8
29	19.04	7.63	10	0	10	35.95	9.02	10	0	10	20.39	9.24	10	0	10	109.70	22.44	4	0	4
30	24.79	7.77	10	0	10	45.12	15.22	10	0	10	20.99	12.23	10	0	10	339.23	409.29	9	0	9
31	17.12	5.39	10	0	10	38.67	10.35	10	0	10	28.87	11.78	10	0	10	238.52	312.44	10	0	10
32	22.10	8.99	10	0	10	46.43	16.13	10	0	10	22.67	12.87	10	0	10	138.26	35.99	9	0	9
33	24.99	11.63	10	0	10	55.61	22.75	10	0	10	30.29	13.91	10	0	10	313.89	238.65	7	0	7
34	23.48	9.38	10	0	10	53.57	20.63	10	0	10	27.49	14.50	10	0	10	145.48	77.56	6	0	7
35	24.59	7.53	10	0	10	55.90	17.79	10	0	10	33.87	12.15	10	0	10	267.12	198.14	7	0	8
36	26.03	10.95	10	0	10	64.76	42.79	10	0	10	44.53	8.51	10	0	10	361.76	426.08	3	0	4

The measurements with varying number of vertices show that an increasing size of the graph makes the problem more difficult irregardless of the topology. This is no surprise, since the larger graph induces overall more constraints which the solver has to satisfy. Additionally, in a larger graph with random placement of the flow origins and destinations, the routes for the flows can be longer which increases the number of edges where the scheduling constraints have to be applied.

2) Influence of the Number of Flows: In Figure 10 and Figure 11, the runtimes for problem instances with a varying number of flows, a fixed number of vertices $|\mathcal{V}| = 8$, and again low transmission frequencies with the transmission periods $\mathbf{p}_F[f]$ drawn from $\{1000, 2000, 4000, 8000\}$ are depicted. The runtime limit for the problem instances depicted in Figure 10 was set to 30 min, but the solving process was always completed within the given time limit.

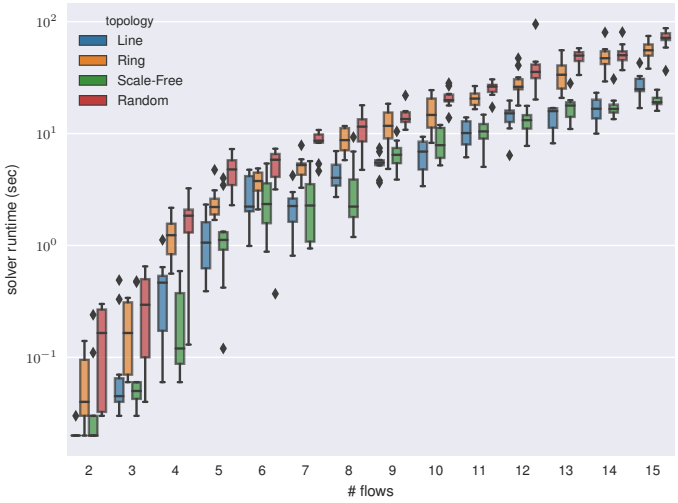

Fig. 10. Box plot of solver runtimes for problem instances with varying number of flows, fixed number of vertices $|\mathcal{V}| = 8$, transmission periods $\in \{1000, 2000, 4000, 8000\}$, runtime limit is set to 30 min.

The runtime limit for the problem instances depicted in Figure 11 was set to 60 min. As can be seen from Table VI, there are two problem instances with random topology that could not be finished before the timeout (21 flows and 30 flows).

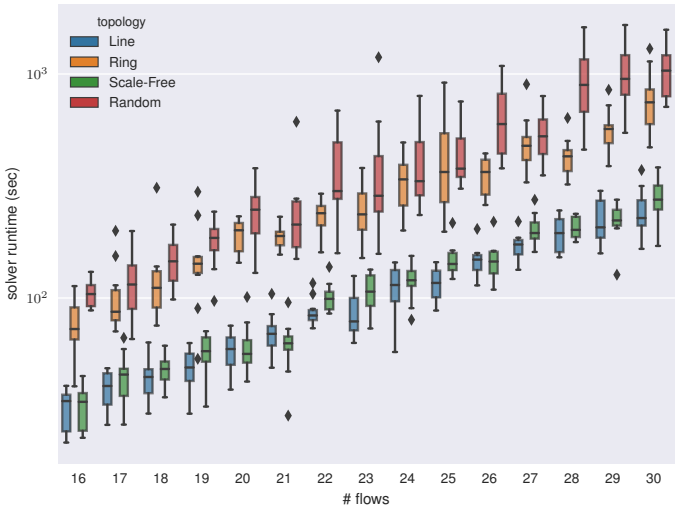

Fig. 11. Box plot of solver runtimes for problem instances with varying number of flows; parameters are the same as in Figure 10 except for the number of flows, and the runtime limit is set to 60 min.

The aggregated data for measurements with varying number of flows are given in Table V and Table VI.

In Figure 12, we applied a non-linear least squares fit of the solver runtimes for the problem instances in Table V to the monomial $x^m \cdot 10^b$ using the Levenberg-Marquardt method and plotted the resulting function. However, if we also fit to the runtime of problem instances with runtime limit of 60 min,

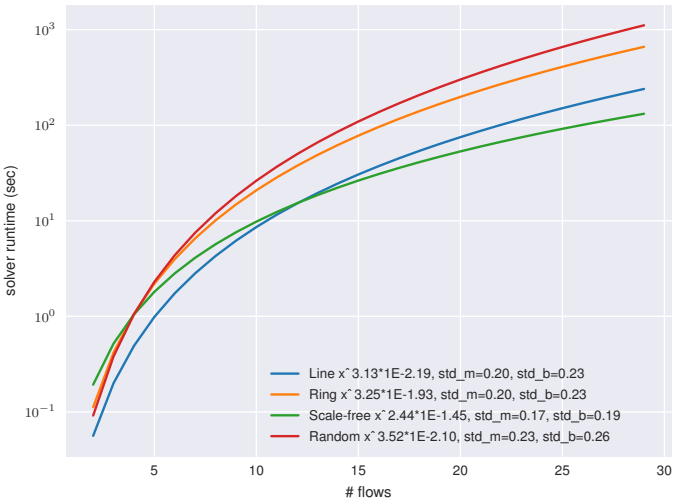

Fig. 12. Fitting of results from Table V to monomial $x^m \cdot 10^b$. The fitted functions reflect the separation of the topologies visible in Figure 10 and Figure 11.

the degree of the monomial rose to ≈ 4, what confirms the non-polynomial scaling behavior of the joint routing and scheduling problem. The same behavior is exhibited by the runtimes of problem instances with varying vertex number.

TABLE V
VARYING NUMBER OF FLOWS, $|\mathcal{V}| = 8$, TRANSMISSION PERIODS $\in \{50, 100, 200, 400, 800\}$ AND $\in \{1000, 2000, 4000, 8000\}$, AND RUNTIME LIMIT OF 30 min.

	Line					Ring					Scale-Free					Random				
Flows	mean	std	fin	infs	tot	mean	std	fin	infs	tot	mean	std	fin	infs	tot	mean	std	fin	infs	tot
2	0.03	0.02	20	0	20	0.06	0.05	20	0	20	0.04	0.05	20	0	20	0.12	0.12	20	0	20
3	0.16	0.20	20	0	20	0.45	0.67	20	0	20	0.13	0.17	20	0	20	0.45	0.52	20	0	20
4	0.45	0.42	20	0	20	1.40	0.87	20	0	20	0.31	0.25	20	0	20	1.71	1.46	20	0	20
5	1.27	0.69	20	0	20	2.84	1.63	20	0	20	1.26	0.96	20	0	20	4.54	2.27	20	0	20
6	2.42	1.30	20	0	20	5.15	3.08	20	0	20	2.05	1.37	20	0	20	6.25	3.94	20	0	20
7	2.86	1.64	20	0	20	7.79	3.21	20	0	20	3.48	2.27	20	0	20	10.13	3.49	20	0	20
8	3.76	1.33	20	0	20	10.95	7.09	20	0	20	5.50	4.01	20	0	20	14.07	4.95	20	0	20
9	6.58	2.65	20	0	20	17.15	8.31	20	0	20	7.26	4.26	20	0	20	20.52	9.23	20	0	20
10	7.14	3.57	20	1	20	22.10	9.11	20	0	20	9.68	3.95	20	0	20	27.24	13.70	20	0	20
11	13.55	6.51	20	0	20	26.27	10.48	20	0	20	11.45	3.62	20	0	20	32.17	8.47	20	0	20
12	16.26	5.22	20	0	20	32.01	11.93	20	0	20	15.41	6.97	20	0	20	54.92	27.61	20	0	20
13	17.72	8.52	20	0	20	49.17	22.29	20	0	20	24.54	15.08	20	0	20	58.74	22.29	20	0	20
14	23.48	10.98	20	1	20	65.04	32.15	20	0	20	20.02	6.16	20	0	20	86.00	51.25	20	0	20
15	31.75	15.69	20	0	20	77.01	34.15	20	0	20	25.21	10.60	20	0	20	112.84	51.85	20	0	20

TABLE VI
VARYING NUMBER OF FLOWS, $|\mathcal{V}| = 8$, TRANSMISSION PERIODS $\in \{1000, 2000, 4000, 8000\}$, AND RUNTIME LIMIT OF 60 min.

	Line					Ring					Scale-Free					Random				
Flows	mean	std	fin	infs	tot	mean	std	fin	infs	tot	mean	std	fin	infs	tot	mean	std	fin	infs	tot
16	31.84	6.68	10	0	10	76.63	21.54	10	0	10	32.99	7.47	10	0	10	104.51	14.60	10	0	10
17	39.08	8.08	10	0	10	103.94	41.51	10	0	10	44.40	12.22	10	0	10	120.71	45.96	10	0	10
18	44.60	10.57	10	0	10	126.88	67.88	10	0	10	47.88	8.20	10	0	10	149.06	36.33	10	0	10
19	49.25	10.30	10	0	10	152.60	68.84	10	0	10	57.01	12.11	10	0	10	179.70	41.53	10	0	10
20	58.21	11.47	10	0	10	191.83	33.25	10	0	10	61.51	16.95	10	0	10	241.87	71.28	10	0	10
21	70.53	15.92	10	0	10	185.89	22.57	10	0	10	62.11	16.91	10	0	10	254.20	141.58	9	0	10
22	87.40	13.19	10	0	10	232.19	40.95	10	0	10	101.42	16.14	10	0	10	370.90	170.34	10	0	10
23	86.84	20.50	10	0	10	247.50	69.91	10	0	10	106.02	22.26	10	0	10	400.43	307.67	10	0	10
24	111.76	26.96	10	0	10	334.76	91.93	10	0	10	119.64	22.37	10	0	10	420.31	202.31	10	0	10
25	116.74	19.19	10	0	10	428.97	224.16	10	0	10	149.12	27.43	10	0	10	449.60	152.85	10	0	10
26	147.02	25.24	10	0	10	354.06	68.11	10	0	10	148.77	30.43	10	0	10	643.39	237.89	10	0	10
27	169.85	25.60	10	0	10	502.09	165.25	10	0	10	203.17	34.09	10	0	10	543.48	134.24	10	0	10
28	196.12	36.78	10	0	10	432.13	92.62	10	0	10	206.94	22.74	10	0	10	937.55	378.95	10	0	10
29	223.86	51.63	10	0	10	564.75	141.50	10	0	10	221.54	40.19	10	0	10	1020.50	356.87	10	0	10
30	245.73	61.84	10	0	10	776.78	269.99	10	0	10	283.55	63.72	10	0	10	1056.51	303.14	9	0	10

3) Influence of the Topology: In Figure 11 (and to a lesser degree in the other box plots) it is visible, that the runtime for ring and random topology is higher than that of the remaining topologies, and the runtime of problem instances with random topology exceeds those of the ring topology. In Figure 13, we show how the average dimension of the ILP (number of constraints and variables) behaves for the problem instances from from Table V and Table VI. Interestingly, even

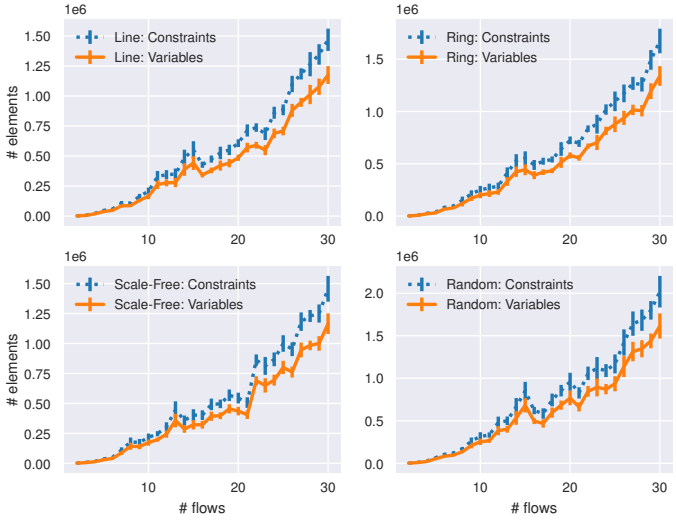

Fig. 13. Average number of ILP constraints and ILP variables for problem instances with varying number of flows from Table V and Table VI.

though only problem instances with random topology result in noticeably (up to 1/4) larger ILPs, than the remaining topologies, problem instances with ring topology take clearly more time to solve in all measurements than those with line and scale-free topology. This indicates, that more routing options increase the solver runtime, since in line and scale-free topologies only one route exists.

4) Influence of the Transmission Frequency: The frequency of the transmissions (the inverse of the period $\mathbf{p}_F$) is different in character from the previously investigated aspects, since it does not influence the number of the constraints, but only the values in the constraints.

In Figure 14, we examine how the frequencies of the transmissions influences the runtime of the solver if the number of vertices is varied. The dashed lines in Figure 14 are the average runtimes for the problem instances with low transmission frequencies (LF), where the transmission period was drawn from $\mathbf{p}_F[f] \in \{1000, 2000, 4000, 8000\}$. These are same problem instances for which the solver runtime was depicted in detail in Figure 8. For the solid lines in Figure 14, we created the same amount of problem instances again with almost the same parameters ($|\mathcal{F}| = 7$, runtime limit of 30 min), except we choose considerably higher transmission frequencies (HF), where the transmission period was drawn from $\mathbf{p}_F[f] \in \{50, 100, 200, 400, 800\}$, and average the observed runtimes. In our measurements, solving the problem instances with higher transmission frequencies takes increasingly more time with increasing number of vertices.

There are few exceptions for line and scale-free graphs for low number of vertices visible in Figure 14 which may be introduced by the random placement of the origin and

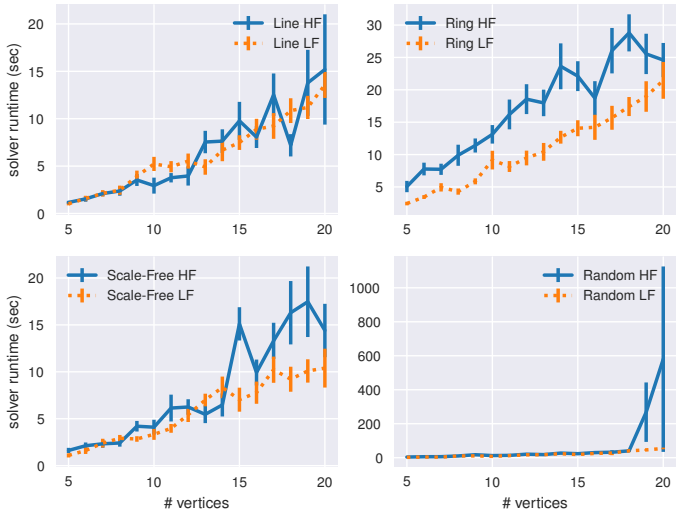

Fig. 14. Comparison of runtimes for problem instances with varying number of vertices, $|\mathcal{F}| = 7$, transmission periods for HF $\in \{50, 100, 200, 400, 800\}$, and for LF $\in \{1000, 2000, 4000, 8000\}$, and runtime limit of 30 min.

destination vertices. Striking is the steep rise of the runtime in HF problem instances with random topology for 19 and 20 vertices, where runtimes up to more than 1000 s were observed. These outliers underline the NP-hard character of the problem, where particular "difficult" problem instances require exorbitant more runtime to solve than the average case.

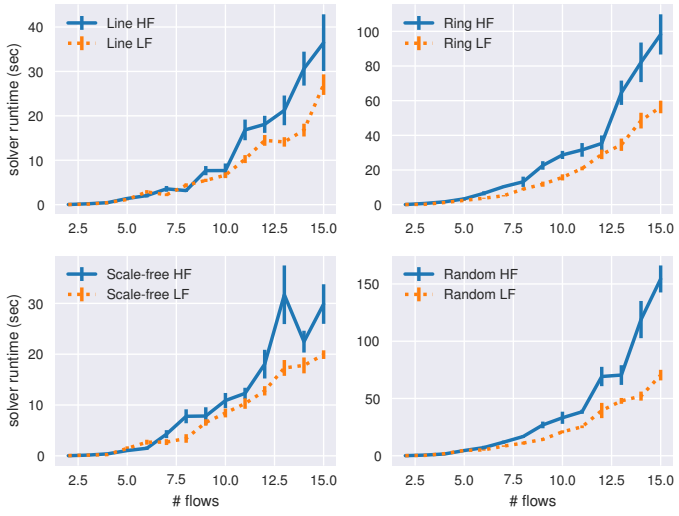

Fig. 15. Comparison of runtimes for problem instances with varying number of flows, $|\mathcal{V}| = 8$, transmission periods for HF $\in \{50, 100, 200, 400, 800\}$, and for LF $\in \{1000, 2000, 4000, 8000\}$, and runtime limit of 30 min.

With the same approach we assess the effect of the transmission frequencies if the number of flows is varied. Again, the dashed line in Figure 15 shows the average runtimes for problem instances with low transmission frequencies ($\mathbf{p}_F[f] \in \{1000, 2000, 4000, 8000\}$), which were presented in Figure 10. The solid line in Figure 15 are the average runtimes for an equally sized set of problem instances with higher transmission frequencies ($\mathbf{p}_F[f] \in \{50, 100, 200, 400, 800\}$)

The same pattern already observed in Figure 14 can be seen again in Figure 15: the solver requires increasingly more time to finish problem instances with higher transmission frequencies compared to problem instances with lower transmission frequencies. Consequently, the transmission frequencies have measurable effects on the time needed for solving a problem instance. Since the range of the reservations is fixed ($\mathbf{r}_F \in \{3, 6, 9\}$), while the period of the transmissions was varied, we effectively changed the fraction of the transmission period that is reserved. That might explain the observations, since —all transmission periods are harmonic— with higher fraction of reservations per cycle there are less possibilities to interleave the reservations of different flows.

VI. Conclusion and Future Work

Joint routing and scheduling problem instances with the number of flows, respectively, the number of vertices in the double digit range result in ILPs with millions of constraints and variables which despite the NP-hardness of the problem can often be solved within reasonable time (seconds to hours) with state-of-the-art equipment. In our evaluations, we made the following three main observations: Firstly, the solver runtime is much more sensitive to an increase in the number of flows than to an increase of the graph size. In our evaluations, additional flows increase the dimension (number of variables and number of constraints) more than increasing the graph size which is one reason for this behavior. Secondly, the solver runtime is influenced by how much of the transmission period is occupied by the transmission, since it is harder to interleave transmissions if larger parts of the transmission periods are already reserved. Thirdly, the graph topology does impact the solver runtime strongly. In our measurements the solver runtime depended massively on how many paths exist between any two nodes, which can be interpreted such that additional routing decisions slow down the solving process, and not simply on the vertex degree.

Besides the development of heuristics, the ILP formulation itself can be extended in future work: The currently "unused" objective function of the ILP can influence the distribution of the reservations on the edges, e.g., to load all edges equally, to minimize end-to-end-latency, to minimize shared switches, to optimize the solution for specific scenarios. Considering artifacts of real-world networks (i.e. synchronization errors, non-constant delays induced by operating conditions or design choices in the switch design) in the ILP, e.g., by expanding the required reservation per transmission depending on the (hop-count) distance from the origin vertex has practical relevance. And with these imperfections, it will be more complicated to preserve transmission order in switches with FIFO queuing, and even more complicated to allow queuing in multiple FIFO queues and the resulting (limited) reordering of transmissions. Finally, an additional degree of freedom can be gained by allowing for dynamic routing, i.e. different transmissions of an application are forwarded along different paths depending on the current time.

ACKNOWLEDGMENT

This work was supported by the German Research Foundation (DFG) under the research grant "Integrated Controller Design Methods and Communication Services for Networked Control Systems (NCS)" (RO 1086/20-1).

REFERENCES

[1] LAN/MAN Standards Committee, "IEEE Standard for Ethernet," *IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012)*, no. IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012), pp. 1–4017, Mar. 2016.

[2] LAN/MAN Standards Committee of the IEEE Computer Society, "IEEE Std 802.1Q™ - 2014 - IEEE Standard for Local and metropolitan area networks— Bridges and Bridged Networks," Dec. 2014.

[3] IEEE Computer Society, "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–57, Mar. 2016.

[4] IEEE Instrumentation and Measurement Society, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–300, Jul. 2008.

[5] IEEE Computer Society, "IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks," *IEEE Std 802.1AS-2011*, pp. 1–292, Mar. 2011.

[6] W. Steiner, "An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks," in *2010 31st IEEE Real-Time Systems Symposium*, Nov. 2010, pp. 375–384.

[7] ——, "Synthesis of Static Communication Schedules for Mixed-Criticality Systems," in *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, Mar. 2011, pp. 11–18.

[8] F. Dürr and N. G. Nayak, "No-wait Packet Scheduling for IEEE Time-sensitive Networks (TSN)," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, ser. RTNS '16, 2016, pp. 203–212.

[9] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, ser. RTNS '16, 2016, pp. 183–192.

[10] P. Pop, M. L. Raagaard, S. S. Craciunas, and W. Steiner, "Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks," *IET Cyber-Physical Systems: Theory Applications*, vol. 1, no. 1, pp. 86–94, 2016.

[11] F. Pozo, G. Rodriguez-Navas, W. Steiner, and H. Hansson, "Period-Aware Segmented Synthesis of Schedules for Multi-hop Time-Triggered Networks," in *2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Aug. 2016, pp. 170–175.

[12] N. G. Nayak, F. Dürr, and K. Rothermel, "Time-sensitive Software-defined Network (TSSDN) for Real-time Applications," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, ser. RTNS '16, 2016, pp. 193–202.

[13] ——, "Incremental Flow Scheduling and Routing in Time-Sensitive Software-Defined Networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2066–2075, May 2018.

[14] F. Smirnov, M. Glaß, F. Reimann, and J. Teich, "Optimizing Message Routing and Scheduling in Automotive Mixed-Criticality Time-Triggered Networks," in *Proceedings of the 54th Annual Design Automation Conference 2017*, ser. DAC '17, 2017, pp. 48:1–48:6.

[15] E. Schweissguth, P. Danielis, D. Timmermann, H. Parzyjegla, and G. Mühl, "ILP-based Joint Routing and Scheduling for Time-triggered Networks," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, ser. RTNS '17, 2017, pp. 8–17.

[16] P. Belotti, P. Bonami, M. Fischetti, A. Lodi, M. Monaci, A. Nogales-Gómez, and D. Salvagnin, "On handling indicator constraints in mixed integer programming," *Computational Optimization and Applications*, vol. 65, no. 3, pp. 545–566, Dec. 2016.

[17] T. Koch, "Rapid Mathematical Programming," Ph.D. dissertation, Technische Universität Berlin, 2004.

[18] M. R. Garey and D. S. Johnson, *Computers and Intractability*, ser. a guide to the theory of NP-completeness. New York, NY: Freeman, 1979.

[19] T. P. Peixoto, "The graph-tool python library," May 2017.

[20] Price Derek De Solla, "A general theory of bibliometric and other cumulative advantage processes," *Journal of the American Society for Information Science*, vol. 27, no. 5, pp. 292–306, Mar. 2007.

[21] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, no. 5439, pp. 509–512, Oct. 1999.

[22] F. Dürr and T. Kohler, "Comparing the forwarding latency of OpenFlow hardware and software switches," Institute of Parallel and Distributed Systems, University of Stuttgart, Stuttgart, Tech. Rep. TR 2014/04, 2014.

[23] IBM, "IBM ILOG CPLEX Optimization Studio — IBM," https://www.ibm.com/products/ilog-cplex-optimization-studio.