Scheduling with No-Wait Decisions in Heavily Loaded TSN Networks*

Jesper Vines[1,2][0000−0002−1228−9759], Hannes Uppman[2][0000−0003−0003−9127], Mikaela Lindberg[2], and Elina Rönnberg[1][0000−0002−2081−2888]

[1] Linköping University, Sweden
jesper.vines@liu.se, elina.ronnberg@liu.se
[2] Saab Aeronautics, Sweden

Keywords: Logic-based Benders Decomposition · Scheduling · Time Sensitive Networking · Avionics

1 Introduction

Time Sensitive Networking (TSN), standardized as IEEE 802.1Q, is a collection of standards that extend the functionality of Ethernet. This technology can be used to guarantee high reliability and bounded latency, which is crucial when working with a hard real-time system, like an avionics system (the electronic system in an aircraft). Our work concerns joint routing and scheduling of time-triggered messages in TSN networks, a problem which is known to be computationally challenging. In practice, routing is therefore often done before the scheduling of traffic. However, in heavily loaded networks, particular routing decisions can render the resulting scheduling problem infeasible, even if the complete problem has a solution. Vlk, Hanzálek and Tang [1] present a Logic-based Benders Decomposition (LBBD) approach for this problem, in which the master problem routes the traffic and the scheduling is handled by the subproblem. This captures the way this problem is often handled in practice, while still protecting against unfortunate routing decisions. This approach can, however, fail at solving larger problem instances, since the resulting scheduling subproblem can become intractable. We therefore propose an extension to this approach, where more decisions are taken in the master problem. This makes it possible to make more iterations and solve more challenging instances. One of our extensions, presented here, uses a technique that is often applied in practice to reduce the complexity of the scheduling: *no-wait scheduling*. As the name implies, no-wait scheduling means that messages are not allowed to wait in switches before being transferred. We incorporate this decision into the LBBD framework, so that we maintain our guarantees while simultaneously making the subproblem easier to solve. This extension allows us to solve a larger number of problem instances and is a first step towards designing a LBBD scheme that can solve even more challenging instances.

* This work was funded by Vinnova within the National Aeronautical Research Program (NFFP8), project 2024-01259.

2 Problem Formulation and Modelling

In this section we present the problem and the decomposed model (CP2), proposed in [1], which we refer to as the baseline model. The network is modeled as a directed graph $(\mathcal{V}, \mathcal{E})$, with the set of nodes $\mathcal{V}$ representing the end systems and switches, and the set of edges $\mathcal{E}$ representing the network links. All network connections allow for full-duplex communication, so $\mathcal{E}$ is symmetric.

TSN devices need to be time-synchronized, and we denote the maximal difference between two clocks in the network by δ. Each edge $(a, b) \in \mathcal{E}$ has a propagation delay δ_{ab}, while each node $a \in \mathcal{V}$ introduces a switching delay δ_a. The ports in TSN switches contain up to eight first-in-first-out queues. We denote the number of queues available for scheduled traffic in the egress port to link $(a, b) \in \mathcal{E}$ by q_{ab} and, for convenience, the set of indices $\{1, \ldots, q_{ab}\}$ by $\mathcal{Q}_{ab}$.

A *stream* is a periodic data transfer from a source end system, known as a *talker*, to a destination end system, known as a *listener*. Let $\mathcal{S}$ be the set of all streams. For each stream $i \in \mathcal{S}$, we denote the talker by t_i, the listener by l_i and the time it takes to transfer stream i over a link $(a, b) \in \mathcal{E}$ by L_{iab}. The period of stream $i \in \mathcal{S}$ is denoted by T_i, and the hyper period T is defined as the least common multiple of all stream periods. It follows that the system is periodic with period T, and that it is sufficient to construct a schedule of this length. During an interval of length T, stream $i \in \mathcal{S}$ must occur $k = T/T_i$ times. These occurrences are referred to as stream *instances*. Let $\mathcal{K}_i = \{0, 1, \ldots, T/T_i - 1\}$ denote the set of instance indices for stream $i \in \mathcal{S}$. For convenience, we also use $\mathcal{K}_i^+ = \mathcal{K}_i \setminus \{0\}$. For the first instance of stream $i \in \mathcal{S}$, the earliest time a talker can start transferring data is known as the stream's *release time*, denoted by r_i, and the latest time the listener must obtain the data is known as its *deadline*, denoted by d_i. It is assumed that $0 \leq r_i \leq d_i \leq T_i, i \in \mathcal{S}$.

The model is decomposed into a MIP master problem, aiming to find a shortest path routing, and a feasibility CP scheduling subproblem. In the master problem, the binary variable r_{iab} is used to indicate if stream $i \in \mathcal{S}$ is routed over edge $(a, b) \in \mathcal{E}$ or not. Correct routing is enforced by the constraints

$$\sum_{b \in \delta^+(t_i)} r_{it_ib} = \sum_{a \in \delta^-(l_i)} r_{ial_i} = 1 \qquad i \in \mathcal{S}, \tag{1}$$

$$\sum_{b \in \delta^-(t_i)} r_{it_ib} = \sum_{a \in \delta^+(l_i)} r_{ibl_i} = 0 \qquad i \in \mathcal{S}, \tag{2}$$

$$\sum_{a \in \delta^-(b)} r_{iab} = \sum_{c \in \delta^+(b)} r_{ibc} \leq 1 \qquad i \in \mathcal{S}, b \in \mathcal{V} : b \neq t_i, b \neq l_i, \tag{3}$$

where $\delta^+(b)$ and $\delta^-(b)$ are the sets of outgoing and incoming edges from $b \in \mathcal{V}$. Later iterations of the master problem also include no-good cuts of the form

$$\sum_{(i,a,b) \in \mathcal{C}} r_{iab} \leq |\mathcal{C}| - 1. \tag{4}$$

Here $\mathcal{C}$ is a minimal (or close to minimal) subset of $\mathcal{S} \times \mathcal{V} \times \mathcal{V}$ such that no solution to the full problem exists where $r_{iab} = 1$ for every $(i, a, b) \in \mathcal{C}$. We find such

sets $\mathcal{C}$ by computation of conflicts (infeasible submodels) for the earlier, infeasible subproblems. We denote the value of the variable r_{iab} in the latest master problem solution by $\hat{r}_{iab}$. For every stream $i \in \mathcal{S}$, we define the subgraph $(\mathcal{V}_i, \mathcal{E}_i)$ of $(\mathcal{V}, \mathcal{E})$, where $\mathcal{E}_i = \{(a,b) \in \mathcal{E} : \hat{r}_{iab} = 1\}$ and $\mathcal{V}_i = \{a \in \mathcal{V} : \exists b \text{ s.t. } (a,b) \in \mathcal{E}_i \text{ or } (b,a) \in \mathcal{E}_i\}$.

In the subproblem, interval variables are used to model when the streams occupy the links or queues in the network. The interval I_{iabk} indicates when instance $k \in \mathcal{K}_i$ of stream $i \in \mathcal{S}$ occupies link $(a,b) \in \mathcal{E}_i$. The interval Q_{ib} indicates when the first instance of stream $i \in \mathcal{S}$ occupies a queue in a port in switch $b \in \mathcal{V}_i$. We further introduce the optional interval variable $\bar{Q}_{ibkq}$, that can either be present or absent in a solution, which indicates when instance $k \in \mathcal{K}_i$ of stream $i \in \mathcal{S}$ occupies queue $q \in \mathcal{Q}_{bc}$. The connection between these variables is modeled by the Alternative and PresenceOf constraints (5)–(6), which ensure that exactly one of $\{\bar{Q}_{ib0q}, q \in \mathcal{Q}_{bc}\}$ is present, and shares start and end times with Q_{ib}, and that this queue is chosen for all instances of a stream. The NoOverlap constraints (7)–(8) ensures that the interval variables present in the constraint sets do not overlap.

$$\text{Alternative}(Q_{ib}, \{\bar{Q}_{ib0q} : q \in \mathcal{Q}_{bc}\}) \qquad i \in \mathcal{S}, (b,c) \in \mathcal{E}_i, \tag{5}$$

$$\text{PresenceOf}(\bar{Q}_{ibkq}) = \text{PresenceOf}(\bar{Q}_{ib(k-1)q}) \qquad i \in \mathcal{S}, (b,c) \in \mathcal{E}_i, q \in \mathcal{Q}_{bc}, k \in \mathcal{K}_i^+ : b \neq t_i, \tag{6}$$

$$\text{NoOverlap}(\{I_{iabk} : i \in \mathcal{S}, k \in \mathcal{K}_i \text{ s.t. } (a,b) \in \mathcal{E}_i\}) \qquad (a,b) \in \mathcal{E}, \tag{7}$$

$$\text{NoOverlap}(\{\bar{Q}_{ibkq} : i \in \mathcal{S}, k \in \mathcal{K}_i \text{ s.t. } (b,c) \in \mathcal{E}_i\})\,(b,c) \in \mathcal{E}, q \in \mathcal{Q}_{bc}. \tag{8}$$

The scheduling constraints (9)–(13) in the CP subproblem enforce release times, deadlines, path ordering, and zero latency variation (*jitter*). Since the transmission interval over a link directly influences when the stream is placed in the following queue, the relations between the queue and link occupation intervals (14)–(17) are also enforced. For every stream $i \in \mathcal{S}$, the constraints are

$$\text{start}(I_{iab0}) \geq r_i \qquad (a,b) \in \mathcal{E}_i, \tag{9}$$

$$\text{end}(I_{iab0}) \leq d_i \qquad (a,b) \in \mathcal{E}_i, \tag{10}$$

$$\text{start}(I_{ibc0}) \geq \text{end}(I_{iab0}) + \delta_{ab} + \delta_b + \delta \qquad (a,b),(b,c) \in \mathcal{E}_i, \tag{11}$$

$$\text{start}(I_{iabk}) = \text{start}(I_{iab(k-1)}) + T_i \qquad (a,b) \in \mathcal{E}_i, k \in \mathcal{K}_i^+, \tag{12}$$

$$\text{size}(I_{iabk}) = L_{iab} \qquad (a,b) \in \mathcal{E}_i, k \in \mathcal{K}_i, \tag{13}$$

$$\text{start}(Q_{ib}) = \text{end}(I_{iab0}) + \delta_{ab} - \delta \qquad (a,b) \in \mathcal{E}_i : b \neq l_i, \tag{14}$$

$$\text{end}(Q_{ib}) = \text{start}(I_{ibc0}) \qquad (b,c) \in \mathcal{E}_i : b \neq t_i, \tag{15}$$

$$\text{start}(\bar{Q}_{ibkq}) = \text{start}(\bar{Q}_{ib(k-1)q}) + T_i \qquad (b,c) \in \mathcal{E}_i, q \in \mathcal{Q}_{bc}, k \in \mathcal{K}_i^+ : b \neq t_i, \tag{16}$$

$$\text{end}(\bar{Q}_{ibkq}) = \text{end}(\bar{Q}_{ib(k-1)q}) + T_i \qquad (b,c) \in \mathcal{E}_i, q \in \mathcal{Q}_{bc}, k \in \mathcal{K}_i^+ : b \neq t_i. \tag{17}$$

3 Further Developments of the Decomposition Scheme

The master problem is often easy to solve, since the networks are relatively small. However, solving a complete scheduling problem in the subproblem can become

challenging for larger problem instances, e.g. with a greater number of streams. This problem is discussed briefly in [1], where it is noted that the method often fails to complete more than a single LBBD iteration—either the subproblem is solved in the first iteration, or the solver times out. This can lead to the problem mentioned in the introduction, where an unfortunate initial routing decision can make it very hard to determine feasibility of the model. In order to make larger problem instances benefit from the LBBD framework, the subproblem needs to be easier to solve.

One way of achieving this is by taking additional decisions in the master problem, to impose greater restrictions on the subproblem. One such decision, that significantly impacts the tractability of the subproblem, is the no-wait decision. For each $i \in \mathcal{S}$, we add the binary decision variable n_i to the master model, indicating if stream i is not allowed to wait in the queue of any switch it is routed through. The objective function tries to maximize the use of no-wait, as well as minimize the path length. We denote the value of n_i in the last master iteration by $\hat{n}_i$, and the constraint

$$\text{start}(I_{ibc0}) = \text{end}(I_{iab0}) + \delta_b + \delta \qquad i \in \mathcal{S} : \hat{n}_i = 1, \tag{18}$$

is added to the subproblem. The no-good cuts take the shape

$$\sum_{(i,a,b)\in\mathcal{C}_r} r_{iab} + \sum_{i\in\mathcal{C}_n} n_i \leq |\mathcal{C}_r| + |\mathcal{C}_n| - 1, \tag{19}$$

where $\mathcal{C}_r$ and $\mathcal{C}_n$ are small sets such that no solution to the full problem exists where $r_{iab} = 1$ for $(i, a, b) \in \mathcal{C}_r$ and $n_i = 1$ for $i \in \mathcal{C}_n$. These sets are again derived from earlier, infeasible subproblems.

For these no-wait streams we only take a single scheduling decision, the send time from the talker, instead of needing a scheduling decision per link. This makes finding a solution for the subproblem, or determining infeasibility and generating cuts, easier. To evaluate our extended approach with the no-wait decisions in the master problem, we have made preliminary computational experiments using a set of 150 instances for larger networks (96 nodes, 220 edges and between 30 and 900 streams). We compare the results from the baseline model from [1] to our extended approach. For the former, 65 instances are solved and for the latter, 72 instances are solved. We can also see that, in the instances that are not solved within the relatively short timeout, we manage to increase the number of LBBD iterations significantly. This means a larger amount of routing options for the streams are evaluated, which increases the chances of finding a solution to the original problem.

Ongoing and future work include further strengthening of the master problem and development of additional acceleration strategies.

References

1. Vlk, M., Hanzálek, Z., Tang, S.: Constraint programming approaches to joint routing and scheduling in time-sensitive networks. Computers & Industrial Engineering **157**, 107317 (2021)