TSN Incremental Routing and Scheduling through parallel MILP computation

Álex Gracia[1], Juan Segarra[1], José Luis Briz[1], Alitzel Torres[1,2], Antonio Ramírez[2], Héctor Blanco[3]

[1] Affiliation: Grupo de Investigación en Arquitectura de Computadores (gaZ)
Instituto de Investigación en Ingeniería de Aragón (I3A)
Universidad de Zaragoza, Mariano Esquillor s/n, 50018, Zaragoza, Spain.
Tel. +34-976762707, e-mail: alex.gracia@unizar.es
[2]Affiliation: CINVESTAV Unidad Guadalajara
[3]Affiliation: Intel Deutschland GmbH

Abstract

Time-Sensitive Networking (TSN) ensures deterministic communication using IEEE 802.1Qbv's Time Aware Shaper (TAS). Generating Gate Control Lists (GCL) is computationally heavy for dynamic systems. We propose an incremental MILP framework for joint routing and scheduling. It finds solutions faster and achieves complete schedules where monolithic methods fail

Introduction

Time-Sensitive Networking (TSN) is a set of IEEE standards that enables deterministic communication over Ethernet and wireless networks, ensuring real-time requirements. The IEEE 802.1Qbv standard delineates a time-triggered communication paradigm, incorporating a Time-Aware Shaper (TAS) that governs the selection of frames at egress queues in accordance with a predetermined schedule. The scheduling problem in TSN is critical to ensuring deterministic transmission of flows. In addition to the scheduling issue, a further significant challenge in TSN is the determination of the optimal routing for each flow. The trajectory of a time-sensitive flow through the network exerts a direct influence on the feasibility of the schedule and the overall performance of the system.

Proposed Approach

We present two different proposals. In the first one, we formulate the MILP model by generating the set of constraints defined in [1], but instead of applying them to a predetermined path, we apply them to each pair (s_{i+1}, j), where s_{i+1} is the new flow entering the system, and $j \in C$ corresponds to each of the previously enumerated candidate paths. In other words, we replicate the same flow s_{i+1} across all its possible paths. This implies that the model's complexity scales with both the number of possible paths and the number of frames per flow.

In the second proposal, we compute all possible paths that a flow can take and generate a separate MILP model for each one, using the same set of constraints from [1]. Once all MILP models are solved, the best solution is selected, and the corresponding path is assigned to the flow. Although the number of models to be solved increases with the number of candidate paths, these models are less complex, independent, and can be solved in parallel.

With the objective of evaluating both our approaches, we use the use case presented in [2] as a reference, which corresponds to an avionics network topology. The network topology consists of fourteen end stations and five bridges connected in a mesh topology, with each end station linked to one of the bridges. The use case includes 241 flows, each one of them requiring the scheduling of between one and 32 frames within the hyperperiod.

Although predefined routes are provided in the original use case, we consider only the source and destination nodes of each flow, allowing our model to determine the optimal path. While the challenge includes additional components such as the Credit-Based Shaper, our evaluation focuses exclusively on scheduling using the TAS.

Experimental Results

The experiments are conducted on anIntel® Xeon® Gold 5120 CPU running at 2.20 GHz, with 56 cores (Skylake architecture). The MILP models are solved using the CBC[3] solver, version 2.10.12.

In the first approach, all processor cores are utilized in parallel to solve a single, complex MILP model. In contrast, the second approach assigns each MILP instance to a separate core, allowing them to be

computed independently. It is important to note that this evaluation does not consider scenarios where the number of flows exceeds the number of available paths.

A one-hour timeout is set for the MILP model in the first approach. In the second one, this time is evenly divided among the individual tasks. The solver may find an optimal solution, a feasible (but not necessarily optimal) solution, no solution within the time limit, or determine that the flow is unschedulable. In the parallel version, some routes may be solved within the time limit while others are not; however, longer routes typically need more computation and are less likely to be solved or selected.

Figure 1 shows the obtained results for our different approaches. The parallel model successfully schedules 206 out of 241 optimally, while the monolithic model only manages 27 out of 241.. We can also compare the results obtained by both approaches. Out of the 50 solutions found within the time limit that are not optimal, 37 match the results of the second approach. For the remaining thirteen solutions, the second approach achieves an average improvement of 42.98 % in the value obtained by the objective function.

Conclusions

This work demonstrates the viability of parallel independent MILP approaches for joint routing and scheduling in TSN networks. While optimality guarantees remain challenging for large-scale instances, the parallel solution enables better trade-offs between solution quality and computation time than the monolithic approach, which is critical for dynamic systems requiring incremental updates.

Future work will investigate the impact of solver selection and configuration more deeply. This includes a systematic exploration of CBC's highly parameterizable settings to reduce analysis time, and evaluating the performance of commercial solvers such as Gurobi [4].

Acknowledgements

This work was supported by MCIN/AEI/10.13039/501100011033 (grant PID2022-136454NB-C22), and by Dept. of Science, University and Knowledge Society, Government of Aragón (grant to reference research group T58_23R), Spain. A. Torres was supported by SECIHTI (grant 1141966), Mexico.

References

[1]. TORRES-MACÍAS, A.G. et al. Optimal and Fast IEEE 802.1Qbv Incremental Scheduling. *Manuscript under review*, 2025.

[2]. BOYER, Marc and HENIA, Rafik. *Industrial challenge: Embedded reconfiguration of TSN*. Working paper. July 2024.

[3]. COIN-OR. *CBC: COIN-OR Branch and Cut* (Version 2.10.12) [software]. August 2024. DOI: 10.5281/zenodo.13347261.

[4]. GUROBI OPTIMIZATION, LLC. *Gurobi Optimizer Reference Manual*. 2024. Avalible in: https://www.gurobi.com.

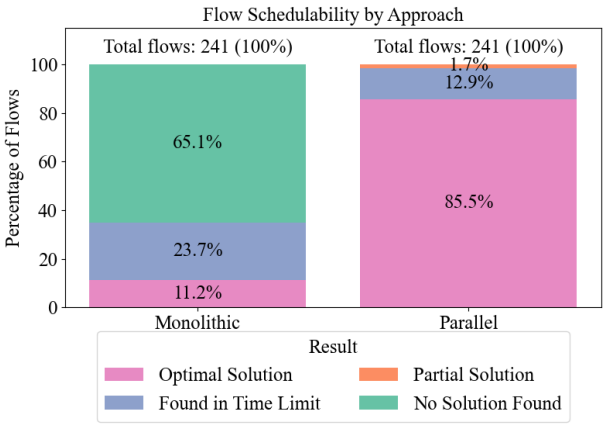

Figure 1: Flow Schedulability Results by Approach