Embedded reconfiguration of TSN: Dual reconfiguration with dropping and reclaiming

Álex Gracia, Alitzel G. Torres-Macías, *Juan Segarra*, José L. Briz, Antonio Ramírez-Treviño, Héctor Blanco-Alcaine

Universidad Zaragoza 1542

Cinvestav

intel

37th Euromicro Conference on Real-Time Systems
Brussels, July 8-11, 2025

GOBIERNO DE ESPAÑA

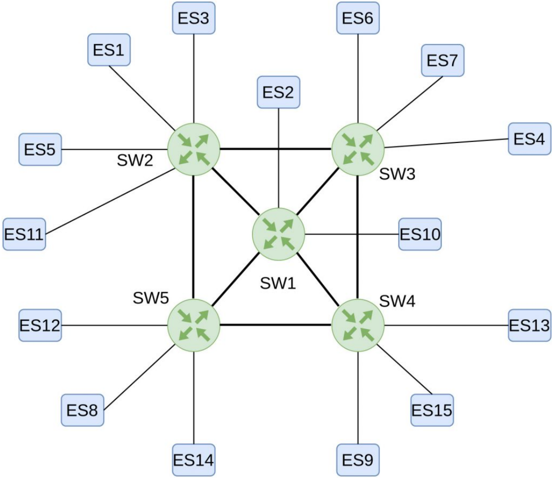
MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES

AGENCIA ESTATAL DE INVESTIGACIÓN

Outline

1. Introduction
    1.1. Time Aware Shaper
2. Proposed procedure
    2.1. Scheduling by MILP models
    2.2. Incremental stage
    2.3. Global stage
3. Experiments
    3.1. Rescheduling times
4. Conclusions and future work

1 Introduction



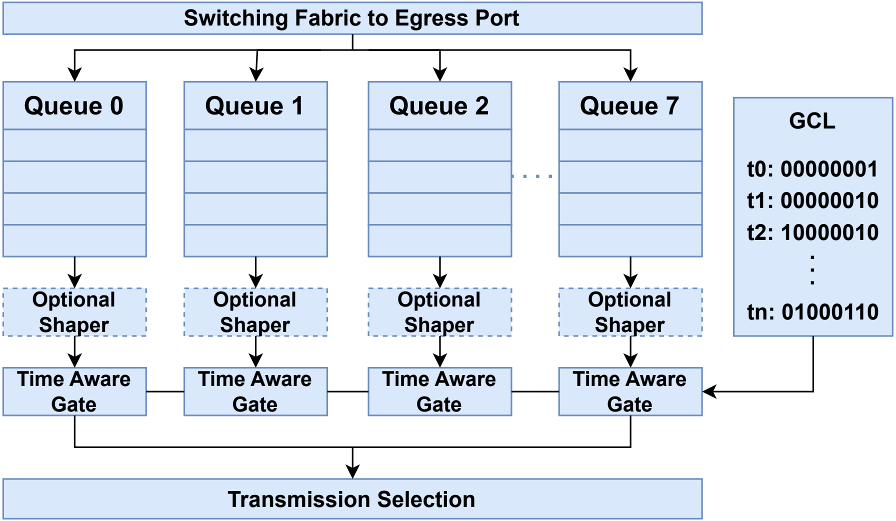
Challenge:

- 241 mixed criticality streams
- 8 potentially failing links
- Reroute and reschedule upon link failures
  - NP-complete problem

Tested solution:

- Mixed Integer Linear Programming models for scheduling
- Incremental + global reconfiguration
- Time Aware Shaper for all streams

1.1 Time Aware Shaper



Switching Fabric to Egress Port

Queue 0

Queue 1

Queue 2

Queue 7

Optional Shaper

Optional Shaper

Optional Shaper

Optional Shaper

Time Aware Gate

Time Aware Gate

Time Aware Gate

Time Aware Gate

GCL

t0: 00000001
t1: 00000010
t2: 10000010
⋮
tn: 01000110

Transmission Selection

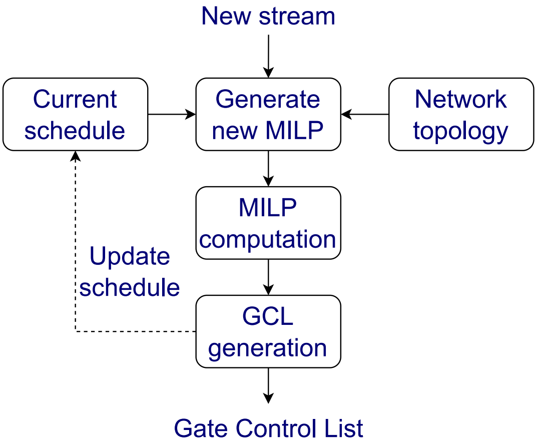
2 Proposed procedure

1. Detect faulty link
2. Get ordered sublist of disposable (affected) schedules and purge them
   - DROP policy until reschedule
3. Incremental stage: schedule and deploy stream-by-stream
   - Schedule streams one-by-one on an existing schedule
   - Faster, one-by-one optimal (no heuristics)

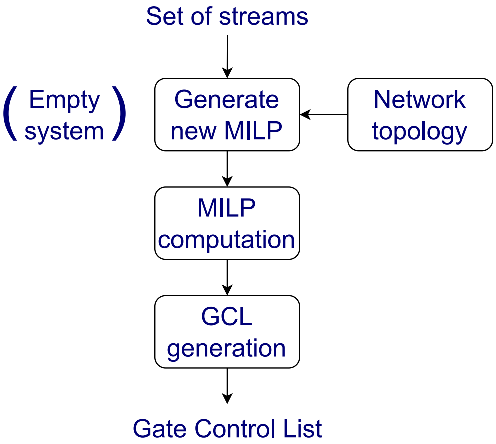
   [1] A. Galilea Torres-Macías et al. Optimal and Fast IEEE802.1Qbv Incremental Scheduling. Submitted to Computer Networks.
4. Global stage, if needed (not needed for the use case)
   - Schedule several streams together on an empty system
   - Slower, better schedulability, globaly optimized, with heuristics

   [2] A. Galilea Torres-Macías et al. Fast IEEE802.1Qbv gate scheduling through integer linear programming. IEEE Access, 12:111239–111250, 2024.

2.1 Scheduling by MILP models



(a) Incremental scheduling [1]



(b) Global scheduling [2]

2.2 Incremental stage

Process streams one by one:

```
for all $stream$ in disposableStreamList do
    $pathList \leftarrow$ Get ordered list of possible paths to reschedule $stream$
    for all $path$ in $pathList$, shortest first do        // Single iteration suffices!
        Generate MILP model [1] to schedule $stream$ on $path$
        Solve model
        if schedulable then
            Deploy schedule for $stream$
            Exit loop on pathList
```

2.3 Global stage

Process set of streams together; maximize its size with binary search:

```
while not satisfied with the schedule do
    Generate MILP model [2] to schedule a set of streams (shortest paths)
    Solve model
    if schedulable then
        Purge system and Deploy
        if scheduled set of streams contains all streams then
            The system is optimized; exit
        else
            Add streams to the set of streams
    else
        Remove streams from the set of streams
    if the set of streams has already been tested then
        The system cannot be further improved; exit
```

3 Experiments

- ➤ Intel Xeon Gold 5120 CPU: 28 cores (56 total threads), released on 2017
- ➤ Frame isolation supported but not addressed (TC7 streams isolated)
- ➤ 1 ms frame processing time; 0.5 ms clock skew

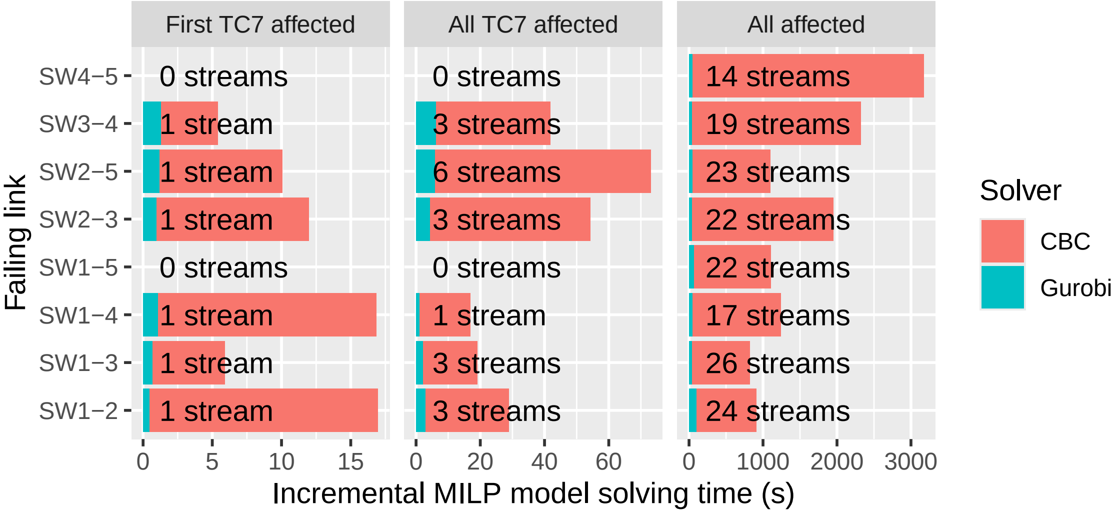
Experiment 1. Network dimmensioning (consecutive link failures)

- ✓ Adequate: all reschedulable with 3 failing links

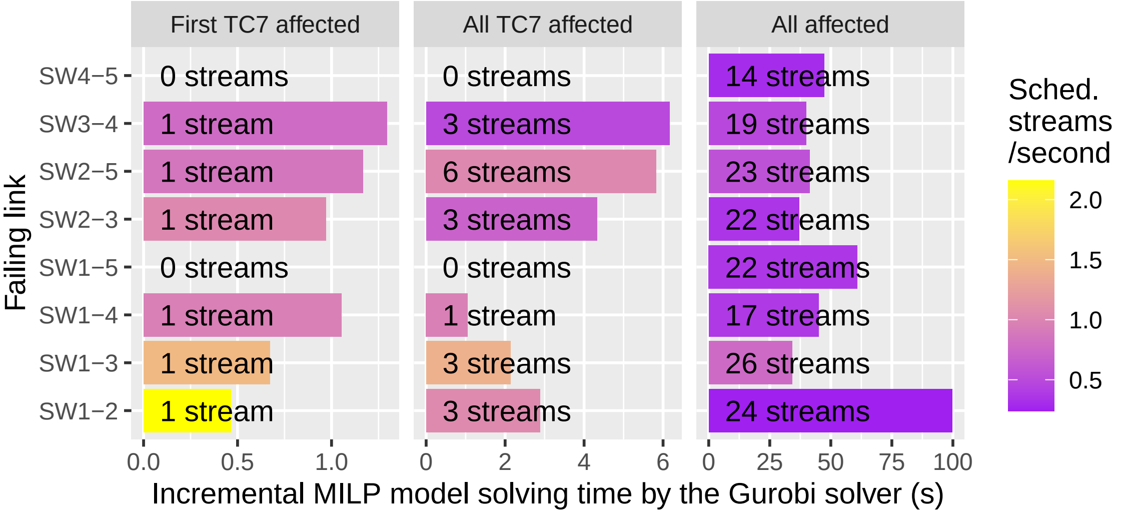
Experiment 2. Rescheduling times (single link failure)

- ✗ Remove streams affected by link failure and lower priority working ones
  - ➤ Rescheduling working schedules is not worth
  - ✓ Reschedule streams affected by link failure only
- ✗ Get valid schedules (not necessarily optimal)
  - ➤ Faster for the first streams, but globaly slower
  - ✓ Get optimal schedules always

3.1 Rescheduling times (single link failure)



3.1 Rescheduling times (single link failure) (II)



4 Conclusions and future work

For the considered use case:

- Recovery time (Gurobi): from 0.5 s (TC7) to 1-2 min. (all streams)
- Incremental (one-by-one) rescheduling [1] suffices
- Hints for rescheduling:
  - Rescheduling working low-criticality streams is globally slower
  - Computing optimal schedules is globaly faster than just valid ones

Work in progress:

- Speed-up solving times through heuristics
- Fault tolerance by CNC-managed frame replication
- Alternative implementations of the Gate Control List of TAS

Embedded reconfiguration of TSN: Dual reconfiguration with dropping and reclaiming

**Álex Gracia, Alitzel G. Torres-Macías, *Juan Segarra*, José L. Briz,
Antonio Ramírez-Treviño, Héctor Blanco-Alcaine**

Universidad Zaragoza 1542 Cinvestav intel

**37th Euromicro Conference on Real-Time Systems
Brussels, July 8-11, 2025**

GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES AGENCIA ESTATAL DE INVESTIGACIÓN