Self-Configuring and Self-Healing Time-Sensitive Networking

Prof. Paul Pop
Technical University of Denmark

Credits: DTU PhD students, postdocs: Dr. Luxi Zhao, Michael Raagaard
Silviu S. Craciunas, TTTech Computertechnik AG, “Deterministic Ethernet” presentation
Some figures reused from Belden/Hirschmann “Time-Sensitive Networking for Dummies”

Time-Sensitive Networking (TSN)

- **IEEE 802.1 Time-Sensitive Networking (TSN)** Task Group proposes sub-standards that extend Ethernet for safety-critical and real-time communication:
  - Time synchronization
  - Scheduling
  - Reservation and configuration, etc.

Standard	Description
802.1ASrev	Timing & Synchronization
802.1Qbv	Enhancements for Scheduled Traffic (Timed Gates for Egress Queues)
802.1Qbu	Frame Preemption
802.1Qca	Path Control and Reservation
802.1Qcc	Central Configuration Management
802.1Qci	Per-Stream Time-based Ingress Filtering and Policing
802.1CB	Redundancy, Frame Replication & Elimination

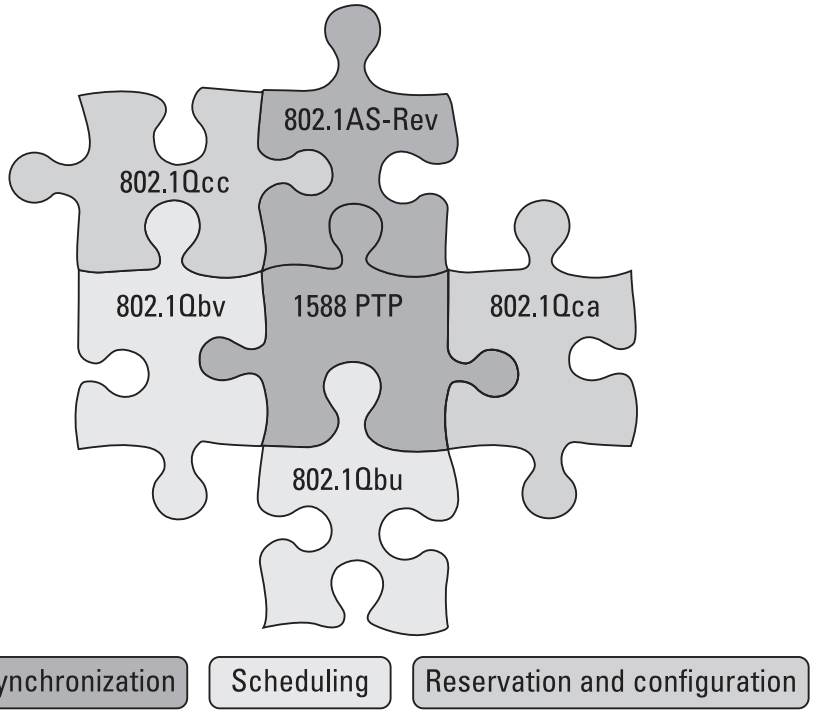


Figure source: TSN for Dummies - Belden's free guide to Time Sensitive Networking

TSN configuration challenges

Each substandard has configuration “knobs” and their configuration forms interconnected intractable optimization problems

GCL synthesis problem

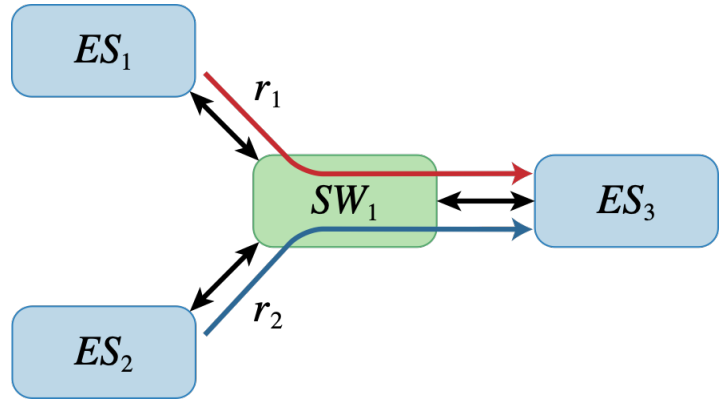
The **TSN (Qbv) schedule** defines open and close events for the **Gate Control List (GCL)** in each output port of every TSN device in the network

The schedule is built **off-line** taking into account the maximum end-to-end latency, frame length, as well as constraints derived from resources and physical limitations.

Advantage: the worst-case delays (latency) and the jitters can be minimized via the way the GCL is built

Given

- A network topology
- A set of TT streams

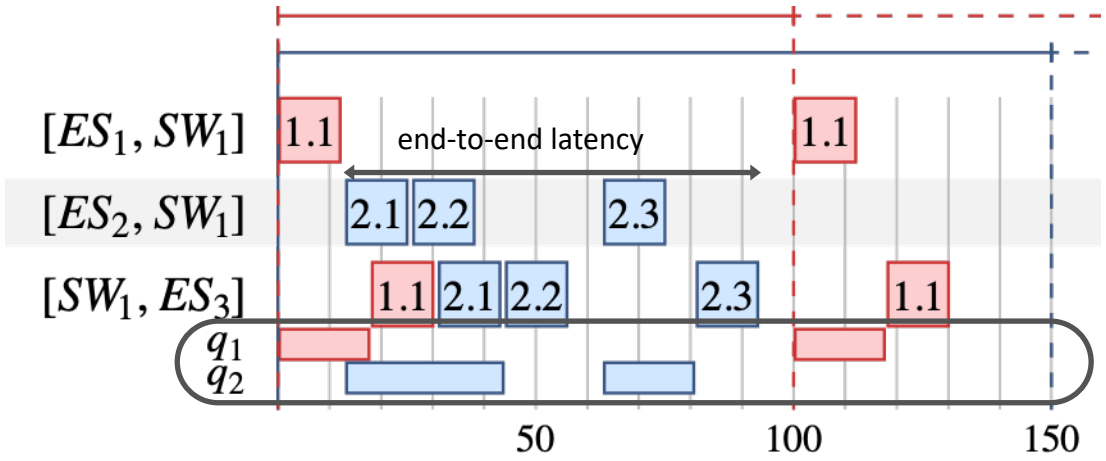


Find A *feasible* schedule

- Assign flows to egress port queues
- Determine periodic offsets for frames

With minimized

- Queue usage
- End-to-end latency



GCL synthesis: an intractable problem

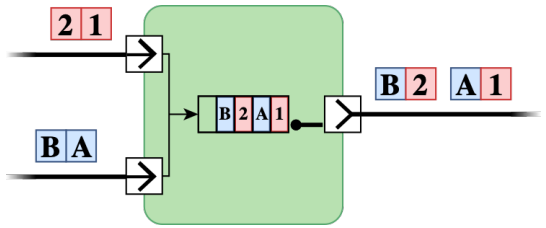
- Combinatorial optimization: finding an optimal object from a finite set of objects; typically, exhaustive search is not tractable
- Intractable problems: cannot be solved by a polynomial-time algorithm
- Solutions proposed in the literature for GCL synthesis:
  SMT/OMT, ILP, CP, Heuristics, Metaheuristics

Assumptions: flow isolation and scheduled end-systems

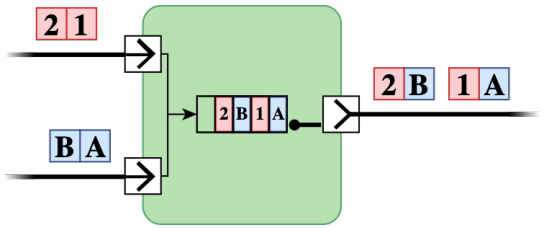
Non-determinism

Roughly same arrival time

Scenario 1

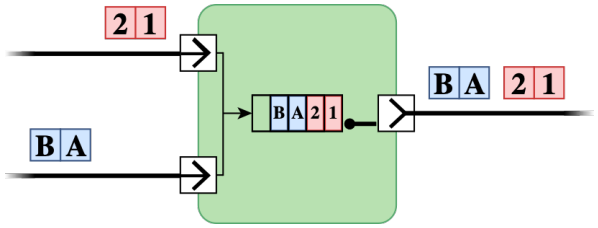


Scenario 2

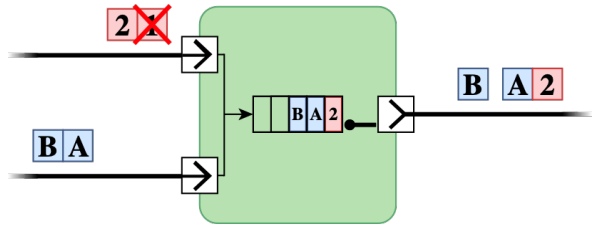


Frame loss

Scenario 1



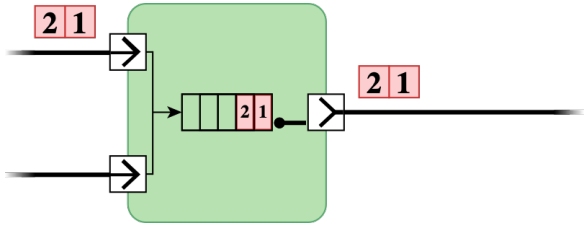
Scenario 2



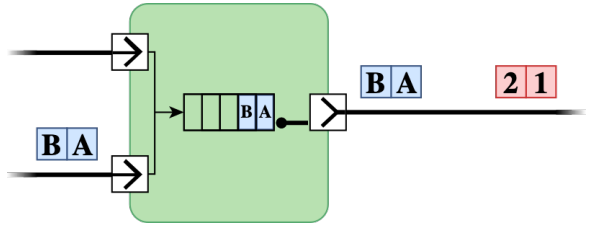
Solution

Stream isolation

Time t_i



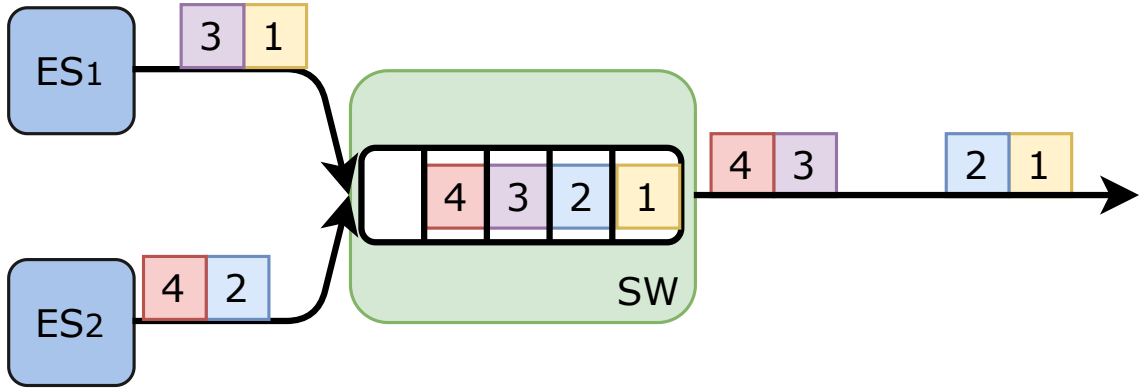
Time t_{i+1}



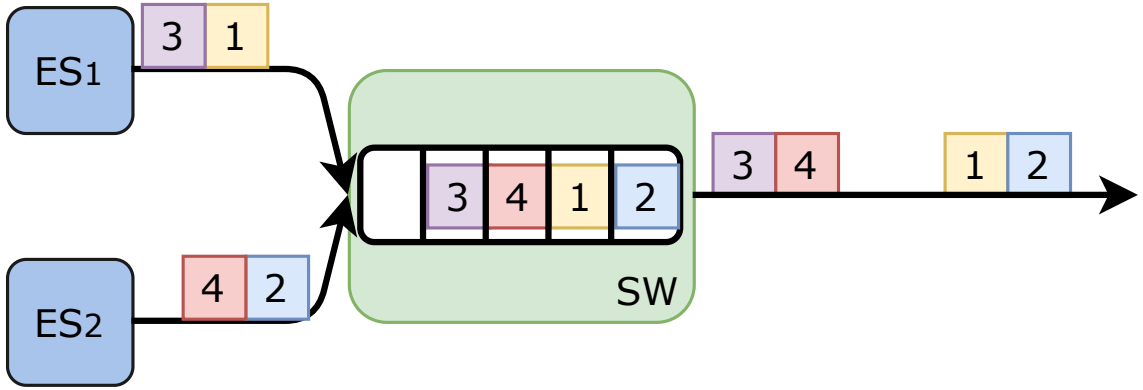
What if the end systems are **not** scheduled?

- If the previous assumptions do not hold, the GCL synthesis problem changes from a combinatorial optimization problem to a **schedulability analysis problem**

Frames could be sent like this…



…or like this



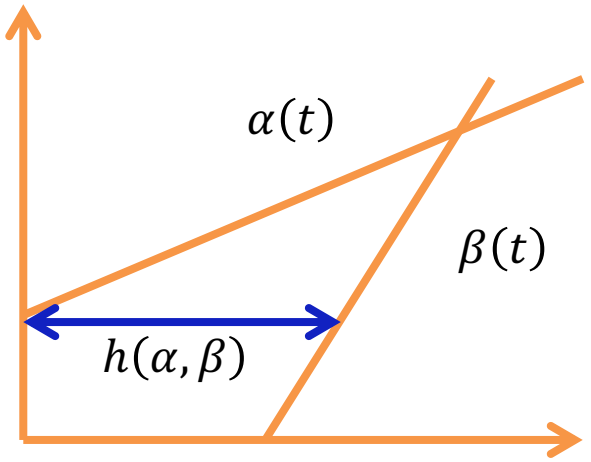
What is the worst-case latency of a frame?

Network Calculus-based schedulability analysis

- Network calculus: a theory to get guaranteed upper bounds h for delays
  - Based on the min-plus (min,+) algebra
  - Two basic mathematics operations: convolution and deconvolution

- Network calculus concepts:
  - Streams and arrival Curve α
  - Servers and service Curve β

- Network calculus is too slow to be used for guiding a search for configurations

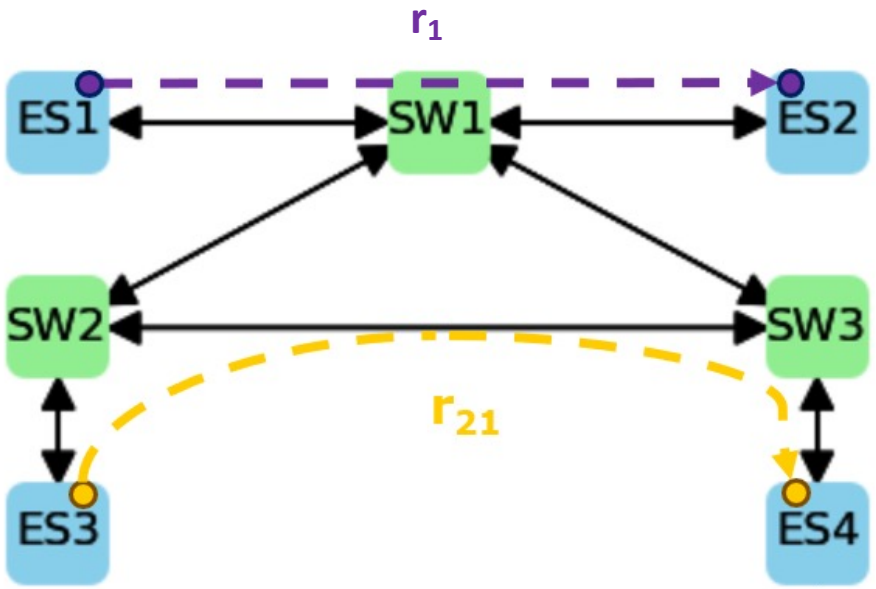


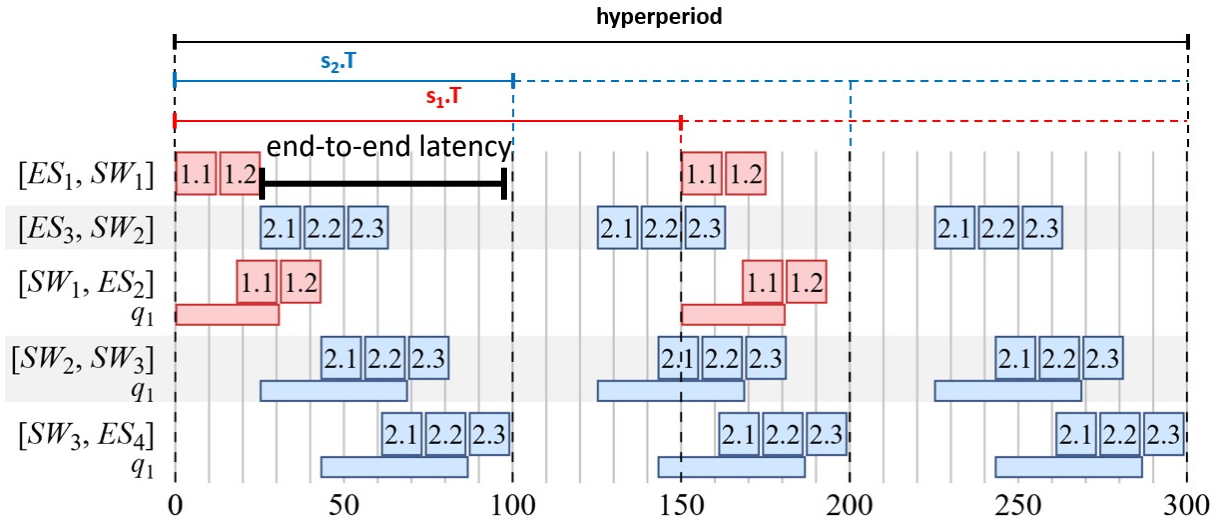
Streams routing problem (for TT)

Given
- A network topology
- A set of TT streams

Find
- Routes for the TT streams
- A feasible schedule

With minimized
- Link utilization
- End-to-end latency

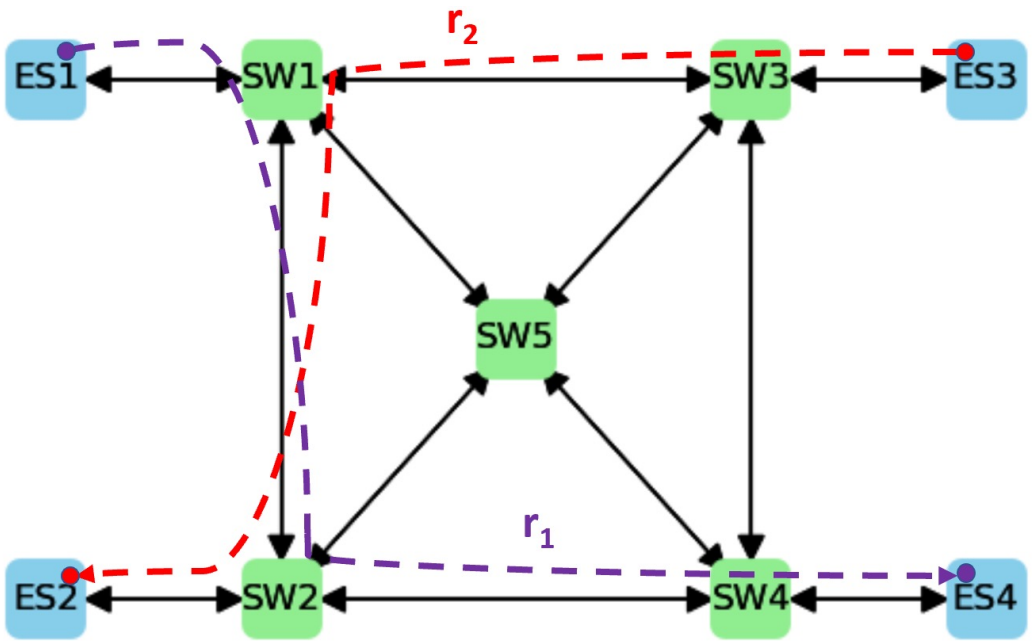




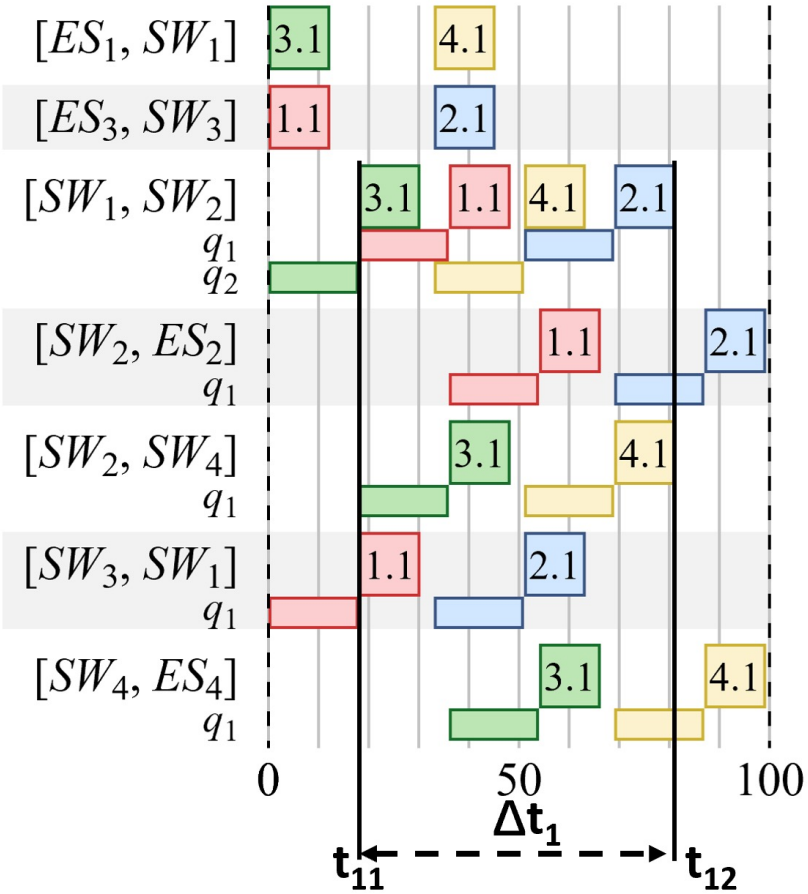
Routing example: shortest paths

Streams s_1, s_2: ES3 => ES2 with route r_2

Streams s_3, s_4, s_5: ES1 => ES4 with route r_1



Congestion on SW_1-SW_2
Stream s_5 cannot be scheduled

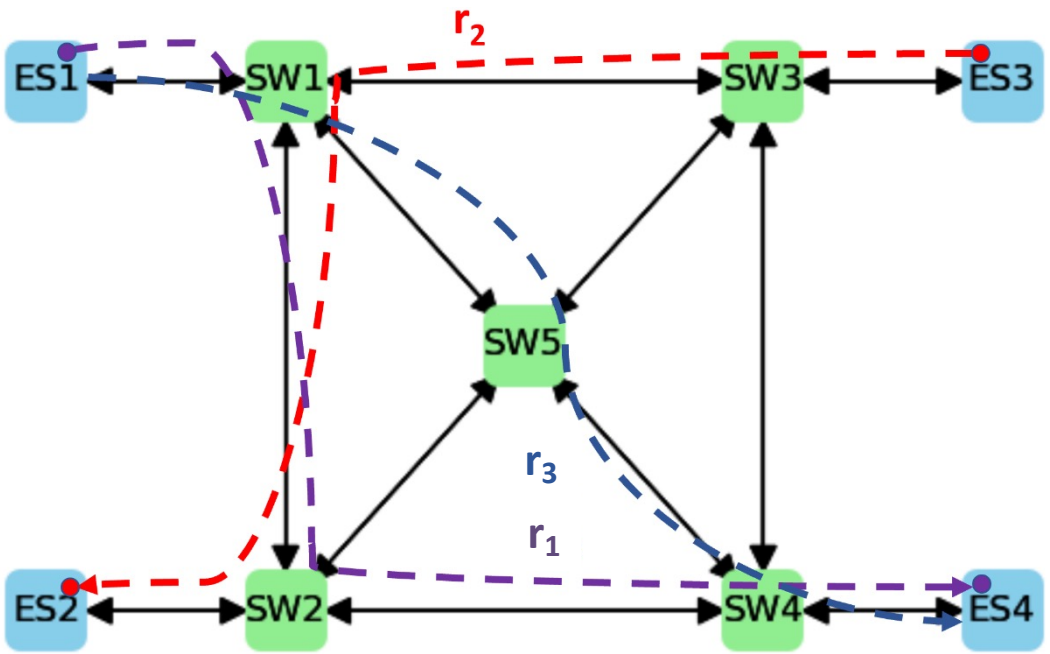


Moving s_1 to a longer route

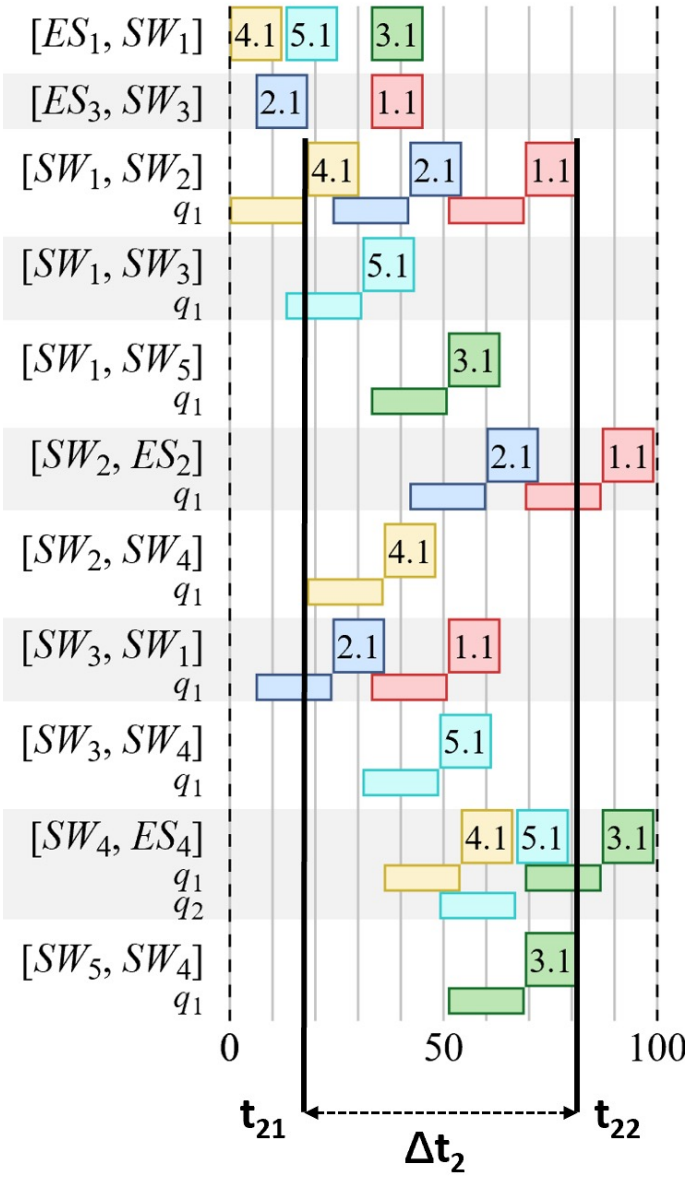
Streams **s_4, s_5** ES3 => ES2 with route **r_2**

Streams **s_2, s_3** ES1 => ES4 with route **r_1**

Stream **s_1** ES1 => ES4 with route **r_3**



Congestion is reduced
A feasible schedule can be derived



IEEE 802.1CB: Seamless redundancy in TSN

- **Seamless redundancy:** All network paths are used in parallel, so no disruption occurs if one path fails.
- **Non-seamless (failover) redundancy:** The protocol recovers the fault by switching from the primary path to the secondary path; it may result in a very brief disruption.
- 802.1CB implements Frame Replication and Elimination for Reliability

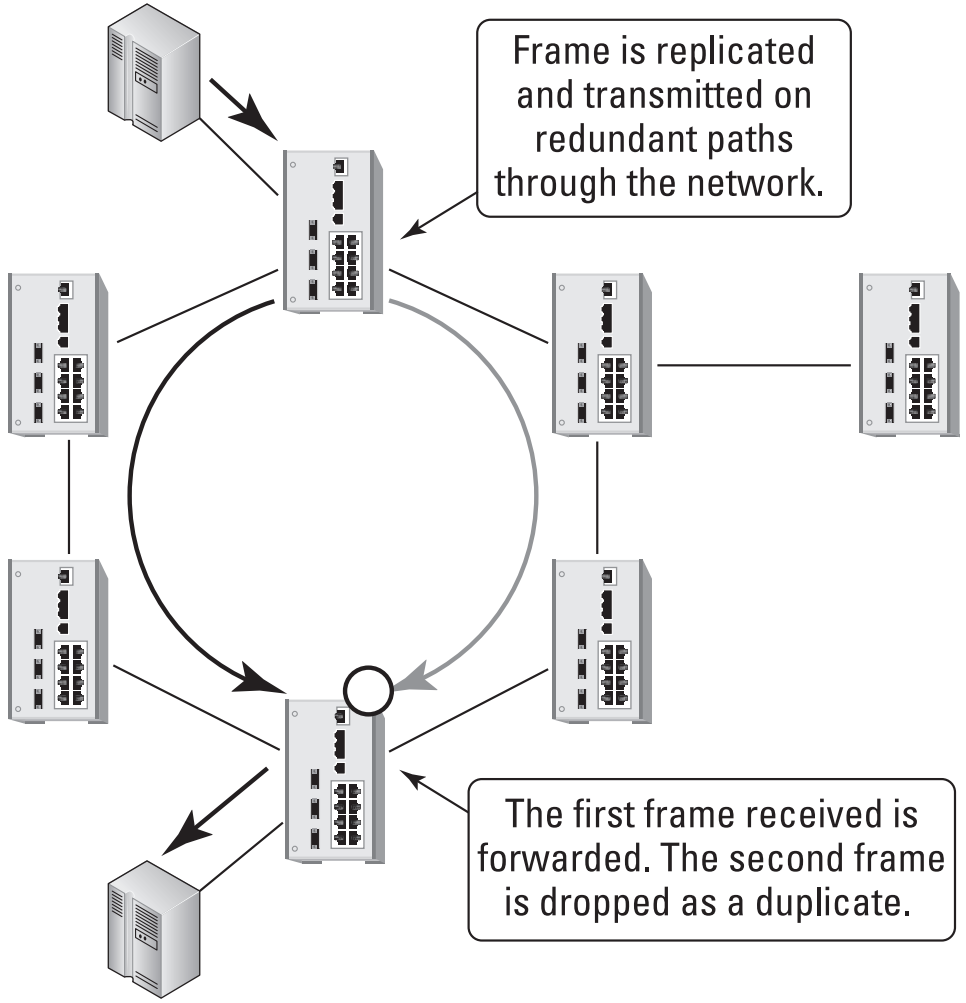


Figure source: TSN for Dummies - Belden's free guide to Time Sensitive Networking

Configuration challenges

Configuration parameters

- Traffic types for flows and their configuration
  - Qbv: Time-Aware Scheduler (TAS)
  - Qch: Cyclic queuing and forwarding
  - Qcr: Asynchronous Traffic Shaping (ATS)
  - QBA: Audio-Video Bridging (AVB)
- Queue assignment
- Gate Control Lists (GCLs)
- Routing including redundancy (802.1CB)

Competing objectives, constraints

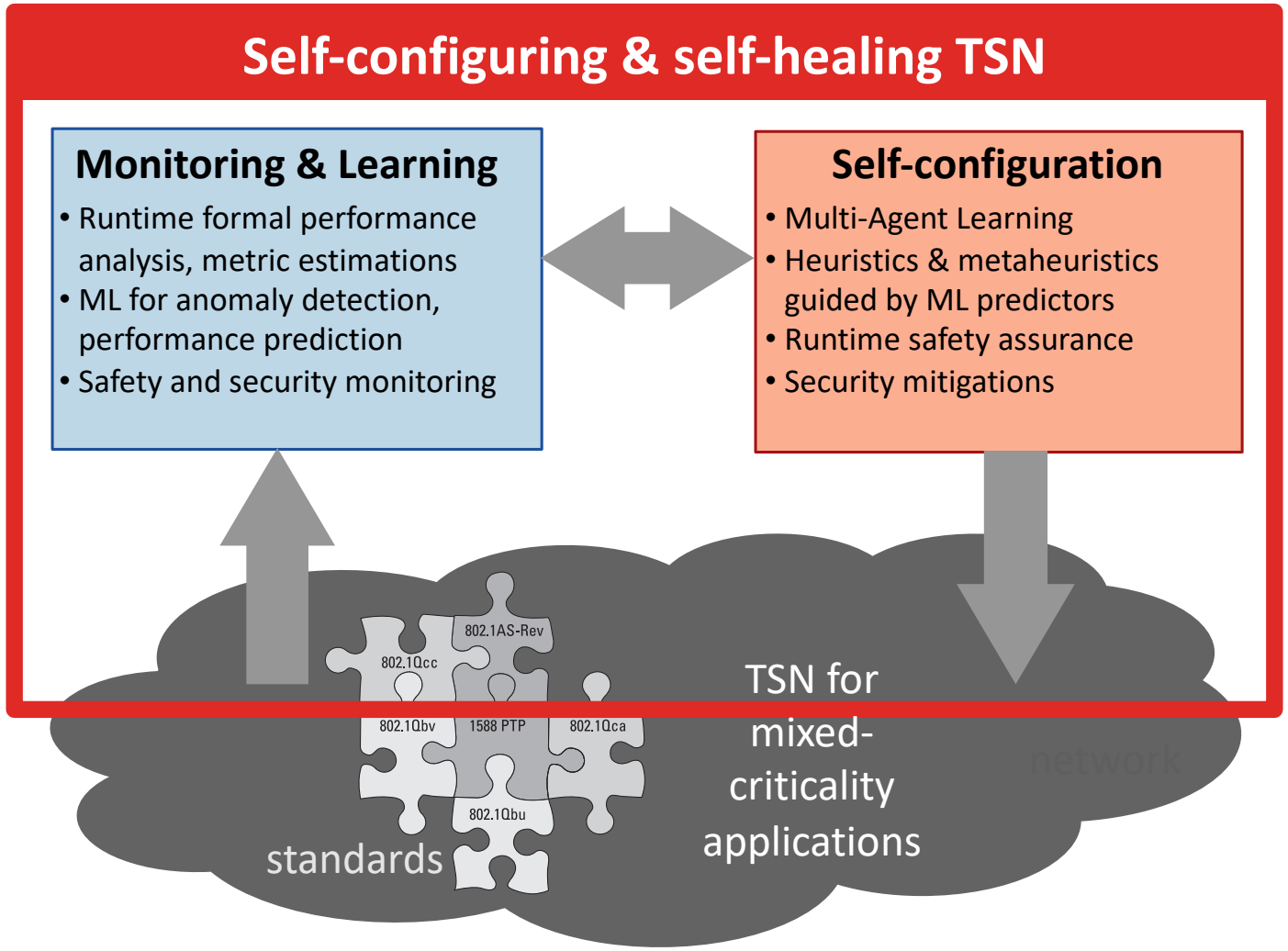
- Mixed-criticality applications: safety-critical, hard and soft real-time, best effort
- Performance
  - Jitter, latency and worst-case delays
  - Link utilization
- Safety and security: guaranteeing constraints and mitigating attacks and failures

Parameters influence each other in unexpected ways
The configuration problems are intractable and interconnected

Vision: Self-configuring and self-healing TSN

We need researchers working on multi-disciplinary challenges to realize this vision

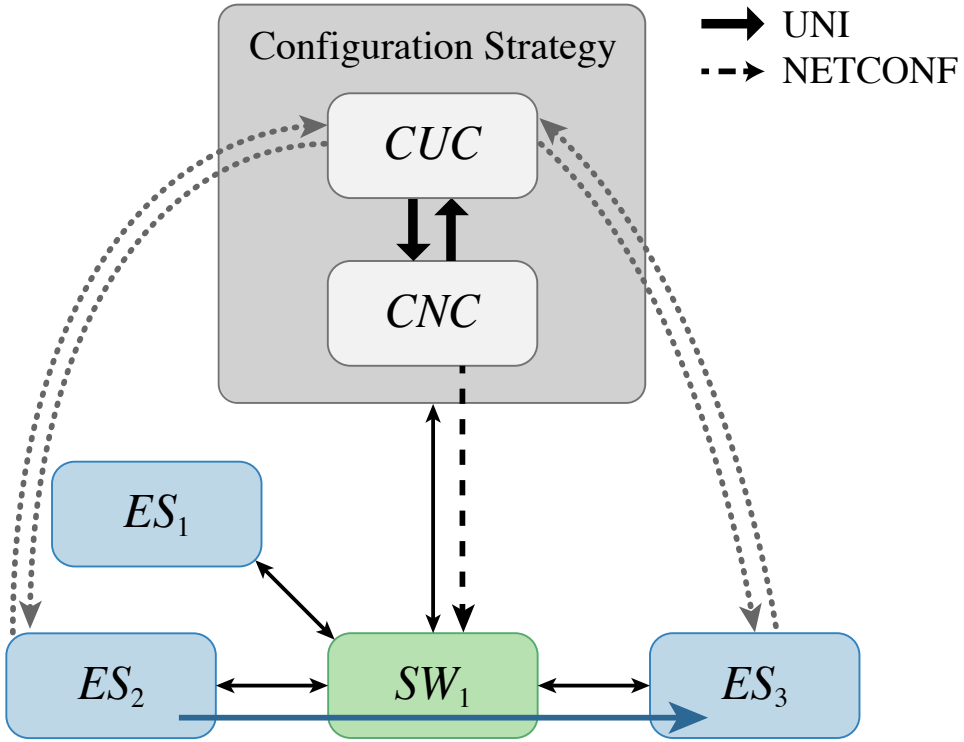
Vision: Self-configuring & self-healing TSN



Related:
Intent-based networking (IBN)
ETSI's Zero touch network & Service Management (ZSM)

Self-configuring TSN

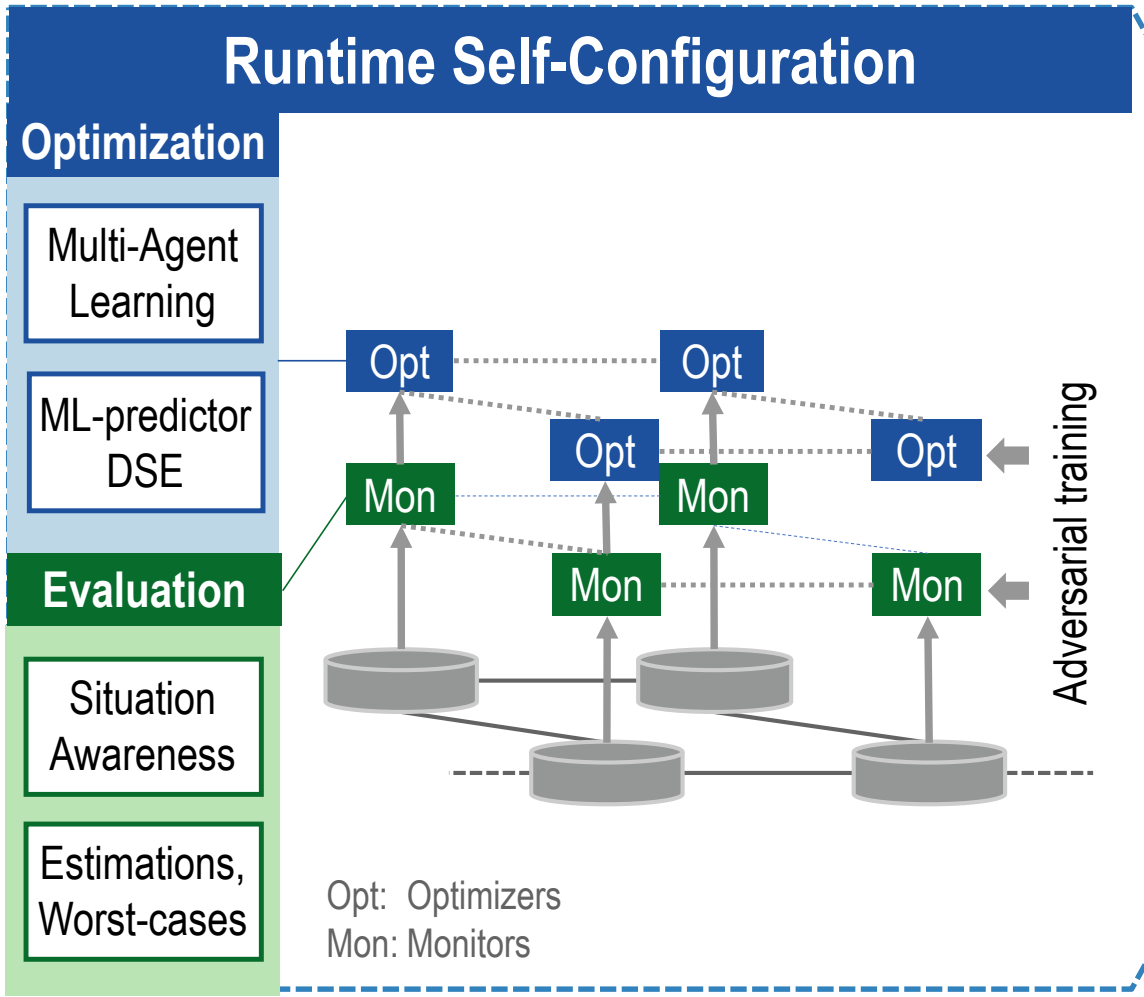
Centralized vs. decentralized configuration

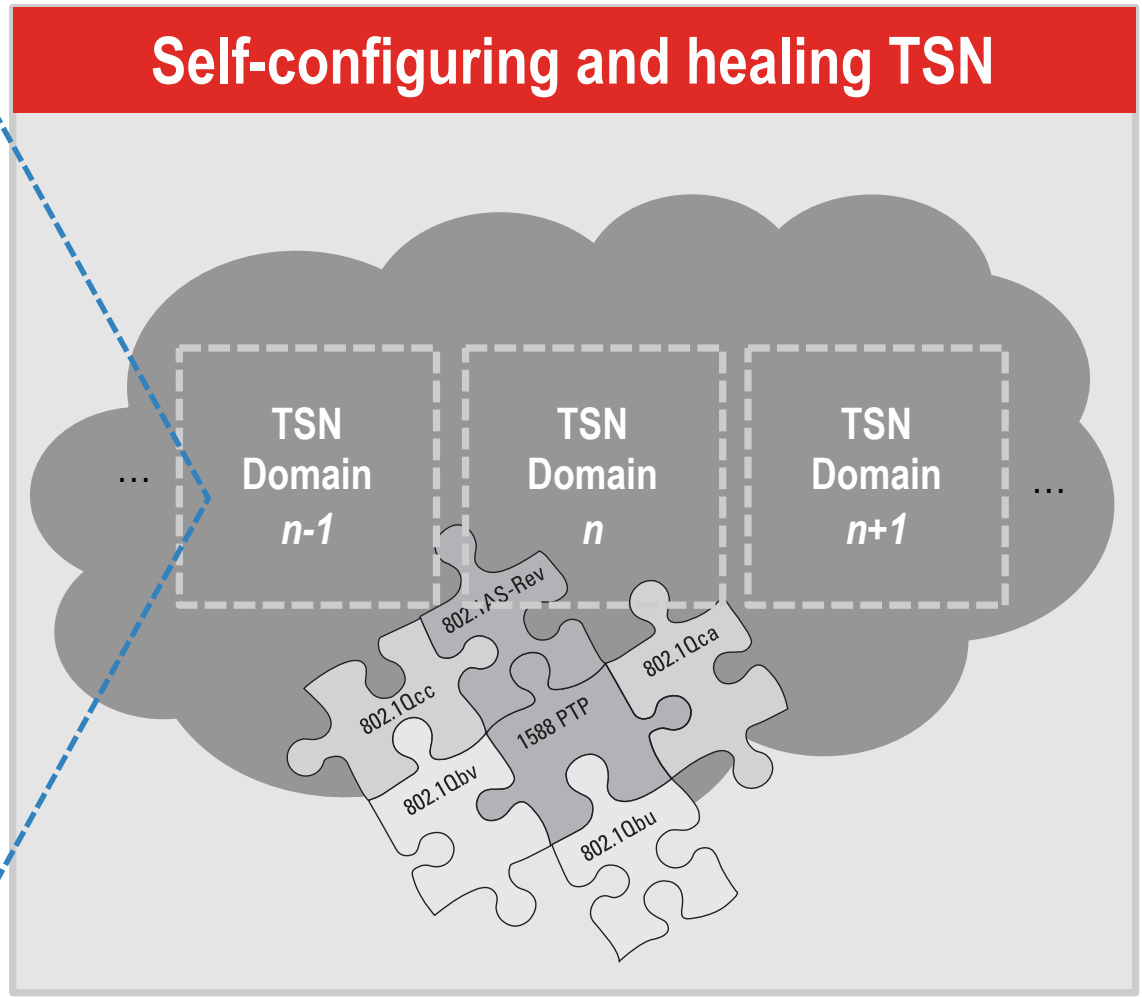


Runtime algorithms

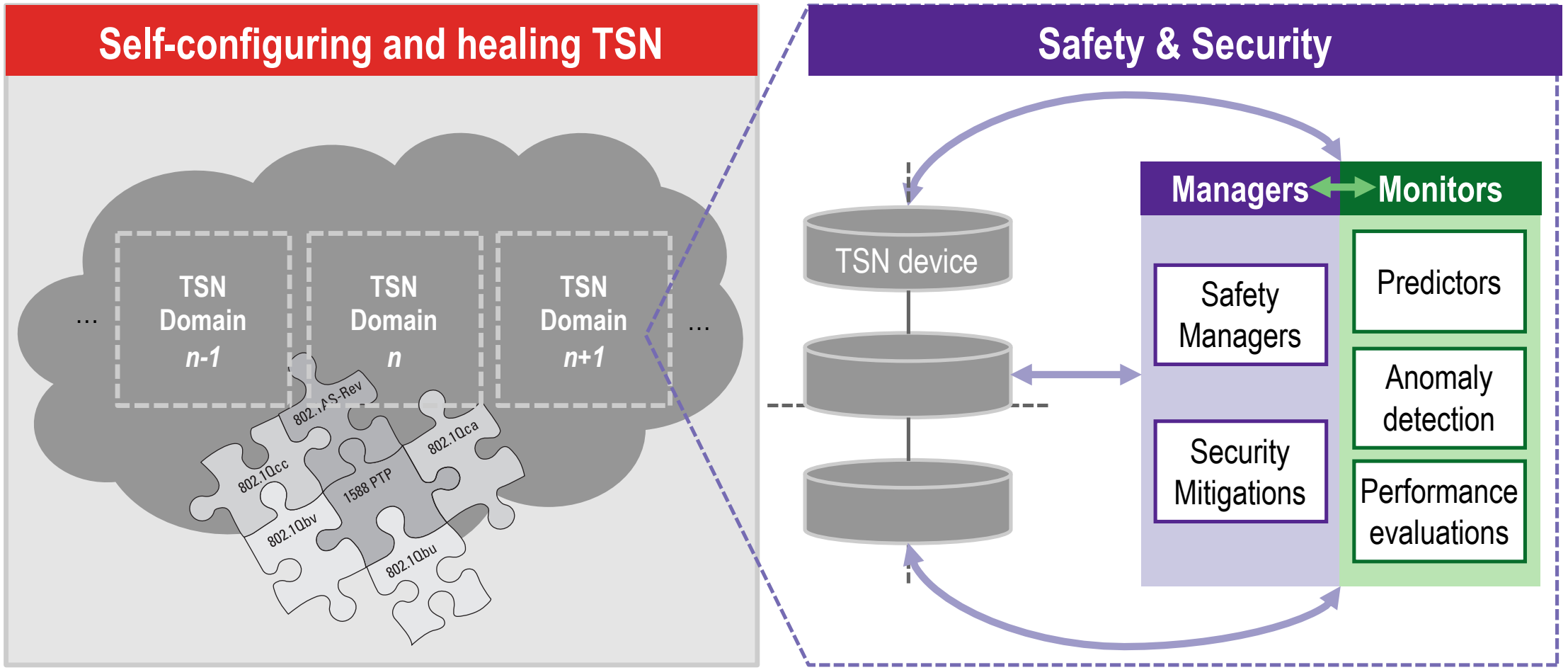
- Receive inputs from the monitoring module.
- AI-based optimization, hybrid heuristic and meta-heuristic algorithms: good quality solutions in a short time.
- Incremental evaluation & impact analysis for objective functions (e.g., incremental network calculus).
- Limitations of state-of-the-art: single domains, single traffic types, limited configuration, algorithms do not scale.

AI for self-configuration





Runtime assurance for safety & security



Example heuristic runtime scheduling

- Given
  - existing configuration (feasible schedule)
  - disappeared flows in **red**
  - appeared flows in **green**
- Determine a new configuration excluding **red** flows and including **green**
- Scheduling Heuristic