Niklas Reusch, Paul Pop (Technical University of Denmark)
Silviu S. Craciunas (TTTech Computertechnik AG)

Work-In-Progress: Safe and Secure Configuration Synthesis for TSN using Constraint Programming

Introduction & Technologies

Our work targets safety-critical real-time systems as found in the automotive, aerospace or industrial domains.

Our network technology of choice is Time-Sensitve Networking (TSN):

- Extension to Ethernet to allow real-time scheduling
- Provides clock synchronisation across network (802.1 ASrev)
- Provides deterministic message scheduling (802.1 Qbv)
- Provides support for message redundancy (802.1CB FRER)

To provide authentication we use Timed Efficient Stream Loss-Tolerant Authentication (TESLA):

- Asymmetric authentication
- Multicast
- Resource efficient

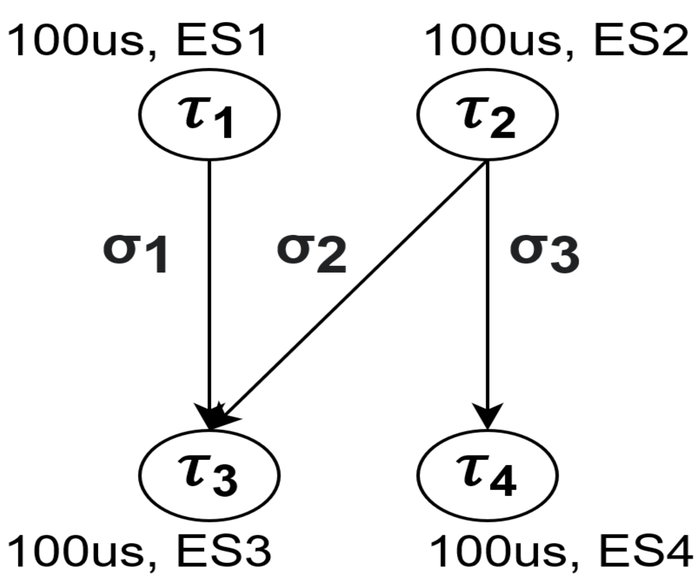

Figure 2: An example application, consisting of multiple tasks with interdependecies, mapped to different end-systems

Model & Problem Formulation

Given:

- TSN network architecture
- Set of periodic applications with security, redundancy & function path (chain of task with deadline) requirements

Determine:

- Set of TESLA security applications generated based on security requirements
- Set of TSN streams
  - Packing
  - Routing
- Static periodic task schedule
- GCL-based network schedule

Such that:

- Redundancy, TESLA security & function path deadline requirements are met
- Latency of function paths is minimized

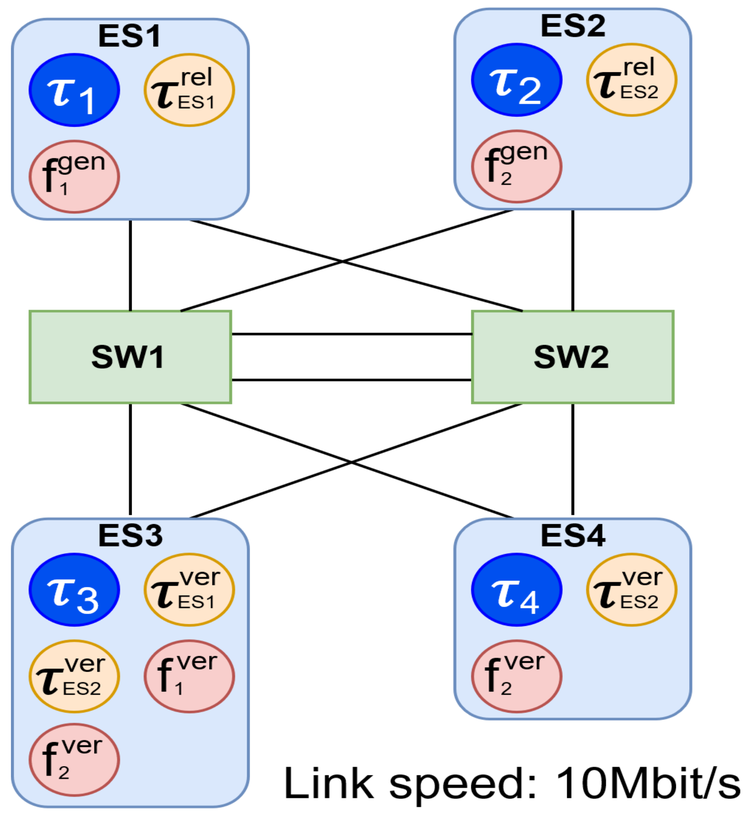

Figure 1: An example network archcitecure consisting of end-systems and switches. Additionally the mappping of tasks to end-systems is shown

Constraints

- TESLA Interval: Maximum value to fulfill requirements.
- Routing: No cycles, Bandwidth not exceeded, Redundant streams don't overlap
- Scheduling: No Overlap, Dependencies, TESLA security condition, TSN frame isolation, Function-path deadlines

Solution

Solution implemented in Python using Google OR-Tools. Created a nice UI using Plotly/Dash, presented in RTSS@Work Paper.

Synthetic + Automotive Testcases

Results

- Result 1: TESLA overhead is considerable for small messages
- Result 2: Our solution significantly improves schedulability/laxity compared to an ASAP solution
- Result 3: Scalable up to medium sized architectures

Tool available on GitHub:
https://github.com/nreusch/TSNConf

Future Work

Compare with heuristic solution
Evaluate scalability
Compare with other authentication protocols

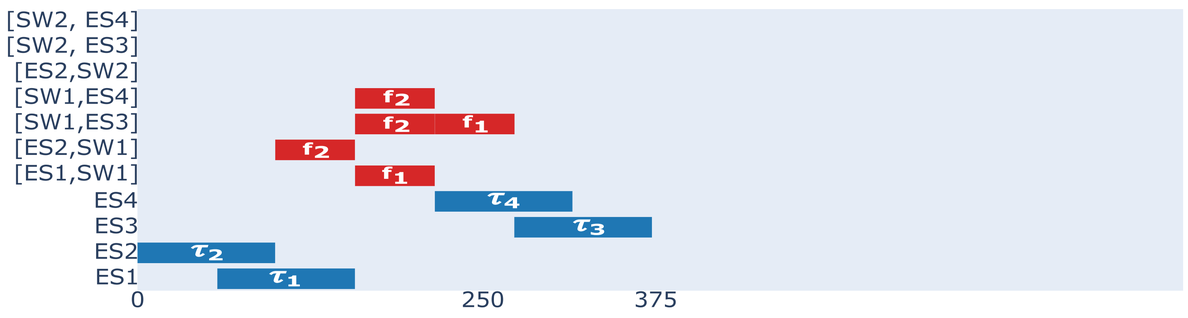

Figure 3.1: Schedule without security & redundancy: TESLA is not used for security and redundant routing is not used for fault-tolerance.

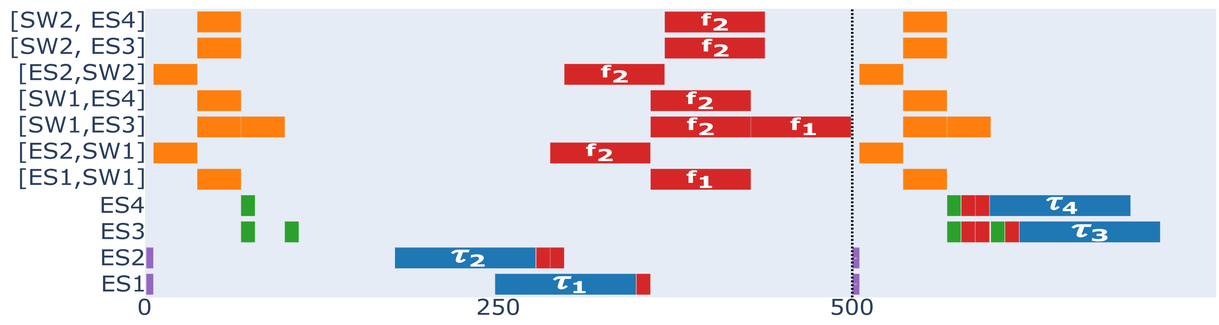

Figure 3.2: Schedule with security & redundancy: TESLA is applied to secure the streams and redundant routing is used for fault-tolerance