TSN scheduling algorithm for real-time applications in a heterogeneous network

IEEE 802.1Qbv time-aware shaper for optimising the estimated end-to-end quality of service.

Master’s Degree in Innovation and Research in Informatics (Computer Networks and Distributed Systems)

Candidate: **Gianluca Graziadei**

Supervisor: **Luis Domingo Velasco Esteban**

Co-supervisor: **Marc Ruiz Ramirez**

June 27th, 2024

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

FIB

TIMING

This thesis is in the scope of NextGeneration UNICO5G Towards a smart and effIcient telecoM Infrastructure meetiNG current and future industry needs (TIMING) (TSI-063000-2021-145).

`https://timing.upc.edu/`

Table of Contents

TSN covers real-time applications crucial network requirements such as latency and transmission determinism guarantees.

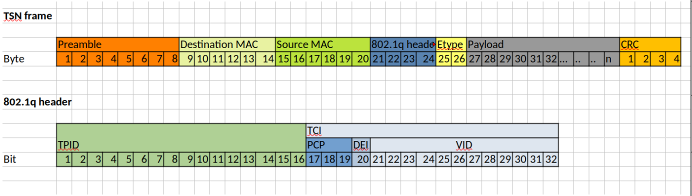
IEEE 802.1Q

Time Sensitive Networking Task Group aims to provide deterministic services through IEEE 802 networks.

- Synchronous network.
- TDMA, time division multiple access.
- Coexistence of different classes, such as TS, QoS, and BE.

IEEE 802.1Q frame

1 Introduction



Tag protocol identifier (TPID)	A 16-bit field set to a value of **0x8100** to identify the frame as an IEEE 802.1Q-tagged frame.
Tag control information (TCI)	
Priority code point (PCP)	A 3-bit field which refers to the IEEE 802.1p class of service (CoS) and maps to the frame priority level.
Drop eligible indicator (DEI)	A 12-bit field specifying the VLAN to which the frame belongs.
VLAN identifier (VID)	If active, this flag marks the frame as eligible to be dropped in the presence of congestion.

Table: IEEE 802.1Q header description

Provisioning of TS and non-TS flows

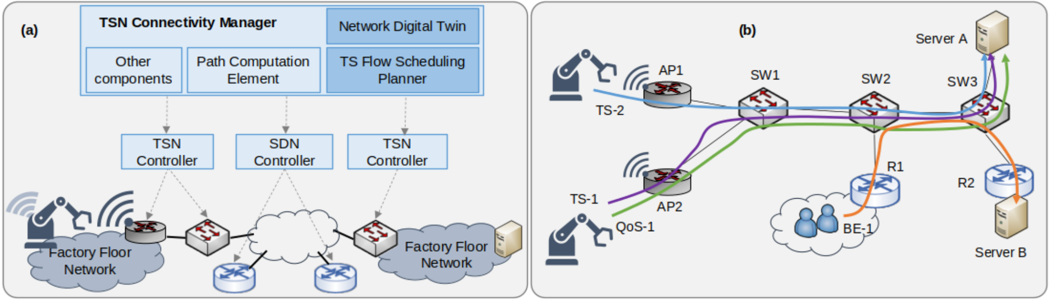
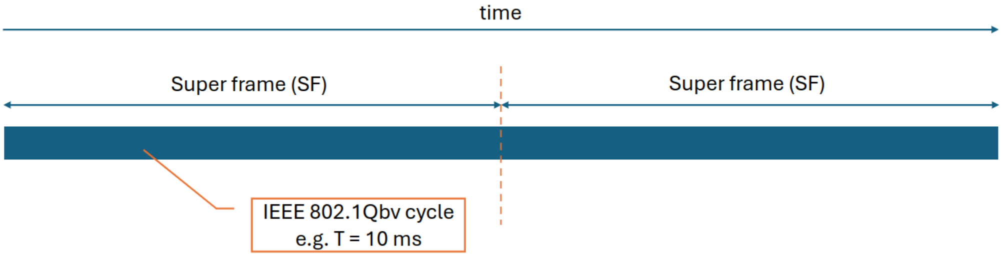


Image credits [9]

IEEE 802.1Qbv Time Aware Shaper (TAS) scheduler

Super Frame (SF)

IEEE 802.1Qbv Time Aware Shaper (TAS), is designed to separate the communication on the Ethernet network into fixed-length, repeating time cycles. The repeating scheduling cycle is called Super Frame (SF).



TSN TAS Scheduler

1 Introduction

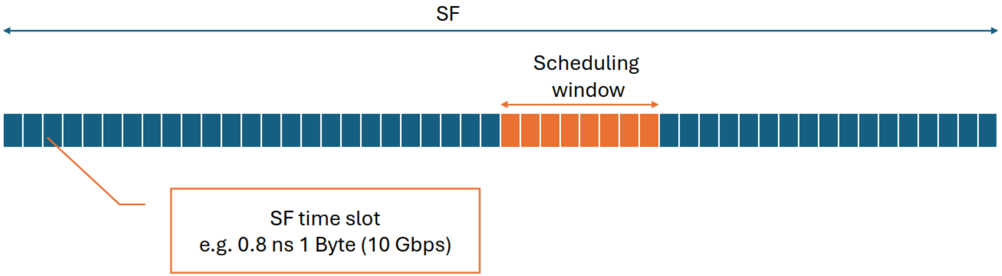
For each network interface, TSN TAS scheduler aims to:

1. Discretize the SF in time slots, which atomic unit in time and bit processed.
2. Assign scheduling windows for each TS-request, which atomic set of time slots.
3. Guarantee the synchronization with Bands of Guard.
4. Ensure time-critical constraints of each TS request.
5. Assign resources as an optimization problem.
6. Given a timestamp and a known data plane, return the corresponding TS-queue.

Scheduling Policy

1 Introduction

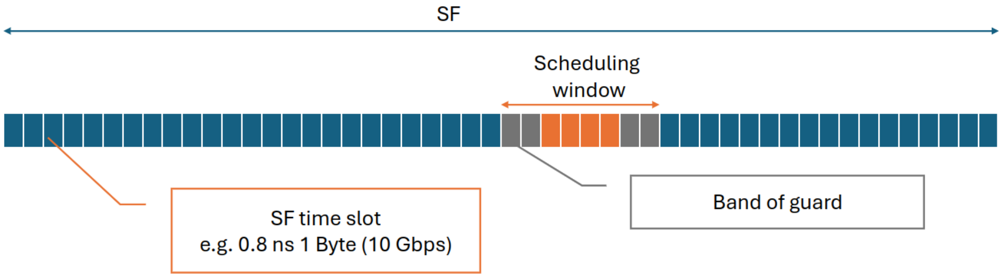
1. Discretize the SF in time slots, which is an atomic unit in time and bit processed.
2. Assign scheduling windows for each TS request, which is an atomic set of time slots.



Band of Guard

1 Introduction

3. Guarantee the synchronization with Bands of Guard.



Scheduling objectives

5. TAS scheduling is an optimization problem.
   - **Accepting the highest number of TS requests**
   - Enhance the overall system performance (in charge of NDT + PCE)
   - **Minimize jitter and delay**
   - **Reducing the changes between reconfigured data planes**

*In bold the objective covered by the scheduler.

Metrics Estimation

The End-To-End (e2e) delay is the additive result of:

- **Delay of the signal**
- **Propagation Delay**
- **Processing Delay**:
  - Reception of Data Packet
  - Buffering
  - Error Checking
  - Transmission
- **Queuing delay / Scheduling delay**

The e2e jitter is the variance of the delay of a transmission which is theoretically periodical:

- For optimization purposes the **difference between the maximum and minimum delay** of iterations of the same TS-flow.

State of the art

1 Introduction

Reference	Category	Approach
[5], [7]	ILP	Job-shop flow
[8], [11]	Heuristic	Vector bin packing
[4]	Heuristic	Dynamic programming + Greedy
[2]	Z3 SMT/OMT	OMNeT++ simulation
[3]	Genetic algorithm	BRKGA
[1], [10]	Machine learning	Deep reinforcement learning
[6]	Machine learning	Reinforcement learning

Table: TAS scheduling algorithm literature

Table of Contents

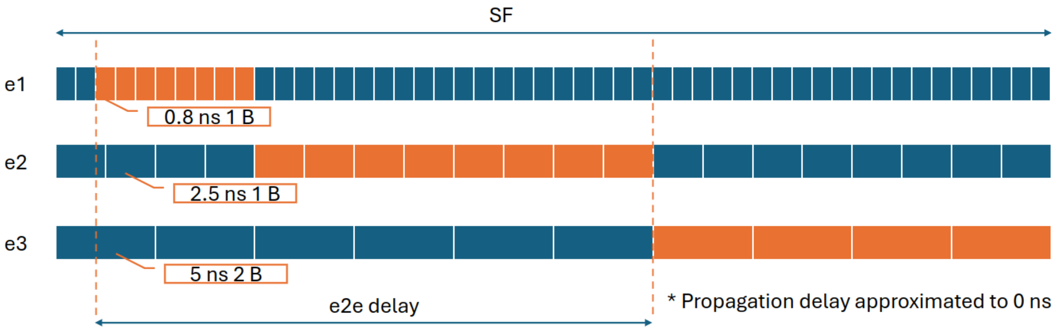
2 Problem statement

Heterogeneous factors

1. Throughput
2. Transmission mode:
   - Simplex.
   - Half-Duplex.
   - Full-Duplex.
3. Processing mode:
   - **Express**: data is transmitted immediately as it becomes available, without waiting for the entire message to be received.
   - **Store-and-Forward**: the entire message is received and stored in a buffer before being forwarded to the next hop on the network.
4. Not available time slots (e.g. preamble)
5. Propagation delay

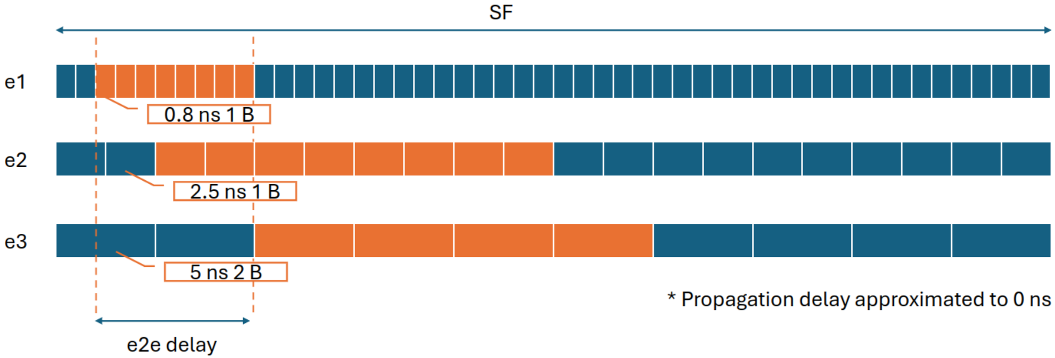
Store-and-Forward Processing Mode

2 Problem statement



Express Processing Mode

2 Problem statement



Inspired by microarchitecture pipeline design.

Objective:

- **Minimizing the time between the start of scheduling windows** of two different contiguous interfaces in the path of a TS request.

Inspired by microarchitecture pipeline design.

Objective:

- **Minimizing the time between the start of scheduling windows** of two different contiguous interfaces in the path of a TS request.

Constraints:

- Constraint 1: The input-output dependency between two interfaces claims that it is impossible to send a bit without the bit being received by the interface.

Pipeline Optimization

Inspired by microarchitecture pipeline design.

Objective:

- **Minimizing the time between the start of scheduling windows** of two different contiguous interfaces in the path of a TS request.

Constraints:

- Constraint 1: The input-output dependency between two interfaces claims that it is impossible to send a bit without the bit being received by the interface.
- Constraint 2: The preemption is not allowed. Each scheduling window is an atomic unit of w_{fe} time slots.

Pipeline Optimization

Inspired by microarchitecture pipeline design.

Objective:

- **Minimizing the time between the start of scheduling windows** of two different contiguous interfaces in the path of a TS request.

Constraints:

- Constraint 1: The input-output dependency between two interfaces claims that it is impossible to send a bit without the bit being received by the interface.
- Constraint 2: The preemption is not allowed. Each scheduling window is an atomic unit of w_{fe} time slots.
- Constraint 3: The propagation delay of the e_1 is the lower bound.

Problem Statements

Given a time-sensitive network with a previous scheduled TS-load:

Scheduling without reconfiguration

Accepting a new TS request if feasible.

Given a time-sensitive network and an **incumbent scheduling plan** with allocated resources of different TS-requests:

Scheduling with reconfiguration

Accepting a new TS-request and producing, if feasible, as output the new scheduling plan **reducing the number of performed changes** between it and the incumbent one.

Scheduling with reconfiguration example

2 Problem statement

<table>
<tr><td>time</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td><td>37</td><td>38</td><td>39</td><td>40</td></tr>
<tr><td>incumbent</td><td colspan="40"></td></tr>
<tr><td>e2</td><td colspan="40"></td></tr>
<tr><td>e1</td><td colspan="8"></td><td colspan="8"></td><td colspan="8"></td><td colspan="8"></td><td colspan="8"></td></tr>
<tr><td>new</td><td colspan="40"></td></tr>
<tr><td>e2</td><td colspan="40"></td></tr>
<tr><td>c_e2</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>e1</td><td colspan="8"></td><td colspan="8"></td><td colspan="8"></td><td colspan="8"></td><td colspan="8"></td></tr>
<tr><td>c_e1</td><td colspan="8">0</td><td colspan="8">1</td><td colspan="8">0</td><td colspan="8">0</td><td colspan="8">1</td></tr>
</table>

* In red are the changes between the incumbent and the new data plane.

Table of Contents

ILP model - Scheduling without reconfiguration

Decision variables:

- Binary $X = \{x_{eti}\}$
- Integer $D = \{d_i\}$
- Integer J
- Integer W

Objective function:

- $minimize\ j$

Constraints:

1. $\sum_{i \in T_f} x_{feti} \leq 1 \quad \forall e \in E, \forall t \in T_e$
2. $\sum_{t \in T_e} x_{eti} = 1 \quad \forall f \in F, \forall e \in E_f, \forall i \in [1; T_f]$
3. $\sum_{t \in T_{e_2}} (t-1) * x_{e_2 ti} \geq pipeline(e_1, e_2) + \sum_{t \in T_{e_1}} (t-1) * x_{e_1 ti}\ \forall i \in [1, T_f], \forall (e_1, e_2) \in E_f$
4. $d_i = \sum_{t \in T_{e_{dest}}} (t-1) * x_{e_{dest} ti} - \frac{P_f}{\tau_{e_{dest}}} (i-1) \quad \forall i \in [1; T_f]$
5. $W \geq d_i * \tau_{dest} + d_{dest} \quad \forall i \in [1, T_f], dest \leftarrow E_f[-1]$
6. $W \leq \delta_f$
7. $j \geq d_{i_2} - d_{i_1} \quad \forall i_1, i_2\ s.t.\ 1 \leq i_1 \leq i_2 \leq T_f,\ d_{i_1} \leq d_{i_2}$
8. $\tau_{e_{dest}} * j \leq \upsilon_f$

ILP model - Scheduling with reconfiguration (1/2)

Decision variables:

- Binary $X = \{x_{feti}\}$
- Binary $X = \{y_{feti}\}$
- Binary $X = \{c_{fet}\}$
- Integer $D = \{d_{fi}\}$
- Integer $J = \{j_f\}$
- Integer Z
- Integer W

Objective function:

- $minimize \;\; p_1 * Z + p_2 * W + p_3 * \sum_{f \in F} \sum_{e \in E} \sum_{t \in T_e} c_{fet}$

Constraints:

1. $\sum_{f \in F} \sum_{i \in T_f} y_{feti} \leq 1 \;\; \forall e \in E, \; \forall t \in T_e$
2. $\sum_{t \in T_e} x_{feti} = 1 \;\; \forall f \in F, \; \forall e \in E_f, \; \forall i \in [1; T_f]$
3. $x_{feti} \leq y_{feki} \forall f \in F, \forall e \in E_f, \forall t \in T_e, \forall i \in [1, T_f], \forall k \in [t; t{+}w_{fe}]$
4. $d_{fi} = \sum_{t \in T_{e_{dest}}} (t-1) * x_{fe_{dest}ti} - \frac{P_f}{\tau_{e_{dest}}}(i-1) \;\; \forall f \in F, \; \forall i \in [1; T_f]$
5. $j_f \geq d_{fi_2} - d_{fi_1} \;\; \forall f \in F, \forall i_1, i_2 \; s.t. \; 1 \leq i_1 \leq i_2 \leq T_f, \; d_{fi_1} \leq d_{fi_2}$
6. $\tau_{e_{dest}} * d_{fi} + d_{e_{dest}} \leq \delta_f \;\; \forall f \in F, \; \forall i \in [1; T_f]$
7. $\tau_{e_{dest}} * j_f \leq \upsilon_f \;\; \forall f \in F$
8. $Z \geq j_f * \tau_{last} \;\; \forall f \in F, last \leftarrow E_f[-1]$
9. $W \geq d_{fi} * \tau_{last} + d_{last} \;\; \forall f \in F, \; \forall i \in [1, T_f], last \leftarrow E_f[-1]$

ILP model - Scheduling with reconfiguration (2/2)

Decision variables:

- Binary $X = \{x_{feti}\}$
- Binary $X = \{y_{feti}\}$
- Binary $X = \{c_{fet}\}$
- Integer $D = \{d_{fi}\}$
- Integer $J = \{j_f\}$
- Integer Z
- Integer W

Objective function:

- $minimize \;\; p_1 * Z + p_2 * W + p_3 * \sum_{f \in F} \sum_{e \in E} \sum_{t \in T_e} c_{fet}$

10. $\sum_{t \in T_{e_2}} (t-1) * x_{fe_2ti} \geq pipeline(e_1, e_2) + \sum_{t \in T_{e_1}} (t-1) * x_{fe_1ti} \forall f \in F,\ \forall i \in [1, T_f],\ \forall (e_1, e_2) \in E_f$
11. $\sum_{t \in T_e} t * x_{feti_2} \geq \sum_{t \in T_e} t * x_{feti_1} + 1 \;\; \forall f \in F, \forall i_1, i_2 \; s.t. \; 1 \leq i_1 < i_2 \leq T_f,\ \forall e \in E_f$

Reconfiguration constraints:

1. $c_{fet} \geq \sum_{i=1}^{T_f} (x_{feti} - s_{feti}) \;\; \forall f \in F,\ \forall e \in E_f,\ \forall t \in T_e$
2. $c_{fet} + \sum_{i=1}^{T_f} (x_{feti} + s_{feti}) \leq 2 \;\; \forall f \in F,\ \forall e \in E_f,\ \forall t \in T_e$
3. $c_{fet} \leq \sum_{i=1}^{T_f} (x_{feti} + s_{feti}) \;\; \forall f \in F,\ \forall e \in E_f,\ \forall t \in T_e$
4. $c_{fet} \geq \sum_{i=1}^{T_f} (s_{feti} - x_{feti}) \;\; \forall f \in F,\ \forall e \in E_f,\ \forall t \in T_e$

Heuristic algorithm

- **Constructive phase**: given a set of TS flows and an empty scheduling data plane generate the local best scheduling. Two sub-problems:
  - Find the **local optimal starting time slot**.
  - Given the starting time slot t define the **optimal allocation**.
- **LS phase**: given a new request that cannot be allocated, explore optimal local solutions that can accept the new scheduling closer to the incumbent scheduling plan.

LS jitter optimizer

3 Models and Algorithms

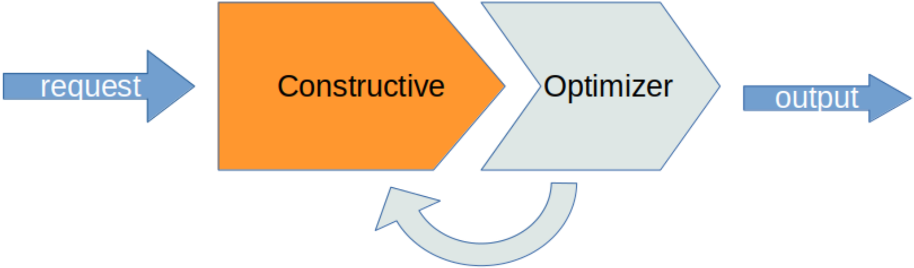


Table of Contents

Heuristic vs ILP

4 Results

Comparing conceptual models:

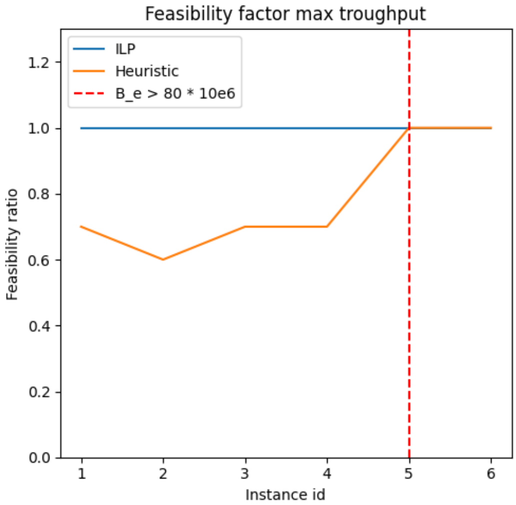
- ILP model without data plane reconfiguration with circular buffer optimization solved with Gurobi.
- Heuristic model without data plane reconfiguration with jitter optimization LS.

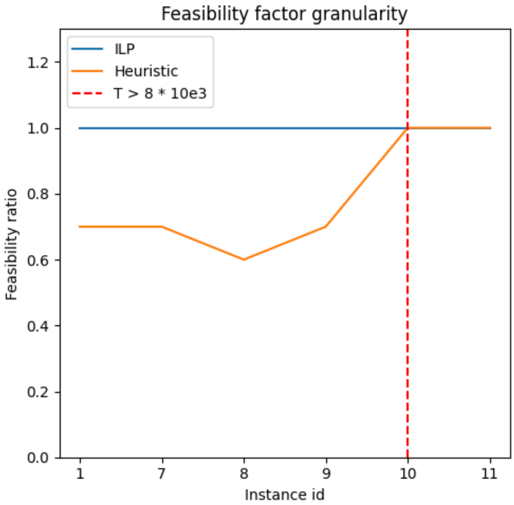
Fractional factorial design of experiment (DoE) with factor T (SF duration) and $max(B_e)$ (max throughput). Six levels for each factor.

T [ms] \ $max(B_e)$ [Mbps]	10	20	30	40	80	100
10	C1	C2	C3	C4	C5	C6
20	C7	-	-	-	-	-
30	C8	-	-	-	CZ	-
40	C9	-	-	-	-	-
80	C10	-	-	CX	-	-
100	C11	-	-	-	-	CY

Feasibility

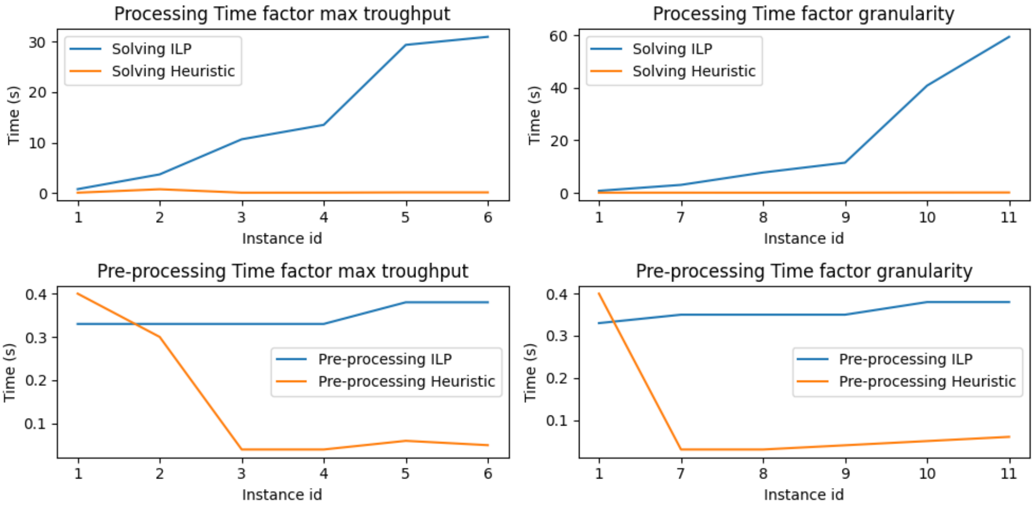
4 Results





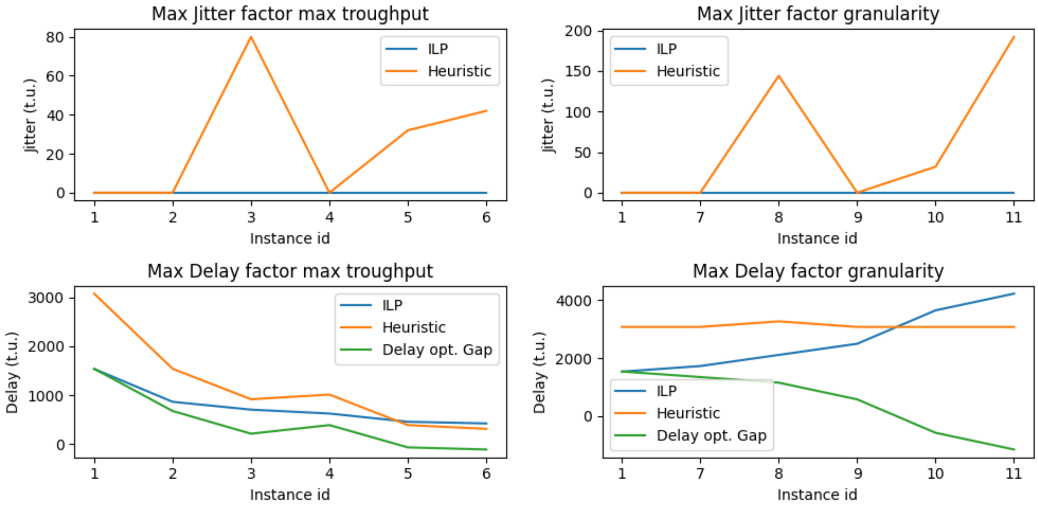
Execution time

4 Results



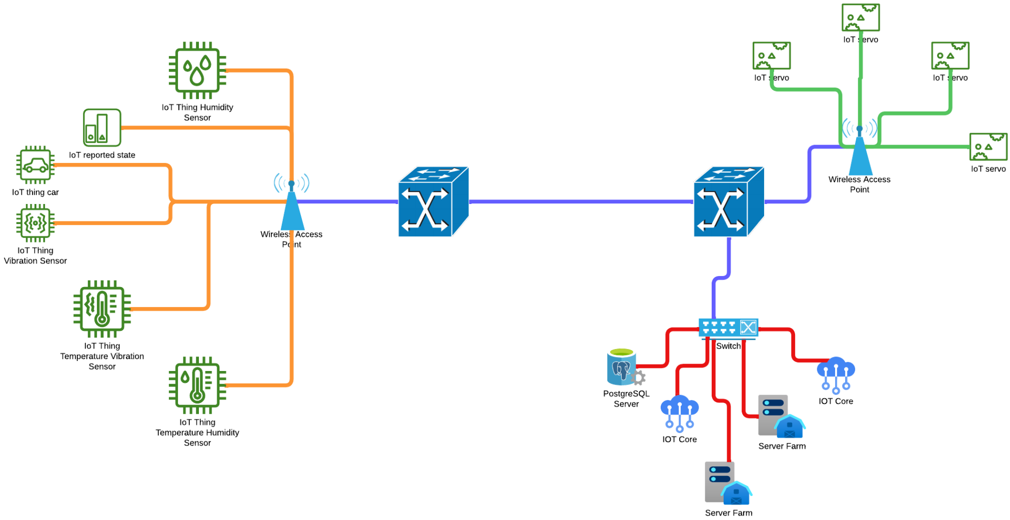
Optimality gap

4 Results



Network configuration

4 Results



Network configuration

SF duration	10 ms
Optical-wired	
B_e : Throughput	10 Gbps
a_e : Processed bits per time slot	1 Byte
τ_e : Time slot duration	0.8 ns
d_e : Propagation delay	100 us / 20 Km
WiFi6 (MCS)-5 64-QAM	
B_e : Throughput	48 Mbps
a_e : Processed bits per time slot	3 Byte
τ_e : Time slot duration	500 ns
d_e : Propagation delay	4 us

Table: Network configuration

TS-traffic

3000 TS-requests generated randomly from Application 1 and Application 2.

Application 1	
P_f : Period of the flow	1ms
δ_f : Maximum admitted delay	1ms
υ_f : Maximum admitted jitter	100 us
$\#f$: Bit transmitted by each iteration	random $[90, 120]$ bytes
E_f : Path	randomMST(src,dest)
Application 2	
P_f : Period of the flow	10ms
δ_f : Maximum admitted delay	10ms
υ_f : Maximum admitted jitter	1ms
$\#f$: Bit transmitted by each iteration	random $[900, 1200]$ bytes
E_f : Path	randomMST(src,dest)

Table: Simulation applications

Simulations

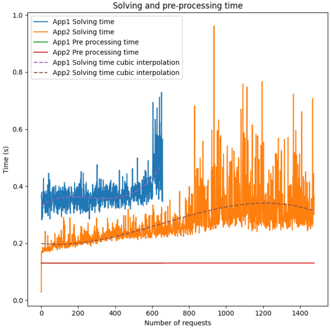
Path \ Processing mode	express	storeAndForward
wifi2wifi	C1: 71.1%	C2: 69.86%
wifi2wired	C3: 80%	C4: 77.43%

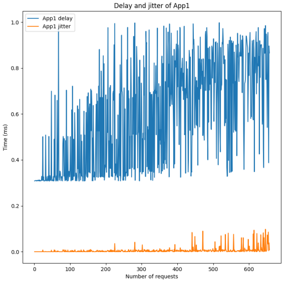
Table: Simulation configurations with feasibility percentage

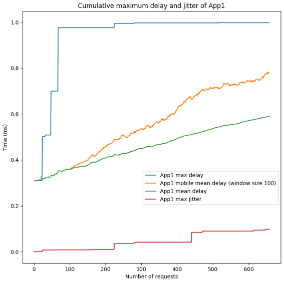
*Simulations run over an OpenStack VM with 4vCore and 32 GB of RAM.

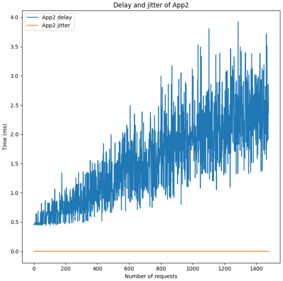
Simulation C1

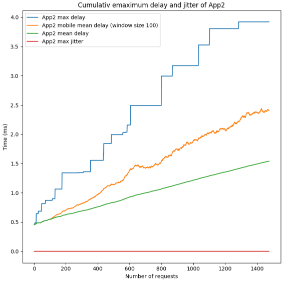
4 Results





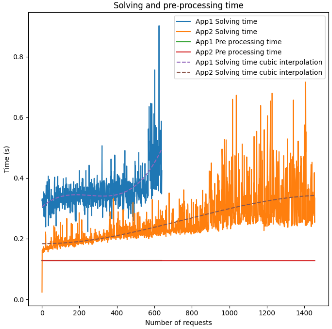


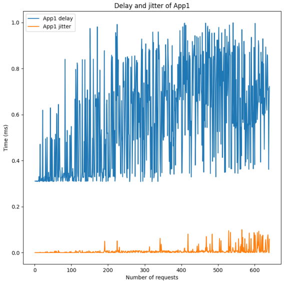


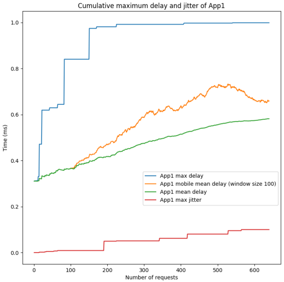


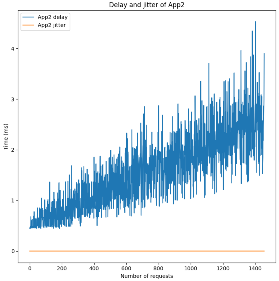
Simulation C2

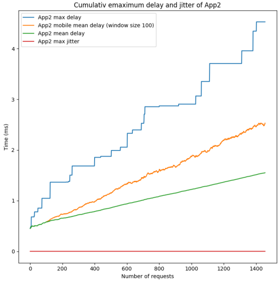
4 Results





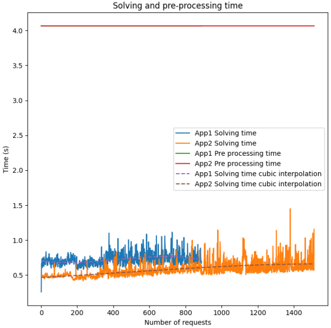


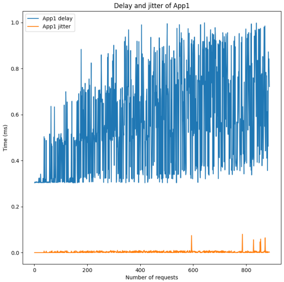


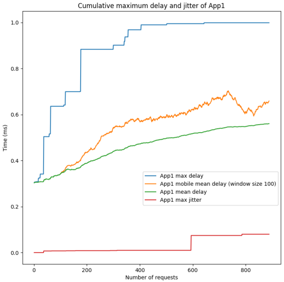


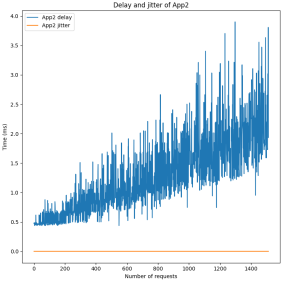
Simulation C3

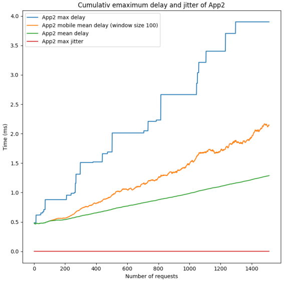
4 Results





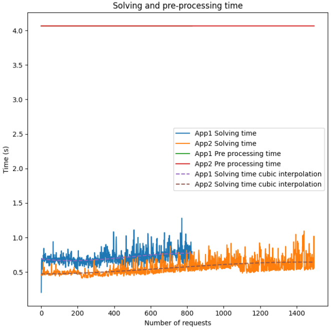


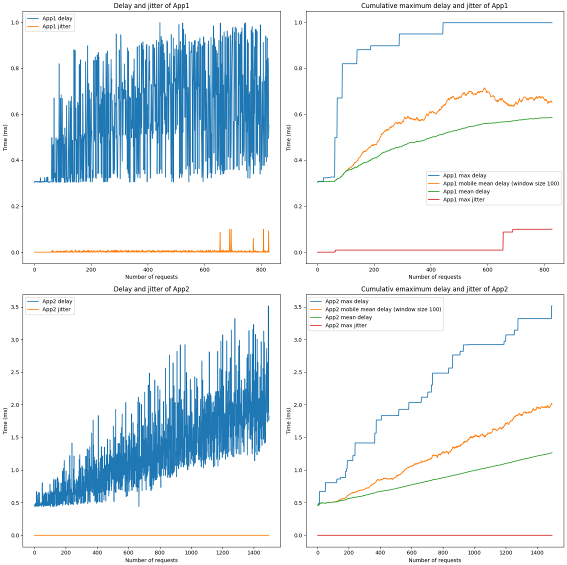




Simulation C4

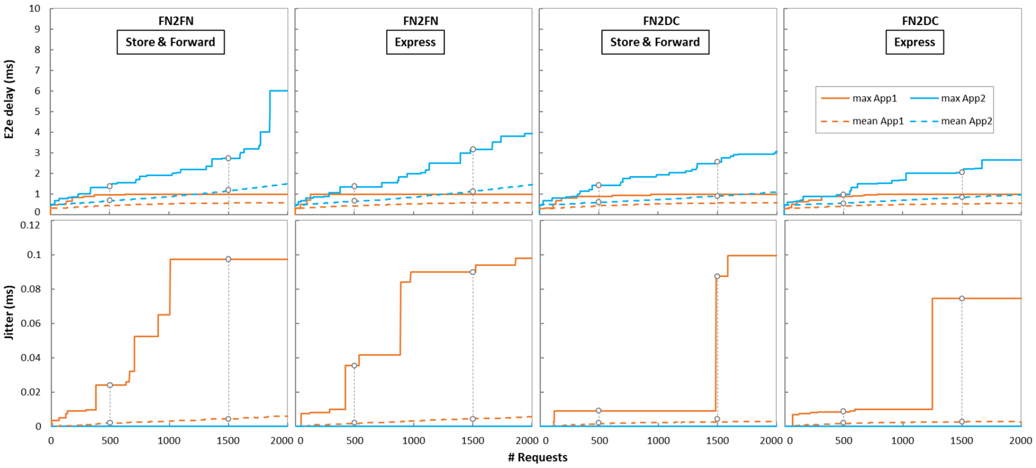
4 Results





Metrics comparison

4 Results



*Orange Application 1, Blue Application 2

Table of Contents

Work Contribution

- **Real-time approach**: TAS scheduler minimizes the data plane reconfiguration changes maximising the determinism of the network during the installation of a new scheduling data plane.
- **Heterogeneous network** TAS has to schedule with interfaces with different throughputs thus different *inertial frames of reference*. Supporting TS traffic over the providers' transport network.
- **Multi periods TS-flows**. The scheduler admits the possibility of multiple iterations of a flow in SF. **Circular shifting** optimization inspired by the *circular buffer* pattern, to increase the number of TS accommodated requests.
- **Scheduling pipelining**: TAS scheduler overlaps the assignation of resources and minimizes the latency following a code **pipeline's fashion approach**.

List of Publications

- L. Velasco, G. Graziadei, Y. El Kaisi, J. Villares, O. Muñoz, J. Vidal, and M. Ruiz, ”Provisioning of Time-Sensitive and non-Time-Sensitive Flows: from Control to Data Plane” accepted in International Workshop on Time-Sensitive and Deterministic Networking (TENSOR), collocated with the IFIP Networking conference, 2024. `https://zenodo.org/records/11393029`

Throughput wasted

Starting from C2 increasing the size of processed bits per time slot.

WiFi Processing [Bytes] \ Optical Processing [Bytes]	1	2	5	10
3	C2.T1: 0.01%	-	-	-
6	-	C2.T2: 0.25%	-	-
15	-	-	C2.T3: 0.45%	-
30	-	-	-	C2.T4: 1%

Table: Simulation configurations throughput evaluation

Future Works

Work improvement:

- Validation with a discrete events simulator (ns-3, OMNET++)
- Estimation of the required threshold between the Band of Guard and throughput wasted

Different Approaches:

- Dynamic programming algorithm (if feasible)
- Accepting dynamic TS-request (with varying required resources in time)

TSN scheduling algorithm for real-time applications in a heterogeneous network

Thank you for listening!

Any questions?

Applications in practice

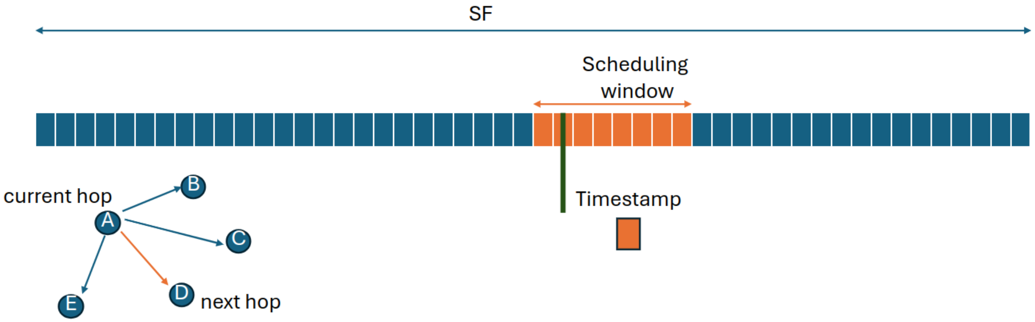
- Industrial automation
- Autonomous Vehicles and Intelligent Transportation Systems
- Energy
- Healthcare
- Finance

A TSN Connectivity Manager provides e2e control and includes:

- **Path Computation Element** (PCE) implementing algorithms with different policies computes the path of a new request
- **Time-aware Shaper** (TAS) in charge of producing scheduling for the TS flows to be deployed in the network
- **Network digital tween** that evaluates a set of KPIs of non-TS flows before new (TS or non-TS) flows are deployed.

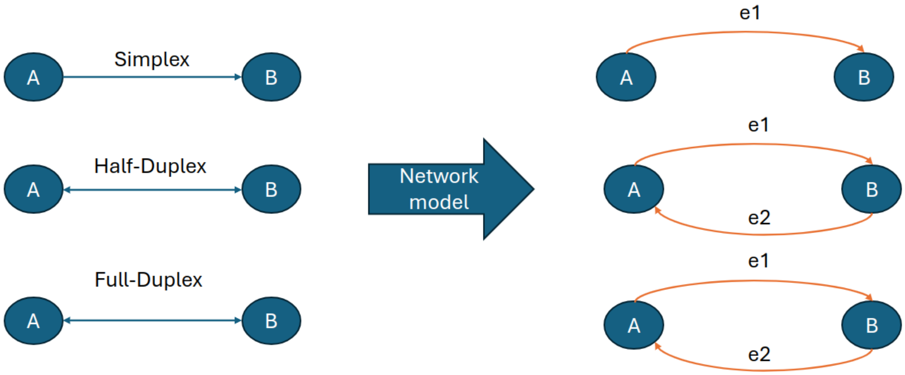
Timestamp → TS-queue

6. Given a timestamp and a known data plane, return the corresponding TS-queue.

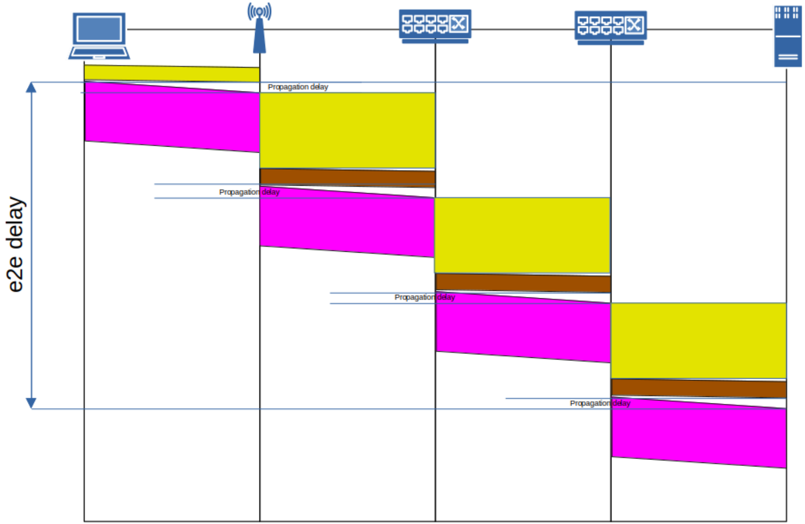


Network Transmission modes

5 Closing Discussion

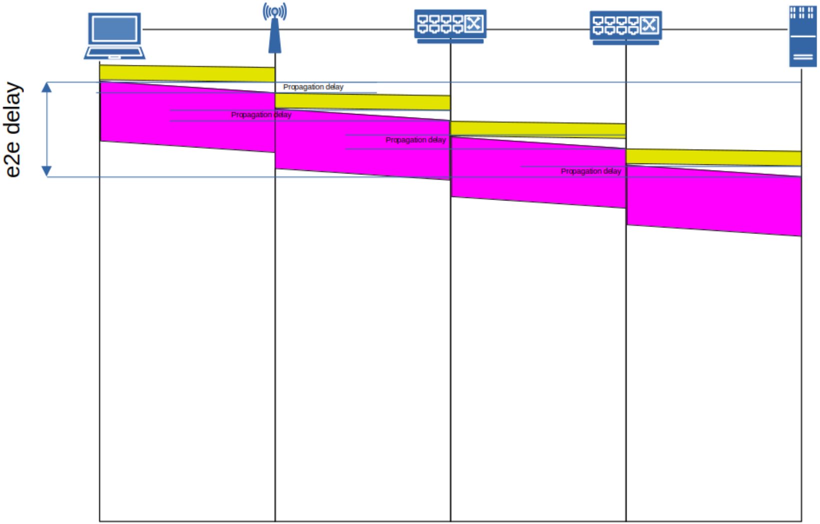


End-To-End Delay. Store and Forward processing



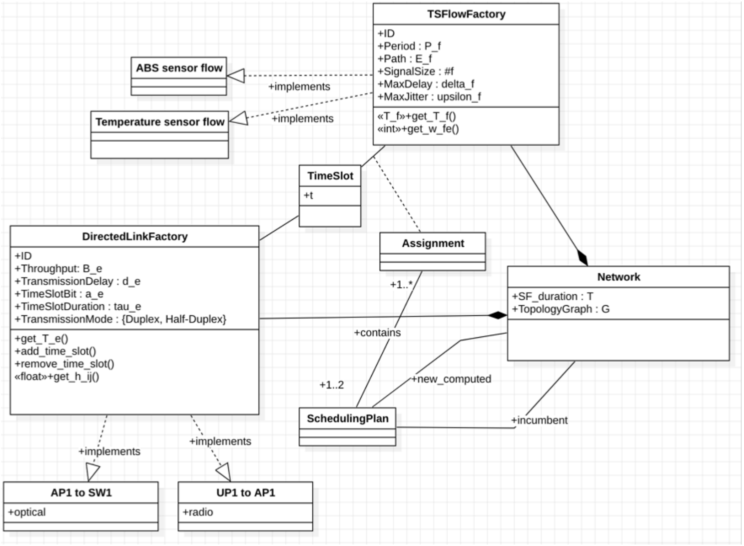
End-To-End Delay. Express processing

5 Closing Discussion



UML diagram

5 Closing Discussion



Transmission mode

Half-Duplex transmission mode

Each interface can only transmit in up-link (UL) time slots, and then the time slots in down-link (DL) are not available. Between e_1, e_2 (links for a half-duplex interface) the following relationship is valid:

$$UL_{e_1} = DL_{e_2} \tag{1}$$

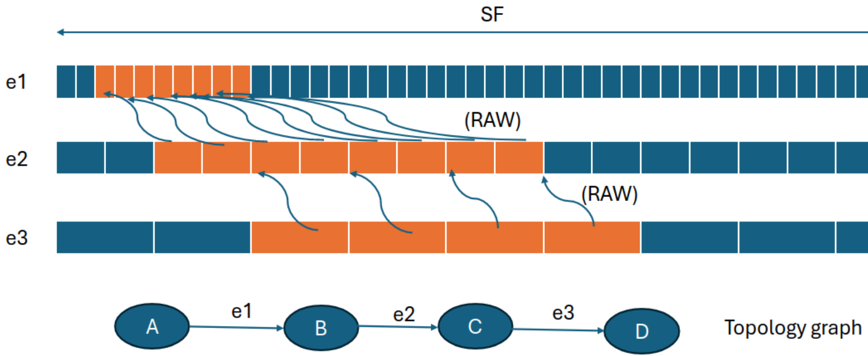
Given a heterogeneous scenario, the number of bits transmitted per time slot depends on the throughput of the network interface.

Space transformation matrix H

The square matrix H contains the space transformation coefficient h_{e_1,e_2} for each couple of network interfaces, such that to pass from interface e_1 to e_2 the coefficient $h_{e_1,e_2} = \frac{a_{e_2}}{a_{e_1}}$ is defined.

Pipeline Dependencies

5 Closing Discussion



Pipeline Optimization Algorithm

Algorithm 4 Overlapping optimization function : pipeline()

Require: $e_1, e_2 \in E, \tau_{e_1}, d_{e_1}, w_{fe_1}, \tau_{e_1}, w_{fe_2}, h_{e_1e_2}, h_{e_2e_1}, l_{e_1e_2}$

$last_{e_1} \leftarrow w_{fe_1} * \tau_{e_1}$

$first_{e_2} \leftarrow timeTransform(e_1, e_2, last_{e_1})$

if $e_1.mode == store\&forward$ **then**

  $sigma \leftarrow first_{e_2}$

else if $e_1.mode == express$ **then**

  if $\tau_{e_2} \leq \tau_{e_1} * h_{e_1e_2}$ **then**

    $sigma \leftarrow first_{e_2} - \tau_{e_2} * (w_{fe_2} - ceil(h_{e_2e_1}))$

  else

    $sigma \leftarrow l_{e_1e_2} * \tau_{e_2}$

  end if

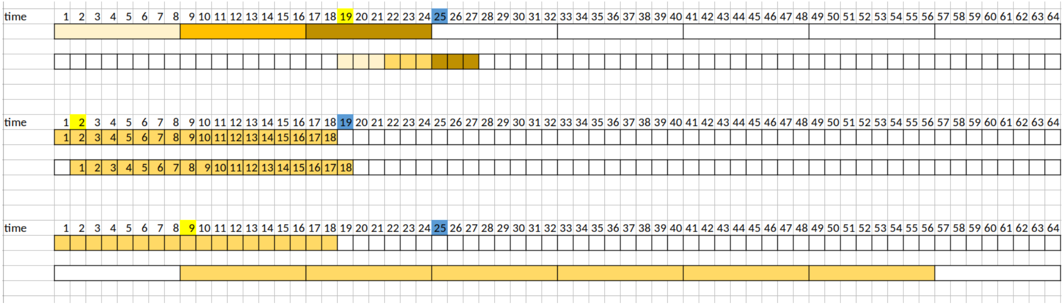
end if

$sigma \leftarrow sigma + d_{e_1}$ // adding propagation delay

  return $< sigma >$

Pipeline Optimization Example

5 Closing Discussion



*All test cases are included in the verification section at unit tests 9 (U9).

Space reduction

5 Closing Discussion

T = 40 t.u; w_fe = 4 slots; P_f = 20 t.u; T_f = 2

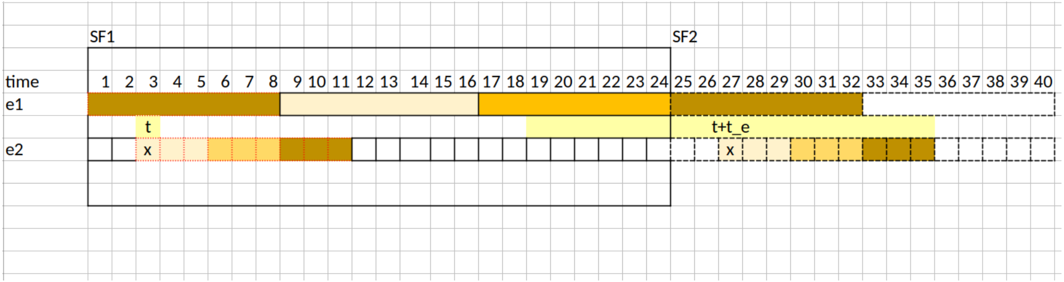
time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T_e	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1
nu_fe1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	1	1
nu_fe2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

time	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
T_e	1	0	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1
nu_fe1	1	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0
nu_fe2	1	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0

*In the graph two iterations of TS flow. In orange is the space reduction according to the pattern search criteria, and in green according to the period criteria.

Circular shifting

5 Closing Discussion



*Circular shifting example for scheduling without reconfiguration. In yellow circular correspondents. In dotted red are the merged time slots.

Input parameters

5 Closing Discussion

Parameter	Description
SF definition	
T	Duration of the scheduling superframe of the time-sensitive network.
E	Set of directed links in the topology; index e.
F	Set of time-sensitive flows. Index f.
SF_e	SF for the network interface e. $SF_e = \{s_{et}\}$, each component is equal to 1 if time slot t is available for e, i.e., both it is not already allocated to an existing TS flow and it can be used for transmission.
NSF	Network SF defined as a set of $SF_e \forall e \in E$
Network interfaces definition, index e	
G_e	Set of fixed not available time slots for the network interface e. It can be extended, in the model it covers the guard time slots and the DL time slots of Half Duplex transmissions.
T_e	The set of time slot t of the interface e
τ_e	Time duration of each $t \in T_e$
a_e	Bit processed by each $t \in T_e$
B_e	Throughput of the network interface e
d_e	Propagation delay of the network interface e
$mode_e$	Frame processing mode for the network interface e
Time sensitive flows definition, index f	
E_f	Sorted list of network interfaces in the path of the flow f
P_f	Period of the flow f
δ_f	Maximum allowed delay for f
υ_f	Maximum allowed jitter for f
New flow request definition r	
$r = \{E_r, P_r, \delta_r, \upsilon_r\}$	New scheduling request for the time-sensitive flow r.

Pre-processed parameters

Parameter	Description
T_f	Number of iterations (a.k.a. periods) in the SF for the scheduled flow f
t_e	Number of time slots in one SF for the network interface e
L	Matrix of network interface time division. $L = \{l_{e_1 e_2}\}$ where $l_{e_1 e_2}$ is the ratio between the duration of the time slot between the network interfaces e_1, e_2.
H	Matrix of network interface space division. $H = \{h_{e_1 e_2}\}$ where $h_{e_1 e_2}$ is the different speed coefficient between the interfaces e_1, e_2.
W	Matrix of scheduling window. $W = \{w_{fe}\}$ where w_{fe} is size of the flow f over the network interface $e \in E_f$.

Control plane flow-chart

5 Closing Discussion

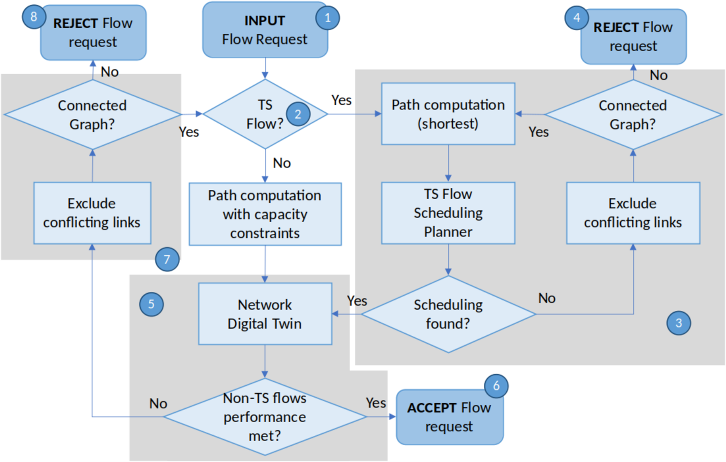
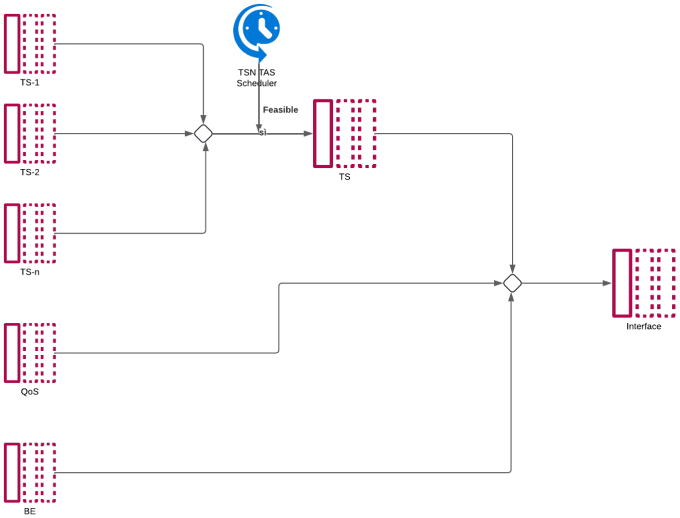


Image credits [9]

Network Interface Queuing System

5 Closing Discussion



Job Shop Scheduling Problem (JSSP)

Given a job shop environment containing several machines $M = \{M_1, M_2, ..., M_m\}$, there are several jobs $J = \{J_1, J_2, .., J_i, .., J_n\}$, each job, say J_i, contains a serial of operations $O_i = \{O_{i1}, O_{i2}, .., O_{ij}, .., O_{in}\}$ which need to be processed in a predefined technological sequence.

JSSP

Each operation is assigned a machine in M to be processed with a given processing time p_{ij}. Sequencing needs to be done for operations in all machines to minimize the maximum completion time of all jobs, i.e., to minimize the make-span.

ILP model

Integer Linear Programming (ILP) for solving a JSSP:

- Each network interface is a machine
- Each iteration of a TS request is a job
- A task is the scheduling window of a TS request over a network interface of its path

Objectives:

- Minimize the e2e estimated jitter
- Minimize the e2e estimated delay
- Minimize the number of changes (for scheduling with reconfiguration)

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

FIB

Heuristic - Construct

5 Closing Discussion

Algorithm 10 Constructive phase for heuristic solution: construct()

Require: F, I //Given a set of requests and a set of iterations for the requests

$S \leftarrow \{\}$
for $f \in F$ **do**
  $f.min\_delay \leftarrow \delta_f$
  $f.max\_delay \leftarrow 0$
  $f.jitter \leftarrow 0$
  for $i \in I$ **do**
    $C \leftarrow \{\}$
    $lt_{start} \leftarrow \frac{P_f * (i-1)}{\tau_{E_f[0]}+1}$
    $ut2_{start} \leftarrow lt_{start} + \delta_f - latency(f)$
    $ut3_{start} \leftarrow t_{E_f[0]} - w_{fE_f[0]} * (T_f - 2)$
    for $t_{start} \in [lt_{start}, min(ut3_{start}, ut2_{start})]$ **do**
      $C \leftarrow C \cup candidate(f, i, t_{start})$
    end for
    if $C == \{\}$ **then return** INFEASIBLE
    end if
    $c_{best} \leftarrow argmin_{c \in C} cost(c)$
    $S \leftarrow S \cup \{c_{best}\}$
    $assing(c_{best})$
    $f.jitter \leftarrow f.jitter + cost(c_{best}).\Delta_j$
    if $f.jitter > v.f$ **then return** INFEASIBLE
    end if
    if $f.min\_delay > cost(c_{best}).delay$ **then**
      $f.min\_delay \leftarrow cost(c_{best}).delay$
    end if
    if $f.max\_delay < cost(c_{best}).delay$ **then**
      $f.max\_delay \leftarrow cost(c_{best}).delay$
    end if
  end for
end for
$j \leftarrow max_{f \in F}(f.jitter)$
$d \leftarrow max_{f \in F}(f.max\_delay)$
  return $< S, j, d >$

Heuristic - Candidate

5 Closing Discussion

Algorithm 11 Candidate definition: candidate()

Require: f, i, t_{start}

for $e \in E_f$ **do**
  $e_{next} \leftarrow e + 1$
  $t \leftarrow findFirst(e, t_{start}, w_{fe})$
  $t_{start} \leftarrow l_{e,e_{next}} * (t * \tau_e + pipeline(e, e_{next}))$
  $c \leftarrow C \cup [(e, t)]$
end for
if $|c| <> |E_f|$ *or* $cost(c).delay > \delta_f$ **then**
    return $< \{\} >$
end if
    return $< c >$

Algorithm 12 Cost function implementation: cost()

Require: c_{fi}

$delay \leftarrow d_{E_f[-1]} + (c_{fi}[-1].t_{start} - 1) * \tau_{E_f[-1]} - P_f * (i - 1)$
if $delay > f.max\_delay$ **then**
  $\Delta_j \leftarrow delay - f.max\_delay$
else if $delay < f.min\_delay$ **then**
  $\Delta_j \leftarrow f.min\_delay - delay$
else
  $\Delta_j \leftarrow 0$
end if
    return $< \Delta_j, delay, c[0].t_{start} >$

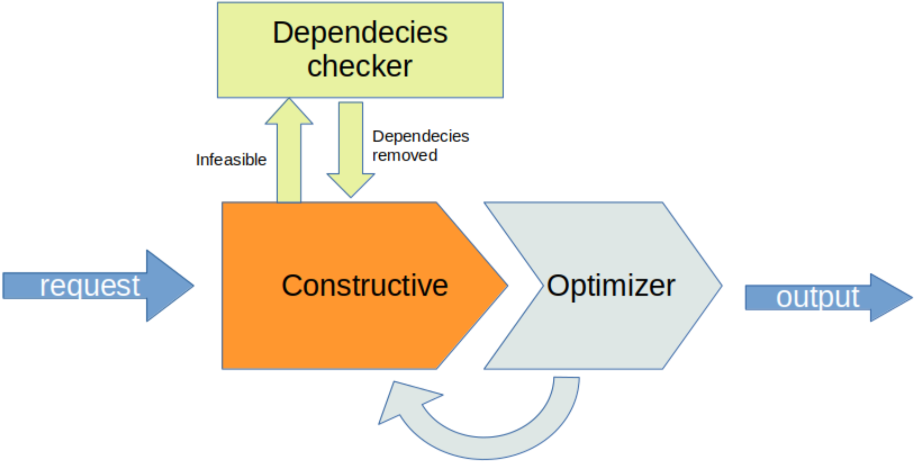
Heuristic - Jitter Optimizer

Algorithm 15 Jitter optimization: jitter_optimize()

Require: f, S

while True **do**
    $c_{min} = argmin_{\{c \in S | c.f == f\}} cost(c).delay$
    $S^{'} \leftarrow S \setminus \{c_{min}\}$
    $deallocate(c_{min})$
    $f.min\_delay \leftarrow min_{\{c \in S | c.f == f\}} cost(c).delay$
    $< \{c\}, j, d >= constructive(c_{min}.f, c_{min}.i)$
    $S^{'} \leftarrow S^{'} \cup \{c\}$
    if $f.min\_delay < cost(c_{min}).delay$ **then**
        $S \leftarrow S^{'}$
    else
        return $< S, j, d >$
    end if
end while

Reconfiguration with LS dependencies checker



Heuristic - Reconfigure

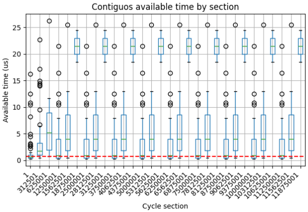
Algorithm 16 LS network reconfiguration: reconfigure()

Require: r, S

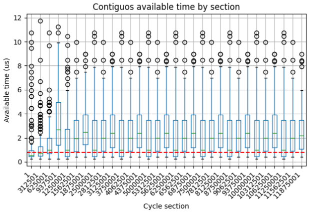
$R \leftarrow \{\}$
$S^{‘} \leftarrow S$
for $i \in [1, T_r]$ **do**
  $R \leftarrow R \cup \{(r, i)\}$
  $lt_{start} \leftarrow \frac{P_f * (i-1)}{\tau_{E_f[0]} + 1}$
  $ut2_{start} \leftarrow lt_{start} + \delta_f - latency(f)$
  $ut3_{start} \leftarrow t_{E_f[0]} - w_{fE_f[0]} * (T_f - 2)$
  $e_{src} \leftarrow E_f[0]$
  for $t_{start} \in [lt_{start}, min(ut3_{start}, ut2_{start})]$ **do**
    if $T_{e_{src}}[t_{start} - 1] == 0$ **then**
      $c \leftarrow findAssignment(S, e_{src}, t_{start})$
      $S^{‘} \leftarrow S^{‘} \setminus \{c\}$
      $deallocate(c.f, c.i)$
      $R \leftarrow R \cup \{(c.f, c.i)\}$
    end if
  end for
end for
$R \leftarrow sort(R, f : \frac{\delta_f}{|E_f|}, ASC)$
$S^{“}, j, d = constructive(R.flows, R.iterations)$
if $S^{“}$ == INFEASIBLE **then** // The request is blocked
  return return $< S, j, d >$
end if
$S, j, d = merge(S^{“}, S^{‘})$
  return $< S, j, d >$

Network loading and scheduling fragmentation - C2.T1

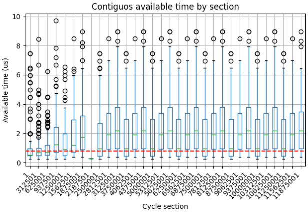
5 Closing Discussion



(a) 1 Byte, 400 req.



(b) 1 Byte, 1200 req.

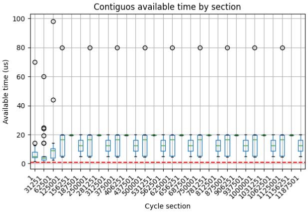


(c) 1 Byte, 2000 req.

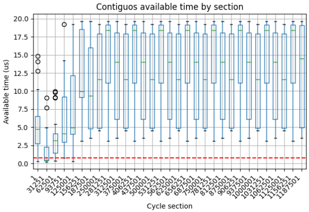
*Interface 1 of transport network

Network loading and scheduling fragmentation - C2.T2

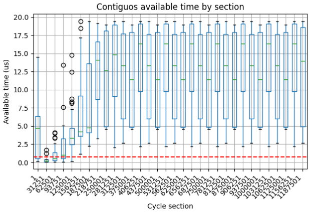
5 Closing Discussion



(a) 10 Byte, 400 req.



(b) 10 Byte, 1200 req.



(c) 10 Byte, 2000 req.

*Interface 1 of the transport network

References

Daniel Bezerra, Assis T. de Oliveira Filho, Iago Richard Rodrigues, Marrone Dantas, Gibson Barbosa, Ricardo Souza, Judith Kelner, and Djamel Sadok.
A machine learning-based optimization for end-to-end latency in tsn networks.
Computer Communications, 195:424–440, 2022.

Bahar Houtan, Mohammad Ashjaei, Masoud Daneshtalab, Mikael Sjödin, and Saad Mubeen.
Synthesising schedules to improve qos of best-effort traffic in tsn networks.
In *Proceedings of the 29th International Conference on Real-Time Networks and Systems*, RTNS '21, page 68–77, New York, NY, USA, 2021. Association for Computing Machinery.

Hyeong Jun Kim, Kyoung Chang Lee, and Suk Lee.
A genetic algorithm based scheduling method for automotive ethernet.
In *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–5, 2021.

Jonatan Krolikowski, Sébastien Martin, Paolo Medagliani, Jérémie Leguay, Shuang Chen, Xiaodong Chang, and Xuesong Geng.
Joint routing and scheduling for large-scale deterministic ip networks.
Computer Communications, 165:33–42, 2021.

Yinzhi Lu, Liu Yang, Simon X. Yang, Qiaozhi Hua, Arun Kumar Sangaiah, Tan Guo, and Keping Yu.
An intelligent deterministic scheduling method for ultralow latency communication in edge enabled industrial internet of things.
IEEE Transactions on Industrial Informatics, 19(2):1756–1767, 2023.

Junhong Min, Yongjun Kim, Moonbeom Kim, Jeongyeup Paek, and Ramesh Govindan.
Reinforcement learning based routing for time-aware shaper scheduling in time-sensitive networks.
Computer Networks, 235:109983, 2023.

References

5 Closing Discussion

Zaiyu Pang, Xiao Huang, Zonghui Li, Sukun Zhang, Yanfen Xu, Hai Wan, and Xibin Zhao.
Flow scheduling for conflict-free network updates in time-sensitive software-defined networks.
IEEE Transactions on Industrial Informatics, 17(3):1668–1678, 2021.

Ammad Ali Syed, Serkan Ayaz, Tim Leinmüller, and Madhu Chandra.
Dynamic scheduling and routing for tsn based in-vehicle networks.
In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2021.

L. Velasco, G. Graziadei, Y. El Kaisi, J. Villares, O. Muñoz, J. Vidal, , and M. Ruiz.
Provisioning of time-sensitive and non-time-sensitive flows: from control to data plane.
IFIP Networking 2024, TENSOR, 2024.

Xiaolong Wang, Haipeng Yao, Tianle Mai, Tianzheng Nie, Lin Zhu, and Yunjie Liu.
Deep reinforcement learning aided no-wait flow scheduling in time-sensitive networks.
In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 812–817, 2022.

Yang Wang, Jidong Chen, Wei Ning, Hao Yu, Shimei Lin, Zhidong Wang, Guanshi Pang, and Chao Chen.
A time-sensitive network scheduling algorithm based on improved ant colony optimization.
Alexandria Engineering Journal, 60(1):107–114, 2021.