Out-of-sync Schedule Robustness for Time-sensitive Networks

Silviu S. Craciunas, Ramon Serna Oliver
TTTech Computertechnik AG
9-11 June 2021
TTTech Group

- Founded in 1998, headquartered in Vienna, Austria, with 19 offices in 14 countries worldwide.
- Develops safe networked computing platforms for a more connected, automated and sustainable world.
- Is based on 20+ years of expertise in scheduled data communication and fault-tolerance principles.
- Strong collaboration with academia; 36 ongoing research projects.

2300+ employees in TTTech Group.
What is a Time-Sensitive Network (TSN)?

Time Sensitive Networking covers a set of Ethernet sub-standards and amendments currently defined in the IEEE 802.1 TSN task group.

- Critical traffic guarantees through *time synchronization* and *scheduled frame transmission*
- TSN supports the coexistence of critical and non-critical traffic over the same communication backbone.
TSN (Qbv) switch

Port A (ingress)

Switching fabric

Priority filter

Priority 7

Priority 6

Priority 5

Priority 4

Port C (egress)

\[ T = 0 \]
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 build off-line taking into account the maximum possible clock deviation (precision) when all clocks are synchronized.

The schedule enforces a deterministic behaviour of frame transmission and reception.
Clock drift

- Each clock $C_i$ has a drift rate $\rho_i$
- The maximum clock drift in the network is $\rho_{\text{max}} = \max_i \{\rho_i\}$
- Typical values of $\rho_{\text{max}}$ are 50 - 100ppm, i.e., between 50 and 100 $\mu$s/sec
- All clocks need to be synchronized with a certain rate - synchronization interval ($I$)
- The envelope, $I$, and $\rho_{\text{max}}$ determine the value of the network precision $\delta$
- The precision is a safe upper bound on the deviation between any two clocks in the network
Time synchronization in TSN

IEEE 802.1AS

- clock synchronization protocol that provides a common clock reference for all network devices called GrandMaster (GM)
- each clock is at most $\delta$ (precision) away from the GM time
- Best Master Clock Algorithm (BMCA) constructs the synchronization spanning tree with the GM as root node
- time is propagated from the root to the leaves
- each bridge corrects the received time by adding the propagation delay and residence time in the bridge and forwards the corrected time to the next nodes in the tree
- if the GM node fails, a new GM has to be elected, and the spanning tree has to be recreated via the BCMA

IEEE 802.1AS-rev

- is an update to 802.1AS
- introduces multiple domains:
  - domains are fully independent
  - separate BMCA
- introduces multiple time scales
- introduces redundancy: configure redundant paths and redundant GMs (hot standby)
Synchronized deterministic network

ES₁

ES₂

SW

SW timeline

Copyright © TTTech Computertechnik AG. All rights reserved.
Perfectly synchronized network

ES₁ time

ES₂ time

SW time
Synchronized network

ES₁ time

ES₂ time

SW time

Copyright © TTTech Computertechnik AG. All rights reserved.
Synchronization loss

ES₁ time

ES₂ time

SW time
Can we compute an upper bound on the time until the network is resynchronized in 802.1AS-rev?

\[ t_{\text{resync}} = \delta_t - \delta_{\text{hop}} \times N_G \]

- \( t_{\text{resync}} \): Out-of-sync interval
- \( \delta_t \): Timeout bound for out-of-sync detection
- \( \delta_{\text{hop}} \): Upper bound for the BMCA to run and propagate information on each node
- \( N_G \): Number of hops for the longest path out of all the possible spanning trees
- \( \times \): Safe upper bound of 1 sec per hop
- Configuration item in 802.1AS-rev: announceReceiptTimeout \times announceInterval
Out-of-sync drift

\[ \Delta_s = 2 \times \rho_{\text{max}} \times t_{\text{resync}} \]

Example:
- \( N_G = 3, \delta_t = 3s, \delta_{\text{hop}} = 1s, \) and \( \rho_{\text{max}} = 100\text{ppm} \)
- the upper bound on the deviation between any two clocks following a GM failure at the point of re-synchronization is 1200\(\mu\)s

\[ \delta + \Delta_s \]

If we can generate a schedule with an extended precision parameter, we can effectively maintain determinism even when sync is temporarily lost → schedule robustness
Schedule robustness

It is trivial to extend the relevant constraints from our previous work* to include the robustness parameter added to the precision when generating the schedule tables.

\[ \forall s_i \in S, \forall (v_a, v_x), (v_x, v_b) \in R_i : \\
\phi_i^{(v_x, v_b)} = (\phi_i^{(v_a, v_x)} + l_i^{(v_a, v_x)}) \geq \delta + \Delta_s. \]

\[ \forall \alpha \in \left[ 0, hp_i^2 / T_i \right], \forall \beta \in \left[ 0, hp_i^2 / T_j \right] : \\
(\phi_j^{(v_y, v_a)} + \beta \times T_j - \phi_i^{(v_a, v_b)} - \alpha \times T_i \geq \delta + \Delta_s) \lor \\
(\phi_i^{(v_x, v_a)} + \alpha \times T_i - \phi_j^{(v_a, v_b)} - \beta \times T_j \geq \delta + \Delta_s). \]

\[ \forall s_i \in S : \phi_i^{\text{dest}(s_i)} + l_i^{\text{dest}(s_i)} - \phi_i^{\text{src}(s_i)} \leq D_i - (\delta + \Delta_s). \]

* our previous work:
  - S.S. Craciunas, R. Serna Oliver, M. Chmelik, and W. Steiner - Scheduling Real-Time Communication in IEEE 802.1 Qbv Time Sensitive Networks
  - R. Serna Oliver, S.S. Craciunas, and W. Steiner - IEEE 802.1 Qbv Gate Control List Synthesis using Array Theory Encoding
Design-space exploration

- we transform the out-of-sync drift $\Delta_s$ to be a variable that is computed by the scheduler
- maximizing the out-of-sync drift $\Delta_s$ can help mitigate cascading failures
- selecting a value for one parameter will constrain the possible values for the other dimensions
- the easiest parameter to change is the out-of-sync detection bound $\delta_t$
- we show the configuration space for different example networks below
Schedule tool and experiment setup

**Satisfiability Modulo Theories**

- satisfiability of logical formulas in first-order formulation
- background theories $\mathcal{L}(\mathbb{Z})$ $\mathcal{BV}$
- variables $x_1, x_2, \ldots, x_n$
- logical symbols $\lor, \land, \neg, (, )$
- non-logical symbols $+, =, \%, \leq$
- quantifiers $\exists, \forall$
- optimization criteria: Optimization Modulo Theories [Bjørner@TACAS15]

---

- Z3 SMT/OMT solver v.4.8.10
- 2 dedicated queues for 802.1Qbv
- macrotick fixed at 1µs
- a constant link latency of 1µs
- homogeneous link speeds of 1Gbps
- Intel i7-8650U CPU @1.90GHz with 16GB RAM
The higher the link utilization, the less robustness can be added

！ Schedulability is reduced

<table>
<thead>
<tr>
<th>$\rho_{\text{max}}$ [ppm]</th>
<th>100</th>
<th>100</th>
<th>50</th>
<th>50</th>
<th>5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_t$ [s]</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\Delta_s$ [\mu s]</td>
<td>1200</td>
<td>800</td>
<td>600</td>
<td>400</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>Max util. [%]</td>
<td>5.76</td>
<td>5.76</td>
<td>5.76</td>
<td>5.76</td>
<td>5.76</td>
<td>5.76</td>
</tr>
<tr>
<td>Runtime [ms]</td>
<td>437</td>
<td>359</td>
<td>343</td>
<td>390</td>
<td>389</td>
<td>422</td>
</tr>
</tbody>
</table>

| Schedulability            | true | true | true | true | true | true |

<table>
<thead>
<tr>
<th>$\rho_{\text{max}}$ [ppm]</th>
<th>100</th>
<th>100</th>
<th>50</th>
<th>50</th>
<th>5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_t$ [s]</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\Delta_s$ [\mu s]</td>
<td>1200</td>
<td>800</td>
<td>600</td>
<td>400</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>Max util. [%]</td>
<td>11.52</td>
<td>11.52</td>
<td>11.52</td>
<td>11.52</td>
<td>11.52</td>
<td>11.52</td>
</tr>
<tr>
<td>Runtime [ms]</td>
<td>219</td>
<td>312</td>
<td>391</td>
<td>407</td>
<td>406</td>
<td>422</td>
</tr>
</tbody>
</table>

| Schedulability            | false | false | true | true | true | true |
Experiments – end-to-end latency

Figure 6. End-to-end latency with and without minimization objectives.

- $P_1$: $10, 20$ ms
- $P_2$: $50, 75, 150$ ms
- $P_3$: $10, 25, 50, 100$ ms

**Table I**

<table>
<thead>
<tr>
<th>$P$</th>
<th>$e_{2e}$ (left y-axis)</th>
<th>$e_{2e}$ min (left y-axis)</th>
<th>runtime (right y-axis)</th>
<th>runtime min (right y-axis)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1$ ms</td>
<td>40 (\mu s)</td>
<td>60 (\mu s)</td>
<td>400 (\mu s)</td>
<td>1200 (\mu s)</td>
</tr>
<tr>
<td>$5$ ms</td>
<td>80 (\mu s)</td>
<td>100 (\mu s)</td>
<td>800 (\mu s)</td>
<td>2000 (\mu s)</td>
</tr>
<tr>
<td>$10$ ms</td>
<td>120 (\mu s)</td>
<td>140 (\mu s)</td>
<td>1200 (\mu s)</td>
<td>3000 (\mu s)</td>
</tr>
</tbody>
</table>

**Figure 7. Scheduler runtime with and without optimization objectives.**

- Non-optimized (runtime) and optimized (runtime min) variants,
- Logarithmic right y-axis using circles and squares for the values.
- Irrespective of the period and non-optimized and the optimized e2e latencies increase significantly smaller than that of minimizing the end-to-end latency.

Copyright © TTTech Computertechnik AG. All rights reserved.
Experiments - schedule synthesis time

**Table I**

<table>
<thead>
<tr>
<th>CONFIGURATIONS</th>
<th>( \delta = 1 , \mu s ) maximize ( \Delta_s )</th>
<th>( \Delta_s = )</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1={10, 20}ms</td>
<td>( \Delta_s = 1.42 , ms )</td>
<td>( \Delta_s = 1.98 , ms )</td>
</tr>
<tr>
<td>P2={50, 75, 150}ms</td>
<td>( \Delta_s = 7.13 , ms )</td>
<td></td>
</tr>
<tr>
<td>P3={10, 25, 50, 100}ms</td>
<td>( \Delta_s = 5.76 , ms )</td>
<td></td>
</tr>
</tbody>
</table>

Copyright © TT Tech Computertechnik AG. All rights reserved.
Thank you!

Email: silviu.craciunas@tttech.com