Schedule Synthesis of Timed Sensitive Network using SMT Solver

Jin Hyun Kim

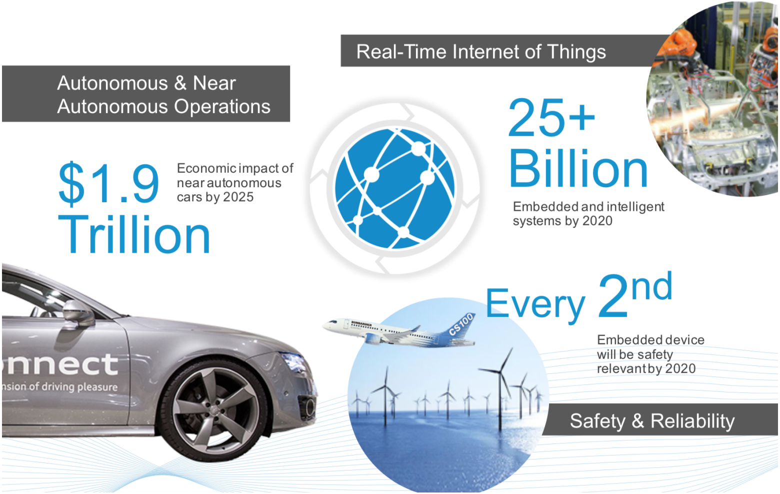
Gyeongsang Univ. *jin.kim@gnu.ac.kr*

September 10, 2019

Overview

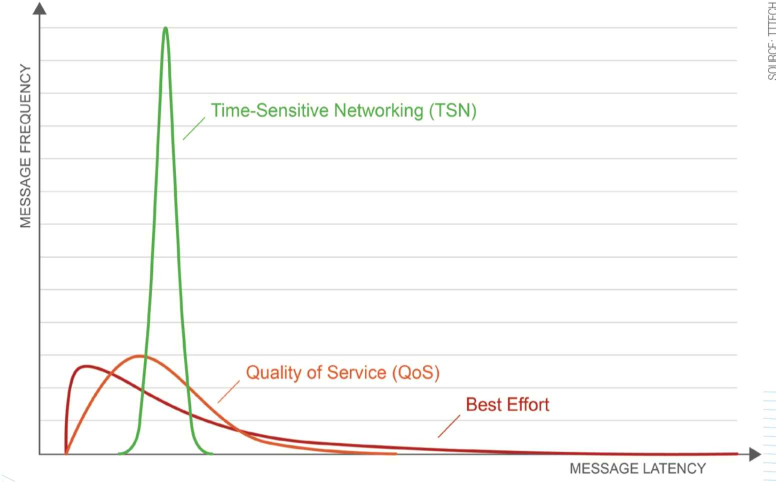
Context

Cyber-Phisical Systems



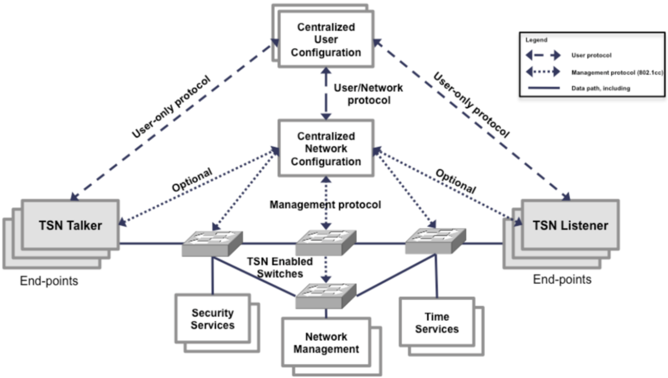
Context

Time-sensitive networking



Context

Time-Sensitive Network (TSN)



Time-Sensitive Network

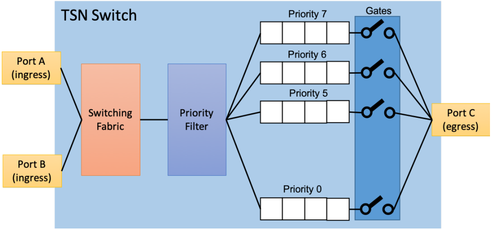


Figure: TSN Switch

Time	G_7	G_6	G_5	G_4	G_3	G_2	G_1	G_0
t_0	O	C	C	C	C	C	C	C
t_1	C	C	C	O	C	C	C	C
t_2	C	O	C	C	C	C	C	C
t_3	C	C	C	C	O	C	C	C
t_4	C	C	C	C	C	O	C	C
t_5	C	C	C	C	C	O	O	C
t_6	C	C	C	C	C	C	C	C

time

Figure: TSN Gate Control List

Problem and Challenge

- Problems

How to a synthesis schedule of TSN such that every frame of streams satisfy maximum latency and jitter constraints?

How to optimize GCL in terms of system jitters, *i.e.*, minimization of weighted jitters of each stream?
How to minimize GCL to meet hardware resource limitations?

- Challenges

Scheduling problem of MULTIPLE scheduling systems, *i.e.*, dependent on each others,

GCL is dependent on limited HW resources.

Underlying Logic Systems

First-order theory of arrays ($\mathcal{T}_A$)

Two axiom of array theory:

$$\begin{aligned}\forall a : array, \forall i, j : index, \forall x : elem \\ i = j \rightarrow a\langle i \leftarrow x\rangle[j] = x \\ i \neq j \rightarrow a\langle i \leftarrow x\rangle[j] = a[j]\end{aligned} \tag{1}$$

System Model

A network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$

If a link is between v_i and v_j, then $(v_i, v_j), (v_j, v_i) \in \mathcal{E}$,

A stream s ($\in S$) is defined by a tuple $\left\langle C_i, T_i, L_i, J_i \right\rangle$, respectively, representing message size, the period, the maximum e2e latency, and the maximum jitter.

$\mathcal{R}_i = [(v_1, v_2), ..., (v_{n-1}, v_n)]$ denotes the route of stream s_i.

$f_{i,j}^{(a,b)} (\in F^{(a,b)})$ denotes an instance of a stream s_i over link $(a, b) \in \mathcal{E}$, *i.e.*, j_{th} frame of stream s_i in between a and b.

$l_i^{(a,b)}$ denotes a frame transmission latency.

System Model

$\mathcal{W}^{(a,b)}$ denotes the maximum number of scheduled windows for the egress port of the edge between a and b.

$\omega_{i,j}^{a,b}$ denotes the index of the scheduled window for each frame $f_{i,j}^{(a,b)} \in F^{(a,b)}$.

Two sorted arrays, $\phi^{(a,b)}$ and $\tau^{(a,b)}$, denote the opening and closing time instances of the indexed windows for egress port associated to link (a, b).

$G(Q) = \{\mathcal{N}, \mathcal{N}_{tt}, \mathcal{N}_{prio}\}$ denotes, a queue configuration.

- $\mathcal{N}$: The total queue numbers per egress port,
- $\mathcal{N}_{tt}$: The number of queues for scheduled traffic, and
- $\mathcal{N}_{prio}$: The number of queues of priority queues for non-scheduled traffic.

System Model

κ is a sorted array denoting the assigned queue for each windows.

The relation between frame, window index and the tuple of open and close time instances:

$$f_{i,j}^{(a,b)} \rightarrow \omega_{i,j}^{a,b}$$
$$\omega_{i,j}^{a,b} \rightarrow (\phi^{(a,b)}[\omega_{i,j}^{a,b}], \tau^{(a,b)}[\omega_{i,j}^{a,b}])$$

The relation between κ and window index:

$$\kappa[\omega_{i,j}^{a,b}] = Qid$$

Running Example: Time-Sensitive Network

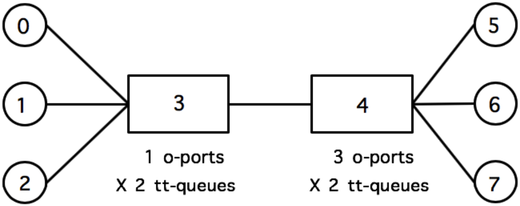
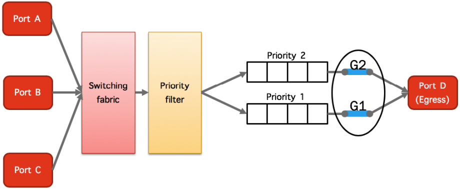


Figure: TSN topology of running example

$$S = \{s_1 = (10, 25, 25, 10), s_2 = (35, 50, 50, 10), s_3 = (100, 100, 100, 10)\}$$

Running Example: Switch 3

Inflow streams at switch 3 are $s_i = \{f_{i,j}^{(3,4)}, f_{i,j+1}^{(3,4)}, \ldots\}$ and $s_{i+1} = \{f_{i+1,j}^{(3,4)}, f_{i+1,j+1}^{(3,4)}, \ldots\}$



Expected results from SMT solver

$$\begin{aligned}\{ \quad & (\phi[w_{i,j}] = 0, \tau[w_{i,j}] = 1) \\ , \quad & (\phi[w_{i,j+1}] = 1, \tau[w_{i,j+1}] = 2) \\ , \quad & (\phi[w_{i+1,j}] = 2, \tau[w_{i+1,j}] = 3) \\ , \quad & (\phi[w_{i+1,j+1}] = 3, \tau[w_{i+1,j+1}] = 4)\ldots\}\end{aligned}$$

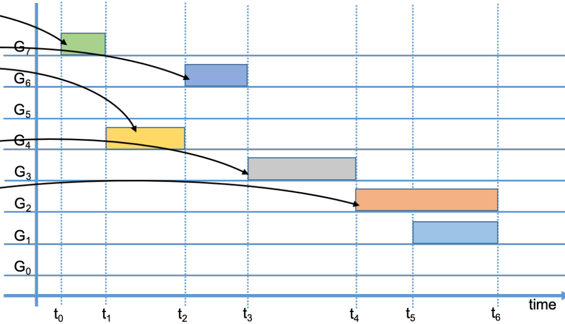
$$\kappa[w_{i,j}] = 2, \kappa[w_{i,j+1}] = 2$$
$$\kappa[w_{i+1,j}] = 1, \kappa[w_{i+1,j+1}] = 1$$

Tick	G2:G1
T1	oC
T2	oC
T3	Co
T4	Co
...	...

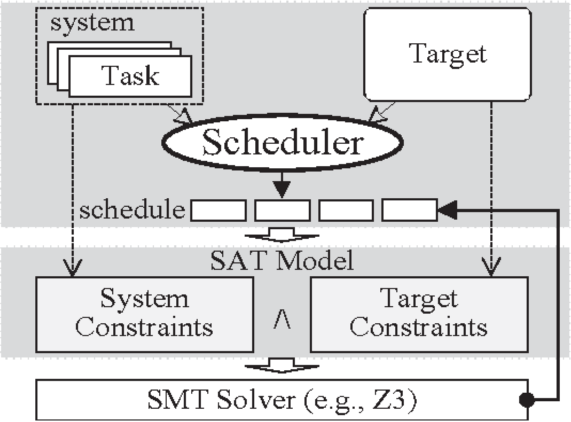
TSN schedule vs Gate Control List

Time	G_7	G_6	G_5	G_4	G_3	G_2	G_1	G_0
t_0	O	C	C	C	C	C	C	C
t_1	C	C	C	O	C	C	C	C
t_2	C	O	C	C	C	C	C	C
t_3	C	C	C	C	O	C	C	C
t_4	C	C	C	C	C	O	C	C
t_5	C	C	C	C	C	O	O	C
t_6	C	C	C	C	C	C	C	C

time



SMT-based Schedule Synthesis



TSN Constraints

- Technology constraints
- User constraints

Technology Constraints

Constraint 1-1: Well-defined windows constraints 1

The open and close event of each window should be greater than or equal to 0 and less then or equal to the schedule cycle (T_s, LCM of stream's periods).

$$\forall (a,b) \in \mathcal{E} : \forall k \in \{1, ..., \mathcal{W}^{(a,b)}\} : \left(\phi^{(a,b)}[k] \geq 0\right) \wedge \left(\tau^{(a,b)}[k] < T_s\right)$$

Constraint 1-2: Well-defined windows constraints 2

Each window is assigned to an egress queue in the range $[0, ..\mathcal{N}_{tt}]$

$$\forall (a,b) \in \mathcal{E} : \forall k \in \{1, ..., \mathcal{W}^{(a,b)}\} : 0 \leq \kappa^{(a,b)}[k] < \mathcal{N}_{tt}$$

Technology Constraints

Constraint 2: Stream Instance Constraints

For a stream s_i through (a, b), each frame instance of s_i has a low bound of the open event $(j \times T_i)$ and a upper bound of window close event $((j + 1) \times T_i)$.

$$\forall s_i \in S : \forall (a, b) \in \mathcal{E} : \forall j \in \left[0, \frac{T_s}{T_i} - 1\right] :$$
$$\left(\phi^{(a,b)}[w_{i,j}^{(a,b)}] \geq j \times T_i\right) \wedge$$
$$\left(\tau^{(a,b)}[w_{i,j}^{(a,b)}] < (j + 1) \times T_i\right)$$

Technology Constraints

Constraint 3-1: Stream Instance Constraints

The frame of a stream $l_i^{(a,b)}$ should always arrive at exactly same time interval T_i.

$$\forall s_i \in S : \forall (a,b) \in \mathcal{E} : \forall j \in \left[0, \frac{T_s}{T_i} - 2\right] :$$
$$\left(\phi^{(a,b)}[w_{i,j+1}^{(a,b)}] - \phi^{(a,b)}[w_{i,j}^{(a,b)}] = T_i\right).$$

Technology Constraints

Constraint 3-2: Stream Instance Constraints

The close event should happen $l_i^{(a,b)}$ time units after the open event happens.

$$\forall s_i \in S : \forall (a,b) \in \mathcal{E} : \forall j \in \left[0, \frac{T_s}{T_i} - 1\right] :$$
$$\left(\tau^{(a,b)}[w_{i,j}^{(a,b)}] - \phi^{(a,b)}[w_{i,j}^{(a,b)}] = l_i^{(a,b)}\right).$$

Technology Constraints

Constraint 4: Ordered Windows Constraint:

Two frames cannot share the egress port, *i.e.*, two windows sharing a gate (egress port) cannot open at the same time.

$$\forall (a,b) \in \mathcal{E} : \forall i, j \in \{1, \ldots, \mathcal{W}^{(a,b)}\}, i \neq j :$$
$$\left(\tau^{(a,b)}[i] \leq \phi^{(a,b)}[j]\right) \vee \left(\tau^{(a,b)}[j] \leq \phi^{(a,b)}[i]\right)$$

Technology Constraints

Constraint 5: Ordered Windows Constraint:

The frame following another frame should be after the close event.

$$\forall (a,b) \in \mathcal{E} : \forall i \in \{1, ..., \mathcal{W}^{(a,b)} - 1\} :$$
$$\left(\tau^{(a,b)}[i] \leq \phi^{(a,b)}[i+1]\right),$$

Technology Constraints

Constraint 6: Ordered Windows Constraint:

For an egress port, the first open event appears at time $t \geq 0$ and the last close event appears before T_s.

$$\forall (a, b) \in \mathcal{E} :$$
$$(\phi^{(a,b)}[1] \geq 0) \wedge (\tau^{(a,b)}[\mathcal{W}^{(a,b)}] < T_s),$$

Technology Constraints

Constraint 7: Frame-to-Window Assignment Constraint:

For all frames in a switch, the window index should be greater or equal to 1 and less then or equal to the maximum window index.

$$\forall (a,b) \in \mathcal{E} : \forall f_{i,j}^{(a,b)} \in \mathcal{F}^{(a,b)} :$$
$$(w_{i,j}^{(a,b)} \geq 1) \wedge (w_{i,j}^{(a,b)} \leq \mathcal{W}^{(a,b)}).$$

Technology Constraints

Constraint 8: Window Size Constraints

Initially, store the uninterpreted term for each open variable in the respective position of the close array:

$$\forall (a,b) \in \mathcal{E} : \forall k \in \{1, ..., \mathcal{W}^{(a,b)}\} : \\ \tau^{(a,b)}\langle k \leftarrow \phi^{(a,b)}[k] \rangle$$

It sets all close events equal to the open event at the same index.

Technology Constraints

Constraint 9: Window Size Constraints

The close event updates its time instant with the open event time instant plus the transmission duration.

$$\forall (a,b) \in \mathcal{E} : \forall f_{i,j}^{(a,b)} \in \mathcal{F}^{(a,b)} :$$
$$\tau^{(a,b)} \langle w_{i,j}^{(a,b)} \leftarrow \tau^{(a,b)}[w_{i,j}^{(a,b)}] + l_i^{(a,b)} \rangle.$$

Technology Constraints

Constraint 10: Stream Constraint

The frames belonging to the same stream must be scheduled sequentially in time along the route.

$$\forall s_i \in S : \forall(\mathcal{V}_k, \mathcal{V}_{+1}) \in \mathcal{R}_i, k \in \{1, ..., n-2\} :$$
$$\forall f_{i,j}^{(\mathcal{V}_k,\mathcal{V}_{k+1})} \in \mathcal{F}^{(\mathcal{V}_k,\mathcal{V}_{k+1})} : \forall f_{i,j}^{(\mathcal{V}_{k+1},\mathcal{V}_{k+2})} \in \mathcal{F}^{(\mathcal{V}_{k+1},\mathcal{V}_{k+2})} :$$
$$\tau^{(\mathcal{V}_k,\mathcal{V}_{k+1})}[w_{i,j}^{(\mathcal{V}_k,\mathcal{V}_{k+1})}] + \delta \leq \phi^{(\mathcal{V}_{k+1},\mathcal{V}_{k+2})}[w_{i,j}^{(\mathcal{V}_{k+1},\mathcal{V}_{k+2})}]$$

For a frame $f_{i,j}$, the following port $(\mathcal{V}_k, \mathcal{V}_{k+1})$ should open the window $w_{i,j}$ for the frame after the preceding port $(\mathcal{V}_{k+1}, \mathcal{V}_{k+2})$ closes the window for the same frame.

Technology Constraints

Constraint 11: Stream Isolation Constraint

$$\forall k \in \left[0, \frac{T_s}{T_i} - 1\right] : \forall l \in \left[0, \frac{T_s}{T_j} - 1\right] :$$

$$\Big((\tau^{(a,b))}[w_{i,k}^{(a,b)}] + \delta \leq \phi^{(y,a)}[w_{j,l}^{(y,a)}]) \vee \quad (2)$$

$$(\tau^{(a,b))}[w_{j,l}^{(a,b)}] + \delta \leq \phi^{(x,a)}[w_{i,k}^{(x,a)}])\Big) \vee \quad (3)$$

$$\Big(\kappa^{(a,b)}[w_{i,k}^{(a,b)}] \neq \kappa^{(a,b)}[w_{j,l}^{(a,b)}]\Big) \vee \quad (4)$$

$$\Big(w_{i,k}^{(a,b)} = w_{j,l}^{(a,b)}\Big) \quad (5)$$

1) A frame of a stream can be enter a queue when every frame of a other stream are completely out of the queue, 2) two different frames enter different queues. Otherwise, the windows are the same.

User Constraints

Constraint 12: Stream E2E Latency Constraint

The difference between the closing time for the last egress port's window and the first egress port's opening time should be less than the maximum latency of requirements plus δ (time precession delay)

$$\forall j \in \{0, ..., \frac{T_s}{T_i} - 1\} : \forall f_{i,j}^{(v_1, v_2)} \in \mathcal{F}_i^{(\mathcal{V}_1, \mathcal{V}_2)}, \forall f_{i,j}^{(v_{n-1}, v_n)} \in \mathcal{F}_i^{(\mathcal{V}_{n-1}, \mathcal{V}_n)} :$$

$$\tau^{\mathcal{V}_{n-1}, \mathcal{V}_n}[w_{i,j}^{(\mathcal{V}_{n-1}, \mathcal{V}_n)}] - \phi^{\mathcal{V}_1, \mathcal{V}_2}[w_{i,j}^{(\mathcal{V}_1, \mathcal{V}_2)}] \leq L_i - \delta$$

User Constraints

Constraint 13: Stream Jitter Constraint for sender

At the sender's egress port, any two closing times has the time difference less than the maximum jitter J_i.

$$\forall j, k \in \{0, ..., \frac{T_s}{T_i} - 1\} : \forall f_{i,j}^{(v_1,v_2)}, f_{i,k}^{(v_1,v_2)} \in \mathcal{F}_i^{(\mathcal{V}_1,\mathcal{V}_2)} :$$

$$(\tau^{\mathcal{V}_1,\mathcal{V}_2}[w_{i,j}^{(\mathcal{V}_1,\mathcal{V}_2)}] - j \times T_i) - (\phi^{\mathcal{V}_1,\mathcal{V}_2}[w_{i,k}^{(\mathcal{V}_1,\mathcal{V}_2)}] - k \times T_i) - l_i^{(\mathcal{V}_1,\mathcal{V}_2)} =$$

$$(\tau^{\mathcal{V}_1,\mathcal{V}_2}[w_{i,j}^{(\mathcal{V}_1,\mathcal{V}_2)}] - j \times T_i) - \tau^{\mathcal{V}_1,\mathcal{V}_2}[w_{i,k}^{(\mathcal{V}_1,\mathcal{V}_2)}] - k \times T_i \leq J_i.$$

$$\tau^{\mathcal{V}_1,\mathcal{V}_2}[w_{i,k}^{(\mathcal{V}_1,\mathcal{V}_2)}] = \phi^{\mathcal{V}_1,\mathcal{V}_2}[w_{i,k}^{(\mathcal{V}_1,\mathcal{V}_2)}]) - l_i^{(\mathcal{V}_1,\mathcal{V}_2)}$$

User Constraints

Constraint 14: Stream Jitter Constraint for receiver

At the receiver's egress port, any two closing times has the time difference less than the maximum jitter J_i.

$$\forall j, k \in \{0, ..., \frac{T_s}{T_i} - 1\} : \forall f_{i,j}^{(v_{n-1}, v_n)}, f_{i,k}^{(v_{n-1}, v_n)} \in \mathcal{F}_i^{(\mathcal{V}_{n-1}, \mathcal{V}_n)} :$$

$$(\tau^{\mathcal{V}_{n-1}, \mathcal{V}_n}[w_{i,j}^{(\mathcal{V}_{n-1}, \mathcal{V}_n)}] - j \times T_i) - \phi^{\mathcal{V}_{n-1}, \mathcal{V}_n}[w_{i,k}^{(\mathcal{V}_{n-1}, \mathcal{V}_n)}] - k \times T_i) - l_i^{(\mathcal{V}_{n-1}, \mathcal{V}_n)} \leq J_i.$$

Evaluation

Apply SMT solver with background theory of *quantifier-free integer-index arrays over integer* (*QF_ARIA*: Quantifier free formulas for linear Array, Real and Integers) [1].

Generates schedules, *i.e.*, $\phi^{(a,b)}[w_{i,j}^{(a,b)}], \tau^{(a,b)}[w_{i,j}^{(a,b)}]$, and $\kappa[w_{i,j}^{(a,b)}]$ for each $f_{i,j}^{(a,b)}$ of stream $s_i \in S$.

[1] Jin: Not sure if z3py needs extra-libraries to support *QF_ARIA*.

Our Evaluation using z3py

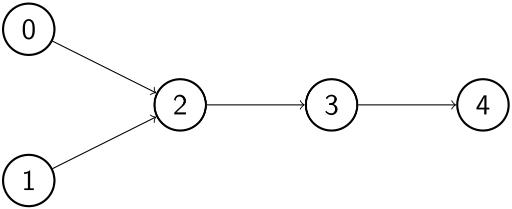
Use z3py

Interpret all technical and user constraints and generate window settings for samples

Validate if all generated constraints of z3 are generated in accordance with TTTech constraints, technical and user constraints, and all generated window schedules are right

TSN Configuration

- Topology:



- $s_i = (C_i, T_i, L_i, J_i), R_i = (\mathcal{V}_0, ...\mathcal{V}_n)$
  - $s_1 = (250, 20, 25, 15), R_1 = (0, 2, 3, 4)$
  - $s_2 = (250, 40, 45, 15), R_1 = (1, 2, 3, 4)$
  - $s_1 = (250, 80, 90, 15), R_1 = (1, 2, 3, 4)$
  - $s_2 = (250, 40, 40, 15), R_1 = (3, 4)$
  - Precision latency: 1

TSN Specification in JSON

- $s_0 = (250, 20, 25, 15)$, $R_0 = (0, 2, 3, 4)$
- $s_1 = (250, 40, 45, 15)$, $R_1 = (1, 2, 3, 4)$
- $s_2 = (250, 80, 90, 15)$, $R_2 = (1, 2, 3, 4)$
- $s_3 = (250, 40, 40, 15)$, $R_3 = (3, 4)$
- Precision latency: 1


```
{
    "units_per_sec": 1e6,
    "sync_precision": 1,
    "vertices": [0, 1, 2, 3, 4],
    "edges": [
        [0,2,3,1e9],
        [1,2,3,1e9],
        [2,3,3,1e9],
        [3,4,3,1e9]
    ],
    "streams": [
    {
        "params": [250,20,25,15],
        "routes": [0,2,3,4]
    },
    {
        "params": [250,40,45,15],
            "routes": [1,2,3,4]
    },
     {
        "params": [250,80,90,15],
          "routes": [1,2,3,4]
    },
     {
        "params": [250,40,40,15],
          "routes": [3,4]
    }
    ]
}
```

Translation Examples

Declare uninterpreted variables:

```
phi = []
Phi = [(Array( 'phi_%d%d'%(key[0],key[1]), IntSort(), IntSort())) for (key, values) in edge_sstat.iteritems()]

tau = []
Tau = [(Array( 'tau_%d%d'%(key[0],key[1]), IntSort(), IntSort())) for (key, values) in edge_sstat.iteritems()]

kap = []
Kap = [(Array( 'kap_%d%d'%(key[0],key[1]), IntSort(), IntSort())) for (key, values) in edge_sstat.iteritems()]

Wn = []

wn_ndx = {}
i = 0
for (key, values) in edge_sstat.iteritems():
    for strm_id in values[EC_STRM_IDS]:
        Wn.append(Array('wn_%s%s_%s'%(key[0],key[1], strm_id), IntSort(), IntSort()))
        wn_ndx.update({(key[0], key[1], strm_id):i})
        i+=1
```

Print out:

```
[phi_12, phi_34, phi_23, phi_02]
[tau_12, tau_34, tau_23, tau_02]
[kap_12, kap_34, kap_23, kap_02]
[wn_12_1, wn_12_2, wn_34_0, wn_34_1, wn_34_2, wn_34_3, wn_23_0, wn_23_1, wn_23_2, wn_02_0]
```

Convert Examples

Constraint 1: Well-defined windows constraints 1

$$\forall (a,b) \in \mathcal{E} : \forall k \in \{1, ..., \mathcal{W}^{(a,b)}\} : \left(\phi^{(a,b)}[k] \geq 0\right) \wedge \left(\tau^{(a,b)}[k] < T_s\right)$$

```python
cstr_tech_01_1 = []
i = 0
for (key, values) in edge_sstat.iteritems():
    for k in range(1, values[EC_TOT_FRM_NUM]+1):
        cstr_tech_01_1.append(And(Select(Phi[i], k) >= 0,
                                  Select(Tau[i], k) < Ts)
                             )
    i += 1

cstr_tech_01_1 = And(cstr_tech_01_1)
```

Figure: z3py source code

```
-------- constraint 1-1 --------------------
And(And(phi_12[1] >= 0, tau_12[1] < 80),
    And(phi_12[2] >= 0, tau_12[2] < 80),
    And(phi_12[3] >= 0, tau_12[3] < 80),
    And(phi_34[1] >= 0, tau_34[1] < 80),
    And(phi_34[2] >= 0, tau_34[2] < 80),
    And(phi_34[3] >= 0, tau_34[3] < 80),
    And(phi_34[4] >= 0, tau_34[4] < 80),
    And(phi_34[5] >= 0, tau_34[5] < 80),
    And(phi_34[6] >= 0, tau_34[6] < 80),
    And(phi_34[7] >= 0, tau_34[7] < 80),
    And(phi_34[8] >= 0, tau_34[8] < 80),
    And(phi_34[9] >= 0, tau_34[9] < 80),
    And(phi_23[1] >= 0, tau_23[1] < 80),
    And(phi_23[2] >= 0, tau_23[2] < 80),
    And(phi_23[3] >= 0, tau_23[3] < 80),
    And(phi_23[4] >= 0, tau_23[4] < 80),
    And(phi_23[5] >= 0, tau_23[5] < 80),
    And(phi_23[6] >= 0, tau_23[6] < 80),
    And(phi_23[7] >= 0, tau_23[7] < 80),
    And(phi_02[1] >= 0, tau_02[1] < 80),
    And(phi_02[2] >= 0, tau_02[2] < 80),
    And(phi_02[3] >= 0, tau_02[3] < 80),
    And(phi_02[4] >= 0, tau_02[4] < 80))
```

Figure: Print out constraints

Output

```
========================== Edge ID: (0,2) ========================

Window ID of Frm (a=0, b=2, i=0, j=0): 1
Q-ID:2
Open:0 ==> Close:2
------------
Window ID of Frm (a=0, b=2, i=0, j=1): 2
Q-ID:2
Open:20 ==> Close:22
------------
Window ID of Frm (a=0, b=2, i=0, j=2): 3
Q-ID:2
Open:40 ==> Close:42
------------
Window ID of Frm (a=0, b=2, i=0, j=3): 4
Q-ID:2
Open:60 ==> Close:62
------------
```

```
========================== Edge ID: (1,2) ========================

Window ID of Frm (a=1, b=2, i=1, j=0): 1
Q-ID:0
Open:0 ==> Close:2
------------
Window ID of Frm (a=1, b=2, i=1, j=1): 3
Q-ID:0
Open:40 ==> Close:42
------------
Window ID of Frm (a=1, b=2, i=2, j=0): 2
Q-ID:0
Open:2 ==> Close:4
------------
```

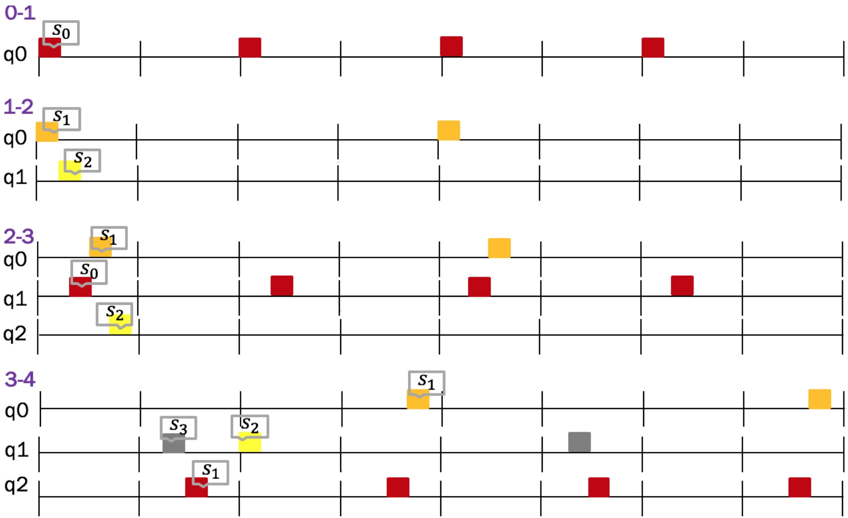
```
========================== Edge ID: (2,3) ========================

Window ID of Frm (a=2, b=3, i=0, j=0): 1
Q-ID:1
Open:3 ==> Close:5
------------
Window ID of Frm (a=2, b=3, i=0, j=1): 4
Q-ID:1
Open:23 ==> Close:25
------------
Window ID of Frm (a=2, b=3, i=0, j=2): 5
Q-ID:1
Open:43 ==> Close:45
------------
Window ID of Frm (a=2, b=3, i=0, j=3): 7
Q-ID:1
Open:63 ==> Close:65
------------
Window ID of Frm (a=2, b=3, i=1, j=0): 2
Q-ID:0
Open:5 ==> Close:7
------------
Window ID of Frm (a=2, b=3, i=1, j=1): 6
Q-ID:0
Open:45 ==> Close:47
------------
Window ID of Frm (a=2, b=3, i=2, j=0): 3
Q-ID:2
Open:7 ==> Close:9
------------
```

```
========================== Edge ID: (3,4) ========================

Window ID of Frm (a=3, b=4, i=0, j=0): 3
Q-ID:2
Open:15 ==> Close:17
------------
Window ID of Frm (a=3, b=4, i=0, j=1): 4
Q-ID:2
Open:35 ==> Close:37
------------
Window ID of Frm (a=3, b=4, i=0, j=2): 7
Q-ID:2
Open:55 ==> Close:57
------------
Window ID of Frm (a=3, b=4, i=0, j=3): 8
Q-ID:2
Open:75 ==> Close:77
------------
Window ID of Frm (a=3, b=4, i=1, j=0): 5
Q-ID:0
Open:37 ==> Close:39
------------
Window ID of Frm (a=3, b=4, i=1, j=1): 9
Q-ID:0
Open:77 ==> Close:79
------------
Window ID of Frm (a=3, b=4, i=2, j=0): 1
Q-ID:1
Open:10 ==> Close:12
------------
Window ID of Frm (a=3, b=4, i=3, j=0): 2
Q-ID:1
Open:13 ==> Close:15
------------
Window ID of Frm (a=3, b=4, i=3, j=1): 6
Q-ID:1
Open:53 ==> Close:55
------------
```

Output



Critical Constraints Missed from TTTech's Work

Tech Constraint - Extra 1: The same stream goes to the same queue.

$$\forall (a,b) \in \mathcal{E} : \forall f_{i,j}^{(a,b)} \in \mathcal{F}^{(a,b)} : \kappa^{(a,b)}[w_{i,j}^{(a,b)}] = \kappa^{(a,b)}[w_{i,j+1}^{(a,b)}]$$

Tech Constraint - Extra 2: The window cannot be shared by different streams

$$\forall (a,b) \in \mathcal{E} : \forall f_{i,k}^{(a,b)}, f_{j,l}^{(a,b)} \in \mathcal{F}^{(a,b)}, i \neq j : w_{i,k}^{(a,b)} \neq w_{j,l}^{(a,b)}$$

Conclusions

- TNS is a new real-time network standard and equipped with timely traffic controls,
- It needs a static scheduling configurations,
- z3 is being used to generate schedules for TSN schedule generation based on TSN constraint specifications.

Current Issues

- Scalability
  - Heuristic algorithm
  - Compositional framework (Divide & Conquer)
  - Incremental techniques