Heuristic List Scheduler for Time-Triggered traffic in Time-Sensitive Networks

Maryam Pahlevan, Nadra Tabassam and Roman Obermaisser
University of Siegen
Siegen, Germany

Background- Time Sensitive Networking

- Standard Ethernet
  - Provides high bandwidth and seamless connectivity
  - But does not offer temporal properties
- Time Sensitive Networking (TSN)
  - Offers deterministic behavior with several Ethernet extensions
  - Introduces new shaping mechanism (Time Aware Shaper)
  - Uses fault tolerant synchronization mechanism (IEEE 802.1ASrev)

Time-Triggered Traffic Scheduling

Requires knowledge of

- Network topology
- TT traffic specification

- Is NP- complete
  - Offline
  - Simplify using several abstractions
- Majority of TT schedulers
  - Fixed routing
  - Employ scheduling constraints

Motivation and Contribution

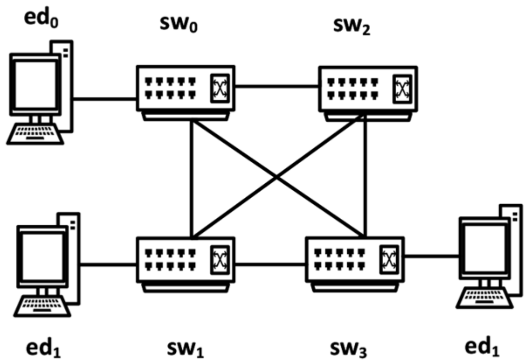
- Heuristic List Scheduler for TSN scheduling problem
  - Joint scheduling and routing constraints
  - Inter-flow dependency
  - Distributed real time application
  - Optimize TT communication overhead and makespan
  - Scalable to large time sensitive systems

Related Works

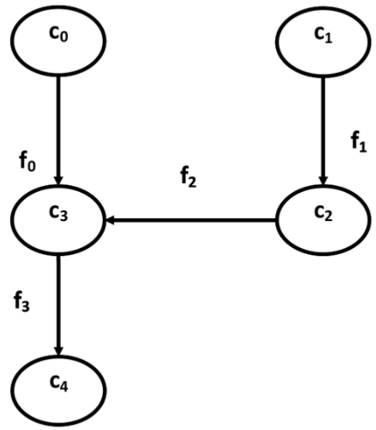
- TT scheduler with fixed routing
  - GCL synthesize using ILP approach (Pop et.al)
  - Define scheduling constraints for TAS and compute GCL using SMT and OMT (Craciunas et.al)
- TT scheduler with joint routing and scheduling constraints
  - ILP based solution and evaluation of network and load dependency (Schweissguth et.al)
  - Introduce Pseudo-Boolean (PB) variables and employ optimization algorithm (Smirnonv et.al)
  - Aforementioned solutions are slow and not support application specific period

System Model

- Application graph : $G_P (T_C, F_{TT})$
    - T_C : computational tasks
    - F_{TT}: TT flows
- Architecture graph : $G_A (R_C, L_d)$
    - R_C : end systems and switches
    - L_d : physical links



(a).Architecture Graph



(b).Application Graph

Problem Formulation

- Compute transmission schedule for TT traffic
  - AVB and BE traffic sent when no TT frame scheduled
- Each computational task identified by
  - *t.ST* : task start time
  - *t.ET* : task execution time
- Each TT flow identified by
  - *f.IT* : when execution of parent task is completed and transmission of flow starts
  - f_n*(size) : the number of TT frames multiplied by frame length*
  - f_d *: maximum admissible end to end latency*
  - *f.e2eD: actual end to end delay for flow delivery*
  - f_p *: periodicity of flow*

Problem Formulation Cont..

- A TT frame remains in TSN capable device

$$f_{PT} = \frac{PR(device)}{f_n}$$

- All GCL of devices start at same time

- Port specific GCL repeated over hyper period

Scheduling and Routing Constraints

- Resource Allocation Constraint
  - Each task assign to only one end system
- Path-Dependent Constraint
  - f_r *comprised of all adjacent links between sender and*
- Contention-free Constraint
  - An exclusive access to all links of f_r *for duration of* $f_{PT} + f.TD$
- Application Specific Periodicity Constraint
  - Each TT flow can be sent over different cycles
  - Each TT flow is scheduled on a certain link considering other TT flows access same link periodically
- Inter-Flow Dependency Constraint
  - *Each task can start transmitting TT frames only after arrival of all incoming flows*
- Delivery Deadline Constraint
  - Each TT flow must delivered to the successor task within f_d

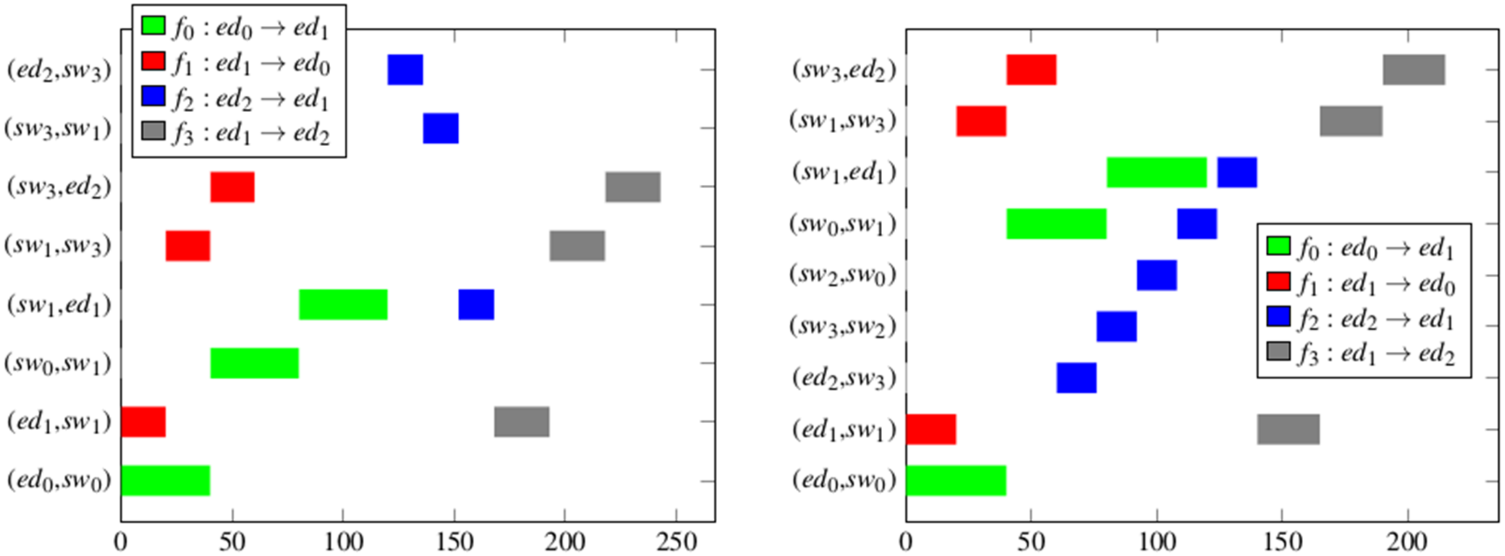
Heuristic List Scheduler (HLS)

- Calculate priority of each task using critical path concept
- Sort Tasks based on their priorities
- For each task
  - If Task has incoming flows, first schedule all predecessor tasks
  - If Task has no child or all predecessor tasks are scheduled, assign available end system to receiver task
- Find all routes between sender and receiver end systems
- For each route, find the earliest injection time
  - Considering contention-free and application specific periodicity constraints
- Considering routes for all incoming flows, choose the receiver
- List scheduler (LS) follow same procedure considering only shortest path

Algorithm 1 Heuristic List Scheduler

```
procedure HUERISITICLISTSCHEDULER
    makespan ← 0
    assign priority to each computational task
    T_sorted ← sort_tasks(on prioirity decsendent order)
    ∀t ∈ T_sorted unscheduled:
    makespan ← Scheduler(t)
    return makespan
procedure SCHEDULER(Task t)
    if task t is unscheduled and task t waits for incoming
TT flows then
        ∀f ∈ F_t.incomingflows: Scheduler(f.sen)
        pred_tasks_scheduled ← true
    else if pred_tasks_scheduled or task t.child==false
then
        ST ← 0
        for p ∈ s.AvailableSystems do
            t.ST ← 0
            f.arr ← 0
            R = findroutes(sen, rec)
            for r ∈ R do
                IT ← FindET
                arr ← f.IT + f.e2eD
                if arr > f_d then:
                    go to the next route
                if f.arr == 0 or arr < f.arr then:
                    f.arr ← arr
                    f_r ← r
                    f.IT ← IT
            ST ← max(ST, f.arr)
            if f.ST == 0 or t.ST > ST then:
                t.ST ← ST
                t.processor ← p
    makespan ← max(makespan, t.ST+t.ET)
    return makespan
```

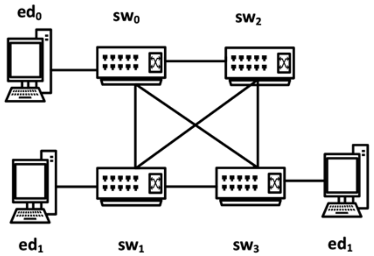
Example of TT Schedule by HLS and LS



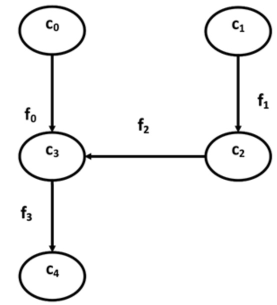
(a) LS TT transmission schedule

(b) HLS TT transmission schedule

Fig. 2: schedule of LS and HLS. Each box shows the time slot assigned to a specific TT flow on a certain physical link



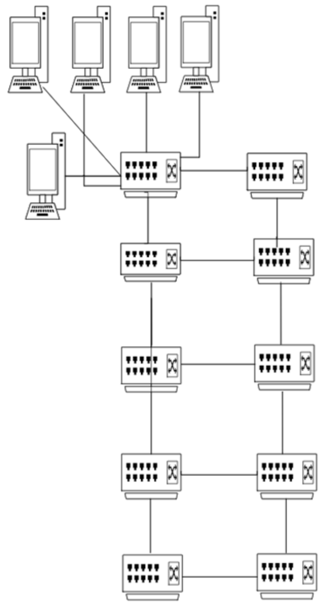
(a).Architecture Graph



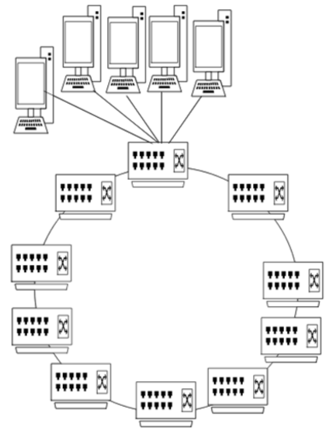
(b).Application Graph

Experimental Set up

- Network topology
    - Ring as a typical industrial structure
    - Meshed to provide higher routing possibilities
    - All links are 1Gbps



Mesh Topology



Ring Topology

- Application graph
    - 20 computational tasks
    - *3 traffic class*

	Period (μs)	deadline (μs)	size (bytes)
TC_1	100	150	100
TC_2	300	150	200
TC_3	500	150	300

Experimental Results

- Traffic load dependency
  - Makespan: HSL improves makespan 28% compared to LS
  - Scheduling capability: schedulability ratio of LS is 0.32 while HSL schedulability ratio is 0.94
  - Execution time: LS is faster than HLS

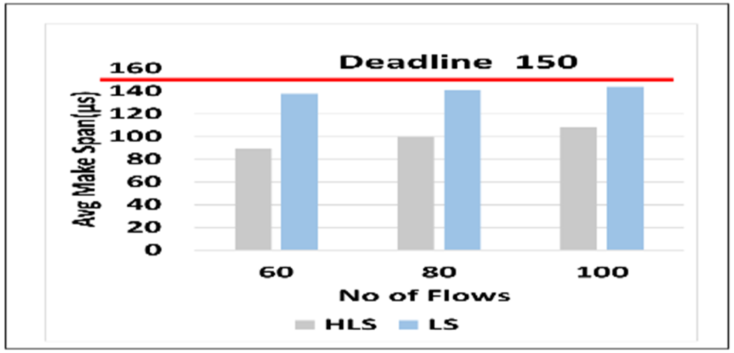


Fig. 4: Average Make Span of LS and HLS

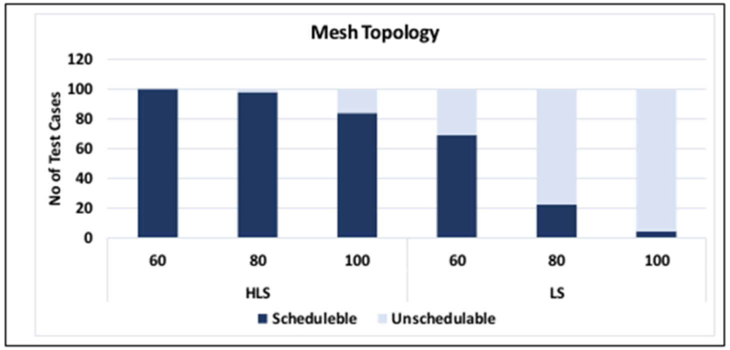


Fig. 5: Schedulability of LS and HLS with varying TT loads and mesh topology

	LS	HLS
TT flows	Avg Exec Time (s)	Avg Exec Time (s)
60	0.102	0.49
80	0.102	0.84
100	0.103	1.58

Experimental Results Cont..

- Network dependency

  - Scheduling capability: schedulability ratio of LS and HLS degraded significantly compared to meshed topology

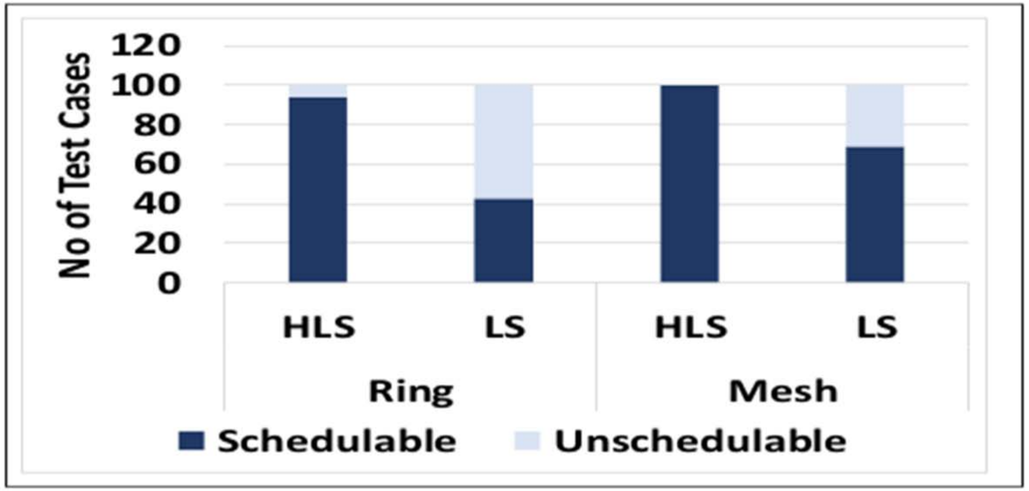


Fig. 6: Schedulability of LS and HLS with 60 TT flows and mesh, ring topology

Conclusion

- HLS outperforms LS in various traffic loads and network topologies
- HLS meets its goal to find TT schedule with optimal makespan
- HLS support inter-flow dependencies and distributed real time application

Thank You

Any Question?

Maryam Pahlevan, University of Siegen

<maryam.pahlevan@uni-siegen.de>

Nadra Tabassam, University of Siegen

<nadra.tabassam@uni-siegen.de>

Prof. Roman Obermaisser, University of Siegen

<roman.obermaisser@uni-siegen.de>