Heuristic List Scheduler for Time Triggered Traffic in Time Sensitive Networks

Maryam Pahlevan[1] Nadra Tabassam[2] and Roman Obermaisser[3]
University of Siegen, Germany
maryam.pahlevan@uni-siegen.de

***Abstract*— Time Sensitive Networks (TSN) are a novel technology that combines the larger bandwidth capabilities of Ethernet with determinism and fault tolerance for safety relevant real time systems. TSN offers bounded latency for the time triggered (TT) communication by transmitting messages according to a global schedule. Most of the scheduling algorithms in this context provide the solution only from the perspective of scheduling constraints and do not consider the impacts of routing on the scheduling problem. Therefore, these algorithms are not capable to provide effective results in the domain of many real time systems. To address interdependence of routing and scheduling constraints, we introduce a heuristic list scheduler (HLS). Our approach generates valid schedules using joint routing and scheduling constraints in one step. Due to this approach, ability to find feasible schedules is dramatically increased in comparison to the schedulers with fixed routing. In addition, HLS supports multi-cast communication, distributed real time applications and inter-flow dependencies. Experimental results shows the significant increase in the schedulability because of the task and message scheduling combined with routing.**

I. INTRODUCTION

Ethernet is capable of providing the high level of bandwidths across the networks but it is not able to fulfill the requirements which are essential for safety-critical real time systems. Industrial applications like robotics, machine control, avionics and railway all require a high level of temporal predictability in data exchanges. TSN is a standard [1] that offers temporal properties with several extensions of Ethernet. TSN provides a promising scheduling technique called Time Aware Shaper (TAS). TAS enables the implementation of time as a correction metric [2] and because of this metric the deterministic behavior can be guaranteed for hard real time systems.

For deterministic delivery of TT streams, all devices should be synchronized according to a global time using a robust clock synchronization mechanism (e.g. IEEE 802.1ASrev). To deploy TAS, every port of a device with TSN capabilities dedicates a certain number of queues for the TT communication and schedules the transmission of the messages using the Gate Control List (GCL) concept. The GCL defines the status of a queue gate at every time instant. TAS by means of GCL ensures that every TT flow has collision-free access to the outgoing port. As a result of the deployment of clock synchronization and TAS, the conventional Ethernet networks can be used for real time applications with strict deadlines and low jitters.

The knowledge of the network topology and the specifications of the TT flow are essential for calculating the global schedule. This computation is off-line because of the complexity exist in the scheduling problems. In TSN, the port-specific GCLs can be calculated as a part of the global schedule generation.

In large time sensitive systems with large number of L2 switches, end devices and links between them, for each TT stream a tremendous number of scheduling possibilities arises. Furthermore, the search space for valid schedules becomes bigger because of the possibility of distributing real time applications over different processing nodes. Therefore in this context it is vital to introduce the search space that results in finding the feasible schedule.

As the scheduling problem of TT communication is NP-complete, for finding a global feasible schedule different system abstractions can be introduced. Most of the algorithms in TT scheduling provide the fixed routing as an input to the scheduler and as a result ignore the effects of routing on scheduling decisions. While this assumption simplifies the overall scheduling problem, it may lead to an infeasible schedule for a system which is schedulable [4]. In [4] and [5] the global transmission schedule is determined by considering routing and scheduling constraints both on one level. An ILP based approach is used for this purpose which slows down the scheduling process and is not feasible for large scale networks. These solutions also do not cover other requirements of real time systems like multi-cast communication patterns, distributed applications and inter flow dependencies.

We propose a Heuristic List Scheduler (HLS) for the global schedule generation in TSN. Our proposed solution merges the scheduling and routing constraints into one set of constraints. To be more specific, our algorithm despite of the majority of the existing solutions, computes the port specific GCL by employing joint routing and scheduling constraints in a single step. HLS also considers TT flows with different priorities and multi-cast patterns. In addition to the aforementioned advantages, our solution enables the distribution of real time applications over available processing nodes.

The main objective of HLS is to meet the deadlines of mission-critical applications and at the same time optimize the TT communication overhead and transmission makespan. For this reason, our algorithm minimizes the time intervals between the scheduled time slots. This strategy decreases

RTN18, July 2018, Barcelona, Spain

the number of guard bands that are reserved before every scheduled time slot to guarantee contention-free access to the outgoing physical link.

We also implement a basic list scheduler (LS) in order to evaluate the schedulability of our algorithm. The basic list scheduler computes the global schedule based on the shortest paths between the end systems. In the experimental evaluation the proposed algorithm is applied to different network structures and TT traffic patterns. The results show that HLS improves the schedulability and transmission makespan of TT traffic remarkably as compared to the basic list scheduler.

The rest of the paper is structured as follows. Section II, discusses the related work. The system model is presented in section III. Section IV discusses the scheduling problem of time sensitive systems. Section V introduces the routing and scheduling constraints. In the following section, our heuristic algorithm is described in more details. Section VII discusses the experimental results. Finally the paper is concluded in the last section.

II. Related Work

In last years extensive work has been done in context of scheduling problems arising from GCL. In [10] authors considers the GCL synthesis for each port of a TSN capable device based on the ILP approach. This work optimally allocates resources to the TT flows and calculates the global schedule. Authors in [7] formulate the different scheduling constraints of TAS in multi hop switched networks. They also validate that GCL guarantees the deterministic delivery of TT messages. In this work, Optimization Modulo Theories (OMT) and Satisfiability Modulo Theories (SMT) are used for finding the feasible schedule. In all above studies, the routing and scheduling problems are solved separately.

Another study in [5] develops an ILP based scheduling solution for the joint routing and scheduling constraints. For the experimental assessment different network topologies and traffic patterns are considered. In addition, these results are evaluated using two performance metrics: (i) scheduling capability and (ii) end to end delay. In [4] routing and scheduling problems of TT communication are solved in a single step by using a set of Pseudo-Boolean (PB) variables. This work also uses a multi objective optimization algorithm to enhance the design space exploration, derived from the joint routing and scheduling constraints. This solution does not consider time sensitive applications with different periodicity which plays a key role in the deployment of TSN. The aforementioned ILP based solutions become time consuming specially in case of large scale time triggered networks. Furthermore, the multi cast requirements and inter dependencies of TT flows are not considered in these works.

Our heuristic list scheduler computes global schedule with short computational time and provides feasible schedules for many real world scenarios. Due to the combined routing and scheduling constraints, the heuristic algorithm finds solution where an optimal scheduler (with fixed routing) would fail to provide a feasible solution while at the same time offering better scalability. HLS performs task binding to support the distribution of real time applications. In addition to the all properties mentioned above, our algorithm supports multi-cast transmission along with the inter-flow dependencies.

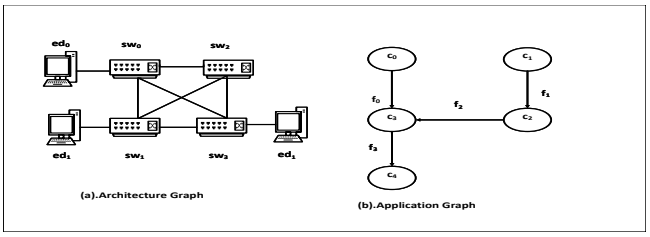


Fig. 1: An example of system model

III. System Model

In this work we employ two graphs. The network topology and TT communication are modeled with an architecture graph and an application graph respectively. These graphs are given as an input to the scheduler. Fig.1 presents an example of the proposed system model. Different scheduling possibilities for TT communication are generated by mapping the application graph to the architecture graph.

A. Application Graph

An application graph is represented by a directed acyclic graph $G_P(T_c, F_{TT})$. G_P comprises of computational tasks as vertexes while the TT flows transmitted between the tasks are considered as edges. Hence each task can have multiple successor tasks and as a result the system model supports multi-cast TT flows [9].

B. Architecture Graph

An architecture graph is represented by an undirected graph $G_A(R_c, L_d)$. This graph comprises of end systems and TSN switches $R_c = ES \cup S$ where R_c is the set of vertexes. There are duplex physical links L_d between different network elements which are considered as edges.

C. Assumptions for System Model

1) All end systems and TSN switches are synchronized according to the global time.
2) Computational tasks generate the TT messages. It is important to note that all physical links $l \in L_d$ are bidirectional. Because of this, multiple TT frames can access the same link via different directions simultaneously.

IV. PROBLEM FORMULATION

In TSN three types of traffic classes are transmitted over the network. These types of traffics are (i) AVB streams, (ii) Best Effort (BE) messages and (iii) TT flows. BE traffic and AVB streams are not having any strict timing requirements. Our scheduling algorithm finds the GCL of each port of a TSN capable device and it reflects the injection time of TT flows routed through that device. The injection time $f.IT$ determines the time instant at that the execution of parent computational task is completed and the sender end system

starts transmitting the TT flow. Our algorithm generates an accurate GCL in order to avoid the interference of BE traffic and AVB streams with TT transmission. We only compute the transmission schedule table of TT messages. Non-TT frames (BE and AVB) are transmitted when no TT messages are scheduled over the same physical links.

For each computational task $t \in T_c$, $t.ST$ is the start time of task and $t.ET$ is the execution time of task. Each TT flow $f \in F_{TT}$ is identified by f_n (size), f_p (period), $f_{sen} \in ES$, $f_{rec} \in ES$, f_d (deadline), $f.e2eD$ (end to end delay) and $f.arr$ (arrival time). f_d determines the maximum admissible end to end latency while $f.e2eD$ specifies the actual end to end delay for flow delivery. The arrival time ($f.arr$) indicates the time instant at that all TT frames are delivered to the receiver end system. A TT flow in TSN can be comprised of more than one frame. f_n is equal to the number of TT frames which are sent consecutively within one f_p multiplied by the length of the frame. f_p represents the periodicity of a TT flow as all frames are sent periodically. For simplicity, we consider that a TT flow remains in a TSN capable device just for the processing time which can be calculated as follow.

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

where PR represents the processing rate. In order to achieve deterministic behavior in scheduled and synchronized networks, all port specific GCL start simultaneously and are repeated over a hyper period. The Hyper period is the Least Common Multiple (LCM) of all f_p values.

V. SCHEDULING AND ROUTING CONSTRAINTS

This section presents the combined constraints of scheduling and routing as defined in [7] and [10].

1) Resource Allocation Constraint: Each computational task is assigned only to one end system. All the information related to the end systems and the requirements of the application are provided by the system designer within the application graph. The end system on which a task can be assigned is selected from the eligible end systems denoted by $t.AvailableSystems$.

$$\forall t \in T_c, es \in t.AvailableSystems:$$

$$t.processor = es$$

2) Path-Dependent Constraint: For the purpose of loop elimination TT flows are restricted to pass through each node only once. f_r is comprised of all the adjacent links which form the path from the sender to the receiver. The value of f_r is equal to one of the routing possibilities between sender and receiver end systems.

$$l_j = (s,t) \in L_d : R = (l_j, ..., l_{j+n}), ..., (l_k ..., l_{k+m})$$

$$f_r \in R$$

3) Contention-free Constraint: Each TT flow can take a particular link for routing, if the TT flow has unique access to the physical link for the time duration of f_{TD} just after its transmission starts. The transmission delay f_{TD} of a TT flow on a particular link can be calculated as follow.

$$l_j \in L_d : f_{lj}.TD = \frac{f_n}{l_j.bw}$$

where bw represents the bandwidth. End devices employ the store and forward approach for switching TT packets.

An important assumption in our algorithm is that a TSN capable device will dedicate only one queue per port to the TT traffic. To avoid interleaving of different TT flows in one TT queue, we employ the flow isolation constraint defined in [7]. This constraint is addressed by collision-free access to egress ports and the attached links for a duration of $f_{PT} + f_{lj}.TD$. This constraint is considered for every link present in f_r.

The duration of exclusive access on two adjacent links in f_r is phase aligned. This assumption implies that buffering of TT frames is not permitted in our proposed system model. The time interval for specific access within each link is calculated with respect to $f_{PT} + f_{lj}.TD$ of the previous adjacent link in f_r.

$$\forall (l_j, l_{j+1}) \in f_r :$$

$$f_{l_{j+1}}.IT = f_{l_j}.IT + f_{PT} + f_{l_j}.TD$$

4) Application-Specific Periodicity Constraint: The periodicity of TT flows can be varied for different real time applications. Therefore the dedicated time slots for each TT flow on a certain link of f_r is scheduled considering other TT streams which access to the same physical links periodically.

5) Inter-Flow Dependency Constraint: Each computational task can only start transmitting its TT messages after the arrival of all TT flows that are sent by its predecessors.

$$\forall f \in F_{TT}, \forall \overline{f} \in pred(f) :$$

$$\overline{f}.e2eD = \sum_{l \in \overline{f}_r} (\overline{f}_{PT} + \overline{f}.TD)$$

$$\overline{f}.IT + \overline{f}.e2eD + f_{sen}.ET \leq f.IT$$

6) Delivery Deadline Constraint: Each TT flow that is transmitted by a computational task should be delivered to the successor task within the predefined deadline of the flow.

$$\forall f \in F_{TT} : f.IT + f.e2eD \leq f_d$$

VI. HEURISTIC LIST SCHEDULER

We develop a Heuristic List Scheduler (HLS) which generates the optimal global schedule for transmission of TT messages. Precisely our goal is to minimize the make span of TT communication using joint routing and scheduling constraints. Our HLS algorithm uses the following steps.

Algorithm 1 Heuristic List Scheduler

```
1: procedure HUERISITICLISTSCHEDULER
2:     makespan ← 0
3:     assign priority to each computational task
4:     T_sorted ← sort_tasks(on prioirity decsendent order)
5:     ∀t ∈ T_sorted unscheduled:
6:     makespan ← Scheduler(t)
7:     return makespan
8: procedure SCHEDULER(Task t)
9:     if task t is unscheduled and task t waits for incoming
   TT flows then
10:        ∀f ∈ F_t.incomingflows: Scheduler(f.sen)
11:        pred_tasks_scheduled ← true
12:    else if pred_tasks_scheduled or task t.child==false
   then
13:        ST ← 0
14:        for p ∈ s.AvailableSystems do
15:            t.ST ← 0
16:            f.arr ← 0
17:            R = findroutes(sen, rec)
18:            for r ∈ R do
19:                IT ← FindET
20:                arr ← f.IT + f.e2eD
21:                if arr > f_d then:
22:                    go to the next route
23:                if f.arr == 0 or arr < f.arr then:
24:                    f.arr ← arr
25:                    f_r ← r
26:                    f.IT ← IT
27:            ST ← max(ST, f.arr)
28:            if f.ST == 0 or t.ST > ST then:
29:                t.ST ← ST
30:                t.processor ← p
31:    makespan ← max(makespan,t.ST+t.ET)
32:    return makespan
```

1) For each task HLS calculates the priority using its critical path. In an application graph, the task's critical path is set to the total communication cost ($CC_f = f_{PT} + f_{lj}.TD$) of the longest path between the predecessor task and that task.
2) After this the HLS sorts the computational tasks and schedules them one by one from highest to lowest priorities. For each task the incoming TT flows are determined first. If the task requires TT flows from predecessor tasks before starting its own TT transmission (as shown in constraint 5) then HLS schedules all the previous tasks first (c.f line 10). If all the predecessor tasks are already scheduled or the current task does not wait for any incoming TT flow then the task is assigned to an available end system from $t.AvailableSystems$.
3) HLS initializes $t.ST$ and $f.arr$ to 0. Then it finds all the routing possibilities between the sender and receiver end systems considering constraint 2.
4) For each of possible routes, HLS finds the earliest injection time considering constraint 3 and 4 (c.f line 19). If the earliest injection time violates the constraint 6 then our algorithm finds the next best route which results in the optimized makespan (c.f line 22). Constraint 6 should not be violated because it will lead to infeasible global schedules. To find the schedule with optimized makespan, the route which results in the minimum $f.arr$ is chosen and $f.r$ and $f.IT$ are updated accordingly (c.f line 23-26).
5) In line 27, the task start time is updated considering constraint (5). According to the routes of all incoming flows, the task is assigned to the end system that leads to the optimized transmission make span (c.f line 28-30). The makespan specifies the time instant at which the execution of all computational tasks is completed.

TABLE I: Parameters for TT flow in Fig.1

	Period (μs)	deadline (μs)	cost (μs)
f_0	400	150	40
f_1	500	70	20
f_2	400	200	16
f_3	500	244	25

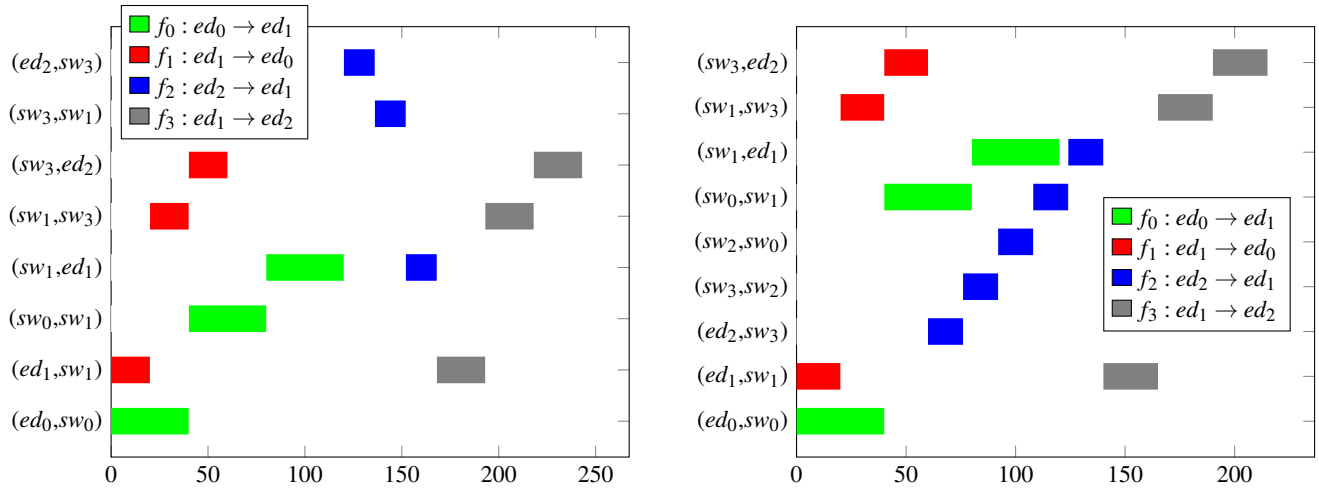
To implement LS, we use same procedure as Algorithm 1. The only difference is we consider the shortest path instead of examining all possible routes (c.f line 28-30). The Gantt charts presented in Fig.2 depicts the transmission schedules calculated by LS and HLS. In this example, the system model shown in Fig.1 is used as an input to our schedulers. Table I lists the communication cost for each TT flow of this application graph (Fig.1 (b)). Fig.2 (b) shows that our overall make span for HLS is improved from 243us to 215us as compared to the make span resulting from LS. The reason for this improvement is that the LS utilizes only a few fixed routes (i.e. shortest path) due to which rest of the routes remain underutilized. This increases the overall load on some of the physical links. HLS selects the best route according to the joint routing and scheduling constraints which increases the load balancing across the network. The optimization of the make span is important in cases where we are specifically dealing with large time triggered systems.

VII. Experimental Evaluation

In our experiments, the SNAP library [12] is used to generate different system models.

A. Network Topology

For our experimental evaluation we consider two types of networks (i) mesh and (ii) ring. Each network as shown in Fig.3 has 10 switches which are connected to 5 end systems. The mesh topology is considered for our algorithm evaluation because it has more routing possibilities and higher connectivity, while the ring topology addresses the typical structure for industrial systems. The bandwidth of all physical links is assumed as 1Gbps. In addition, the processing time for each data byte in every TSN capable device is considered 4 nanoseconds.



(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

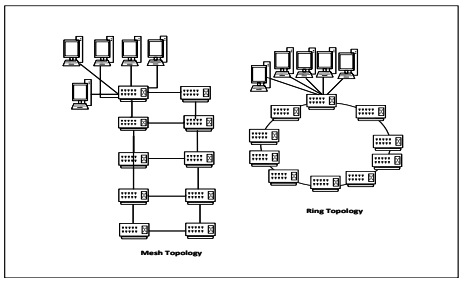


Fig. 3: Topologies (Ring, Mesh). Each switch is connected to 5 hosts.

B. Application Graph

In our application graph we uses 20 tasks having 3 different sets of TT flow. The parameters of these flow sets are presented in Table II. The application graph is structured as a random fire forest directed graph [12]. For simplicity, every multi-cast traffic is considered as sets of unicast TT flow. It is important to note the potential systems on which computational tasks can be processed is selected randomly. The execution time of all tasks is set to 3 microseconds. The implementation of HLS and LS is performed in C++. The experiments are run on a T460 ThinkPad with 32GB of memory and 2.4GHz Intel i5 CPU.

TABLE II: Parameters for TT flow

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

C. Experiments and Results

This section shows the results of our HLS algorithm in comparison with the LS which serves as a typical two-step scheduling solution. We evaluate the impact of varying loads using these parameters (i) make span, (ii) schedulability and (iii) execution time. In the experiments, we employ the mesh structure and three different numbers of TT flows (60,80,100). The ratio of each aforementioned flow set in every TT traffic pattern is 33.3 %. In addition, for every network utilization pattern, we run 100 different test cases.

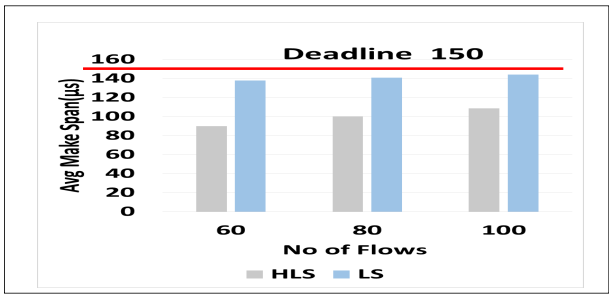


Fig. 4: Average Make Span of LS and HLS

1) Make span: As shown in Fig.4, the make span of HLS for varying traffic loads is improved by an average of 28% as compared to the LS. HLS improves the make span by decreasing end to end delays of TT flows. The main objective of HLS is to compute the TT transmission schedule which leads to better bandwidth utilization and the optimum number of guard bands. To achieve this, HSL minimizes the gap between scheduled time slots and also merges the time slots of consecutive TT frames on a specific link.

2) Schedulability: The graph in Fig.5 shows the schedulability ratio of HLS and LS based on the varying loads. The schedulability ratio of LS is 0.32 on average whereas for HLS this ratio is 0.94 on average. The graph shows that HLS outperforms LS in terms of schedulability specially when the number of TT flows increases. The reason is that the increasing number of TT flows in case of LS leads to the violation of the delivery delay constraint because of over

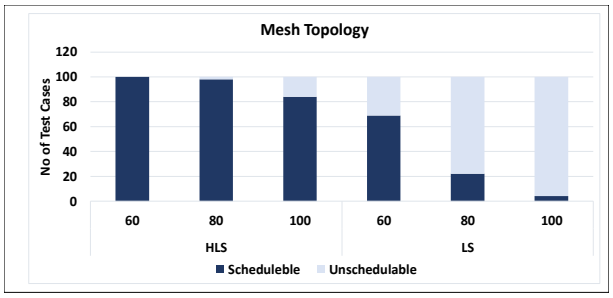


Fig. 5: Schedulability of LS and HLS with varying TT loads and mesh topology

utilization of certain physical links. HLS avoids this issue by sending TT flows on multiple routes.

TABLE III: Execution Time 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

3) Execution Time: As listed in Table III HLS needs more time to find the global schedule of TT communication as compare to LS. LS is faster because it selects only the shortest path for routing while HLS considers different paths and applies the joint scheduling and routing constraints for finding the schedule.

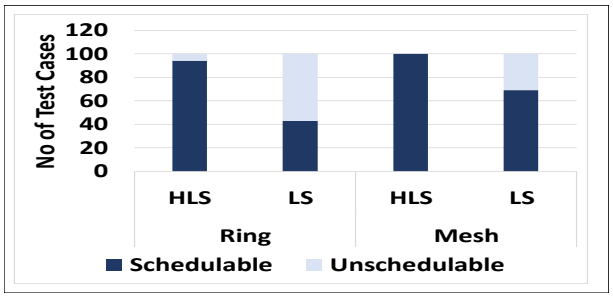


Fig. 6: Schedulability of LS and HLS with 60 TT flows and mesh, ring topology

4) Network Dependency: We repeated the test scenarios with 60 different TT flows but using the ring topology. This is done to show the impact of different network structures on HLS performance as compare to LS. It is clear from the Fig.6 that the schedulability of LS is significantly degraded when the ring topology is used. The reason is that the ring topology with the same number of TT flows may lead to higher utilization of network resources as compared to the mesh structure. Therefore, LS is not able to support the delivery deadline constraint of TT flows. HLS in both cases performed better. In the mesh topology we have more routing possibilities. This feature enhances the load balancing capability of our schedulers. Because of this reason the scheduelability of both HLS and LS for mesh is better as compared to the ring topology in case of similar traffic loads.

VIII. CONCLUSION

In this paper we proposed the heuristic list scheduler to generate a feasible TT transmission schedule. In contrast to the conventional scheduling techniques which apply the routing and scheduling constraints separately, our solution solves both routing and scheduling problems in one step. We introduce a system model comprised of an application graph and an architecture graph which is used as an input to our scheduler. As an additional benefit, the proposed scheduling technique also allows the distribution of real time applications and multi-cast traffic patterns. In order to evaluate our algorithm, a basic list scheduler is implemented which uses the fixed routing mechanism.

The simulation results shows the impact of varying network designs and utilization on the make span and execution time. Overall HLS improves the make span by 28% as compared to LS. The reduced make span may also lead to a lower number of guard bands and more compact TT transmission schedules. Results also shows that in case of high traffic loads HLS enhances the overall schedulability ratio over LS.

ACKNOWLEDGMENT

This work was done as part of Safe4RAIL Project, Grant Agreement No. 730830.

REFERENCES

[1] “Institute of electrical and electronics engineers, time-sensitive networking,” in Time-Sensitive Networking Task Group.http://www.ieee802.org/1/pages/tsn.html, IEEE, 2017
[2] M. A. Weiss et al., “Time-aware applications, computers, and communication systems (taaccs),” in Technical Note (NIST TN)-1867, 2015.
[3] “Institute of electrical and electronics engineers, inc.802.1as-rev - timing and ynchronization for time-sensitive applications,” in Time-Sensitive Networking Task Group. http://www.ieee802.org/1/pages/802.1AS-rev.html, IEEE, 2017.
[4] F. Smirnov, M. Glaß, F. Reimann, and J. Teich, “Optimizing message routing and scheduling in automotive mixed-criticality time triggered networks,” in Design Automation Conference (DAC), pp.1-6, 2017 .
[5] E. Schweissguth et.al, “Ilp-based joint routing and scheduling for time-triggered networks,” in Proceedings of the 25th International Conference on Real-Time Networks and Systems, pp. 8–17,2017.
[6] “Institute of electrical and electronics engineers, inc. 802.1qbv - enhancements for scheduled traffic,” in Time-Sensitive Networking Task Group. http://www.ieee802.org/1/pages/802.1bv.html, IEEE, 2016.
[7] S. S. Craciunas, R. S. Oliver, M. Chmelik, and W. Steiner, “Scheduling real-time communication in ieee 802.1 qbv time sensitive networks,” in Proceedings of the 24th International Conference on Real-Time Networks and Systems, pp. 183–192, ACM, 2016.
[8] M. Lukasiewycz et.al, “Combined system synthesis and communication architecture exploration for mpsocs,” in Proceedings of the Conference on Design,Automation and Test in Europe, pp. 472–477,2009.
[9] R. W. Lucky, ÒAutomatic equalization for digital communication,Ó Bell Syst. Tech. J., vol. 44, no. 4, pp. 547Ð588, Apr. 1965.
[10] P. Pop et.al, “Design optimisation of cyber-physical distributed systems using ieee timesensitive networks,” IET Cyber-Physical Systems: Theory and Applications, vol. 1, no. 1, pp. 86–94, 2016.
[11] “Galib documentation,” in http://lancet.mit.edu/galib-2.4/, 2017.
[12] “Snap library 4.0, user reference documentation,” in https://snap.stanford.edu/snap/doc/snapuser-ref/index.html, 2017.