Dynamic Scheduling and Routing for TSN based In-vehicle Networks

Ammad Ali Syed, Serkan Ayaz, Tim Leinmüller
DENSO AUTOMOTIVE Deutschland GmbH,
Germany
Corporate R&D
{a.syed, s.ayaz, t.leinmueller}@eu.denso.com

Madhu Chandra
Chemnitz University of Technology,
Germany
madhu.chandra@etit.tu-chemnitz.de

***Abstract*—The future autonomous vehicle is not only processing the copious amount of indispensable data generated by its onboard sensors but also utilizing the data from other vehicles, roadside unit (RSU) etc. Managing the mixed-criticality data requires intelligent time-sensitive scheduling and routing within the in-vehicle network (IVN) infrastructure. Use-cases related to self-adaptivity (including vehicular communication), partial networking and embedded virtualization require to change the configuration of the IVN at runtime. State-of-the-art IEEE Time-Sensitive Networking (TSN) standards possess a grave challenge in handling runtime reconfigurations. Above mentioned use-cases foster the development of scalable and efficient dynamic scheduling and routing algorithms for TSN based IVN. In this paper, four meticulously designed heuristics are analyzed for dynamic scheduling and routing on-the-fly in TSN based IVN. One of the algorithms, Bottleneck heuristic outperforms others in term of schedulability and response time. It schedules around $16-22\%$ more flows as compared to other developed heuristics depending on the network load.**

***Index Terms*—time-sensitive network (TSN), in-vehicle network (IVN), vector bin packing (VBP), area control unit (ACU), telematic control unit (TCU), gate control list (GCL)**

I. Introduction

Advancements in the development of various on-board sensor technologies including camera, lidar and radar as well as vehicular communication technologies pave the way for the development of the autonomous vehicle. Different electronic components installed in the vehicle rely on the in-vehicle network to communicate with each other. The demand for electric and electronic (EE) systems is expanding exponentially with the increasing number of in-vehicle applications. Modern vehicle may implement new applications by adding new electronic control unit (ECU) as well as new actuators/sensors in the vehicle network. Some state of the art in-vehicle communication technologies include controller area network (CAN), FlexRay, local interconnect network (LIN), Media Oriented Systems Transport (MOST). As the automation level of vehicle increases, it will require more intelligence supported by a diverse set of actuators/sensors and ECUs to fulfil the functional safety requirements. These actuators/sensors are connected through the in-vehicle network on the physical level and interacting with the central controller (i.e., central processing unit (CPU)). These sensors will generate an ample amount of data which will traverse the IVN to reach to CPU which performs different tasks (e.g., sensor fusion, driving monitoring, motion/mission planning, driving control) based on the received information. The low bandwidth legacy IVN technologies cannot support the ample amount of data required by new applications. The gradual increase in the number of ECUs and the complexity in the IVN lead to demand for more bandwidth and scalable communication technologies. Therefore, Ethernet and its Time-Sensitive Networking (TSN) extensions are now being considered as a potential candidate for next-generation vehicles because of its low cost, scalable bandwidth, flexibility and advanced capabilities supporting mixed critical in-vehicle services (i.e., time-triggered, rate constraint, and best-effort) with different data rates.

Traditionally, the in-vehicle network is configured statically at the designed time to guarantee the quality of service (QoS) requirements of the applications. Due to the static network configuration, it is normally difficult to introduce new applications during the lifetime of the vehicle. The need for an IVN supporting dynamic traffic will increase as the number of features and functionalities requiring dynamic traffic handling in the vehicle increases. Some of the dynamic applications are vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I) and vehicle-to-network (V2N), adaptive cruise control, truck trailer systems and over the air (OTA) software updates. Therefore, automotive applications require dynamic reconfiguration facilities to meet the requirements of new evolving features.

Time-Sensitive Networking (TSN) is a set of IEEE standards that extend the capabilities of Ethernet for safety-critical hard real-time applications. Among them, the IEEE 802.1 Qbv provides a time-aware shaper (TAS) which is the highest level of determinism in Ethernet-based network [1]. In TAS, clock synchronization protocol (IEEE 802.1ASrev) ensure that all the network devices are synchronized with the global time. Each egress port of network device has eight priority queues and the incoming flows are filtered and placed in a dedicated queue. The queues are controlled by Gate Control List (GCL). The GCL has open and close events which determine the time instant at which a queue is eligible to transmit the packet. The global schedule is computed based on network topology and flow characteristics. These schedules are translated to port specific open and close events of the GCL.

In our previous work [1], Mixed-integer programming based joint static scheduling and routing algorithm is designed which would be conducive for dynamic traffic. The quest for this research work has been motivated by asking how to find a "most" suitable schedule and route of certain dynamic traffic on-the-fly on top of already running static schedules in TSN based Ethernet network. In this work, we proposed four different heuristics algorithms to solve the scheduling and routing problem of dynamic applications at runtime in the zonal architecture of the IVN. The scheduling and routing algorithm is a part of the centralized network controller (CNC) that reconfigures the GCL of every port in the network using the network configuration protocol (NETCONF).

The remainder of the paper is organized as follows. Section II provides a survey of the related work. Section III describes the use-cases of dynamic reconfiguration in IVN. The system model is introduced in Section IV. Section V presents problem formulation and four different heuristics for routing and scheduling. Section VI analyzes experimental evaluations. Finally, we conclude the paper in section VII.

II. Related Work

Some of the preeminent work has been done in designing the static schedule of the time-aware shaper in recent years. Among them, some work focuses only on the optimization of the schedule [2], while others take into account both schedules and routes [3]. In [2], the author proposes an ILP to optimize the makespan of the flowshop problem after transforming the packet scheduling problem to flowshop problem. To reduce time complexity, the author proposed a tabu search based meta-heuristic. Another ILP based strategy for solving routing and scheduling problem of TT communication in TSN system was designed by Gavrilut et. al. [4]. In [5] authors used satisfiability modulo theories (SMT) based solver (Z3) to compute a feasible schedule for TAS. Their results showed when the number of flows increases linearly the computational complexity increases exponentially. The author in [2] solved the scheduling and routing problem of time-triggered flows jointly with ILP. Their work solved the transmission schedules of TT flows within a second for a large network.

All the aforementioned seminal work that is used for solving scheduling and routing in TSN's based time-aware shaper belongs to the realm of static networks. These fixed network configurations are not flexible to produce schedules at runtime. The dynamic insertion and withdrawal of the task of TT schedule have been proposed by Zheng et. al. [6] in the avionics network. For the first time, the author in [7], proposed runtime reconfiguration of TSN schedules for Fog computing. In [8], the author proposed offline and online scheduling and routing algorithm for virtual machine (VM) migration in TSN's based time-aware shaper framework. The offline schedule is designed based on minimum distance tree (MDT), which considers only the potential dynamic targets. In this work, four meticulously designed heuristics are analyzed for dynamic scheduling and routing in IVN.

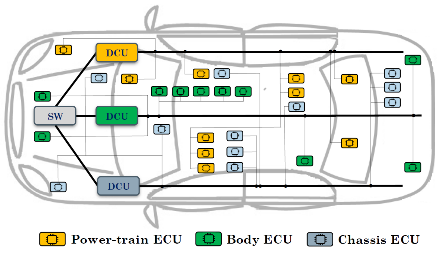
III. Use-cases of Dynamic reconfiguration in IVN

In the course of the analysis, the three main use-cases (i.e., Self-adaptive functionality, Partial networks, and Embedded virtualization) for dynamic reconfiguration in IVN were derived. These use-cases underpin the development of dynamic scheduling and routing algorithms.

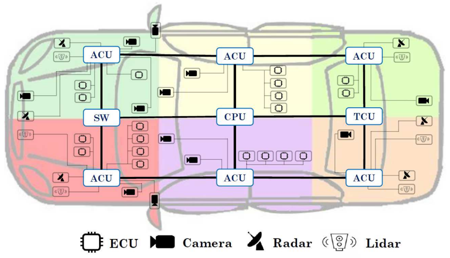
Self-adaptive functionality is widely studied in the automotive system. It is defined as the reconfiguration of the whole system (typically software reconfiguration) while the system is running smoothly [9]. Once a vehicle is built, the embedded computer ECU is planted and is fixed. The software is mainly responsible to activate some functions. Some of the functions are only needed to be active in a certain driving situation such as adaptive cruise control in traffic congestion or Vehicle-to-X (V2X) applications, such as platooning or sensor exchange among vehicles wherein the availability of neighbouring vehicles change dynamically [10]. Another example is the truck-trailer system, in which the trailer is connected/disconnected several times in a day and replace by other [9]. The idea lies behind self-adaption which relies on changing the software component partially or fully and added new features. These dynamic concepts increase the efficient utilization of resources, increase flexibility by extending car features after-market product. These changes establish new objectives or requirements to the network. Therefore, a new real-time network configuration is imperative on medium access control (MAC) level which fulfils the new delay/latency as well as reliability requirements. In this group, over the air (OTA) software updates can be classified as well since certain software update of an ECU may also change the communication behaviour of the ECU with other network elements (e.g., sensors/actuators or ECUs).

Reducing energy consumption is an important optimization parameter of the electric and hybrid vehicle. With the increasing number of ECUs, the power demand of the vehicle will increase, which in turn will also increase the CO_2 emission [11]. Energy saving in IVN can be achieved by using **Partial Network** strategy [11]. In a partial network (PN), some individual ECUs are shutdown (sleep mode) using selective wake-up functions [11] when they are not needed. Besides, they can be wake up when the functionality of the ECU is required. This switch on and off functionality also impose reconfiguration of the network to utilize the available resources efficiently.

The increasing number of features and functionalities in-car possess a demand for powerful hardware platforms and software-defined functions. In central gateway architecture [12], the automotive industry was facing complexity problem with the myriad of ECUs, for each vehicle function require a separate new ECU. Therefore, architectural changes were required to circumvent this limitation. The development of domain control unit (DCU) architecture, surmounted this complexity by consolidating many application-specific ECUs to one powerful DCU [12], as shown in Fig. 1a. Instead of using the high-performance single-core processor in DCU, the automotive industry moved to multi-core architecture which scales better [13]. The application requirements in the automotive domain are so diverse (safety and non-safety requirements) and complex, multi-core processors have become a method of choice to fulfil the application requirements. With the advent of multicore architecture in automotive, **embedded virtualization** also gained traction because of its numerous advantages [14]. It allows execution of different software which poses different functional requirements (i.e. safety-critical, security-critical and non-safety-critical) on a single piece of hardware by dividing the resources of a processor into safely separated virtual machines (VMs) [14]. Each core of CPU could be a virtual machine [14], that provide a safe and secure of software partitions. Virtualization also aids in load balancing by migrating task from one virtual machine to another. This migration of tasks is often needed when there is a fault in one VM. Sometimes it needs to achieve good performance by maintaining a balanced workload among DCUs and utilize the resource efficiently. This task migration incurs dynamics changes in transmission requirements therefore routing and scheduling need to be recomputed. The concept of virtualization is also used in new zonal architecture, which was proposed to reduce the wiring harness in the vehicle [12]. The idea of new zonal architecture is to consolidate many different functionalities while keeping physical domain secluded [1]. The conceptual architecture of area control unit (ACU) in which vehicle area is partitioned into six zones, as shown in Fig. 1b.



(a) Domain Controller architecture

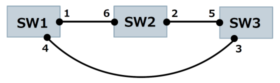


(b) Zonal architecture

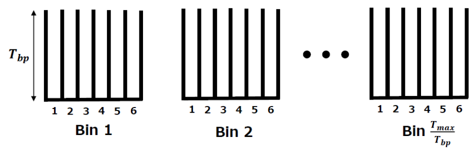
Fig. 1: EE architectures

IV. SYSTEM MODEL

In this work, we model the network topology and flow mapping with two separate entities. The network topology is modelled as unweighted directed graph $G < \alpha, \beta >$,



(a) Network topology



(b) Vector bin packing

Fig. 2: Network topology and flow assignment model

where α is the set of network nodes, having switching capabilities, each node deems as the set of ACUs, CPUs, TCU (Telematics control unit) and ECUs and β is the set of links connecting ACUs, CPUs, TCU and ECUs. For example, a link $[\alpha_{acu1}, \alpha_{acu2}]$ of β transmit flows from source node ACU1 to the destination node ACU2 and similar a link $[\alpha_{acu2}, \alpha_{acu1}]$ of β transmit flows in opposite direction. The set of inserted flows is denoted by F_a and characterized by a tuple $< src_j, dst_j, T_j, w_j >$, where src_j is the source node, dst_j is the destination node, T_j is the periodicity, w_j is the size of the flow. There is a minimum and maximum period that the network support and represent by T_{min} and T_{max} respectively. The set of egress ports of switching devices and end systems is denoted by E e.g. $E = e_1, e_2, \ldots, e_l$, wherein an egress port and respective ingress port constitute a link.
The flow assignment is modelled as a vector bin packing (VBP), in which the items are viewed as flows with d-dimensional resource demand vector. The resource demand vector tells the required egress ports of the network. There may be multiple resource demand vectors of a flow in case the flow transmission occurs over different redundant paths. The size of a flow is defined as the transmission time of the flow and remains the same in all dimensions. The minimum-period T_{bp} partitions the maximum-period T_{max} into bins $[i.T_{bp}, (i+1).T_{bp})$. There are T_{max}/T_{bp} number of bins which are static and known at design time. Taking Fig. 2a into account, it can be observed that there are six egress ports in the network. These egress ports in the network are modelled as the dimensions of the bin, i.e., dimensions from 1 to 6, as shown in Fig. 2b.

V. PROBLEM FORMULATION

In this work, a new network configuration (GCL) is determined based on the existing network configuration. The evaluated new configuration makes sure that the existing configuration is not jeopardized. The centralized network controller receives two sets of flows each time, namely set of flows to be inserted in the current flows and the set of de-inserted flows.
The set of inserted and de-inserted flows are denoted by $\mathcal{F}_a$ and $\mathcal{F}_d$ respectively. The existing running configuration is denoted by $\mathcal{S} \equiv (\mathcal{F}, \mathcal{D}, \mathcal{B}_s)$, where $\mathcal{F}$ represents set of flows

that configured in the network, $\mathcal{D}$ maps flow to the egress ports of the network and $\mathcal{B}_s$ is the current state of the bins. The algorithm determines the new schedule configuration $\mathcal{S}'$ $\equiv (\mathcal{F}', \mathcal{D}', \mathcal{B}_s')$, with the given the inserted flows $\mathcal{F}_a$, and de-inserted flows $\mathcal{F}_d$.

A. Scheduling and Routing Heuristics

In this section, we present four heuristics proposals wherein each heuristic is designed to solve the vector bin packing problem with packet scheduling and routing constraints. The algorithm is shown in Algorithm 1. The first step is to remove the de-inserted flows $\mathcal{F}_d$ from the current schedule configuration $\mathcal{S}$. The algorithm reads the status of the current bins state (line 3). Then the algorithm evaluates the scalar value for all possible paths and bin vectors if all of the constraints are met (line 7-8). The scalar value is a measure of score that determines how well the flow j fit to bin i (i.e., evaluation of the quality of different mappings between flow demands of all possible paths and all possible bins). One typical approach is to evaluate from vector of capacities and flow demands. The Extractor function (line 19) is used to extract the path and bins of a flow which has maximum scalar value (line 15).

Algorithm 1 : Dynamic Scheduling and routing algorithm

```
1: procedure CONFIGURATOR($\mathcal{S}$, $\mathcal{F}_a$, $\mathcal{F}_d$)
2:     $\mathcal{S}_d \leftarrow$ REMOVEFLOWS($\mathcal{S}, \mathcal{F}_d$)
3:     $\mathcal{B}_s \leftarrow \mathcal{S}_d.\mathcal{B}_s$
4:     for all $j \in \mathcal{F}_a$ do
5:         for all $k \in j.K$ do
6:             for all $m \in j.M$ do
7:                 $A \leftarrow$ CHECKCONST.($\mathcal{B}_s, j, k, m$)
8:                 if $(A == True)$ then
9:                     $Score_j^{km} \leftarrow$ SCORE($\mathcal{B}_s, j, k, m$)
10:                else
11:                    $Score_j^{km} \leftarrow -\infty$
12:                end if
13:            end for
14:        end for
15:        MAXVAL $\leftarrow max\ \ Score_j^{km}$
16:        if (MAXVAL $== -\infty$) then
17:            PRINT("$Flow\ j\ is\ unschedulable$")
18:        else
19:            EXTRACTION( MAXVAL)
20:        end if
21:    end for
22: end procedure
```

The feasible schedule must satisfy following constraint set for the realization of a flow setup for a certain scheduling and routing in our TSN based network.

- **Capacity Constraint**: This constraint makes sure that the capacity of each bin dimension should not exceed when flows are placed in bin dimension.
- **Instance Constraint**: The number of occurrences of a flow within maximum transmission cycle must be met.
- **Path Constraint**: Only one path should be selected out of all possible paths.
- **Periodic Constraint**: The other instances of a flow are placed w.r.t periodicities.
- **Delay Constraint**: The provided delay requirement of a flow must be met.

Consider three switches connected with each other as shown in Fig. 2a. The minimum and maximum period in the network are $5ms$ and $20ms$ and that corresponds to 4 bins $(= T_{max}/T_{min})$. For example, a flow j sends from switch 1 to switch 3 with a periodicity of $20ms$. There is only one instance of the flow j in the transmission cycle i.e. $T_{max}/T_j = 20ms/20ms$. As there are 4 bins the flow j can be placed in any of the bin i.e. the set of possible bin vectors are $\mathcal{B}_{ij}^m = [[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$. From switch 1 to switch 3 there are two possible paths i.e. $\mathcal{D}_j^{kl} = [[1,1,0,0,0,0],[0,0,0,1,0,0]]$. Therefore, there are 8 scalars values and the algorithm select the best one. On the other hand, if flow j has a periodicity of $10ms$ then it has 2 instances i.e. $20ms/10ms = 2$. In this case, the set of possible bin vectors are $\mathcal{B}_{ij}^m = [[1,0,1,0],[0,1,0,1]]$. The allocation gap between two consecutive instances of a flow is calculated as $g_j = (T_j * I)/(T_{max})$. The list of variables are shown in Table I.

TABLE I: List of Variables

Symbol	Definition
$\mathcal{F}_a \equiv \{f_j\}$	Set of inserted flows. J is the total number of flows. $j \in [1,\dots,J]$
$\mathcal{F}_d \equiv \{f_j\}$	Set of de-inserted flows.
$\mathcal{D}_j^{kl} \equiv \{d_j^{kl}\}$	Set of demand vectors of a flow j. This set composes of possible path through which flow may be routed. K is the total number of paths. $k \in [1,\dots,K]$. L is the total number of dimensions. $l \in [1,\dots,L]$
C_i^l	Capacity of dimension l in bin i. I is the total number of bins. $i \in [1,\dots,I]$
$\mathcal{B}_{ij}^m \equiv \{b_{ij}^m\}$	Set of possible bin vectors in which flow j can be placed. The possible bin vectors depend on the periodicity of the flow. M is the total number of possible bin vectors for flow j. $m \in [1,\dots,M]$
$V_i^l(t)$	The vector of remaining capacities of bins at time t. The vector is evaluated from current state of the bins $\mathcal{B}_s$.
w_j	Size of flow j. The size of flow is calculated as $\frac{paylaod(bits)}{Bandwidth}$
T_j	Periodicity of flow j.

1) Modified Dot Product Heuristic: In this heuristic, flow placement in bins is characterized based on dot product. One possible usage of the dot product heuristic for scheduling is presented in [15]. The main idea of the heuristic is to consider both resource demand of flow d_j^l and the remaining capacities of the dimension of bins. The vector of remaining capacities of bins at time t is denoted by $V_i^l(t)$ i.e., the total demand of all the flows placed in the bin are subtracted from the bin's total capacity. The algorithm placed the flow in the bin that has maximum dot product value. The dot product of flow j in bin i is expressed as follows:

$$dp_{ij} = \sum_{l=1}^{L} d_j^l \times V_i^l(t), \qquad \forall i \in I, \forall j \in J \qquad (1)$$

As flow can have multiple bin assignment that explicitly depends on the periodicity of the flow. The resource demand of flow varies which depends on the path. With the modification of the dot product, this formula could be used in our problem. By introducing these two parameters, the mathematical formula is expressed as follows:

$$Score_j^{km} = \sum_{i=1}^{I} \frac{\sum_{l=1}^{L} \mathcal{D}_j^{kl} \times V_i^l(t) \times w_j}{\sum_{l=1}^{L} \mathcal{D}_j^{kl}} \times \mathcal{B}_{ij}^m, \qquad \forall j \in J, \forall k \in K, \forall m \in M \qquad (2)$$

The score value of a flow j is evaluated for every path l and possible bin vectors m. Then the path and bin vector which has maximum score value is selected.

2) Modified Most Loaded Heuristic: In this heuristic, flow placement in bins is characterized based on the current load of the bins. In this approach, flow j is placed on the bins which are least loaded. The algorithm placed the flow in the bins that has a maximum score value. With modification, this heuristic adapts to our problem and formalized as follows:

$$Score_j^{km} = -\sum_{i=1}^{I} \sum_{l=1}^{L} \frac{\mathcal{D}_j^{kl} \times w_j}{V_i^l(t)} \times \frac{\mathcal{B}_{ij}^m}{\sum_{l=1}^{L} \mathcal{D}_j^{kl}}, \qquad \forall j \in J, \forall k \in K, \forall m \in M \qquad (3)$$

3) Bottleneck Heuristic: In this heuristic, flow placement in bins is prioritized based on bottleneck bin. The algorithm searches the bottleneck dimension for given paths and required bin vectors. The bottleneck dimension is evaluated as the minimum remaining capacity of the required dimensions of all required bins. The heuristic tends to place the flow on the bins which has maximum value among the bottleneck dimensions of all possible paths and bin vectors. This heuristic also takes into consideration the flow path length. Mathematically, this heuristic is formalized as follows:

$$Score_j^{km} = \min_{\forall i \in I,\ \forall l \in L} S_i^l(j,k,m), \qquad \forall j \in J, \forall k \in K, \forall m \in M \qquad (4)$$

where,

$$S_i^l(j,k,m) = \begin{cases} \frac{\mathcal{D}_j^{kl} \times V_i^l(t) \times \mathcal{B}_{ij}^m}{\sum_{l=1}^{L} \mathcal{D}_j^{kl}}, & \text{if } \mathcal{D}_j^{kl} + \mathcal{B}_{ij}^m = 2, \\ \infty, & \text{otherwise} \end{cases} \qquad (5)$$

4) Coefficient of variation Heuristic: Load balancing has been used in the majority of a distributed embedded system to improve resource utilization. Standard deviation is widely used imbalance metric of the resource utilization of physical server [16]. Using standard deviation as a metric means, if the load is evenly distributed then the standard deviation is very small. Inspiring from the work [16], in which the coefficient of variations is used as a metric to trigger virtual machine migration to have a load-balanced system. This metric is used in many research areas of computer science including queuing theory [17]. We adapted the heuristic to our problem with the following formula:

$$Score_j^{km} = -\frac{\sigma_j^{km}}{\mu_j^{km}}, \qquad \forall j \in J, \forall k \in K, \forall m \in M \qquad (6)$$

Where, σ_j^{km} and μ_j^{km} are standard deviation and mean load of the bins when a flow j traverse through path k and placed in bins that corresponds to bin vector m and can be expressed as follows:

$$\mu_j^{km} = \frac{\sum_{i=1}^{I} W_{ij}^{km}}{I}, \qquad \forall k \in K, \forall m \in M \qquad (7)$$

$$\sigma_j^{km} = \sqrt{\frac{\sum_{i=1}^{I} (W_{ij}^{km} - \mu_j^{km})^2}{I}}, \quad \forall k \in K, \forall m \in M \qquad (8)$$

When a flow j is requested for schedules the load of the bin is accumulated and the new load value is evaluated as follows:

$$W_{ij}^{km} = W_i + \frac{\sum_{l=1}^{L} w_j \times \mathcal{D}_j^{kl} \times \mathcal{B}_{ij}^m}{L}, \qquad \forall i \in I, \forall k \in K, \forall m \in M \qquad (9)$$

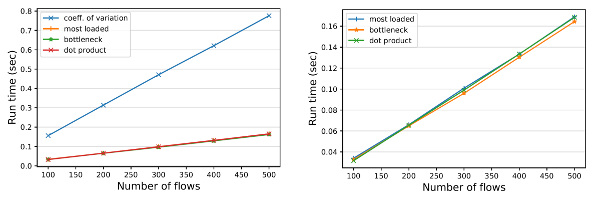
The load of bin i is evaluated as follows:

$$W_i = \frac{\sum_{l=1}^{L} C_i^l - V_i^l(t)}{L}, \qquad \forall i \in I \qquad (10)$$

VI. Experimental Results

In this section, we present the evaluation of four developed heuristics for dynamic scheduling and routing for time-aware shaper in zonal architecture. The heuristics are implemented in python and the experiments were run on 64-bit dual cores 1.7GHz Intel(R) Core i5 processor with 8GB memory. To analyze the efficacy of developed heuristics, we use response time and schedulability on varying load as a performance metric.

The network topology is shown in Fig. 1b. The minimum period in our network is $5ms$ and the maximum period is $50ms$, that corresponds to 10 bins. Each bin composes of 24 dimensions (i.e., the total number of egress ports connecting ACUs, CPU, TCU and switch to each other in the topology), and each dimension has a capacity of $5ms$. The bandwidth of every link is set as 1 Gbps. The flow composes of path sets that contain paths with no more than four hops. This reduction in search space helps in improving the response time and bound the delay of a flow. The maximum bounded delay of a flow j which traverse through 4 hops is calculated as $\frac{w_j(bytes)}{Bandwidth} \times 4 + T_{sw} + T_{que}$, T_{sw} and T_{que} are switching and queuing delay respectively. For a 50 KB flow, the maximum bounded delay is $\frac{50 \times 10^3 \times 8}{1 \times 10^9} \times 4 \approx 1.6ms$, as switching delay is negligibly small compared to transmission delay and queuing delay is zero due to no-wait scheduling assumption [2].



(a) Four Heuristics (b) Three Heuristics

Fig. 3: Runtime of Heuristics

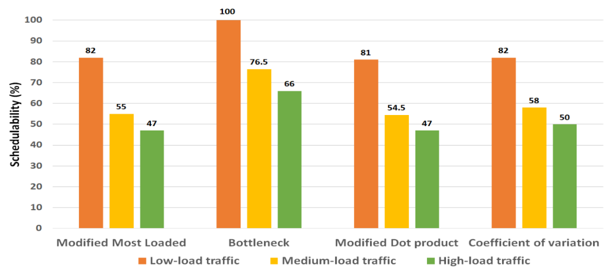


Fig. 4: Schedulability of Heuristics

In order to analyze the response time of the heuristics, we evaluate the worst-case execution time of heuristics on a set of synthetic benchmark by scheduling and routing all flows. The response time of the Coefficient of Variation is worst among other heuristics as it takes $780ms$ to schedule and route 500 flows as shown in Fig. 3a. As it requires more computational power to evaluate mean and standard deviation. On the other hand, the response time of other heuristics are faster and yield schedules within $170ms$ (Fig. 3b).

For schedulability analysis, three synthetic configurations namely low-load, medium-load and high-load traffic are run in zonal architecture. In all of three configuration, data is sent from ACU to CPU, ACU to ACU, TCU to CPU and CPU to TCU with a periodicity of 5, 10, 25 and $50ms$. The data composes of 63 KB, 93 KB and 125 KB in low, medium and high load traffic respectively. $10,000$ runs of randomly shuffled 220 flows are given to four heuristics and average schedulable flows are recorded. The result in Fig. 4 shows that the Bottleneck heuristic outperforms others, as it schedules all the flows in low-load configuration while other schedules $81 - 82\%$ of the total flow. Increase in the load doesn't deteriorate the performance of Bottleneck as much as it degrades others. In medium-load traffic Bottleneck schedules 76.5% of all the flows which is approximately $18.5 - 22\%$ more as compared to other heuristics. The trend continues to be the same in high-load traffic, in which the Bottleneck is $16 - 19\%$ ahead of other heuristics.

VII. Conclusion

In this paper, we have developed dynamic scheduling and routing heuristics for TSN's time-aware shaper. We transformed the problem into vector bin packing and devised four different heuristics. We have examined four heuristics under the performance metric of response time and schedulability with varying load conditions. The Bottleneck heuristic outperforms other heuristics in terms of schedulability and response time as it schedules approximately $16 - 22\%$ more traffic as compared to others. The response time shows that Bottleneck schedules 500 flows in approximately $170ms$ which is crucial for dynamic reconfiguration in the use-cases discussed above.

Acknowledgment

This work was partially funded by the Federal Ministry of Education and Research (BMBF) under KI-PRO project with funding code FKZ:16ES1004.

References

[1] A. A. Syed, S. Ayaz, T. Leinmueller, M. Chandra, "MIP-based Joint Scheduling and Routing with Load Balancing for TSN based In-vehicle Networks," IEEE Vehicular Networking Conference (VNC), 2020.
[2] N. G. Nayak, "Scheduling and Routing Time-triggered Traffic in Time-sensitive Networks," Ph.D. dissertation, Graduate School of Excellence advanced Manufacturing Engineering, University of stuttgart, November 2018.
[3] F. Smirnov, M. Glaß, F. Reimann, J Teich, "Optimizing message routing and scheduling in automotive mixed-criticality time triggered networks," in 54th Design Automation Conference (DAC), pp.1-6, June 2017.
[4] V. Gavrilut, L. Zhao, M. l. Raagaard, and P. Pop, "AVB-Aware Routing and Scheduling of Time-Triggered Traffic for TSN," IEEE Access vol. 6, pp. 75229-75243, November 2018.
[5] S. S. Craciunas, R. S. Oliver, M. Chmelík, W. Steiner, "Scheduling Real-Time Communication in IEEE 802.1 Qbv Time Sensitive Networks," in 24th International Conference on Real-Time Networks and Systems (RTNS), October 2016.
[6] Z. Zheng, F. He, Y. Xiong, "The Research of Scheduling Algorithm for Time-Triggered Ethernet Based on Path-hop," IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), 25-29 Sept. 2016.
[7] M. L. Raagaard, P. Pop, M. Gutierrez, W. Steiner, "Runtime Reconfiguration of Time-Sensitive Networking (TSN) Schedules for Fog Computing," IEEE Fog World Congress (FWC), 2017.
[8] Q. Yu, Hai Wan, X. Zhao, Y. Gao, and M. Gu, "Online Scheduling for Dynamic VM Migration in Multicast Time-Sensitive Networks," IEEE Transactions on Industrial Informatics, vol. 16, no. 6, June 2020.
[9] M. Wagner, A. Meroth and D. Zöbel, "Developing self-adaptive automotive systems On the integration of service-orientation into automotive development processes," in Des Autom Embed Syst, 2014.
[10] A. Ballesteros, M. Wagner, D. Zöbel, "Designing service communication in adaptive automotive," in 8th IEEE international symposium on industrial embedded systems (SIES 2013), Porto, 2013.
[11] C. H. Yi, J. W. Jeon, "Power Saving using Partial Networking in Automotive System," in IEEE International Conference on Information and Automation, Lijiang, China, 2015.
[12] M. Haeberle, F. Heimgaertner, H. Loehrz, N. Nayak, D. Grewe, S. Schildt, and M. Menth, "Softwarization of Automotive E/E Architectures: A Software-Defined Networking Approach", IEEE Vehicular Networking Conference (VNC), 2020.
[13] D. Reinhardt, "Domain Controlled Architecture," in International Conference on Pervasive, Barcelona, 2013.
[14] M. Strobl, M. Kucera, A. Foeldi, T. Waas, N. Balbierer, C. Hilbert, "Towards Automotive Virtualization," in International Conference on Applied Electronics, Pilsen, Czech Republic, 2013.
[15] Panigrahy R, Talwar K, Uyeda L, Wieder U, "Heuristics for vector bin packing," Technical report, Microsoft Research, 2011.
[16] E. Arzuaga, D. Kaeli, "Quantifying load imbalance on virtualized enterprise servers," in Proceedings of the first joint WOSP/SIPEW International Conference on Performance Engineering, San Jose, California, USA, January 28-30, 2010.
[17] L. Kleinrock. Queueing Systems. Volume I: Theory. John Wiley and Sons, New York, 1975.