CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

Routing Heuristics for Load-balanced Transmission in TSN-Based Networks

RTN 2019 is a satellite workshop of ECRTS 1919.

Mubarak Ojewale
Patrick Meumeu Yomsi

CISTER-TR-190507

2019/07/09

Routing Heuristics for Load-balanced Transmission in TSN-Based Networks

Mubarak Ojewale, Patrick Meumeu Yomsi

CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: mkaoe@isep.ipp.pt, pamyo@isep.ipp.pt

https://www.cister-labs.pt

Abstract

A carefully designed routing synthesis can help system designersachieve a better load balancing in TSN-based networksand avoid congestion. To this end purpose, this workproposes two heuristics referred to as (1) LB-DRR, whichaims at achieving a better load balancing and compute asmuch disjoint routing paths as possible for each replicated flow; and (2) CR-DRR, which recomputes paths for time-sensitive flows in congestion situations. Extensive simulationsdemonstrate that the proposed approach outperformsthe classical Shortest Path (SPA) and the weighted EqualCostMulti-path (wt-ECMP) algorithms in terms of the maximumload transmitted on a link by more than 70% and 20%,respectively.

Routing Heuristics for Load-balanced Transmission in TSN-Based Networks

Mubarak Adetunji Ojewale
mkaoe@isep.ipp.pt

Patrick Meumeu Yomsi
pmy@isep.ipp.pt

CISTER Research Centre, ISEP
Polytechnic Institute of Porto, Portugal

ABSTRACT

A carefully designed routing synthesis can help system designers achieve a better load balancing in TSN-based networks and avoid congestion. To this end purpose, this work proposes two heuristics referred to as (1) LB-DRR, which aims at achieving a better load balancing and compute as much disjoint routing paths as possible for each replicated flow; and (2) CR-DRR, which recomputes paths for time-sensitive flows in congestion situations. Extensive simulations demonstrate that the proposed approach outperforms the classical Shortest Path (SPA) and the weighted Equal Cost Multi-path (wt-ECMP) algorithms in terms of the maximum load transmitted on a link by more than 70% and 20%, respectively.

Keywords

Time Sensitive Networking; Routing Algorithms; Congestion; Load-Balancing

1. INTRODUCTION

Ethernet in its original specification was not designed with real-time communication in mind. The IEEE Time Sensitive Networking (TSN) Task Group [7] acknowledged this fact and has been investing considerable workforce to come up with a set of new standards to address this limitation. In this context, the group has designed sophisticated mechanisms to achieve temporally predictable and reliable transmission of packets over switched Ethernet networks. Specifically, key features like flow-synchronization; -management; -control; and -integrity, have been instanced. For a given network, deriving an efficient and cost-effective flow control scheme is paramount. It would make it possible for users and operators to centrally and dynamically discover; configure; monitor; and report on the capabilities of switches and end-stations (a.k.a. nodes) [9]. In a nutshell, the TSN flow control mechanism can be considered from two perspectives: (1) the scheduling (i.e., when each flow shall be transmitted); and (2) the routing (i.e., on which path each flow shall be transmitted). Dürr et al. [2] demonstrated that the scheduling problem of real-time (a.k.a. time-sensitive) flows can be reduced to the No-Wait Job Shop Problem (NW-JSP), which is NP-Hard. On another front, Wang and Crowcroft [14] proved that any routing problem that is subject to two or more independent additive or multiplicative tree constraints is NP-Hard. This is the case for time-sensitive flows, unfortunately. They are subject to timing, bandwidth, cost and reliability constraints. Consequently, seeking for an exact solution is very challenging and computationally expensive. Designing efficient heuristics is the only viable alternative.

In recent years, the scheduling problem has received significantly more attention by the research community than the routing. However, Nayak et al. [10], Singh [12], and Gavriluţ et al. [4] among others raised voices and stressed on the importance of routing in achieving low latency, predictability, and reduced architecture cost. In this work, we follow the same path and focus on the routing problem of TSN flows as an improper routing strategy may increase the number of transmission operations, thereby incurring additional delay. Also, it may increase the blocking time of flows in the network if too many flows try to simultaneously traverse the same path. We believe that a strategy that minimizes the number of transmission operations and the blocking times suffered by each flow would help get around and/or mitigate these situations.

▷ **Limitations of the state-of-the-art.** The TSN standard on path control and reservation [8] recommends the Constrained Shortest Path First (CSPF) routing scheme for the transmission of time-sensitive flows (see page 71). It dictates that this scheme

> "*essentially performs shortest path routing on the topology that only contains the links meeting the constraint(s).*"

From this quote, it follows that CSPF is similar to the Shortest Path Algorithm (SPA) in its operation. Consequently, it is also exposed to congestion and increased blocking time for flows. To illustrate this claim, let us consider the network topology in Figure 1, where six nodes (Node 1 to Node 6) and six switches (S_1 to S_6) are connected by full duplex links. Nodes communicate through flow transmissions over the links and switches. In this example, we consider three flows – flow $f1$ (green) is transmitted from Node 1 to Node 6; $f2$ (yellow) from Node 2 to Node 5; and finally, flow $f3$ (brown) is transmitted from Node 4 to Node 4. We assume that the CSPF routing policy is adopted and all valid paths from each source to each destination node allows each flow to satisfy its end-to-end timing requirement. Then, all these flows are transmitted via the "direct link" (in red) between S_1 and S_6, thus increasing the eventual blocking time over this link for each flow. This state of facts

RTN'2019, Stuttgart, Germany

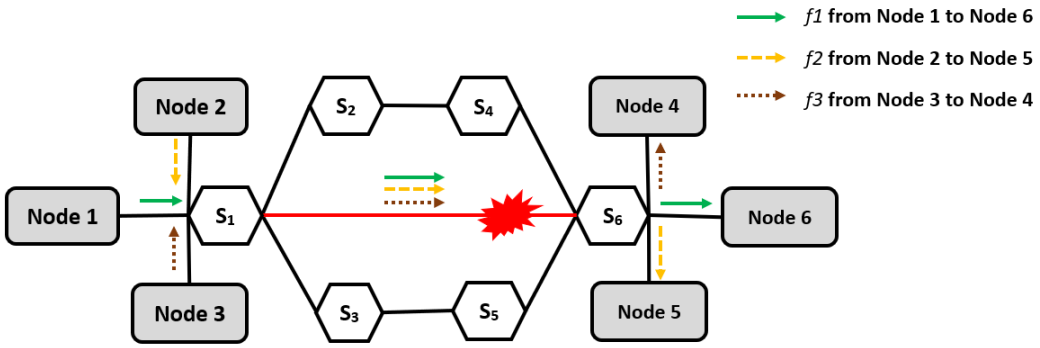


Figure 1: Congestion under CSPF routing policy.

makes this link the potential single point of failure of the network and may cause congestion despite the high level of connectivity. The same limitation applies to the Equal Cost Multi-Path (ECMP) and the weighted ECMP (wt-ECMP) routing schemes [12], unfortunately. The basic idea of these two routing schemes is as follows. Under ECMP, instead of computing a single shortest route like this is the case with SPA, multiple shortest routes are computed and from these, one or several routes are selected arbitrarily. The wt-ECMP scheme distinguishes itself from ECMP only in the selection mechanism. Here, for all the computed shortest routes, a "weight" is assigned to each route to make sure that selection is not performed in an arbitrary manner.

▷ **Our contribution.** To get around the aforementioned hurdles and to fully take advantage of the network connectivity, we suggest the adoption of a routing strategy that ensures load balancing, i.e., a strategy that distributes the transmission operations among the links as even as possible. In addition, this approach ensures that no link becomes the only potential point of failure of the network. In this scope, this paper proposes two heuristics, referred to as LB-DRR and CR-DRR, with the following objectives:

- ▷ LB-DRR: which aims at finding a feasible route for each flow so that the traffic on each link is minimized[1].
- ▷ CR-DRR: which aims at computing alternative routes for each flow in a situation of congestion at run-time.

Although the proposed routing schemes are motivated by the limitations observed in the specifications of TSN, they can be ported (with minor efforts) to a large portion of real-time Ethernet networks.

▷ **Paper organization.** The rest of this paper is structured as follows. Section 2 presents the model of computation and introduces the notations adopted in this work. Our proposed heuristics (LB-DRR and CR-DRR) are detailed in Section 3. Section 4 reports on the experiments carried out and discussions about these. Section 5 discusses the related works on the topic in the literature. Finally, Section 6 concludes the paper and provides future research directions.

2. MODEL OF COMPUTATION

In this section, we define the network topology and the flows specification assumed throughout this paper. Also, we introduce the notations and parameters necessary for a good and crystal clear understanding of our proposed heuristics.

▷ **Network topology specification.** We modelled the network as an undirected graph $G = (V, E)$, where the set $V = N \cup S$ of vertices in G is composed of a finite set N of nodes and S of switches (see Figure 1 for an example). The vertices are connected by a set E of full duplex links or edges. This means that each edge $e \in E$ is defined by a couple $(v_1, v_2) \in V \times V$ of two connected nodes.

▷ **Flow specification.** By default, every TSN-based network addresses *recurrent* (periodic and/or sporadic) flows grouped in classes (e.g., CDT, Audio/Video, etc.). These flows are transmitted within so-called cycles[2] and within a cycle, each flow is treated individually (irrespective of its period) [10]. When all flows are released simultaneously (as assumed in this work), we can safely restrict our attention to a single cycle (the first one). As such, we consider a set of n *aperiodic* time-sensitive flows $F \stackrel{\text{def}}{=} \{f_1, f_2, \ldots, f_n\}$. Each flow $f_i \stackrel{\text{def}}{=} (src_i, dst_i, rep_i, C_i, T_i, D_i) \in F$ is characterized by a 5-tuple, where: (1) src_i is the source node; (2) dst_i is the destination node; (3) rep_i is the replication level (i.e., the number of replicates of f_i allowed to be transmitted from src_i to dst_i); (4) C_i is the size; and finally (5) D_i is the deadline of the flow, i.e., the latest time instant by which at least one copy (original or replicates) of f_i must reach dst_i. We assume all flows are uni-cast, i.e., each flow has a unique destination. We define the set of replicates of f_i as $rep_{f_i} \stackrel{\text{def}}{=} \{f_{i,1}, f_{i,2}, \ldots, f_{i,rep_i}\}$ and assume that each flow and all its replicates are transmitted *simultaneously* over the network.

[1]This heuristic also makes sure that replicated flows are transmitted on routes as disjoint as possible.

[2]The length of each cycle is computed as the Least Common Multiple (L.C.M.) of the periods of all flows.

3. PROPOSED SOLUTION

In this work, we assume that all edges are homogeneous (i.e., they all have the same characteristics and are interchangeable). Before we detail our proposed routing strategy, let us first define a number of concepts for a better understanding of our approach from the reader standpoint.

DEFINITION 1 (ROUTE). *A* route r_i *of flow* f_i *is defined as an ordered list* $\langle (src_i, v_{i,1}), (v_{i,1}, v_{i,2}), \ldots, (v_{i,p}, dst_i) \rangle$ *of edges that can be traversed by* f_i *from its source to its destination.*

DEFINITION 2 (VALID ROUTE). *A* valid route *for* f_i *is defined as any route* r_i *that meets its timing requirement* D_i.

DEFINITION 3 (LENGTH OF A ROUTE). *The length of a route* r_i *denoted by* $len(r_i)$ *is defined as the number of edges along the route.*

DEFINITION 4 (LOAD OF A EDGE). *For every edge* $e = (v_1, v_2) \in V \times V$, *we define the* load of e, *denoted by* $load(e)$, *the sum of the sizes of all flows traversing* e. *Formally, the load of edge* e *is defined by Equation 1.*

$$load(edge) \stackrel{\text{def}}{=} \sum_{f_i \text{ traversing } edge} C_i \quad (1)$$

DEFINITION 5 (MAXLOAD OF A ROUTE). *The* MaxLoad *of a route* r_i, *denoted by* $Maxload(r_i)$, *is defined as the* maximum *load of all edges in* r_i. *Formally, the MaxLoad of route* r_i *is defined by Equation 2.*

$$Maxload(r_i) \stackrel{\text{def}}{=} \max_{edge \in r_i} \{load(edge)\} \quad (2)$$

At this stage, we have all the tools we need to describe our proposed routing solution. The basic idea is as follows. In contrast to the traditional routing schemes (e.g., SPA, ECMP and wt-ECMP), where the underlying strategy is to focus on finding the shortest route for each flow, here we explore all the valid routes. If we denote by R_i the set of all valid routes for flow f_i, then our routing strategy consists in selecting the route that results in the best load distribution in R_i, i.e., the route that minimizes the cost function defined in Equation 3.

$$Cost(r_i, K) \stackrel{\text{def}}{=} Maxload(r_i) + K \cdot len(r_i) \quad (3)$$

In this Equation 3, parameter $K > 0$ is a penalty constant value defined by the user. This parameter is meant to penalize the routes with longer lengths. To make a long story short, it must be looked at as trade-off. It must be set in such a way that the weight of $K \cdot len(r_i)$ in the cost function is significant and $Maxload(r_i)$ does not dominate it and vice-versa. In the latter case, if $K \cdot len(r_i)$ dominates $Maxload(r_i)$, then the cost function would behave like wt-ECMP. On the other front, *Maxload*(r_i) is computed to penalize solutions where some edges in the route are transmitting a high number of flows[3]. Last but not least, if several routes return the same lowest-cost value, then we select one of these routes in an arbitrary manner. Formally, for each flow f_i, its best route $Best(f_i)$ is defined by Equation 4.

$$Best(f_i) \stackrel{\text{def}}{=} \min_{r_i \in \text{valid\_routes}} \{Cost(r_i, K)\} \quad (4)$$

[3]Hence making these edges become potential bottlenecks.

In this equation, variable "valid_routes" denotes the set of all valid routes for flow f_i. Consequently, wt-ECMP is a special case of the proposed approach, where parameter K is sufficiently large and $K \cdot len(r_i)$ dominates $Maxload(r_i)$. Now, we can proceed with the details of our proposed routing schemes.

⋈ **On load-balancing (Algorithm 1)**. The load balancing routing scheme (LB-DRR) takes three components as inputs, namely: (1) the network topology G; (2) the set F of flows to be routed; and finally (3) the user-defined penalty variable K. In the description of the algorithm, the notation $|A|$ refers to the cardinal of set A.

For each flow f_i, after the initialization phase (lines 1 to 3), LB-DRR computes the best route by using Equation 4 (line 6). Then, the load of all edges on this route is updated (line 8) and the selected route is appended to the list of best routes R_i of flow f_i. If the number of replicas of f_i is strictly greater than zero, then all the edges that have already been traversed by the original flow f_i are recorded in variable *used_edge* (line 12). Next, all the valid routes are computed (line 13) and the route $r_{i,j}$ that has the minimum overlap with *used_edges* is selected for replica $f_{i,j}$ (with $j \in [1, rep_i]$) (line 15). If several routes return the same minimum overlap with *used_edges*, then one of these routes is selected arbitrarily and the load of all edges on $r_{i,j}$ is updated (line 17). Note that the edges belonging to *used_edges* are also updated so as to take into account those traversed by replica $f_{i,j}$ (line 19). Thereafter, route $r_{i,j}$ is appended to R_i (line 20) and R_i, which is the list of selected routes for f_i and its replicas, is appended to the list R of the selected routes for all flows (line 23). When this process is completed for all f_i to be transmitted, the algorithm returns the list R (line 25).

⋈ **On congestion recovery (Algorithm 2).** This algorithm, referred to as CR-DRR, is based on the *Tabu metaheuristic* [5] and is reactive in that it aims at re-routing the flows caught in a congestion situation. In a nutshell, the main intuition behind any tabu-based meta-heuristic is to temporarily mark some moves as forbidden so as to force the algorithm to seek for alternative solutions, potentially better in comparison to the current one with respect to a given metric. With this concept in mind, the CR-DRR scheme operates as follows. It takes five components as input: (1) the network topology G; (2) the original routing configuration for all flows R; (3) the congestion threshold *cgst_threshold*[4]; (4) the list of the loads on each edge (*load*); and finally (5) the user-defined penalty variable K. All congested edges according to parameter *cgst_threshold* are stored as "tabu-edges" (*csgt_edges*) and are temporarily removed from the network topology (line 2). For every congestion situation, we initialized the set of congested routes *csgt_routes* (i.e., all routes containing at least one congested edge); the set of flows (*csgt_flows*) traversing the congested routes; and the new set of routes R_{new} to include all routes in R except the congested routes (lines 3 to 5). Then, we seek for alternative routes on the new topology for each congested flow $f_i \in csgt\_flows$ (line 7). From these alternative route(s), we select the best route[5] r_i by using Equation 3 (line 9).

[4]This parameter defines the upper-limit of the load admissible on an edge, otherwise it is deemed as congested.

[5]Again, if several routes return the same minimum cost, then we select one of these in an arbitrary manner.

Algorithm 1: LB-DRR routing scheme.

Data: Network topology G; Set of flows F; Constant K
Result: List of best routes for each flow in F

```
1  R ← empty list[];
2  edges ← Set of all edges in G;
3  load ← zeros[|edges|];
4  foreach f_i ∈ F do
5      R_i ← [];
6      Compute r_i = Best(f_i) (see Equation 4);
7      foreach edge ∈ r_i do
8          load[edge] = load[edge] + C_i;
9      end
10     R_i.append(r_i);
11     if rep_i > 0 then
12         used_edges ← {edge ∈ r_i};
13         routes = valid_routes(G, src_i, dst_i);
14         for j = 1 to rep_i do
15             r_{i,j} = arg min_{r∈routes} (|used_edges ∩ {edge ∈ r}|);
16             foreach egde ∈ r_{i,j} do
17                 load[edge] = load[edge] + C_i;
18             end
19             used_edges = used_edges ∪ {edge ∈ r_{i,j}};
20             R_i.append(r_{i,j});
21         end
22     end
23     R.append(R_i);
24 end
25 return R
```

We check if re-routing flow f_i will not cause congestion on any edge in r_i (line 10). If it does, we leave f_i on its original route old_r_i (line 24). At the end of this process, we update the list *load* in two phases: (i) on the old route old_r_i: we deduct C_i from all edges (line 12) and (ii) on the new route r_i: we augment C_i to all the edges (line 15). We update $cgst\_edges$ (lines 17 to 22). In case there is no alternative route for f_i in $New\_Topology$, it is kept on its original route old_r_i (line 27). Finally, the computed route is appended to R_{new} (line 29) and when all congested flows have been re-routed, the list R_{new} is returned for all flows (line 31).

Algorithm 2: CR-DRR routing scheme.

Data: Network topology G; Original routing configuration R; List of loads on each edge (*load*); Congestion threshold $cgst\_threshold$; Constant K
Result: A new routing configuration R_{new}

```
1  cgst_edges ← {edge in G | load(edge) > cgst_threshold};
2  New_Topology ← G \ cgst_edges;
3  cgst_routes ← {r ∈ R | r ∩ cgst_edges ≠ ∅};
4  cgst_flows ← {f_i traversing a route in cgst_routes};
5  R_new ← R \ cgst_routes;
6  foreach f_i ∈ cgst_flows do
7      routes ← valid_routes(New_Topology, src_i, dst_i);
8      if (routes ≠ ∅) then
9          r_i = arg min_{r∈routes}(Cost(r, K));
10         if (Maxload(r_i) ≤ cgst_threshold) then
11             foreach edge ∈ old_r_i do
12                 load[edge] = load[edge] − C_i ;
13             end
14             foreach edge ∈ r_i do
15                 load[edge] = load[edge] + C_i ;
16             end
17             foreach edge ∈ cgst_edges do
18                 if (load(edge) ≤ cgst_threshold) then
19                     New_Topology =
                         New_Topology.add(edge);
20                     cgst_edges = cgst_edges \ {edge};
21                 end
22             end
23         else
24             r_i = old_r_i
25         end
26     else
27         r_i = old_r_i
28     end
29     R_new.add(r_i);
30 end
31 return R_new
```

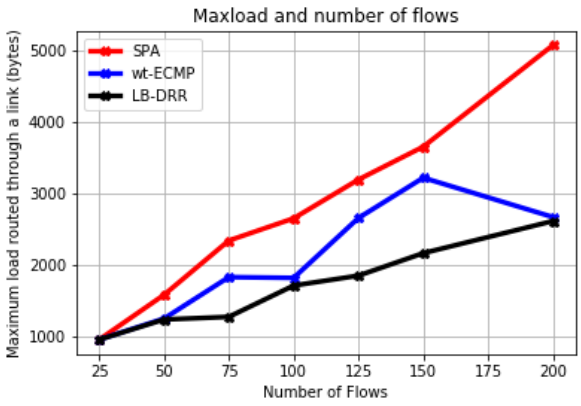
4. EXPERIMENTAL RESULTS

In this section, we report on the experiments conducted on synthetic workloads to evaluate the performance of the proposed heuristics (LB-DRR and CR-DRR) in terms of maximum load transmitted on an edge against SPA and wt-ECMP. Then, we assessed the scalability of the proposed algorithms to demonstrate their applicability.

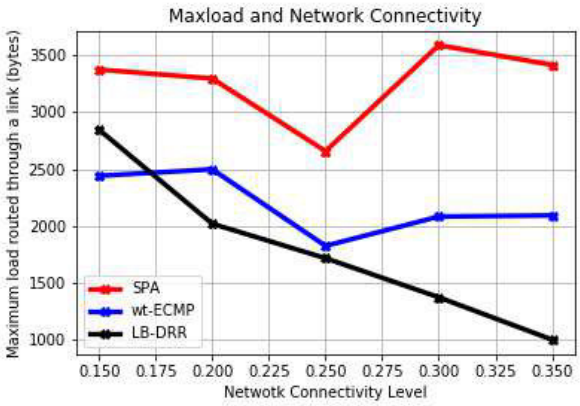
▷ **Setup.** We considered a TSN network, modeled as an *Erdős-Rényi graph* [3] with 50 nodes and a connectivity level falling in the interval $[0.15, 0.35]$. We set $K = 100$ and randomly generated up to 1000 real-time flows in the window $[25, 200]$. For each flow, we assume that its size is between 200 and 1000 bytes and its replication level is randomly chosen between 0 and 2. Also, to constrain the solution space (i.e., to limit the set of valid routes for each flow), we consider the deadline of each flow in the range of 2 to 5 time units and assume a constant traversal time of 1 time unit per edge. In the first batch of experiments, we assumed the LB-DRR routing scheme and the second batch, we assumed the SPA routing scheme. In the latter case, we applied the CR-DRR algorithm to re-route the flows under congestion situations.

▷ **Results and discussion.** From the first batch of experiments, we observed that LB-DRR reduces the maximum load transmitted on an edge (Maxload) by 70.3% and 23.3% in average as compared to SPA and wt-ECMP, respectively. Figure 2a shows the Maxload for each routing scheme when the numbers of flows varies and LB-DRR clearly dominates both SPA and wt-ECMP. By varying the connectivity level of the network (see Figure 2b), we observed that LB-DRR performs better as the network connectivity increases and its Maxload decreases significantly. Note that higher connectivity brings about longer run-time overhead due to the increasing number of routes to be considered.

Figure 3a illustrates the scalability of LB-DRR w.r.t. increasing number of flows. Regarding the increase of the number of flows, we observed that LB-DRR scales linearly, but very slowly (it took only 26 seconds to compute routes

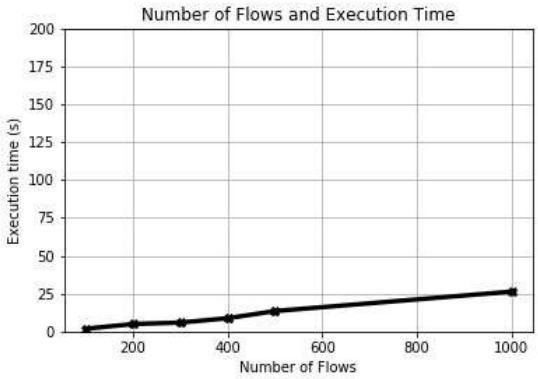


(a) Load balancing: LB-DRR vs. SPA and wt-ECMP.

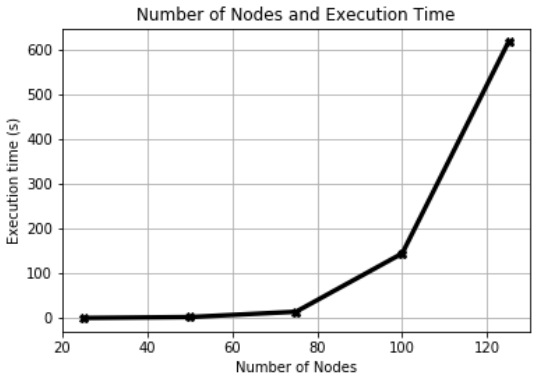


(b) Performance improvement w.r.t. network connectivity.

Figure 2: Load balancing and performance improvement of LB-DRR**.**



(a) Scalability w.r.t. number of flows.



(b) Scalability w.r.t. number of nodes.

Figure 3: Scalability of LB-DRR**.**

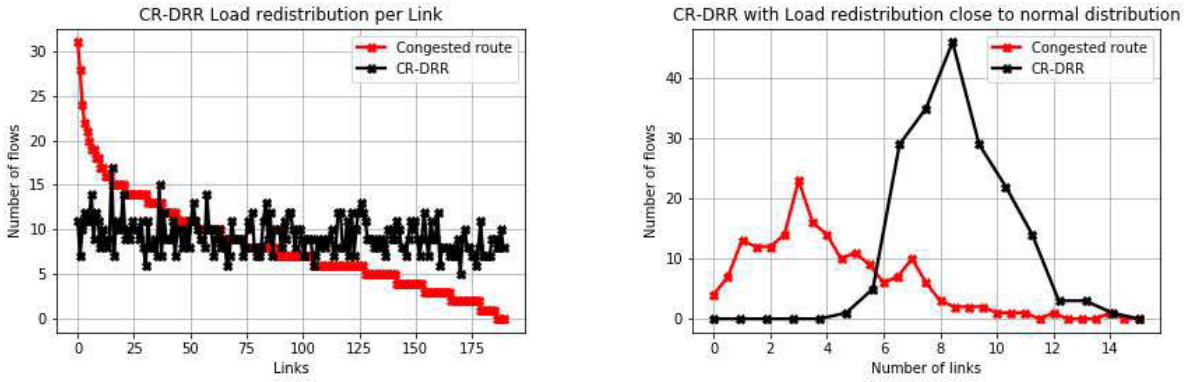
for 1000 flows). Now, regarding the increase of the number of nodes, we set the network connectivity level to 0.2 and consider 100 real-time flows. Figure 3b shows that the execution time of LB-DRR grows exponentially as the number of nodes exceeds 75. However, it could still compute routes for 125 nodes in 11 minutes.

From the second batch of experiments, we routed 750 real-time flows by using SPA, and we observed a huge congestion on the selected routes. Figure 4 shows the congestion recovery and the load redistribution results. In Figure 4a, the network load was initially unbalanced (see the red curve) with several flows routed only on a limited number of edges (see the peak on the far-left), while several edges were left unused (see the long tail to the far-right). By applying the CR-DRR scheme, a tremendous improvement has been observed (see the black curve). Figure 4b shows the load distribution of the congested network before and after CR-DRR is applied. From this figure, the load distribution curve of CR-DRR is close to the normal distribution. Finally, CR-DRR presented the same behavior as LB-DRR in terms of scalability.

5. RELATED WORK

Traffic routing of time sensitive (real-time) flows is non trivial [9]. Routing optimization has been well studied in literature and sophisticated techniques have been proposed [6, 13]. However, contributions on TSN routing schemes have started less than a decade ago. In this context, both the rapid spanning tree protocol and shortest path bridging algorithms have widely been adopted in practice [11]. On another front, the IEEE802.1 Qca standard [8] specifies the Constrained Shortest Path First routing algorithm for TSN transmissions, but this algorithm does not prevent congestion situations and can increase contention in the network, unfortunately. Arif and Atia [1] proposed a methodology to evaluate the routes of a TSN end-to-end connection, but load-balancing was not part of their objectives.

Nayak et al. [10] explored ILP-based algorithms for routing time sensitive flows in TSN networks with Time Aware Shapers (a.k.a. IEEE-802.1Qbv). The proposed approach in their work differs from ours in that it does not address the congestion and load-balancing problems. Targeting a better load balancing for a TSN network, Singh [12] presented an algorithm, based on meta-heuristics, capable of routing new traffic flows at runtime with minimal overhead. But, the proposed approach adopts the shortest path algorithm (SPA) as initial solution and not all feasible routes are considered. This limits the solution search-space, unfortunately. Gavrilut et al. [4] also took the same path and



(a) CR-DRR adopted to recover from congestion

(b) Load distribution under CR-DRR.

Figure 4: CR-DRR **Congestion recovery.**

proposed heuristic methods for topology and routing synthesis. Their method tries to achieve an optimal usage of the switches and links as well as an efficient routing of flows. However, they did not consider load-balancing. This paper fills this gap: it solves the problem of load-balancing, disjoint routing for duplicated flows and dynamic re-routing in congestion situation.

6. CONCLUSION

In this work, we proposed two routing heuristics, referred to as LB-DRR and CR-DRR, in order to address the problems of load-balancing and congestion in TSN-based networks. We evaluated the performance of the proposed schemes against the popular SPA and wt-ECMP routing algorithms and showed an improvement of more than 70% and 20%, respectively. This improvement has been observed w.r.t. the maximum load transmitted on an edge. On another front, the proposed heuristics exhibited high scalability w.r.t. an increase in the number of flows. Given these promising results, we plan to investigate both the routing and scheduling of time-sensitive flows simultaneously. Also, it would be interesting to address multicast flows; develop techniques to reduce the search space of valid routes as the number of nodes increases and finally quantify the impact of these techniques on the end-to-end timing requirements.

Acknowledgment

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UID/CEC/04234)

7. REFERENCES

[1] F. A. R. Arif and T. S. Atia. Load balancing routing in time-sensitive networks. In *Problems of Inf. Science and Technology (PIC S&T), Third Int. Scientific Practical Conf.*, pages 207–208. IEEE, 2016.

[2] F. Dürr and N. G. Nayak. No-wait packet scheduling for ieee time-sensitive networks (TSN). In *Proc. of the 24th Int. Conf. on RTNS*, pages 203–212. ACM, 2016.

[3] L. Erdős, A. Knowles, H.-T. Yau, J. Yin, et al. Spectral statistics of erdős–rényi graphs i: local semicircle law. *The Annals of Probability*, 41(3B):2279–2375, 2013.

[4] V. Gavrilut, B. Zarrin, P. Pop, and S. Samii. Fault-tolerant topology and routing synthesis for IEEE time-sensitive networking. In *25th Int. Conf. on RTNS*, pages 267–276. ACM, 2017.

[5] F. Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[6] M. D. Grammatikakis, D. F. Hsu, M. Kraetzl, and J. F. Sibeyn. Packet routing in fixed-connection networks: A survey. *Journal of Parallel and Distributed Computing*, 54(2):77–132, 1998.

[7] IEEE. Time-Sensitive Networking Task Group.

[8] IEEE. *IEEE Standard for Local and metropolitan area networks— Bridges and Bridged Networks - Amendment 24*. IEEE, 2016.

[9] A. Nasrallah, A. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. Elbakoury. Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research. *IEEE Comm. Surveys & Tutorials*, pages 1–59, 2018.

[10] N. G. Nayak, F. Duerr, and K. Rothermel. Routing Algorithms for IEEE802. 1Qbv Networks. *RTN workshop, ECRTS*, 2017.

[11] P. Pop, M. L. Raagaard, S. S. Craciunas, and W. Steiner. Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks. *IET Cyber-Physical Systems: Theory & Applications*, 1:86–94, 2016.

[12] S. Singh. Routing Algorithms for Time Sensitive Networks. Master's thesis, Univ. of Stuttgart, 2017.

[13] B. Wang and J. C. Hou. Multicast routing and its qos extension: problems, algorithms, and protocols. *IEEE network*, 14(1):22–36, 2000.

[14] Z. Wang and J. A. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Sel. Areas in Comm.*, 14:1228–1234, 1996.