Scheduling Tool for Deterministic Communication in Distributed Real-Time Systems

Jan Mach[1], Valéria Tašková[2] and Lukáš Kohútka[2]

[1] *Institute of Computer Engineering and Applied Informatics, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Slovakia*
[2] *Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Slovakia*

Abstract
A network with critical data streams, where the timing of incoming and outgoing data is a necessity, is called a deterministic network. These networks are mostly used in association with real-time systems that use periodic hard-real-time data which need to be scheduled. The algorithm proposed in this paper is focused on achieving the lowest possible latency within the specified network topology depending on the physical parameters of the network and the periodic requirements of the communicating nodes. The algorithm allows to add priorities to the data flow between nodes and also the option to specify if preemption is allowed within the network, since not all devices support it. The result is a schedule of hard-real-time data communication within a specified network topology where all requirements of communicating nodes are met if such a solution is possible.

Keywords
deterministic scheduling, time-sensitive networking, scheduling algorithm, Ethernet, deadline

1. Introduction

Every day more and more devices get connected to the Internet and other networks. These devices all have different requirements, regarding quality and timing of data transmission. Some of them need to transfer time-sensitive data, which means that not only does it matter that the data was received, it also matters when it was received by the destination. While we are perfectly fine if a video, we stream is lagging a bit, may that be a bit troublesome, but if data about e.g., an overheating component of a machine gets delayed, that may have some more serious consequences [1-5]. Time critical data can be categorized into soft and hard real time data, the difference lies in the abiding of the deadline, in which a task must be completed. In hard real-time systems, missing a deadline is considered a failure of the whole real-time system. In large networks with many endpoints scheduling time sensitive traffic can be problematic, because of conflicts that can arise if more than one endpoint wants to send data over the same path. To solve these conflicts and create a schedule for communicating endpoints the need for an algorithm has arisen. In this paper we will discuss communication planning procedures and propose a solution for scheduling data in a deterministic network.

2. Related work

Publicly available tools for planning deterministic communication do not exist at the time of writing this paper. However, there is a large number of communication planning and shaping procedures that we will introduce in this section [6].

SQAMIA 2023: Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, September 10--13, 2023, Bratislava, Slovakia
jan.mach@stuba.sk (J. Mach); valeria.taskova@stuba.sk (V. Tašková); lukas.kohutka@stuba.sk (L. Kohútka)
0000-0003-4810-1655 (J. Mach); 0009-0008-2622579 (V. Tašková); 0000-0002-5679-6250 (L. Kohútka)

2.1. Quality of Service and priorities

The definition of Quality of Service (QoS) is not entirely unambiguous, as each technology or service defines it according to its needs. The most widely known standards in this area are the IEEE 802.1p and 802.1Q standards and their definition of QoS and their assignment of priorities [7, 8]. According to these standards IEEE proposed eight priority levels where 0 is the lowest priority and 7 is the highest. A recommended type of traffic was also assigned to these priorities.

Zero priority traffic is recommended to be Best Effort traffic and the highest priority should be assigned to Network control traffic. Using these priorities, traffic can be shaped. The priority information is in the four-byte 802.1Q header, which is inserted into the Ethernet frame between the source MAC address and Ethertype fields [9].

2.2. Strict priorities

The algorithm for strict priorities is supported by CISCO routers. These routers support priorities. Each interface supports multiple streams which correspond to these priorities. According to the strict priority algorithm the higher priority packets get transmitted first. Packages from lower priority streams are not processed until packages from a higher priority stream are sent [10].

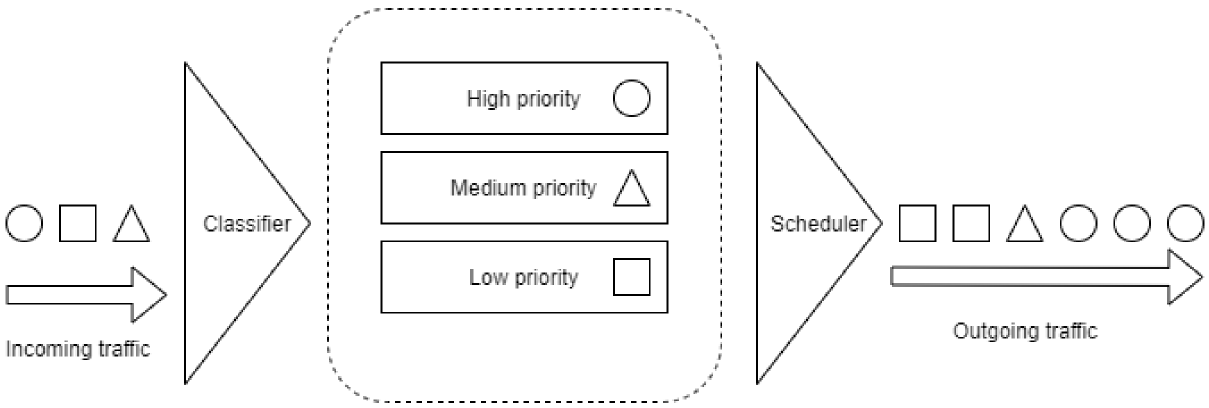


Figure 1: Example of streams with strict priorities. Source: [11].

2.3. Weighted fair queuing

There is a number of variations of round-robin scheduling algorithms for data streams, which allow each stream to have a fair share of resources. It is possible to fairly distribute transmissions from all streams, however, fairness does not mean that all streams can send the same number of packages [12]. This depends on what the algorithm deems as the weight (e.g. earliest finish time of processing). Priorities could be added to ensure that higher priority streams get to send more, in which case these higher priority streams will be served for longer than lower priority streams [13].

2.4. Hierarchical scheduling

Hierarchical scheduling is a framework that is supported and described by CISCO. It allows service providers to manage their QoS at three or four levels of the hierarchy. Instead of allocating an implicit resource guarantee to each queue, the three-level scheduling parameter uses the ratio of the remaining resources to allocate unused resources (such as bandwidth) to each logical queue [14].

The three-level hierarchy would look like this [14]:

- Physical Layer - Used to shape a physical interface, such as an OC-3 port.
- Logical Layer - Used to schedule child interfaces, such as VLAN or PPP sessions.
- Class layer - Used for series classes defined by QoS policy.

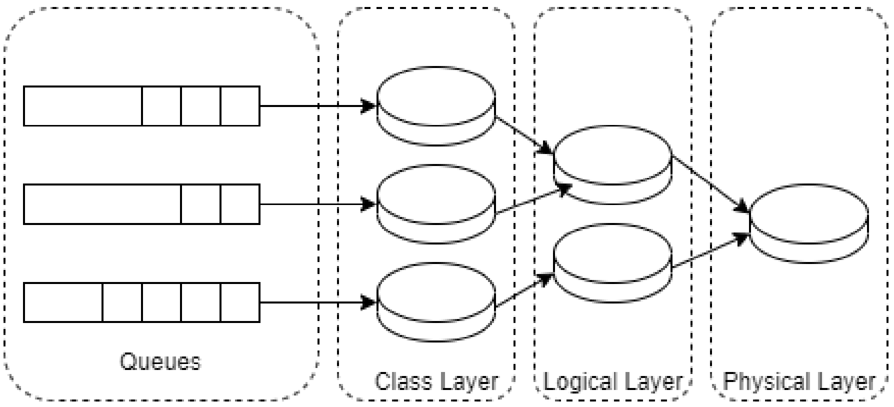


Figure 2: Example of streams with strict priorities. Source: [11].

2.5. Credit-based scheduling

Credit-Based Traffic Shaping is used primarily for forwarding audio and video frames. The individual frames are organized into streams according to their priorities. Each stream is then assigned with a specific credit score. Frames are only sent if this credit is greater than or equal to zero. This traffic shaping is a great solution for achieving smooth video and audio transmissions. It adds flexibility as new data can be dynamically added to streams [15].

2.6. Time aware scheduling and preemption

Time aware scheduling is used primarily in industrial networks, where we can specify the reference time (defined according to IEEE 802.1AS) and the list of events. The event list consists of a time interval and the event itself. The event may be, for example, opening a port [16]. Of course, there may be a situation where a lower priority frame is still transmitted at the time of the event. For this case, a so-called guard zone is introduced. This guard zone is as large as the largest possible frame that could be transmitted and interfere with the planned communication. In case this band is too large, preemption is going to be used on the frame with lower priority. It is going to be divided into smaller fragments and the guard zone must therefore only be the size of the largest possible fragment [17, 18].

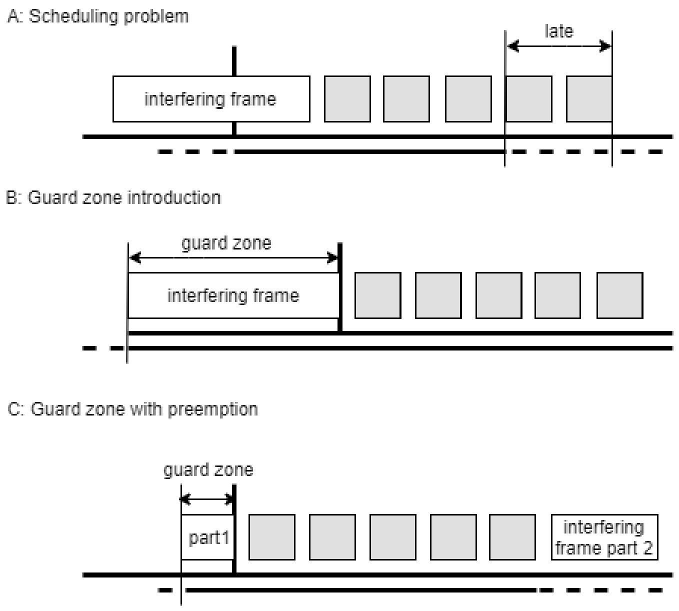


Figure 3: Visualization of Time aware scheduling. Source: [17].

3. Proposed solution

The previously mentioned methods focused primarily on prioritizing and scheduling data, in a way that they could determine, if the scheduling is possible. However, they would be either not suitable for larger networks or they did not guarantee that all data, even the lowest priorities, would get transmitted in time.
The input for the algorithm contains data about the allowance of preemption, the topology of the network, such as the throughput, and latency as well as data about the communication pairs (period in which data is being sent, the data size and the priority of it).
The program then creates a weighted graph, where the weight of the edges is equal to the latency between nodes. For each communication pair the shortest path is then found and saved. Taking into consideration these shortest paths, the number of conflicts on the edges is then computed. If two or more communication pairs have the same edge in their shortest path, a schedule has to be made [19].

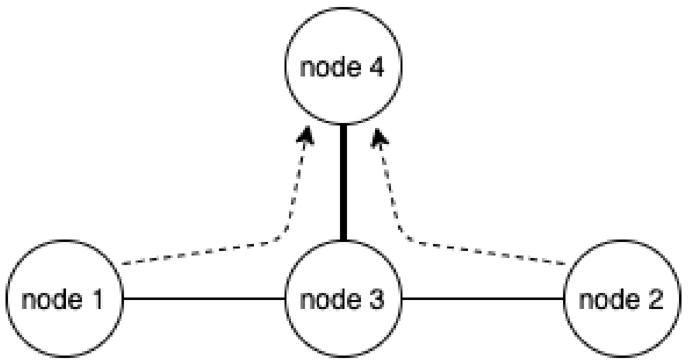


Figure 4: Visualization of a path with conflict where node 1 and 2, want to send data to node 4. Source: author's contributions.

The algorithm labels all edges with conflicts on them as unresolved. It then starts with the edge that has the most communication pairs wanting to send data through them. At first a period has to be set, which is the lowest common multiple of all the periods of communication pairs that use that edge. Next the algorithm calculated the time that is needed to send all the data from all of the communication pairs. If this time is lower than the calculated period, then offsets for each communication pair that uses the edge are calculated, starting with the ones that have the highest priority.

If the time is greater than the calculated period, then a different route than the shortest has to be found, with emphasis on the lowest possible overall latency of the network. The algorithm uses hill-climbing to search for this solution.

In case that while solving a collision edge where one of the communication pairs already has an offset that has been set in earlier iterations, the option of preemption plays a role. If preemption is allowed in the topology, then data can be split into more packets to accommodate the already set offset. If preemption is not an option, then the already set offset is altered to be the last offset on the collision edge and a "backwards check" is triggered. This is to see if the change has had impact on any previously solved conflict edges. If yes, then the offset is once again set anew to be the last offset on that already set edge and the control is triggered anew. This control is running until either an offset is found that does not interfere with any other data offsets, or until a solution is found. The proposed offset plus the time to send data has to be greater than the period on that conflict edge in which case a new path using hill-climbing has to be found. The algorithm either finds a schedule solution, where all communication pair prerequisites have been met with the lowest possible network latency, or informs the user that no solution is possible.

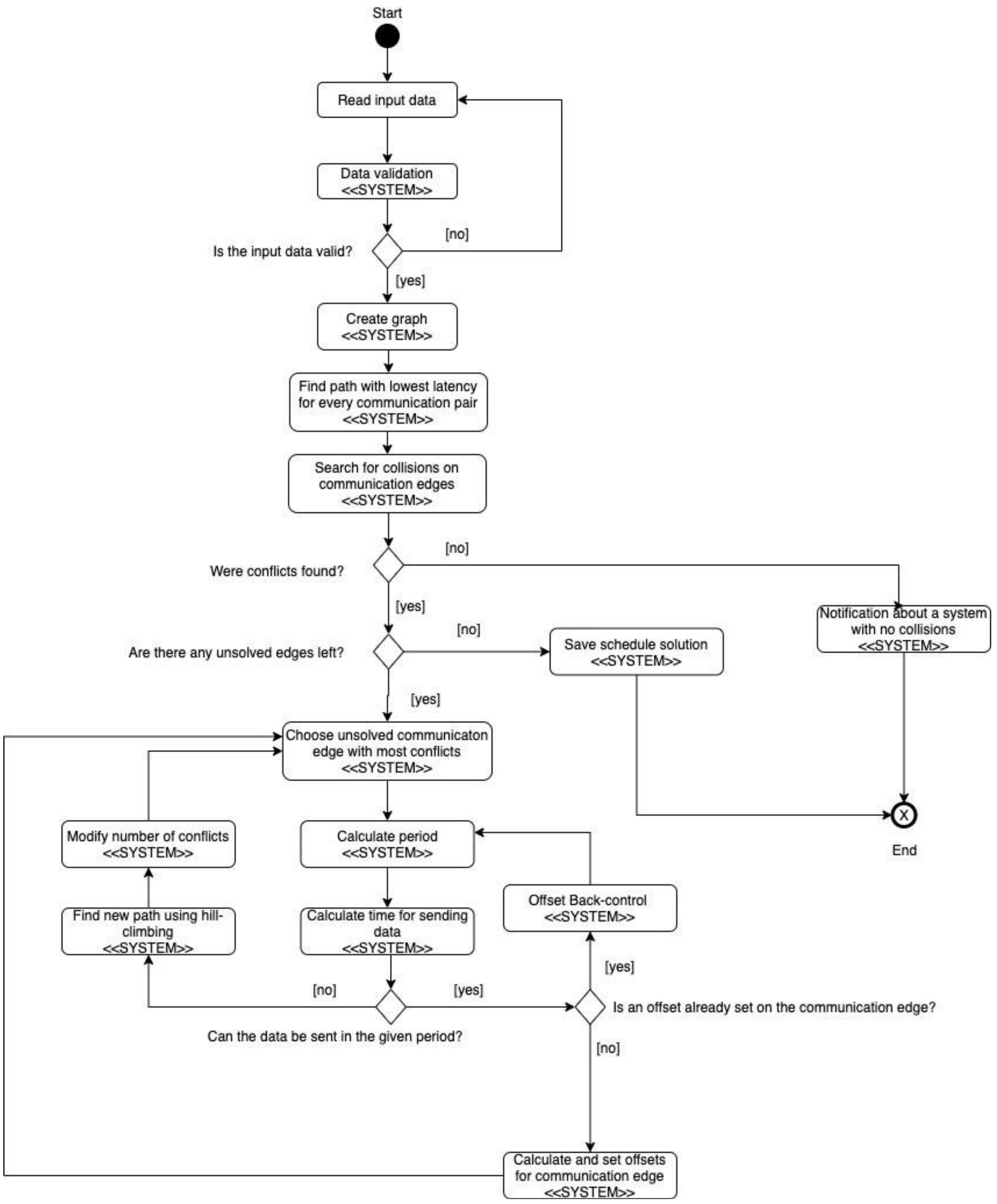
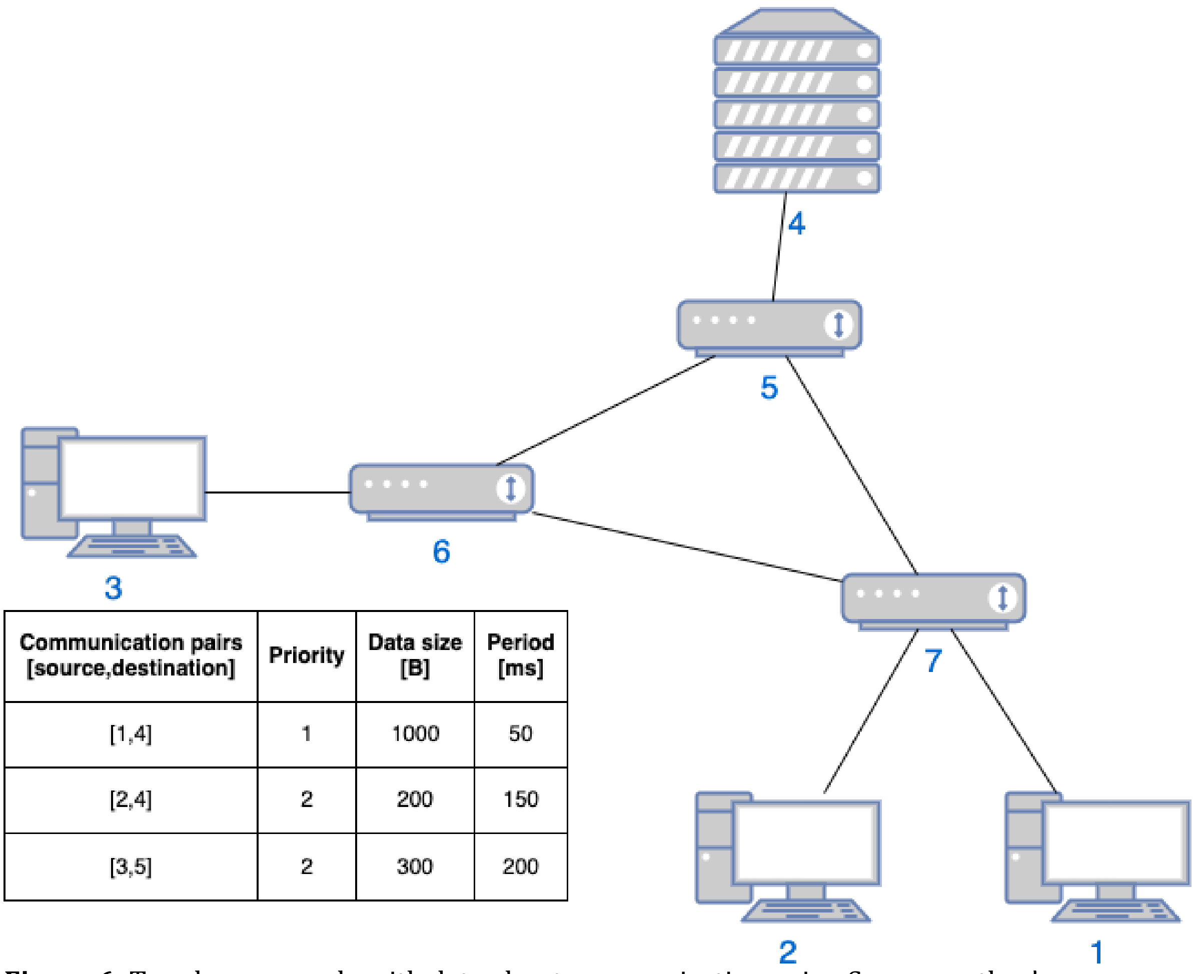

Start
Read input data
Data validation <<SYSTEM>>
Is the input data valid?
[no]
[yes]
Create graph <<SYSTEM>>
Find path with lowest latency for every communication pair <<SYSTEM>>
Search for collisions on communication edges <<SYSTEM>>
Were conflicts found?
[no]
[yes]
Notification about a system with no collisions <<SYSTEM>>
Are there any unsolved edges left?
[no]
Save schedule solution <<SYSTEM>>
[yes]
Choose unsolved communicaton edge with most conflicts <<SYSTEM>>
Modify number of conflicts <<SYSTEM>>
Calculate period <<SYSTEM>>
End
Offset Back-control <<SYSTEM>>
Find new path using hill-climbing <<SYSTEM>>
Calculate time for sending data <<SYSTEM>>
[yes]
[no]
[yes]
Is an offset already set on the communication edge?
Can the data be sent in the given period?
[no]
Calculate and set offsets for communication edge <<SYSTEM>>

Figure 5: Activity diagram of the proposed solution. Source: author's contribution.

3.1. Schedule example

To demonstrate the algorithm on an example, lets propose a simple topology as illustrated in Figure 6. Both node 1 and node 2 want to send data to node 4 and node 3 wants to send data to node 5. The size of the data and period of the communication are displayed in the table.



Communication pairs [source,destination]	Priority	Data size [B]	Period [ms]
[1,4]	1	1000	50
[2,4]	2	200	150
[3,5]	2	300	200

Figure 6: Topology example with data about communication pairs. Source: author's contribution.

The algorithm then creates a weighted graph of the topology as shown in Figure 7. It then calculates the shortest paths from the source to the destination using Dijkstra's algorithm, where the weight of the edges is equal to the latency on that edge. In the next step it looks for conflicts on the edges. In this example both communication pairs 1-6 and 4-6 have the edge C5 7 in their shortest path. This means that this edge is marked as unresolved, and the algorithm tries to create a schedule based on the period of both communication pairs.

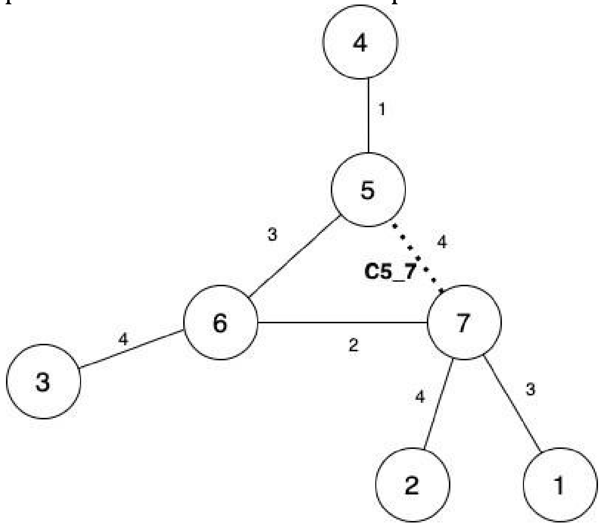


Figure 7: Weighted graph of the topology. Source: author's contribution.

Since the communication pair 1−4 wants to send a big amount of data in a short time frame, it is not possible to also schedule the other communication pair. The pair 2−4 would still be transmitting data and a conflict would arise (Figure 8). Since the communication pair 1−4 has a higher priority, the algorithm tries to reschedule the pair 2−4.

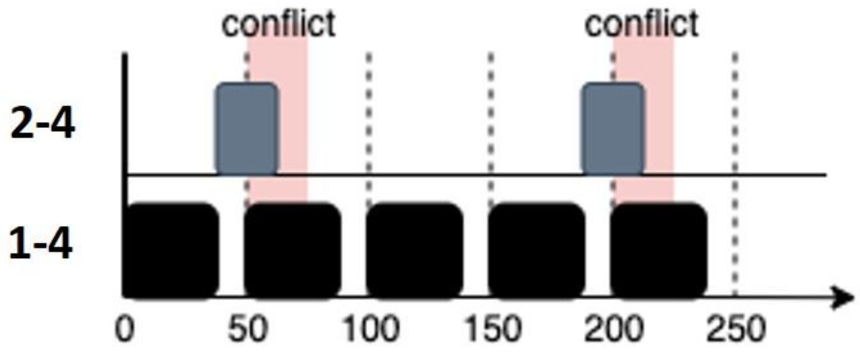


Figure 8: Example of schedule with conflicts on two different links. Source: author's contribution.

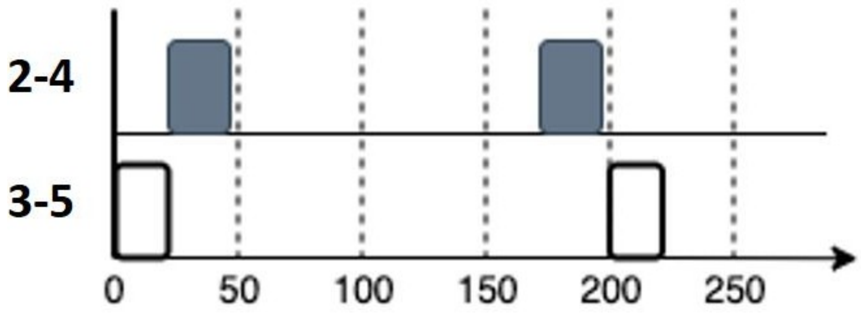
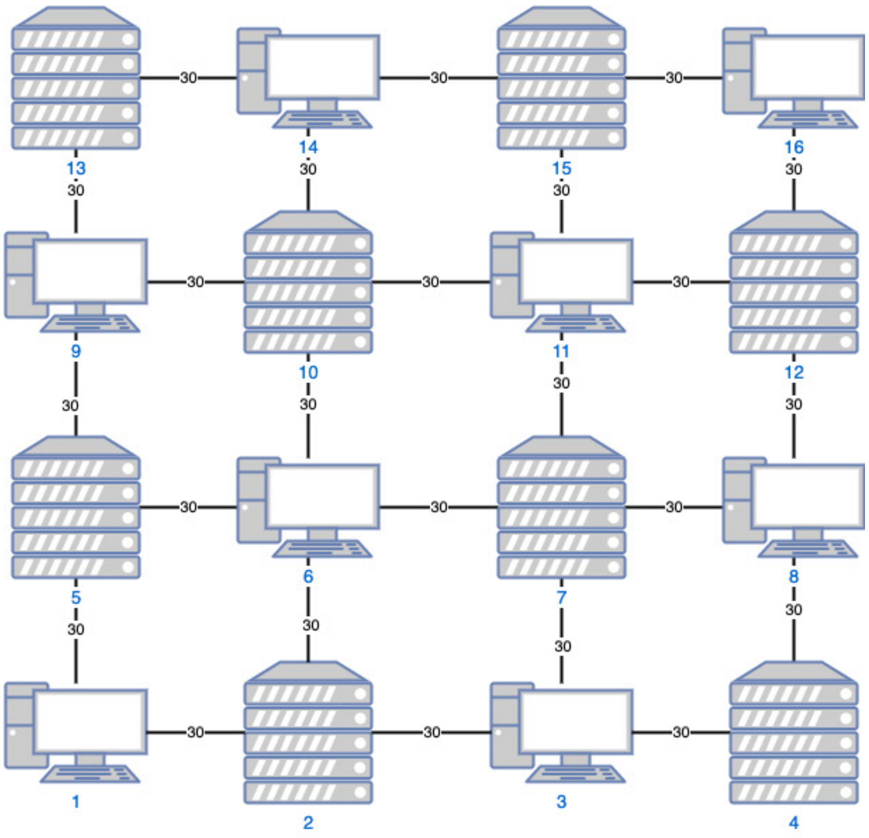


Figure 9: Example of schedule without conflicts for two different links. Source: author's contribution.

The algorithm finds a new path for the communication pair 2−4, if such a path exists, with emphasis on the lowest possible latency. This time the new path would create a conflict on the edge between node 6 and node 5. This new edge is then marked as unresolved, and a schedule based on the period of both communication pairs is calculated. In this example both pairs could fit into the period (Figure 9). If a schedule is possible, the algorithm will notify the user of the offsets of this schedule otherwise it would inform the user that a solution does not exist.

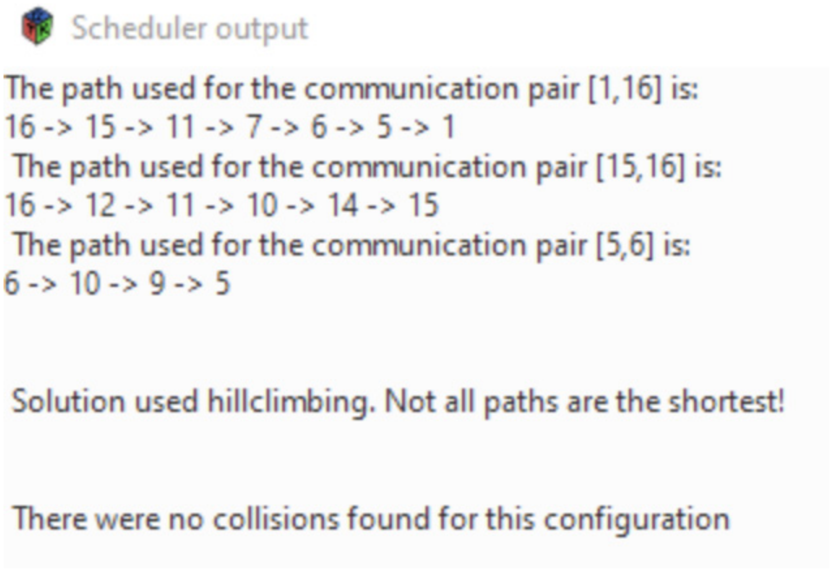
To demonstrate the algorithm further, we present the results for the topology depicted in Figure 10. The blue numbers are identification numbers of nodes, be it a network switch or endpoint node. The black numbers on lines represent the transmission delays of individual links between a pair of two nodes.

We created a communication pair with a higher priority that would take the whole period to transmit. This way, the other communication pairs would have to be transferred through a different route. The algorithm is able to find new paths that ensure that the latency as a whole is the lowest possible and creates an output schedule, where there are no collisions (Figure 11).



Source	Destination	Period [ms]	Data size [B]	Priority
1	16	5	6000	2
5	6	5	6000	2
1	16	30	600	1

Figure 10: Example of topology without collisions after finding a new path. Source: author's contribution.



```
The path used for the communication pair [1,16] is:
16 -> 15 -> 11 -> 7 -> 6 -> 5 -> 1
 The path used for the communication pair [15,16] is:
16 -> 12 -> 11 -> 10 -> 14 -> 15
 The path used for the communication pair [5,6] is:
6 -> 10 -> 9 -> 5


 Solution used hillclimbing. Not all paths are the shortest!


 There were no collisions found for this configuration
```

Figure 11: Scheduler output for topology displayed in Figure 9. Source: author's contribution.

3.2. Comparison to other approaches

The algorithms and approaches mentioned in Section 2 also support non-scheduled traffic, which our solution does not support. In order to properly compare our proposed solution, we also needed a topology that has throughput and latency described. Chen et al. [20] described their credit-based low latency packet scheduler (CBLLPS) on the bottleneck topology shown in Figure 12. Nodes marked S are source nodes, D are destination nodes and Rs and Rd are nodes that represent routers. Figure 13 illustrates queuing delays in their CBLLPS compared to other approaches (LLEPS = Low Latency and Efficient Packet Scheduling, SFQ = Start-time Fair Queuing, WF2Q+ = Worst-case fair weighted fair queueing, SCFQ = Self-clocked fair queueing, NDRR = Nested deficit round robin). Using three 1Mbps flows CBLLPS is capable of sharing bandwidth.

We recreated the topology where nodes 1-10 are source nodes, 11-20 are destination nodes and nodes 21-22 represent the routers. For our solution to create a schedule, the bottleneck resource needs to be evenly distributed between all communication pairs. This means that if we were to transmit 1 MB data from all sources to all destinations a schedule would not be possible. The credit-based scheduler on the other hand is capable of scheduling all the data, but there would be delays, which is not allowed for hard-real time data. To send all more data, either the allowed period should be greater, or a cable with a bigger throughput should be used.

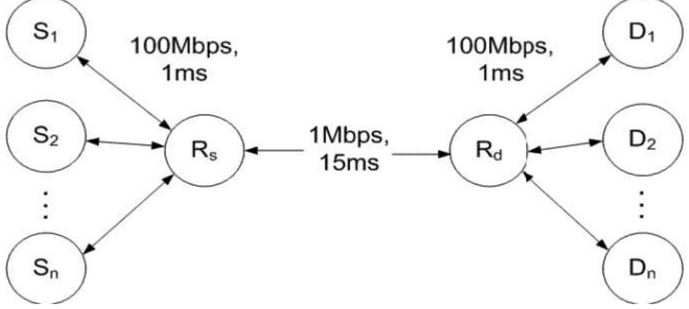


Figure 12: Topology used for comparison. Source: [20].

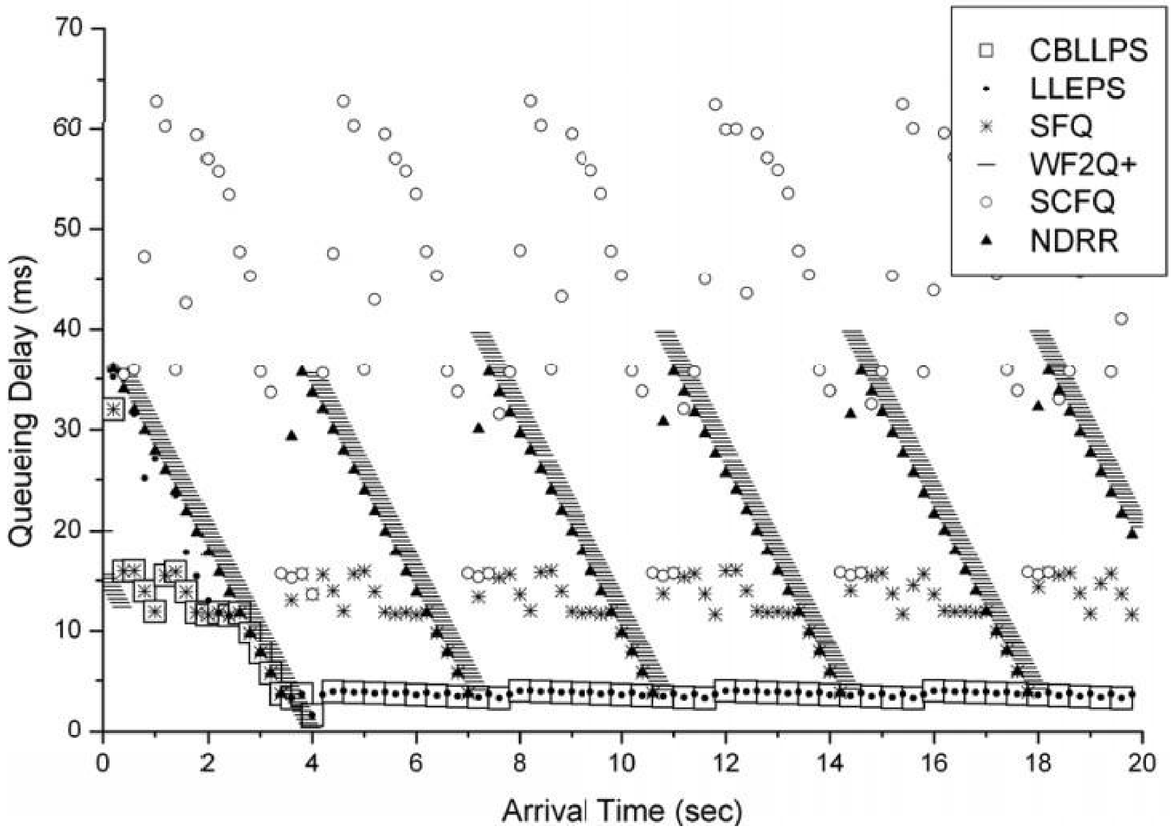
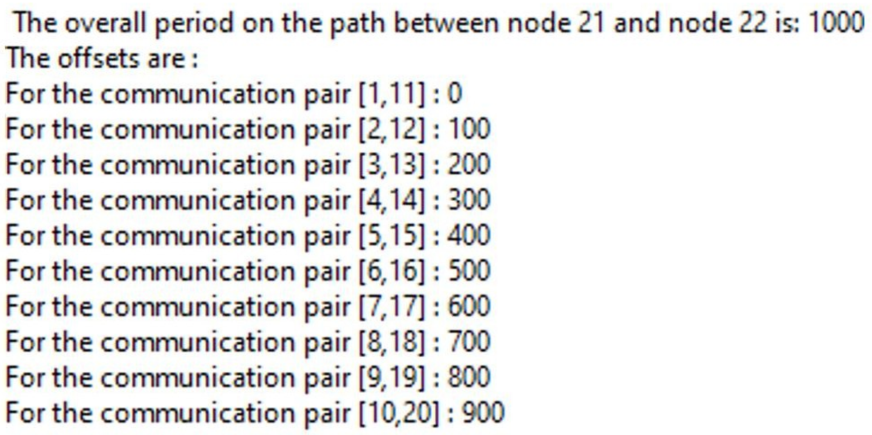


Figure 13: Queuing delays on the conflict edge using different approaches. Source: [20].



```
The overall period on the path between node 21 and node 22 is: 1000
The offsets are :
For the communication pair [1,11] : 0
For the communication pair [2,12] : 100
For the communication pair [3,13] : 200
For the communication pair [4,14] : 300
For the communication pair [5,15] : 400
For the communication pair [6,16] : 500
For the communication pair [7,17] : 600
For the communication pair [8,18] : 700
For the communication pair [9,19] : 800
For the communication pair [10,20] : 900
```

Figure 14: Schedule generated by proposed scheduling tool. Source: author's contribution.

In Table 1. we both solutions are compared. Our first point of comparison is the time to create a schedule for the data, in the given topology. For the given topology, our algorithm was able to create an output in less than a second. The credit-based scheduler does not create any schedule, it just assigns credits to every communication pair. The second point of comparison was the time in which data is transmitted. Our solution creates a predictable schedule, where every communication pair is given an offset and every period data is sent at time of the given offset. The Credit-based scheduler sends data of the communication pairs when there are credits available to the pair, this means data transmission time cannot be predicted at all. We also measured the success rate of the data transmission schedule. If every communication pair wants to send more than 0,1MB/s our solution is not able to create a schedule, while the credit-based scheduler sends data even with a delay. If the data to send for every communication pair is 0,1 MB/s or less, the proposed solution is able to schedule all pairs. The credit-based scheduler is also able to send all data, but the periods of the communication pair are not fulfilled. Regarding the delay, the proposed solution is an all-or-nothing approach, it either can schedule everything without a delay or is not able to create a schedule. The credit-based scheduler has an average of 7s of delay. The next point of comparison is predictability or the ability to send data periodically. As mentioned above, the proposed solution supports this kind of data while the credit-based scheduler does not. The credit-based scheduler also does not take any network latency into consideration, while the proposed solution would search for the lowest latency path for every communication pair.

Table 1
Comparison of proposed solution with credit-based scheduling for given topology.

Comparison in:	Proposed solution	Credit-based scheduling
Time to create schedule	less than 1s	none
Data transmission time	based on offsets and periods	when there are credits
Success rate 1Mb/s per pair	could not create schedule	transmission with delay
Success rate 0.1Mb/s per pair	scheduled all pairs	transmission not predictable
Delay	none if schedule is possible	7ms in average
Periodicity/predictability	according to schedule	not possible to plan
Network latency	the lowest possible	not considered

4. Future work

Scheduling of communication between endpoints over Ethernet in the domain of real-time distributed systems is closely linked to task scheduling in individual endpoints and therefore it should be seen as one complex distributed real-time system where there are coordinated task plans in individual endpoints as also communication between these points using this solution. The aim for the future is to combine this proposed solution with real-time distributed systems.

5. Conclusion

In this paper we proposed a new scheduling tool for deterministic networks that creates a schedule with emphasis on the overall lowest latency of the network. In general, our proposed solution is not suitable for systems with best effort traffic, or with non-periodical data. Our proposed scheduling tool would not be able to create a schedule at all in the example of topology we used even if one communication pair wants to send more data.

For these cases, other solutions such as the abovementioned credit-based scheduler is more suitable. On the other hand, for real-time networks or for systems with hard-real time behavior where no delays of data transmission over network is allowed, our proposed solution is more suitable, since it creates hard offsets that do not allow any delays and all data are guaranteed to be sent on time, and no queuing is needed.

Acknowledgements

The work reported here was supported by the Operational Programme Integrated Infrastructure for the project Advancing University Capacity and Competence in Research, Development and Innovation (ACCORD) (ITMS code: 313021X329), co-funded by the European Regional Development Fund (ERDF). This publication was also supported in part by the Slovak national project KEGA 025STU-4/2022, APVV-19-0401 and APVV-20-0346. This article was also written thanks to the generous support under the Operational Program Integrated Infrastructure for the project: "Support of research activities of Excellence laboratories STU in Bratislava", Project no. 313021BXZ1, co-financed by the European Regional Development Fund.

References

[1] Buttazzo, G.C. Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications; Springer: New York, NY, USA, 2011.

[2] Caccamo, M.; Buttazzo, G. Optimal scheduling for fault-tolerant and firm real-time systems. In Proceedings of the Fifth International Conference on Real-Time Computing Systems and Applications (Cat. No.98EX236), Hiroshima, Japan, 27–29 October 1998; pp. 223–231.

[3] Buttazzo, G.; Conticelli, F.; Lamastra, G.; Lipari, G. Robot control in hard real-time environment. In Proceedings of the Fourth International Workshop on Real-Time Computing Systems and Applications, Taipei, Taiwan, 27–29 October 1997; pp. 152–159.

[4] Spuri, M.; Buttazzo, G.; Sensini, F. Robust aperiodic scheduling under dynamic priority systems. In Proceedings of the 16th IEEE Real-Time Systems Symposium, Pisa, Italy, 5–7 December 1995; pp. 210–219.

[5] Buttazzo, G.; Stankovic, J. Adding Robustness in Dynamic Preemptive Scheduling. In Responsive Computer Systems: Steps toward Fault-Tolerant Real-Time Systems; Fussell, D.S., Malek, M., Eds.; Kluwer Academic Publishers: Boston, MA, USA, 1995.

[6] Serna Oliver, R., Craciunas, S. S., Stoger, G.: Analysis of Deterministic Ethernet scheduling for the Industrial Internet of Things. 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, (2014)

[7] Craciunas, Silviu S., Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. 'Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks'. In Proceedings of the 24th International Conference on Real-Time Networks and Systems, 183–92. RTNS '16. New York, NY, USA: Association for Computing Machinery, 2016. https://doi.org/10.1145/2997465.2997470.

[8] Cisco Systems, Inc: Internetworking Technology Handbook, chap. QoS, http://index-of.co.uk/Tutorials/Networking%20set%20of%20books.pdf.
[9] Huawei, Inc: IEEE 802.1Q Frame Format, https://support.huawei.com/enterprise/en/doc/EDOC1100088104.
[10] Cisco Systems, Inc: Configuring QoS Queues, https://www.cisco.com/assets/sol/sb/SG220 Emulators/SG220 Emulator v10018 201406.
[11] Kamayseh, Y.: Strict Priority Scheduler (2011), https://www.researchgate.net/figure/StrictPriorityScheduler fig2 51953677.
[12] Yang, Li, Chengsheng Pan, Erhan Zhang, and Haiyan Liu. 'A New Class of Priority-Based Weighted Fair Scheduling Algorithm'. Physics Procedia 33 (2012): 942–48. https://doi.org/10.1016/j.phpro.2012.05.158.
[13] Qadeer, M. A. Javaid, A. Y., Habib S.: Apportioning Bandwidth to Clients as per Privileges. International Journal of Future Generation Communication and Networking, 2009.
[14] Cisco Systems, Inc: Hierarchical Scheduling and Queuing https://www.cisco.com/c/en/us/td/docs/ios/12 2sb/feature/guide/3levelsch.html.
[15] Thangamuthu, S.: Analysis of automotive Traffic shapers in Ethernet In-Vehicular Networks (2015)
https://pdfs.semanticscholar.org/2307/00933e09df00fadaaf812ff3ad4fc97a087d.pdf.
[16] Boiger, Ch.: Time Aware Shaper (2012), http://www.ieee802.org/1/files/public/docs2012/bvboigertimeawareshaper0712v01.pdf
[17] M. H. Farzaneh and A. Knoll, "Time-sensitive networking (TSN): An experimental setup," 2017 IEEE Vehicular Networking Conference (VNC), Turin, Italy, 2017, pp. 23-26, doi: 10.1109/VNC.2017.8275648.
[18] Houtan, B., Ashjaei, M., Daneshtalab, M., Sjodin, M., Mubeen, S.: Work in Progress: Investigating the Effects of High Priority Traffic on the Best Effort Traffic in TSN Networks. 2019 IEEE Real-Time Systems Symposium (RTSS), 2019.
[19] Bucsiova, V.: A new scheduling algorithm for Real-time networks, IIT.SRC 2017.
[20] Chen, L.-H., Wu, E. H.-K., Hsieh, M.-I., Horng, J.-T., Chen, G.-H.: Credit-based low latency packet scheduling algorithm for real-time applications. 2012 IEEE International Conference on Communication, Networks and Satellite (ComNetSat), 2012.