Improvements to Deep-Learning-based Feasibility Prediction of Switched Ethernet Network Configurations

Tieu Long Mai
University of Luxembourg
Esch-sur-Alzette, Luxembourg
long.mai@uni.lu

Nicolas Navet
University of Luxembourg
Esch-sur-Alzette, Luxembourg
nicolas.navet@uni.lu

ABSTRACT

Graph neural network (GNN) is an advanced machine learning model, which has been recently applied to encode Ethernet configurations as graphs and predict their feasibility in terms of meeting deadlines constraints. Ensembles of GNN models have proven to be robust to changes in the topology and traffic patterns with respect to the training set. However, the moderate prediction accuracy of the model, 79.3% at the lowest, hinders the application of GNN to real-world problems.

This study proposes improvements to the base GNN model in the construction of the training set and the structure of the model itself. We first introduce new training sets that are more diverse in terms of topologies and traffic patterns and focus on configurations that are difficult to predict. We then enhance the GNN model with more powerful activation functions, multiple channels and implement a technique called global pooling. The prediction accuracy of ensemble of GNNs with a combination of the suggested improvements increases significantly, up to 11.9% on the same 13 testing sets. Importantly, these improvements increase only marginally the time it takes to predict unseen configurations, *i.e.*, the speedup factor is still from 50 to 1125 compared to schedulability analysis, which allows a far more extensive exploration of the design space.

KEYWORDS

Machine learning, Graph Neural Network, Schedulability analysis, Design Space Exploration, Time-Sensitive Networking.

ACM Reference Format:
Tieu Long Mai and Nicolas Navet. 2021. Improvements to Deep-Learning-based Feasibility Prediction of Switched Ethernet Network Configurations. In *29th International Conference on Real-Time Networks and Systems (RTNS'2021), April 7–9, 2021, NANTES, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3453417.3453429

1 INTRODUCTION

Context. Ethernet is becoming the prominent wired high-speed network technology, be it in the automotive, aerospace or industrial domains. One of the reasons is that Ethernet has been constantly evolving and adapting to successfully address the needs of the new systems being developed. In particular, the IEEE802.1 TSN TG (Time Sensitive Networking Task Group), develops the technologies to address QoS requirements pertaining to timing, reliability and security. This work focuses on the basic timing requirement, which is to ensure that communication latencies are below the deadlines in any possible circumstances.

Deep learning in the design of critical systems. Two well identified use-cases of deep-learning, and machine learning (ML) at large, in the design of critical systems are 1) fast prediction techniques that can replace, at the design-space-exploration stage of the design, exact approaches, and 2) technology-agnostic configuration algorithms, i.e. algorithms not relying on extensive domain knowledge. This study proposes a contribution to the first use-case.

Previous works [18] have explored the use of Graph Neural Network (or GNN for short, see [3]) to predict the feasibility of priority-based TSN configurations. The interactions between flows, links and queues at egress ports of the switches are embedded into a graph neural network model. The GNN model is effectively able to predict the feasibility of unseen TSN network configurations and generalize to different topologies and traffic patterns. The experimental evaluation on 13 independent sets of realistic automotive TSN configurations clearly shows the advantage of GNN over traditional ML algorithms: all sets achieve prediction accuracy higher than 79% with a speedup factor above 70 times compared to schedulability analysis.

Contributions of the paper. In this study, we improve upon [18] by progressively introducing several enhancements. First, we build the training set in such a way that, statistically, it contains more configurations that are known hard to predict from prior experiments in [18]. We also increase the size of the training set as classically done to boost prediction accuracy, which comes with some drawbacks discussed in the paper. We then improve the GNN model with more expressive activation functions and a mechanism called global pooling, and increase the size of the model e.g. with multiple channels, which leads to better performance. Importantly, the execution overhead of these enhancements is modest since the speedup factor remains from 50 to 1125 times compared to network-calculus based schedulability analysis.

Organization of the paper. The remainder of this paper is organised as follows. Section 2 presents the TSN network model. In Section 3, we describe the base GNN model that will serve as a basis for comparison with the successive models. In Section 4, we introduce an improved GNN model, that is trained with an enhanced training set and that embeds advanced ML techniques. In Section 5, we explore the interest of expanding the GNN model. The advantages and drawbacks of a larger training set are also discussed in

RTNS'2021, April 7–9, 2021, NANTES, France

ACM ISBN 978-1-4503-9001-9/21/04…$15.00
https://doi.org/10.1145/3453417.3453429

Table 1: Characteristics of the four types of streams used in [18] to create the TSN networks belonging to the training set. Frame sizes indicated here are data payload only. Each stream possesses one to three receivers. In this study, we re-use these characteristics for the training in Experiment 0. In later experiments, more diversity is introduced in the training sets: proportions are random in range 20% to 40%, size frame is chosen with a granularity of 1 byte and deadlines are chosen in a broader range (from 40% to 100% of periods). The traffic patterns in the testing sets is specific to each case-study and remains the same as in [18].

Command & Control	• from 50 to 150 byte frames, with a step of 10 bytes • periods: one frame each [10,20]ms • deadlines constraints: 20ms • proportion: 20/100
Audio Streams	• from 300 to 500 byte frames, with a step of 100 bytes • periods: one frame each 1ms • deadlines equal to 90% of periods • proportion: 20/100
Video Streams	• either 1000 or 1500 byte frames with burst size in [15, 30] • period: one frame every 30ms • deadlines equal to 90% of periods • proportion: 30/100
Best effort Streams	• from 100 to 1500 byte frames, with a step of 100 bytes • period randomly chosen in [100, 200, 250, 500, 750, 1000]ms • deadlines equal to 90% of periods • proportion: 30/100

this section. Relevant studies about the use of machine learning in real-time systems and Ethernet networks are presented in Section 6. Finally, Section 7 summarizes the main findings of the study and highlights a number of possible research directions.

2 SYSTEM MODEL

We consider switched Ethernet networks comprising switches, full-duplex links and end-nodes. The networks support unicast and multicast communications. In the following, the term *traffic flow* and *traffic stream* refer to a sequence of frames sent from one sender to one or several receivers (*i.e.*, unicast or multicast flows).

2.1 Assumptions on the network

In this study, a number of assumptions about the networks are made:

- The routing in the network is static, as it is the norm in critical systems.
- There is no transmission errors and buffer overflows that would lead to packet losses.
- Streams are either periodic, sporadic or sporadic with bursts (e.g., video streams from cameras).
- An upper bound on the packet switching delay is known. It is set to 1.3us in the experiments, which is in line with [28].
- The maximum size of frames belonging to a stream is known, as required by the schedulability analysis.
- The network topology and links' speed have been decided before the communication needs are entirely known, as it is typically the case in the aerospace and automotive domains.

2.2 Feasibility of TSN configurations

A *configuration* is a TSN network whose parameters have been all set. A configuration is *feasible* if all timing constraints on the flows are met, which are assumed here to be deadline constraints.

The configuration problem in TSN networks involves two main sub-problems: 1) priority assignment: grouping streams into traffic classes and setting the relative priorities of the classes, and, beyond the priorities of the streams, 2) optionally selecting additional QoS mechanisms: Frame Preemption [14], shaping with the Credit-Based Shaper (CBS [12]), time-triggered transmission using the Time-Aware Shaper (TAS [13]), etc.

This work focuses on priority assignment, which is the fundamental QoS mechanism. The Optimal Priority Assignment (OPA) algorithm for mono-processor system [2] has been proven to be optimal for many scheduling problems [5, 6]. Although OPA has not been shown to be optimal with the network calculus-based Worst-Case Traversal Time (WCTT) used in this study, preliminary experiments suggest that it is still very effective. In the experiments, to derive the feasible priority assignment of TSN configurations, we rely on a variant of OPA called "Concise Priorities" [1], available in the RTaW-Pegase software [1].

2.3 Network Topology

The base experimental setup, denoted "experiment 0" in the following (see 3.3), uses the same patterns for topology and traffic as used in [18] for building the training set of the GNN model:

- Star topology: each network has a central switch and two to five switches connected to the central switch.
- Each switch is connected to two to five end-nodes.

[1] "Concise Priorities" differs from OPA only in how unfeasible configurations are handled, which is not relevant in the context of this work.

- The speed of each link is either 100 or 1000Mbit/s.
- The characteristics of the traffic for the base training set are shown in Table 1. This is a typical simplified automotive traffic as used in [4, 21, 23, 26].

To allow for comparison, we re-use the exact same testing sets as in [18] but the training sets may differ in terms of traffic characteristics and shape of the topology. These differences are highlighted where appropriate.

3 SUMMARY OF GRAPH NEURAL NETWORK

This section summarizes the graph neural network (GNN) model for TSN proposed in [18] that is the base model in our study. Over the successive experiments, we introduce several improvements to the model but the encoding of TSN configurations and the learning process remain identical.

3.1 The GNN model

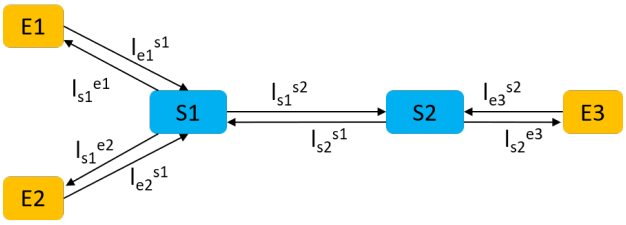
Encoding of TSN configurations. The GNN model considers a TSN configuration as structured data comprised of physical and logical components that affect the feasibility. These are the flows, links and egress queues in the switches, which are connected by the topology and routing. A configuration is encoded as a graph made up of nodes and connections between them. To each node in the graph is associated a vector called the *embedding* that captures some if its characteristics, while the whole graph is characterized by a *global attribute*, which is a vector of the same size as the embeddings. The initialization of embeddings is as follows:

- Each traffic *flow*, be it unicast or multicast, is encoded by one node with the initial embedding capturing the packet payload size, burstiness, transmission period, and deadline.
- the two directions in a bi-directional *link* between a switch and an end-node are represented by two separate nodes whose initial embeddings are the link speed.
- Any waiting *queue* at an egress port of a switch corresponds to a node with the initial embedding set of 0 to indicate that it is empty when generating the configuration.
- The initial values of the global attribute is set to 0.

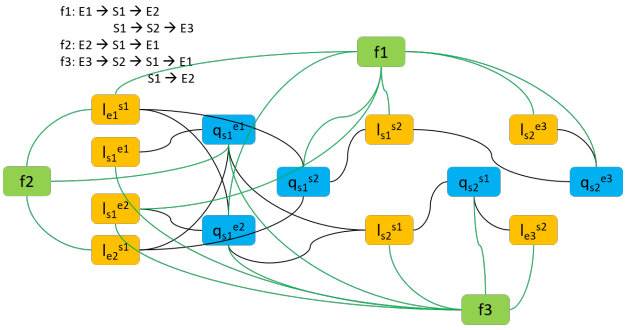
An example of TSN configuration is shown in Figure 1(a) and the graph that encodes this configuration is in Figure 1(b). A *connection matrix* contains the connections, *i.e.*, non-directed edges between nodes, in the graph. The matrix is initialized with 0s, while the elements set to 1 indicate the existence of connections between a queue and a link, a queue and a flow, and a link and a flow. As a special case, to know the direction of a flow, the flow and the link connected to the flow's source end-node is marked by the value 2 in the matrix. It is worth noting that the values in the connection matrix are permanent for a TSN configuration, while the embeddings of nodes and the global attribute are updated during the learning process described in the following.

GNN model structure. The inputs of the GNN model are the embeddings initialized with the raw features of the topology and traffic, and the output is a prediction about the feasibility of the configuration. The GNN model is depicted in Figure 2(a) as a collection of consecutive *layers*. Each layer in the base model is comprised of a *channel*, which has two neural networks (NNs). The first NN f_v takes the embeddings and the global attribute from the previous layer and updates the embeddings with a learning algorithm. The second NN f_u uses the aggregation (*i.e.* the mean is used as in [3]) of all embeddings (a single vector) as input and produces the global attribute of that layer.

It has to be noted that for each layer, there is only one neural network f_v that processes each embedding independently. The global attribute of the last layer is then taken as input by another neural network f_{pred} that predicts the feasibility of the configuration. The output of this final neural network is the prediction expressed in terms of a probability.



(a) A topology with 3 end-nodes and 2 switches. l_{s1}^{e1} denotes the link from switch S1 to end-node E1.



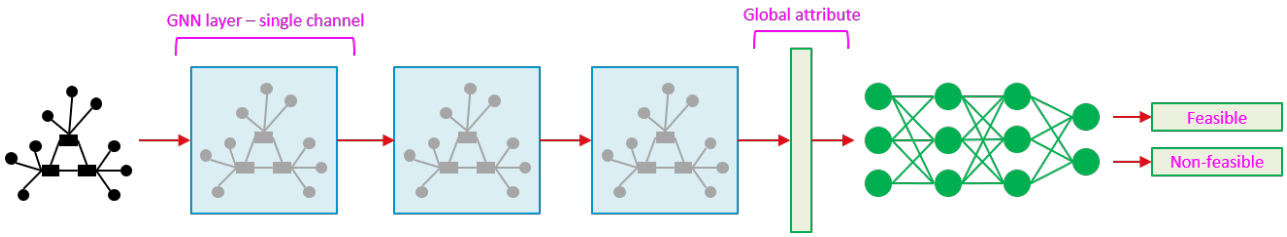
(b) Adding three flows $f1$ and $f3$ (multicast), and $f2$ (unicast). Yellow is used for links, blue for queues and green for flows.

Figure 1: Encoding a TSN configuration as a graph.

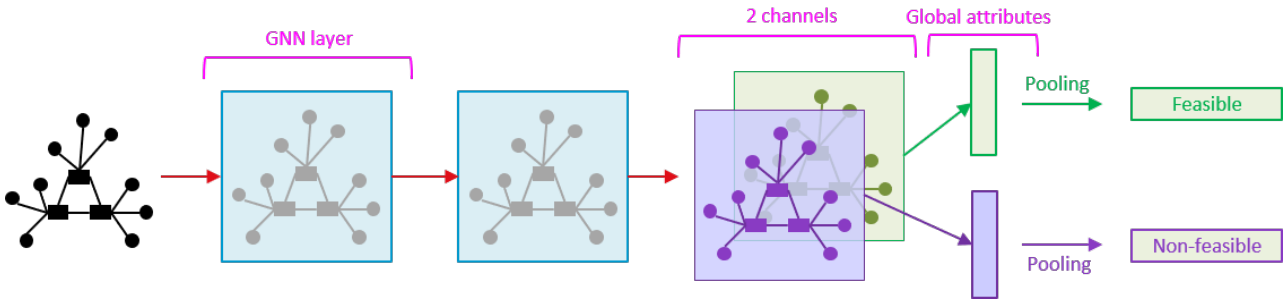
The base GNN model depicted in Figure 2(a) was first proposed in [18]. It features three layers each with a single channel. The four neural networks involved (one for each layer and a final one for the actual feasibility prediction) share the same structure: they are fully-connected feed-forward networks and possess three layers [2].

The learning algorithm. The GNN is trained with a set of TSN configurations with labels, i.e., the feasibility of the configurations. At each layer in the GNN model, the embedding of each node in the graph is updated with the information aggregated (*i.e.*, the sum) from the other connected nodes and the global attribute provided by the previous layer. The neural networks in the GNN model will synthesize features that consider the interferences between flows, the expected delays on links and queues, and the interactions between flows, links and queues. After each mini-batch of 32 training

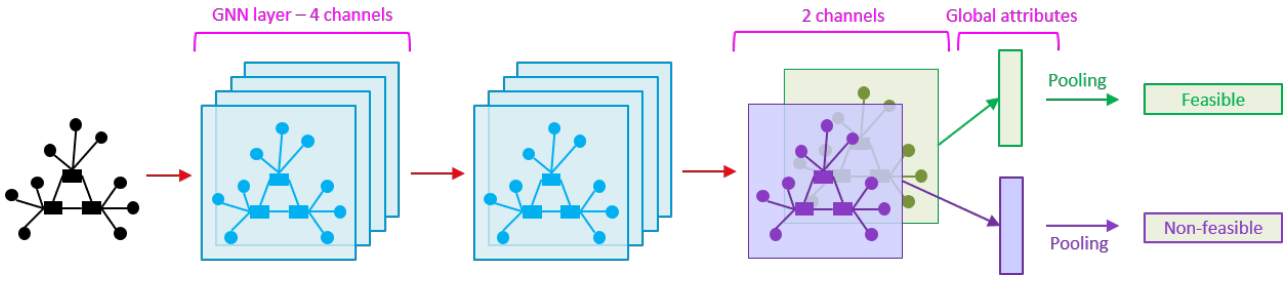
[2]It should be noted that the term *layer* has two distinct meanings depending on the context: layers in the GNN model, which are consecutive blocks of the model, and layers in a neural network.



(a) Base GNN model.



(b) Global pooling.



(c) Multiple channels

Figure 2: Improvement of the base GNN model (a): global pooling layer (b), and multiple channels (c).

samples, all neural networks in the base model are updated to minimize the difference between predictions and true labels. As it has been shown effective to enhance the robustness of the prediction in [18], the final prediction is obtained from an ensemble of 32 GNN models trained individually.

3.2 Training and testing sets

The base training set contains 10000 TSN configurations with random star-shaped topologies, while the traffic is generated with the patterns in Table 1. All training configurations are labeled by the feasibility derived from a network-calculus based schedulability analysis, which is considered as 100% accurate.

There are overall 13 testing sets re-used from [18] for the sake of comparison. Three sets contain configurations with random topologies and 10 sets with specific topologies. The random topologies are star-shaped while the others are either ring, backbone or "dumbell" topologies. The non-random topologies come from studies with automotive OEMs: Renault topology in [25], FACE topology from Renault group as well in [26], Volvo topology in [21] and a early prototype TSN network from Daimler [24]. Finally, some topologies are modified, yet realistic, versions of the OEMs' topologies. It should be noted that all training and testing sets do not contain any overloaded configurations, *i.e.*, configurations that have at least one link whose load is larger than 1. Overloaded configurations have been discarded since they are with certainty non-feasible.

The goal of having multiple test sets is to check whether the GNN model is able to generalize to a wide range of topologies and traffic, *i.e.*, that it is able to predict the feasibility of unseen configurations significantly different from the ones in the training set. Further information about the testing sets can be found in [18].

3.3 Experiment 0: The base GNN model

The first experimental setup considered, Experiment 0, is very similar to the one in [18]. The GNN model has 3 layers with single-channel and the length of the embedding vector is 16, as it was shown to best value during the development of the model [18]. The identity neuron activation function is used. We introduce two changes compared to [18]. First, the maximum number of nodes in

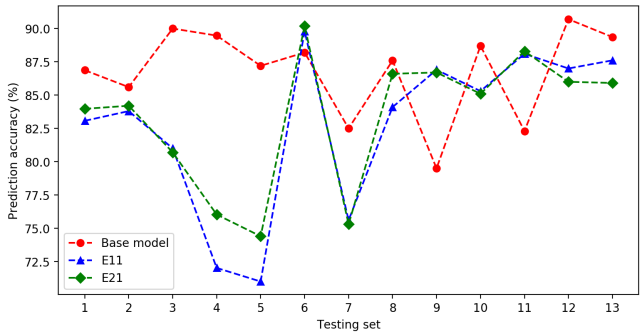


Figure 3: Experiment 1 (E11) and Experiment 2 (E21) use two different training sets that are not star topologies, unlike the training set of the base model. This leads to an improvement of the performance in some testing sets but is detrimental in star topologies (*i.e.* test sets #1 to #5).

the graph is reduced from 500 to 350 to decrease the training time and allow for an efficient parallelisation given memory constraints on GPUs. All in all, less than <1% of the training samples with more than 350 nodes are discarded. Second, we train the models over 20 epochs and all training samples are used at each epoch unlike in [18] where random sampling is used.

4 IMPROVING THE TRAINING SET AND THE GNN MODEL

In this section, we try to enhance the overall performance with incremental changes in the training set and the model. These new experiments are motivated by insights gained into the training data and recent advances in the field of deep-learning.

4.1 Experiments 1 and 2: Improving the training set

The experiments in [18] show that the GNN model has a low prediction accuracy on several testing sets with non star-shaped network topologies presumably because all samples in the training set were star-shaped. Therefore, to introduce more diversity, this experiment denoted as experiment 1, or E11 for short (meaning experiment 1 - step 1), introduces two changes in the training set: the topologies are totally random in terms of shapes, and the packet payload sizes and deadline constraints are chosen at random in a broad range (see Table 1).

We also observe that the wrong predictions of the GNN model mostly happen on a certain range of traffic intensity. Typically, when the load is such that above 40% of the configurations are overloaded (*i.e.*, the load of one link is above 1), the configurations are almost all non-feasible, which the GNN model is able to predict correctly. On the other hand, when no randomly generated configurations are overloaded, the problem is easier as well for the GNN as, by far, most configurations will be feasible. Therefore, we introduce in Experiment 21 (E21) a *focused training set* that is based on the random data set in E11 but focuses on the region between 0 and 40% of overloaded configurations. Precisely, two third of the training samples are taken at random in this interval.

The GNN model in E11 and E21 has the same hyper-parameter setup as the base GNN model. The results are shown in Figure 3. In the figure, the red curve with dots is the base model, the blue curve with triangles is the result of E11, and the green line with diamonds is the result of E21. Compared to the base model with star-based topology, the training set with random topology (E11) improves the prediction accuracy in some testing sets that are not star-shaped (#6, #9 and #11). With the focused training set (E21), the prediction accuracy is better than with the random data set (E11) in general. However, the focused training set lower the performance of the GNN model in many testing sets made up of star-shaped networks.

Since the star and focused training set have advantages and drawbacks, we use both in the next experiments.

4.2 Experiments 3, 4 and 5: Improving the GNN models

Experiment 3: Using a more expressive neuron activation function. Although the identity activation function facilitates the training of the GNN model, it hinders the learning of complex non-linear relationships in the graph. In experiment 3 comprising E31 and E32, we use a more expressive neuron activation function for all the neural networks. In the development of the model, we have tested several non-linear activation functions namely sigmoid, tanh, Exponential Linear Unit (ELU), and Scaled Exponential Linear Unit (SELU). The sigmoid and tanh functions do not allow the training to converge, while the SELU function shows a performance that is similar to the ELU function. In all following experiments, we use the simpler ELU activation functions.

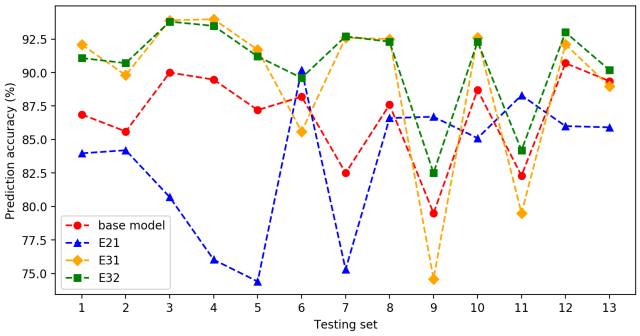


Figure 4: Experiment 3 (E31 and E32) evaluates the influence of using ELU activation function. E31 is the base model but using ELU. E32 is equivalent to E31 but trained with the focused training set. E21 (focused training set, not using ELU) is shown for comparison purpose. E31 and E32 outperform the models without ELU except on test cases #6, #9 and #11.

The experimental results are shown in Figure 4. The red curve with dots is the base model (identity function and star topology training set). The orange curve with diamonds (E31) is the base model that uses ELU activation function. The blue curve with triangles (E21) and green line with squares (E32) show the results of the model trained on the focused training set but respectively using identity and ELU activation functions. Overall, the models using

Table 2: Summary of all experimental conditions. 1^{gl} and 4^{gl} mean global pooling at the last layer and resp. 1 and 4 channels. Exp. and Abbr. resp. mean Experiment and Abbreviation. Imp. and Ext. resp. mean Improved and Expanded model. Abbreviations E31 and E32 correspond to Experiment 3 at step 1 and step 2. This naming convention is also applied to the other experiments.

Exp.	Abbr.	GNN model					Training		
		# nodes	Activation	# layers	# channels	Emb. size	# iters	# samples	Training set
Exp. 0	**Base**	**350**	**identity**	**3**	**1**	**16**	**20**	**10000**	**Star-shaped**
Exp. 1	E11	–	–	–	–	–	–	–	Random
Exp. 2	E21	–	–	–	–	–	–	–	Focused
Exp. 3	E31	–	ELU	–	–	–	–	–	Star-shaped
	E32	–	ELU	–	–	–	–	–	Focused
Exp. 4	E41	–	–	–	1^{gl}	–	–	–	Star-shaped
	E42	–	–	–	1^{gl}	–	–	–	Focused
Exp. 5	E51	–	–	–	4^{gl}	–	–	–	Star-shaped
	E52	–	–	–	4^{gl}	–	–	–	Focused
	Imp.	**350**	**ELU**	**3**	1^{gl}	**16**	**20**	**10000**	**focused**
Exp. 6	E61	–	–	–	–	–	50	–	focused
	E62	–	–	5	–	–	20	–	–
	E63	–	–	3	–	32	–	–	–
	E64	–	–	3	–	32	50	–	–
	Ext.	**350**	**ELU**	**3**	1^{gl}	**32**	**50**	**10000**	**Focused**
Exp. 7	E71	–	–	–	–	16	20	20000	–
	E72	–	–	–	–	32	50	–	–

ELU provide a clear boost in prediction accuracy. Nonetheless, the ELU activation function is detrimental on testing set #9 and #11.

This experiment also shows the benefit of the focused training set as E32 outperforms or matches E31 on all test cases.

Experiment 4: Using neural network with global pooling. The GNN model, shown in Figure 2(a), learns the relationships between the components of the graph and aggregates them into a global attribute. This vector, whose size must be equal to the size of the embeddings, is provided as input to a neural network that predicts the feasibility of the configuration. This raises the question of whether the final neural network is essential or whether it could be replaced by a global pooling layer, a technique commonly used in Convolutional Neural Networks (CNN), as shown in Figure 2(b). Here average pooling is used, which means taking the average of the global attribute vector. There are two global attributes generated by two channels of the third layer, which are two independent sets of neural networks sharing the same input from the previous layer. The global average pooling layer provides two values corresponding to the prediction of feasible and non-feasible probability. It should be noted that the GNN model does not decide which value is feasible and non-feasible probability, it only tries to predict the true label provided in a predefined order [feasible, non-feasible].

The experimental results of using global pooling are shown in Figure 5, with a comparison to Experiment 3. The red curve with dots (E31) and orange curve with diamonds (E41) are the GNN model trained with the star topology, respectively without and with the global pooling layer. The blue curve with triangles (E32) and green curve with squares (E42) are the GNN model trained with the focused training set, while only E42 uses the global pooling layer. The results with the global pooling layer almost overlap the results without using it, which suggests that removing the final neural network and replacing it with the global pooling layer does not affect the performance of the model overall. In the rest of the experiments we will always use global pooling as it reduces the overall complexity of the GNN model's structure by removing the final neural network used for prediction, therefore facilitating the development and maintenance of the model.

Experiment 5: Using multiple channels. In this experiment, we exploit the same approach as in the previous experiments by introducing multiple channels but this time at the two first layers of the GNN model. For each layer, the number of channels is increased from one to four, as illustrated in Figure 2(c). The purpose is to answer the question of whether with multiple channels, the GNN model would be able to explore the graph with more perspectives. As in Experiment 4, each channel in a layer is an independent set of neural networks: one neural network f_v to update embeddings of nodes in the graphs, and one neural network f_u to update the global attribute. All channels in the same layer share the same input provided by the previous layer.

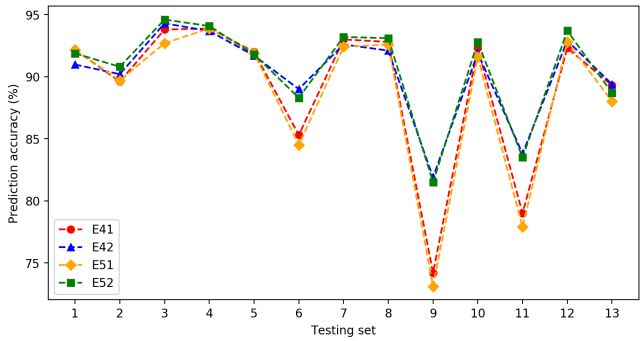


Figure 6: Experiment 5 evaluates the influence of using multiple channels at the first two layers of the GNN model as done for E51 (star topology training set) and E52 (focused training set). E41 and E42 that use a single channel and the same two different training sets are shown for comparison.

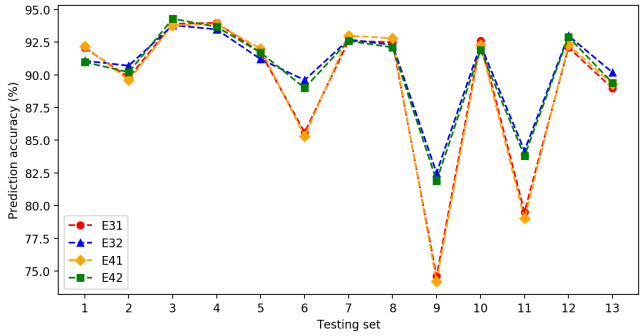


Figure 5: Experiment 4 evaluates the influence of using a global pooling layer, as implemented in E41 trained with star topologies and E42 trained with the focused training set. E31 and E32 from Experiment 3 are shown for comparison purpose. The use of global pooling does not lead to significant changes in accuracy but simplifies the GNN model's structure as the final neural network is suppressed.

The experimental results are shown in Figure 6. The red curve with dots (E41) is the GNN model trained with the star topologies and using a single channel, while the orange curve with diamonds (E51) is identical to E41 but with multiple channels. The blue curve with triangles (E42) and the green curve with squares (E52) are the results with the focused training set, respectively without and with multiple channels. The use of multiple channels does not lead to noticeable improvements, on the contrary a slight loss in accuracy is usually observed, while it requires additional training time. In the following experiments, all GNN models have a single channel at their first two layers.

In summary of this Section, Experiments 3 and 4 have shown the positive effect of the ELU activation function and the global pooling layer on the GNN model, resp. in terms of accuracy and reduced complexity. On the other hand, the approach with multiple channels does not bring benefits as observed in Experiment 5. In all these experiments, the GNN model trained with the focused training set outperforms the model trained with star topologies only. As a result, we define an *improved model* that includes ELU activation, global pooling, and is trained with the focused training set.

5 EXPANDING THE GNN MODEL AND THE TRAINING SET

The improved model, as derived in Section 4 does not increase the size of the GNN model, on the contrary since the final neural network has been suppressed. This section explores the possibility of improving the prediction accuracy with a larger GNN model and a larger training set.

5.1 Experiment 6: Expanding the GNN model

Based on the improved model, Experiment 6 progressively introduces modifications to the GNN model and the training step:

- E61 increases the number of training epochs from 20 to 50.
- E62 increases the number of layers in the GNN model from 3 to 5.
- E63 expands the size of the embeddings from 16 to 32 to allow the neural network to capture more information.
- E64 combines the expansion in size of the embeddings and an increase in the number of training epochs.

The results are shown in Figure 7. An increase in the number of training iterations (E61) from 20 to 50 yields a positive impact on the overall performance. In contrast, adding more layers in the GNN model (E62) is detrimental to the prediction accuracy[3]. An increase of the embedding size from 16 to 32 (E63) also enhances the generalization ability of the model. A combination of more training iterations and larger embeddings (E64), *i.e.* what we will call the *expanded model* in this following, brings the most benefits.

5.2 Experiment 7: Expanding the training set

We now explore the possibility to improve the performance by enlarging the training set. The number of training samples in the focused training set is increased from 10000 to 20000. The *expanded training set* is used to train the improved GNN model and the expanded GNN model separately. The questions we want to answer is whether 1) the GNN model is able to take advantage of a larger training set and, if yes, 2) whether the improvements to the GNN model introduced in the previous section are actually needed with the larger training set.

The results of the experiment are shown in Figure 8. In E71, the improved model trained over the expanded training set (the orange line with diamonds) yields an improvement in *generalization* ability: indeed, there is no testing set that has a prediction accuracy lower than 84.4%. This result is consistent with many other studies in the field of machine learning, e.g. [10], that show the benefits of larger training sets.

In E72, the expanded GNN model trained with 20000 samples has a high prediction accuracy in all testing sets except testing set #13 with Volvo topology. On the one hand, a combination of both

[3]The reason is that additional layers make a phenomenon called *gradient vanishing* more severe here and training the GNN model becomes more difficult (it may take much longer or even fail). A popular approach to mitigate this problem is the residual network, which is out of the scope of this study.

Table 3: Summary of the experimental results obtained with the different GNN models in terms of prediction accuracy expressed in %. Baseline is the feasibility ratio of the samples in the testing sets as derived by schedulability analysis.

Testing sets	Baseline	Base GNN model	Improved GNN model	Expanded GNN model	Expansion of training set	
					Improved model	Expanded model
Testing set 1	57.9	**86.9**	91.0	92.3	91.4	92.6
Testing set 2	53.9	**85.6**	90.2	91.5	90.7	90.0
Testing set 3	47.5	**90.0**	94.3	95.4	93.3	94.6
Testing set 4	62.2	**89.5**	93.7	94.5	93.0	94.4
Testing set 5	73.3	**87.2**	91.7	91.0	90.5	89.0
Testing set 6	36.0	**88.2**	89.0	88.7	90.5	91.8
Testing set 7	51.4	**82.5**	92.6	93.6	93.4	94.5
Testing set 8	41.6	**87.6**	92.1	95.3	93.3	94.0
Testing set 9	22.7	**79.5**	81.9	81.1	84.4	87.1
Testing set 10	39.1	**88.7**	91.9	92.7	91.7	92.0
Testing set 11	22.0	**82.3**	83.8	83.9	86.0	88.1
Testing set 12	48.0	**90.7**	92.9	93.9	92.7	93.7
Testing set 13	57.8	**89.4**	89.4	88.9	91.2	82.2

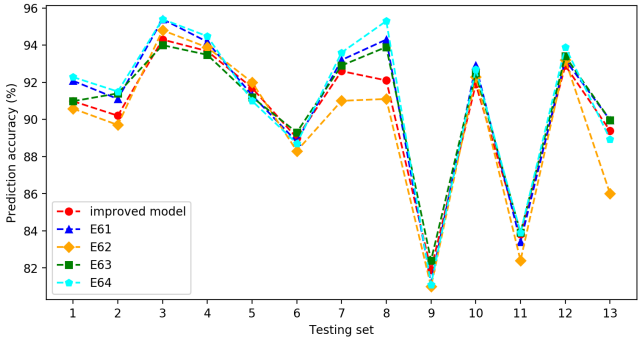


Figure 7: Experiment 6 evaluates the influence of scaling up the size of the GNN model: increasing the number of training epochs from 20 to 50 (E61), increasing the number of layers in the GNN model from 3 to 5, E63 (E62), expanding the size of the embeddings from 16 to 32 (E63) and combining the expansion in the embeddings' size and the increase in the number of training epochs (E64).

a larger model and training set improves the generalization of the expanded GNN model, as 12 testing sets out of 13 have a prediction accuracy higher than 87.1%. On the other hand, the larger model can learn more from a larger training set, which is a drawback to testing set #13 as it has a very distinctive traffic pattern, namely AVB-shaped streams, that does not appear in any sample of the training set. Actually, the GNN model should be extended to capture the characteristics of the streams in terms of whether they are AVB streams or not. It is worth noting that except for the special case of testing set #13, the expanded GNN model trained with more training samples (*i.e.*, E72) is better than trained with less training samples (*i.e.*, expanded model). Although E72 may be slightly less accurate for some testing sets, it is more consistent in the sense that its performances do not drop too low, which is important in practice.

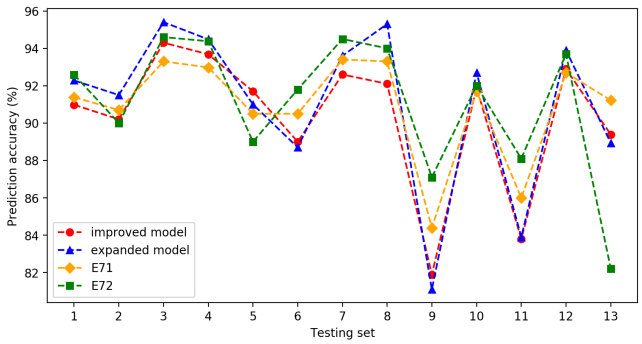
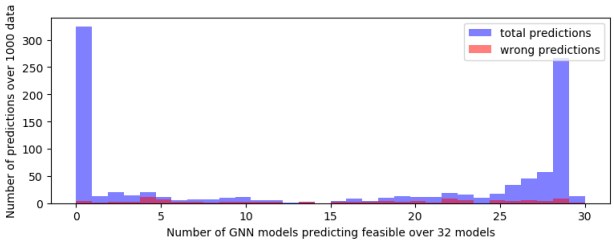


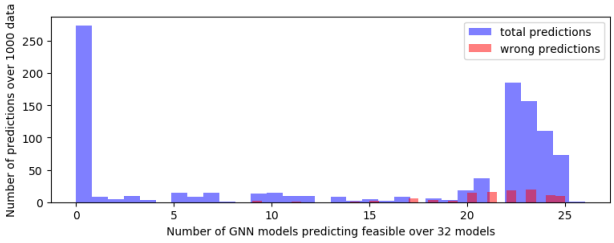
Figure 8: Experiment 7 evaluates the influence of larger training sets made up of 20000 samples instead of 10000, which are used to train the improved GNN model (E71) and the expanded GNN model (E72). A larger training set allows both models to generalize significantly better (e.g., test sets #9 and#11). The special case of test case #13 is discussed in §5.3

5.3 Summary of the experimental results

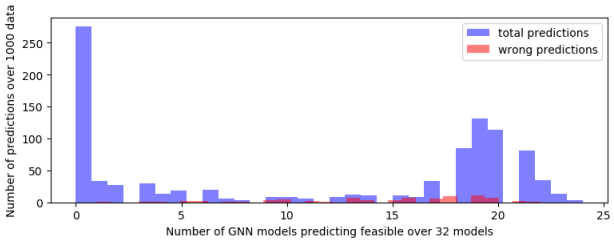
Table 3 presents the prediction accuracy of the GNN models experimented in this study. The baseline is the prediction accuracy of a trivial model that invariably predicts feasible for all testing configurations. While the base GNN model has a prediction accuracy in the range [79.5%, 90.7%], the improved and expanded GNN models range between [84.4%, 93.3%] and [82.2%, 94.6%] respectively. The improvement of the GNN models by comparison with the base model is summarized in Figure 9.



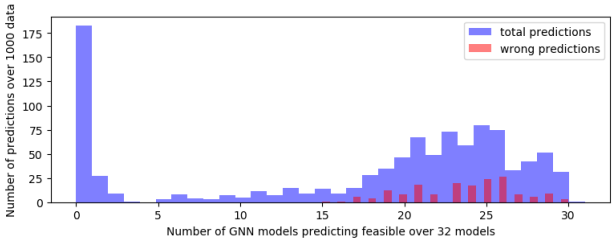
(a) The improved model.



(b) The expanded model.



(c) The improved model with expanded training set.



(d) The expanded model with expanded training set.

Figure 10: Test-case #13 (Volvo topology): Histogram of the number of GNN models predicting feasible (among the 32 models in the ensemble of GNN models) for each of the 1000 test configurations. The bins 0 and 32 correspond to the case where all GNNs in the ensemble return the same prediction.

It is noteworthy that although the GNN models are trained with random topologies in the focused training set, they still improve the prediction accuracy on the testing sets that have star topology, *i.e.*, testing set #1 to #5. There is a steep increase in the prediction accuracy for testing set #7 (Renault topology) with more than 10% improvement in all four models. In general, the prediction accuracy of all testing sets increases considerably with the exception of testing set #13.

The more training data, the more a machine learning model will learn about it and becomes specialized and efficient on data similar to the one is the training set. The consequence is that on unseen data very dissimilar to the training set an expressive model may be outperformed by simpler models, models trained with less data or models trained for a shorter duration. The decrease in the prediction accuracy of testing set #13 suggests to us that this phenomenon may be happening.

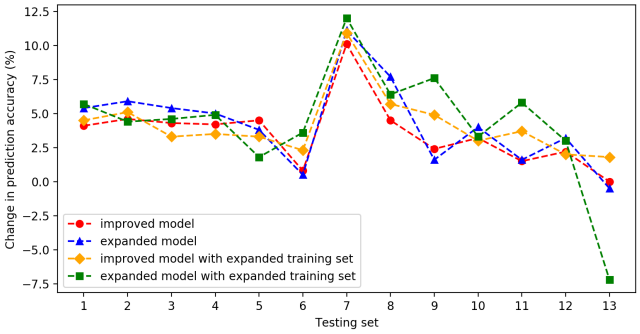


Figure 9: Variations in prediction accuracy (%) of the different GNN models compared to the base model.

We further analyze the predictions for testing set #13 in Figure 9. For each testing configuration, we count the number of GNN models that predict "feasible" among the ensemble of 32 models. Ideally, when a configuration is feasible, the prediction of all 32 models should be "feasible" and the bin 32 is incremented by one. We observe that the larger the model, the larger the training set, the more spread the empirical distributions in Figure 9. This suggests that bigger models necessitate ensemble techniques because there is a high risk that an individual model is wrong. It also suggests that larger number of models in the ensemble could lead to better accuracy, at the expense of execution times. An hybrid approach explored in [16] for conventional ML algorithms is to rely on schedulability analysis when the uncertainty is too high, which could be measured here by the proportion of GNN models returning conflicting predictions.

5.4 Execution times of training and testing

All experiments in this study have been conducted on the High-Performance Computing (HPC) cluster [30] of the University of Luxembourg. The predictions are obtained with ensembles of 32 GNNs, trained in parallel on 32 GPUs (NVIDIA V100).

The training, evaluation and test configurations have been generated with a Java program using the library of the RTaW-Pegase 3.7.5 software [1] on Java JDK-13. The labels (*i.e.*, the feasibility of the configurations) have been derived with the version of the OPA algorithm available in the Pegase library. RTaW-Pegase implements a network-calculus based schedulability, which takes on average 470ms for a 500-stream TSN configuration (see measurements in [22]).

The execution time of the GNN models on a testing set with 1000 TSN configurations is shown in Table 4. The execution of a

Table 4: Testing times of the GNN models by comparison with schedulability analysis. The first row gives the testing times (*i.e.*, absolute values) of 1000 configurations on a single GPU, which is at most 32 seconds. It is important to note that the execution time of schedulability analysis depends on the complexity of the configurations, while a GNN model predicts the feasibility of any configuration in almost constant time. The speedup factor of the GNN models over schedulability analysis is given in the first column of the second row. The first value is the smallest speedup factor, while the second value is the largest speedup factor, as observed over the entire 13 testing sets. The rest of the cells shows the speedup factor with respect to the base GNN model (*i.e.*, value 1.0).

Testing time	Schedula -bility analysis	Base GNN model	Improved GNN model	Expanded GNN model	Expanded training set	
					Improved model	Expanded model
single GPU	–	0m21s	0m24s	0m32s	0m24s	0m32s
Relative time	77 / 1715	**1.0**	1.14	1.52	1.14	1.52

GNN model does not depend on the size of the configurations, as the maximum number of nodes in the graph is always assumed [4]. The execution time needed to encode TSN configurations as graphs is ignored since it is in the milliseconds order and since this step can be executed in parallel.

The base GNN model performs the best in terms of testing time with 0m21s for 1000 samples, but the more sophisticated GNN models do not lead to a considerable increase in execution time in absolute value (less than 11 seconds). An expansion of the GNN model has a modest effect on the execution time while the improvement on the prediction accuracy is significant as discussed in §5.2 and §5.1. Overall the speedup factor of using GNN models over conventional schedulability analysis ranges from 50 to 1715.

6 RELATED WORKS

Machine learning for networking. Machine learning techniques [10, 11] have already been used in wide range of applications including networking [31]. For instance, ML is used in [32] to create application-specific variants of the TCP protocol, and in [27] to improve the prediction of the TCP round-trip time. Deep learning is utilized for dynamic packet routing in [19] and evolutionary algorithms are used in [15] to balance workload on edge servers.

Recently, GNN has been shown to be a powerful tool to encode graph-based relationships in networks. Important contributions in that line of work have been done by Geyer et al., which for instance use GNN to predict the throughput of TCP flows [7], and, in [9], to create routing protocols used in network management. The GNN model is also applied to Software-Defined Networking [29] and to predict the interference between flows in Network Calculus analysis [8].

ML for Ethernet TSN network design. Conventional ML algorithms have been applied to predict whether a TSN configuration is feasible [16, 17, 22]. Although these studies show positive results for four distinct QoS mechanisms, they have an important limitation: prediction in only efficient on the specific topologies used for the training. These conventional ML algorithms do not generalize well beyond the training set.

A recent work [18] uses GNN to encode TSN networks as graphs. This GNN model, which is referred to as the base model in this paper, can effectively predict the feasibility of a wide range of network topologies that are not part of the training set. An ensemble of 32 GNN models has been shown effective at reducing the risk of wrong predictions. This paper is a follow-up work of [18] that explores the possibility of improving the prediction accuracy with better training sets and more sophisticated GNN models.

[4] As classically done in neural networks, zero-padding is used for empty nodes.

7 CONCLUSION AND FUTURE WORK

This study is a contribution towards making the graph neural network model more practical for the verification of Ethernet TSN networks by improving its prediction accuracy. A practical advantage of deep learning is that the feature engineering step is automated and no domain expertise is required. In that regard, the same model could potentially be efficient in other areas of real-time computing.

In this paper we first introduce a new training set that goes beyond star topologies with more diverse traffic patterns. Over successive experiments, more advanced techniques have been introduced: more powerful activation functions, multiple channels, global pooling and increased embeddings' size. A combination of some of these techniques proves to have a strong positive effect on the performance of GNN, with an accuracy higher than 90% on 10 out of 13 testing sets, and never less than 80%. The drawback is a limited increase in the execution time of the prediction on unseen configurations, while the structure and learning process of the GNN remain identical. Recent studies [20, 33] suggest that some machine learning techniques such as manual batch normalization and focal loss can help with the calibration of deep learning models. Adapting and integrating these potential improvements in the GNN model for feasibility prediction is our ongoing work.

In our experiments, the use of ensemble GNN models provides a speedup factor ranging from 50 to 1715 compared to schedulability analysis. Such speed-up factors open new possibilities for design space exploration like the development of "near-interactive" design tools that would enable designers to observe in real-time the effects of their choices. In design space exploration the risk of configurations deemed feasible while they are not is avoided, as long as the retained solutions, the ones presented to the designer, are verified by conventional schedulability analysis.

REFERENCES

[1] RealTime at Work. [n.d.]. RTaW-Pegase: Modeling, Simulation and automated Configuration of communication networks. Retrieved 2019/01/24, https://www.realtimeatwork.com/software/rtaw-pegase.

[2] N.C. Audsley. 2001. On priority assignment in fixed priority scheduling. *Inform. Process. Lett.* 79, 1 (2001), 39–44.

[3] P.W. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

[4] O. Creighton, N. Navet, P. Keller, and J. Migge. 2020. Towards Computer-Aided, Iterative TSN-and Ethernet-based E/E Architecture Design. In *2020 IEEE Standards Association (IEEE-SA) Ethernet & IP @ Automotive Technology Day*. Munich. http://hdl.handle.net/10993/44490

[5] R.I. Davis, L. Cucu-Grosjean, M. Bertogna, and A. Burns. 2016. A Review of Priority Assignment in Real-Time Systems. *J. Syst. Archit.* 65, C (April 2016), 64–82. https://doi.org/10.1016/j.sysarc.2016.04.002

[6] R.I. Davis and N. Navet. 2012. Controller area network (CAN) schedulability analysis for messages with arbitrary deadlines in FIFO and work-conserving queues. In *9th IEEE International Workshop on Factory Communication Systems*. 33–42.

[7] F. Geyer. 2017. Performance evaluation of network topologies using graph-based deep learning. In *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*. 20–27.

[8] F. Geyer and S. Bondorf. 2019. DeepTMA: predicting effective contention models for network calculus using graph neural networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 1009–1017.

[9] F. Geyer and G. Carle. 2018. Learning and generating distributed routing protocols using graph-based deep learning. In *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*. 40–45.

[10] I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep learning*. MIT press.

[11] T. Hastie, R. Tibshirani, and J. Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

[12] IEEE. 2009. *IEEE Standard for Local and Metropolitan Area Networks – Virtual Bridged Local Area Networks Amendment 12 Forwarding and Queuing Enhancements for Time-Sensitive Streams* (Std 802.1Qav-2009 ed.).

[13] IEEE. 2016. *IEEE Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic* (IEEE Std 802.1Qbv-2015 ed.).

[14] IEEE. 2016. *IEEE Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks – Amendment 26: Frame Preemption* (IEEE Std 802.1Qbu-2016 ed.).

[15] T.L. Mai, N.N. Dao, and M. Park. 2018. Real-time task assignment approach leveraging reinforcement learning with evolution strategies for long-term latency minimization in fog computing. *Sensors* 18, 9 (2018), 2830.

[16] T.L. Mai, N. Navet, and J. Migge. 2019. A hybrid machine learning and schedulability analysis method for the verification of TSN networks. In *2019 15th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 1–8.

[17] T.L. Mai, N. Navet, and J. Migge. 2019. On the use of supervised machine learning for assessing schedulability: application to Ethernet TSN. In *Proceedings of the 27th International Conference on Real-Time Networks and Systems*. 143–153.

[18] Tieu Long Mai and Nicolas Navet. 2020. *Deep Learning to Predict the Feasibility of Priority-Based Ethernet Network Configurations*. Technical Report. University of Luxembourg. submitted to ACM TECS.

[19] B. Mao, Z.M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani. 2017. Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning. *IEEE Trans. Comput.* 66, 11 (2017), 1946–1960.

[20] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip HS Torr, and Puneet K Dokania. 2020. Calibrating Deep Neural Networks using Focal Loss. *arXiv preprint arXiv:2002.09437* (2020).

[21] N. Navet, H.H. Bengtsson, and J. Migge. 2020. Early-stage Bottleneck Identification and Removal in TSN Networks. In *Automotive Ethernet Congress*. Munich. http://hdl.handle.net/10993/46282

[22] N. Navet, T.L. Mai, and J. Migge. 2019. *Using machine learning to speed up the design space exploration of Ethernet TSN networks*. Technical Report. University of Luxembourg. http://hdl.handle.net/10993/38604

[23] N. Navet, J. Migge, J. Villanueva, and M. Boyer. 2018. Pre-shaping Bursty Transmissions under IEEE802.1Q as a Simple and Efficient QoS Mechanism. *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 11 (04 2018). https://doi.org/10.4271/2018-01-0756

[24] N. Navet, J. Seyler, and J. Migge. 2015. Timing verification of real-time automotive Ethernet networks: what can we expect from simulation?. In *the SAE World Congress 2015, "Safety-Critical Systems" Session*. Detroit, USA.

[25] N. Navet, J. Villanueva, and J. Migge. 2018. Automating QoS protocols selection and configuration for automotive Ethernet networks. In *SAE World Congress Experience (WCX018), session "Vehicle Networks and Communication (Part 2 of 2)"*. Detroit, USA.

[26] N. Navet, J. Villanueva, and J. Migge. 2019. Early-stage topological and technological choices for TSN-based communication architectures. In *2019 IEEE Standards Association (IEEE-SA) Ethernet & IP @ Automotive Technology Day*. Detroit, Mi. http://hdl.handle.net/10993/40623

[27] B.A.A. Nunes, K. Veenstra, W. Ballenthin, S. Lukin, and K. Obraczka. 2014. A machine learning framework for TCP round-trip time estimation. *EURASIP Journal on Wireless Communications and Networking* 2014, 1 (2014), 47.

[28] Plexxi. 2016. Latency in Ethernet Switches. Retrieved 2019/04/25, http://www.plexxi.com/wp-content/uploads/2016/01/Latency-in-Ethernet-Switches.pdf.

[29] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio. 2019. Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN. In *Proceedings of the 2019 ACM Symposium on SDN Research*. 140–151.

[30] Sébastien Varrette, Pascal Bouvry, Hyacinthe Cartiaux, and Fotis Georgatos. 2014. Management of an academic HPC cluster: The UL experience. In *2014 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 959–967.

[31] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang. 2018. Machine Learning for Networking: Workflow, Advances and Opportunities. *IEEE Network* 32, 2 (March 2018), 92–99. https://doi.org/10.1109/MNET.2017.1700200

[32] K. Winstein and H. Balakrishnan. 2013. TCP Ex Machina: Computer-generated congestion control. *ACM SIGCOMM Computer Communication Review* 43, 4, 123–134.

[33] Marvin Zhang, Henrik Marklund, Abhishek Gupta, Sergey Levine, and Chelsea Finn. 2020. Adaptive Risk Minimization: A Meta-Learning Approach for Tackling Group Shift. *arXiv preprint arXiv:2007.02931* (2020).