A Hybrid Machine Learning and Schedulability Analysis Method for the Verification of TSN Networks

Tieu Long Mai
University of Luxembourg
Luxembourg
long.mai@uni.lu

Nicolas Navet
University of Luxembourg
Luxembourg
nicolas.navet@uni.lu

Jörn Migge
RealTime-at-Work (RTaW)
France
jorn.migge@realtimeatwork.com

***Abstract*—Machine learning (ML), and supervised learning in particular, can be used to learn what makes it hard for a network to be feasible and try to predict whether a network configuration will be feasible without executing a conventional schedulability analysis. A disadvantage of ML-based timing verification with respect to schedulability analysis is the possibility of "false positives": configurations deemed feasible while they are not. In this work, in order to minimize the rate of false positives, we propose the use of a measure of the uncertainty of the prediction to drop it when the uncertainty is too high, and rely instead on schedulability analysis. In this hybrid verification strategy, the clear-cut decisions are taken by ML, while the more difficult ones are taken by a conventional schedulability analysis. Importantly, the trade-off achieved between prediction accuracy and computing time can be controlled. We apply this hybrid verification method to Ethernet TSN networks and obtain, for instance in the case of priority scheduling with 8 traffic classes, a 99% prediction accuracy with a speedup factor of 5.7 with respect to conventional schedulability analysis and a reduction of 46% of the false positives compared to ML alone.**

***Index Terms*—Timing verification, machine learning, supervised learning, unsupervised learning, schedulability analysis, real-time systems, Time-Sensitive Networking (TSN), design-space exploration.**

I. Introduction

Context: Ethernet TSN (Time-Sensitive Networking [1], see [2] for a survey) is becoming the prominent layer 2 protocol for high-speed communication in real-time systems. TSN includes several Quality of Service (QoS) protocols relying on priorities, traffic shaping (e.g., Credit-Based Shaper in IEEE802.1Qav), time-triggered transmission (Time-Aware Shaper in IEEE802.1Qbv) and frame preemption (IEEE802.1Qbu). If these mechanisms open up a lot of possibilities for the designer to meet diverse and demanding communication requirements, they increase exponentially the size of the design space, as, to the best of our knowledge, no optimal protocol selection and configuration strategy exist in the general case.

Verification of timing constraints in Design-Space Exploration: The complexity of designing and configuring TSN networks calls for Design-Space Exploration (DSE) algorithms to assist in the selection and configuration of TSN protocols. The ZeroConfig-TSN (ZCT) algorithm, presented in [3], [4] and implemented in the RTaW-Pegase commercial tool [5], is a first step in that direction. Such DSE algorithms typically consists of three stages executed iteratively: creating candidate solutions, configuring them and evaluating their performance by schedulability analysis and simulation. The performance evaluation step is a limiting factor with respect to the search space that can be explored. For instance, measurements in [6] show that assessing the feasibility of a 500-flow TSN network, with the simple topology of Figure 1 and static priority scheduling with three priority classes, requires an average 470ms computation time (Intel I7-8700 3.2Ghz) per configuration, resulting in about 131 hours for 10^6 candidate solutions. Machine learning algorithms offers a faster alternative to conventional schedulability analysis (e.g., with a speedup factor up to 200 for supervised learning and 1440 for unsupervised learning in [6]). However the percentage of wrong predictions of ML techniques (up to 7% for supervised learning and 11% for unsupervised learning in [6]) is an hindrance to their application.

Contribution of the paper: this work is an extension of [6] which explored the use of supervised and unsupervised ML to predict the feasibility of TSN networks. In this study, we improve upon [6] by proposing an hybrid method combining both ML and schedulability analysis. Importantly, this approach offers user-adjustable trade-offs between accuracy and computation time. The baseline hybrid method is extended with the use, before possibly conducting precise schedulability analysis, of a fast but approximate schedulability analysis. This fast analysis serves to identify feasible network configurations in a compute-efficient manner.

Existing works: ML techniques have been already extensively applied to many areas such as natural language processing, autonomous driving and software engineering. ML has also been applied to networking, especially networking for the Internet, to solve problems like intrusion detection, on-line decision making (e.g., dynamic routing), protocol design, traffic and performance prediction, etc. For instance, in [7] an expert framework algorithm predicts at run-time the round-trip time (RTT) of TCP connections, decreasing the difference between the estimated RTT and the true RTT. Deep Belief Networks are used in [8] to compute the packet routes dynamically instead of using conventional solutions based on OSPF (Open Shortest Path First). An impressive application of ML is [9] where synthesis-by-simulation is implemented to generate better congestion control protocols for TCP, some comprising more than 150 control rules. The reader is referred to [6], [10] for a more thorough overview of the state-of-the art in ML applied to networking and real-time systems.

Organisation: The remainder of the paper is organised as follows. In Section II, the TSN network model is introduced. Section III highlights situations in which ML is likely to lead to erroneous decisions. Section IV presents the hybrid verification strategy and experimental results. This method is refined in Section IV with the use of approximate schedulability analysis as a pre-test. Finally, Section VI concludes with a summary of the results and perspectives.

II. System Model

We consider switched Ethernet networks supporting unicast and multicast communications. The term *traffic flow* or *traffic stream* refers to data sent to the receiver of an unicast connection or sent to a certain receiver in a multicast connection. We assume that the routing of the packets is static and that the maximum size of the successive frames belonging to a stream is known.

A. Topology and traffic

In this work, we consider that the network topology (layout, link data rates, etc) has been set as the well as the TSN protocols that the network devices support. This is realistic for industrial domains, like the automotive and aeronautical domain, where most design choices pertaining to the topology of the networks and the technologies are made early in the design cycle at a time when the communication needs are not entirely known.

The topology considered in this this study, intentionally chosen simple, is the same as in [11] and similar in terms of structure to the prototype Ethernet network developed by an automotive OEM a few years ago [12]. As shown in Figure 1, the network comprises two switches and eight nodes. The data transmission rate is 100Mbps on all links except 1Gbps on the inter-switch link to avoid the severe bottleneck that can occur with such dumbbell topology. The packet switching delay is assumed to be 1.3us at most in the experiments.

The traffic is made up of three classes whose characteristics are summarized in Table I. Characteristics of the streams and their proportion are the same as in [6], [11] and inspired from a case-study provided by an automotive OEM in [13], [14]. In the experiments, the number of streams varies but the proportion of each stream (indicated in Table I) is a fixed parameter of the stream generation procedure chosen as in [11]. Each stream is either unicast or multicast with a probability 0.5. The number of receivers for a multicast stream is chosen at random between two and five. Sender and receiver(s) of a stream are chosen at random.

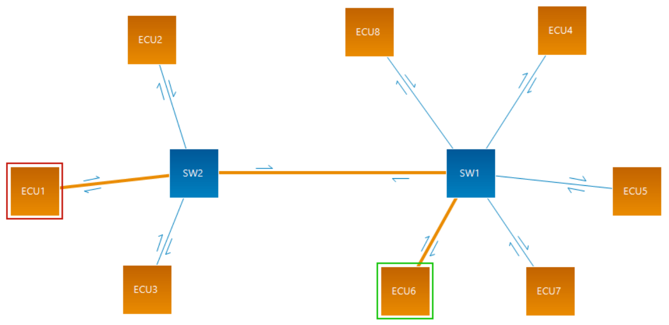

Fig. 1. Topology of the network used in the experiments. The unicast stream shown here goes from ECU1 to ECU6.

TABLE I
Characteristics of the three types of traffic. The performance requirement is to meet deadline constraints. The frame sizes indicated are data payload only.

Audio Streams	• 128 or 256 byte frames • periods: 1.25ms deadline • deadline constraints either 5 or 10ms • proportion: 7/46
Video Streams	• ADAS + Vision streams • 30*1500byte frame each 33ms (30FPS camera for vision) • 15*1000byte frame each 33ms (30FPS camera for ADAS) • 10ms (ADAS) or 30ms (Vision) deadlines • proportion: 7/46
Command & Control	• from 53 to 300 byte frames • periods from 5 to 80ms • deadlines equal to periods • proportion: 32/46

B. Scheduling solutions

In the following, a *configuration* refers to a TSN network that has been fully configured: streams have been allocated to the traffic classes whose relative priorities have been set, scheduling mechanism (e.g., CBS at priority levels 1 and 2) and its configuration parameters (e.g., values of the Idle Slope on each egress port) have been chosen for all traffic classes. A *feasible configuration* is a TSN configuration that meets all the applications constraints. In this study, we consider deadline constraints : the Worst-Case Traversal Time (WCTT) of each stream, computed by schedulability analysis, must be less than the stream's deadline. The scheduling mechanisms considered include the two main TSN QoS strategies, static priority scheduling and traffic shaping, with different trade-offs between complexity and ability to meet timing constraints:

1) FIFO scheduling (*FIFO*): all streams belong to the same traffic class and thus possess the same priority level.
2) Priority scheduling with manual classification (*Manual*): the streams are grouped into the three classes in Table I with priorities set based on the criticality of the streams: Command & Control above audio above video class.
3) Priority scheduling with eight priority levels (*CP8*): priorities are set by the "Concise Priorities" assignment algorithm in RTaW-Pegase, which, in this setup, imple-

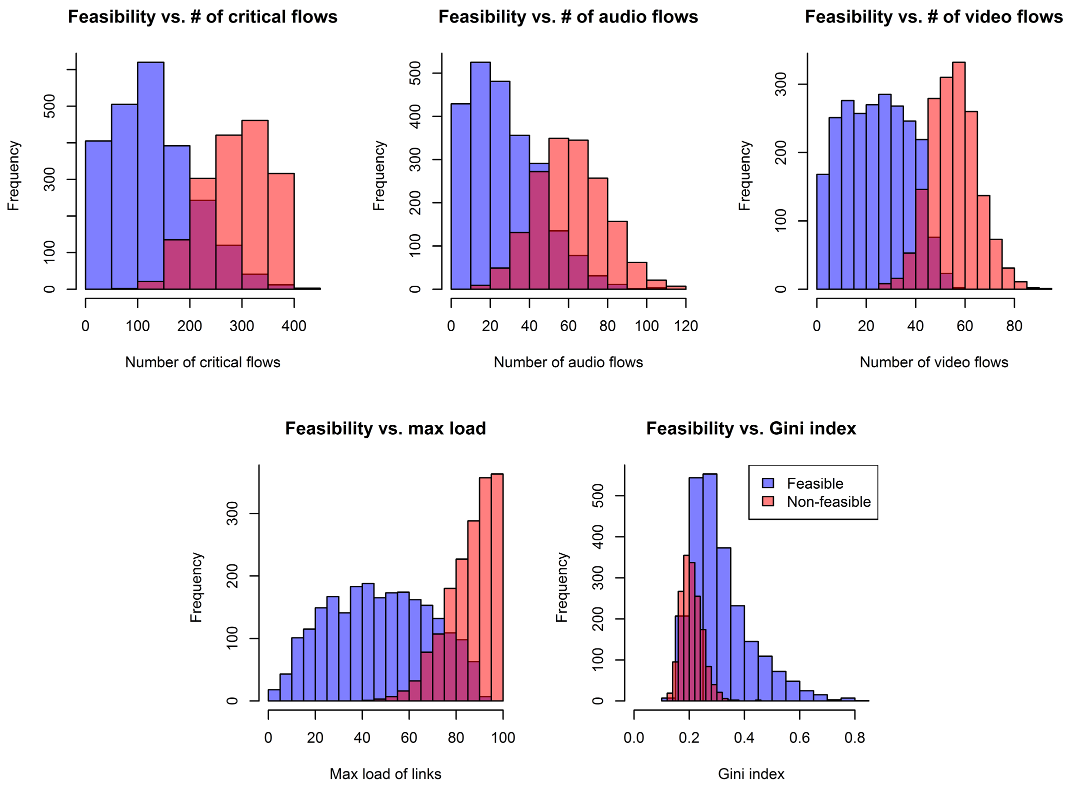

Fig. 2. Frequency histograms of feasible and non-feasible configurations in the training set (4000 configurations) with respect to features' values. The histograms show that the features are able to cluster feasible and non-feasible configurations. However, the purple areas overlapping the two clusters suggest that these features are not predictive enough to avoid any incorrect prediction.

ments the Optimal Priority Assignment algorithm [15] shown optimal for the transmission of periodic/sporadic streams in switched Ethernet network [16].

4) Manual classification with pre-shaping (*Preshaping*): we re-use the manual classification with three priority levels but apply a traffic shaping strategy called pre-shaping in transmission" to all video-streams. Traffic shaping is achieved by inserting idle times, pauses, between the times at which the successive frames of a segmented message (e.g. camera frame) are enqueued for transmission. In the case-study in [17], this strategy has been shown to perform as well as CBS without the need for dedicated hardware and dedicated schedulability analysis (*i.e.* it is compatible with the static priority scheduling analysis). The principles of the algorithm to set the idle times are described in [17].

Two schedulability analyses in Network-Calculus (NC), based on results published in [18]–[20], are used in the experiments:

- The "approximate analysis" which computes WCTTs in linear time with respect to the number of streams.
- The "precise analysis": WCTT computations execute in a time that depends on the least common multiple (LCM) of the frame periods, which can lead to an exponential computation time if periods are coprime.

The reader can refer to [6] for typical computation times and information about the class of (min,+) functions used in each analysis. Importantly, the lower bounds proposed in [21], [22] and used in [23] suggest that the existing NC schedulability analyses for static priority scheduling are precise in terms of the distance between the computed upper bounds and the true worst-case latencies.

III. Insights into when machine learning fails

In this section, we apply the ML technique introduced in [6] and try to gain a better understanding of when ML fails. Precisely, to predict the feasibility of TSN configurations, we apply k-Nearest Neighbors (k-NN), a simple though powerful supervised learning algorithm introduced in standard ML textbooks like [24]. k-NN predicts the label (feasible/non-feasible here) of unseen data points (TSN configuration here) based on the label of the majority of their k nearest neighbors in the feature space.

The performance of k-NN algorithm is evaluated through 1) the percentage of correct predictions (*i.e.*, the overall accuracy), 2) the true positive rate (TPR) (*i.e.* the percentage of correct feasible predictions), and 3) the true negative rate (TNR) (*i.e.* the percentage of correct non-feasible predictions). In our experiments, the training set is made up of 4000 labelled configurations while the testing set comprises 1000 unlabelled

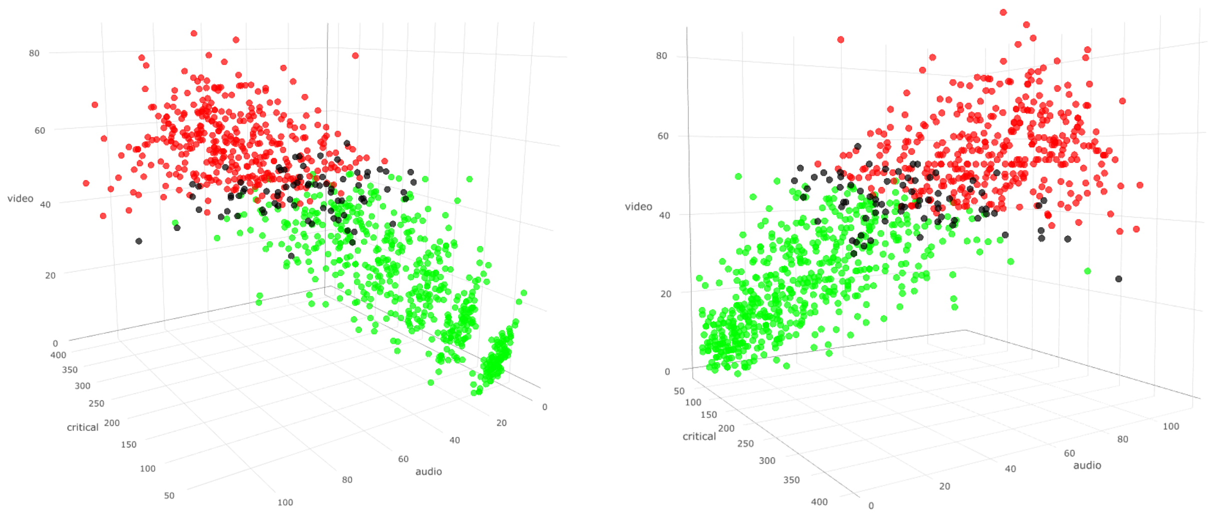

Fig. 3. 3D plots of k-NN feasibility predictions for Preshaping scheduling for the 1000 configurations of the testing set. The green, red and black points represent true positive, true negative and wrong predictions, respectively. Wrong predictions mostly happen when the configuration lies between feasible and non-feasible clusters.

configurations. Previous work [6] shows that k-NN has a good prediction accuracy (from 92% to 96%, see Section IV). However, unlike schedulability analysis, it may lead to ”false positives” (*i.e.*, configurations deemed feasible while they are not). In this section, we try to shed light on when k-NN produces incorrect predictions.

A. Predictive power of the selected features

Features in an ML algorithm are meant to capture the domain-specific knowledge that is needed for ML to ”learn” from the data. Here, we are not concerned with selecting, or ”engineering” better features but aim to understand when ML cannot be trusted. We re-use the five features selected in [6]: number of critical flows, audio flows, video flows, maximum load over all links, balance the link loads (as measured by the Gini index). The ability that a given feature has to predict feasibility can be estimated with the frequency histogram of feasible and non-feasible configurations with respect to the feature values. Figure 2 shows the predictive power of the five features when Preshaping scheduling is used. Similar patterns are observed for the other scheduling mechanisms. Table II provides further insights into the predictive ability of the features with the Pearson coefficients of correlation between the features values and feasibility.

From Figure 2 and Table II the most predictive features are the maximum load, the number of critical flows and video flows. The latter can be explained since video flows require a large bandwidth and may have tight deadlines (up to 10ms for the last of the 15 packets making up a camera image used in an ADAS). In a similar manner, some command and control frames have deadlines as low as 5ms. Intuitively, a maximum load that is high will make it harder to find feasible scheduling parameters for the flows routed on the most loaded link(s). The histogram for the Gini index of the link loads (last histogram in Figure 2) shows that non-feasible configurations tend to be better balanced (*i.e.* lower value) than feasible configurations. Since the source and destination of flows are chosen at random, the load of the links in a configuration is more evenly balanced if there are many flows (the difference to the mean decreases exponentially fast with the number of flows because of the large deviation principle, see [25] for an application).

TABLE II
PEARSON COEFFICIENT OF CORRELATION BETWEEN FEATURES VALUES AND FEASIBILITY OF TSN NETWORKS. THE LABEL OF A NETWORK IS SET TO 0 IF FEASIBLE OR 1 IF NON-FEASIBLE. THE CORRELATION BETWEEN GINI INDEX AND FEASIBILITY IS NEGATIVE SINCE THE LARGER THE GINI INDEX (*i.e.,* THE LESS BALANCED THE LOAD), THE MORE LIKELY THE NUMBER OF FLOWS TO BE SMALL AND THE NETWORK TO BE FEASIBLE.

Feature	**FIFO**	**Manual**	**CP8**	**Preshaping**
# of critical flows	0.38	0.6	0.73	0.75
# of audio flows	0.35	0.55	0.68	0.69
# of video flow	0.43	0.67	0.79	0.8
Max load	0.5	0.72	0.8	0.73
Gini index	-0.55	-0.67	-0.51	-0.5

In general, the selected features are predictive with respect to feasibility of TSN configurations. However, the histograms in Figure 2 shows that feasibility cannot be determined by the individual features, and there are ”gray areas” in which networks can be feasible or non-feasible.

B. Why ML prediction may fail?

The combined use of all the features enables ML to achieve a better accuracy than using any of the features alone. Still, there are areas in the feature space where both feasible and non-feasible networks can be found. This can be seen in Figure 3 which represents the correct (feasible networks in green and non-feasible networks in red) and incorrect predictions (in black) in the feature space. The three dimensions are the number of flows of the different types. These three

TABLE III

PERFORMANCE OF THE HYBRID VERIFICATION TECHNIQUE WHEN THE THRESHOLD (*i.e.*, PROPORTION OF THE MINORITY IN THE SET OF K NEAREST NEIGHBORS) VARIES FROM 5% TO 40%. THE FIRST ROW SHOWS THE BASELINE RESULTS WITH K-NN ALONE. COLUMN (#) IS THE NUMBER OF CONFIGURATIONS REQUIRING PRECISE ANALYSIS. ACCURACY (ACC), TRUE POSITIVE RATE (TPR) AND TRUE NEGATIVE RATE (TNR) ARE EXPRESSED IN PERCENTAGE. EXPERIMENTS CONDUCTED WITH A TESTING SET OF 1000 TSN CONFIGURATIONS.

	FIFO				**Manual**				**CP8**				**Preshaping**			
Threshold (%)	(#)	Acc	TPR	TNR	(#)	Acc	TPR	TNR	(#)	Acc	TPR	TNR	(#)	Acc	TPR	TNR
–	**0**	**96.4**	**68**	**98.7**	**0**	**95.5**	**84.49**	**98.03**	**0**	**94.7**	**95.39**	**93.89**	**0**	**92.4**	**93.85**	**90.36**
5	120	99.6	94.67	100	208	99.4	96.79	100	319	100	100	100	383	100	100	100
10	100	99.4	93.33	99.89	164	99	94.65	100	258	99.9	100	99.78	297	100	100	100
15	86	99.1	90.67	99.78	136	98.4	92.51	99.75	206	99.5	99.82	99.13	239	99.3	99.49	99.04
20	68	98.6	86.66	99.57	107	97.8	90.37	99.51	165	98.9	99.26	98.47	190	98.8	98.97	98.55
25	47	98.3	82.67	99.57	84	97.3	88.24	99.38	121	98.3	98.89	97.6	151	97.5	97.44	97.59
30	39	98.1	80	99.57	61	96.9	87.17	99.14	87	97.7	98.15	97.16	124	96.8	97.09	96.39
35	33	97.9	78.67	99.46	38	96.6	86.1	99.02	69	97.1	97.05	97.16	91	95.8	96.24	95.18
40	18	97.4	76	99.14	24	96.5	85.56	99.02	53	96.7	96.68	96.72	62	95.3	95.9	94.46

features directly determines the load of the network: in the configurations with the largest number of flows, the ones that are the most likely to be non-feasible, the maximum load will be also high.

We observe that k-NN tends to produce wrong predictions in areas where feasible and non-feasible configurations are mixed. In those ”gray” areas, feasible neighbors do not significantly outnumber non-feasible neighbors, and vice-versa. We could not find any effective voting rule for these areas. We tested Support Vector Machines (SVM, see [24]) and faced similar issues in the determination of the Maximum Margin Hyper-plane (MMH) defining the boundary between feasible and non-feasible configurations.

In the rest of the paper, we propose a method to identify TSN configurations located in areas where k-NN is likely to be wrong and conduct schedulability analysis on those configurations to reduce the risk of incorrect prediction.

IV. HYBRID VERIFICATION STRATEGY COMBINING MACHINE LEARNING AND SCHEDULABILITY ANALYSIS

From the insights in Section III and the experimental results in terms of accuracy, we know that a nearest neighbors algorithm like k-NN cannot be fully trusted for configurations located in the gray areas. We therefore propose an hybrid verification strategy[1] combining k-NN and schedulability analysis that aims to reduce the rate of wrong predictions of ML, while still improving the computation time compared to the use of schedulability analysis alone.

A. Principles

The hybrid method relies on the following simple principles. Given an unseen configuration, if the majority of its neighbors does not ”significantly” outnumber the minority, the unseen data is identified as in a gray area and its feasibility is determined by the precise schedulability analysis. On the other hand, if there is a clear consensus among the neighbors then k-NN is trusted. It is worth noting that although the proposed hybrid method is specified for the k-NN algorithm, it can be adapted to any other ML algorithms with changes in how the gray areas are identified. For instance, the distance of the configuration under test to the MMH would be a natural criterion for SVMs.

[1]The data and the R code used in this study is available as open-source (AGPL V3.0) at https://github.com/crtes-group-unilux/ML4TSN-Schedulability.

A crucial step of the hybrid method is to find a decision criterion to trigger the execution of the schedulability analysis. We use here the proportion of nodes in the minority group of the k neighbors: schedulability analysis is conducted if the proportion is over a certain threshold (referred to as *threshold for analysis*). The performance of the hybrid method with various threshold values is shown in Table III. The value of k used for k-NN is the optimal value for a given scheduling mechanism as determined in [6]: 30, 20, 80 and 20 for FIFO, Manual, CP8 and Preshaping, respectively.

When the threshold for analysis is low, logically there are more configurations requiring schedulability analysis (see columns (#) in Table III), which leads to a higher accuracy. In other words, the threshold value controls the prediction accuracy. Importantly, when the threshold is low enough (*i.e.*, 5% for FIFO and 10% for the other policies), the hybrid method does not return any false positive (FP) prediction. At the threshold of 40%, compared to k-NN, the hybrid method allows a decrease of 33%, 50%, 46% and 43% of the rate of FP respectively for FIFO, Manual, CP8 and Preshaping scheduling. It is also noteworthy that, with CP8 and Preshaping scheduling, the hybrid method achieves 100% accuracy without the need to perform schedulability analysis on all configurations, but on 32% and 38% of them respectively. However, one should bear in mind that when using this hybrid approach one cannot rule out the possibility of misprediction, even if it is less likely than using ML alone (e.g., for Preshaping scheduling the accuracy improvement ranges from 2.9% to 7.6%, see Table III).

Table IV shows estimations, obtained by linear interpolation from the numbers in Table III, of the number of configurations requiring schedulability analysis to obtain a 99% prediction accuracy. These estimations are provided for the k values determined as optimal in [6] and k equals 100. It shows that the more complex and powerful in terms of feasibility the

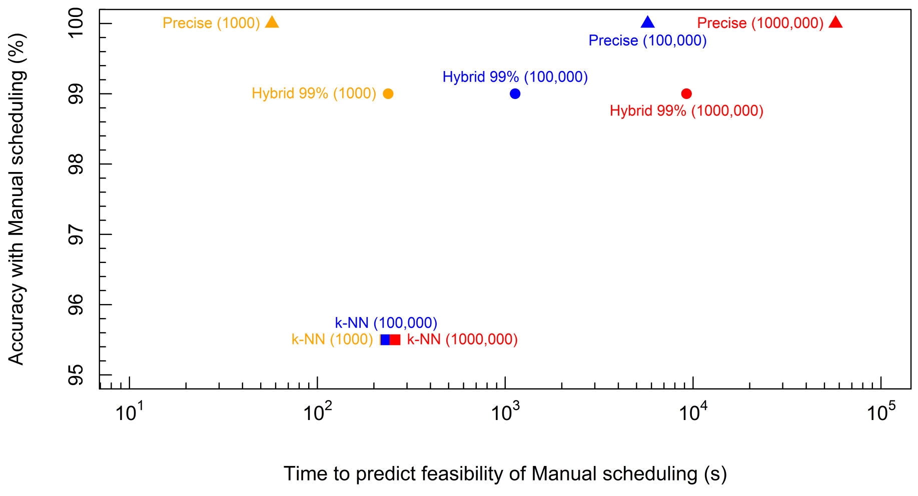

Fig. 4. Accuracy (%) versus running time on 6-core 3.2Ghz Intel 8700 (seconds in log scale) with Manual scheduling for 10^3, 10^5 and 10^6 TSN configurations. Squares, points and triangles identify results of k-NN, hybrid method and Precise analysis, respectively. The threshold for analysis of the hybrid method is set to obtain 99% accuracy. Except for 10^3 configurations where precise analysis dominates the other techniques, each of the methods offers a meaningful trade-off in terms of accuracy and execution times.

TABLE IV
ESTIMATED NUMBER OF CONFIGURATIONS REQUIRING PRECISE ANALYSIS TO OBTAIN 99% PREDICTION ACCURACY, WITH K* THE OPTIMAL K VALUE FOR K-NN FOR EACH SCHEDULING SOLUTION. COLUMN (#) GIVES THE NUMBER OF CONFIGURATIONS REQUIRING PRECISE ANALYSIS OUT OF 1000 CONFIGURATIONS.

	FIFO		**Manual**		**CP8**		**Preshaping**	
k	30*	100	20*	100	80*	100	20*	100
(#)	82	82	164	178	171	176	191	231

scheduling mechanism (see [6]), the more configurations are requiring precise analysis. In addition, it shows that the values of k identified as optimal in [6] still lead to better results than the arbitrary value of 100. In practice, the accuracy that can be achieved for a certain threshold value will depend on how representative the training set is with respect to the unseen data because the hybrid method involves supervised ML. To mitigate this problem, as suggested in [26], one could implement a measure to estimate the distance between the training data and the unseen data, and lower the threshold value to compensate for a lower efficiency of ML. Experiments in [6] show however that feasibility prediction with k-NN is robust to departure from the traffic characteristic assumptions made for the training set.

Table V shows estimates (obtained by linear interpolation from the numbers in Table III) of the prediction accuracy of the hybrid method when exactly 10% of the configurations undergo precise analysis. In practice, the maximum percentage of schedulability analyses that can be conducted depends of the CPU time budget available, from which the threshold for analysis can be derived (e.g., by binary search). Table V shows a decrease in accuracy when the scheduling mechanism becomes more complex. Based on these estimations, using the k values that are optimal for the standard ML algorithm is still preferable, except in the case of Manual scheduling where k equals 100 is a better choice.

TABLE V
ESTIMATED PREDICTION ACCURACY (%) OF THE HYBRID METHOD IF RUNNING PRECISE ANALYSIS IN 10% OF THE CONFIGURATIONS. ACC IS ACCURACY IN %.

	FIFO		**Manual**		**CP8**		**Preshaping**	
k	30*	100	20*	100	80*	100	20*	100
Acc	99.4	99.4	97.65	97.96	97.93	97.92	96.07	95.83

B. New trade-offs between accuracy and computation times

Figure 4 shows for Manual scheduling the trade-offs between accuracy and running times obtained with standard k-NN, the hybrid method and precise analysis (measurements on a 6-core 3.2Ghz Intel 8700 CPU). The other scheduling mechanisms lead to similar patterns in terms of accuracy versus running times. The speedup factors for a 99% accuracy obtained for all scheduling mechanisms are shown in Table VII.

These experimental results lead to the observations summarized below:

- For a small number of configurations (10^3 here), neither ML or the hybrid method are competitive with schedulability analysis as they are both less accurate and slower. The latter is due to the time it takes to create the training set (around 3 hours for all four scheduling mechanisms).

TABLE VI

ACCURACY OF THE MULTI-STAGE HYBRID METHOD COMBINING K-NN, APPROXIMATE AND PRECISE ANALYSIS OBTAINED OVER 1000 TEST CONFIGURATIONS. THE FIRST ROW SHOWS THE RESULTS WITH K-NN ONLY WHILE THE OTHERS ARE FOR ANALYSIS THRESHOLD VALUES FROM 5% TO 40%. THE VALUE OF K FOR K-NN IS 30 FOR FIFO AND 20 FOR MANUAL SCHEDULING. COLUMNS (1) AND (2) ARE THE NUMBER OF CONFIGURATIONS REQUIRING THE EXECUTION OF THE APPROXIMATE AND THE PRECISE ANALYSIS, RESPECTIVELY. ACCURACY (ACC), TRUE POSITIVE RATE (TPR) AND TRUE NEGATIVE RATE (TNR) ARE SHOWN IN PERCENTAGE (%).

	FIFO					**Manual**				
Threshold (%)	(1)	(2)	Acc	TPR	TNR	(1)	(2)	Acc	TPR	TNR
–	**0**	**0**	**96.4**	**68**	**98.7**	**0**	**0**	**95.5**	**84.49**	**98.03**
5	181	137	99.8	97.33	100	208	166	99.4	96.79	100
10	100	64	99.4	93.33	99.89	164	132	99	94.65	100
15	86	53	99.1	90.67	99.78	136	109	98.4	92.51	99.75
20	68	40	98.6	86.67	99.57	107	81	97.8	90.37	99.51
25	57	35	98.4	84	99.57	84	62	97.3	88.24	99.38
30	39	27	98.1	80	99.57	61	44	96.9	87.17	99.14
35	33	23	97.9	78.6	99.46	38	28	96.6	86.1	99.02
40	18	9	97.4	76	99.14	24	18	96.5	85.56	99.02

TABLE VII

ESTIMATES OF THE SPEEDUP FACTORS OF THE HYBRID METHOD COMPARED TO SCHEDULABILITY ANALYSIS FOR ALL SCHEDULING MECHANISMS AND AN INCREASING NUMBER OF CONFIGURATIONS (COLUMN (#)). THE HYBRID METHOD IS CONFIGURED TO ACHIEVE A 99% PREDICTION ACCURACY. ESTIMATES OBTAINED BY LINEAR INTERPOLATION BASED ON THE ACCURACY OBTAINED WITH THRESHOLD VALUES RANGING FROM 5 TO 40% BY STEP 5%.

(#)	**FIFO**	**Manual**	**CP8**	**Preshaping**
10^3	0.24	0.24	0.24	0.24
10^5	8.19	4.9	4.74	4.33
10^6	11.62	5.95	5.71	5.13

- For a medium and large number of configurations (10^5 and 10^6 here), each method is Pareto optimal and offers a distinct and meaningful trade-off between accuracy and running times. A 99% accuracy can be obtained with the hybrid method with a speedup factor ranging from 4.33 to 11.62 over schedulability analysis. ML alone is faster (e.g., speedup of 26 over the analysis for 10^5 configurations with Manual scheduling) but the accuracy lower (*i.e.*, 95.5%).

V. MULTI-STAGE HYBRID APPROACH WITH AN APPROXIMATE ANALYSIS

For the same system, one can develop schedulability analyses with different degrees of pessimism and different execution times. Up to now, all experiments were conducted with a precise analysis that, considering Manual scheduling for instance, takes about 40 times longer than the approximate analysis introduced in Section II (see [6] for comprehensive measurements).

The approximate analysis, because it is more pessimistic, will lead to more false negatives (*i.e.*, configurations deemed unfeasible while they actually are feasible) than the precise analysis but will never lead to any false positives. Based on this observation, we can combine the use of k-NN, approximate and precise analysis to create a hybrid multi-stage verification method as follows:

1) Predict feasibility of the configurations with k-NN.
2) Use the threshold criteria introduced in Section IV to identify the set of configurations in the gray area and run the approximate analysis on them. If the approximate analysis concludes that a configuration is feasible, the configuration is removed from the ”gray” set.
3) Run the precise analysis to determine the feasibility for the rest of the configurations in the ”gray” set.

The approach could be extended to include additional stages if other schedulability analyses are available, or any other means of verification that possesses the property to not create false positives.

The performance of the new method is presented in Table VI. Results are not shown for CP8 and Preshaping scheduling, the latter because we don’t have an approximate analysis for Preshaping. Regarding CP8, the approximate analysis was not able to identify any configuration in the gray area as feasible, and thus did not bring any improvement. This does not invalidate the approach but shows that the power of the approximate analysis (*i.e.*, the rate of false negatives) plays an important role in the overall efficiency. For FIFO and Manual scheduling, Table VI shows a reduction in the number of configurations requiring precise analysis with respect to the baseline hybrid method. Precisely, with a 99% prediction accuracy target, the multi-stage hybrid approach allows to reduce the number of precise analyses by 38% for FIFO and 19% and Manual scheduling.

VI. SUMMARY AND CONCLUSIONS

The proposed hybrid method combining ML and scheduling analysis can decrease the rate of false positive predictions and increase the overall accuracy with respect to the use of ML alone while reducing the computation time by a several-fold factor with respect to a precise schedulability analysis. If a sound schedulability analysis ensures no false positives, they cannot be ruled out with ML predictions. This is why, in a design-space exploration workflow, we think that solutions deemed feasible by ML and retained to be part of the final set of solutions proposed to the designer, must undergo schedulability analysis. Still, schedulability analysis will only be performed for the final set of solutions and not for each of the candidate solutions which has been created during the exploration of the search space.

In this study, we implemented the hybrid method with k-NN as the ML algorithm but the concept of hybrid verification techniques for feasibility analysis can be adapted to any other supervised ML algorithms with changes in the criterion that triggers schedulability analysis. Importantly, the hybrid method neither requires a huge amount of data nor important computing power, it can be integrated into the toolbox of the network designers and run on standard desktop computers.

Increasing the control, and the trust we can have on ML-based techniques is crucial for their adoption in the industry. This is true in particular for design-space exploration and generative design approaches leveraging the new possibilities brought by ML. By further automating the design and configuration, these algorithmic tools will help designers handle the increasing size and complexity of communication architectures, often based on multiple protocols and communication paradigms. Upcoming communication systems, be it in the automotive or industrial domain, will increasingly be subject to dynamically evolving communication requirements that may require network re-configuration at run-time. To support these new use-cases, we need techniques like proposed in this paper that help speed up the timing verification of critical communication systems.

References

[1] IEEE, *IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks*, IEEE Std 802.1Q-2018 ed., 2018.

[2] A. Nasrallah, A. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. E. Bakoury, "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, 2019.

[3] J. Migge, J. Villanueva, N. Navet, and M. Boyer, "Performance assessment of configuration strategies for automotive Ethernet quality-of-service protocols," in *Automotive Ethernet Congress*, Munich, January 2018.

[4] N. Navet, J. Villanueva, and J. Migge, "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, April 2018.

[5] "RTaW-Pegase: Modeling, simulation and automated configuration of communication networks," RealTime-at-Work, retrieved 2019/01/24, https://www.realtimeatwork.com/software/rtaw-pegase.

[6] N. Navet, T. Mai, and J. Migge, "Using machine learning to speed up the design space exploration of Ethernet TSN networks," University of Luxembourg, Tech. Rep., 2019. [Online]. Available: http://hdl.handle.net/10993/38604

[7] N. Arouche, A. Bruno, K. Veenstra, W. Ballenthin, S. Lukin, and K. Obraczka, "A machine learning framework for TCP round-trip time estimation," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, p. 47, Mar 2014. [Online]. Available: https://doi.org/10.1186/1687-1499-2014-47

[8] B. Mao, Z. M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Transactions on Computers*, vol. 66, no. 11, pp. 1946–1960, Nov 2017.

[9] K. Winstein and H. Balakrishnan, "TCP Ex Machina: Computer-generated congestion control," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 123–134.

[10] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, pp. 92–99, March 2018.

[11] N. Navet and J. Migge, "Insights into the performance and configuration of TCP in automotive Ethernet networks," in *2018 IEEE Standards Association (IEEE-SA) Ethernet & IP @ Automotive Technology Day*, London, October 2018.

[12] N. Navet, J. Seyler, and J. Migge, "Timing verification of real-time automotive Ethernet networks: what can we expect from simulation?" in *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, Toulouse, France, Jan. 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01292312

[13] J. Migge, J. Villanueva, N. Navet, and M. Boyer, "Insights on the performance and configuration of AVB and TSN in automotive Ethernet networks," in *Embedded Real-Time Software and Systems (ERTS 2018)*, Toulouse, France, January 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01746132

[14] N. Navet, J. Villanueva, J. Migge, and M. Boyer, "Experimental assessment of QoS protocols for in-car Ethernet networks," in *2017 IEEE Standards Association (IEEE-SA) Ethernet & IP @ Automotive Technology Day*, San-Jose, Ca, October 2017.

[15] N. C. Audsley, "On priority asignment in fixed priority scheduling," *Inf. Process. Lett.*, vol. 79, no. 1, pp. 39–44, May 2001. [Online]. Available: http://dx.doi.org/10.1016/S0020-0190(00)00165-4

[16] T. Hamza, J. Scharbarg, and C. Fraboul, "Priority assignment on an avionics switched Ethernet network (QoS AFDX)," in *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014)*, May 2014, pp. 1–8.

[17] N. Navet, J. Migge, J. Villanueva, and M. Boyer, "Pre-shaping bursty transmissions under IEEE802.1Q as a simple and efficient qosmechanism," *SAE Int. J. Passeng. Cars Electron. Electr. Syst.*, vol. 11, 04 2018.

[18] A. Bouillard and E. Thierry, "An algorithmic toolbox for Network Calculus," *Discrete Event Dynamic Systems*, vol. 18, no. 1, pp. 3–49, Mar 2008. [Online]. Available: https://doi.org/10.1007/s10626-007-0028-x

[19] M. Boyer, J. Migge, and N. Navet, "A simple and efficient class of functions to model arrival curve of packetised flows," in *1st International Workshop on Worst-case Traversal Time, in conj. with the 32nd IEEE Real-Time Systems Symposium (RTSS 2011)*, Nov. 2011.

[20] R. Queck, "Analysis of Ethernet AVB for automotive networks using Network Nalculus," in *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, July 2012, pp. 61–67.

[21] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Applying trajectory approach with static priority queuing for improving the use of available AFDX resources," *Real-Time Systems*, vol. 48, no. 1, pp. 101–133, Jan 2012. [Online]. Available: https://doi.org/10.1007/s11241-011-9142-9

[22] H. Bauer, J. Scharbarg, and C. Fraboul, "Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 521–533, Nov 2010.

[23] M. Boyer, N. Navet, and M. Fumey, "Experimental assessment of timing verification techniques for AFDX," in *6th European Congress on Embedded Real Time Software and Systems*, Toulouse, France, Feb. 2012. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01345472

[24] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, ser. Springer series in statistics. Springer, 2009.

[25] N. Navet, L. Cucu, and R. Schott, "Probabilistic Estimation of Response Times Through Large Deviations," in *Work-in Progress of the 28th IEEE Real-Time Systems Symposium (RTSS'2007 WiP)*, Tucson, United States, Dec. 2007.

[26] I. Juutilainen and J. Roning, "A method for measuring distance from a training data set," *Communications in Statistics - Theory and Methods*, vol. 36, no. 14, pp. 2625–2639, 2007.