Efficient Network Telemetry based on Traffic Awareness

Cesar A. Gomez, *Member, IEEE*, Abdallah Shami, *Senior Member, IEEE*, and Xianbin Wang, *Fellow, IEEE*

Abstract Network Telemetry (NT) is a crucial component in today's networks, as it provides the network managers with important data about the status and behavior of the network elements. NT data are then utilized to get insights and rapidly take actions to improve the network performance or avoid its degradation. Intuitively, the more data are collected, the better for the network managers. However, the gathering and transportation of excessive NT data might produce an adverse effect, leading to a paradox: the data that are supposed to help actually damage the network performance. This is the motivation to introduce a novel NT framework that dynamically adjusts the rate in which the NT data should be transmitted. In this work, we present an NT scheme that is traffic-aware, meaning that the network elements collect and send NT data based on the type of traffic that they forward. The evaluation results of our Machine Learning-based mechanism show that it is possible to reduce by over 75% the network bandwidth overhead that a conventional NT scheme produces.

Index Terms— **Machine Learning, Network Telemetry, network traffic classification, overhead reduction, telemetry control.**

I. Introduction

With the advancement of Software-Defined Networks (SDN) paradigm and the development of its programmable data plane (PDP) technologies, the network telemetry (NT) notion has emerged differing from the traditional network measurement schemes, as it comprises an automated procedure for remotely gathering and processing network data [1]. Moreover, traditional network monitoring technologies usually rely on active probes that are protocol-specific, such as Internet Control Message Protocol (ICMP) and Simple Network Management Protocol (SNMP) packets, or passive methods of measurements, which are based only on observations of undisturbed and unmodified packet streams of interest [2]. That is why NT is deemed as a suitable answer to the challenges that the traditional network measurement technologies face in terms of adequate network visibility with better scalability, accuracy, and coverage, as well as hardware and protocol independencies.

The study of how to get high-quality network measurement data at low cost is important, since NT produces massive data in real network environments. The main goal of any NT scheme is to generate and collect measurement data locally at network nodes, depending on different service requirements, and transmit those data to a centralized controller for enabling an optimal network management. Therefore, an efficient telemetry deployment strategy is needed to compensate for the network performance loss due to the impact of gathering and transmitting the telemetry data themselves. Networks' failures and performance problems can have a variety of causes, which requires different types of information to diagnose. That is why the ideal telemetry scenario contemplates the gathering of all the fine-grained data at a fine time scale. However, this means a high cost in terms of communication overhead. On the other hand, network managers need to get the telemetry information in a timely manner to quickly identify, isolate, and fix performance problems in order to minimize the impact on users and organization's revenue. Yet, it is difficult to measure many flows and packets with constrained resources at the network elements, which focus more on control functions such as packet forwarding. Since NT not only processes all the packets but also stores information about the packets, NT sometimes requires even more resources than the control functions do.

Today's NT practices follow a bottom-up approach, *i.e.* network managers collect data from network elements, aggregate it in a centralized collector, and extract the information they need. This approach poses several problems like having too many data to process. For this reason, a new approach is needed, one that provides network managers with abstractions of the metrics they are interested in [3]. Based on those interests, the granularity of the measurements should be different allowing to minimize the overhead produced by the telemetry data's transmission. In this way, different levels of measurement accuracy can also be obtained considering the network resources' limitations. Nevertheless, the task of matching network managers' desires with specific telemetry

Paper submitted for review on July 26, 2021.

C. A. Gomez is a PhD candidate with the Department of Electrical and Computer Engineering, Western University, 1151 Richmond Street London, Ontario, Canada (e-mail: cgomezsu@uwo.ca).

A. Shami is a professor with the Department of Electrical and Computer Engineering, Western University, 1151 Richmond Street London, Ontario, Canada (e-mail: abdallah.shami@uwo.ca).

X. Wang is a professor with the Department of Electrical and Computer Engineering, Western University, 1151 Richmond Street London, Ontario, Canada (e-mail: xianbin.wang@uwo.ca).

granularities might be challenging due to the network's changing conditions.

Moreover, NT applications only care about the telemetry data, instead of how to obtain those data. Then, a sort of telemetry tasks orchestration should be used in order to achieve efficient tasks distribution and telemetry data acquisition. In addition to upper-level monitoring applications, the orchestration of NT tasks should consider real-time and changing network flows. Nevertheless, how to achieve high-quality network measurement at low cost according to the existing network status is a key issue of NT that needs further research and development [1].

We then propose to address the problem of efficiently gathering NT data through a modular framework that is independent of the NT scheme in use. The core of our solution is Machine Learning-based NT Controller, which autonomously decides the granularity of the measurements to be transmitted. This decision is made by taking into account the network managers' needs and the traffic that a network element is experiencing. To achieve so, we consider an anomaly detection mechanism, which aims to discover unexpected events in the traffic data. In this way, several types of traffic are identified and the telemetry data are selectively transmitted based on those traffic types.

Accordingly, our proposed mechanism utilizes a classifier to detect anomalous behaviours in the traffic that a network element is forwarding. The classification model considers the traffic characteristics that common cyberattacks expose, so that the flows are segmented in different types (including benevolent traffic) based on those characteristics. Thus, our design aims to classify the network traffic anomalies and, according to this segmentation, decides the level of granularity of the telemetry data that a network element should transmit. Our rationale behind this proposal is that malicious traffic patterns can be exploited to determine the frequency in which NT data should be sent. In other words, when normal patterns of flows are detected, there is no need for a very fine granularity in the NT data gathering. In this way, for example, the queue occupancy measurements are not to be transmitted very frequently unless malicious traffic is negatively affecting the network elements' buffers. In fact, this kind of approach has been researched in the literature. For instance, authors in [4] study the behaviour of some network performance metrics, such as the buffer occupancy, as a consequence of malevolent traffic produced by attacks like Denial of Service (DoS), Distributed Denial of Service (DDoS), and SYN/TCP flooding (a type of DoS/DDoS flood attack using the TCP protocol). Therefore, we aim to take advantage of such a relationship between the traffic patterns that typical cyberattacks pose and the metrics that an NT mechanism usually collects and transmits.

For the reasons explained above, we denote our solution as Traffic-Aware Network Telemetry, or TANT for short. A general overview of the TANT solution is shown in Fig. 1. As can be seen, the main components of the system at the network elements are a traffic flow classifier and an NT controller, which operates according to the telemetry standard in use. The NT controller determines the granularity of the telemetry data to be transmitted depending on the outcomes of the local traffic classifier. In summary, the main contributions of the TANT scheme and this work are:

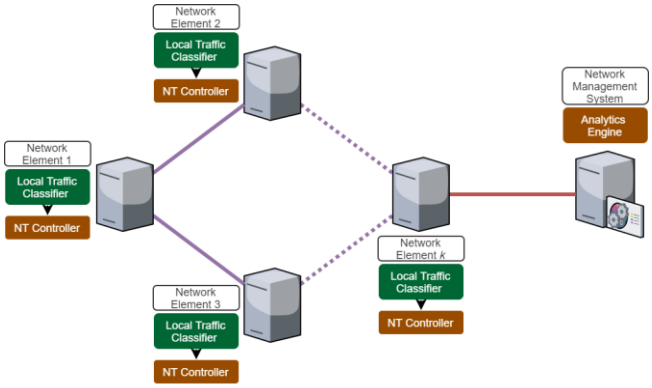


Fig. 1. TANT system overview. Each network element comprises a traffic classifier and an NT controller, which transmits the NT data to the NT engine.

- A flexible framework to achieve efficient NT that can be adapted to a variety of NT schemes regardless their way of operation (in-band or out-of-band).
- A proof-of-concept on a non-static NT mechanism, which can be intelligently adjusted to mitigate the network overhead that NT data gathering and transmission produce.
- The design of a lightweight traffic classifier that does not consider the classical 5-tuple (protocol type, source IP address and port, and destination address and port) to identify different types of traffic.
- A methodology to evaluate and implement inference acceleration of ML algorithms making predictions in real-time scenarios, such as the NT use case presented in this paper.
- An open-source environment for real-time evaluation whose code is publicly available for further research and development.

It is important to point out that, although the TANT framework could be applied in networking setups, such as Wide Area Networks (WAN) and Internet Service Providers (ISP) networks, its application would be more representative in networks delimited by the local management of a single organization, like enterprises or campuses networks. Also, the implementation of the TANT framework and the utilization of its NT data to tackle inter-domain scenarios' problems, like the one presented in [5], needs further research that is out of the scope of this work.

A. Related Work

The challenges that NT poses have been addressed by the research community, in both academic and industry settings, with diverse approaches that generally fell in one of these two main categories: in-band telemetry and out-of-band telemetry. In-band telemetry refers to the case when the NT data transmission usually shares the same link, path, or packet with the users' data whereas the transportation of out-of-band telemetry data does not [1]. The in-band telemetry solutions

reviewed in this subsection are related to the In-band Network Telemetry (INT) Dataplane Specification [6]. This specification defines the monitoring system as a system that collects telemetry data sent from different network elements. The components of the monitoring system may be physically distributed but logically centralized. Additionally, with INT, the original data packets are monitored and may be modified to carry INT instructions and INT metadata (telemetry data). It is important to highlight that there are other in-band telemetry specifications different from the INT standard. For this reason, we make the distinction between these two terms.

Existing NT systems usually trade off expressiveness (accuracy of the measurements) for scalability (amounts of the telemetry data collected), or vice versa. That is why most of the INT-based schemes aim to reduce the telemetry data transportation overhead and, at the same time, try to avoid losing too much measurement accuracy. Accordingly, authors in [7] present a sampling-based INT mechanism, in which the source node inserts INT headers into the packets at a configurable rate to reduce the overhead. To compensate for the accuracy, their solution also supports a sampling based on events, in which metadata is inserted only when the latency difference between the last hop and the current hop exceeds a predefined threshold. Similarly, Chowdhury *et al.* propose a lightweight INT-based scheme to reduce the overhead by trying to estimate the amount of error that can be introduced at the INT collector if the requested telemetry data are not piggybacked on the current packet [8]. For estimating this error, a predictor function based on Exponentially Weighted Moving Average is used for each telemetry data item of interest.

By encoding the requested data on multiple packets, authors in [9] introduce a probabilistic INT method that bounds the per-packet overhead as low as one bit. The solution supports several aggregation operations that allow efficient encoding of the aggregated data onto packets: per-packet aggregation, static per-flow aggregation, and dynamic per-flow aggregation. Conversely, Wang *et al.* introduce a bandwidth-efficient INT system by tracking the rules matched by the packets of a flow in a previous period [10]. Their proposed solution assigns globally unique IDs to every rule and stores rule-changed INT reports in a database server so that the rate of generated INT reports is reduced. In contrast, [11] considers the overhead not only at the data plane, but also at the control and management planes while employing INT. The authors model the INT orchestration as an optimization problem and propose two heuristic algorithms to produce feasible solutions in polynomial computational time with respect to the network size and number of flows. From [11], we find interesting the idea of taking into account the three SDN planes to reduce the INT overhead in an orchestrated manner. Finally, Kim *et al.* present a selective INT scheme where an algorithm adjusts the insertion ratio of packets to be monitored according to the frequency of significant changes in network data [12].

What all the solutions reviewed above have in common is the goal to make NT efficient in terms of the usage of the network resources, such as bandwidth and network elements' computational limitations. However, those schemes delimit their applicability to the INT specification, as the per-packet NT data overhead is assumed as the main issue to solve. Although INT is becoming the mainstream telemetry standard, we advocate for a more generalized framework that can also be applied to other in-bound telemetry mechanisms or even out-of-band ones. On the other hand, [7] and [12] are the schemes that relates the most to our proposed framework in terms of the adjustment of the NT data granularity (or rate) to reduce overhead.

II. TANT Traffic Classifier

The traffic classification process involves the identification of both normal and different types of abnormal traffic flows. We then design the traffic classifier of our solution using the CICIDS2018 dataset as a benchmark [13]. This and other datasets from the Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick have been widely used by researchers worldwide to evaluate their network traffic-related methods, such as Internet traffic classification. The CICIDS2018 dataset contains benign and common attacks, which resembles true real-world network data. It also includes the results of the network traffic analysis with labeled flows based on the time stamp, source and destination IP addresses, source and destination ports, and protocols. The dataset was generated with realistic background traffic to profile the abstract behavior of human interactions and includes benign traffic. The final dataset was gathered from different attack scenarios whose attacking infrastructure considers 50 machines and the victim organization has 420 hosts and 30 servers. More than 80 statistical features are extracted from the network traffic in forward and backward directions, as described in [13].

Therefore, the traffic classifier considers multiple classes, including benign traffic and the malicious traffic described by these attacks: DoS-Hulk, DoS-SlowHTTP, DDoS-HOIC, DDoS-LOIC, FTP-BruteForce, and SSH-BruteForce. We chose these attacks because they are the most representative classes in the CICIDS2018 dataset and encompass both TCP and UDP flows. A description of these attacks and the methodology used to obtain their traffic data can be found in [14]. After merging and cleaning the data subsets corresponding to the chosen attacks, the final dataset ended up containing 4,723,155 samples. For the training and test of the traffic classifier, the final dataset is split into 70% and 30%, respectively.

On the other hand, one of our goals is to design a lightweight and protocol-independent scheme to identity network traffic. To achieve so, we first perform an explainable feature engineering process. As we are interested in controlling the granularity of the NT, there is an initial feature selection that considers all time-related features, 27 in total, which are based on traffic flows' metrics (see Table I). It is important to highlight that, in the context of this work, we consider a traffic flow according to the IETF's RFC 7011, Specification of the IP Flow Information Export (IPFIX): "A Flow is defined as a set of packets or frames passing an Observation Point in the network during a certain time interval. All packets belonging to a particular Flow have a set of common properties." [15]. Those common properties include the packet header fields, *i.e.* the 5-tuple of source IP address, destination IP address, source port, destination port, and protocol type. Similarly, we point out that the data used for our analysis and proposed solution correspond to the RFC 7011's definition of Flow Records, which contain measured properties of the flows at the Observation Point. In this way, the features

TABLE I. TIME-RELATED TRAFFIC FEATURES

Feature	Description
Active Max	Maximum time a flow was active before becoming idle
Active Mean	Mean time a flow was active before becoming idle
Active Min	Minimum time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Total	Total time between two packets sent in the backward direction
Bwd Packets/s	Number of backward packets per second
Flow Byte/s	Number of flow bytes per second
Flow duration	Duration of the flow in microseconds
Flow IAT Max	Maximum time between two packets sent in the flow
Flow IAT Mean	Mean time between two packets sent in the flow
Flow IAT Min	Minimum time between two packets sent in the flow
Flow IAT Std	Standard deviation time between two packets sent in the flow
Flow Packets/s	Number of flow packets per second
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Total	Total time between two packets sent in the forward direction
Fwd Packets/s	Number of forward packets per second
Idle Max	Maximum time a flow was idle before becoming active
Idle Mean	Mean time a flow was idle before becoming active
Idle Min	Minimum time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active

of the input data for the traffic classifier are based on the Flow Records but not on the flows' common properties themselves, such as the 5-tuple.

As a next step in the feature engineering process, we normalized the values of the preselected features and perform a correlation analysis of them. Intuitively, one can suppose that several time-related features described in Table I are strongly correlated. For example, some of the forward-direction metrics should have a significant correlation with their backward-direction counterparts, since the majority of the traffic data correspond to TCP flows. For this reason, we perform another

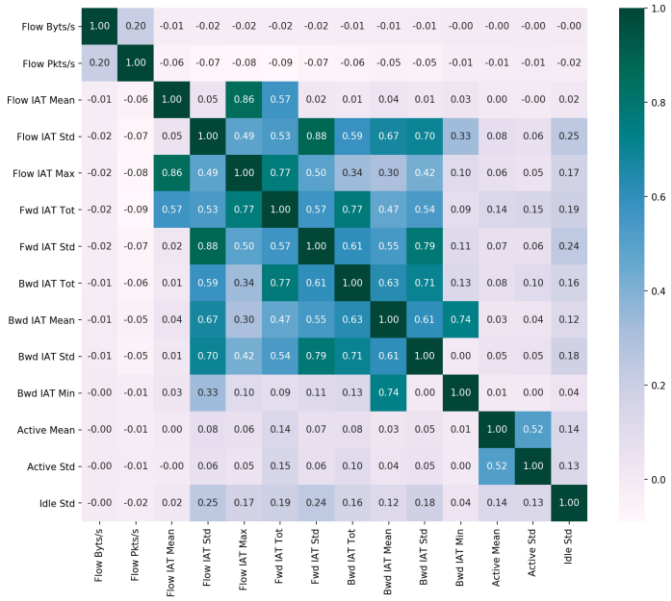


Fig. 2. Correlation matrix of the selected features based on the Pearson coefficients.

feature selection using the Pearson correlation coefficients. These coefficients are a statistical measure of the linear dependency between two vectors, which are assumed to be normally distributed and to contain n elements each [16]. Thus, the Pearson correlation coefficients are calculated as follows:

$$r(x_1, x_2) = \frac{\sum_{i=1}^{n}\left(x_1^{(i)} - \bar{x}_1\right)\left(x_2^{(i)} - \bar{x}_2\right)}{\sqrt{\sum_{i=1}^{n}\left(x_1^{(i)} - \bar{x}_1\right)^2}\sqrt{\sum_{i=1}^{n}\left(x_2^{(i)} - \bar{x}_2\right)^2}} \quad (1)$$

where x_1 and x_2 are the vectors of the two features being analyzed, $\bar{x}_1$ and $\bar{x}_2$ the mean values of those feature vectors, respectively, and $x_j^{(i)}$ refers to the value of the instance i from feature j. For each coefficient, $r(x_1, x_2) \in [-1, 1]$ and a positive number close to 1 means that an increase or decrease in the values of x_1 is met with the same trend, increase or decrease, in the values of x_2. Accordingly, we discard one of the features whose values have a correlation greater than 0.9 with another feature. The resulting 14 features and their coefficients after carrying out the correlation analysis are shown in Fig. 2.

For the traffic flows classifier, we consider the following classification techniques, which are deemed by ML researchers and practitioners as efficient methods for multi-class problems: Logistic Regression with Stochastic Gradient Descent training (LR-SGD), linear Support Vector Machines with Stochastic Gradient Descent training (SVM-SGD), Random Forest (RF), Extra Trees (ET), Light Gradient Boosting Machine (LightGBM), and Extreme Gradient Boosting (XGBoost). In order to compare the outcomes of these methods, we use the F1-score as the statistical measure of the classification quality, since the dataset happens to be imbalanced. The F1-socre is defined by the harmonic mean of the precision and the recall [17], as follows:

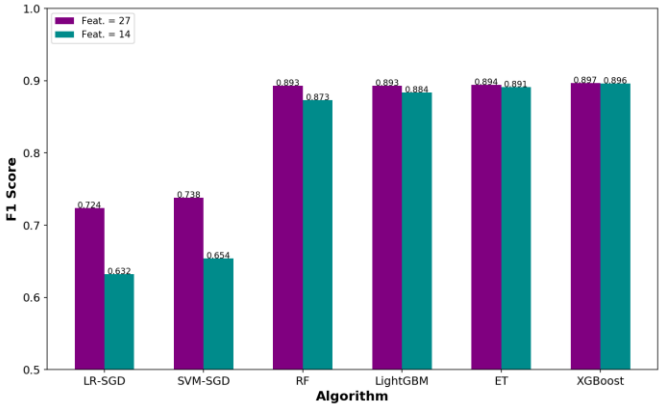


Fig. 3. Classifiers' scores comparison before and after first feature selection.

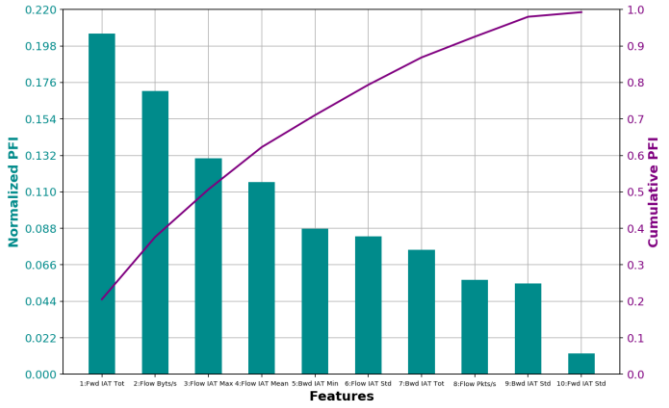


Fig. 4. Feature ranking based on PFI calculation.

$$F_1 = 2\frac{P \cdot R}{P+R} \tag{2}$$

where the recall, R, represents the ratio between the number of correct positive results and the number of all relevant samples, and the precision, P, is the relation between the number of correct positive results and the number of positive results. Fig. 3 shows the comparison of the F1-scores of the abovementioned classifiers before and after performing the feature selection based on the Pearson correlation analysis.

As can be seen, the accuracy of the LightGBM, ET, and XGBoost classifiers are slightly lower when almost half of the features (14 out of 27) are used. In contrast, although faster in training, LR-SGD and linear SVM-SGD algorithms are outperformed by the other three in both cases. It is important to highlight that we are more interested in the inference times, rather than the training times, as our goal is to come up with a lightweight traffic classifier to efficiently make predictions in real time. That is why we are not comparing the training times that the algorithms take, however, the inference times will be compared in the performance evaluation of the proposed solution. For the classifier design, we intent to engineer a method that employs a reduced number of features without sacrificing the accuracy too much, so that its complexity is lowered, especially when inferencing traffic anomalies for the NT control process.

As a further step, we complete another feature extraction based on the importance analysis, which helps identify what the most informative features are during the classification process. In this way, we could reduce even more the number of features needed for the inference, if the accuracy is not significantly degraded. We explore this possibility by calculating the Permutation Feature Importance (PFI): a model inspection technique and especially useful for non-linear classifiers. This technique is agnostic to any model and breaks the relationship between the feature and the target. The PFI for a feature x_j is defined as the average increase in prediction loss, $\mathcal{L}$, when the feature is permuted in training or test dataset [18], as follows:

$$PFI_j = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{n}\sum_{i=1}^{n}\left(\mathcal{L}\left(y^{(i)}, f\left(\hat{x}_j^{m(i)}, x_{\bar{j}}^{(i)}\right)\right) - \mathcal{L}\left(y^{(i)}, f\left(x^{(i)}\right)\right)\right) \tag{3}$$

where $f\left(x^{(i)}\right)$ and $y^{(i)}$ refer to the model predictions and the targets, respectively , $\hat{x}_j^{m(i)}$ is a permutation of x_j, M is the number of repeated permutations, and $x_{\bar{j}}$ refers to the complementary feature space. Fig. 4 shows the PFI coefficients calculated for the classification algorithm with the highest F1-score, *i.e.* XGBoost, over the test subset (meaning $n = 1{,}416{,}947$) and with $M = 15$. As can be seen, the first nine features in importance contribute to over 95% of the classification process.

Finally, we test the LightGBM, ET, and XGBoost classifiers only with the top nine features selected from the PFI analysis. The new F1-scores are compared with the previous ones in Table II, which reveals that the changes in the classification accuracy are minuscule, especially for the LightGBM and XGBoost algorithms. This explainable feature space reduction allows the traffic identification process to be less complex and, as a result, to achieve shorter inference times for the NT control mechanism at each network element. The inference performance will be evaluated and discussed in the next subsection. For that evaluation, we compare the best two techniques in terms of accuracy in the reduced feature space, *i.e.* LightGBM and XGBoost, which yield F1-scores of 0.899 and 0.897, respectively, after tuning their hyperparameters. Note that we avoid overturning the hyperparameters that add complexity and make the models more likely to overfit, such as the maximum depth and the maximum leaves of the trees. In this way, we keep the structure of both the LightGBM and XGBoost models comparable for the inference performance benchmarking as well as more generalized for making predictions on unseen data.

A. Inference Acceleration

As explained earlier, the main goal of having a reduced feature space without significantly sacrificing the traffic classification accuracy is to decrease the model's complexity and, therefore, the inference time. In achieving so, the classifier

TABLE II. F1-SCORES COMPARISON AFTER SECOND FEATURE SELECTION

Classifier	Feat. = 27	Feat. = 9	Difference
ET	0.89370	0.89102	0.00268
LightGBM	0.89300	0.89232	0.00068
XGBoost	0.89667	0.89585	0.00082

may be implemented in more realistic network scenarios and operate in real time. We go further towards this goal by utilizing an ML inferencing accelerator. For this work, we employ the tools from the Open Neural Network Exchange (ONNX) framework to improve the performance of our model. ONNX is an open ecosystem that provides a standard format for representing the prediction function of trained ML models [19]. It defines an extensible computation graph model and the models trained using several ML frameworks can be exported to ONNX. With ONNX, each computation dataflow graph is structured as a list of nodes that form an acyclic graph, a process known as serialization. As a result, ONNX offers a convenient interoperability of ML models across frameworks and that is why it is widely backed by important companies in the Artificial Intelligence (AI) industry.

We then operationalize the optimized traffic classifier by ONNX with the ONNX Runtime: a high-performance and resource-efficient inference engine for ML models that takes advantage of the specific hardware capabilities where the model is run on [20]. ONNX Runtime can perform inference for any prediction function converted into the ONNX format and its cross-platform nature allows it to be run on different hardware and operating systems. In this manner, ONNX Runtime tries to parallelize the model's operations and optimizes the model graph by applying graph transformation, that is, elimination and fusion of graph nodes.

Accordingly, we assess the efficiency of the LightGBM and XGBoost classifiers when making predictions for one observation at a time, a common situation in computer networking scenarios such as the use case for this work. To achieve so, we take 15,000 random samples from the resulting dataset after reducing the feature space, as explained in the previous subsection, and measure the processing time that each model takes to predict the type of traffic flow (one sample corresponds to one flow). Similarly, we trace the allocated memory to process each prediction.

Fig. 5 shows the averaged computation times and the averaged RAM usage over the 15,000 samples. Note the logarithmic scale used for comparing the processing times. As can be seen, XGBoost algorithm achieves faster predictions than LightGBM on batches of one sample in size. More importantly, it is evident that ONNX optimization does accelerate the inference time by a factor of 4.9x and 3.6x for LigthGBM and XGBoost, respectively. Similarly, the memory usage is significantly reduced when ONNX is employed, being nearly the same for both algorithms and improved by a factor of 15.9x, for LigthGBM, and a factor of 15.3x, for XGBoost. All these measurements were obtained by running the ONNX inference calls on a machine with Intel® Xeon® CPU E5-2686, four cores @ 2.30 GHz, and Ubuntu 18.04.4. We also point out that these

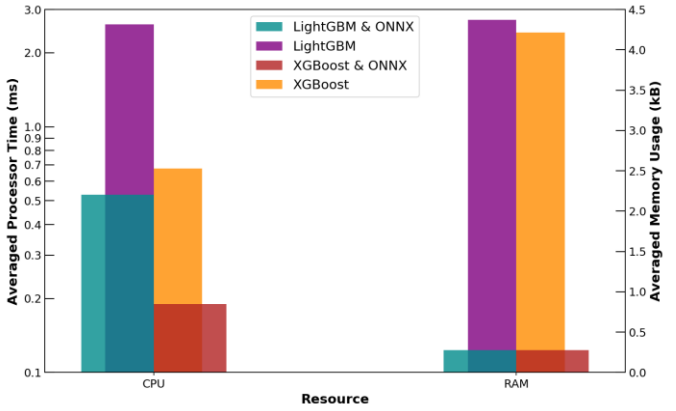


Fig. 5. Computational resources used by the classifier algorithms for inferring in a single call.

measured values correspond to the complexity exhibited by a single flow and that complexity grows linearly with the number of flows, N_f, meaning a time complexity of $\mathcal{O}(N_f)$ when multiple flows are considered.

It is important to highlight that several research works have reported lower processing times of LightGBM, although with lower accuracy scores too, when compared to XGBoost. However, LightGBM may be faster when being trained or making batch predictions, but not when inferencing on one observation at a time. This is due to the hyperparameter tuning that the LightGBM needs in order to get similar or higher accuracy than XGBoost. That tuning might result in a more complex model, which significantly affects the inferencing performance. Therefore, we consider the XGBoost algorithm for our proposed efficient telemetry scheme and the performance assessments in the rest of this work. Again, as we discussed in the introduction, we are more interested in attaining a reasonable traffic identification accuracy with an algorithm that provides fast inference in a single call.

III. TANT CONTROLLER

Network applications require NT to be elastic enough in order to efficiently use the network resources and reduce the performance degradation. Also, routine network monitoring should cover the entire network with low data sampling rate. However, NT data rate may be boosted when issues arise or trends emerge [21]. That is the ultimate goal of the Network Telemetry Controller module in our scheme. As a use case, we evaluate our solution by means of a postcard-like telemetry mechanism, such as the Postcard-Based Telemetry (PBT) or the INT in eXport Data mode (INT-XD). In this mode, INT nodes directly export metadata from their dataplane to the monitoring system based on the INT instructions configured at their Flow Watchlists. A Flow Watchlist is a dataplane table that matches on packet headers and applies INT instructions on each matched flow. The instructions indicate which INT metadata to collect at each INT node and they are either configured at each INT-capable node's Flow Watchlist or written into the INT header. Although INT-XD is a valid mode of operation, it does not represent the classic and the default hop-by-hop INT operation, where the INT devices embed both instructions and metadata, *i.e.* telemetry data, and the packets are modified the most [6].

TABLE III. GRANULARITY LEVELS

Granularity (ms)	Type(s) of Traffic
100	Benign
10	DDOS attack-HOIC, DoS attacks-Hulk
1	SSH-BruteForce
0.5	DDOS attack-LOIC-UDP
0.1	FTP-BruteForce, DoS attacks-SlowHTTPTest

Similarly, the PBT-M, a packet-marking variation of the PBT, does not require the encapsulation of telemetry instruction headers, avoiding some of the implementation challenges of the instruction-based PBT and the default INT, also known as on-path telemetry in passport mode [22]. PBT-M uses a marking-bit in the existing headers of user packets to trigger the NT data collection and export. If an NT node detects the mark, a postcard (a dedicated packet triggered by a marked user packet) is generated and transmitted to the NT collector. This postcard packet contains the data requested and configured by the management plane. The main advantage of PBT-M is that it avoids growing user packets with new headers and introducing new data plane protocols. However, the data plane devices need to be configured to know what NT data to collect. Another important benefit of PBT-M is that the collected NT data can be transported independently through in-band or out-of-band channels and the types of data collected from each node may be different according to the application requirements and nodes' capabilities. Nevertheless, since each postcard packet has its own header, the overall network bandwidth overhead of PBT may be higher than the passport-based NT, depending on the number of postcards to be transmitted.

For the reasons explained above, our TANT solution is designed as a PBT-M-like scheme that additionally takes into account the granularity of the NT data to be transported, so that the network bandwidth overhead is minimized. To achieve so, we assume that the levels of granularity can be marked through some or all of the 8 bits of the Type of Service (ToS) field of a standard IP packet header. In this way, a network device acting as the NT Source detects the type of traffic that is forwarding and, based on it, marks the level of granularity needed. Then, both NT Source, NT Transit, and NT Sink devices send postcard packets to the NT Monitoring Engine. Finally, the NT Sink unmarks the IP headers. It is important to highlight that, similar to the PBT-M scheme, TANT assumes that the NT devices are instructed on what kind of NT data collect and transmit by the management plane beforehand.

With respect to the granularity levels of the NT data, we analyze the packets' Inter-Arrival Time (IAT) of the types of traffic identified by the classifier. To this end, we explore the values of the attribute describing the average IAT between two packets sent in the forward direction (Fwd IAT Mean) from the whole CICIDS2018 dataset. We point out that, by selecting these IAT values, our analysis is more realistic so that the granularity levels are applicable to real-time scenarios. Also, the selection of the IAT values in the forward direction is consistent with the NT specifications described above, in which a network element triggers the telemetry tasks and forwards the NT instructions to the next elements ahead. On the other hand, the

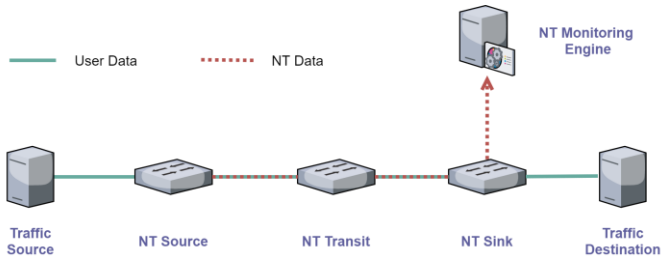


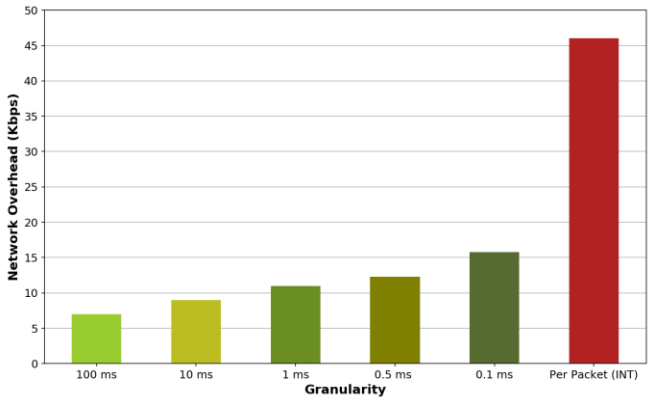
Fig. 6. Network scenario for the TANT use case evaluation.

packet IAT values have a significant relationship with the type of traffic that a network element is forwarding and cannot be easily obfuscated or manipulated [23]. For this reason, the IAT characteristics of packets have been used to detect malicious traffic patterns, such as the one described by DDoS attacks [24]. Consequently, we define five levels of NT granularity according to values of the Fwd IAT Mean feature for each traffic class, as shown in Table III. Note that this attribute is not part of the group of nine features finally selected for the proposed traffic classifier (see Fig. 4).

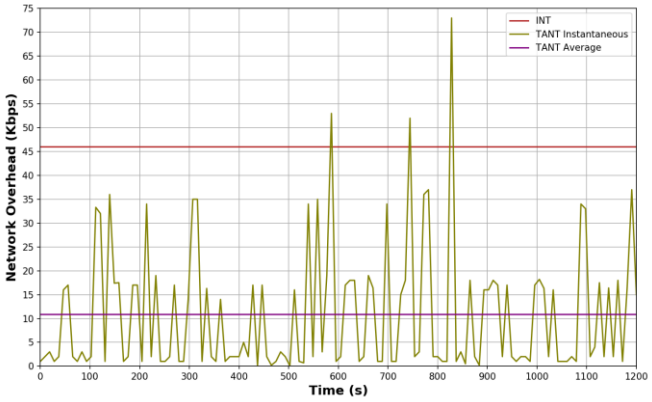
IV. EXPERIMENTATION AND EVALUATION RESULTS

In this section, we provide the details about the experimentation setup for the evaluation of our proposed solution. The network topology for our experiments is similar to the one presented in [12], although we consider one path only for the NT Transit devices, instead of two, Fig. 6. The reason why is because their experiments focus on the path changes whereas ours are focused on the variations of the types of traffic. Nevertheless, the ultimate goal is the same: to compare the performance of the proposed NT mechanism against INT when the traffic flows are affected, either by the paths they are being transported on or the specific traffic type they are carrying. Likewise, we use the Mininet emulator as the software tool to implement and evaluate the TANT prototype in the described network topology [25]. Mininet is a suitable tool for our use case, as it allows a flexible SDN environment with high degree of confidence for real-time tests [26].

On the other hand, we use Scapy [27] as the tool to manipulate the standard IP packets in the TANT implementation. Scapy can be used to construct packets of a variety of existing or new protocols, send and receive them, match requests and replies, and more [28]. Accordingly, the NT Source, NT Transit, and NT Sink devices transmit manipulated PBT-M-like packets to the NT Monitoring Engine, as described in the previous section, by means of Scapy's protocol stacking and fields manipulation functionalities. More specifically, an IP packet is created with the standard IP header and 12 bytes are stacked as the payload of that packet. The rationale behind having a 12-byte payload is that we intend to compare our TANT solution to the conventional INT and the solution presented in [12], which is a scheme based on INT, although with modifications. According to that work, three types of INT metadata are considered as examples for its evaluation: switch ID, hop latency, and queue occupancy. These NT data is inserted



a)



b)

Fig. 7. Evaluation resutls of TANT and its comparison against the classic INT. a) Network bandwidth overhead reduction per granularity level. b) Instantaneous and average network overhead measured during 1,200 seconds of network emulation.

into the user data every hop by each network element involved in the INT process, *i.e.* the NT Source, the NT Transit, and the NT Sink. Taking into account the INT specification [6], INT metadata per measured variable occupy 32 bits (4 bytes). Therefore, we consider NT data of 12 bytes in length to make it comparable to the three INT measurements considered in the experiments of [12].

In relation to the evaluation of both TANT modules working together, we run the Mininet emulation as follows: the Traffic Source host picks traffic samples randomly, meaning that any of the types of traffic described in Table III may be selected. Based on this random selection, the TANT Traffic Classifier determines the type of traffic by considering the selected flow attributes, as explained in Section II: Flow Byts/s, Flow Pkts/s, Flow IAT Mean, Flow IAT Std, Flow IAT Max, Fwd IAT Tot, Bwd IAT Min, Bwd IAT Std, and Bwd IAT Tot. Based on the identified traffic, the NT granularity is established for that flow according to the levels showed in Table III. Afterwards, the NT Source creates an IP packet and uses three of the eight ToS bits to indicate the level of granularity needed, as explained in Section III. Note that three bits are enough to mark all the five granularity levels that the TANT scheme considers for the use case presented in this paper. However, more bits could be used for that purpose. Additionally, the NT Source inserts the 12-byte payload and send the NT packet to the NT Monitoring Engine. As TANT implements a PBT-like NT, all the NT Transits and the NT Sink perform a similar task in order transmit their NT data to the NT Monitoring Engine. Recall that, similar to the PBT-M specification, we are assuming that all the NT nodes know the measurement data that they need to collect and send a priori. This could be accomplished by means of instructions from the management plane. Finally, each NT node starts transmitting the NT data of the pre-determined measurements in the granularity intervals specified in the ToS bits of the IP packet.

In order to determine the network overhead, we measure the throughput every five seconds using the iPerf tool [29]. More specifically, we set up a pair of monitor hosts in the emulation environment, one of them actively logging the measured throughput by means of sending probe packets to the other one in 5-second intervals. In this way, we measure the throughput without transmitting any NT data and, right after that, the network throughput while the NT data is being transmitted for another 5 s. The network overhead is then calculated by subtracting the measured throughput with NT data from the measured throughput without it. Again, this method is similar to the one utilized in [12].

Fig. 7 depicts the results of our solution evaluation. As can be seen, the proposed TANT scheme achieves a substantial lower overhead when compared to the regular INT mechanism: the worst-case granularity, *i.e.* 0.1 ms, represents less than 50% of the INT's overhead (Fig. 7a). With respect to the emulation in real time, TANT attains a reduction of 76.4% in network overhead, on average (Fig. 7b). Furthermore, this overhead decrease outperforms the reduction reported in [12], which is 37% less than the conventional INT. It is important to point out that there are some spikes of the instantaneous measures of TANT that overpass the INT overhead. These spikes are due to the abnormal traffic detected by the classifier, which, at the same time, lowers the granularity. However, our TANT mechanism adaptively changes the granularity of the NT data when normal flows or other types of traffic flows are detected. As a result, the overall network overhead is considerably lesser than that produced by the per-packet INT's granularity.

Finally, we would like to mention that the code of the experiments described in this subsection is publicly available at [30]. We intent to make our contribution accessible to researchers and developers who are actively working on similar problems of efficient NT. Please cite this paper if you use any posted script for your own works.

V. Conclusions and Future Work

In this work, we presented a novel framework for efficient collection and transportation of network telemetry data by making the network devices "aware" of the traffic types that they are forwarding. To accomplish so, our TANT scheme comprises two principal modules: an ML-based traffic classifier and an NT controller that adjusts the level of granularity of the telemetry data. We also showed how the inference process of the classifier can be accelerated in order to make per-flow predictions in shorter times and using less memory, important characteristics for any NT mechanism working in real time. Finally, we

evaluated the performance of the proposed scheme by means of network emulations and demonstrated that TANT can reduce the network bandwidth overhead to about ¼ of the overhead caused by the classic INT scheme.

As a future work, it would be interesting to include subcategories of benign traffic for the flow classification process. In this way, other types of traffic can be detected event if they do not correspond to cyber attacks. These subclasses of benevolent, but abnormal, traffic might be very useful to detect and take actions on events that can degrade the network performance. However, quality datasets of real network traces that include those situations need to be generated or made publicly available without compromising private data. Additionally, it would be worth exploring the application of the Federated Learning approach to the traffic classifier in the TANT scheme for use case scenarios such as the one presented in [5]. In this way, a more scalable solution could be accomplished by decentralizing the learning process and, as a result, a more seamlessly deployment across several local networks or even WANs would be also possible.

REFERENCES

[1] L. Tan *et al.*, "In-band Network Telemetry: A Survey," *Comput. Netw.*, vol. 186, p. 107763, Feb. 2021, doi: 10.1016/j.comnet.2020.107763.

[2] A. Morton, "Active and Passive Metrics and Methods (with Hybrid Types In-Between)," RFC Editor, 7799, May 2016. doi: 10.17487/RFC7799.

[3] M. Yu, "Network telemetry: towards a top-down approach," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 1, pp. 11–17, Feb. 2019, doi: 10.1145/3314212.3314215.

[4] N. Ouroua, W. Bouzegza, and M. Ioualalen, "Formal Modeling and Performance Evaluation of Network's Server Under SYN/TCP Attack," in *Mobile, Secure, and Programmable Networking*, Cham, 2017, pp. 74–87. doi: 10.1007/978-3-319-67807-8_6.

[5] C. A. Gomez, X. Wang, and A. Shami, "Federated Intelligence for Active Queue Management in Inter-Domain Congestion," *IEEE Access*, vol. 9, pp. 10674–10685, 2021, doi: 10.1109/ACCESS.2021.3050174.

[6] "In-band Network Telemetry (INT) Dataplane Specification," The P4.org Applications Working Group, Version 2.1, Oct. 2020. Accessed: Feb. 11, 2021. [Online]. Available: https://github.com/p4lang/p4-applications/tree/master/docs

[7] D. Suh, S. Jang, S. Han, S. Pack, and X. Wang, "Flexible sampling-based in-band network telemetry in programmable data plane," *ICT Express*, vol. 6, no. 1, pp. 62–65, Mar. 2020, doi: 10.1016/j.icte.2019.08.005.

[8] S. R. Chowdhury, R. Boutaba, and J. François, "LINT: Accuracy-adaptive and Lightweight In-band Network Telemetry".

[9] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "PINT: Probabilistic In-band Network Telemetry," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, New York, NY, USA, Jul. 2020, pp. 662–680. doi: 10.1145/3387514.3405894.

[10] S.-Y. Wang, Y.-R. Chen, J.-Y. Li, H.-W. Hu, J.-A. Tsai, and Y.-B. Lin, "A Bandwidth-Efficient INT System for Tracking the Rules Matched by the Packets of a Flow," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019, pp. 1–6. doi: 10.1109/GLOBECOM38437.2019.9013581.

[11] J. A. Marques, M. C. Luizelli, R. I. Tavares da Costa Filho, and L. P. Gaspary, "An optimization-based approach for efficient network monitoring using in-band network telemetry," *J. Internet Serv. Appl.*, vol. 10, no. 1, p. 12, Jun. 2019, doi: 10.1186/s13174-019-0112-0.

[12] Y. Kim, D. Suh, and S. Pack, "Selective In-band Network Telemetry for Overhead Reduction," in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, Oct. 2018, pp. 1–3. doi: 10.1109/CloudNet.2018.8549351.

[13] "IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB." https://www.unb.ca/cic/datasets/ids-2018.html (accessed Mar. 03, 2021).

[14] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in *ICISSp*, 2018, pp. 108–116.

[15] P. Aitken, B. Claise, and B. Trammell, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," RFC Editor, RFC 7011, Sep. 2013. doi: 10.17487/RFC7011.

[16] M. Braschler, T. Stadelmann, and K. Stockinger, *Applied Data Science: Lessons Learned for the Data-Driven Business*. Springer International Publishing, 2019.

[17] O. Campesato, *Artificial Intelligence, Machine Learning, and Deep Learning*. Mercury Learning & Information, 2020.

[18] C. Molnar, G. König, B. Bischl, and G. Casalicchio, "Model-agnostic Feature Importance and Effects with Dependent Features -- A Conditional Subgroup Approach," *ArXiv200604628 Cs Stat*, Jun. 2020, Accessed: Mar. 29, 2021. [Online]. Available: http://arxiv.org/abs/2006.04628

[19] "Open Neural Network Exchange," *GitHub*. https://github.com/onnx (accessed Mar. 09, 2021).

[20] "ONNX Runtime." https://www.onnxruntime.ai/ (accessed Apr. 10, 2021).

[21] H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia, and A. Wang, "Network Telemetry Framework," Internet Engineering Task Force, Internet-Draft draft-ietf-opsawg-ntf-07, Feb. 2021. Accessed: Apr. 13, 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-ntf-07

[22] H. Song *et al.*, "Postcard-based On-Path Flow Data Telemetry using Packet Marking," Internet Engineering Task Force, Internet-Draft (work in progress) draft-song-ippm-postcard-based-telemetry-09, Feb. 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-song-ippm-postcard-based-telemetry-09

[23] C. Rottondi and G. Verticale, "Using packet interarrival times for Internet traffic classification," in *2011 IEEE Third Latin-American Conference on Communications*, Oct. 2011, pp. 1–6. doi: 10.1109/LatinCOM.2011.6107404.

[24] O. Osanaiye, K. R. Choo, and M. Dlodlo, "Change-point cloud DDoS detection using packet inter-arrival time," in *2016 8th Computer Science and Electronic Engineering (CEEC)*, Sep. 2016, pp. 204–209. doi: 10.1109/CEEC.2016.7835914.

[25] "Mininet Overview - Mininet." http://mininet.org/overview/ (accessed May 06, 2021).

[26] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, New York, NY, USA, Oct. 2010, pp. 1–6. doi: 10.1145/1868447.1868466.

[27] P. Biondi, "Scapy - Packet crafting for Python2 and Python3." https://scapy.net/ (accessed May 13, 2021).

[28] R. R. S, R. R, M. Moharir, and S. G, "SCAPY- A powerful interactive packet manipulation program," in *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*, Dec. 2018, pp. 1–5. doi: 10.1109/ICNEWS.2018.8903954.

[29] "iPerf - The ultimate speed test tool for TCP, UDP and SCTP." https://iperf.fr/iperf-doc.php (accessed Apr. 15, 2021).

[30] C. A. Gomez, *cgomezsu/TANT*. 2021. Accessed: Jul. 20, 2021. [Online]. Available: https://github.com/cgomezsu/TANT

Cesar A. Gomez (Member, IEEE) received the B.E. degree in electronics engineering from St. Thomas Aquinas University at Bogota, in 2005, and the M.E.Sc. degree in telecommunications engineering from the National University of Colombia, in 2010. He is currently pursuing the Ph.D. in Electrical and Computer Engineering with Western University, Canada. His background includes industrial experience for over ten years in several network engineering roles at companies, such as Siemens, ZTE, and Nortel. His current research interest includes application of machine learning techniques towards the realization of the intelligent networking automation paradigm. He is also the Vice-Chair of the IEEE Computer Chapter, London Section, Region 7.

Abdallah Shami (Senior Member, IEEE) received the B.E. degree in electrical and computer engineering from Lebanese University, Beirut, Lebanon, in 1997, and the Ph.D. degree in electrical engineering from the City University of New York, New York, NY, USA, in 2002. He is currently a Professor with the ECE Department, Western University. He is also the Director of the Optimized Computing and Communications Laboratory. His research interests include performance and optimization modeling, machine learning and data analytics, the IoT, virtualization, cloud computing, and software-de_ned networks. He has chaired key symposia for IEEE GLOBECOM, IEEE ICC, IEEE ICNC, and ICCIT. He was the elected Chair of the IEEE Communications Society Technical Committee on Communications Software, from 2016 to 2017, and the IEEE London Section Chair, from 2016 to 2018. He is also an Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE NETWORK, and the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS.

Xianbin Wang (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the National University of Singapore, in 2001. From January 2001 to July 2002, he was a System Designer with STMicroelectronics. From July 2002 to December 2007, he was a Research Scientist/Senior Research Scientist with the Communications Research Centre Canada (CRC). He is currently a Professor and a Tier 1 Canada Research Chair with Western University, Canada. He has over 400 peer-reviewed journal and conference papers, in addition to 30 granted and pending patents and several standard contributions. His current research interests include 5G and beyond, the Internet-of-Things, communications security, machine learning, and intelligent communications. He is a Fellow of the Canadian Academy of Engineering and the Engineering Institute of Canada. He has received many awards and recognitions, including the Canada Research Chair, the CRC President's Excellence Award, the Canadian Federal Government Public Service Award, the Ontario Early Researcher Award, and six IEEE Best Paper Awards. He was involved in many IEEE conferences, including GLOBECOM, ICC, VTC, PIMRC, WCNC, and CWIT, in different roles, such as symposium chair, tutorial instructor, track chair, session chair, and TPC co-chair. He is also serving as the Chair of the IEEE London Section and the Chair of ComSoc Signal Processing and Computing for Communications Technical Committee. He was also an Associate Editor for the IEEE TRANSACTIONSONWIRELESS COMMUNICATIONS, from 2007 to 2011, and the IEEE WIRELESS COMMUNICATIONS LETTERS, from 2011 to 2016. He currently serves as an Editor/Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON BROADCASTING, and the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He is an IEEE Distinguished Lecturer.