Machine-Learning Based Active Measurement Proxy for IoT Systems

Wenqing Yan†, Christofer Flinta*, and Andreas Johnsson*†
* Ericsson Research, Sweden, Email: {*christofer.flinta, andreas.a.johnsson*}*@ericsson.com*
† Uppsala University, Department of Information Technology, Sweden, Email:{*wenqing.yan, andreas.johnsson*}*@it.uu.se*

***Abstract*—Network operators are accustomed to using IP-layer active measurements for assessing end-to-end network performance and expect that new technology, such as IoT, provides similar means. Unfortunately, active measurements in IoT systems are associated with both energy and network overhead.**

This paper presents and evaluates a novel active-measurement proxy approach, based on machine learning, that enables reduction of active measurement overhead in IoT systems. The paper describes the approach and its implementation. Further, the approach is evaluated in a IEEE 802.15.4 testbed, and the results show high-performing and accurate modeling.

***Index Terms*—Active Measurements, IoT, Machine Learning, Wireless Networks, Network Management.**

I. Introduction

Internet of Things (IoT) is an emerging technology that comes with great opportunities in health care, digitalized industry, and smart homes. However, to make IoT robust and reliable many challenges are still only partly addressed, for example network management and observability [1].

Wireless sensor networks (WSN), as part of an IoT system exemplified in Figure 1, monitors and interacts with the environment for different use-case driven purposes. A key challenge is observing and managing the wireless devices while keeping the operational expenditures and overhead low.

Telecommunication operators are accustomed to use Internet Protocol (IP) active measurements, such as ICMP [2] or TWAMP [3][4], for assessing observability in terms of network performance. They expect that new technology, such as IoT, provides a similar toolkit. Performance observability enables anomaly detection, bottleneck localization and timely mitigation, which lowers the costs for operations while providing a smoother service for the consumers.

In previous work [5] we presented an active measurement tool based on TWAMP, for IoT devices running Contiki OS and communicating over IEEE 802.15.4. The main contribution was a study on the relation between active-measurement frequency and overhead; specifically power consumption, self-induced loss, and round-trip time, see an example in Figure 2. Power, loss, and round-trip time (RTT), scaled from [0,8] seconds, are small and stable for measurement frequencies below 0.03Hz, marked by the dashed vertical line. Increasing the frequency further results in an unacceptable overhead. Active measurements in IoT systems must provide performance observability while at the same time minimize overhead. From the figure it is evident that the active measurement frequency should be limited.

978-3-903176-15-7

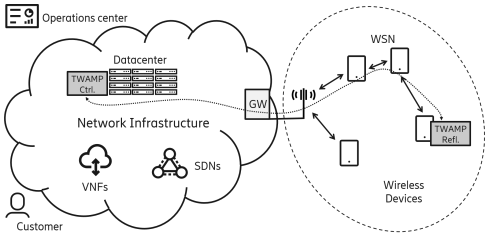

Fig. 1. An IoT system includes devices, wireless sensor networks (WSN), gateways (GW), a network infrastructure, operators, and customers. Active measurements for network performance assessment, using e.g. TWAMP, may cause overhead in the WSN.

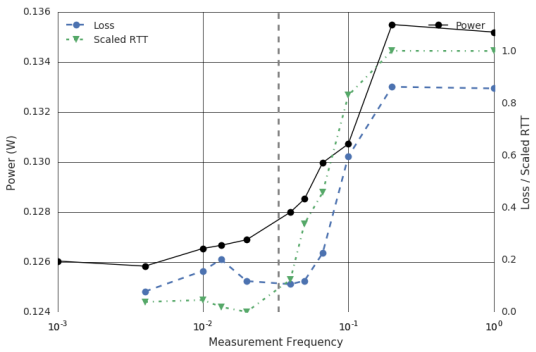

Fig. 2. Power, loss, and RTT vs measurement frequency, originally from [5].

This paper extends our work in [5] and contributes with a novel proxy-based approach, using machine learning, aiming at reducing the overhead of active measurements in WSNs while keeping the observability. The proxy, residing in the gateway (GW) in Figure 1, intercepts and drops measurement packets, with a source address in the network infrastructure and a destination address in the WSN. For each packet it estimates the WSN contribution to, for example, the total RTT. This contribution is the expected delay if the packet had been forwarded from the gateway and reflected back by the IoT device. The gateway waits during this estimated time before returning an emulated measurement packet to its origin. This paper specifically targets machine-learning models for estimation of the WSN contribution to the performance metrics. A testbed evaluation shows that the approach is accurate and generalizes across multiple scenarios.

The rest of the paper is organized as follows. Section II describes background and related work. Section III describes the measurement proxy and problem setting. Section IV describes the experimental setup, and Section V provides details about features and data traces. Section VI provides an evaluation. Section VII provides conclusions and future work.

II. Background and Related Work

The purpose of an IoT system is to monitor and/or control an environment or process using multiple sensors and actuators. Sensing and actuation produces business-critical data, and an example of this is monitoring and control of smart cities as discussed in [6]. However, from a network management perspective there are other coveted data; including performance metrics, errors, bottlenecks, and expected lifetime. For example, the authors of [7] consider performance metrics for an industrial IoT settings, which requires not only strict performance boundaries but also real-time monitoring. Normally, the observability of an IoT system in terms of performance is reduced by energy constraints, limited sensor capabilities, unreachable sensors, or limited performance. Hence, much work has focused on predictions, inference, process models, and measurements for increased observability.

Prediction-based examples include the work by Wolosz et al [8] where the throughput of a WSN is modeled using measurements of radio receiver signal strength indicators. Another example is modeling round-trip time (RTT) as a linear function of the distance [9]. A review of other predictive techniques for IoT is available in [10]. Further, performance degradation and fault localization techniques for WSNs are reviewed in [11]. Note however that these methods do not relate to IP-layer active measurements - one of the most commonly used troubleshooting approaches in telecom networks.

The purpose of active measurements is to observe the network health to enable proactive actions for optimization with respect to an objective. Metrics of interest include availability, one-way delay, RTT, delay variation, and loss. The basic idea in this study is to use an active measurement protocol with packet reflection, where measurement packets are sent from a controller to a reflector which sends back response packets to the controller. Time stamps are generated at the transmit and receive events on both sides, as in IETF TWAMP [3][4] or only at the sender side, as in IETF ICMP [2]. Ethernet and MPLS networks have their own versions based on ITU-T Y.1731 [21]. In TWAMP each probe packet is time stamped 4 times, enabling estimation of RTT, one-way delay if clocks are in sync, and the reflector processing time.

In [14] Metongnon et al presents a light-weight approach for active measurements in heterogeneous IoT networks consisting of both IEEE 802.11 and IEEE 802.15.4 networks. They discuss RTT measurements, and propose a device communication technology identification approach. Their work does not target the balance between overhead associated with active measurements and observability.

Active measurements have also been used for bandwidth estimation in IEEE 802.15.4 networks. For example, an enhanced approach considering MAC layer overhead for accurate estimation results is given in [19].

In [20] Steinert and Gillblad describe a method for adjusting the active-measurement frequency based on the stability of the measurement results. This technique mainly targets transport networks with more stable characteristics, but can also be applied in IoT systems to reduce overhead, for example in combination with the approach described in this paper.

In previous work we presented a first implementation and testbed evaluation of using standardized active measurements, based on TWAMP, for IoT devices running the Contiki OS [5]. Further, the paper also provided initial results on inference of active measurement results for localizing performance bottlenecks. This is an example of an approach for increased observability, often referred to as network tomography, which has been extensively investigated in wired networks, e.g. in [15][16], but is not well-understood for IoT systems.

This paper describes and evaluates a new approach aiming at reducing the overhead associated with observability in IoT systems, specifically using active measurements. A novel proxy-based approach using machine learning reduces the active measurement traffic. The machine-learning methods used by the proxy are briefly reviewed below.

Lasso regression is an interpretable linear machine learning approach performing both feature selection and regularization. It was introduced by Tibshirani [22], and mitigates overfitting with an additional regularization parameter, which tends to result in a model with smaller linear coefficients.

A decision tree differentiates from linear methods by iteratively partition the feature space to find the best mapping between features and targets. Examples of partitioning algorithms include ID3, C5.0, and CART. Random forest [23] is an ensemble learning method, trying to mitigate the decision tree overfitting problem, by constructing multiple decision trees in the training phase. The output of the model is based on targets from the individual trees.

Data used for machine-learning often contains redundant or irrelevant features [24]. This can result in longer training times, overfitting, and reduced prediction performance. Feature selection, either based on human domain knowledge or statistical methods, effectively overcomes many of these problems. In this paper we specifically use the univariate feature selection, along with manual feature selection.

In univariate feature selection, features are selected based on a univariate statistical test. In this paper an F-test is used to determine the linear effect from individual features on the target output. Then the K top-ranked features are selected. The manual feature selection is based on domain knowledge, which is described in more detail in Section VI.

III. Active-Measurement Proxy using Machine Learning for Reduced Overhead

This paper proposes an active-measurement proxy approach to overcome the overhead associated with measurements in IoT systems, see Sections I and II. The main functionality of the proxy is to intercept measurement packets, for example

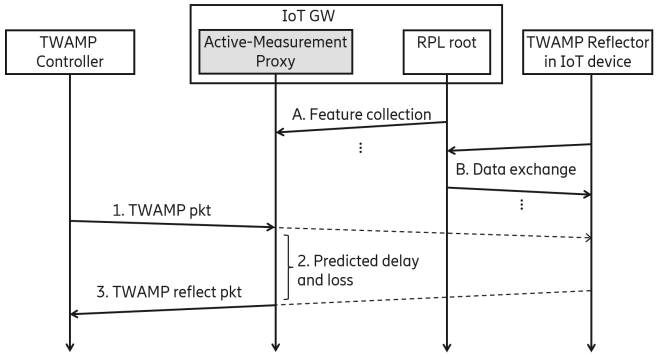

Fig. 3. Interaction between a TWAMP controller, the Active-Measurement Proxy residing in the IoT gateway, and IoT devices. Instead of forwarding the TWAMP packet, the proxy intercepts, predicts a delay or loss, and returns an emulated TWAMP reflect packet.

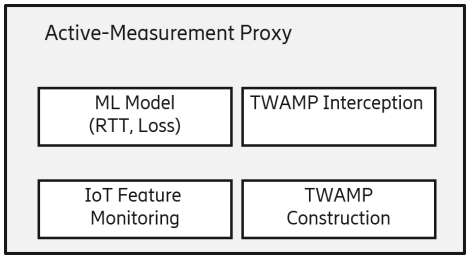

Fig. 4. The main components of the active-measurement proxy. This paper focus on the *ML Model* component.

based on TWAMP, sent from a controller towards a reflector located in an IoT device in the WSN. Figure 3 shows an example. A TWAMP packet is sent from the controller in step 1. The packet is intercepted by the proxy, instead of being forwarded to the reflector (as illustrated by the dashed line). The proxy predicts a delay (or loss) corresponding to the expected delay in the WSN, in step 2. Then, in step 3 the proxy returns an emulated TWAMP reflect packet. The delay is estimated using a machine-learning model, based on features extracted from the WSN, such as radio signal quality, logical and geographical topology, and sensor outputs. The machine-learned model may also predict a probability for packet loss for each packet. Packets are then dropped randomly based on this probability, which means that no reflect packet will be sent back by the proxy for those packets.

Section III-A formalizes the machine learning problem, which is part of the main contribution of this paper, while Section III-B outlines the proxy architecture.

A. Problem Setting and Modeling

Figure 1 shows the system under consideration. In an IoT system scenario a set of devices are connected via an IoT gateway to a network infrastructure. The devices are assumed to execute a service S, such as collecting data via a set of sensors, or affecting their environment through a set of actuators. The service operates under strict service-level agreements with requirements on network performance.

The network operator is interested in determining network performance, using active measurements, in terms of metrics such as RTT and packet loss, between two devices in the sensor network, a device and the gateway, or a device and a network node in the network infrastructure. However, as discussed in Section I and II, active measurements are associated with overhead in terms of network load and power consumption, and should therefore be limited if possible.

Hence, the goal with the approach presented in this paper is to accurately predict the WSN contribution to network performance metrics Y_t at time t for a path including for example the gateway and a specific device i, based on knowing all or a subset of features X_t. In terms of machine learning, the first problem is to find a model $M : X_t \rightarrow \hat{Y}_t$, such that $\hat{Y}_t$ closely approximates Y_t for a given X_t. The second problem is to find a small subset of features X_t to reduce costs of transferring feature data across the WSN. The model M can then be implemented in the proxy, which is described in the next subsection, and shown in Figure 4.

In this setting, features X^i for devices $i = 1..s$ refer to metrics related to characteristics of device i, including position, signal quality, sensor output, and routing data. The device statistics X is the concatenation of all device statistics, i.e. $X = [X^1, X^2, ..., X^s]$. The exact means of feature collection depends on the capabilities of the WSN. This work leverages testbed properties as discussed in Section IV. However, for a real scenario other mechanisms may be deployed, but this is outside the scope of this paper.

As stated earlier, the network performance metrics Y_t^i corresponds to the WSN contribution to RTT and packet loss, that is the performance between the gateway and a device i at time t, or between two devices i and j.

The metrics X and Y evolve over time, influenced by the network load, radio channel dynamics, the terrain, movements, etc. We model the evolution of the metrics X and Y as time series $\{X_t\}$, $\{Y_t\}$, and $\{(X_t, Y_t)\}$. Note that features X may not be collected exactly at the same time as targets Y, for example due to power saving strategies. In this case X is binned over a time interval containing the time stamp of Y_t.

B. Active-Measurement Proxy Architecture

The main components of the active-measurement proxy are shown in Figure 4 and an example of interaction between the TWAMP controller, the IoT gateway, and a reflector implemented in an IoT device, is illustrated in Figure 3.

The active-measurement proxy consists of four components. The *TWAMP Interception* component captures incoming TWAMP packets with a destination in the WSN. Instead of forwarding a measurement packet the component extracts data such as destination address and packet size. The *IoT feature monitoring* component collects data from the WSN, such as radio signal quality, topological data, and sensor/actuator results. This is shown as steps A and B in Figure 3. This paper describes a testbed-oriented approach for collecting features, see Section IV, but not a generic approach. The *ML Model* component learns several models M_Y that predicts e.g. Y_{RTT}

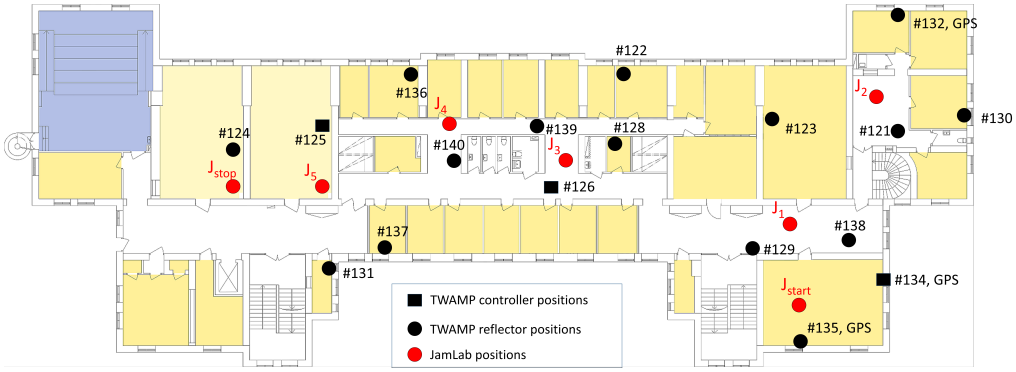

Fig. 5. Illustration of the placement of motes in the testbed, the TWAMP controller positions, and JamLab positions.

and Y_{loss} between the gateway and the destination device, using input from *IoT feature monitoring* and *TWAMP Interception* component. Finally, the *TWAMP Construction* component prepares and transmits a TWAMP reflection packet, with proper delay and time stamps, emulating a response from the IoT device at the destination address. If the *ML Model* component predicts a loss, the packet is immediately dropped.

Note that the RPL root, which is a functionality of the WSN routing protocol, can be co-located with the active-measurement proxy, as shown in Figure 3, and discussed further in Section IV.

Further note that the *ML Model* component may have to re-train the machine-learning models due to changes in the wireless environment. In this case new samples of X_t and Y_t are obtained by allowing a fraction of the TWAMP packets through at the gateway.

IV. Experimental Setup

A. Testbed

The EWSN'17 testbed [17] is used to evaluate the active-measurement proxy approach with specific focus on the *ML component*, discussed in Section III.B, and its predictive capabilities with respect to RTT and packet loss.

The testbed is located at Uppsala University, Sweden, and has 18 Tmote Sky Motes[1]. A mote (or device) is a tiny but complete execution environment including CPU, memory, radio, antenna, battery, and sensors for temperature, humidity, and light. The mote placement is shown in Figure 5. They reside in a building that constitutes a challenging environment for wireless communication due to thick stone walls and rebars. There is also an uncontrolled impact from faculty members and students, and their wireless devices. The motes communicate, for the experiments considered in this paper, over IPv6 enabled by IETF 6LoWPAN [12], running on top of IEEE 802.15.4. All motes in the testbed are configured to run the default Contiki Radio Duty Cycle and MAC protocols, which provides low-power communication and operation as described in IETF 6LoWPAN. Further, IETF RPL [13] is used for wireless routing, which enables a dynamic three-shaped routing topology relying on transient network performance. The root of the tree is denoted *RPL root*. Figure 5 also show positions for JamLab, further described below.

For the purpose of experiment administration and monitoring, separate monitoring modules, connected to a administrative central node over a fixed network, based on Raspberry Pi 3 are attached to each mote. The monitoring modules provide capabilities for easy and large-scale reprogramming of motes, power profiling, and high-precision event detection, without interfering with the motes nor the wireless medium during execution of experiments. They are connected to the motes via USB to enable logging and reprogramming. The monitoring modules are also connected to the mote via general purpose input/output (GPIO) pins available on the integrated circuit. The monitoring frequency is set to a default value of 62.5kHz, which enables event detection with 16 μs resolution. Two monitoring modules are connected to GPS time synchronized clocks, marked with "*GPS*" in the figure, enabling precise and accurate time stamping of GPIO events. This capability has been used to verify, with a positive outcome, the device time stamping precision and accuracy. The monitoring module also passively monitor voltage and current to calculate power consumption per mote.

B. TWAMP Active Measurements for Contiki

We implemented IETF TWAMP Light [3] for a Contiki 3.0 environment[2], the predominant OS for IoT devices, to enable active measurements between the motes. The implementation supports GPIO signaling upon packet receive and transmit events, which allows the monitoring modules to detect events occurring at the motes, and hence we can accurately calculate RTT using the monitoring-module clock, in addition to the RTT calculation using device clocks.

The same code runs on all motes, but executes along different paths depending on the mote identifier, that is whether the mote should act as a TWAMP controller or reflector. The TWAMP controller is placed on one of the motes $[125, 126, 134]$, see Figure 6. There can only be one controller

[1]http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf

[2]http://http://www.contiki-os.org/

TABLE I
SELECTION OF REPRESENTATIVE EXPERIMENTS AND PARAMETERS.

Exp. ID	TWAMP Ctrl.	Exec. Time	JamLab pos.
1	#125	144h	-
2	#126	144h	
3	#134	144h	-
4	#126	24h	J_{start}
5	#126	24h	J_{stop}
6	#126	7h	$J_{start} \rightarrow J_{stop}$

for a given experiment. Hence, there is always 1 controller and 17 reflectors in the testbed for a given experiment.

The testbed is configured to co-locate the TWAMP controller and the RPL root functionality to emulate the gateway between the WSN and the network infrastructure, as exemplified in Figure 1. That is, the testbed is used to execute TWAMP measurement experiments to generate data traces X_t and Y_t for the WSN only.

The TWAMP controller conducts measurements towards the reflectors in a round-robin fashion. The probe-packet interval, that is the time between two consecutive measurements for a controller-reflector pair, is set to 6 seconds to avoid measurement interference. The value is found through experimentation and is valid for this specific testbed. Further, for each controller-reflector pair the measurement is repeated 5 times (found experimentally) to enable result averaging.

The MTU for the network is 127 bytes, and to avoid packet fragmentation the probe-packet size is set to 100 bytes, including the TWAMP header and payload, and IPv6/MAC headers and preambles.

C. Radio interference

The testbed is mounted in a lively radio environment with several sources of noise and interference, as mentioned above. However, in order to create additional and partly controlled scenarios for radio interference we utilize *JamLab* [18], which is a system that can record and playback interference patterns as well as generate customizable and repeatable interference in real-time. JamLab is executed on a separate Tmote Sky device. Its position is changed manually.

Three scenarios with JamLab are considered; a JamLab mote at position J_{start}, at position J_{stop}, and moving the JamLab device from J_{start} to J_{stop} at discrete times, with intermediate positions J_i, see Figure 5. The JamLab device generates interference by emulating video streaming, at a high transmission power. No other traffic, except for TWAMP packets, is carried over the WSN testbed during the experiments.

We choose to inject noise only at the radio layer, mainly because we do not expect large volumes of cross traffic in a commercially deployed IoT network. Rather, we expect mainly controlled application traffic between devices and gateway.

D. Experiments

This paper reports on a *selection* of representative experiments, summarized in Table I, with the aim of evaluating the active-measurement proxy approach, with focus on the performance of the *ML Module*. TWAMP measurements are executed in a variety of scenarios to create X_t and Y_t time series for training of a model M.

Experiments 1-3 run for approximately 6 days each, and are executed without JamLab, and serve as a baseline. They consider three logical wireless topologies; two multi-hop trees with TWAMP controllers co-located with the RPL root on motes 125 (to the left in Figure 5) or 134 (to the right), and one star-like topology with the TWAMP controller and RPL root at mote 126 (in the middle of Figure 5).

In experiment 4 and 5 the JamLab device is stationary. Now the TWAMP controller and RPL root resides on mote 126, since such star-like topologies are common in industrial settings. These experiments runs only for 24h each, to limit disturbance from JamLab in the building hosting the testbed.

The last experiment, with identifier 6, considers a scenario where the JamLab device moves from position J_{start} to J_{stop} via the intermediate positions shown in Figure 5. The JamLab device stays at each position for 1h. This experiment lasts for 7h, also in order to limit disturbances.

V. DEVICE FEATURES X AND TARGETS Y

Data traces in terms of features X and targets Y are generated through a set of experiments using the testbed, as described in Section IV. This section describes and explores the features and targets in detail.

A. Device Features X

All features X are described below, and summarized in Table II for completeness. Note that the features are collected using the Raspberry Pi 3 monitoring modules, as described in Section IV, which eliminates any impact on radio network performance, in terms of RTT and loss.

The first category of features describe the current state of the devices in the WSN, their position and configuration, time, and radio signal quality. They are described below.

- Geographical coordinates (x_{con}, y_{con}) of the TWAMP controller node. The features are approximated by manual inspection of a map.
- Geographical coordinates (x_{ref}, y_{ref}) of a TWAMP reflector node. The features are approximated by manual inspection of a map.
- Euclidean distance D between the TWAMP controller and a TWAMP reflector node, calculated using respective coordinates.
- Time of day T is a categorical feature with a granularity of 4 hours, which is considered enough for the purpose of this paper. This feature is engineered from timestamps in each TWAMP packet.
- Weekday is defined as a categorical feature $W \in [0, 6]$, and is derived from the same timestamps.

TABLE II
FEATURES X COLLECTED FROM THE TESTBED.

Features	Description
x_{con}, y_{con}	Position of the controller
x_{ref}, y_{ref}	Position of each reflector
D	Distance (m) between controller and each reflector
T	Time of day, a categorical feature $\in [0, 5]$
W	Weekday, a categorical feature $\in [0, 6]$
IP_{con}, IP_{ref}	IPv6 addresses for controller and reflector
R_{CH}	Radio Channel, an ordinal feature $\in [1, 16]$
R_{LQI}	LQI measured at the controller
R_{RSSI}	RSSI measured at the controller
$r_1, ..., r_{k-1}, r_k$	RPL rank for k motes
$n_1, ..., n_{k-1}, n_k$	Number of neighbors for k motes
$L_1, ..., L_{k-1}, L_k$	Light radiation registered by each mote

- IP_{con} and IP_{ref} corresponds to the IP addresses of the TWAMP controller and reflector.
- Wireless transmission channel, R_{CH}, corresponding to one of the 16 available channels defined in the IEEE 802.15.4, 2.4GHz band.
- Received Signal Strength Indicator, R_{RSSI}, measured at the TWAMP controller. This metric is a measurement of power present in a received radio signal, and gives an indication of the radio environment at the gateway.
- Link Quality Indicator, R_{LQI}, measured at the TWAMP controller. This is a metric of the current quality of the received signal. That is, it provides an estimate of the difficulty of demodulation, and gives an additional indication of the radio environment at the gateway.

Further, topological information is collected during the experiments. As described in Section IV the network uses RPL for routing. Each node maintains a neighbor list with the information about the number of neighbors, the rank value of each neighbor, as well as the rank value of the node itself. The following features are extracted.

- Rank r_i for a device i is a topological property of the device that depends on current radio conditions and its logical position. The rank determines how packets are routed through the WSN.
- Number of neighbors n_i for a mote i corresponds to the number of motes within radio reach for that specific device.

In addition to the above features the testbed also collects light radiation statistics L_i from each mote i. It is sampled for every TWAMP measurement.

B. Targets Y

The target metrics Y corresponds to performance metrics observed between the TWAMP controller, residing at the gateway, and a TWAMP reflector on an IoT device. The target metrics are obtained through active measurements. This paper specifically considers two targets, namely round-trip time Y_{RTT} and packet loss Y_{loss}.

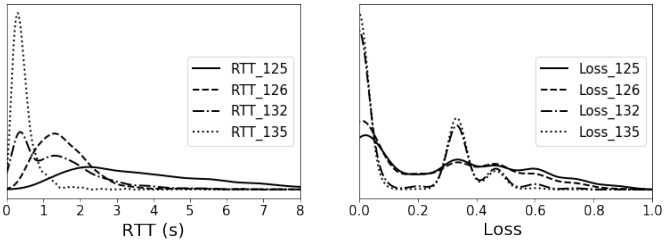

Fig. 6. Representative examples of RTT and loss density functions. The graphs are generated from experiment 3 for a selection of motes.

Representative density functions for Y_{RTT} and Y_{loss} are illustrated in Figure 7. The plots correspond to experiment 3, where the TWAMP controller is located at mote 134. For the purpose of illustration the figure only display density functions for a selection of motes; where mote 135 is closest, 132 and 126 on similar distances but different positions, while mote 125 represents a greater controller-reflector distance. The density functions are generated using the kernel density estimation function in Scikit-Learn.

It is striking how both Y_{RTT} and Y_{loss} show completely different characteristics compared to traditional wired networks. For example, RTT samples above 1 seconds are very common, for most motes. Further note that the density function average value and variance seems to increase with the distance between the TWAMP controller and reflector, which is also expected.

Note that Y_{loss} is defined as the fraction of lost TWAMP measurement packets during a 20 minute time interval, which is supported by industry standards such as ITU-T Recommendation Y.1540. This results in maximum 200 samples (X_t, Y_t) per 20 minute interval, assuming a 6 second interval between measurements, as described in Section IV. The 20 minute interval is needed for collecting enough samples per device for accurate performance modeling.

The other experiments show similar distributions for the target metrics, however they are omitted due to page limitations.

VI. EVALUATION

The evaluation in this paper focuses on the performance of several machine-learning models for RTT and loss given data from the experiments described in Section IV.

The evaluation metric reported in this paper is the *Normalized Mean Absolute Error*, defined as $NMAE = \frac{1}{\bar{y}}(\frac{1}{m}\sum_{t=1}^{m}|y_t - \hat{y}_t|)$, where $\hat{y}_t$ is the model prediction for the measured performance metric y_t, and $\bar{y}$ is the average of the samples y_t of the test set of size m. The train and test split is 90/10. Further, the standard deviation is also reported for each model. Other evaluation metrics such as *MAE*, *MSE*, and *NMSE* showed similar results. Note that each model is tuned with respect to its hyper-parameters to lower the NMAE.

Each model is evaluated using three feature sets. X_{All} is self-explanatory and contains all features discussed in Section V-A. Further, the set X_{uni} contains features selected from

TABLE III
PREDICTION RESULTS FOR RTT AND LOSS MODELS IN SCENARIOS WITHOUT JAMLAB.

Data source	Features	RTT				Loss			
		RF NMAE (%)	σ	LR NMAE (%)	σ	RF NMAE (%)	σ	LR NMAE (%)	σ
Experiment #1, TWAMP Controller at mote 125	X_{All}	10.5	0.661	44.5	1.15	17.7	0.795	86.6	2.64
	X_{uni}	9.88	0.853	46.2	1.47	15.8	1.08	88.4	1.48
	X_{Rank}	6.87	0.464	66.8	1.42	13.1	0.781	98.5	2.68
Experiment #2, TWAMP Controller at mote 126	X_{All}	7.89	1.24	46.8	1.37	22.5	0.804	102	8.23
	X_{uni}	5.93	1.19	46.7	1.69	16.9	2.30	130	6.15
	X_{Rank}	6.52	0.428	53.5	1.40	16.8	2.29	136	7.98
Experiment #3, TWAMP Controller at mote 134	X_{All}	8.49	0.288	45.4	1.10	14.9	0.469	82.3	1.44
	X_{uni}	8.16	0.323	47.3	0.653	15.2	0.958	83.0	1.65
	X_{Rank}	6.78	0.289	71.1	1.31	12.4	0.674	100	1.82

using univariate feature selection. A study of feature importance shows that rank, Euclidean distance, and geographical coordinates are dominating. Finally, the X_{Rank} feature set contains r_i, for all i.

Scenarios without JamLab are discussed in Section VI.A while Section VI.B. shows the impact from JamLab. The scenario configurations are given in Table I.

A. Prediction results for scenarios without JamLab

This section discusses model accuracy for RTT and packet loss in scenarios without a JamLab device, corresponding to experiments 1 - 3 in Table I. That is, we would like to determine the NMAE for a model $M : X_t \rightarrow Y_t$, for RTT and packet loss, respectively.

Results for prediction of Y_{RTT} are located in the left part of Table III[3]. The table shows three main rows corresponding to experiments 1 - 3. Two machine-learning approaches are used to build the prediction models, namely lasso regression (LR) and random forest (RF), briefly reviewed in Section II. Each model is trained and evaluated using 3 different feature sets based on X, see above.

The results for RTT show a set of interesting phenomenons. First, the NMAE for the linear models range from 44.5% to 71.1% and thus they can not capture the mapping between X_t and Y_{RTT}. Random forest on the other hand has lower NMAE across all the scenarios. Lowest NMAE is obtained using the rank features X_{Rank} in experiment 1 and 3, while the univariate feature set X_{uni} produces an NMAE of 5.93% in experiment 2. Most likely, the additional features in X_{All} introduce noise that reduce model performance. Finally, note that the models are capable of predicting RTT with low NMAE irrespectively of the logical network topology, given by the location of the TWAMP controller/RPL root.

The results for modeling Y_{loss}, over a 20 minute time interval, are shown in the right part of Table III. Similarly to RTT, the linear-model approach is not complex enough to capture the mapping between X_t and Y_{loss}. Random forest models prove useful again, with lowest NMAE using the X_{Rank} feature set, ranging from 12.4% to 16.8%.

Generally, the models for loss have a higher NMAE compared to RTT. Averaging packet loss over longer time intervals, for example 60 minutes, reduces the NMAE to approximately 7%, which is comparable to the RTT models. That is, more samples per loss interval increases the model performance. However, the loss interval, measurement frequency, and overhead needs to be balanced for a specific deployment scenario.

It is clear from the results above that linear models do not perform well. An explanation for this is the observed low linear correlation between target Y and X_{Rank}, which is the dominating feature set in most experiments. The rank describes the topological structure and performance of the wireless network and is defined as the accumulated sum, over a number of hops, of the inverse of the link-loss probability. Hence, it can be assumed that Y_{loss} is a non-linear function of the dominating X_{Rank}. Further, since the link-loss probability reflects the channel noise, which can increase the transmission time in 802.15.4 networks, it is also assumed that Y_{RTT} is a non-linear function of rank.

B. Prediction results for scenarios with JamLab

This section reports on prediction accuracy for RTT and loss models in scenarios with a JamLab device in different positions in the testbed, corresponding to experiments 4 - 6 in Table I. The question is identical - what is the NMAE for a model $M : X_t \rightarrow Y_t$, for RTT and packet loss, respectively.

In the previous subsection it became evident that a linear approach does not capture the mapping between X_t and Y_t. Hence, this subsection only present results from using random forest. Table IV shows results from modeling RTT and loss in three different JamLab scenarios, two with a stationary JamLab device and one where it moves.

For the two stationary JamLab scenarios it is clear that the lowest RTT model NMAE values are obtained using using univariate feature selection; ranging from 7.85% in experiment 2 up to 8.65% in experiment 1. The NMAE increases to

[3]Note that the target Y_{RTT} is averaged over 5 measurements in order to reduce noise. Without such averaging the NMAE, using random forest, is approximately 60% across all scenarios and experiments.

TABLE IV
PREDICTION RESULTS FOR RTT AND LOSS MODELS FOR SCENARIOS WITH JAMLAB.

Data source	Features	RTT		Loss	
		RF NMAE (%)	σ	RF NMAE (%)	σ
Experiment #4, TWAMP Controller at mote 126, JamLab at J_{start}	X_{All}	11.9	3.85	16.5	2.65
	X_{uni}	8.65	3.64	12.4	2.49
	X_{Rank}	9.63	3.60	13.4	4.35
Experiment #5, TWAMP Controller at mote 126, JamLab at J_{stop}	X_{All}	9.84	1.12	19.1	3.83
	X_{uni}	7.85	1.97	15.8	2.35
	X_{Rank}	8.78	1.90	16.2	3.92
Experiment #6, TWAMP Controller at mote 126, moving JamLab	X_{All}	15.2	2.94	16.6	3.63
	X_{uni}	13.3	2.31	16.5	6.49
	X_{Rank}	15.0	2.35	18.6	3.50

approximately 13% in the moving JamLab scenario, for the univariate feature set. Scenario complexity is the main reason for increased NMAE for RTT models, compared to the results in Table III. An additional reason is the lower number of samples in all JamLab experiments, which punishes the model performance. This is even more evident in the results from experiment 6. This corroborates with the well-known fact that the number of samples is correlated with model performance.

Table IV also shows corresponding results for modeling of Y_{loss}. Again, the NMAE is higher for loss prediction compared to RTT model results. For example, the NMAE for experiment 4, with JamLab at position J_{start}, for a loss model using univariate feature selection is 12.4% compared to 8.65% for the Y_{RTT} model.

The most challenging scenario for loss prediction is experiment 6, with a moving JamLab device, where the NMAE is 16.5% using the X_{uni} feature set. Averaging over longer time intervals decreases the NMAE as discussed above, while also decreasing the utility of the model.

It is clear from the results above that the feature set X_{Rank} provides the lowest NMAE for scenarios without JamLab, and X_{uni} the lowest NMAE with JamLab activated. An investigation reveals that features such as position (including $x_{con}, y_{con}, x_{ref}, y_{ref}$, and D) and time T increases in importance in the JamLab scenarios. These features get included in the univariate feature selection approach and hence reduces the NMAE. Most likely, these additional features provide information regarding the JamLab position and its behavior.

Finally a note on standard deviation for the JamLab scenarios, which is generally higher in Table IV compared to Table III, due to the increased complexity when JamLab is activated.

VII. CONCLUSIONS AND FUTURE WORK

This paper presented a novel active-measurement proxy approach, based on machine learning, aiming at reducing the overhead associated with active measurements in IoT systems, with focus on wireless IEEE 802.15.4 networks. The main functionality of the proxy is to capture each measurement packet, and instead of forwarding it the proxy emulates a measurement-packet response, and sends it back after a delay corresponding to an estimated delay in the wireless network. The delay is estimated using machine learning and features such as radio quality, distance, and routing properties.

The paper described a Contiki OS implementation of the IETF TWAMP standard for active measurements, which was deployed in a real-world testbed to generate data traces in a variety of scenarios. Machine-learning models for estimating the delay and loss, originating from the wireless network, were then developed and evaluated.

The paper shows that the proxy approach is applicable in all considered scenarios. The RTT and loss were predicted with a normalized mean absolute error close to 7% and 14%, respectively. The most complex scenario, with added interference from a moving jammer device, increased the error for RTT prediction to approximately 13%, whereas loss prediction only shows a slight increase to 16%.

Furthermore, random forest was proven to be superior over linear machine-learning models due to non-linearity in the dominating features. A study of feature importance indicated that routing data is most valuable for high-performing models. In more complex scenarios even lower errors were achieved when including device position and time-related features.

Future work includes implementation and evaluation of the whole proxy functionality, and development of new models for improvement of predictive capabilities. Further, new lightweight mechanisms for feature collection in the wireless network will be studied.

ACKNOWLEDGMENT

We would like to acknowledge and thank Christian Rohner, Thiemo Voigt, and Per Gunningberg for hosting the experiments at Uppsala University. Further, we would like to thank Carlo Alberto Boano and Markus Schuss, both at TU Graz, for their endless testbed support, and Tim Josefsson for supporting the initial implementation of TWAMP for Contiki.

This research is sponsored by the Swedish Foundation for Strategic Research under grant SM15-0026.

REFERENCES

[1] L. D. Xu, et al, "Internet of things in industries: A survey." IEEE Transactions on Industrial Informatics 10.4, 2014.

[2] J.Postel, "Internet Control Message Protocol", IETF RFC 792, 1982.

[3] K. Hedayat, et al, "A Two-Way Active Measurement Protocol (TWAMP)", IETF RFC 5357, 2008.

[4] S. Baillargeon, et al, "Ericsson Two-Way Active Measurement Protocol (TWAMP) Value-Added Octets ", IETF RFC 6802, 2012.

[5] A. Johnsson, C. Rohner, "On Performance Observability in IoT Systems using Active Measurements", IFIP/IEEE Network Operations and Management Symposium (NOMS), 2018.

[6] A. Zanella, et al, "Internet of Things for Smart Cities", IEEE Internet of Things Journal, Vol 1, Issue 1, 2014.

[7] V. Pereira, "Performance Measurement in Wireless Sensor Networks", PhD Thesis, University of Coimbra, 2016.

[8] K. Wolosz, et al, "A Measurement Study for Predicting Throughput from LQI and RSSI", In Multiple Access Communications (MACOM), Lecture Notes in Computer Science, Springer, 2012.

[9] R. N. Duche, N. P. Sarwade, "Round trip delay time as a linear function of distance between the sensor nodes in wireless sensor network", Journal of Engineering Sciences and Emerging Technologies, 2012.

[10] CW Tsai, et al, "Data Mining for Internet of Things: A Survey", IEEE Communications Surveys and Tutorials, Vol. 16, 2014.

[11] B.U. Ranjini, T. Ravichandran, "Survey on Fault Localization Techniques in Wireless Sensor Networks", International Journal of Advanced Research in Computer Science Engineering and Information Technology, Volume: 3 Issue: 1, 2014.

[12] G. Montenegro, et al, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", IETF RFC 4944, 2007.

[13] T. Winter, et al, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", IETF RFC 6550, 2012.

[14] L. Metongnon, et al, "Efficient probing of heterogeneous IoT networks", In Proceedings to the IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017.

[15] A. Johnsson, et al, "Online network performance degradation localization using probabilistic inference and change detection", IFIP/IEEE Network Operations and Management Symposium (NOMS), 2014.

[16] A. Johnsson, C. Meirosu, "Towards automatic network fault localization in real time using probabilistic inference", IFIP/IEEE Integrated Network Management (IM), 2013.

[17] M. Schuss, et al, "A Competition to Push the Dependability of Low-Power Wireless Protocols to the Edge", In Proceedings of the International Conference on Embedded Wireless Systems and Networks, 2017.

[18] C. Boano, et al, "JamLab: Augmenting sensornet testbeds with realistic and controlled interference generation", In Proceedings to the International Conference on Information Processing in Sensor Networks (IPSN), 2011.

[19] M. O. Farooq, T. Kunz, "Proactive Bandwidth Estimation for IEEE 802.15.4-Based Networks", In IEEE Vehicular Technology Conference (VTC Spring), 2013.

[20] R. Steinert, D. Gillblad, "Performance evaluation of a distributed and probabilistic network monitoring approach", In Proceedings of the International Conference on Network and Service Management (CNSM), 2012.

[21] Operation, administration and maintenance (OAM) functions and mechanisms for Ethernet-based networks, ITU-T Y.1730, 2018.

[22] R. Tibshirani, "Regression Shrinkage and Selection Via the Lasso", In Journal of the Royal Statistical Society, Series B, 1994.

[23] L. Breiman, "Random Forests", In "Machine Learning", 2001.

[24] T. Hastie, et al, "The Elements of Statistical Learning", Springer Series in Statistics, 2001.