The CityLab Testbed - Large-scale Multi-technology Wireless Experimentation in a City Environment: Neural Network-based Interference Prediction in a Smart City

Jakob Struye, Bart Braem, Steven Latré, Johann Marquez-Barja
IDLab - University of Antwerp - imec
Middelheimlaan 1, 2020 Antwerp, Belgium, firstname.lastname@uantwerpen.be

***Abstract*—Smart cities form an important new paradigm for future cities, where technology assists people, local economy and government. For smart cities to mature, it is crucial to enable experimental evaluation of current and new technologies, in realistic conditions. This paper contributes to the network testbeds domain by introducing the CityLab testbed, where researchers can experiment with a variety of smart city network technologies in parallel, including IEEE 802.11, IEEE 802.15.4, and sub-GHz protocols —on bare metal hardware enabling full software flexibility. As a second contribution, one aspect of realism, interference, is shown to be measured and predicted, based on data from the CityLab deployment. Specifically, we predict interference one hour into the future using a neural network based on a Gated Recurrent Unit. Compared to a naive predictor, the neural network is over 6.5 times as accurate.**

I. Introduction

The arrival of the Internet of Things (IoT) is changing the way technology interacts with daily life. In a broad range of application domains, from mobility to health, IoT is having a significant impact. All of these IoT application domains come together in the smart city paradigm, where technology supports all actors to improve life in the city and its citizens. This new paradigm is gaining traction all over the world, but is still in need of further evaluation. Testbeds provide an excellent means to validate Research & Development (R&D) results in realistic conditions. In this paper we describe the implementation of a highly flexible smart cities testbed, CityLab, which offers the experimenters, through standard experimentation tools, bare metal access to multi-technology nodes located within the streets of a city. Then we demonstrate the usefulness and potential of data gathered, by accurately predicting background noise.

A number of testbeds for smart city network experimentation already exist, we discuss the most relevant ones in what follows. SmartSantander [1] in Santander, Spain is one of the most well-known testbeds in Europe, which provides its users with data sets from a vast amount of sensors. The Ubiquitous Oulu Smart City testbed [2] in Oulu, Finland is focused on interaction between citizens and government. Google's Sidewalk Labs [3] in Toronto, Canada studies the application side of smart cities. Bristol is Open [4] in Bristol, UK offers 5G and optical network experimentation.

CityLab puts forward realism, a multi-technology approach and high flexibility, which differentiates itself from these existing initiatives. Presenting the main architecture and capabilities of CityLab is the main contribution of this work. As a second goal, leveraging the capabilities offered by CityLab, this paper attempts to characterize interference in a smart city. Unexpected interference experienced by CityLab nodes can be detrimental to the quality and usability of experiment results. Predicting interference based on previous measurements could therefore improve the value of CityLab experimentation. One indicator of interference is the Received Signal Strength Indication (RSSI), which can be converted to signal strength. Several papers have presented methods to predict these RSSI values. As RSSI values can vary greatly between subsequent measurements, many other methods first apply some sort of smoothing to the raw results. Long and Sikdar predict interference using a linear regression model on smoothed data [5]. This method applies an adaptive window mechanism, meaning the lookback window size is increased when recent RSSI measurements are relatively stable, and decreased in case of sudden changes. Chin et al. smooth the data by modeling it using an Ornstein-Uhlenbeck diffusion process, noting that RSSI values often exhibit mean-reverting behavior [6]. They then also apply linear regression.

While many RSSI prediction methods have been developed, all focus on short-term prediction. Predicting the interference at most a few seconds into the future is mainly useful to orchestrate channel hopping. We instead aim at a more long-term general prediction of interference. Specifically, we predict the smoothed interference one hour into the future in order to provide important information to experimenters, and to make correct assumptions during the current and future experimentation. We achieve this through a neural network using a Gated Recurrent Unit (GRU), which excels at predicting temporal data. We focus on RSSI measurements with high seasonality, specifically higher interference during daytime and lower interference during nighttime. We show

that, after careful tuning, our network can learn and predict these values with reasonably high accuracy and a low runtime.

The remainder of this work is structured as follows. Section II lists the requirements to a smart city testbed, followed by Section III which describes how CityLab implements those requirements. Section IV then describes the setup of an experiment on CityLab, followed by the results in Section V. Finally, Section VI concludes this paper.

II. Smart City Testbed Requirements

Based on the requirements from SmartSantander, in what follows we identify key requirements for a smart city testbed which enables low-level network experimentation.

A. Experimentation Realism

A wide range of testbeds for wireless experimentation already exists, often in a controlled environment. Such infrastructures form a perfect next step after simulations, however they still suffer from limited realism, e.g., background noise needs to be added artificially. In order to fully analyze research results, we strongly advocate the need for the next step: the use of testbeds deployed in realistic locations. For a smart city testbed, such a step implies the need to deploy experimental hardware outdoors, in the city itself. Moreover, in a smart city, the interaction between user and technology is key to realizing compelling solutions.

B. Reliability

A testbed should offer a high reliability to experimenters, enabling full control of their experiments. Moreover, the experiments require to be remotely manageable to lower the operational overhead for testbed operators and to be able to leverage federation with other testbeds. This implies the need for remote node recovery mechanisms, where on-site maintenance is limited to a minimum. On a network level, this translates to the need for continuous high-speed low-latency access to the testbed nodes.

C. Multi-technology

Smart cities are a relatively young IoT application domain. In this respect, the choice of the one dominating wireless technology or technologies is not yet clear, compared to e.g. smart homes where IEEE 802.11 and Bluetooth can be arguably be considered dominant network choices. Therefore, it can be argued that smart cities testbeds should start from a multi-technology approach and even should be open to new technologies when they arise, rather than picking one technology upfront. In contrast, in the pioneering SmartSantander testbed only ZigBee and Bluetooth are deployed.

III. CityLab

A. Overall Architecture

To realize a smart city testbed which fulfills the requirements outlined above, the CityLab testbed is based on the architecture shown in Figure 1. Researchers use jFED [7]

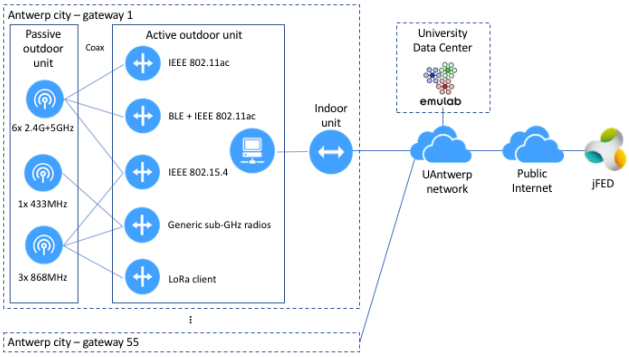

Fig. 1. CityLab architecture overview.

Fig. 2. CityLab gateway components, at the top the passive and active outdoor units, at the bottom the indoor unit.

to access an EmuLab-based experiment management system [8], located in a data center of the University of Antwerp / imec, to control gateways distributed in the city of Antwerp, Belgium. The experiment management system is connected to the gateways over an academic fiber network, with each gateway connected over 100Mbit Ethernet and most gateways over Gigabit Ethernet.

B. Gateways

The CityLab testbed nodes are called gateways, which are composed of three components: an indoor unit, an active outdoor unit and a passive outdoor unit, as shown in Figure 2.

Using jFED, the experimenter gets access to the outdoor unit, where an X86 PCEngines APU embedded device allows the experimenter to run bare metal Linux booting over Preboot eXecution Environment (PXE). The embedded system is then connected to multiple radios, supporting a variety of network technologies as summarized in Table I. To enable flexible addition of new and arising network technologies, the outdoor unit also relies strongly on an embedded USB hub, which allows connecting additional network technologies easily.

TABLE I
CITYLAB GATEWAY RADIOS AND ANTENNAS

Network Technology	Antennas and Bands
IEEE 802.11ac (2.4GHz+5GHz, client+AP)	3x 2.4+5GHz
IEEE 802.11ac (2.4GHz+5GHz, client) + Bluetooth Low Energy	2x 2.4+5GHz
IEEE 802.15.4 at 2.4GHz and 868MHz	1x 433MHz + 1x 868MHz
Sub-GHz protocols e.g., DASH-7, IEEE 802.15.4g	1x 433MHz + 1x 868MHz
Single-channel LoRa client	1x 868MHz

Each radio is connected to dedicated to antenna(s) enclosed in what is called a passive outdoor unit, as shown in Table I.

The indoor unit then connects the outdoor units to the academic network over two Power Over Ethernet+ cables. As these cables provide both power and data, a simple power switching component in the indoor unit allows rebooting the outdoor unit and recovering it in (almost) any situation.

C. Experiment Management

The gateways are managed by an EmuLab installation which is available through jFED over a Slice-based Federation Architecture (SFA), building on the foundations of the Fed4FIRE+ EU project [9]. In a traditional lab testbed setup, EmuLab expects all devices to be directly reachable via a single managed switch to enable imaging and (PXE) booting or rebooting of systems. In the case of CityLab, this is unfeasible given the distributed deployment of CityLab gateways. These are deployed in different streets, connected to different subnets of the university network. To tackle this, the indoor units transparently tunnel the outdoor units to a single logical subnet in the gateway management network. With this approach, to the existing EmuLab systems, the outdoor units transparently appear local and all existing tools can be leveraged.

D. Deployment

The CityLab gateways are deployed in the City Campus neighborhood of the city of Antwerp, the second largest city in Belgium. As shown in Figure 3, about 30 gateways are actively deployed, with 20 being installed in the coming months. Moreover, a *smart zone* is being designed, where a dense sensor deployment will be supported by 20 more gateways. Figure 3 also shows an example deployments of a CityLab gateway in the streets of Antwerp.

CityLab focuses on network experimentation to increase maturity of smart city connectivity, while in parallel user involvement should be evaluated through Living Labs. Therefore, our smart cities testbed is also part of a bigger program [10] where smart cities use cases are deployed and evaluated, based on more traditional network technology. Mobility has not been taken into account yet, because of the initial focus on the static infrastructure. First trials on small-scale use cases with mobile data gathering have been started, based on stable LPWAN technology.

To demonstrate the realism and capabilities of CityLab, we started from the observation that interference is an important characteristic of realism. Therefore, in the following sections we describe an experiment where we characterize and even predict the interference.

Fig. 3. Map of CityLab gateway locations and example deployment, center right in the picture.

IV. USE CASE: PREDICTING CROSS-TECHNOLOGY INTERFERENCE

A. Description

As a use case of the CityLab testbed, we measure and ultimately predict the background noise the nodes experience. Such knowledge can help us in planning future experiments and in interpreting their results. We first measure background noise per 2.4 GHz Wi-Fi channel once per minute over the period of three weeks. After processing the raw data, we train and test a neural network to predict the intensity of the noise. As we measure the interference in a technology-agnostic way, our results are useful for experiments using any technology in the 2.4 GHz bands.

B. Experimental setup

1) Measurements: We used eleven CityLab nodes at different locations to measure background noise. These nodes are equipped with a COMPEX WLE900VX-7A network adapter, which contains a Qualcomm Atheros QCA9880 wireless chipset. As with all 802.11ac and 802.11n chipsets by Qualcomm Atheros, this chipset supports a mode called *spectral scan*. In this mode the chipset passively measures radio activity on a channel (20 MHz wide by default). As the chipset will measure any activity, as opposed to just IEEE 802.11 traffic, it in essence functions as a simple spectrum analyzer. The feature has been incorporated into the open source `ath10k` driver. While the driver does not (yet) support a mode sweeping all supported channels, this behaviour can be achieved by running the chipset in `background` mode and then performing an `iw scan` across all frequencies. The latter listens for access point beacons on all provided frequencies while the former scans the current channel.

Spectral scan mode reports the noise floor and the measured RSSI. While the calculation of the RSSI is not standardized, Qualcomm Atheros is known to use signal strength in dBm minus the noise floor, meaning we can easily derive the signal strength from the reported values.

2) Data Processing: To extract signal strength from the raw data dumps created by the chipset, we use the FFT_eval tool by Simon Wunderlich[1]. While intended to visualize the signal strengths across multiple channels at one point in time, we modified it to instead process temporal data for one channel. Across multiple measurements we noticed a high variability in measured signal strengths. Occasionally the measured signal strengths were unrealistically high, reaching 129 dBm, which is almost $8\,\mathrm{GW}$. After trial and error analysis, we decided to discard all samples over $0\,\mathrm{dBm}$ and then take the fifth highest remaining sample to avoid taking these outliers into account. Even then, the data is still highly variable and thus impossible to predict with reasonable accuracy. We therefore smooth the data using the Savitzky-Golay filter [11], which has been applied to RSSI values before [12] [13]. For every point k, the filter fits a polynomial $p(x)$ to all points in a window centered around k, calculating $p(k)$ as the smoothed value [14]. The window size and polynomial degree are the two main parameters of the algorithm. After manual experimentation, we set these to 1101 and 2. With these parameters, the filter removes minute-to-minute variations, revealing any daily patterns.

3) Neural Network Design: Next to analyzing activity in the radio spectrum, we also want to predict it. Specifically, we want to predict the activity one hour into the future, based on previous measurements. One system particularly fit for such tasks is Recurrent Neural Networks (RNNs). These neural networks can recognize temporal patterns by remembering aspects of past inputs in their internal memory. Classical RNNs are very susceptible to the vanishing and exploding gradient problems. Other implementations of the RNN concept, such as the GRU [15], manage to avoid these problems, while performing even better than classical RNNs [16]. A GRU is based on two types of gates: the update gates and the reset gates. Each hidden unit inside a GRU retains a state. When new data is fed to the GRU, the reset gates decide which aspects of the state to forget, while the update gates decide how to incorporate the new data into the state. The exact behaviour of these gates is decided by their weights. Such networks are trained by repeatedly feeding inputs, each time adjusting the weights to bring the network's actual output closer to the known correct output. While it is possible to stack multiple GRU layers, this did not improve performance for our application. As such, our network consists of only one GRU layer, followed by a fully connected layer which reduces the outputs to a single value.

[1] https://github.com/simonwunderlich/FFT_eval

4) Neural Network Tuning: The behaviour and structure of a GRU layer are further decided by a number of hyperparameters, which we must tune to approach the layer's optimal performance. We tuned the following hyperparameters:

1) `epochs`: The number of times the entire training set is fed to the system during the training phase. A higher value may lead to more accurate results at the cost of a higher learning time. However training for too long may worsen the network's predictive capabilities [17].
2) `units`: The number of units in the GRU layer. Increasing the number of units may allow the layer to learn more complicated patterns, at the cost of additional runtime. On the other hand setting this too high may again have a negative effect [17].
3) `lr`: The learning rate. This controls how significantly the weights are adjusted after every epoch. Through a higher learning rate the optimal values of the weights can be approached faster, however they can also be 'overshot' more easily [18].
4) `lookback`: Instead of only feeding the latest measurement to the network at each step, we feed the latest `lookback` measurements. Increasing the lookback beyond 1 usually improves performance up to a point, at the cost of additional runtime.
5) `retrain_interval`: After `retrain_interval` predictions, we retrain the network on the most recent data. This way, the network will adapt to changing patterns in the data.

The network was trained using the Adam optimizer [19], optimizing the Mean Absolute Error (MAE).

As input we feed the network the set of most recent measurements. During training, the expected result is the measurement sixty data points (i.e., one hour) into the future. We initially train the network on the first three days worth of data, followed by occasional retrains on equally sized but more recent data sets. We normalize the data of the first three days by subtracting the mean, and dividing it by the standard deviation. Future data is also normalized using the mean and standard deviation of only the first three days, meaning normalization is consistent across the entire data set without having future data points influence the normalization. As long as the range of future data points does not differ significantly from the first three days, this should not have a negative effect on prediction accuracy. The full procedure is summarized in Algorithm 1.

To evaluate the final result we use the Mean Absolute Scaled Error (MASE), which expresses the network's performance relative to a simple baseline prediction [20]. As the data set is highly seasonal (high activity during daytime, low activity during nighttime), we use the (smoothed) RSSI measured exactly 24 hours prior to the value to be predicted as a baseline. A MASE of 1 would indicate the network's MAE is equal to the baseline prediction's, while a MASE under 1 indicates better performance.

Algorithm 1 Neural network training/testing

function GETINPUT(*data*,*index*)
  input $\leftarrow data[index - lookback, index]$
function GETOUTPUT(*data*,*index*)
  output $\leftarrow data[index + predictionStep]$
$data \leftarrow$ LOADDATA()
REMOVEOUTLIERS($data$)
APPLYSAVITZKYGOLAY($data$)
NORMALIZE($data$)
$net \leftarrow$ NEURALNET(GRU, FULLYCONNECTED)
TRAIN($net, data[, trainSize]$)
for *index* **in** [trainCount, SIZE(*data*)] **do**
  result $\leftarrow$ PREDICT(net, GETINPUT($data, index$))
  expectedResult $\leftarrow$ GETOUTPUT($data, index$)
  if TIMETORETRAIN( ) **then**
    TRAIN($net, data[index - trainSize, index]$)

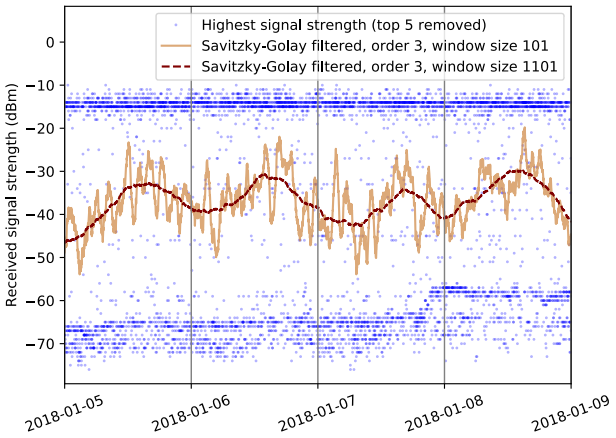

Fig. 4. The smoothed data.

TABLE II
TUNED HYPERPARAMETERS

`epochs`	85
`units`	53
`lr`	0.00077
`lookback`	52
`retrain_interval`	1180

V. RESULTS

All but one of the nodes did not exhibit any seasonal patterns even when smoothed, instead showing almost no activity, constant high activity or randomly fluctuating activity. None of these are interesting or even possible to predict. The one clear daily pattern was observed on channel 1 across the entire three weeks. Fig. 4 shows a subset of the raw and smoothed results, while Fig. 5 visualizes only the smoothed results across the entire range. We used only this data set

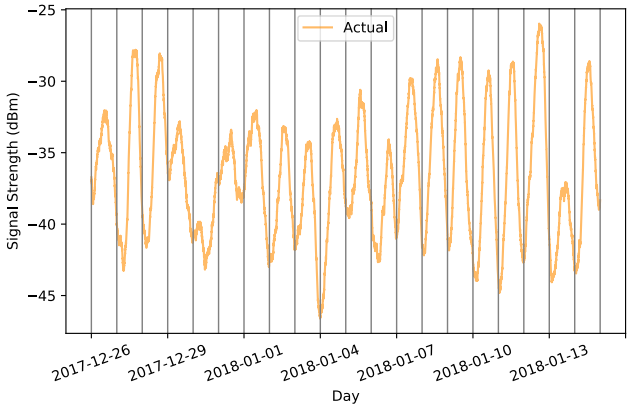

Fig. 5. All smoothed measured data.

TABLE III
RESULTS OVERVIEW

RSSI range (smoothed)	$[-43, -27]$ dBm
runtime	∼1.5 minutes
MASE	0.15
MAE	0.52 dBm

to tune, train and test the neural network. To determine the set of hyperparameters, we first established rough intervals for well performing values through manual experimentation, followed by automated random exploration of the remaining space. Out of over 2500 configurations we chose the best performing one and continued the search in its vicinity for another 1000 configurations. Table II shows the resulting hyperparameter set. We implemented the neural network using the Keras framework[2] with Tensorflow backend[3]. Table III provides an overview of the results after running the network on a 1.8 GHz processor.

We observed that with low `lookback` values, the network would often simply generate predictions very close to its latest input, meaning the predicted RSSI was merely the measured RSSI delayed by one hour. Manual examination of the results shows that this effect diminishes when increasing the `lookback` up to a point, although it is still noticeable in some places. Further increasing the `lookback` past 52 only decreased prediction accuracy, likely due to overfitting. Fig. 6 shows part of the resulting prediction. The first peak is accurately predicted. The major decrease in signal strength is first predicted too early, although the prediction adjusts itself and eventually predicts the major decrease at roughly the correct time. Next, Fig. 7 shows the peak of January 13. This peak was considerably lower than during previous days, as shown in Fig. 5. The predictor however easily adapts to this and does not significantly overestimate the RSSI.

Finally we notice that, with the Savitzky-Golay filter,

[2]https://keras.io/
[3]https://www.tensorflow.org/

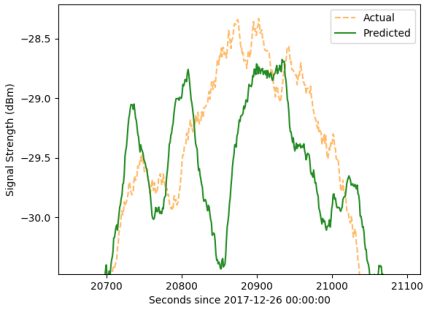

Fig. 6. Predictions for Jan. 9, daytime

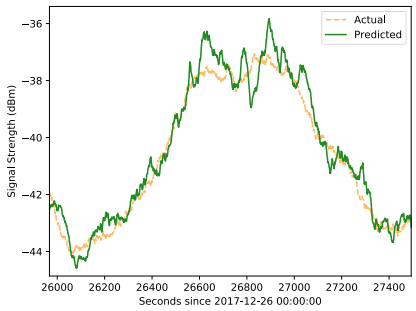

Fig. 7. Predictions for Jan. 13, daytime

a point's smoothing is influenced by points following it, meaning future data influences the smoothing of current data. One way to mitigate this would be to increase the sampling rate. With a sampling interval of $100\,\mathrm{ms}$, a point's smoothing would only take the next minute worth of data into account. By disregarding the final minute of data, we could predict data without any influence of future data points. Another option would be to look into different smoothing algorithms altogether. We will further investigate our smoothing approach in future work.

VI. Conclusions

This paper has introduced CityLab, a novel smart cities experimental facility, which enables multi-technology experimentation in a realistic smart cities context, at a large scale. The testbed is based on gateways, composed of three units, which enable outdoor experimentation with multiple wireless technologies. Eleven gateways in the CityLab deployment are then used to characterize interference. After filtering our measurements, we revealed a clear daily pattern in the interference levels in one case. We then designed and tuned an efficient neural network to predict this interference. Our network outperformed a naive predictor by a factor 6.5, with a mean error of $0.52\,\mathrm{dBm}$, making it usable in future experiments on the CityLab testbed.

Acknowledgements

This work has partially received funding from the European Union's Horizon 2020 framework under grant agreement no. 688941 (FUTEBOL). The equipment used in this work was funded under the Flemish Hercules program. The authors would like to thank the Fed4FIRE+ team for their support.

References

[1] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis *et al.*, "Smartsantander: Iot experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, 2014.

[2] F. Gil-Castineira, E. Costa-Montenegro, F. Gonzalez-Castano, C. López-Bravo, T. Ojala, and R. Bose, "Experiences inside the ubiquitous oulu smart city," *Computer*, vol. 44, no. 6, pp. 48–55, 2011.

[3] D. Budds, "How google is turning cities into r&d labs: From autonomous vehicles to building codes, sidewalk labs is thinking about problems and solutions that could shape cities for centuries," *Fast Company & Inc*, 2016.

[4] D. Simeonidou, "Bristol is open," in *5G Radio Technology Seminar. Exploring Technical Challenges in the Emerging 5G Ecosystem*. IET, 2015, pp. 1–32.

[5] X. Long and B. Sikdar, "A real-time algorithm for long range signal strength prediction in wireless networks," in *2008 IEEE Wireless Communications and Networking Conference*. IEEE, mar 2008.

[6] E. Chin, D. Chieng, V. Teh, M. Natkaniec, K. Loziak, and J. Gozdecki, "Wireless link prediction and triggering using modified ornstein–uhlenbeck jump diffusion process," *Wireless Networks*, vol. 20, no. 3, pp. 379–396, jul 2013.

[7] B. Vermeulen, W. Van de Meerssche, and T. Walcarius, "jfed toolkit, fed4fire, federation," in *GENI Engineering Conference (GEC)*, 2014.

[8] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 255–270, 2002.

[9] P. Demeester, P. Van Daele, T. Wauters, and H. Hrasnica, "Fed4fire : the largest federation of testbeds in europe," in *Building the future internet through FIRE*, 2016, pp. 87–109.

[10] S. Latre, P. Leroux, T. Coenen, B. Braem, P. Ballon, and P. Demeester, "City of things: An integrated and multi-technology testbed for iot smart city experiments," in *Smart Cities Conference (ISC2), 2016 IEEE International*. IEEE, 2016, pp. 1–8.

[11] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, jul 1964.

[12] S. J. Halder, P. Giri, and W. Kim, "Advanced smoothing approach of RSSI and LQI for indoor localization system," *International Journal of Distributed Sensor Networks*, vol. 11, no. 5, p. 195297, jan 2015.

[13] S. T. Ali, V. Sivaraman, and D. Ostry, "Eliminating reconciliation cost in secret key generation for body-worn health monitoring devices," *IEEE Transactions on Mobile Computing*, vol. 13, no. 12, pp. 2763–2776, dec 2014.

[14] R. W. Schafer, "On the frequency-domain properties of savitzky-golay filters," in *2011 Digital Signal Processing and Signal Processing Education Meeting (DSP/SPE)*. IEEE, jan 2011.

[15] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[16] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[17] I. V. Tetko, D. J. Livingstone, and A. I. Luik, "Neural network studies. 1. comparison of overfitting and overtraining," *Journal of Chemical Information and Computer Sciences*, vol. 35, no. 5, pp. 826–833, 1995.

[18] M. Anthony and P. L. Bartlett, *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization."

[20] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, oct 2006.