

Open Call 3

ATLANTIC-eVISION: Cross-Atlantic Experimental Validation of Intelligent SDN-controlled IoT Networks

Deliverable 3: Experiment Results and Final Report

Authors	Sachin Sharma, Avishek Nag, Byrav Ramamurthy, Saish Urumkar, Boyang Hu, Shideh Mehr, Sai Venkat Suman Lamba Karanam, Gianluca Fontanesi, Tiarnan Rush, Catherine Mulwa
Due Date	1st July 2022
Submission Date	1st July 2022
Keywords	SDN, OpenFlow, Machine Learning, Testbeds

Deliverable 3: Part I

Analysis, results, and wider impact

1 Abstract

In this deliverable, we have reported all the experimental results from the ATLANTIC-eVISION project. In particular, we have presented: (1) A performance comparison of all the EU and US testbeds we worked with, (2) Inter-testbed experimental results, (3) Failure-recovery results, (4) Automatic discovery of OpenFlow on a wireless ad-hoc network, (5) An ML-based path computation validation, and (6) Validation of two edge-cloud use cases (eHealthcare and attack detection) involving deep-learning running in parallel with OpenFlow. We also report the issues faced during experiments with EU and US testbeds. Furthermore, we have stated the impacts of this project, the TRL of the project, and future research directions emanating from the project.

2 Project Vision

The main vision of this project is to pioneer future Internet experiments escaping simulations. Simulation has long been a valuable tool in testing and analysing the behaviour of new protocols and ideas in computer networking. Simulations are important, as they mimic the testing environment as closely as possible. It is possible, however, that the simulation environment might not match the real system if some simulation parameters are not configured properly, or some parameters are

unknown. Test-beds are viewed as a means of tackling the problem by allowing protocols and initial prototypes to be tested in an environment which is non-idealistic and closely replicates the real system, even if some parameters are unknown. Moreover, when we move from simulation to testbeds we add more realism to the evaluation of the protocol/system. A simulation environment can be seen as an ideal environment in which a protocol/system can behave well. However, when unexpected events and dynamic environmental constraints are induced (such as in a testbed including real physical systems) the behaviour of the protocol/system can be evaluated with higher confidence [1]. Our second vision is to extend a pervasive network management protocol like OpenFlow towards wireless ad-hoc networks that can be managed with OpenFlow and will pave the path for scalable, low-cost, self-configurable, and programmable future IoT networks supporting a plethora of use cases. The technologies and approaches we are proposing in this project can have a huge impact on the speed at which IoT service providers can make their infrastructure efficiently evolve with their market evolution. In fact, the impact of this approach for wireless ad-hoc IoT networking can be extended easily to other IoT device communication platforms (e.g., MQTT) for a comprehensive solution. Furthermore, we aim to create exemplary knowledge by performing experimentation on one of the world's largest and most advanced wireless testbeds, located over two continents. By remotely running experimentation across the Atlantic the project will "stress-test" the performance of novel algorithms and achieve the project's KPIs in one of the most challenging scenarios in terms of round-trip latency and network heterogeneity. Additionally, developing and experimenting with novel algorithms on such an advanced Cross-Atlantic testbed is going to be an invaluable and truly unique experience for postdocs and research assistants working on the project, with a strong impact on the advancement of their future research careers. In general, the knowledge developed by this project will impact on creating new curricula and research spokes in the higher education sector and help in training skilled personnel in the combined area of AI-aided IoT networks management that the project addresses.

3 Details on participants (both EU and US)

The project was done under the leadership of Dr. Sachin Sharma, Dr. Avishek Nag and Prof. Byrav Ramamurthy (US). 4 research assistants (RA) from the EU and 3 PhD students from the US were also employed. The short CVs and the roles of all the personnel involved in the project are given below.

Dr Sachin Sharma Dr. Sachin Sharma is a lecturer at TU Dublin. He also works as an associate investigator at the CONNECT Centre. Previously, he worked as a lecturer and programme director at National College of Ireland. He performed his post-doctoral research at NEC Laboratories Europe. Dr. Sharma has worked on several EU and Flemish projects: FP7-SPARC, FP7-OFELIA, FP7-CityFlow, FP7-UNIFY, FP7-CleanSky and MECANO where he extensively experimented with the Fed4Fire testbeds to publish 40+ papers with 1300+ citations. Dr



Sharma is the coordinator of the project and supervised two RAs. He also co-lead WP1 and lead WP3.

Dr Avishek Nag Dr Avishek Nag is currently an Assistant Professor in the School of Electrical and Electronic Engineering at University College Dublin in Ireland. Dr Nag received his PhD degree from University of California, Davis in 2012. He worked as a research associate at the CONNECT centre for future networks and communication in Trinity College Dublin. Dr Nag is the recipient of the Best Paper Award at the 2nd IEEE Advanced Networks and Telecommunication Symposium in 2008 and has published over 70 publications with over 1550 citations. His research interests include Mathematics of Networks (Optimisation, Graph Theory), Network Virtualisation, Software-Defined Networks, Machine Learning, Data Analytics, Blockchain, and the Internet of Things. Dr Nag is a senior member of the Institute of electronics and electrical engineers (IEEE) and also the outreach lead for Ireland for the IEEE UK and Ireland Blockchain Group. Dr Nag is the Co-PI from the EU side and supervised 2 RAs. Dr Nag lead WP2 and mainly overseen most of the machine-learning related aspects of the project.

Prof Byrav Ramamurthy Dr Byrav Ramamurthy is currently a Professor and former Graduate Chair in the Department of Computer Science and Engineering at the University of Nebraska-Lincoln (UNL). He is the author of the book "Design of Optical WDM Networks - LAN, MAN and WAN Architectures" and a co-author of the book "Secure Group Communications over Data Networks" published by Kluwer Academic Publishers/Springer in 2000 and 2004 respectively. He served as the Chair of the IEEE Communication Society's Optical Networking Technical Committee (ONTC) during 2009-2011. He served as the IEEE INFOCOM 2011 TPC Co-Chair. He is currently the Editor-in-Chief for the Springer Photonic Network Communications (PNET) journal. His research areas include optical and wireless networks, peer-to-peer networks for multimedia streaming, network security and telecommunications. His research work is supported by the U.S. National Science Foundation, U.S. Department of Energy, U.S. Department of Agriculture, NASA, AT&T Corporation, Agilent Tech., Ciena, HP and OPNET Inc. Prof Ramamurthy lead the US side of this project. He co-lead WP1 with Dr Sharma and lead WP4.

Saish Urumkur (Research Assistant) is working as a senior research assistant at Technological University Dublin. His previous experiences involved working in Maynooth University and Trinity College Dublin. He worked on the jFed test bed in order to set up non-standalone and standalone 5G. The task also involved working on the SRS LTE test-bed to collect data and build machine learning models for optimizing network resources such that the network throughput is not reduced drastically at the same time power consumed is kept to minimum. He was also involved in developing code for API calls from IOT devices regarding information like uptime status, power consumed, turning on and off the devices remotely. He also has around 4 years of industrial experience working with Huawei Technologies Ireland and India. Saish has a Masters in Computer and Communication System from University of Limerick, Ireland, Post graduate



diploma in Artificial Intelligence and Machine Learning from NIT Warangal, India and BEng. Electronics Engineering from Mumbai University, India. Saish contributed to WPs 2, 3, and 4 of the project.

Boyang Hu (Research Assistant) Boyang Hu is a Ph.D. Student in the Computer Science and Engineering department at the University of Nebraska-Lincoln. His current work is on network softwarization and security at the Networks Research Group (NetGroup), under the supervision of Dr. Byrav Ramamurthy. His research interests include Software Defined Networking (SDN), Advanced Computer Networks, Network Security, and applications of Machine Learning in Optical Networks. He worked on the US side, focusing on building IoT networks on US testbeds and Validation and testing (WP4).

Shideh Yavary Mehr (Research Assistant) Shideh Yavary Mehr is currently pursuing her Ph.D. degree in computer engineering from the University of Nebraska at Lincoln. She received her B.S. and M.S. in Electrical Engineering from Iran, Tehran, in 1998 and 2006. Her research interests include Software Defined Networking, Optical networks and security of Machine learning and big data. She worked on the US side, focusing on building IoT networks on US testbeds and Validation and testing (WP4) and task 3.3 of work package 3 on ML based path selection.

Sai Venkat Suman Lamba Karanam (Research Assistant) Sai Suman is a PhD student in the Computer Science and Engineering Dept. at UNL. He is from Hyderabad, a city in the south part of India. He works with Prof. Byrav Ramamurthy and his research interests are run-time optimization using dynamic programming and machine learning techniques in i) large-scale optical and multi-domain networks and ii) large data transfers. He worked on the US side, focusing on building IoT networks on US testbeds and Validation and testing (WP4) and Task 3.5 on IoT application development of WP3.

Dr Gianluca Fontanesi (Research Assistant) Gianluca Fontanesi received the B.Sc and M.Sc degree from Politecnico di Milano, Italy in 2009 and 2012, respectively. He has several years of industrial experience as consultant for Nokia and as Software and Radio Integration Engineer with Azcom Technology. He commenced his PhD study in the RF & Microwave Group at UCD in October 2017 under the joint supervision of Prof. Anding Zhu and Dr. Hamed Ahmadi, funded by the Irish Research Council under the Government of Ireland Postgraduate Scholarship Scheme. Gianluca graduated with a PhD in 2022. Gianluca contributed to WPs 2, 3, and 4 of the project.

Dr Catherine Mulwa (Senior Research Assistant) Dr. Catherine Mulwa is an associate faculty (part-time) lecturer at National College of Ireland since 2010. She has a PhD in Computer Science from Trinity College, an M.Sc. in computing and a BSc (Hons) in Computer Science. As part of pedagogical career development she has a Postgraduate Diploma in Third Level Learning and Teaching, a certification in Academic Practice and E-Learning. Dr. Mulwa has over 12+ years teaching experience in third level institutions in Ireland; lecturing



on a variety of computer science programmes on both postgraduate and undergraduate programmes. She has successfully supervised to completion over 300+ MSc projects in Data Analytics, Cloud Computing (software defined networking) and software engineering. In addition, Dr. Mulwa has over 13 years' research experience in the fields of technology enhanced learning, cognitive systems and predictions. Her research has focused on recommender systems, data analytics i.e. healthcare analytics and evaluation of adaptive systems. She was a postdoctoral researcher from 2015 to 2019 where she worked on the EAGLE EU funded project and also was PI of the Enhance project funded by the Irish Research Council. Her research has resulted in three innovative web-based systems i.e. A recommendation system for evaluations of adaptive systems (2014), a web-crawler for crawling published articles which supports researchers at an early stage (2010) and a recommendation web accessibility checker which globally can validate any web-based application/website and present the results to a user in PDF and word format (2019). Catherine contributed to WP2 and WP3 during the first three months of the project.

Tiarnan Rush (Research Assistant) Tiarnan Rush completed a BSc in Networking Applications and Services from Technological University Dublin. He has a keen interest in networking shown through his final year project during his degree, "Future Internet Simulations", which involved creating topologies in a wireless testbed and comparing how it would perform against a simulated environment using NS3 in a similar topology. Tiarnan worked closely in WP2, and parts of WP3 and WP4 during the first three months of the project.

4 Results

4.1 Discussion and Analysis on Results

4.1.1 Testbed Preparation

This section discusses the steps to configure an experiment on different testbeds. Generally, the test configuration involves different steps. The first step for each testbed is the account registration and nodes reservation.

POWDER

The following are the steps to configure an experiment at POWDER:

1. Register an account and join or create a project at the POWDER main page, <https://www.powderwireless.net/signup.php>
2. To start an experiment, choose on the top left panel list "Experiments" the option "Start Experiment"
3. The experiment will guide to the choice of the profile, and the time/hours scheduling. New profile can be created to satisfy the user requirement or the user



can use some public example experiment profiles to get used with the testbed. For our test, we created profiles to manage wireless nodes.

4. In order to start an experiment a new profile consisting of Rspec need to be prepared
5. Run the experiment defined in the profile file.
6. To access the nodes, double click and enter the ssh. Ubuntu 20.04 is used as the Linux version to setup the 4 node experiment consisting of only one wireless interface. Before proceeding install via bash wireless tools, network manager and hostapd. We used this one wireless interface to create two virtual interface one acting as master adhoc and another acting as wpa_supplicant to connect to wireless node.

w-lilab1.t

1. Prerequisite for performing tests on W-iLab1.t is the Fed4Fire profile registration (<https://portal.fed4fire.eu/profile>), the Jfed Experimenter toolkit and the SSH installation.
2. Tests can be started using jFed. The user needs first to reserve the nodes. Log in into <https://boss.wilab1.ilabt.iminds.be/inventory/?viewMode=inventory#> . Select the reservation option in <https://boss.wilab1.ilabt.iminds.be/inventory/?viewMode=inventory#>. Once defined the desired time window, the user can see the availability of the node for each test floor.
3. In jFed select the new button and drag "Wireless node" onto the canvas. To edit the name of each node, double click on the node box. In the "Name" field at the top, replace "node0" with the desired name. Repeat for each node in the test architecture, if desired.
4. Then, right-click on the nodebox and select the reserved node with wireless capabilities. When reserving nodes on W-iLab1.t the user must note that only NUC's nodes support wireless setup. If NUC nodes are reserved, the wireless tests cannot be performed. Edit the node selecting the chose, testbed and image from the drop-down menu. Repeat the operation for each node. Alternatively, in the reservation window of <https://boss.wilab1.ilabt.iminds.be/inventory/?viewMode=inventory#>, click on the JFed symbol to copy a Jfed RSPEC of the reservation into the clipboard. The reservation can be then used to import the nodes configuration in Jfed, in RSPEC editor window.
5. Click the Run button near the top of the window.

w-ilab2.t

1. Prerequisite for performing tests on W-iLab2.t is the Fed4Fire profile registration (<https://portal.fed4fire.eu/profile>), the jfed Experimenter toolkit and the SSH installation. In addition, it is mandatory to reserve nodes on W-iLab2.t to downgrade the current chrome's version to version 80.



2. Tests can be started using jFed. The user needs first to reserve the nodes. Log in to <https://inventory.wilab2.ilabt.iminds.be/>. Select the reservation option. Once defined the desired time window, the user can see the availability of the node for each test floor.
3. In jFed select the new button and drag "Wireless node" onto the canvas. To edit the name of each node, double-click on the node box. In the "Name" field at the top, replace "node0" with the desired name. Repeat for each node in the test architecture, if desired.
4. Then, right-click on the nodebox and select the reserved node with wireless capabilities. In W-iLab2.t both APU and zotack nodes support wireless interfaces which have 2 interfaces.
5. Click the Run button near the top of the window.

CityLab

1. Prerequisite for performing tests on CityLab is the Fed4Fire profile registration (<https://portal.fed4fire.eu/profile>), the jFed Experimenter toolkit, and the SSH installation.
2. Tests can be started using jFed. The user has to select "CityLab" in the from the "Select testbed" drop-down. With the CityLab testbed, it is possible to let the system pick any available node at the moment, without the need of reservation. However, the user should note that unlikely he will find more than 4 or 5 nodes available.
3. In jFed select the new button and drag "Wireless node" onto the canvas. To edit the name of each node, double-click on the node box. In the "Name" field at the top, replace "node0" with the desired name. Repeat for each node in the test architecture, if desired.
4. Click the Run button near the top of the window.

COSMOS

1. A prerequisite for performing tests on Cosmos is the ORBIT profile registration (<https://cosmos-lab.org/portal-2/>).
2. Reserve the nodes using the ORBIT portal. In order to do so, double-Click on the desired timeslots. It is desirable to perform a wireless test to reserve the sandbox number 4. Please note that the printed time is ORBIT Time (Eastern/NY).
3. Once the reservation is confirmed, it is possible to access the selected nodes via ssh. The connection can be configured in the local computer or using the ORBIT online portal.
4. To start the test, run the appropriate lperf commands

GPULAB



- 1) Get the credentials and login to <https://www.fed4fire.eu/testbeds/gpulab/>
- 2) Open the jupyterhub using the link <http://jupyterhub.ilabt.imec.be>
- 3) Select the server options: project name, docker image type and resources to be allocated.
- 4) Click start to run the server and a workspace with the specified configuration.
- 5) Upload your data onto the server and create a new jupyter notebook.
- 6) Write the code for the project in the jupyter notebook and execute it.

Our Wireless Experiment on each testbed

W-ilab1.t, W-ilab2.t and Powder are emulab based testbeds. Currently, emulab testbeds provide a simple way to create wired networks topologies using an Ethernet switch. All the different types of wired topologies - linear, ring, grid, mesh - can be created. However, there are not many studies performed on how to set up wireless ad hoc network topologies in emulab. For wireless experiments, Emulab installs real wireless devices on a building space. Most of these wireless devices contain IEEE 802.11 a/b/g WiFi interfaces (Atheros and INTEL chipset), and are located around at various places in a large building or in a city. Further, all these wireless devices contain only two WiFi interfaces. The problem is that not all the nodes/WiFi interfaces support the ad-hoc mode. Therefore, we create the ad-hoc network topologies (linear, ring, grid and mesh) using the access-point mode and use different non-interfering frequencies to create different links.

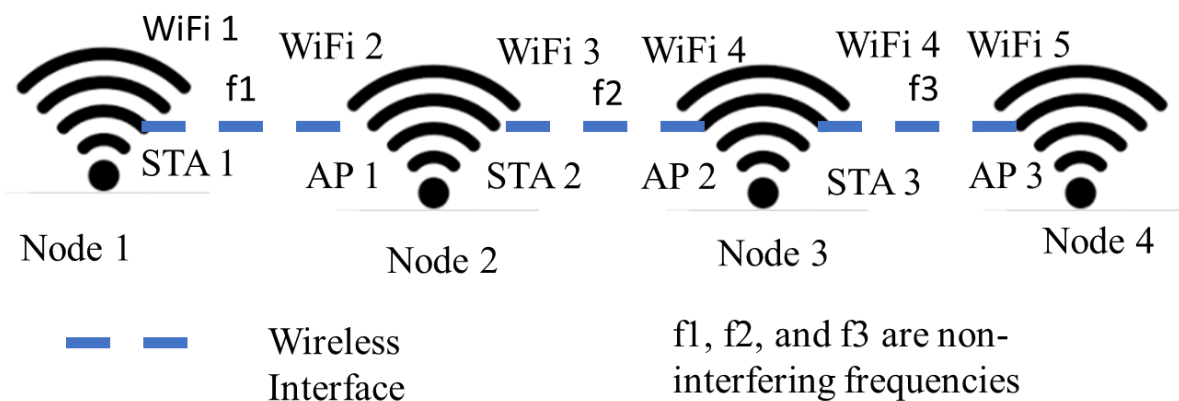


Figure 1: Wireless Ad hoc Network Topology Creation at Emulab (W-ilab1.t, W-ilab2.t and Powder)

Our method for creating a wireless ad-hoc network topology on a linear topology is shown in Figure 1. In the diagram, node 1, node 2, node 3 and node 4 with WiFi interfaces are shown. Since all WiFi interfaces support the AP (access point) mode, we created ad-hoc network topologies using this mode. The links are created using non-interfering frequencies (f1, f2 and f3). In the first link, WiFi1 and WiFi2 are connected via frequency f1 where WiFi1 serves as a station and WiFi2 acts as an

access point. Further, frequency f2 is used for the second link, where WiF4 is the AP and WiF3 is the station. Moreover, the third link is created by using frequency f3, where WiFi 5 acts as an AP and WiFi 4 acts as a station. We enabled OLSR (Optimised Link State Routing) in each node so that each node can communicate with each other.

The hardware details of all used nodes are presented in Table 1. According to Table 1, all nodes used on the CityLab testbed include AMD processors with a frequency of 1 GHz on each of the four cores available. Moreover, nuc0-3, nuc0-5, nuc0-6, nuc0-8, nuc0-9, nuc0-10, nuc0-11, nuc0-14, nuc0-15, nuc0-17 of W-ilab1.t contains Intel processors with 1.3 GHz frequencies in each of the 4 cores while apuW2, apuW4, apuW5, zotack6, apuV5, apuU6, zotacI6, apuT5, zotacH6, apuS2 of W-ilab2.t contains Intel processors with 1.8 GHz frequencies in each of the 4 cores. In POWDER there are 4 nodes supporting wireless setup and it contains Intel processors with 2.3 GHz frequencies in each of 4 cores. Table 1 also shows that the RAM of CityLab nodes is 4 GB, as opposed to 8 GB for W-iLab1.t, W-iLab2.t and POWDER. All the testbed nodes are selected based on their availability at the time of the experimentation. The information on topologies with respect to each testbed is provided below:

Table 1: Hardware Resources used in the single-testbed experiments

Testbed	Name	Location	CPU	RAM (Memory)
W-iLab1.t	nuc0-3, nuc0-5, nuc0-6, nuc0-8, nuc0-9, nuc0-10, nuc0-11, nuc0-14, nuc0-15, nuc0-17	Zwijnaarde, Ghent	Intel(R) Core(TM) i5-4250U CPU@ 1.30 GHz Quad-core	8 GB
W-iLab2.t	apuW2, apuW4, apuW5, zotack6, apuV5, apuU6, zotacI6, apuT5, zotacH6, apuS2	Zwijnaarde, Ghent	AMD G-T40E Processor Dual-core, Intel(R) Atom(TM) CPU D525 @ 1.80 GHz Quad-core	4 GB
CityLab	node71, node72, node73, node74	Antwerpen, Belgium	AMD GX-412TC SOC@ 1 GHz Quad-core	4 GB
POWDER	ota-nuc1, ota-nuc2, ota-nuc3, ota-nuc4	Salt Lake, Utah	Intel(R) Core(TM) i5-5300U CPU@ 2.30 GHz Quad-core	8 GB
COSMOS	Large and Medium Nodes: N310 and 2974; Big portable: B210; Small portable: B205mini; Handheld: E312	Rutgers	Large Nodes: Dedicated Servers; Medium Nodes: Shared Servers; Big Portable: 45W Intel; Small Portable: 15W Intel; Handheld: Embedded ARM	NA

POWDER

1. **Linear Topology:** In powder the available nodes supporting wireless interfaces were 4. Based on the available nodes the 4 nodes linear topology was setup as can be seen in Figure 2. F1, F2, F3 are non-interfering frequencies of 2.4 GHz channel which are 1, 6, 11 respectively. The connectivity is done by using a single wireless

interface which was only available in Powder Lab nodes. This single interface have been configured as 2 virtual interfaces to support function of Wi-Fi and Station as shown in the setup description of Figure 1.

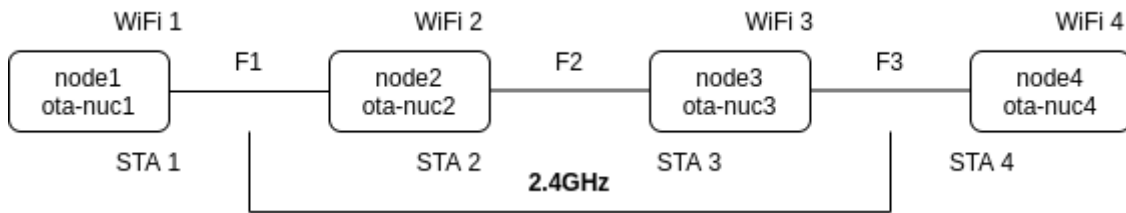


Figure 2: POWDER-Linear Topology Node Setup

2. Ring Topology: Ring Topology setup consisting of 4 powder nodes can be seen in Figure 3. Here the combination of 2.4 GHz channel and 5 GHz frequency were used. Here F1,F2,F3 are frequencies of channel 1, 6, 11 while F4 here is a 5 GHz channel which is 36. OLSR helped to optimize the routing and hence balance the routes between the 2 paths

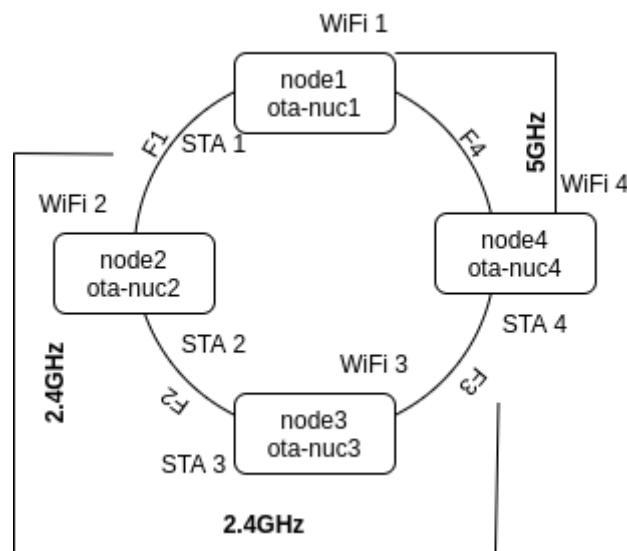


Figure 3: POWDER-Ring Topology Node Setup

COSMOS

We deployed a 4-node linear topology in COSMOS and performed similar experiments as we did in POWDER. For COSMOS, we faced some issues. The COSMOS sandbox 4 is better suited for our NGIAtlantic experiments listed at. However, getting reservations on sandbox 4 (which is ideal for our needs) has been difficult, since it is always in high demand. In addition, COSMOS node images do not come with the drivers for their WiFi cards, so they must be installed with additional packages. Because of these issues we are not able to collect results for ring and grid topologies.

W-iLab1.t

1. **Linear Topology:** In W-iLab1.t more nodes were available. 10 node setup was done in W-iLab1.t based on the availability at the time of reservation as displayed in Figure 4. F1, F2, and F3 are 2.4 GHz channels i.e., 1, 6, 11 while F4, F5, F6, F7 are 5 GHz lower channels i.e., 36, 40, 44, 48, and F8, F9 are 5 GHz upper channel which is 149 and 153.

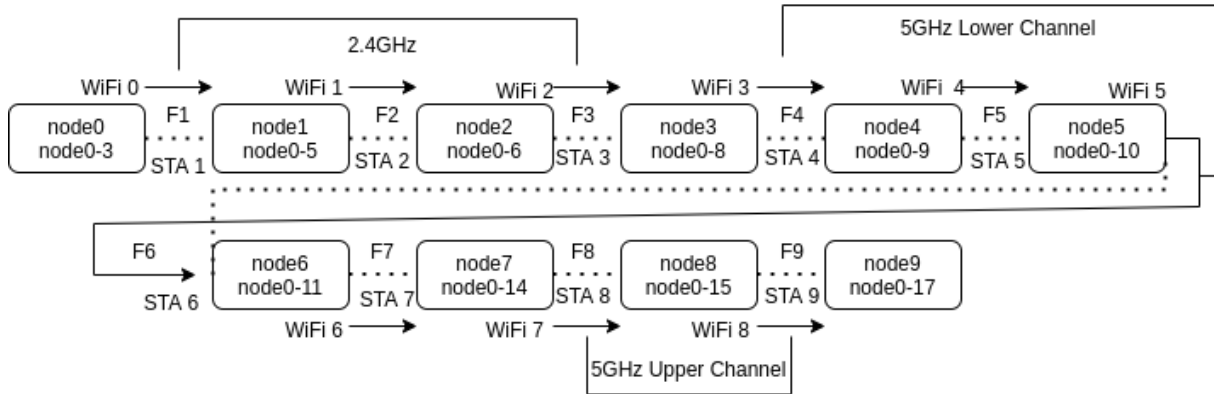


Figure 4: W-iLab1.t-Linear Topology Node Setup

2. **Ring Topology:** In Ring topology setup is similar to W-iLab1.t Linear topology only difference is the addition of one more 5 GHz upper channel in F10 which is 157 to connect node9 to node 0 as can be seen in Figure 5.

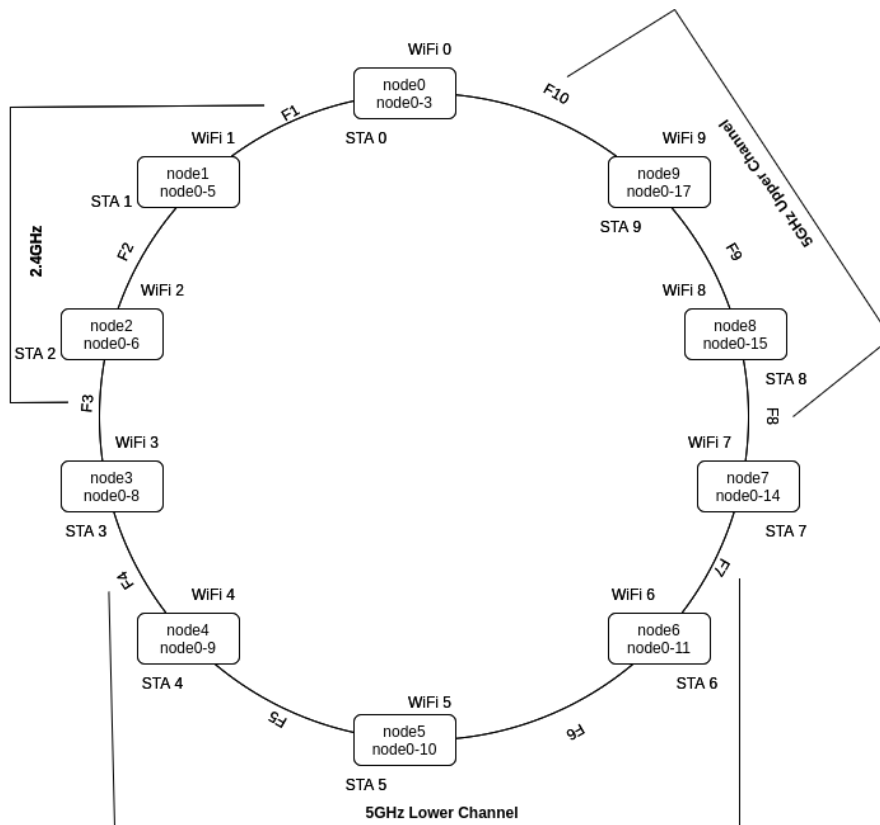


Figure 5: W-iLab1.t-Ring Topology Node Setup

3. **Grid Topology:** The formation of grid topology is designed in such a way that it uses the available 2 wireless interfaces to form a grid between 10 nodes. The network setup is shown in Figure 6. F1,F2, and F3 make use of channels 1, 6, and

11. F4, F5, and F6 make use of channels 36, 40, and 44 i.e., 2.4 GHz and 5 GHz Lower frequency channels.

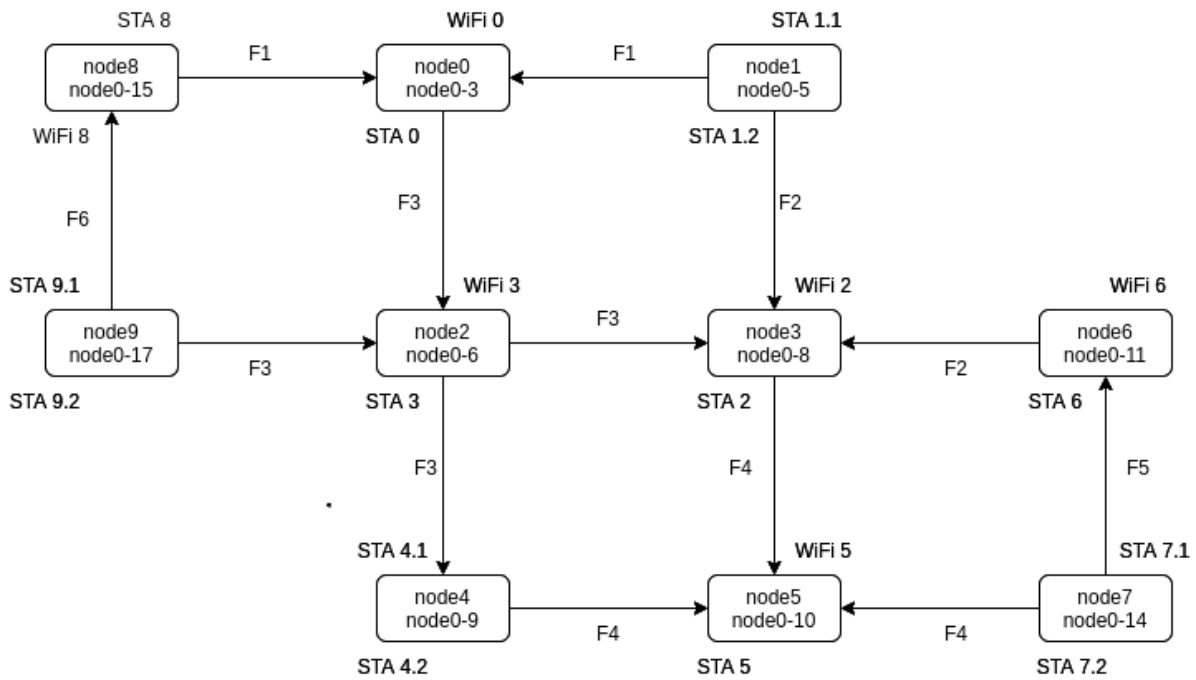


Figure 6: W-iLab1.t-Grid Topology Node Setup

W-iLab2.t

1. **Linear Topology:** In W-iLab2.t similar to W-iLab1.t 10 nodes are setup as per the availability at the time of reservation as displayed in Figure 7. F1, F2, F3 are 2.4 GHz channels i.e. 1, 6, 11 while F4, F5, F6, F7 are 5 GHz lower channels i.e., 36, 40, 44, 48, and F8, F9 are again 2.4 GHz channel i.e., channel 1 and 6 because zotac node was unable to connect to upper 5 GHz channel as it supports only 802.11 a/b/g/n wifi whereas APU supports 802.11ac i.e. 5 GHz channels and is also backward compatible with 2.4 GHz channel i.e., 802.11n which is also backward compatible with 802.11b/g.

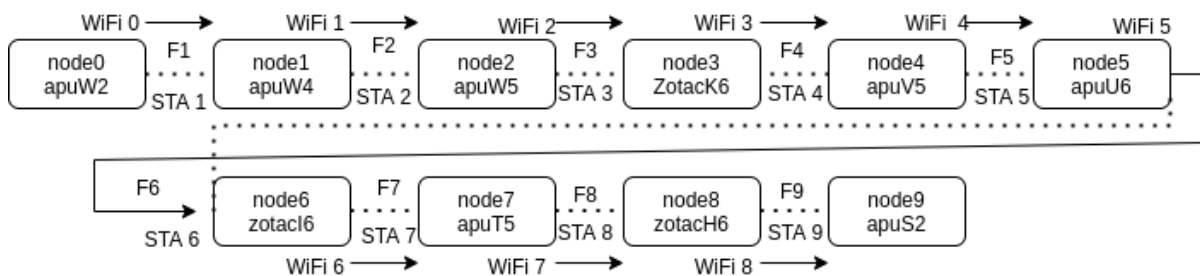


Figure 7: W-iLab2.t-Linear Topology Node Setup

2. **Ring Topology:** For ring topology in W-iLab2.t, apu node supports 802.11 ac i.e., all 5 GHz channel. The configuration was done on F10 for upper 5 GHz channel i.e., 157 which connects node 9 to node 0 similar to Figure 5.

3. Grid Topology: The formation of grid topology is designed in such a way that it uses the available 2 wireless interfaces to form a grid between 10 nodes. The network setup is shown in Figure 8. F1, F2, and F3 make use of channels 1, 6, and 11. F4, F5, F6 make use of channels 36, 4 and 0, 44 i.e., 2.4 GHz and 5 GHz Lower frequency channel. The Zotac node supports 5 GHz lower channel but not higher channels.

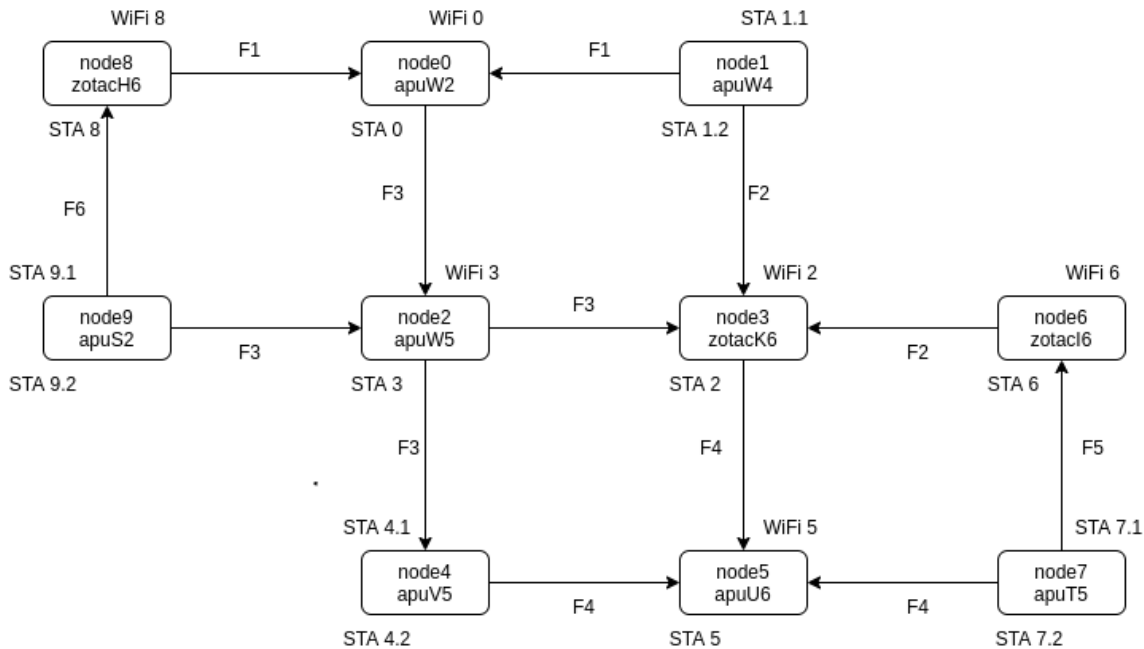


Figure 8: W-iLab2.t-Grid Topology Node Setup

CityLab

In CityLab we performed experiments on 4 nodes depending upon the availability of hardware compatibility. There were other 3 nodes named node6, node23 and node27, node6 and node23 are physically too far away from each other, so the signal strength was too weak to perform the connectivity between the two. Node 23 and node 27 were also a little far from each other and their hardware is little different, i.e., node23 supports both ath10k and ath9k module while node27 support ath10k and intel 7260 because of different hardware on the second wireless interface the connectivity was giving some problem and also the proximity between node 27 and 71 is too much causing weak signal. Node71, node 72, node 73, and node 74 were found to be perfect in terms of proximity and same hardware, so the setup was done on these nodes according to availability.

1. Linear Topology: A linear topology between 4 nodes was setup as shown in Figure 9. Similar to the POWDER setup, the configuration has been done with non-interfering channels, i.e., 2.4 GHz channels where F1, F2, F3 are 1, 6, 11 respectively.

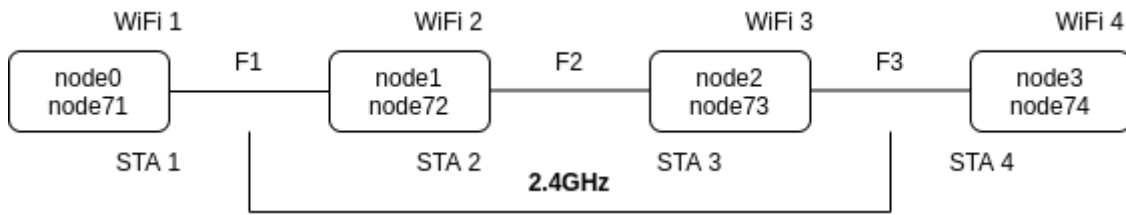


Figure 9: CityLab-Linear Topology Node Setup

Ring Topology: Ring Topology setup is done similar to 4 POWDER nodes, it can be seen in Figure 10. Here the combination of a 2.4 GHz channel and 5 GHz frequency was used. Here F1, F2, F3 are frequencies of channels 1, 6, and 11, while F4 here is a 5 GHz lower channel which is 36.

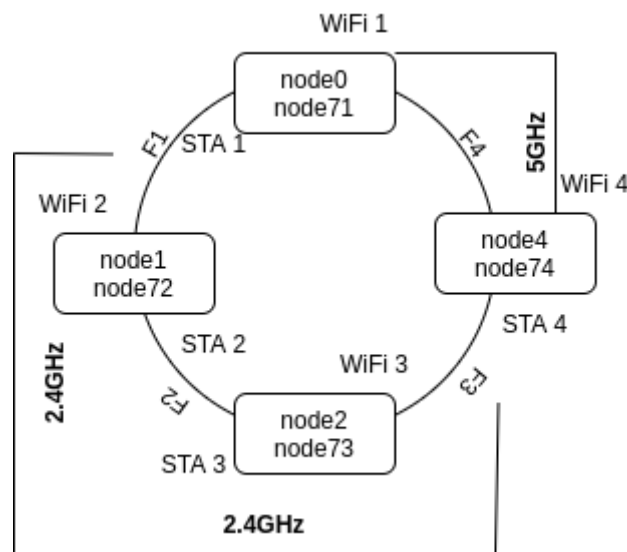


Figure 10: CityLab-Ring Topology Node Setup

The objectives of these basic benchmark experiments include:

1. To understand the hardware and software supported by each testbed.
2. Finding testbeds' compatibility with the proposed experiments of this project.
3. To find out the bandwidth bottleneck for each testbed.

Result Analysis

POWDER

POWDER lab setup is done using 4 nodes. The results show increased bandwidth for ring topology compared to linear topology because of the use of 5 GHz lower channel which increases the speed of data transmission through with low coverage as the second path for ring topology is done by using the combination of 5 GHz lower channel and 2.4 GHz channels. Because of less number of available nodes only 2 topologies are possible in POWDER i.e., linear and ring topology. Results for the POWDER lab are as follows:

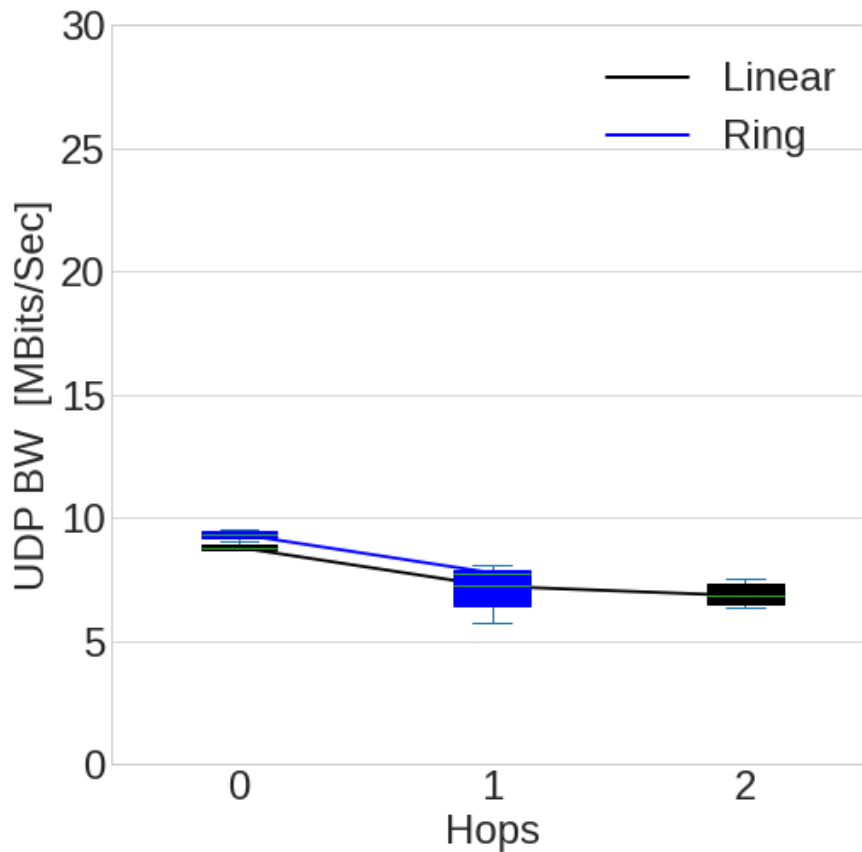


Figure 11: POWDER LAB - UDP Bandwidth

1. UDP Bandwidth: The UDP bandwidth for POWDER lab setup shows that the maximum bandwidth is around 8 Mbps for hop 0, and it is decreased to around 7 Mbps for hop 1 and for hop 2 to around 6 Mbps for linear topology (Figure 11). For ring topology only 2 hops are present on both paths. The results show that for ring topology for hop 0 the bandwidth is around 9 Mbps and for hop 1 the bandwidth is around 7.5 Mbps (Figure 11).

2. UDP Loss: The UDP loss for POWDER lab shows that the percentage loss is much more in linear topology than in ring topology refer Figure 12. In linear topology and ring topology for hop 0 there is no loss whereas for hop 1 in linear topology the loss is increased to around 22 percent and for hop 2 it is around 25 percent. In ring topology for hop 1 loss is increased to around 8 percent. There is no hop 2 in ring topology as it has only 2 hops in both paths.

3. UDP Jitter: The UDP jitter is much more in ring topology than in linear topology refer Figure 13. In linear topology and ring topology for hop 0 the jitter is around 1 ms and around 5 ms respectively, whereas for hop1 in linear topology the jitter is increased to around 4.9 ms and 5.5 ms respectively. For hop 2 it is around 25%. In ring topology for hop 1 loss is increased to around 8 percent.

4. TCP and SCTP Bandwidth: TCP and SCTP bandwidth is more for hop 0 in both linear and ring topology which is around 10.7 Mb/s for TCP and 9 Mb/s in SCTP for ring topology whereas it is 8.2 Mb/s for TCP and 6.1 Mb/s for SCTP in a linear topology. For hop 1 TCP and SCTP in linear topology is around 5.5 Mb/s whereas in

ring topology TCP is 5.5 Mb/s and SCTP is around 4.5 Mb/s. Hop 2 exists in linear topology only, and it is 3 Mb/s for TCP and around 2.9 Mb/s for SCTP (refer to Figure 14).

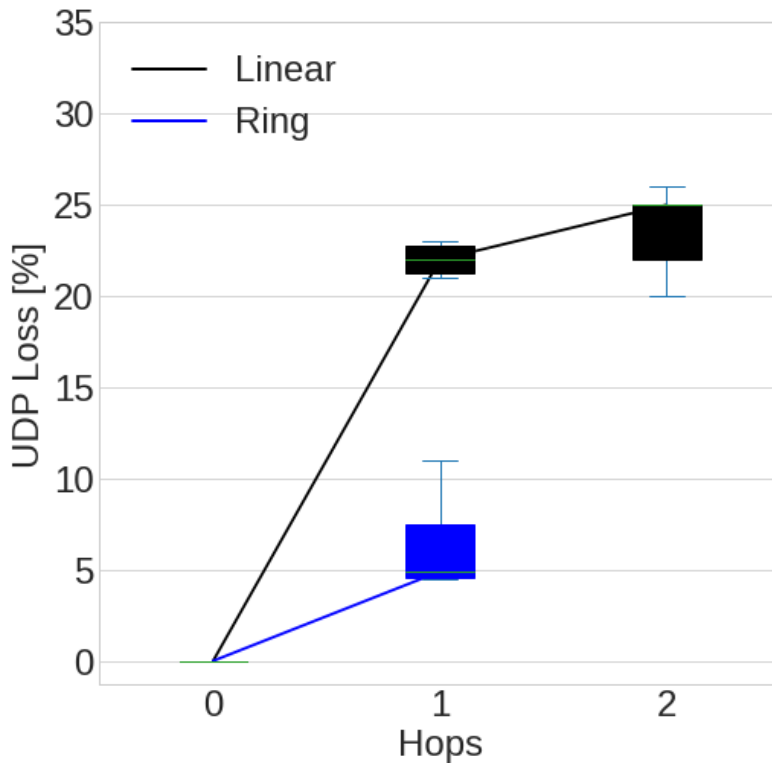


Figure 12: POWDER LAB - UDP Loss

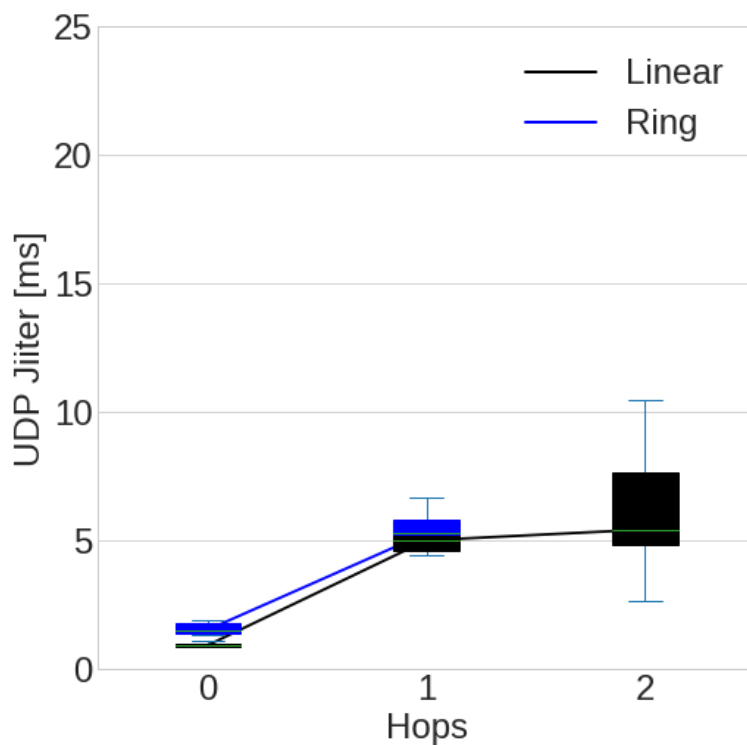


Figure 13: POWDER LAB - UDP Jitter



5. TCP and UDP Latency: TCP and UDP latency are between 0.00 to 0.1 seconds for hop0 and hop1 in both linear and ring topology whereas for hop 2, TCP and UDP latency increased to 0.06 seconds (refer to Figure 15).

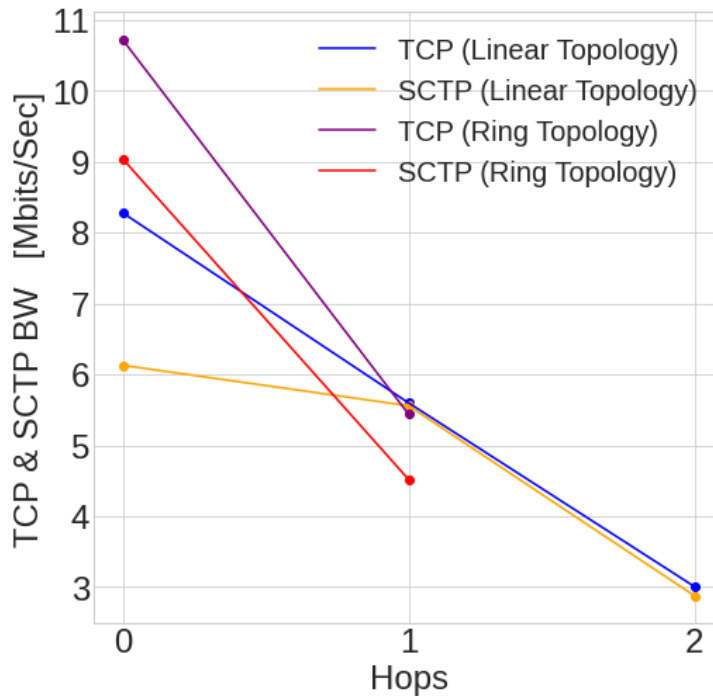


Figure 14: POWDER LAB - TCP and SCTP Bandwidth

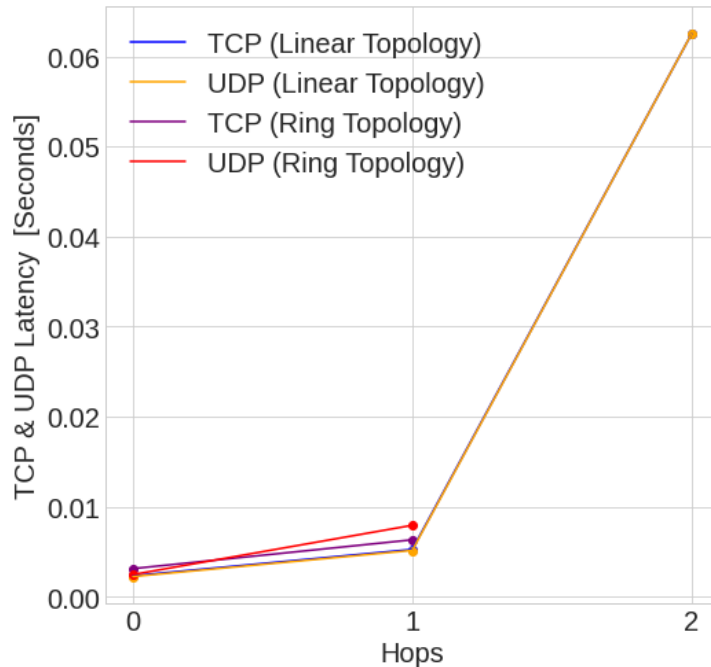


Figure 15: POWDER LAB - TCP and SCTP Latency

W-iLab1.t

The results for W-iLab1.t are based on 10 nodes as compared to 4 nodes in POWDER. W-iLab1.t setup consists of linear, ring, and grid topology.



1. UDP Bandwidth: In W-iLab1.t maximum UDP bandwidth is observed in Ring topology around 27 Mb/s for hop 0 because of the presence of a 5 GHz upper channel connecting node9 to node0. The plot shows a decrease in the bandwidth with an increase in hops for all topologies, i.e., linear, ring, and grid (refer to Figure 16).

2. UDP Loss: There is no almost loss until hop 7 but for hop 8 it is a slight increase i.e. around 5 percent in a linear topology. For ring, there is a negligible loss for hop 0 and hop 4 while it is around 18 percent for mid-hop 2 and hop 3. For the grid topology, the loss is negligible for hop 0 and hop 1 while it is 25 percent for hop 2, hop 3 and hop 4. The loss is 35 percent for hop 5 and then decreased to 22 percent for the last hop i.e., hop 6. The reason for such an increase in a loss in grid topology is due to the use of one node to connect to multiple nodes and perform an exchange of traffic (refer to Figure 17).

3. UDP Jitter: The Jitter is increased as hops are increased and this is the same for the linear, ring, and grid topology (refer to Figure 18).

4. TCP and SCTP Bandwidth: TCP and SCTP bandwidth decreases as hops increases for a linear, ring, and grid topology refer to Figure 19. For hop1 bandwidth is not decreased much because of the use of a 5 GHz lower channel and the proximity to node 0 and node2 (refer to Figure 6) for the understanding of the setup and relation with results.

5. TCP and UDP Latency: UDP latency is increased in linear topology because of the distance from node 0 to node 9 i.e. from 575 us to 1 sec in order to cover more number of hops while TCP latency is minimal i.e., between 500 us to 24 ms. For other topologies i.e., ring and grid, UDP latency is minimal (Figure 20).

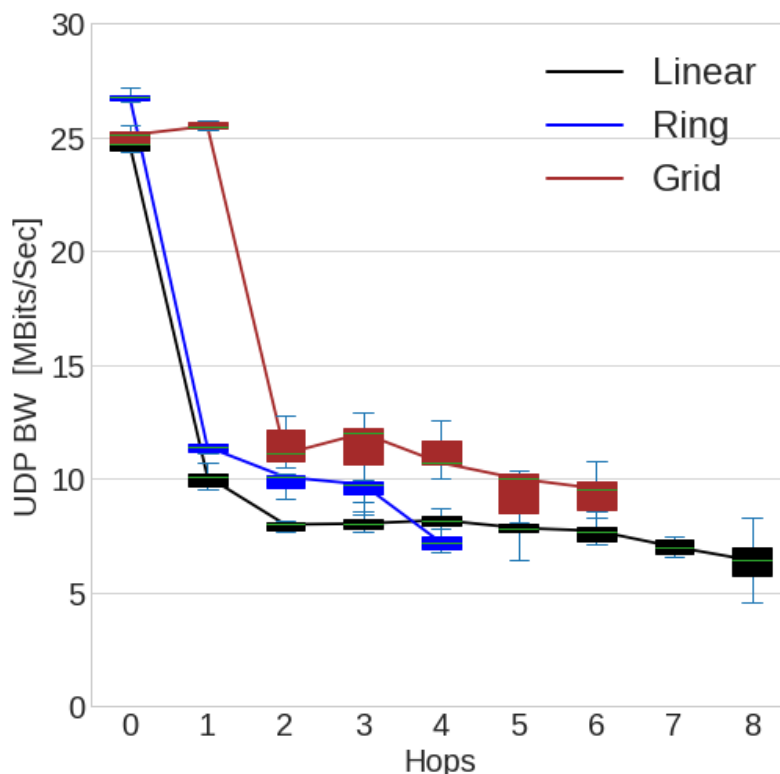


Figure 16: W-iLab1.t- UDP Bandwidth

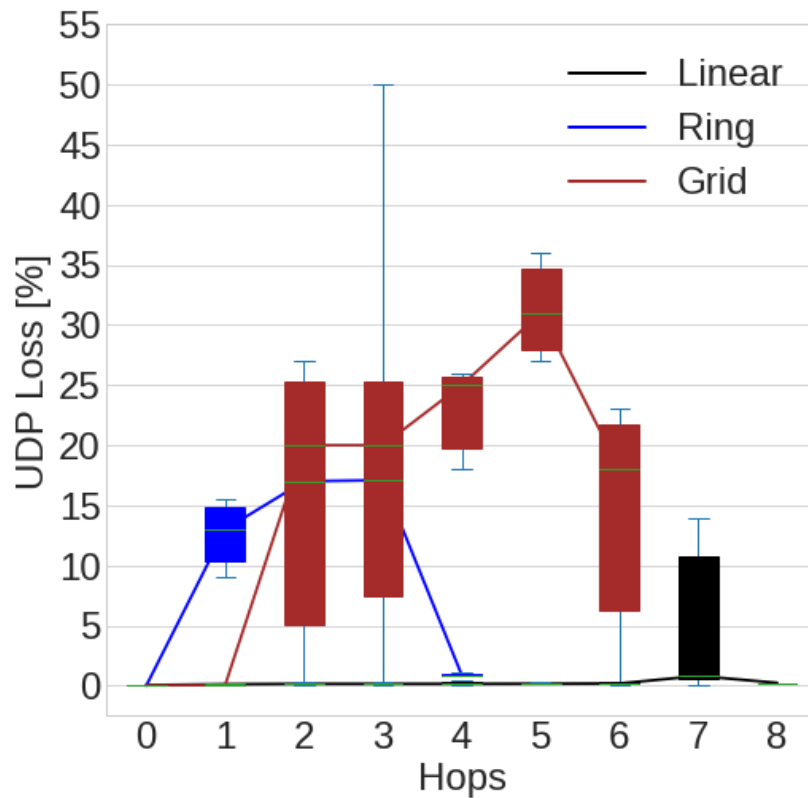


Figure 17: W-iLab1.t- UDP Loss

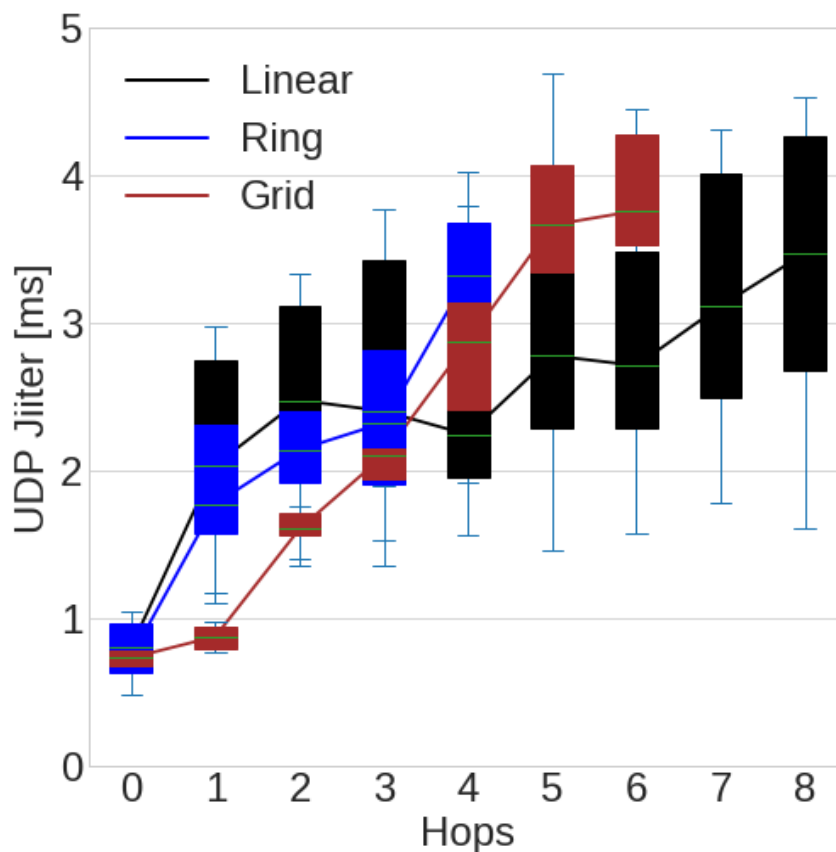


Figure 18: W-iLab1.t- UDP Jitter

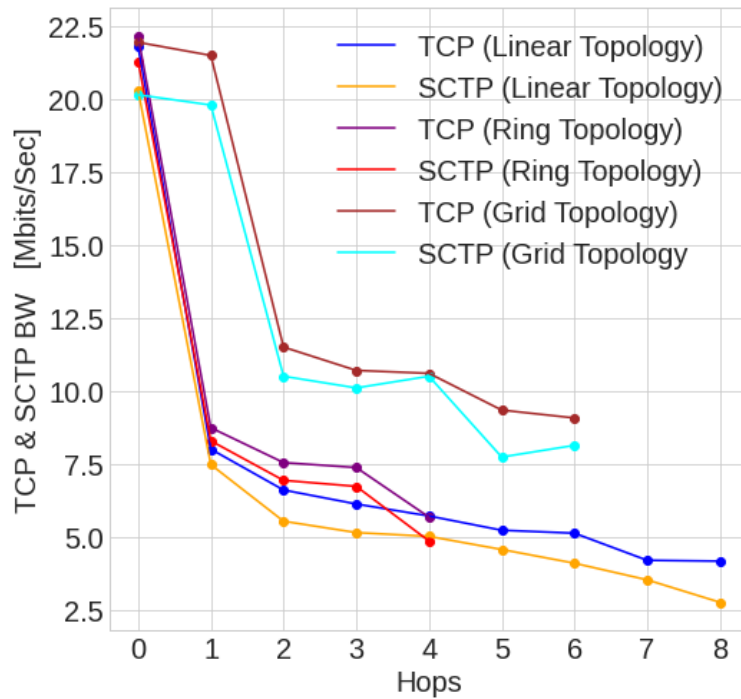


Figure 19: W-iLab1.t- TCP and SCTP Bandwidth

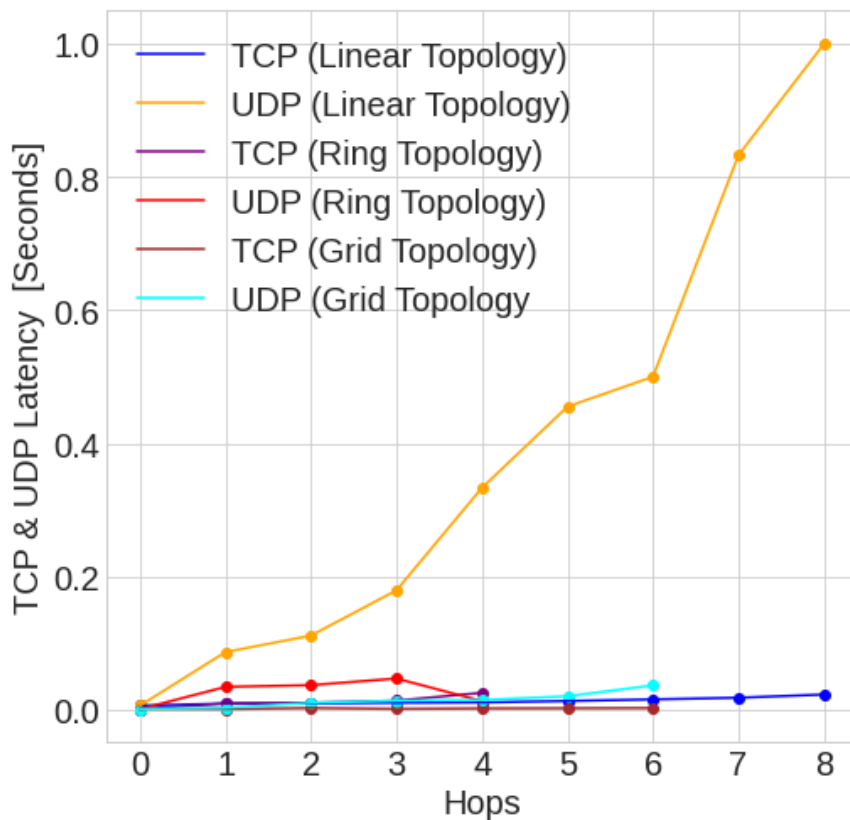


Figure 20: W-iLab1.t- TCP and UDP Latency

W-iLab2.t

The setup for W-iLab2.t is similar to W-iLab1.t. The difference in results is due to different hardware and the placement and proximity of nodes from each other. The



channels used for W-iLab2.t is a little different from W-iLab1.t as not all nodes support all channels.

1. UDP Bandwidth: In W-iLab2.t maximum UDP bandwidth is observed in Ring topology around 25 Mb/s for hop 0 because of the presence of a 5 GHz upper channel connecting node9 to node0. The plot shows a decrease in the bandwidth with the increase in hops for all topologies i.e., linear, ring, and grid. We can see a sharp drop in the bandwidth for hop 7 because of the use of a 2.4 GHz channel after a 5 GHz lower channel as zotac APU which was a node at hop 7 did not support an upper 5 GHz channel refer to Figure 21.

2. UDP Loss: There is no almost loss except for hop 4 and hop 7 but for hop 8 it is slightly increased i.e., around 2 percent in linear topology refer to Figure 22. For ring there is a loss of around 3 percent which is almost constant for all 4 hops. For grid topology, the loss increases from hop 0 until hop 4 i.e., from 1 to 7 percent while it decreases to 2 percent for hop 5 and then increases to 7 percent for hop 6 2.4 GHz channel usage after 5 GHz channel (refer to Figure 8).

3. UDP Jitter: The Jitter is increased as hops are increased and this is the same for linear, ring and grid topology (refer to Figure 23).

4. TCP and SCTP Bandwidth: TCP and SCTP bandwidth decrease as hops increase for linear, ring, and grid topology (refer to Figure 24). For hop2 bandwidth is little increased compared to hop1 because of the use of a 5 GHz lower channel (refer to Figure 8) for the understanding of the setup and relation with results.

5. TCP and UDP Latency: TCP and UDP latency is increased in linear, ring, and grid topology i.e., from 0.0004s to 0.006sec for linear topology, in ring topology the latency is from 0.0005 sec to 0.003s, and in grid topology latency is from 0.0004sec to 0.002sec (refer to Figure 25).

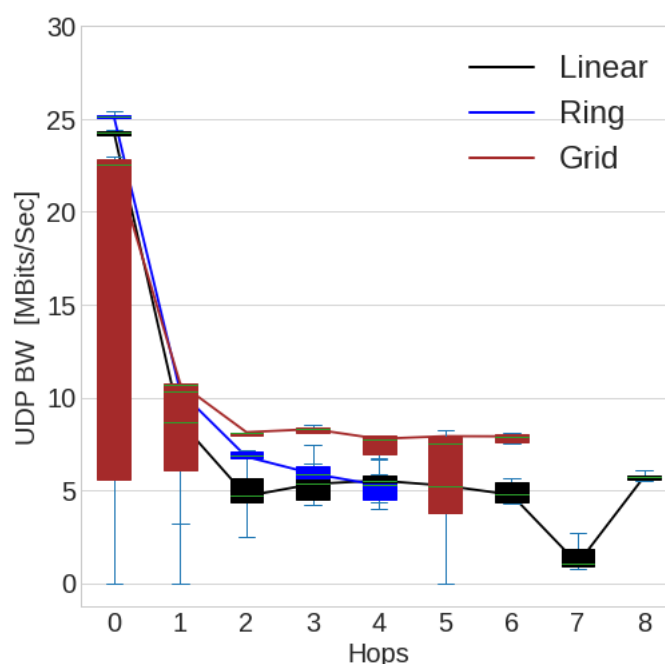


Figure 21: W-iLab2.t - UDP Bandwidth

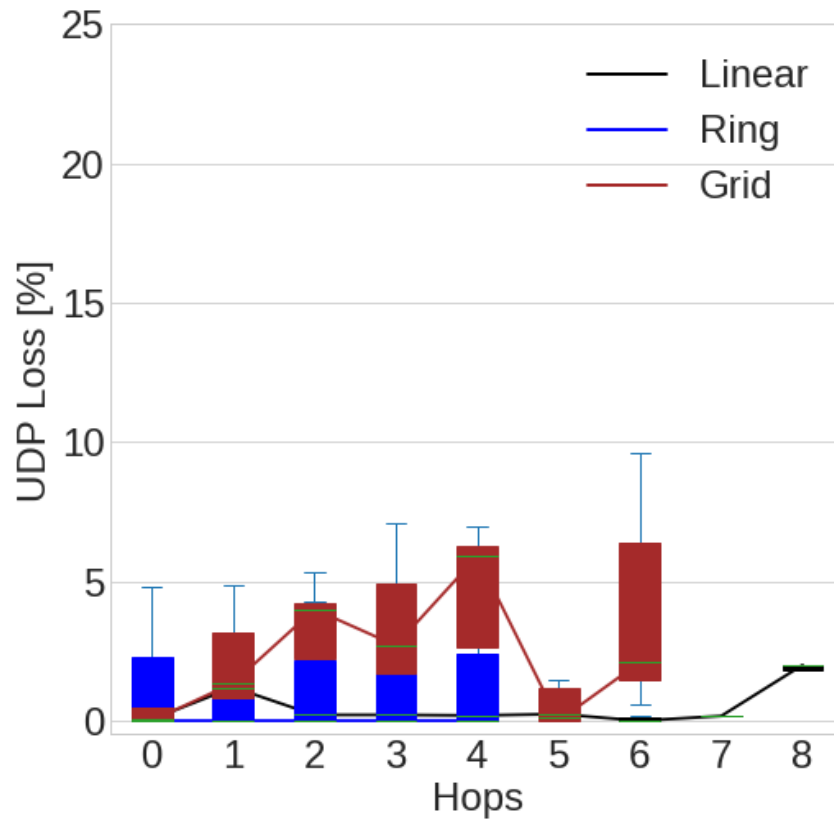


Figure 22: W-iLab2.t - UDP Loss

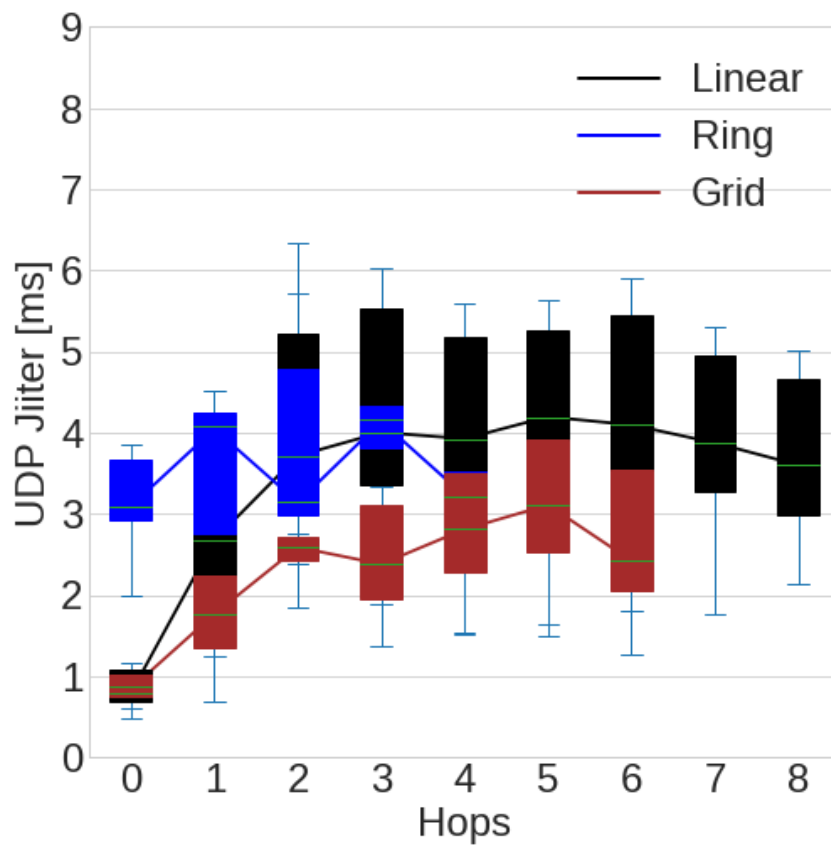


Figure 23: W-iLab2.t - UDP Jitter



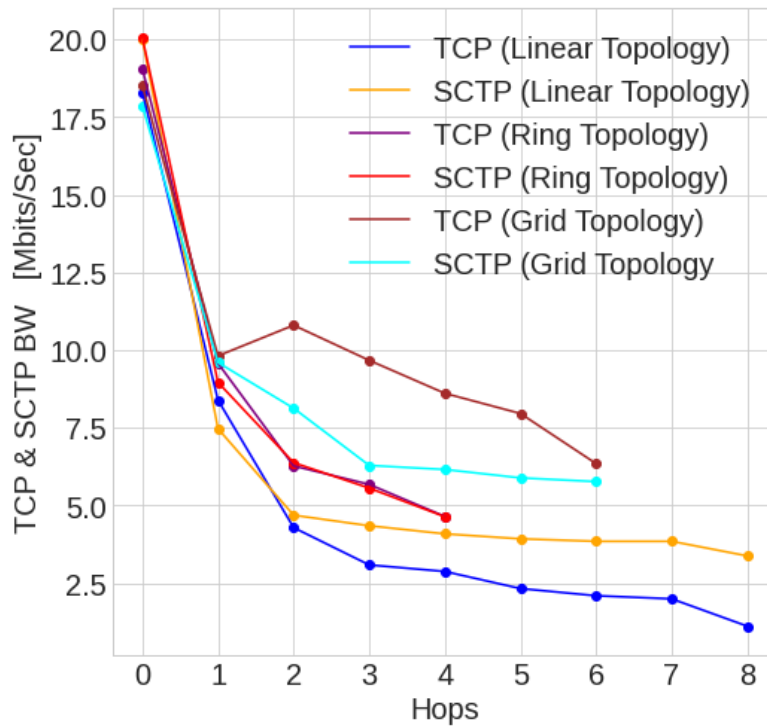


Figure 24: W-iLab2.t - TCP and SCTP Bandwidth

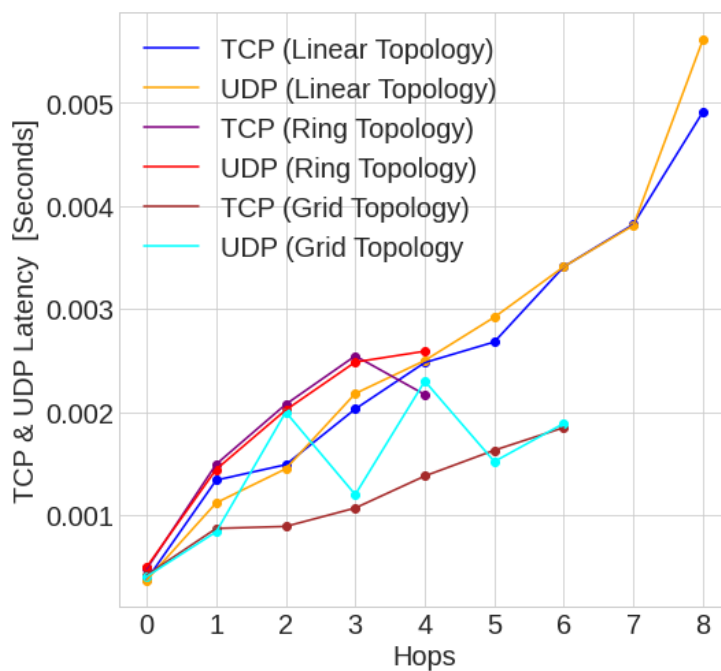


Figure 25: W-iLab2.t - TCP and UDP Latency

CityLab

1. UDP Bandwidth: The UDP bandwidth for CityLab setup shows that the maximum bandwidth is around 27 Mbps for hop 0 and it is decreased to around 13 Mbps for hop 1 and for hop 2 to around 7 Mbps for linear topology. For ring topology, only 2 hops are present on both paths. The results show that for ring topology for hop 0 the bandwidth is around 26.5 Mbps and for hop1 the bandwidth is around 13 Mbps (refer to Figure 26).

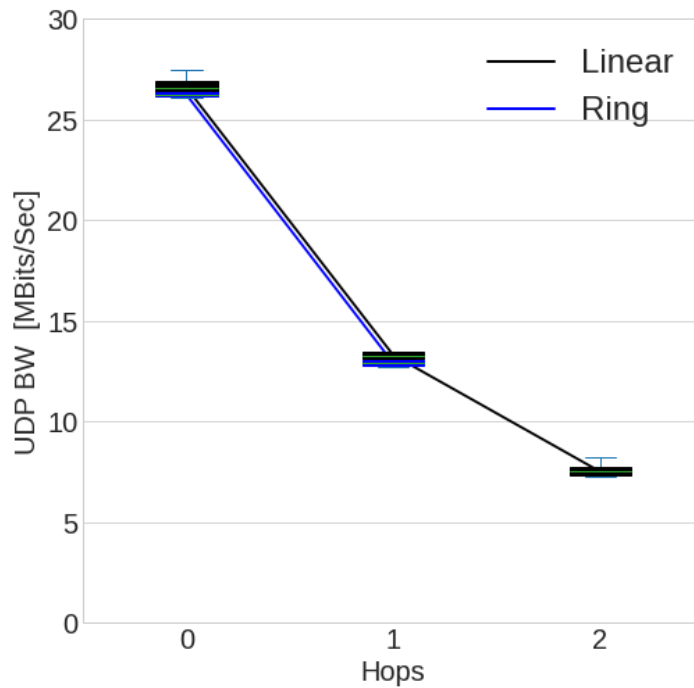


Figure 26: CityLab - UDP Bandwidth

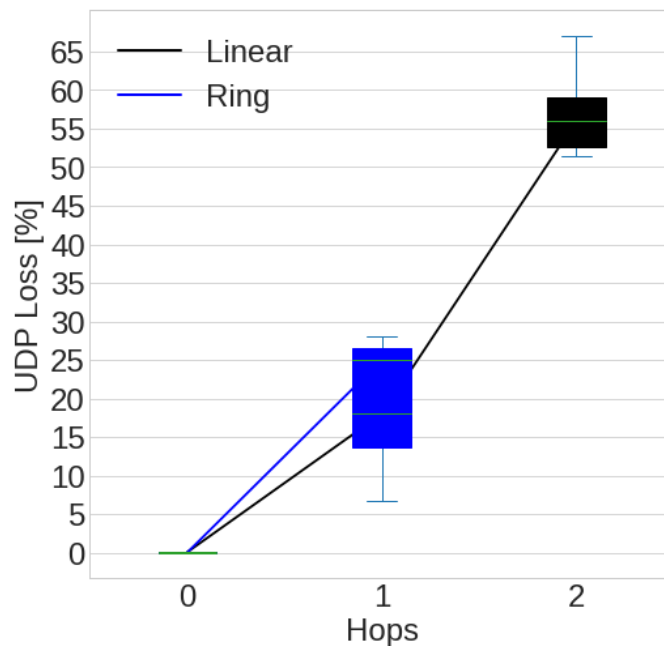


Figure 27: CityLab LAB - UDP Loss

2. UDP Loss: The UDP loss for CITY lab shows that the percentage loss is much more in linear topology than in ring topology (refer Figure 27). In linear topology and ring topology for hop 0, there is no loss whereas for hop 1 in linear topology the loss is increased to around 18 percent and for hop 2 it is around 58 percent. In ring topology for hop 1 loss is increased to around 26 percent. There is no hop 2 in ring topology as it is only 2 hops in both paths.

3. UDP Jitter: The UDP jitter is much more in ring topology than in linear topology refer Figure 28. In linear topology and ring topology for hop 0, the jitter is around 1 ms and around 5 ms respectively, whereas for hop 1 in linear topology the jitter is

increased to around 4.9 ms and 5.5 ms respectively. For hop 2 it is around 25 percent. In the ring topology, for hop 1, loss is increased to around 8 percent.

4. TCP and SCTP Bandwidth: TCP and SCTP bandwidth are more for hop 0 in both linear and ring topology which is around 21 Mb/s for TCP and 20 Mb/s in SCTP for ring topology whereas it is 21.5 Mb/s for TCP and 20 Mb/s for SCTP in a linear topology. For hop 1 TCP and SCTP in linear and ring topology is around 11 Mb/s. Hop 2 exists in linear topology only, and it is 7.1 Mb/s for TCP and SCTP (refer to Figure 29).

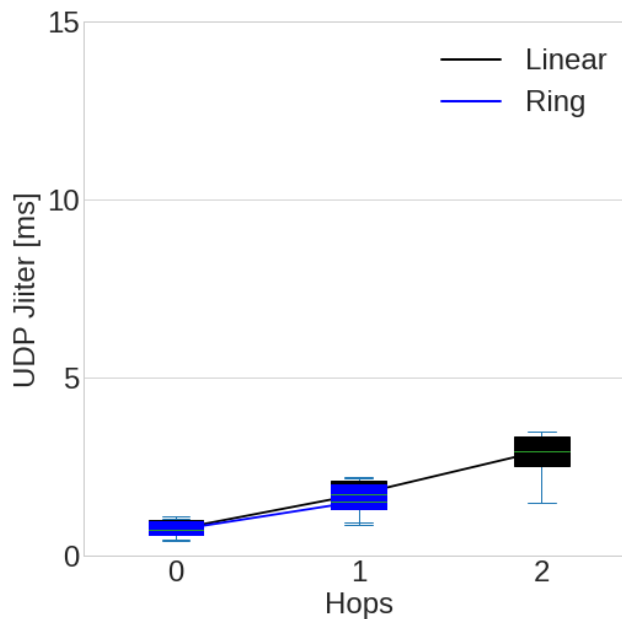


Figure 28: CityLab UDP Jitter

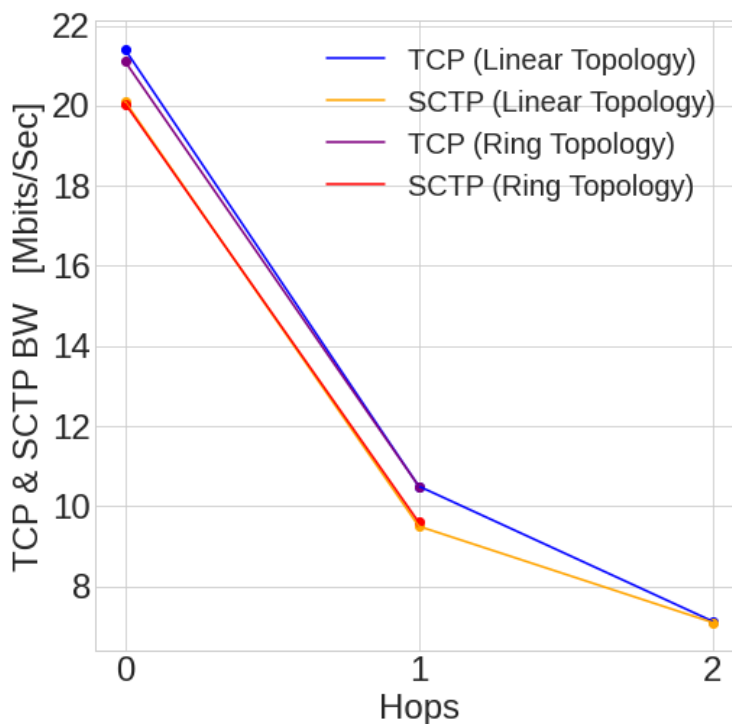


Figure 29: CityLab- TCP and SCTP Bandwidth



Comparison of Different Testbeds

We performed a benchmark experiment on W-iLab1.t, W-iLab2.t, CityLab, and POWDER testbeds. In this experiment, we set up a grid wireless IoT ad-hoc network topology on each of the considered testbeds (shown in Figure 30) and send traffic to find the bottleneck bandwidth of each testbed. We deployed wireless nodes with IEEE 802.11 a/b/g WiFi Network Interface Cards (NIC) and used 2.4 GHz (with non-overlapping channels 1, 6, or 11) and 5.0 GHz bands (with non-overlapping channels 36, 40, or 44) to deploy the topology. Topology is created using the method provided previously. Depending on the availability of wireless nodes at the time of experimentation, we used 9 or 4 wireless nodes to create a wireless ad-hoc network topology. On W-iLab1.t and W-iLab2.t, we used 10 nodes. Further, on CityLab and POWDER, we used 4 wireless nodes to create the topology. Figure 30 shows only the topology with 9 nodes. The topology with 4 nodes is the same as the 9 nodes' topology. The only difference is that it contains 4 nodes. In W-iLab1.t, W-iLab2.t, and CityLab, there are two WiFi ports available with Athero and Intel processors. However, in POWDER, there is only one WiFi port available to create a link. In order to have additional WiFi interfaces, we created virtual WiFi interfaces on top of physical WiFi interfaces.

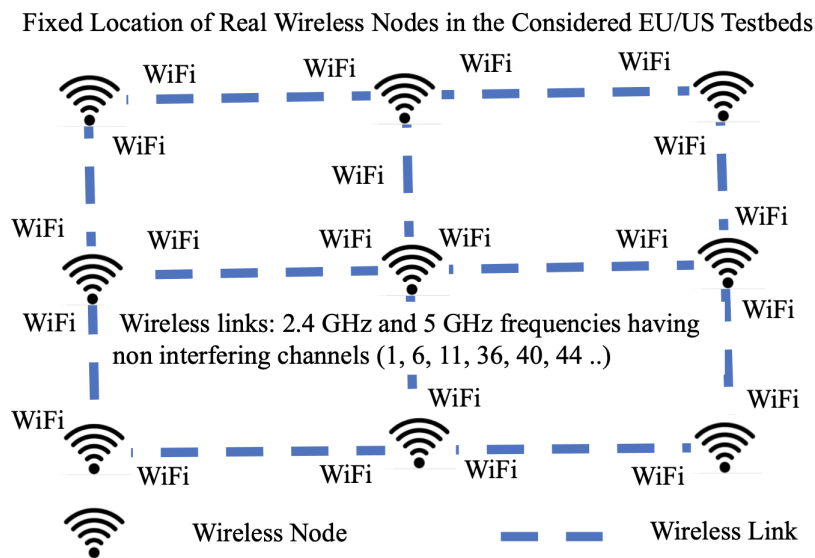


Figure 30: Grid Topology Deployed on EU/US testbeds for Comparison Experiments

After deploying the topology, we configured a private IPv4 address on each WiFi interface in the form of 192.168.X.Y/255.255.255.0 where X and Y are numbered between 1 and 255. Here, both interfaces of a link have the same network address and different link interfaces have a different network address. Further, the Optimized Link State Routing (OLSR) protocol is run on each node to reach any of the other nodes. The OLSR Hello Interval was kept for 2 seconds and Validity Time Interval was kept for 50 seconds. We used Iperf to send and receive Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) traffic from one node to another. The experiments are performed to obtain the bottleneck bandwidth and OLSR failure recovery time of our deployed topology on each testbed. For the failure recovery experiment, we failed one of the links of our wireless topology by disabling one of the

WiFi interfaces and obtained the OLSR failure recovery time. This is the time difference between the time the link failed and the time the receiver started receiving all the traffic even though the link failed.

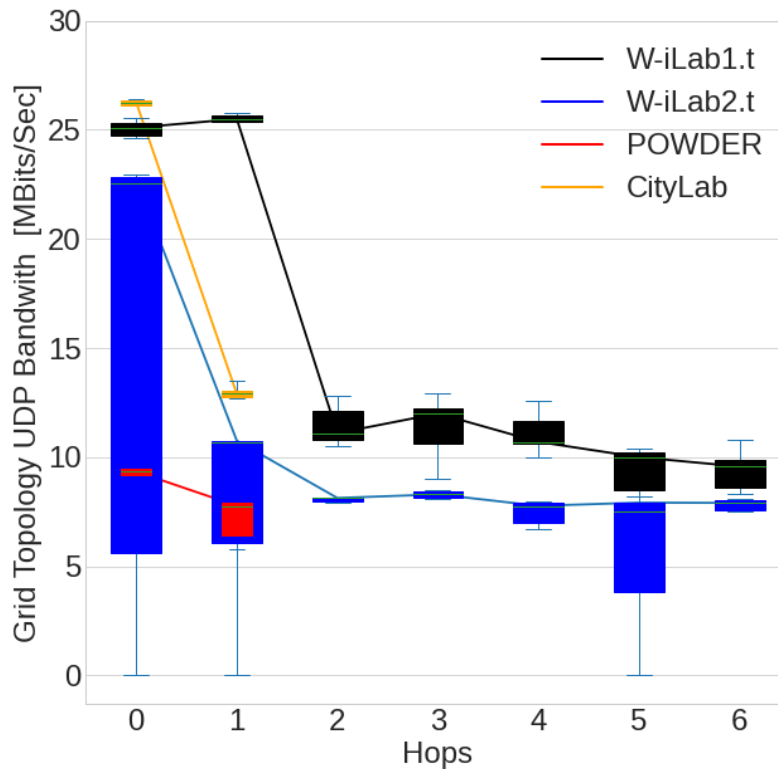


Figure 31: UDP Traffic Bottleneck Bandwidth

Figure 31 shows the UDP bottleneck bandwidth with respect to the number of hops traveled in each testbed. The UDP bottleneck bandwidth refers to the bandwidth after which the reception or transmission of UDP data is not possible due to the limited bandwidth of WiFi interfaces. The 0 hop in Figure 31 means that the wireless nodes are directly connected to each other. The n-hop ($n > 0$) means that wireless nodes send traffic through n nodes in the network. For CityLab and POWDER, we do not see the value of the bottleneck bandwidth at hop greater than 1. This is because there were only four nodes available for experimentation in these testbeds and the nodes are up to 1-hop away from each other. The maximum bandwidth available was 25 Mb/s. We also see that the bottleneck bandwidth of the topology in the POWDER testbed is lower than in any other testbeds. This is because only one WiFi interface is present in POWDER and therefore, we had to create more virtual interfaces in POWDER on top of physical interfaces to create the topology. This added extra overheads, which resulted in lower bottleneck bandwidth. Further, as the number of hops traveled increased, the bottleneck bandwidth decreased. This is because the intermediate nodes add extra overheads. Figure 31 also shows that the deployed topology in W-iLab1.t has a higher performance than W-iLab2.t. This is because the available nodes in W-iLab1.t were APU based which contained quad-core processors with 8GB RAM. However, the available nodes in W-iLab2.t were a combination of APU and ZOTAC which contained dual and quad cores with 4 and 8GB RAM.

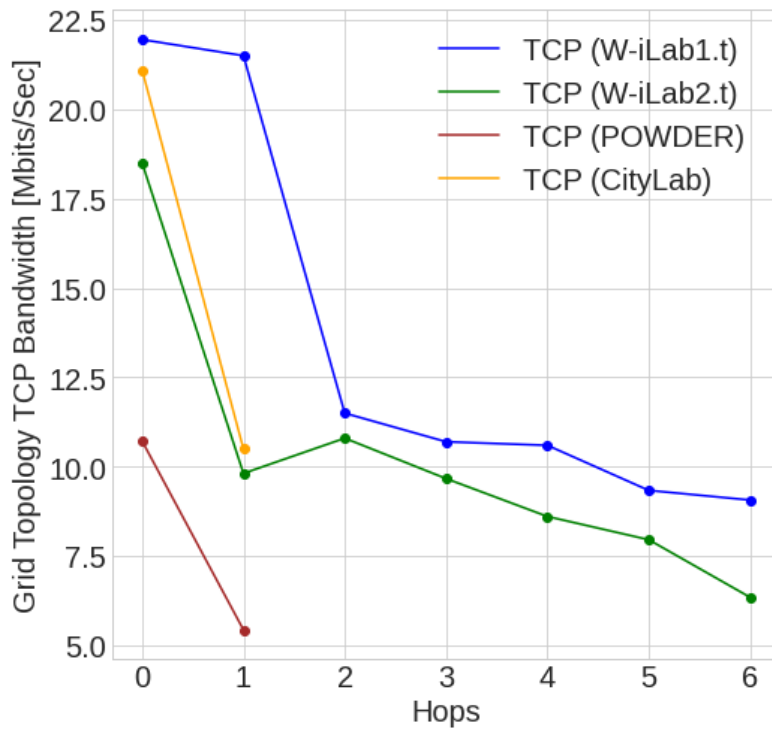


Figure 32: Average TCP Bottleneck Bandwidth

Figure 32 shows TCP bottleneck bandwidth when TCP traffic is transported over a network. While a sender node waits for an acknowledgment in TCP, its maximum bottleneck bandwidth is less than the maximum bandwidth obtained with UDP traffic (e.g., in Figure 4). In this case, the maximum bandwidth obtained was 22 Mb/s. Additionally, the POWDER testbed bandwidth is lower than any of the other testbeds. Moreover, W-iLab1.t provided the highest bottleneck bandwidth. Figure 32 also shows that the bottleneck bandwidth decreases as the number of hops increases.

4.1.2 Automatic Configuration of SDN

Task Summary

The objective of this task is to automatically configure OpenFlow in a wireless ad hoc IoT network topology deployed on different testbeds. The choice of OpenFlow is already considered for wireless networks in [1][2]. We consider an ad hoc network scenario in which the controller is directly reachable to only a few wireless IoT devices in a network. This research addresses the challenge for IoT devices that cannot directly reach the controller and where they have to find a path to the controller through other devices in the network.

The project utilizes an automatic configuration method for the deployment of SDN in a wireless ad-hoc network proposed previously by us [1]. Previously, this method was tested in a wireless network created using mininet [3] which deploys a virtual OpenFlow network. The purpose of the work done in this project is to demonstrate how SDN/OpenFlow can be configured automatically using the above method in real testbed settings. In our validation testing, we first used the jFed tool [4] and the w-iLab.t testbed to demonstrate the working of this method. This work has been accepted for demonstration at the IEEE LANMAN 2022 conference and the video demonstration of our work can be found at [5].



In the Fed4Fire's testbed, we configured OpenFlow using two steps. Below are the steps:

1. Deployment of a wireless topology on a testbed

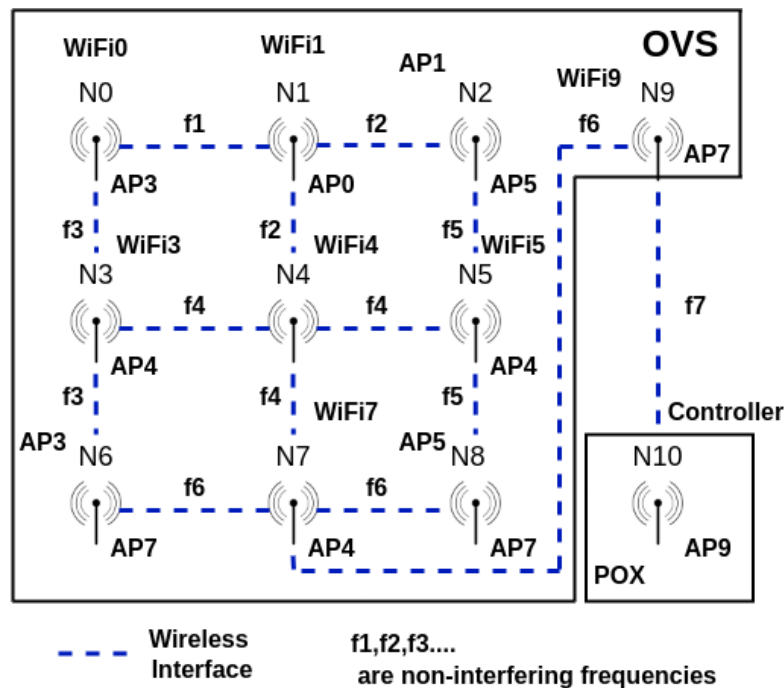


Figure 33: OpenFlow Topology Setup

With most testbeds, including Fed4Fire wireless testbeds, the nodes are positioned in fixed locations and most of them can be directly reachable to each other. Consequently, the creation of a multi-hop topology is not possible on these testbeds without tweaking the parameters of the links. In this demo, we created multi-hop links by using different frequencies (f1, f2, f3 .. etc. as shown in Figure 33) of channels. This scenario is discussed in [6] (or in the task discussed above). In this scenario, access points (AP) with different frequencies, as shown in Figure 1, are used to create different links. For example, in Figure 33, in order to send traffic from node N0 to N2 through node N1, access point AP3 sends traffic to AP0 using frequency f1 and then AP0 sends to AP1 using frequency f2. Further, a unique IP address is assigned to each WiFi interface used in our topology.

We created a grid topology using the above scenario where 10 nodes (nodes N0 to N9) are OpenFlow switches and node N10 is the controller. In this setup, Open vSwitch is used to deploy an OpenFlow switch and POX is used to deploy the controller node.

2. Automatic Configuration of OpenFlow

The problem to configure OpenFlow in the WiFi interfaces was that the links created using the mechanism provided in the previous subsection stop working once they are added to the Open vSwitch. This is because Open vSwitch works based on the IEEE 802.3 MAC protocol, while WiFi links work based on wireless MAC protocols, such as IEEE 802.11. In order to transport traffic from Open vSwitch to WiFi interfaces, we created GRE tunnels, as described in [1]. All the creation of GRE tunnels and addition of the controller information are done automatically using the method provided in [1]. Further, using the above method, the OLSR (Optimized Link

State Routing) protocol is used to route the traffic (i.e., control traffic) between the controller and switches and the OpenFlow protocol is used to forward the traffic (i.e., data traffic) between switches. The steps that are executed in our automatic configuration method are below:

- a. All nodes (N0 to N10) run OLSR.
- b. All nodes except the controller node run Open vSwitch.
- c. Node 10 runs the POX controller.
- d. Once a switch detects a neighbour using the OLSR, it adds a GRE tunnel to reach the neighbour using the OVS-DB protocol. %which is a Open vSwitch Data Base Management protocol (RFC 7047).
- e. Once the controller detects a new switch using the OLSR, it adds its own IP address, transport-layer protocol, and port number to the switch using the OVS-DB protocol.

Using the above mechanism, OpenFlow is configured automatically.

Results and Analysis

We demonstrate the creation of a wireless grid topology and automatic configuration of OpenFlow using wireless NUC2014 nodes of W-iLab1.t. We configure wireless links and IP addresses by using a multi-command terminal in jFed. Furthermore, we then demonstrate the automatic configuration method of OpenFlow by displaying the OpenFlow sessions established by the POX controller.

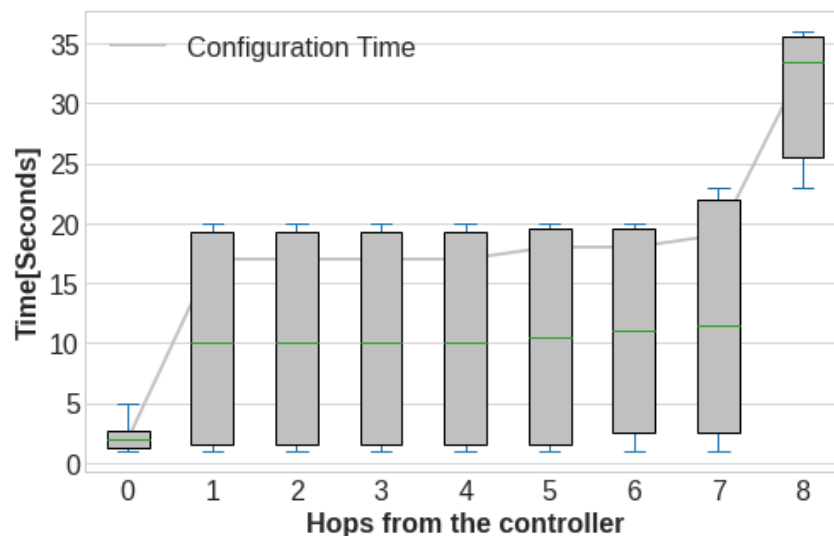


Figure 34: Automatic Configuration Time

Figure 34 shows that the controller is also able to create sessions with all OpenFlow switches including the nodes that are not directly connected to it (nodes N0 to N8 in Figure 33). The results will be collected for the amount of time taken by the controller to connect to all OpenFlow nodes and to configure the GRE tunnels. The number of hops depends on the path taken by the controller to reach an OpenFlow node. Figure 2 shows the results of configuration time i.e., time taken by the controller node to detect OpenFlow switches, set up the routes, ports, GRE tunnels, and corresponding IP addresses. The nodes located at hop 0 have the lowest automatic configuration time, as these nodes are directly connected with the controller. The automatic configuration time from hop 1 to 7 is approximately the same in our topology. Further, hop 8, has significantly more configuration time. These results are different from

those reported in [1], where we observed a linear increase in the automatic configuration time with respect to the number of hops increased. This difference is observed, as this setup is run in the testbed and previously, it was run using mininet.

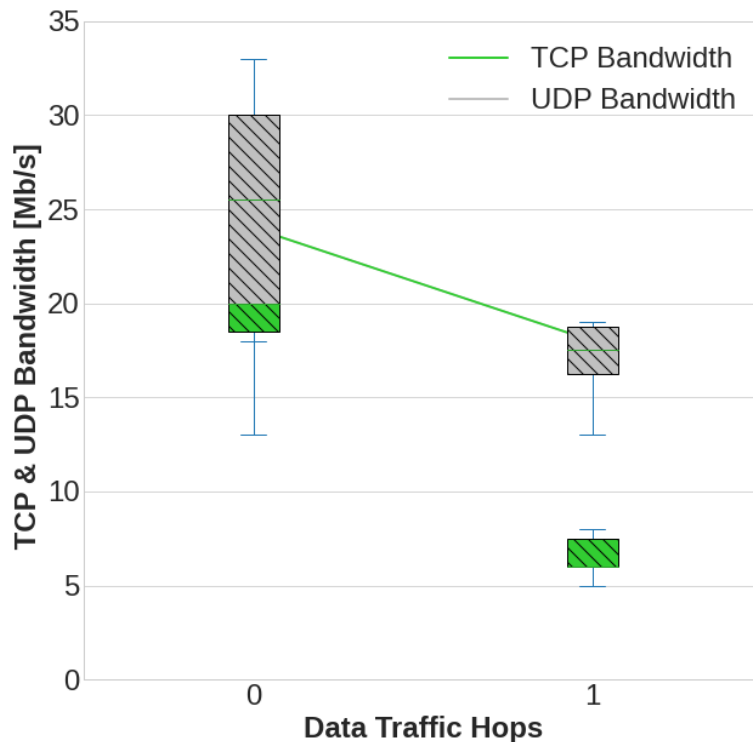


Figure 35: TCP/UDP Bandwidth Utilization

Figure 35 shows the TCP/UDP bandwidth utilization for data traffic by measuring the bandwidth using iperf. The results show that UDP bandwidth is higher than the TCP bandwidth. This is due to the fact in TCP, the sender has to wait for the acknowledgments.

4.1.3 Failure Recovery Results and Analysis

This subsection reports the failure recovery functionality of our implemented functionalities. We divide this section into the following:

- 1) Failure Recovery using OLSR running on different topologies at different testbeds
- 2) Failure Recovery of OpenFlow

Failure Recovery using OLSR



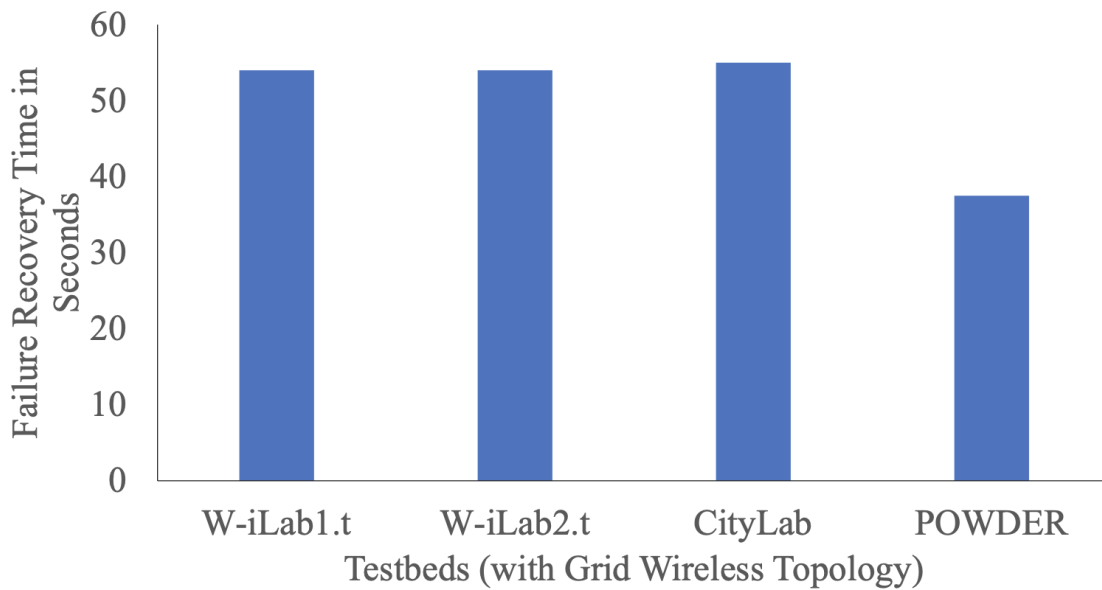


Figure 36: Average Failure Recovery Time in Seconds

Figure 36 shows the failure recovery time in seconds when one of the links in the topology (ring and grid topology, as described in section 4.1) is failed and all the traffic from that link is redirected to a failure-free path. It shows that the failure recovery time of W-iLab1.t and W-iLab2.t is approximately the same, i.e., around 52 seconds. However, the failure recovery time of the CityLab testbed is around 55 seconds. Further, the failure recovery time of the POWDER testbed is as low as 37.5 seconds. In our experiments, the validity timeout is 50 seconds, which means that OLSR detects the failure 50 seconds after the failure is induced in the network. Therefore, the failure recovery time in W-iLab1.t, W-iLab2.t and in CityLab is greater than 50 seconds. However, it is significantly low in POWDER, as we obtained more packet loss in POWDER. This led to the detection of the failure earlier than expected. This resulted in a lower failure recovery time in the POWDER testbed.

Failure Recovery Time of OpenFlow

We calculate two results for OpenFlow Failure Recovery with Grid topology on w-ilab1.t and w-ilab2.t: (1) Control Traffic Failure Recovery and (2) Data Traffic Failure Recovery. In the case of the control traffic failure recovery, the control link is broken, and the failure recovery is calculated, whereas in the case of the data traffic failure recovery, OpenFlow data traffic link (i.e., GRE tunnel) is broken and the data traffic failure recovery time is calculated.

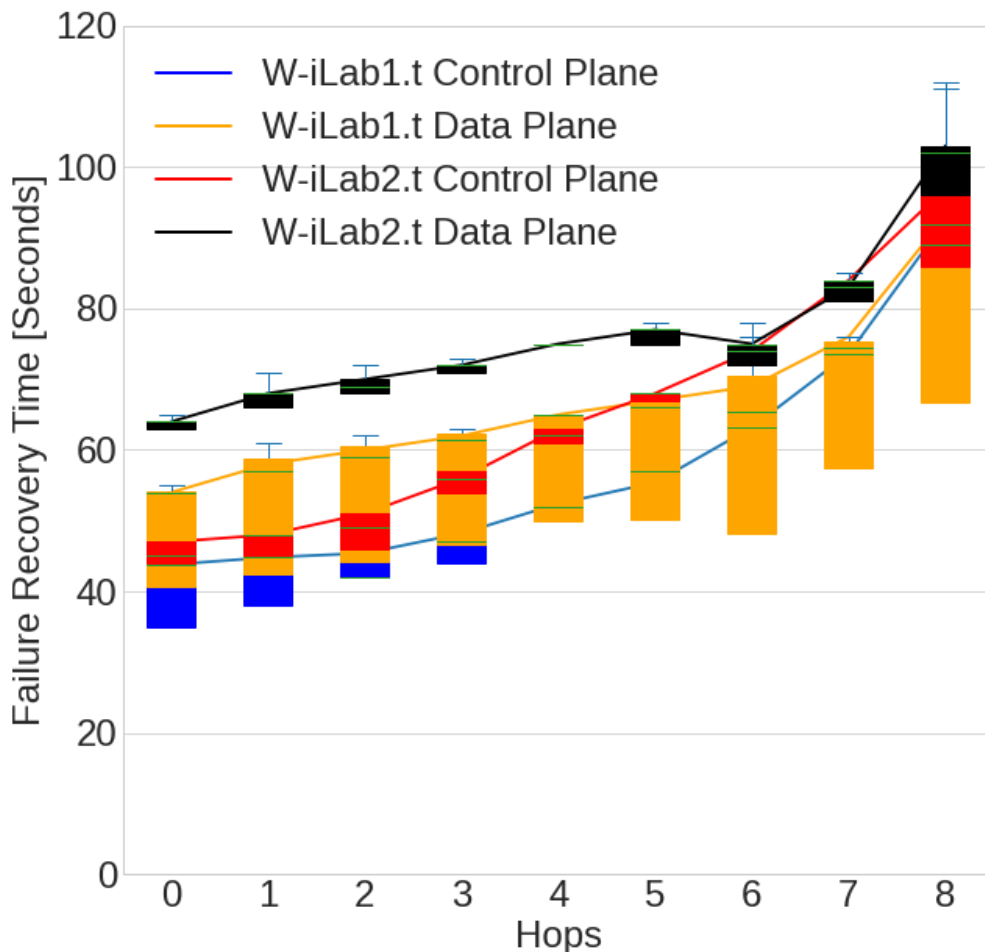


Figure 37: Control & Data Plane Failure Recovery Time

Figure 37 shows the control plane and data plane failure recovery time with the number of hops traveled for w-ilab1.t and w-ilab2.t testbeds. For the first hop, the control plane failure recovery time is more than 40 seconds and for the next hops, it increases with the hops increased. When a failure happens, OpenFlow detects the failure after an ECHO timeout, and OpenFlow switches get disconnected from the controller. However, once OLSR detects a new failure-free path from a disconnected switch to the controller, the controller again gets connected with the controller. As the number of hops increases, OLSR takes more time to set up the paths, therefore, we see an increase in the failure recovery time with the increase in the number of hops.

Figure 37 also shows the data plane failure recovery time when the GRE link is broken. It shows a data plane failure recovery time of more than 55 seconds. This is because the idle timeout of the forwarding entries is 60 seconds, which means that if a forwarding entry is idle for 60 seconds, it will be deleted and then the new entry will be established once traffic is received. In our experiments, traffic is transmitted after regular intervals. Therefore, data traffic failure recovery time is close to 60 seconds from hop 1 to hop 7. However, for hop 8 and hop 9, as more flow entries need to be established, we observed a longer data traffic failure recovery time.

4.1.4 Data Collection and ML-based Path Selection

Task Summary



To collect the data used in the ML algorithm we use OpenFlow messages and Pox controller (<https://noxrepo.github.io/pox-doc/html/>).

This subsection describes the ML algorithm for path selection.

The ML algorithm is based on a Graph Neural Network and Deep Q-Network framework. Specifically, we focus on message-passing GNN that has been used to solve routing optimization problems in optical networks.

The data collection and route selection algorithm is developed in different steps that can broadly summarized as follows:

1. Setup of the environment: in this step, the variables are initialized, and the desired network is recreated via networkx Python library considering the number of nodes and edges, representing the wi-Fi links. The network representation represents the state of the DQN framework. Initially, the nodes are assigned nominal WiFi values of capacity.
2. The algorithm starts its training given a random demand to be satisfied, an initial and a destination node. The algorithm computes the K=2 shortest paths based on the number of hops.
3. In the third step, the algorithm interacts with the environment after choosing the path based on an epsilon-greedy choice.
4. After the action selection, the algorithm updates the state graph features, collecting data from the environment. Via API messages, the algorithm collects data measurement from the POX controller node with OpenFlow messages. At the present stage of the project, we collect the data loss using the last and first node of the selected path.

Testbed Preparation

We tested the algorithm on the wilab-1 and wilab-2 testbed configuration as for Section 4.1.2. To make the algorithm run on the real testbed, an initial preparation step is required. In this step, the best Python library versions have to be found and installed to ensure compatibility with the Tensorflow and Keras features. As a consequence, a virtual environment with Python 3.8 has been created in each controller node. While pox has been originally developed with Python 2 versions, this approach avoids the user migrating the RL code to older and unsupported versions of Tensorflow and Python. Here below, we list all the necessary requirements.

Requirements:

Python==3.8

Keras==2.4.0

Networkx==2.6.3

Matplotlib==3.5

Numpy == 1.21.4



Tensorflow==2.7.0

Tensorboard==2.7.0

Network Representation

To ensure the generalisation of the proposed approach, we tested our algorithm on different testbeds and network topologies, as described in the previous section. In order to do that, we use the networkx python library (<https://networkx.org/>), that is able to represent any network topology. In Fig. 38 we show an example of the W-ilab1.t network topology representation in Networkx.

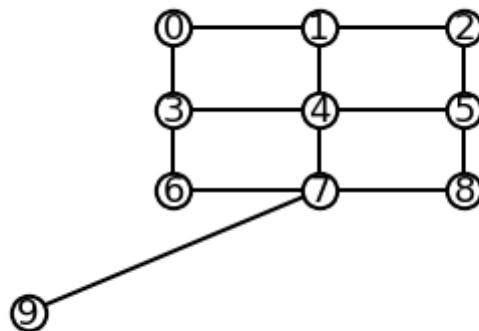


Fig 38: W.ilab.t grid network representation in Python as input of the network

Algorithm description

In this paragraph, we describe in detail the algorithm steps, summarized in Fig. 39.

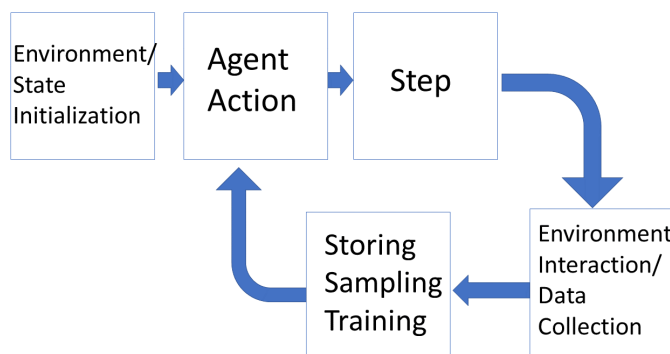


Fig 39: Steps of the proposed DQN+GNN approach for route optimization.

1. At the beginning of the algorithm, the network representation is taken as a state of the DQN algorithm and initialized with some features: available bandwidth and nominal capacity, corresponding to the max theoretical WiFi data rate of a corresponding testbed node. Thus, at the beginning of the algorithm iterations, we consider having as prior knowledge only the topology of the network, as a result of the testbed configuration following section 4.1. In

addition, the algorithm initializes the initial and destination nodes, and the capacity to be allocated.

2. In a second step, the algorithm starts the training, based on the current information, computes the K-shortest paths and allocates the capacity by choosing one of the shortest paths with an epsilon-greedy strategy. The GNN measures the q-value function taking as input a graph with node and edge features and computes a function that depends both on the features and the graph structure. The objective of the GNN is to estimate the q-value $Q(s,a)$ of applying a routing action for the current traffic demand and network state. When the DRL agent allocates a demand, it receives an immediate reward being the normalized allocated bandwidth.

3. The capacity is allocated in the selected path and the agent starts interacting with the environment to optimize the path selection. In this step, the algorithm communicates with the nodes using Openflow messages. In particular, we use OpenFlow PortStatsReply messages to monitor the number of bytes transmitted and received at each port connected to the controller. Specifically, we poll flow statistics from the first and last switch of each path in order to determine the available bandwidth. We derive the available bandwidth by subtracting the bytes an as output of the port switches from the nominal WiFi speed. Using this measurement, we compute the latency of the selected path and check if the requirement is respected. It is also possible to obtain an accurate measurement of the packet loss from the port statistics message, but this option is left for future work. If the demanded capacity has been allocated, a new pair of source and destination is chosen.

4. The memory is filled with a tuple [state, action, reward, new_state] and the DQN network is trained by sampling from the replay memory. The circle restarts until the demand capacity is finished, or the latency requirement is not satisfied

Algorithm Results

The algorithm is trained for a minimum of 800 episodes in the available testbeds. Preliminary experiments were carried out to choose the appropriate hyperparameter values for our DQN+GNN agent. In particular, we evaluated the effect of the memory size on the training time of the algorithm. We use a learning rate of 0.0001. We select 25 hidden states for the neural network with a batch size of 32 and $T=8$ message-passing steps. The optimizer used is a Stochastic Gradient Descent.

The ϵ -greedy exploration variable is initialized $\epsilon = 1$ and decreased each 10 episodes.

We train the algorithm on the grid network topology represented in Figure 38, with 10 nodes plus a controller connected to node 9. We intentionally excluded a ring and star topology since the valid candidate paths would be very limited.

The links between the nodes are WiFi links with theoretical max speed typical of 802.11 a/b/g protocols. The link capacity is initialized with nominal rates between [1-54 Mbps]. To simulate a real IoT scenario where the WiFi links are utilized for



different purposes, we launch iperf sessions with different throughputs on the links. We consider a latency requirement of 10ms. The episode is considered successful if the demanded bandwidth is allocated and the latency requirement is respected, and unsuccessful otherwise.

In Figure 40 we show the stability of the proposed algorithm, considering a number of 800 episodes. The reward is averaged over 100 episodes.

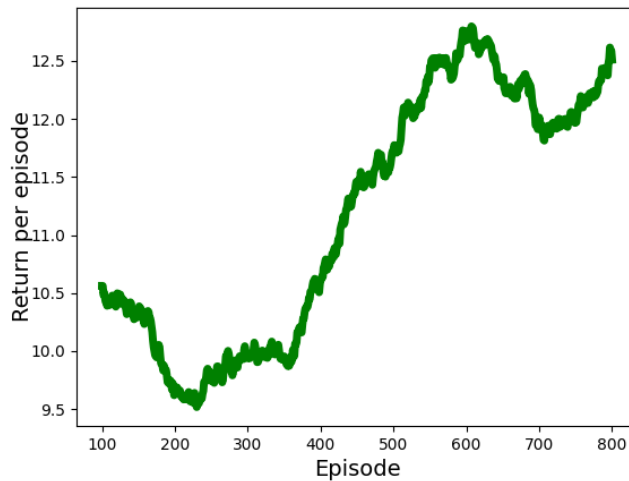


Figure 40: Reward behavior: after an initial oscillation, the reward function has a stable behavior.

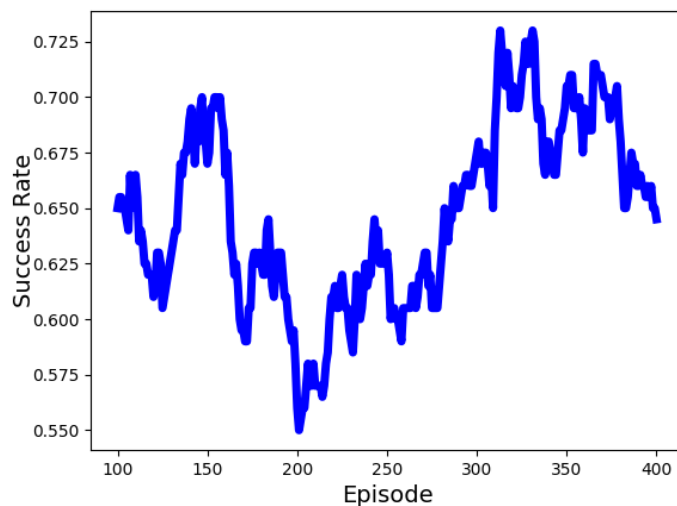


Figure 41: Success Rate

In Figure 41 we present preliminary results on the success rate of the algorithm. We run 400 episodes and compute the performance of the algorithm in terms of successful episodes. We recall that an episode is successful if the demand bandwidth is allocated and, at the same time, the latency requirement is respected. This figure reveals the ability of our agent to reach a good performance after an initial period of training.

4.1.5 Inter-testbed Scalability Experiments and Results

Table 2: Hardware Resources Used in the inter-testbed experiment

Testbed	Name	Location	CPU	RAM (Memory)
CityLab	Node6	Middelheimlaa, Antwerpen	AMD GX-412TC 1GHz Quad-core	4GB
	Node18	Vekestraat, 2000 Antwerpen		
W-ilab1.t	Nuc03	Zwijnaarde, Ghent	Intel i5-4250U 1.3GHz Quad-core	8GB
	Nuc04			
W-ilab2.t	NucX3	Zwijnaarde, Ghent	Intel i5-4250U 1.3GHz Quad-core	8GB
	NucX4			
Virtual Wall	n1011-06	Zwijnaarde, Ghent	Intel i5-9400, 2.9GHz, 6 Core	32GB
	n1013-01			
POWDER	Nuc3	Salt Lake, Utah	Intel i5-5300U, 2.3GHz, Quad-Core	8GB
	Nuc4			

Our inter-testbed experiment deploys a mix of wireless/wired nodes using bare-metal machines in CityLab, w-ilab1.t, w-ilab2.t, virtual wall, and POWDER testbeds (Table 2). We selected the testbed's nodes based on their availability at the time of experimentation. We installed Ubuntu 18.04 on each node and collected CPU and memory information. Table 2 depicts this information along with location information which is available on the testbed's website [9,10]. The results in this subsection are averaged over 50 readings. This work has been published at IEEE Networking Letters [11].

1. Availability of Public IPv4 and IPv6 addresses

Figure 42 illustrates if the selected nodes at each testbed have public IPv4 or IPv6 addresses. The results are calculated using a Linux command (such as ifconfig). Figure 42 shows that the nodes of w-ilab1.t and w-ilab2.t do not have public IPv4 addresses. Additionally, while public IP addresses are not assigned by default in the virtual wall, public IP addresses can be requested as a separate resource. In fact, w-ilab1.t, w-ilab2.t and Virtual Wall create private IPv4 networks using private IPv4 addresses to establish a network requested by the user. Additionally, as stated earlier, a public IPv4 address can be requested at a virtual wall node. However, there are limited public IPv4 addresses available at the virtual wall. In the two virtual wall testbeds in Fed4Fire, the first virtual wall has 47 public IPv4 addresses available and in the second virtual wall, there are 53 public IPv4 addresses available. Moreover, nodes in the POWDER testbeds have public IPv4 addresses by default. Further, all the nodes except the nodes in the powder testbed have public IPv6 addresses.



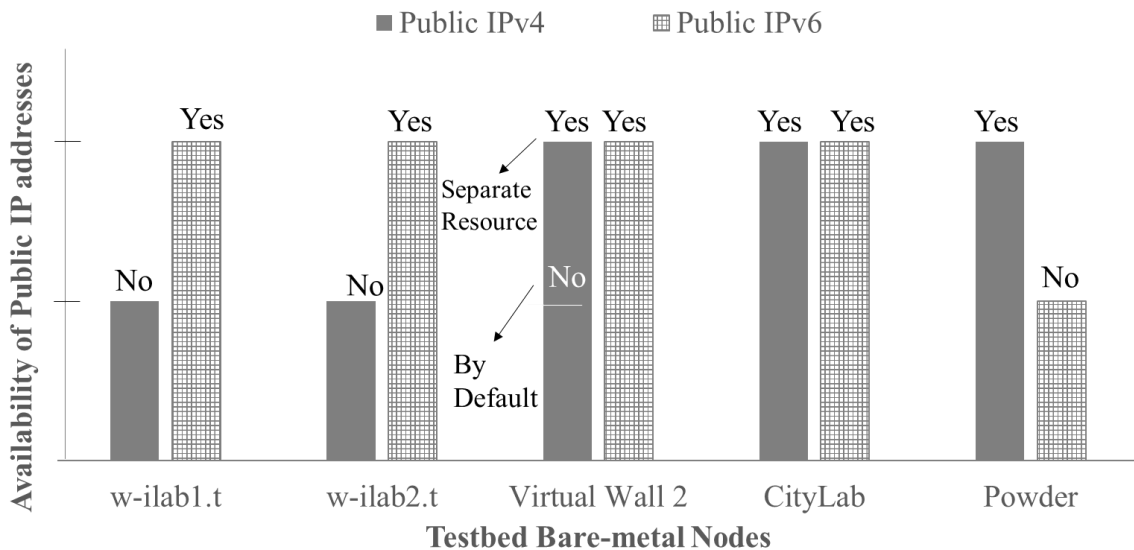


Figure 42: Availability of Public IP addresses at the nodes of different testbeds

2. Topology of our inter-testbed experiment

Figure 43 shows the discovered topology of our inter-testbed experiment by running commands such as mtr and traceroute at the nodes of the testbeds. w-ilab1.t, w-ilab2.t, and virtual wall testbeds are connected via a router (routerG) located at Ghent University. Moreover, w-ilab1.t, w-ilab2.t, and virtual wall are reachable via their private IPv4 addresses through this router, meaning they share a private network (i.e., 10.x.x.x). Additionally, routerG is connected to the Belnet network, which connects external educational institutions, research centres, scientific institutes, and government centres in Belgium. CityLab is also located in Belgium, so the Belnet network is directly connected with CityLab through a router at Antwerp University (RouterA), which connects CityLab to the Belnet network. Moreover, we found that Belnet is directly connected to an independent US network called Internet2, which is dedicated to research and education. Further, there is a direct connection between Internet2 and the POWDER testbed.

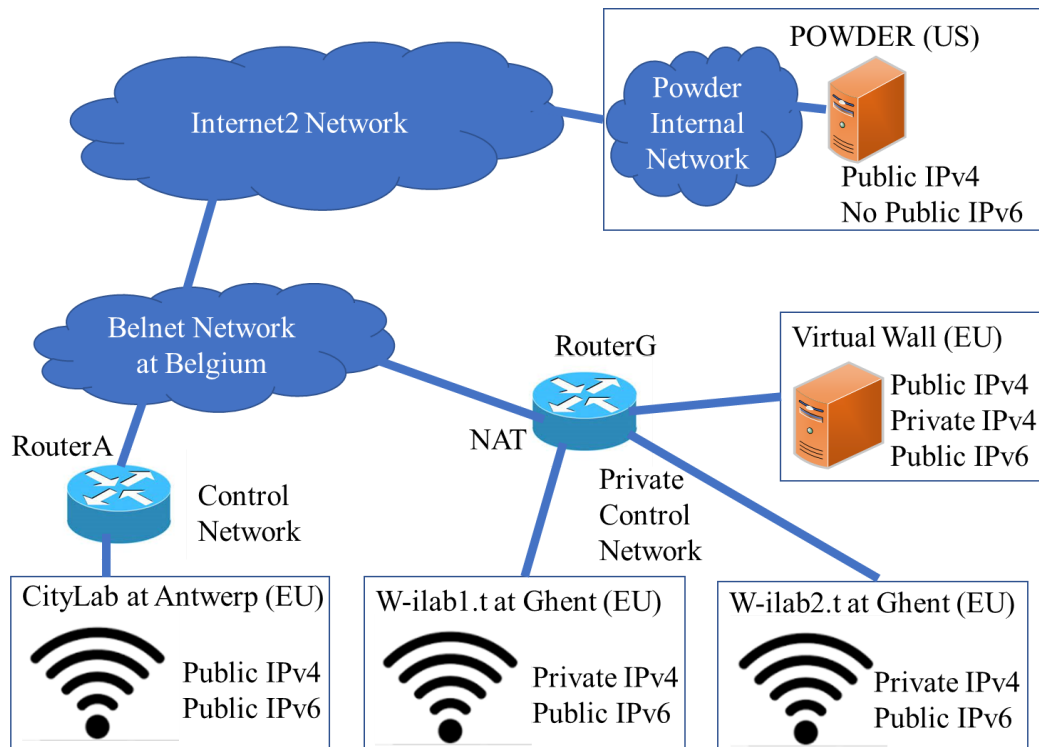
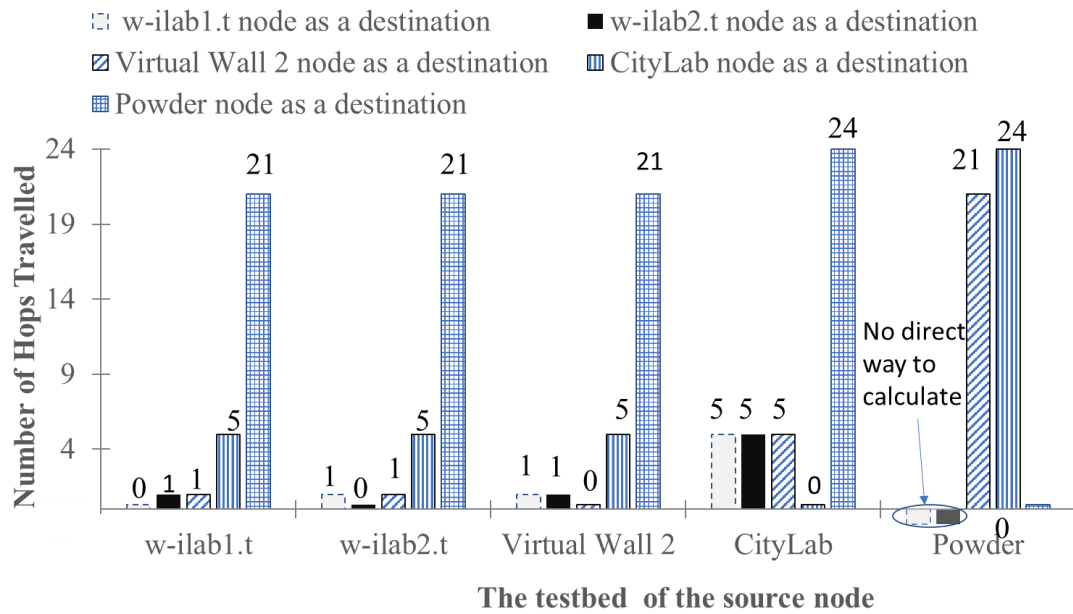


Figure 43: Discovered Topology of our inter-testbed experiment

The following are the further learning from the above experiment:

1. Communication between virtual wall, w-ilab1.t, and w-ilab2.t nodes is possible through private IPv4 addresses assigned to them.
2. CityLab, w-ilab1.t, w-ilab2.t, and virtual wall nodes are reachable to each other through public IPv6 addresses.
3. Communication between CityLab and the POWDER testbed's nodes is possible only using public IPv4 addresses. It is not possible to use public IPv6 addresses, as POWDER nodes do not have public IPv6 addresses.
4. As nodes at the POWDER testbed just have only public IPv4 addresses and public IPv4 addresses could be requested for the nodes at the virtual wall, communication between the virtual wall and the POWDER testbed's nodes is possible using public IPv4 addresses.
5. Communication to the POWDER nodes (IPv4 address) from w-ilab1 and w-ilab2.t testbeds is possible through a NAT (Network Address Translation) function enabled at the router (routerG).
6. It is not currently possible to communicate with the w-ilab1.t and w-ilab2.t testbeds from the POWDER testbed, as there is no public IPv4 assigned at the nodes of w-ilab1.t, and w-ilab2.t and POWDER testbed nodes do not have public IPv6 addresses. This may be possible in the future by redirecting the traffic through the virtual wall.

3. Number of Hops Travelled



Here, hop in negative means that it is not calculated as there is no direct way to calculate it within the testbed.

Figure 44: Number of Hops Travelled

Figure 44 shows the hops between different nodes when using a traceroute application. We used either public IPv4 addresses, public IPv6 addresses, or private IPv4 addresses in order to figure out the number of hops using the information presented in the previous subsection. The Y-axis in Figure 44 shows the number of hops travelled, while the X-axis shows the source testbed nodes. The figure shows that nodes in a testbed are directly connected to each other since they are all within zero hops of each other. Further, w-ilab1.t, w-ilab2.t, and the virtual wall testbeds are one hop away from each other. In addition, the CityLab is 5 hops away from w-ilab1, w-ilab2.t, and the virtual wall testbeds. Moreover, the POWDER nodes are 21 hops away from w-ilab1, w-ilab2.t, and virtual wall nodes, and 24 hops away from the CityLab nodes. We did not calculate the number of hops from the POWDER nodes as a source node to the w-ilab1.t and w-ilab2.t nodes as a destination node, as there is no direct way to calculate this (as mentioned in the previous subsection).

4. Round-Trip Time

Figure 45 shows the round-trip time of ping between two nodes. The X-axis represents the testbed of the source node, while the Y-axis shows the average round-trip time in ms. It also displays the minimum and maximum round-trip times through an error bar. The shortest overall round-trip time (0.3ms) was recorded between nodes of the same testbed. The round-trip times between w-ilab1.t and w-ilab2.t are less than 1 ms. The round-trip time between CityLab and any of the other testbeds in Belgium is less than 5 ms. The

longest round trip-time is between EU and US testbeds, which is around 140 ms.

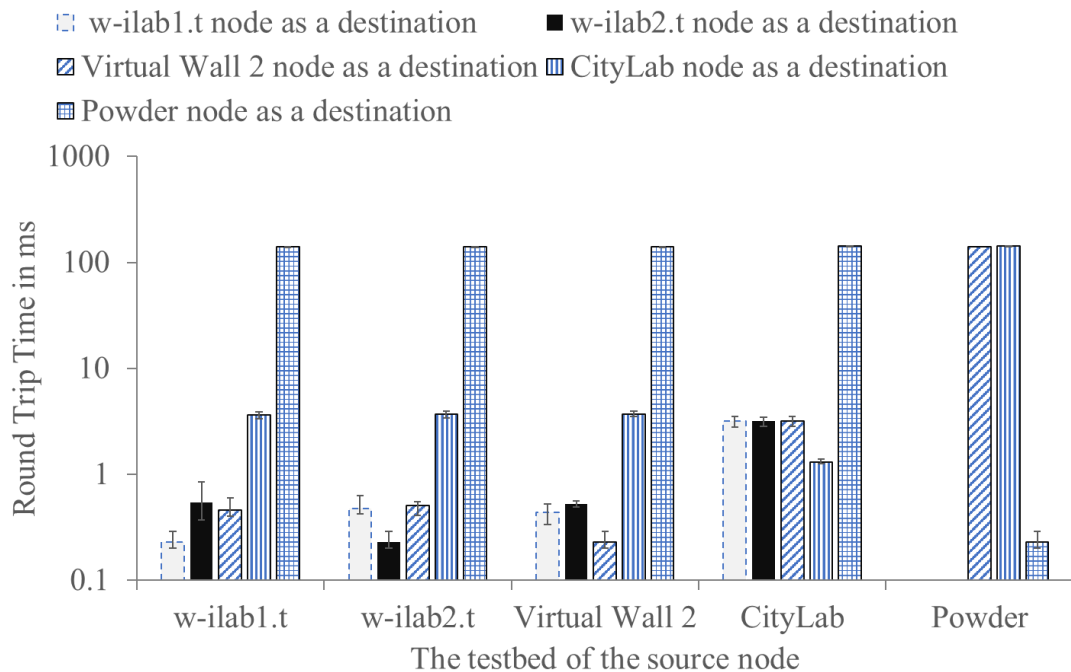
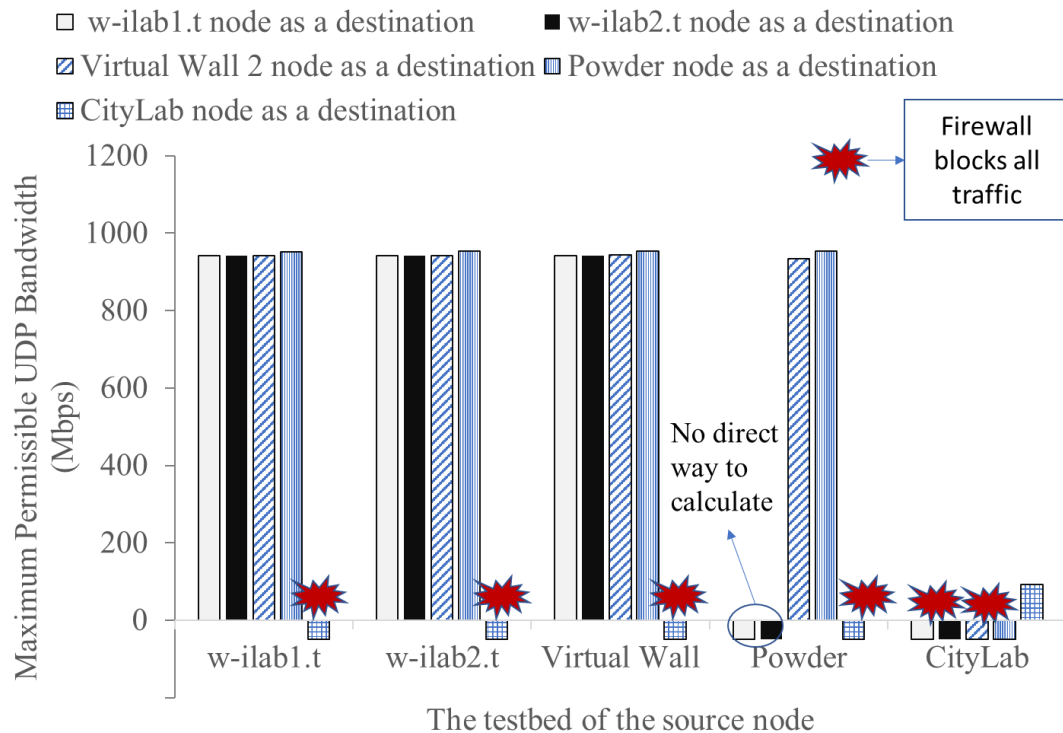


Figure 45: Round-Trip Time in ms

5. UDP Stress Test

Figure 46 illustrates the maximum permissible UDP bandwidth between testbed nodes. Iperf is used to determine the maximum permissible bandwidth. This is the limit after which traffic will be dropped due to insufficient bandwidth. As UDP does not wait for acknowledgement from a sender, Figure 46 shows that distance travelled has no effect on the maximum permissible bandwidth. For w-ilab1.t, w-ilab2.t, Virtual Wall and POWDER, the maximum bandwidth is approximately 940 Mbps. The maximum bandwidth between CityLab nodes is approximately 92 Mbps. Furthermore, since CityLab has a firewall that blocks external UDP traffic, we cannot calculate the maximum bandwidth between CityLab and any other testbed. Figure 46 shows that ping between CityLab and any other testbeds works fine. This is because the Citylab firewall just passes only ICMP traffic. Additionally, we did not calculate the maximum bandwidth available from POWDER nodes to w-ilab1.t and w-ilab2.t nodes as there are no public IP addresses available in these testbeds and there is no direct way to calculate this without for instance using tunnelling through the virtual wall.

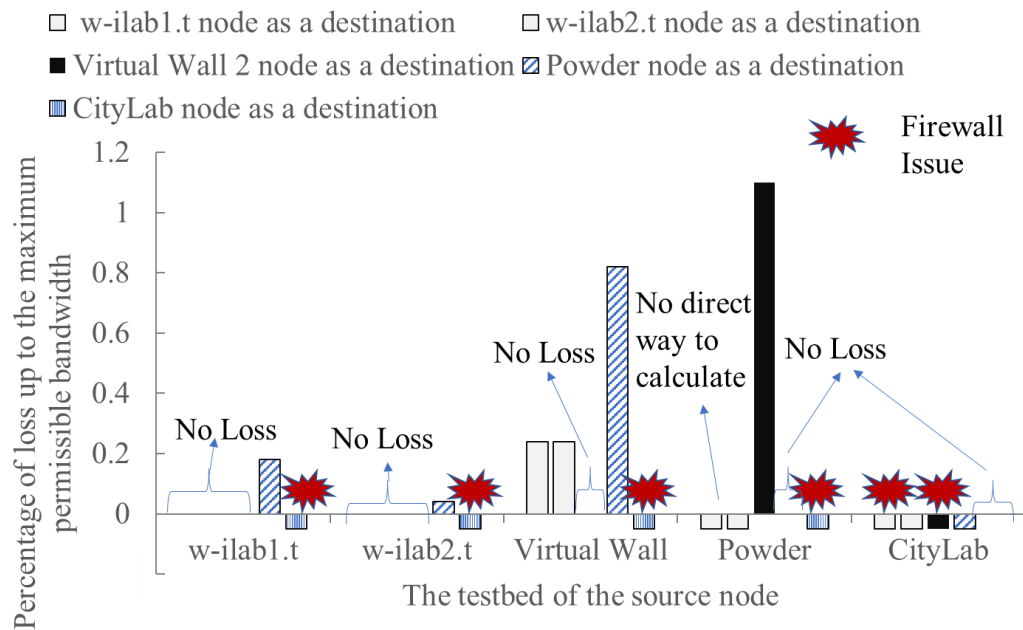


Here, bandwidth in negative means that it is not calculated as there is a firewall which blocks the traffic or there is no direct way to calculate.

Figure 46: Bottleneck Bandwidth in Mbps

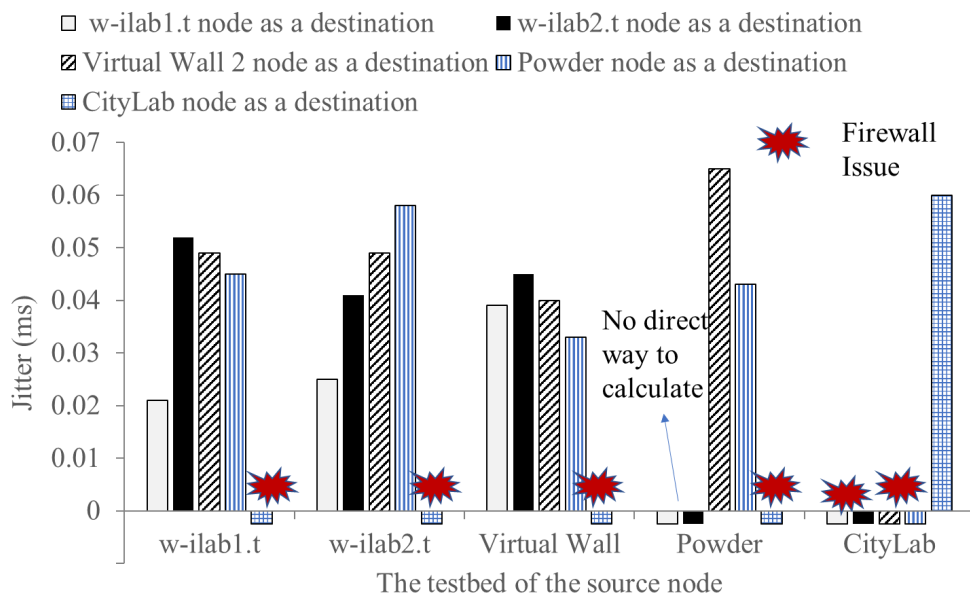
Figure 47 shows the percentage of loss when the traffic is sent between nodes up to the amount of the bottleneck bandwidth achieved in Figure 46. When traffic is sent within a testbed, there is no loss. Additionally, we did not observe any packet loss when traffic is sent from one w-ilabs.t node to another w-ilabs.t node or a virtual wall node. There is however a loss of traffic when the virtual wall or any other testbed is used. When traffic is sent from the virtual wall to any of the w-ilab.t testbeds, the packet loss is less than 0.4%. It appears to be an issue with RouterG that is causing the packet loss. Since users cannot access this router, we cannot determine the cause of this loss. Packet loss increases to approximately 1% when traffic is sent between the virtual wall and the POWDER testbeds. Packet loss could also have occurred somewhere in the Belnet or Internet2 or at the router (RouterG) placed at the virtual wall. This minimal loss makes w-ilab1.t, w-ilab2.t, virtual wall, and POWDER testbeds suitable for inter-testbed experiments.

The average value of the jitter is shown in Figure 48. It shows the jitter is under 0.1 ms, which is minimal for an inter-testbed experiment where nodes are located throughout the world (in the EU and the US).



Here, loss in negative means that it is not calculated as there is a firewall which blocks the traffic or there is no direct way to calculate.

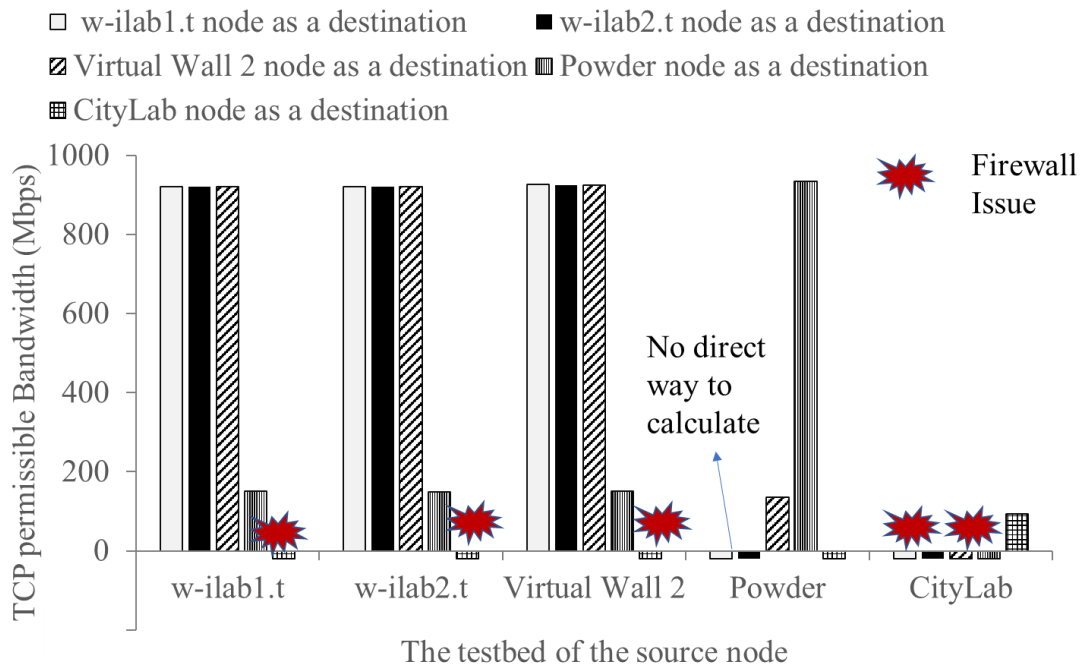
Figure 47: Percentage of Loss when traffic is sent up to the amount of the bottleneck bandwidth



Here, jitter in negative means that it is not calculated as there is a firewall which blocks the traffic or there is no direct way to calculate.

Figure 48: Average value of Jitter in ms

6. TCP Stress Test



Here, bandwidth in negative means that it is not calculated as there is a firewall which blocks the traffic or there is no direct way to calculate.

Figure 49: Bottleneck in TCP bandwidth - Mbps

Figure 49 shows the maximum TCP bandwidth in Mbps as traffic is sent between different nodes. The figure shows the effect of distance on the maximum TCP bandwidth, since TCP waits for an acknowledgement from the previous segment before sending the next segment, and also retransmits any unacknowledged segments. The maximum TCP bandwidth is lower than the maximum UDP bandwidth, as UDP does not wait for an acknowledgement. For traffic sent between w-ilabs.t or virtual wall nodes, the maximum TCP bandwidth is approximately 921 Mbps. It is worth noting, however, that the maximum bandwidth between POWDER and any of the other testbeds is approximately 150 Mbps, which is considerably lower than the UDP bandwidth (941 Mbps). That is because POWDER is located far away from the rest of the testbeds and acknowledgements from the receiver have to travel a long path. Further, in the POWDER testbed, the maximum TCP bandwidth between nodes is approximately 934 Mbps.

SDN Inter-Testbed Results



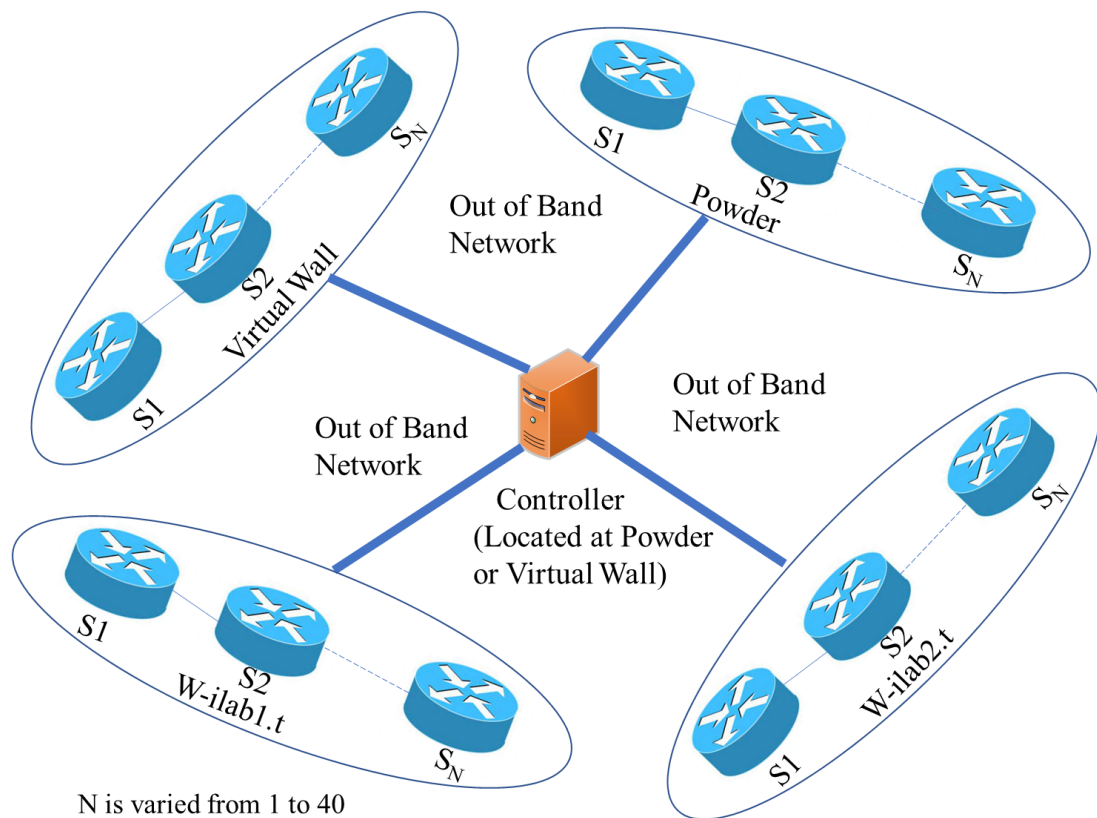


Figure 50: OpenFlow Networks created between different testbeds

We use only w-ilab1.t, w-ilab2.t, virtual wall, and POWDER testbeds for our SDN experiment, since only these testbeds are found to be appropriate for inter-testbed experiments. For this experiment, OpenFlow is used as an SDN protocol, and a single controller controls all the switches present in each testbed (see Figure 51). Open vSwitch is used as the OpenFlow switch software and the POX controller is used as the controller software. The controller is placed at the POWDER node or the virtual wall node and communicates with the OpenFlow networks created on each testbed through an out-of-band network shown in Figure 50. A separate node of the testbed is used to deploy the controller and OpenFlow network topology. Figure 50 also shows the deployed OpenFlow switch topology. Mininet is used to create this OpenFlow network topology in each testbed. In order to compute the performance of this SDN network located at several testbeds, the number of OpenFlow virtual switches is increased.



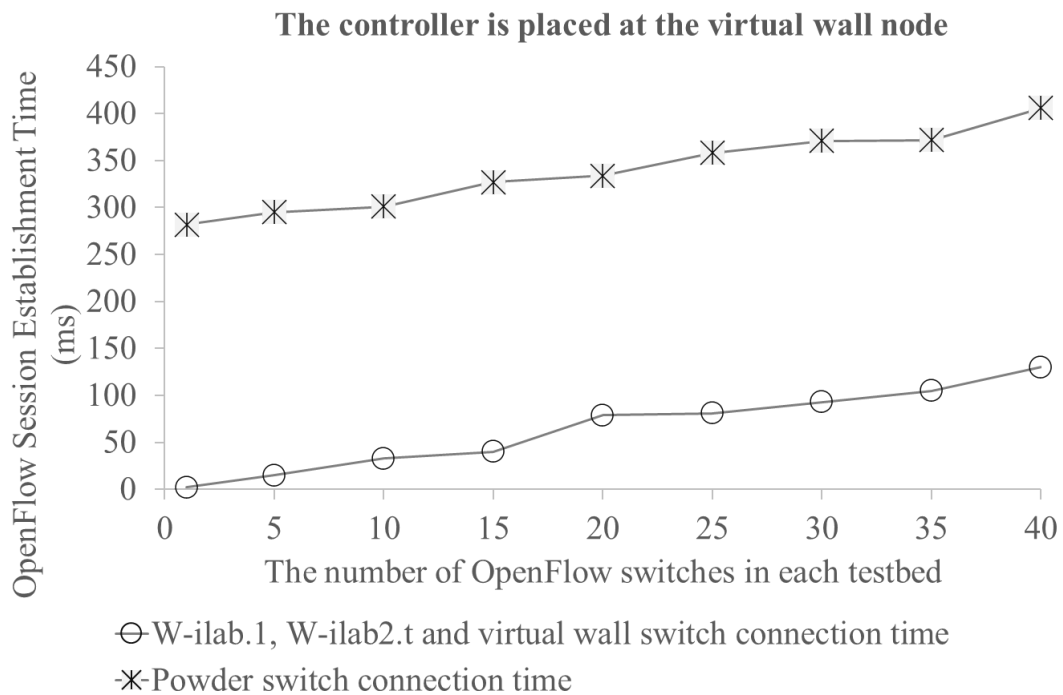


Figure 51: OpenFlow Session Establishment time when the controller is placed at a Virtual Wall node

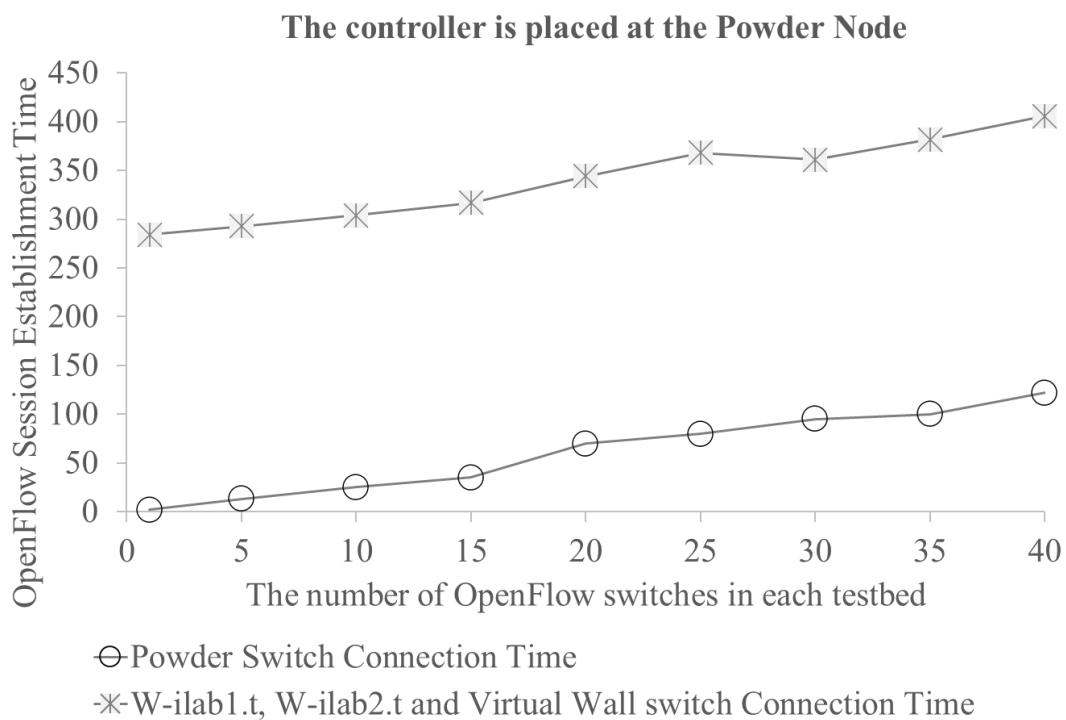


Figure 52: OpenFlow Session Establishment time when the controller is placed at the Powder node

Figures 51 and 52 show OpenFlow session establishment times when the controllers are placed at the virtual wall and at the POWDER testbed, respectively. When the number of switches increases, there is an approximately linear increase in the OpenFlow session establishment time. We observed no significant difference in the results when an OpenFlow network is created at w-ilab1.t, w-ilab2.t, or virtual wall testbeds as they are

located in the same building in Ghent. Thus, all the results from these testbeds are combined, and the average is presented in Figures 51 and 52. The results show that when the controller is located far away, the OpenFlow session establishment time is significantly long.

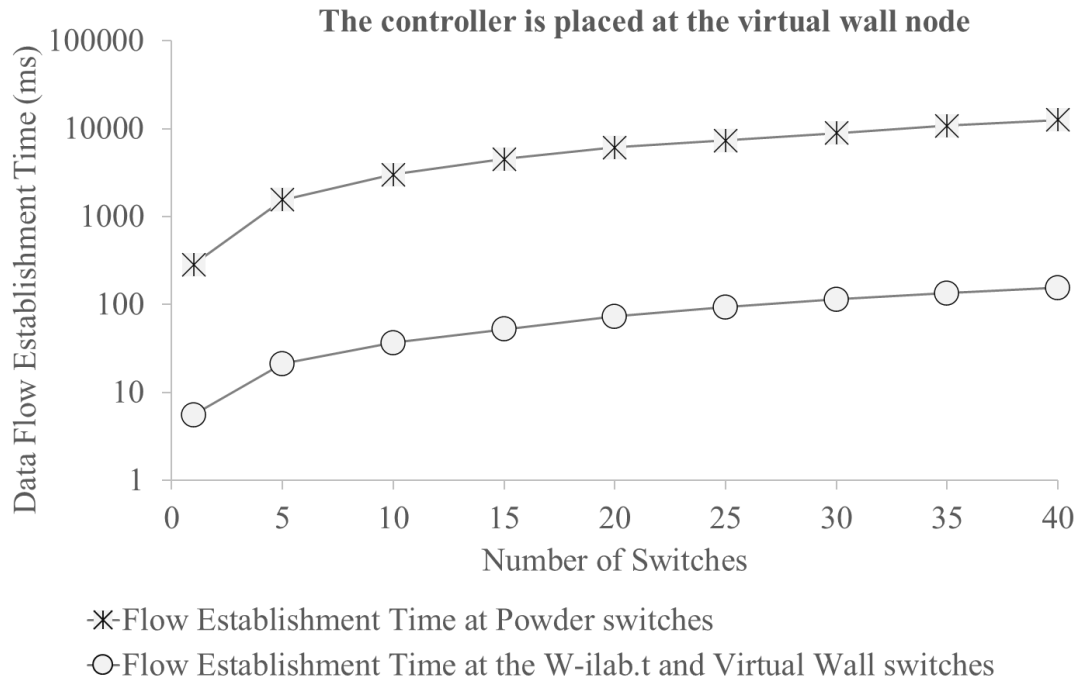


Figure 53: Data Flow Entry Establishment time of all switches when the controller is placed at the virtual wall node (i.e., at an EU testbed)

Figure 53 shows the data flow establishment time, which is defined as the instant when the controller is able to establish forwarding entries in all the switches along the path to the destination. Figure 53 shows the time when the controller is placed at the virtual wall node. It shows that when the number of switches along a path to the destination increases, the data flow establishment time increases approximately linearly. This is because the controller has to establish more forwarding entries, as the number of switches along the path increases. As flow establishment time is approximately the same for OpenFlow switches in w-ilab1.t, w-ilab2.t, and virtual wall, the results are combined, and the average is shown in Figure 53. However, as the flow establishment of switches located at the POWDER testbed is significantly large compared to the switches at other testbeds, this time is shown as a separate line in Figure 53. The flow establishment time for POWDER switches is larger compared to switches at other testbeds, as switches at POWDER nodes are located far away from the controller located at the virtual wall.

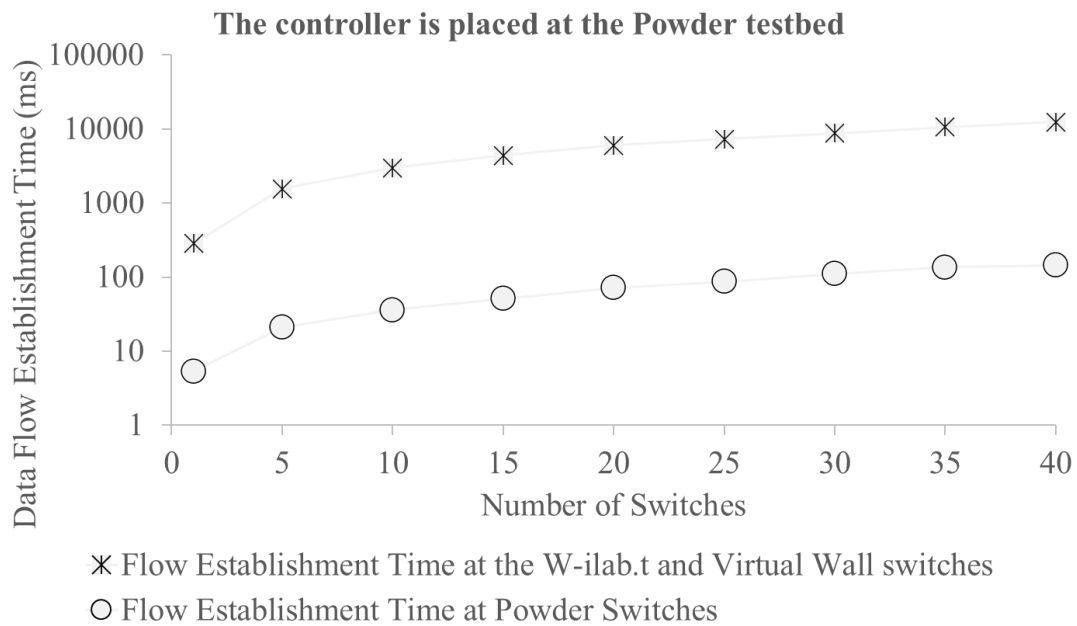


Figure 54: Data Flow Entry Establishment time of all switches when the controller is placed at the Powder node (i.e., at the US testbed)

Figure 54 shows the data flow establishment time when the controller is located at a POWDER node. It shows that the data flow establishment time is lower for the switches in the POWDER testbed only. This is because the controller and switches are located in the same testbed. However, it is higher for switches located at other testbeds (i.e., at EU). This is because these testbeds are located far away from the controller node located at the POWDER testbed (i.e., in the US).

The following are the learnings from our SDN experiment:

1. OpenFlow sessions and Flow Entry Establishment times are significantly longer when the controller is in the EU and the OpenFlow switches are in the US, and vice versa.
2. In case the controller and OpenFlow switches are located at the same location (e.g., in the EU or US), these times are significantly shorter.
3. Experimenting with EU and US testbeds can be useful when creating edge computing-type scenarios in which some functionality can be moved closer to users to allow for faster decisions and other functionality can be kept at a faraway testbed. The next experiment, which relates to machine learning decisions, illustrates this.

Resource Intensive Machine Learning Use Case

An edge-computing use case is implemented on the testbeds located in the EU and the US. This use case illustrates that when we include all the functionality on an edge node with limited resources, the edge node's functions will be delayed. By moving resources-intensive functions to the cloud, those decisions can be made more quickly. During this experiment, a controller is placed at the virtual wall to control the OpenFlow switches in w-ilab1.t and w-ilab2.t. In addition to making forwarding decisions, the controller has to perform resource-intensive machine learning

applications that process a lot of data, train and make decisions about machine learning. We consider two use cases. In the first case, the controller located at the virtual wall performs all functions. In the second case, the controller offloads the resource-intensive tasks to the controller located at the POWDER node (in the US).

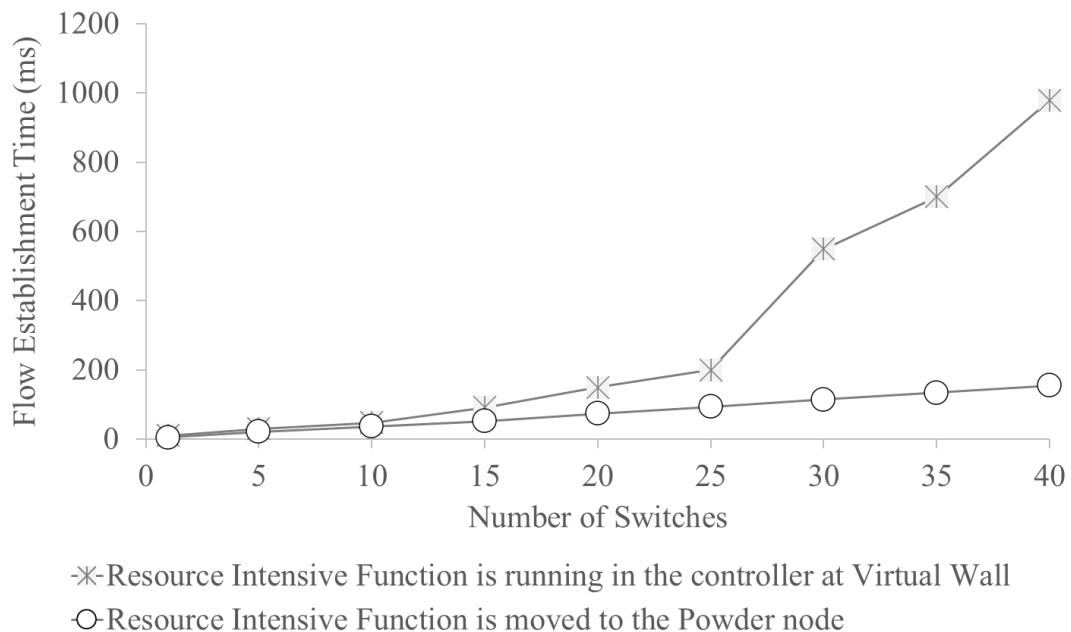


Figure 55: Edge Computing Usecase with the testbeds in the EU and US

In this experiment, all controller tasks are performed by a single CPU of a virtual wall node. Figure 55 shows the results of both the cases considered. In the first case, the resource-intensive functions are running concurrently with the controller's normal forwarding function, causing tasks such as establishing Flow Entries to the switches to be delayed. Accordingly, Figure 55 shows a high Flow Establishment time when the controller itself is running resource-intensive functions. Consequently, the resource-intensive machine learning functions are moved to the POWDER node in the US. Therefore, the controller is able to perform normal actions (such as establishing forwarding entries) quickly.

4.1.6 IoT Application

In this subsection, we first explore an IoT secure application which is able to detect denial of service, authentication and probe attacks. We used an edge-cloud model proposed in our inter-testbed activity (also reported at deliverable 2) in this work. We first perform the simulation testing of our application using NSL-KDD dataset and then test it on the GPULAB Fed4Fire testbed. The simulation work of this IoT application has been published in the Future Internet journal and the GPULAB FedFire work has been accepted for publication as a demo paper in the IEEE LANMAN 2022.

We then implement an e-healthcare application on the US testbed and connect with AWS (Amazon Webservice Service) servers. Both of these applications are described below:

Our secure IoT application

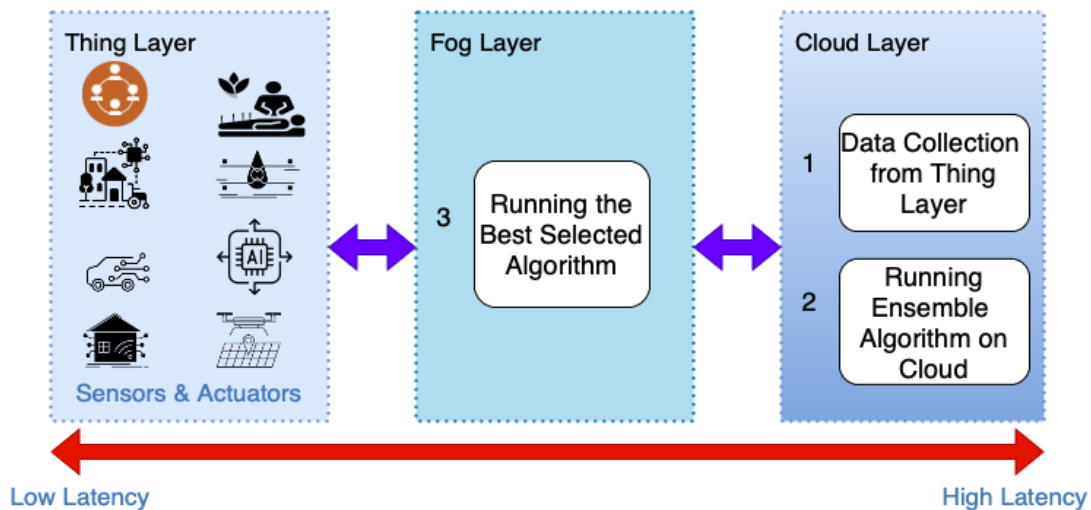


Figure 56: Our proposed IoT Secure Application for detecting Attacks

Our objective to implement the application is to use ensemble machine learning techniques for detecting attacks in an IoT system. This is because deep neural networks require substantial resources, such as memory. The goal is to come up with the best ensemble method and to apply it for real-time attack detection. Figure 56 outlines the proposed approach with three layers: thing, fog and cloud. It involves the following three steps (also shown in Figure 56): (1) data collection at the cloud layer, (2) running the ensemble algorithm on the cloud and selecting the best model, and (3) running the best-selected algorithm in the cloud. The description of the above tasks is given below.

a. Data Collection at Thing Layer:

This step involves collecting data from the thing layer and passing it to the cloud layer. To accomplish this, data from the thing layer can first be transported to the fog layer. The fog layer can then transport it to the cloud layer. While transporting the data to the cloud layer, the fog layer can also filter data to decide which data to be transported to the cloud. IoT attacks can be predicted using the following attributes: (1) login details, (2) the fields of network data packets, such as fragment details, protocol type, source and destination address, (3) service type, (4) flags, and (5) duration.

b. Selecting a best model on the cloud

The objective of this step is to combine various basic machine learning classifiers (such as naïve Bayes, KNN, and decision trees) with ensemble techniques (such as stacking, bagging, and voting) to obtain optimal results (accuracy, precision, execution time). As this is a time-consuming step, we recommend running it in the cloud. In addition, we simply apply the basic machine learning classifiers, as they require a short execution time.

Algorithm 1 describes the above-proposed approach in detail. The input parameters of the algorithms are: (1) base classifiers (i.e., $B = B_1, B_2, B_3, \dots, B_n$), (2) ensemble methods (i.e., $E = E_1, E_2, E_3, \dots, E_m$), and (3) training dataset (D). At the first two lines of the algorithm, the output and the result (i.e., variable OUTPUT and Result in Algorithm 1) are initialized to NULL. The third line initializes the execution time to the maximum value.

In the fourth line, we store all the combinations of the base classifiers (i.e., using the function findAllCombinations) in variable C. The proposed approach aims to determine the best combination and the best ensemble method. Therefore, in line 5, we iterate each of the combinations, and then, again, in line 7, each base classifier in the corresponding combination is iterated. Each base classifier is applied to the training dataset (D) with the outcome being stored in o (line 8).

Line 10 involves an iteration of the ensemble methods and the application of each ensemble method to the outcome (o) at step 11. At line 12, the ensemble result is calculated in terms of accuracy, precision, recall, etc. Further, at line 13, the execution time of the combination of base classifiers and an ensemble method is calculated. The new result (r) and execution time (time) is then compared to the previous best result (Result) and time (ExecutionTime). If this is the best result so far, the corresponding combination and ensemble method are stored in the output (OUTPUT); see line 14. Further, the result is stored in line 15. In the end, the best output is returned at line 21



Algorithm 1: Find a best model

```

procedure FINDABESTMODEL(B, E, D)
    // B ← B1, B2, B3, ... Bn. Here, B1, B2, B3, ... Bn are Base
    Classifiers.
    // E ← E1, E2, E3, ... Em. Here, E1, E2, E3, ... Em are Ensemble
    methods.
    // D is the training dataset.
    1 OUTPUT ← NULL;
    2 Result ← NULL;
    3 ExecutionTime ← MAX;
    4 C ← findAllCombinations(B);
    // Find all the combinations of the Base Classifiers (B).
    5 foreach c ∈ C do
        // Iterate each combination.
        6 o = 0; // Initialize an outcome.
        7 foreach Ea ∈ c do
            // Iterate each Base Classifier in c.
            8 o ← o, ApplyBaseClassifier(Ba, D);
            // Apply Ea over dataset D and store the outcome of each
            Ea in the form of the ROC curve or any performance measure
            in o.
        9 end
        10 foreach Ea ∈ E do
            // Iterate each Ensemble method.
            11 e ← applyEnsembleMethod(Ea, o);
            // Apply Ensemble Method Ea on o.
            12 r ← findResult(e);
            // Find the result in form of ROC or any other Performance
            measures
            13 time ← findExecutionTime(c, Ea);
            // The execution time of the combination of base
            classifiers (c) and an ensemble method (Ea) is calculated
            14 if isResultBetter(r, time, Result, ExecutionTime) then
                // If r and time is better than Result and
                ExecutionTime.
                15 OUTPUT ← c, Ea;
                // Store the base classifiers and Ensemble method over
                OUTPUT.
                16 Result ← r;
                17 ExecutionTime ← time;
            18 end
        19 end
    20 end
    21 Return : OUTPUT;
    // Return the best model with base classifiers and an ensemble
    method
    
```

Algorithm 1: Detecting Attacks using an IoT application

c. Running the best model in the fog layer

This step involves executing the model selected in the previous step over the fog layer with the real-time data collected from the thing layer. The model consists of a combination of base classifiers and an ensemble method.

Simulation Results and Analysis

We performed two experiments of our approach: (1) with a server containing a CPU Core E7400 processor and 3.00 GB of RAM and a 32-bit operating system with 2.80

GHZ and (2) with the GPULab testbed with varies GPUs, CPUs and CPU memory. Both the experiments are described below:

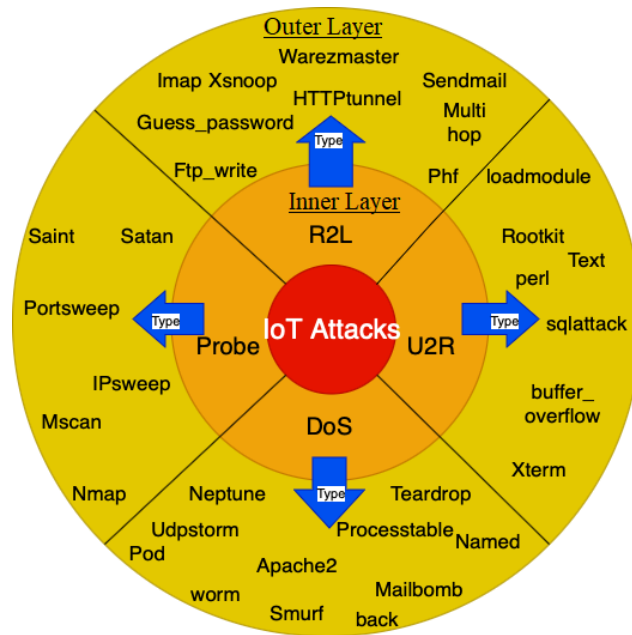


Figure 57: Dataset Description

The NSL-KDD dataset [7] is used to simulate this work. It contains 41 features to describe each specific entity in an IoT network. Details on network intrusions with these 41 features can be segmented into computational information (service, flag, land, etc.), content-based information (login information, root shell information, etc.), duration-based (such as the duration from host to destination transfer, error rates), and host-based information (host and destination ports and counts information).

In Figure 57, the NSL-KDD dataset is represented by two layers: (1) the inner layer represents different types of IoT attacks in the dataset, such as Probe, DoS, U2R, and R2L; (2) the outer layer represents examples of attacks within each category. Attacks such as Saint, Satan, Nmap, and portsweep, which can be found in Figure 601, come under the Probe IoT attack category. In these attacks, the attacker scans a network device to determine potential weaknesses in its design, which are subsequently exploited in order to gain access to confidential information.

Likewise, attacks such as Neptune, Teardrop, Worm, and Smurf fall into the category of DoS attacks. These attacks cause a denial of service when an attacker consumes resources unnecessarily, making the service unavailable for legitimate users. Moreover, Sendmail, Multihop, and phf belong to R2L (remote-to-user) attacks, while Perl, text, and sqlattack belong to U2R (user-to-root) attacks. In Figure 601, variables are underlined according to their segment. Most variables in this dataset are nominal. There are three basic protocol types, TCP (transmission control protocol), UDP (user datagram protocol), and FTP (file transfer protocol), that exist in the dataset.

For simulation, a significant subset of the NSL-KDD dataset is used in the cloud layer for training and validation, while the rest of the unlabeled data is considered for

real-time processing in the fog layer for testing. Moreover, K-cross validation is used with an 80:20 ratio at the cloud layer.

Table 3: Base classifier combinations: decision tree (DT), random forest (RF), K-nearest neighbor (KNN), logistic regression (LR), naïve Bayes (NB).

Model	Base Classifier Combinations		
1	DT	RF	KNN
2	RF	KNN	LR
3	KNN	LR	NB
4	LR	NB	DT
5	NB	DT	RF
6	DT	KNN	LR
7	RF	LR	NB
8	KNN	NB	DT
9	LR	DT	RF
10	NB	RF	KNN

Simulating the proposed approach included the use of five machine learning classifiers and two ensemble methods. The classifiers used are: (1) decision tree (DT), (2) random forest (RF), (3) K-nearest neighbors (KNN), (4) logistic regression (LR), and (5) naïve Bayes (NB), while ensemble techniques are voting and stacking. Table 3 shows the detail of each combination of base classifiers in the base layer. A total of 10 different model combinations are tested. The models are listed in Table 3. This is because we selected five base classifiers, and we created combinations of two. Therefore, we end up with 10 models (5C2).

We evaluate the results of the proposed approach for the cloud and fog layers using three factors: (1) execution time, (2) performance measures, and (3) error associated with the final model. On the cloud layer, a larger amount of data (training) is used to build models and conduct experiments. Testing data is considered new data and is tested on the fog layer. In the cloud layer, the best model is selected, and in the fog layer, it is evaluated using real-time data. Our first objective is to summarize the results, including the cloud layer, and the method by which model 8 (distributed in Table 3), with an ensemble method, was selected to be applied to the fog layer. Following that, we show the results obtained from the real-time data in the fog layer.

a. Cloud Layer Results

Figure 58 displays the execution time for voting and stacking ensemble methods over all the models described in Table 3. The X-axis in Figure 58 refers to the duration in seconds to execute a model, while the Y-axis refers to the model number. Compared to the voting ensemble method, stacking takes a much higher execution time. According to our results, model number 8, with the voting technique, shows minimal execution time (9.96 s), with KNN, NB, and DT used as base classifiers.

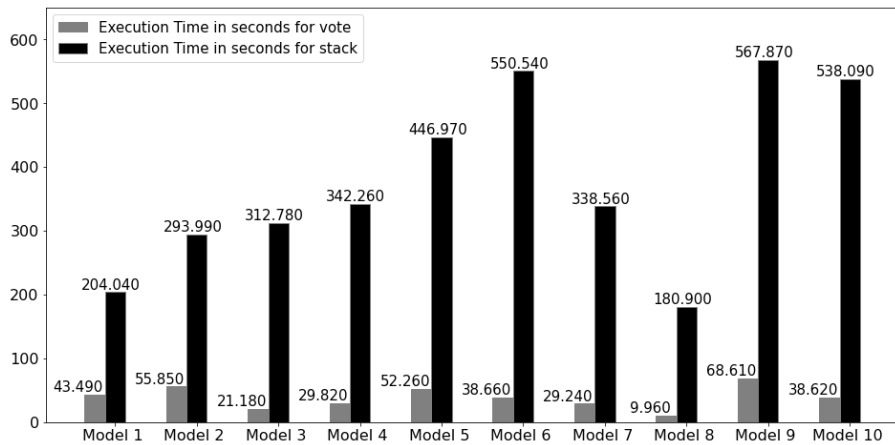


Figure 58: Execution times of all models.

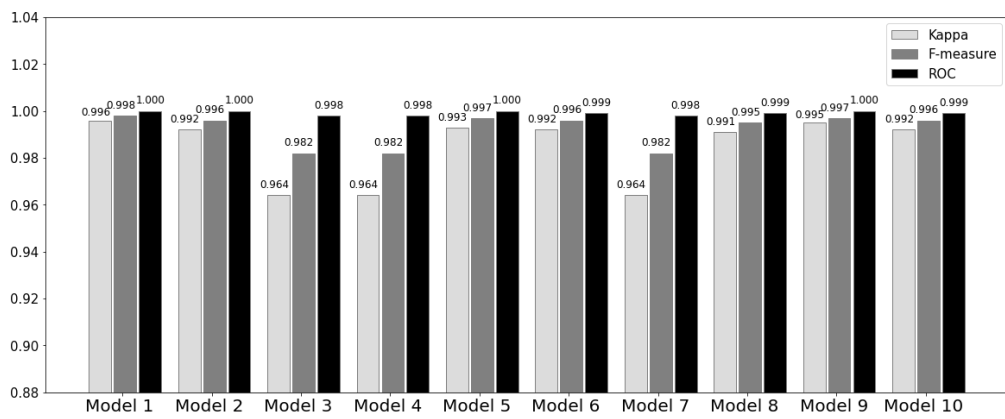


Figure 59: Performance of all the models in terms of Kappa, F-measure and ROC value

Figure 59 shows overall performance as measured by kappa, F-measure, and the ROC area. It shows that all the models have values greater than 0.99, with model 8 providing the kappa value 0.991, the F-measure value 0.995, and the ROC area 0.999.

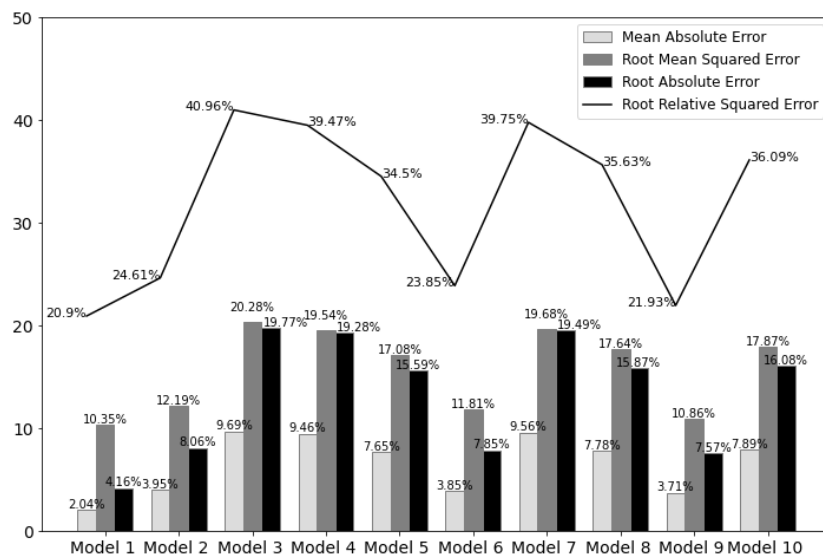


Figure 60: Error Associated with each model



Figure 60 shows the errors with voting as an ensemble method in terms of mean absolute error, root mean square error, relative absolute error, and root-relative squared error. Model 1, with voting, exhibits significantly fewer errors than any other model. In this model, DT, RF, and K-NN are used as base classifiers, and voting is used as an ensemble technique. In spite of this, we selected model 8 with voting to run in the fog layer, as it performed well in terms of execution times and other performance parameters, as shown in Figures 58-60.

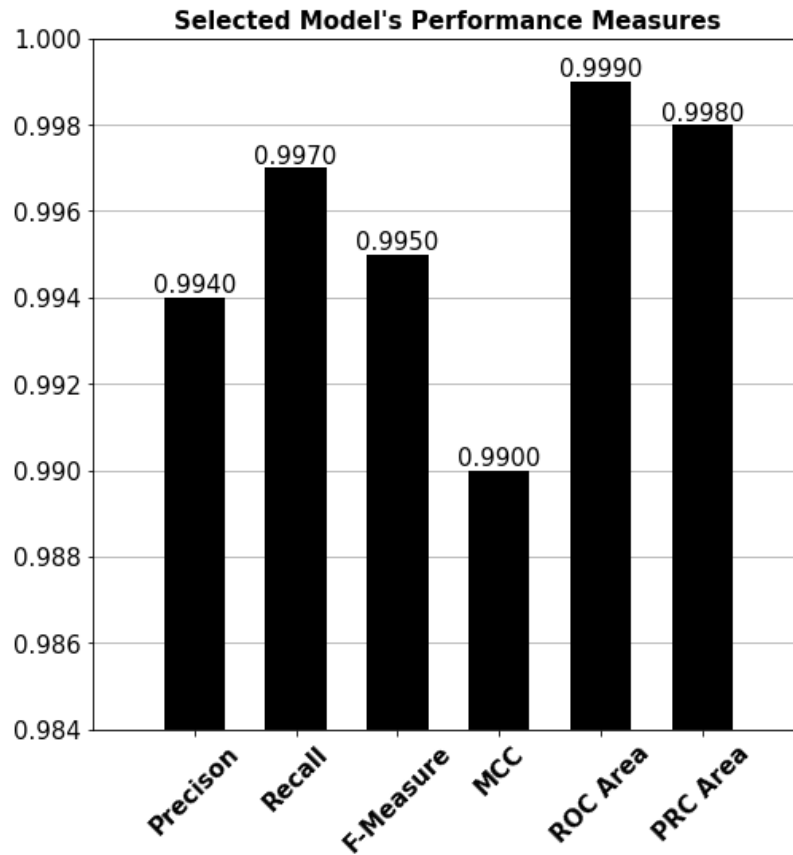


Figure 61: Performance of model 8 (selected model)

To verify further, we calculate the performance of model 8 in terms of precision, F-measure, MCC, and PRC area (Figure 61), in addition to all other metrics. Through the Y-axis, the result is accurate to three decimal places. The most significant performance metric is MCC, which indicates how random or real the prediction is. It ranges from -1 to 1. Model 8's values in the experiment are typically closer to 99.99 percent. In general, model 8 with voting is highly optimized to run on the fog layer, according to the requirements of real-time execution and excellent performance.

We found that model number 8, using K-nearest neighbor, naïve Bayes, and decision trees as the base classifiers outperform all other models with respect to execution time and performance metrics (such as kappa, F-measure, ROC, and MCC). Since time is an important factor in the selection of any model, the voting ensemble technique determines that model 8 takes the least time: 1.15 s. Additionally, kappa, F-measure, ROC, and MCC have maximum values of 6.39, 98.20, 99.60, and 96.40, respectively. There is also a mean absolute error of 7.78 percent, a root mean square error of 17.64 percent, a relative absolute error of 15.87 percent, and a root-relative

squared error of 35.63 percent. Further, the root-relative squared error of model 8 is 27.94 percent, and the minimum impact is 0.6 percent. In fact, model 8 is the most time-efficient and resource-intensive model, which is why it has the greatest impact.

b. Fog Layer Results

With the new data now being included, we measure the performance of model 8, with this model having KNN, NB, and DT as the base classifiers, as well as voting as an ensemble model.

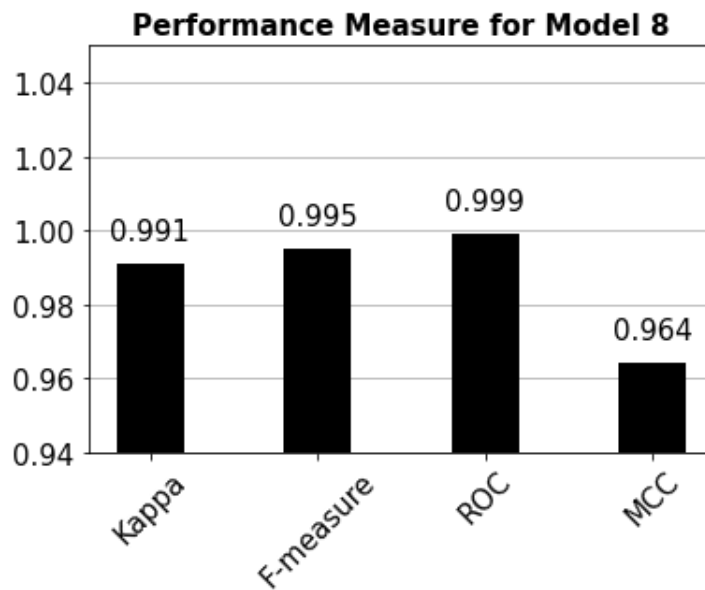


Figure 62: Performance of the selected model in the fog layer

Performance measures such as kappa, F-measure, and ROC indicates how well the model performs in the fog layer. Figure 62 illustrates that all performance indicators in the selected model are almost equal and at the top. The values are 96.39, 98.20, 99.60, and 96.40 for kappa, F-measure, ROC, and MCC, respectively.

Figure 63 represents the mean absolute error (MAE), root mean square error (RMSE), relative absolute error (RAE), and root-relative squared error (RRSE). Our experiment yielded mean absolute error, root mean square error, relative absolute error, and root-relative squared error values of 7.78, 17.64, 15.87, and 35.63 percent, respectively.

Along with the previously discussed performance metric, we also calculated the execution time of the chosen model, as well as all other models (not selected at the cloud layer) using voting as an ensemble method on the fog node. This execution time is shown in Figure 64.

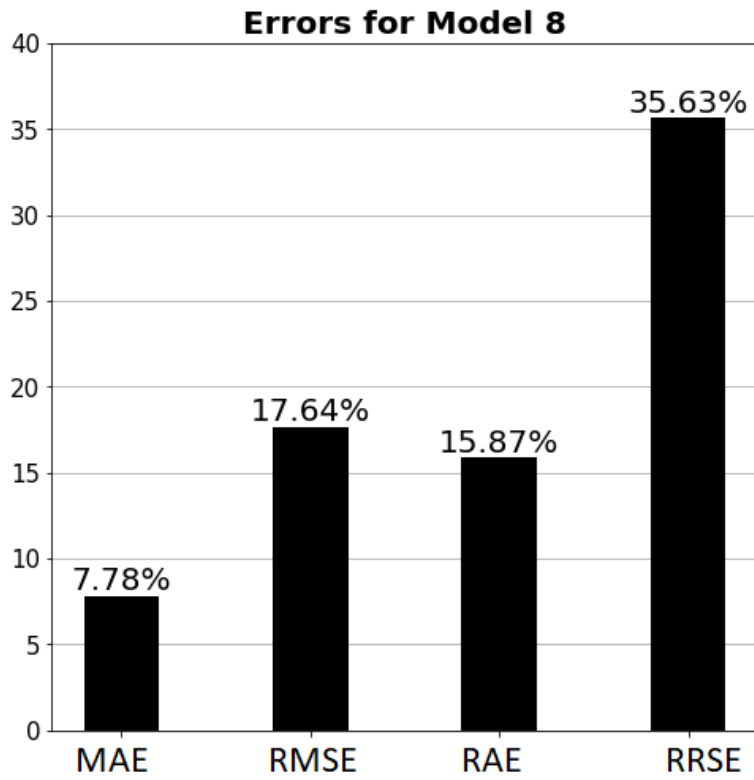


Figure 63: Error associated with the selected model in the Fog layer

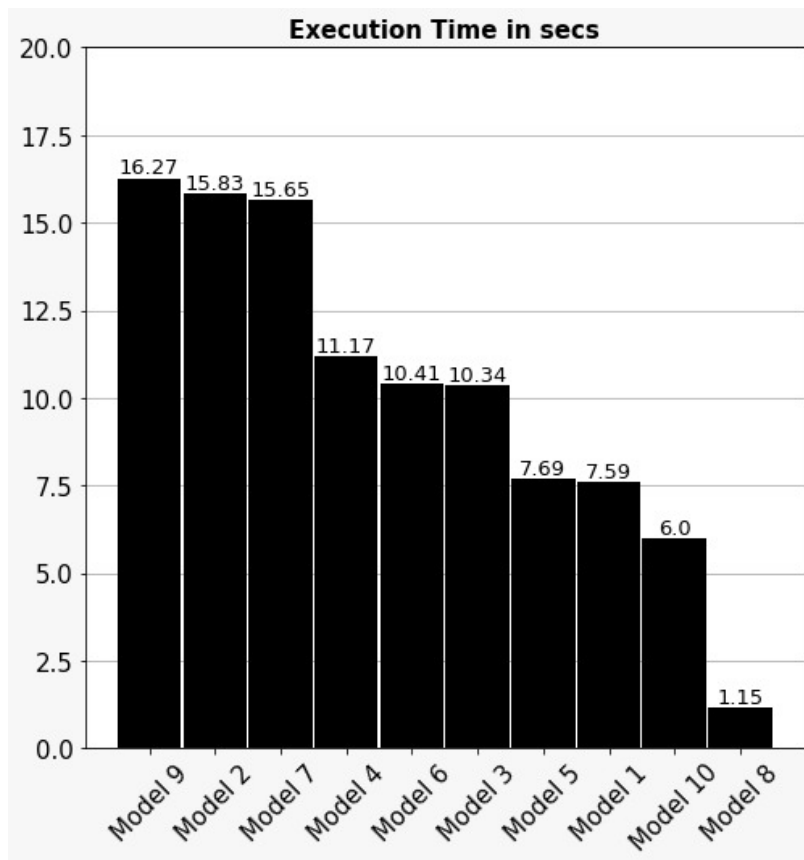


Figure 64: Execution Time of all models in the fog layer



This is to determine whether we selected the correct model in terms of execution time. The fog node execution time of model 8 with voting was the fastest of all models.

Additionally, we calculated the CPU consumption within the fog layer. Less than 10% of the CPU is consumed by the fog layer. Therefore, our method does not require additional resources from fog nodes. Moreover, our approach has a low execution time. This shows that our approach is highly cost-effective.

Results on the Fed4Fire GPULAB Testbed

The problem with the above testing on the IoT application is that this approach took around two hours (after collecting the data) with a relatively limited set of basic classifiers and ensemble techniques (i.e., only 10 models). Therefore, we use the Fed4Fire GPULab testbed to run our approach. GPULAB is a freely available Fed4Fire testbed that contains GPU and CPU resources to run resource-intensive algorithms [8].

In this work, we first demonstrate how our approach can be deployed on GPULAB. Using GPULAB to run our above training process with 10 models took around 10 minutes. Therefore, we run a larger number of models on GPULab (i.e., 30 models) and test their performance. The results show the applicability of our approach to detect attacks in IoT.

Table 4: Resources at GPULAB. Exp is Experiment

Cluster ID	#GPU	#CPU	CPU Memory (GB)	#Slaves	Used in our Exp
1	2	16	31.47	1	Yes
3	1	12	31.66	1	No
4	44	128	1072.77	4	Yes
5	8	80	536.24	2	No
6	16	96	1619.95	1	Yes
7	24	176	2158.91	2	No

Resources at GPULAB are divided into Clusters from 1 to 7. Table 4 lists the total resources available in the GPULAB in terms of GPU and CPU cores, CPU memory and slaves. The availability of resources and clusters for an experiment at a time depends on the resources used by other experimenters at that time. Table 6.12 also shows the clusters we used in our experiments (Cluster 1, 4, and 6). At the time of our experiments, Cluster 1 had only 1 CPU and 8 GB of CPU memory, Cluster 4 had 6 GPUs, 12 CPUs and 32 GB of memory, and Cluster 6 had 12 CPUs and 32 GB of memory.

To test our IoT application, we need a system with numerous resources for the cloud layer steps and a system with minimal resources for the fog/edge layer steps. Therefore, we run the cloud layer tasks at Cluster 4 and 6, and the fog/edge layer tasks at Cluster 1 (Figure 65). An experimenter running our approach in Figure 65 performs two tasks: (1) training ensemble models at the cloud layer and (2) making real-time decisions at the fog layer. For training, our approach containing N ensemble models (which contain a combination of basic classifiers and an ensemble technique)

is tested with the NSL-KDD dataset, and the best model (model m) with respect to the execution time and performance is selected. This model is then used to make real-time decisions in the fog/edge layer, where Cluster 1 is used to run the model (see Figure 65). In our experiments, N is 30 and all the other testing procedures are the same as given in the previous subsection.

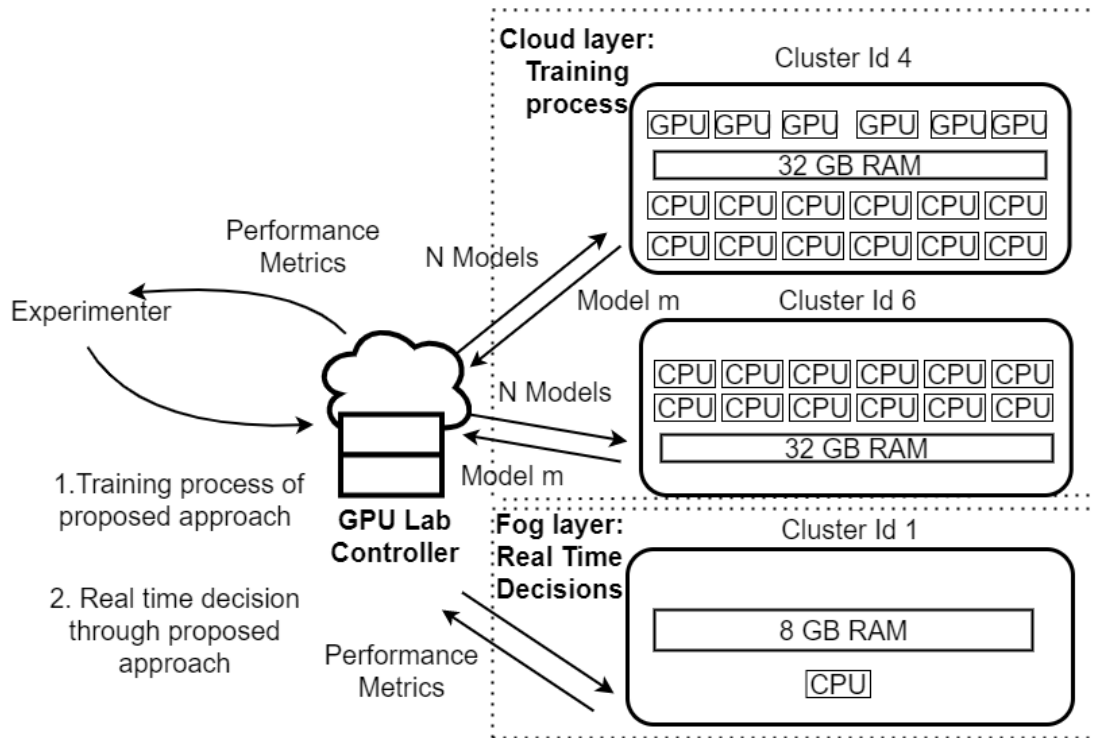


Figure 65: Running the proposed approach on the GPULAB Fed4Fire testbed

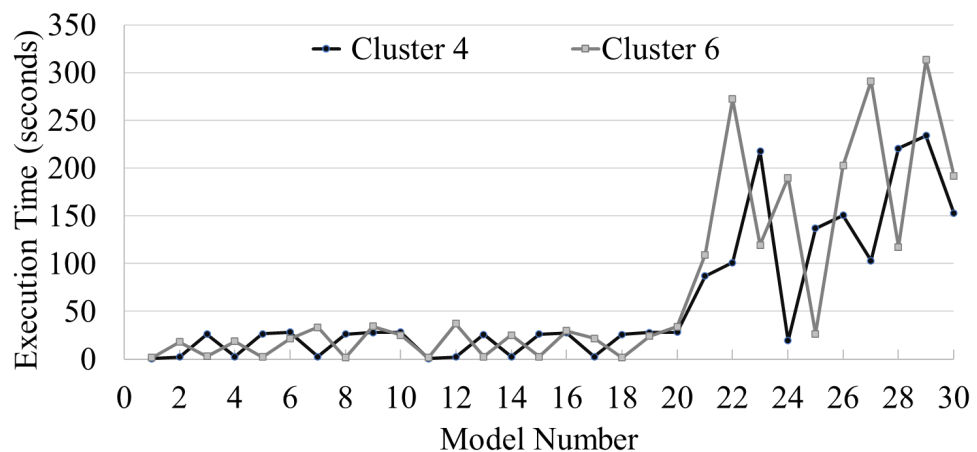


Figure 66: Execution Time in seconds with 30 models.

We selected five basic classifiers for our experiments: (1) Naive Bayes, (2) KNN, (3) Decision Tree, (4) Logistic Regression, and (5) Random Forest, and three ensemble techniques: (1) soft voting, (2) hard voting, (3) stacking. Different models are created by combining the above classifiers and selecting one of the ensemble techniques. In the training process, all these models are tested on Clusters 4 and 6, as shown in Figure 66. Figure 66 shows this execution time. The total training process took less

than 2000 seconds (i.e., <30 minutes) in both the clusters. Furthermore, we calculated accuracy, F-score, Precision, Recall, MCC, ROC, Kappa, and Errors. Our approach selected model 24 to be run on the fog layer based on the best performance metrics and shortest execution time. There are three basic classifiers in this model: Naive Bayes, Decision Tree, Logistic Regression, and the stacking ensemble model.

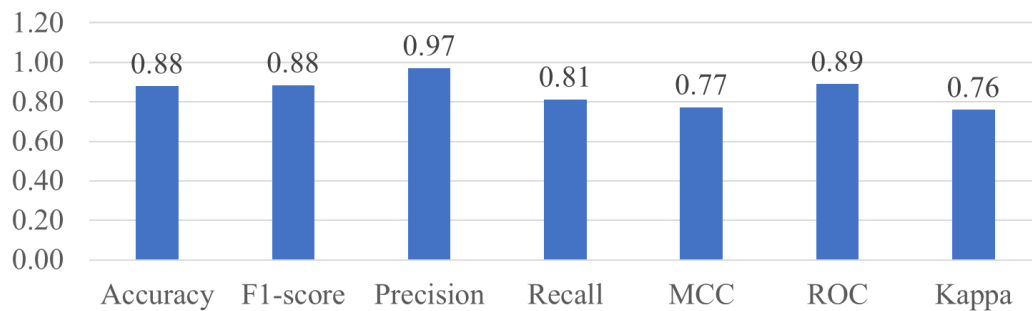


Figure 67: Performance of the selected model on the GPULAB testbed

Model 24 was then tested on Cluster 1 (which contains minimal resources) and the execution time was 351 seconds. The values of performance metrics are given in Figure 67.

This work shows an example of an edge-cloud computing model where high resource-intensive tasks can be run on the cloud and the real-time decisions could be run on the edge/fog layer which is closer to the thing layer. This was also concluded in our inter-testbed experiment, discussed earlier.

Our E-healthcare IoT Application

We present a proof-of-concept demonstrating the feasibility of integrating wireless testbeds with cloud computing platforms in the realm of eHealthcare and remote health monitoring technologies.

The data collection and processing pipeline is outlined in Figure 68. The major goals that we plan to address with our experimental setup are outlined below:

In our experiment, the sensor data transfer can be batch-based or real-time depending on the type of automation that drives the data collection and the chosen data processing technique.

a. Data Collection and Transfer

The data collection process involves the collection of IoT sensor data from the wireless testbeds, accumulation of that data followed by the transfer of data to the cloud computing endpoints over the public wide area network (WAN).

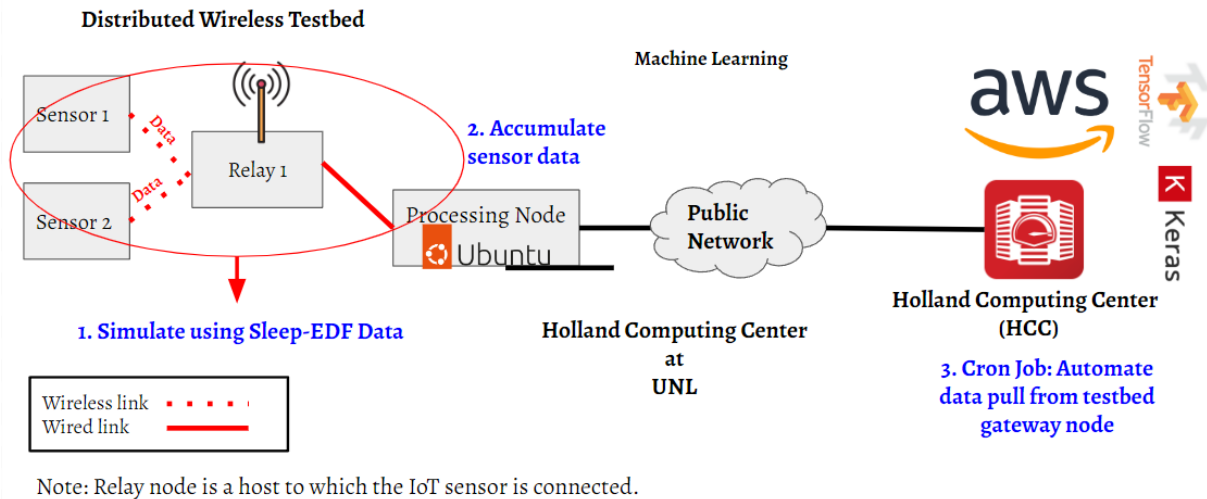


Figure 68: Data collection and processing pipeline

eHealthcare data from the IoT nodes connected to the wireless testbeds is accumulated at a single node that has modest storage, processing, and networking capabilities. The gateway node is often configured with a commercial off-the-shelf (COTS) CPU and hosts a flavor of Linux operating system. This node aggregates and stores the eHealthcare data intermittently before securely transferring it to the compute endpoint over the public WAN. During our research, we have found the following three ways to move the eHealthcare data from the testbeds to compute facilities. (i) A server pull-based data transfer technique using SCP or rsync is more useful as it circumvents the need to use a VPN. HPC facilities more often than not operate within their virtual network and hence require a VPN which may not always be feasible for a few reasons. Firstly, the hardware on the wireless testbeds may not always support a VPN, secondly, most VPN setups include duo-factor authentication which eliminates the possibility of automation and finally, even if such a setup is possible, the established VPN connection needs to be alive for the duration of the experiment. Here, the Linux instance running on compute infrastructure pulls the eHealthcare IoT data from the gateway node on the wireless testbeds intermittently during certain scheduled times. (ii) Push-based data transfer technique where the gateway node on the wireless testbed pushes the aggregated eHealthcare data collected from the sensors to compute endpoint (HPC or cloud) at regular intervals.

When using an HPC compute infrastructure for data processing, this type of data transfer often requires a VPN connection establishment between the HPC facility and the gateway node on the wireless testbed as the HPC resources are only accessible within its virtual network (i.e. can only be accessed using their private IP addresses). In our research, we have found that commercial cloud platforms such as Amazon AWS and Azure do not require a VPN to realize a push-based data transfer. (iii) Using cloud-based Software-as-a-Service (SaaS) such as Globus or xRootD. This software is more applicable when the amount of data that needs to be transferred is quite large and requires high network bandwidth. A well-known example of such data is the experimental data from the particle accelerators at the Large Hadron Collider



(LHC). However, most eHealthcare scenarios would not benefit from such software as the amount of data to be transferred is relatively low. The overhead of the software setup far outweighs any benefit they offer for an eHealthcare scenario.

b. Data Processing

We utilize two types of separate compute facilities, high-performance computing (HCC) and cloud computing (AWS). Each type of computing infrastructure has its own advantages and disadvantages that may prompt the experimenter to choose one over the other. Both are capable of providing multiple instances of Linux compute nodes with flexible memory and storage options. Additionally, both support high network bandwidth to allow for seamless data collection from the testbeds.

Deep Learning supported Sleep Study

Sleep is classified into five stages: wake, N1, N2, N3, and REM. N1 to N3 are considered non-rapid eye movement (NREM) sleep, with each stage being deeper than the previous. About 75% of sleep is spent in non-rapid eye movement (NRM) stages, with the majority occurring in the N2 stage. There are normally 4 to 5 sleep cycles in a typical night, with sleep stages progressing in the following order: N1, N2, N3, N2, REM. For our deep learning application experiment, we chose to use the TinySleepNet, an open-source deep learning-based model built using TensorFlow and Python3. TinySleepNet trains over raw, single-channel electroencephalogram (EEG) data for sleep stage scoring. The input data format is in the form of European Data Format (EDF), a widely used data format for storing medical time series data. The dataset of choice was the Sleep-EDF dataset to simulate the data collection from the IoT sensors connected to the remote testbeds. The Sleep-EDF dataset (8GB) contains sleep recordings, EEG, EOG, respiration and body temperature data. TinySleepNet is only an example usecase chosen for our proof-of-concept and hence, virtually any eHealthcare usecase can be substituted in its place.

Experimental Setup

1) Scenario 1- High-Performance Computing (HPC): Figure 69 presents experimental scenario 1. Scenario 1 is based on the COSMOS testbed while scenario 2 is based the on POWDER testbed. Scenario 1 is identified by the HPC as the primary data processing or compute infrastructure. We chose the Holland Computing Center (HCC) located at Lincoln, Nebraska as the HPC infrastructure for data processing. We configured a virtual machine with 16 vCPUs, 64GB memory, 256GB storage and The HCC compute node running Linux server instance collects the data from the wireless testbeds. Once the data collection for that run is completed, TinySleepNet is run on the collected data to determine the sleep stages and sleep patterns of the hospital patients. The HCC node pulls the aggregated eHealthcare data from the wireless testbeds (POWDER or COSMOS) and trains the TinySleepNet model on that data.

After the training stage is finished, the prediction stage begins on the real eHealthcare data (also simulated) and the predicted results can be stored and



uploaded to any eHealthcare database or portal. Medical personnel can access these results and make inferences, diagnosis or prescribe modes of treatment to the patients.

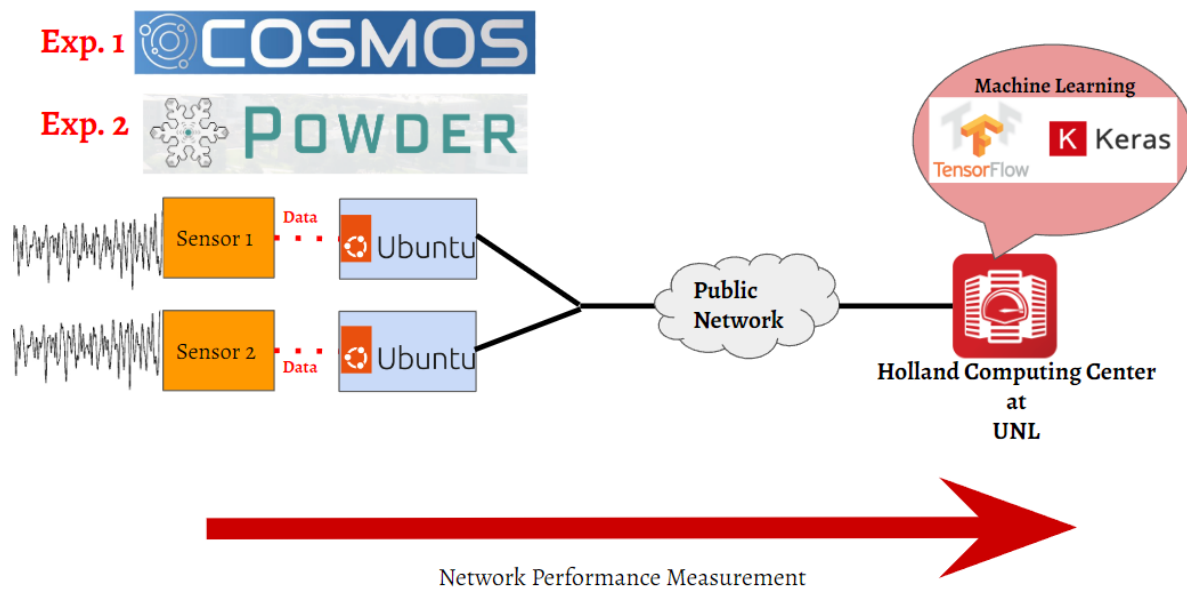


Figure 69: Experimental Scenario 1 for HPC

2) Scenario 2- Cloud Computing: We created a EC2 Linux server instance on AWS and used it to collect data transferred from the POWDER nodes. We used three POWDER nodes in our experiments to simulate the distributed data transferring process. All the data has been stored in our AWS instance. The average bandwidth between the POWDER nodes and AWS instance was found to be 50.58 Mbits/sec.

The AWS EC2 instance type we chose is g4dn.2xlarge, which comes with 8 vCPUs and 32GB of memory. Our instance runs on Ubuntu 18.04. The instance is equipped with one NVIDIA T4 Tensor Core GPU that has 320 Turing Tensor cores, 2,560 CUDA cores, and 16 GB of memory. G4 instances are ideal for machine learning inferencing, computer vision, video processing, real-time speech, and natural language processing.

We leverage the deep learning model named TinySleepNet for sleep stage scoring based on raw, single-channel EEG data. The total training time for the TinySleepNet model is about 24 hours in our AWS instance. For the prediction results, the overall accuracy of the model is 84.6. The model also classifies the sleep stage into five categories, W, N1, N2, N3, and PEM. The per-class precision is 90.1, 45.2, 88.3, 88.1, and 83.8, respectively.

4.2 Issues Faced

We faced several issues while performing experiments on both the EU and the US testbeds. We directly reached out to the testbeds’ owners and the troubleshooting teams to resolve the issues which we discuss in this section.

There were a lot of discussions with the POWDER team regarding the wireless interfaces. In order to select specific nodes for experiments, many options were considered and discussed with the team on the POWDER user google group. Multiple hardware were checked and reserved in the POWDER portal to find a solution. In the Linux terminal inside hardware, we executed the lshw command to find hardware information of the nodes. A snapshot of the same is attached below which was posted to the POWDER team for investigation. This is because they suggested, that no hardware currently has antennas. During discussion, we came up with the idea of using an Intel chipset inside an OTA-NUC to use as a wireless interface card. They found it interesting and performed a physical inspection of those specific NUCs. After the physical inspection, they confirmed having Intel wireless cards on this hardware and sent information of lshw of this OTA-NUC. We were then allowed and advised to use these NUCs for an experiment.

The next hurdle was configuring the Rspec. The node type of indoor ota-nuc was different; it was nuc8559 and not nuc5300 which we were using. We modified everything as per requirement and were able to set up 4 wireless nodes on POWDER for experiments, and it was found out that most of them didn't have the wireless card which was Intel Wireless AC- 9560.



Kirk Webb <kwebb@cs.utah.edu>

to powder-users ▾

Hello Saish,

To clarify, when you say "wireless ad hoc network", you mean 802.11 wireless ad hoc? I think it would be helpful to hear if people on this list have successfully run 802.11 stacks on POWDER using SDR resources. Timing constraints for 802.11 are quite tight, and therefore difficult to meet with processing that includes a software-based PHY running on general purpose compute. I am not aware of anyone on the POWDER team who has done this yet, but perhaps other users have.

The indoor OTA lab resources are probably best suited for this work if anything is (see the "indoor OTA lab" table under "Experiments -> Resource Availability"). Assuming you want homogenous resources, you could allocate the four NUC+B210 devices, or the four X310 radios (with associated compute connected over Ethernet). It is not clear to me if one type or the other of these radio resources would be more suitable for your work (if any).

BR,
-Kirk



Saish Urumkar <urumkars@tcd.ie>
to powder-users ▾

Hi kirk,
I was thinking of Intel built in wireless module chipset in OTA-NUC to work as adhoc wireless device.

Thanks
Saish

Kirk Webb <kwebb@cs.utah.edu>
to powder-users ▾

Hmm, interesting idea. We don't have antennas connected to those currently, and I haven't checked to see if the specific NUCs we have in the OTA lab have those WiFi interfaces. Let me talk with the team and see if this is something we can provision for you. As things are setup right now though, you won't be able to do this.

-Kirk

Kirk Webb <kwebb@cs.utah.edu>
to Kirk, powder-users ▾
Saish,

27 Jan 2022, 23:48 ☆ ↶ ⋮

The WiFi interfaces are present on the indoor OTA lab NUCs (confirmed by physical inspection). Attached is a snippet from the output of 'lshw' that was run on 'ota-nuc3' a while back. The other OTA lab nucs show the same. They should all be connected to (internal) antennas as well. You should be able to find and use the Intel WiFi interfaces 'ota-nuc1' through 'ota-nuc4'.

The node you ran 'lspci' on appears to be one of the NUCs that are part of the PhantomNet controlled RF matrix setup. Those do not have WiFi interfaces, and even if they did, would not be useful as each NUC+B210 in the PhantomNet setup is inside of its own RF isolation enclosure, with only the RF ports for the B210 (channel A, TX/RX and RX2) plumbed through the matrix.

-Kirk

Topology View List View Powder Map Manifest Graphs node1 x node2 x node3 x node4 x

```
Setting up bzip2 (1.0.8-2ubuntu1) ...
Setting up libxml-sax-expat-perl (0.51-1) ...
update-perl-sax-parsers: Registering Perl SAX parser XML::SAX::Expat with priority 50...
update-perl-sax-parsers: Updating overall Perl SAX parser modules info file...
Replacing config file /etc/perl/XML/SAX/ParserDetails.ini with new version
Processing triggers for install-info (6.7.0.dfsg.2-5) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
Processing triggers for systemd (245.4-4ubuntu3.15) ...
Processing triggers for man-db (2.9.1-1) ...
root@node1:/users/saish# lspci
00:00.0 Host bridge: Intel Corporation Broadwell-U Host Bridge -OPI (rev 09)
00:02.0 VGA compatible controller: Intel Corporation HD Graphics 5500 (rev 09)
00:03.0 Audio device: Intel Corporation Broadwell-U Audio Controller (rev 09)
00:14.0 USB controller: Intel Corporation Wildcat Point-LP USB xHCI Controller (rev 03)
00:16.0 Communication controller: Intel Corporation Wildcat Point-LP MEI Controller #1 (rev 03)
00:19.0 Ethernet controller: Intel Corporation Ethernet Connection (3) I218-LM (rev 03)
00:1b.0 Audio device: Intel Corporation Wildcat Point-LP High Definition Audio Controller (rev 03)
00:1d.0 USB controller: Intel Corporation Wildcat Point-LP USB EHCI Controller (rev 03)
00:1f.0 ISA bridge: Intel Corporation Wildcat Point-LP LPC Controller (rev 03)
00:1f.2 SATA controller: Intel Corporation Wildcat Point-LP SATA Controller [AHCI Mode] (rev 03)
00:1f.3 SMBus: Intel Corporation Wildcat Point-LP SMBus Controller (rev 03)
root@node1:/users/saish#
```

In Fed4Fire, there were a lot of challenges concerning the reservation. For w-ilab2.t they require a lower version of Google Chrome which is version 80. In w-ilab1.t NUCs support wireless setup whereas in wilab2 both APU and zotack nodes support wireless interfaces which have 2 interfaces. During setting up the wireless ad-hoc network one of the hurdles that we faced was an issue with olsrd not being able to set up routes for a prolonged time and used to reset it in a few seconds. We found out about the issue with the version of OLSRD not supporting a hello interval and hello validity time. After finding out the version that supports this, we did the configuration in the OLSRD config file and also did a more brief configuration of parameters. This issue solved the problem of routes being deleted.

In the jFed experiment tool, we tried setting up 9 nodes with different channels for ad-hoc networks. In order to avoid interference we used channel 1,6,11 for 2.4GHz and channel 36,40,44,48, 149, 153, 157 on 5GHz. It was found out that channels

52,56,till channel 144 needed dfs and radar detection. The problem with dfs is that it requires scanning for available channels. Sometimes the channel can even take 10 minutes. These dynamic frequencies sharing specific channels also have license constraints causing further problems. Another problem is in service monitoring that happens for finding the presence of radar signals appearing on the channel. Detection of RADAR further causes channel shutdown.

Link:-

<http://wifinigel.blogspot.com/2018/05/the-5ghz-problem-for-wi-fi-networks-dfs.html>

```
* 5260 MHz [52] (23.0 dBm) (no IR, radar detection)
* 5280 MHz [56] (23.0 dBm) (no IR, radar detection)
* 5300 MHz [60] (23.0 dBm) (no IR, radar detection)
* 5320 MHz [64] (23.0 dBm) (no IR, radar detection)
* 5500 MHz [100] (23.0 dBm) (no IR, radar detection)
* 5520 MHz [104] (23.0 dBm) (no IR, radar detection)
* 5540 MHz [108] (23.0 dBm) (no IR, radar detection)
* 5560 MHz [112] (23.0 dBm) (no IR, radar detection)
* 5580 MHz [116] (23.0 dBm) (no IR, radar detection)
* 5600 MHz [120] (23.0 dBm) (no IR, radar detection)
* 5620 MHz [124] (23.0 dBm) (no IR, radar detection)
* 5640 MHz [128] (23.0 dBm) (no IR, radar detection)
* 5660 MHz [132] (23.0 dBm) (no IR, radar detection)
* 5680 MHz [136] (23.0 dBm) (no IR, radar detection)
* 5700 MHz [140] (23.0 dBm) (no IR, radar detection)
* 5720 MHz [144] (23.0 dBm) (radar detection)
```

An unexpected problem that was faced during performing experiments in jFed was an active slice with no resource allocated or a phenomenon that caused resource deletion. An issue was raised with the jFed team, and it was found out that there was a gap of 1 second between the reservation of nodes and that actually led to the deletion of resources and hence loss of complete experimentation.

There was another issue that led to the non-availability of certificates which was a kind of upgrade by the jFed team. This issue came to light when the issue was raised with the team.

Vincent Sercu (UGent-imec)
to me, Wim, jfedbugreports@ilabt.imec.be, jfed-bugreports@intec.ugent.be, Pieter ▾
Hi Saish,

I see there is a small gap between the 2 reservations, although only 1 second it could be that the slot is ended and the experiment terminated. I tried to reach your nodes and they are not ping-able anymore so I am afraid something like that happened.

Vincent Sercu (UGent-imec)
to me, Pieter, Wim ▾
Hi Saish,

For wilab2 you'll have to use an older browser, due to legacy TLS versions.
Please see the red text on top of the page, any chrome < version 80 should be fine.

Best regards,
Vincent Sercu



Vincent Sercu (UGent-imec)

to Pieter, me ▾

At the moment there are some technical difficulties regarding certificates. We are trying to resolve it asap, will keep you posted.

Best regards,
Vincent

Vincent Sercu

Further, adding a large pool of resources from testbeds such as w-ilab1.t, w-ilab2.t was challenging as we could not reserve more than 10 wireless nodes from each testbed.

Several issues have been raised when applying reinforcement learning on the real testbed such as w-ilab1.t and w-ilab2.t. First, w-ilab2.t showed compatibility issues during the setup of the python environment and libraries installation. Due to the old hardware and drivers installed on the controller node in W-ilab2.t, we had to install older and unsupported python versions and TensorFlow libraries. As a consequence, the code tested and installed in our local machines has been changed accordingly to make it compatible.


The second main class of issues regards the testing of the code on the testbed is that W-ilab.t testbeds provide a GUI interface, jFed, that allows the user to recover the experiment anytime before the test expiration date (that comes from the nodes reservation, as explained in a previous deliverable). In addition, jFed allows sharing of a test between several users, and the recovery function is needed to help the users coordinate their time and use the testbed in different time slots. However, this useful feature has several issues and bugs that we reported to the jFed and W-ilab.t teams. Some of them have been solved after some iterations. Our general feedback on this feature is that the recovery function is essential to perform long tests and trials on the testbed, especially when a time-consuming algorithm such as DQN is involved.

Here below we provide two general examples of issues faced when using the recovery function:

- the sharing/unsharing function is not completed
- the tests are shared but they cannot be recovered


The above issues have been reported to the W-ilab.t team via the bug report function in jFed and solved with the email exchange.



Select the experiment(s) to recover. 

and project: All projects

Active	Experiment name	Project	Expiration
<input checked="" type="checkbox"/>	iextconn	sdnnfv	in 6 days and 12 hours
<input type="checkbox"/>			
<input type="checkbox"/>			
<input type="checkbox"/>			
<input type="checkbox"/>			
<input type="checkbox"/>			
<input type="checkbox"/>			
<input type="checkbox"/>			

 Use SHIFT and CTRL for multi-select Show Exact Expire ti...

Also, sometimes when the reservation is made the slice becomes unavailable because of node deletion. Please see the email chain below:

Hello,

I've noticed that you have reserved the wireless test bed @ our data centre in iGent for more than a month. However, typically the nodes that you have reserved are used for educational purposes during the month of March. The time period for this labwork is March 9th to March 31st. May I ask you to shorten your current work & reservation to two weeks, and plan further work in the month of April if that would be necessary?

Kind regards,
 Andy Van Maele
 labwork organiser @ UGent

PS:

Begin: Mon Feb 21 2022 14:53:00 GMT+0100 (Central European Standard Time)
End: Thu Mar 31 2022 22:59:59 GMT+0200 (Central European Summer Time)
Email: saish15@gmail.com

Fixed nodes in experiment (10):

nuc0-20	nuc0-21
nuc0-22	nuc0-24
nuc0-25	nuc0-27
nuc0-30	nuc0-31
nuc0-33	nuc0-34

Begin: Thu Feb 24 2022 12:00:00 GMT+0100 (Central European Standard Time)
End: Thu Mar 31 2022 22:59:59 GMT+0200 (Central European Summer Time)
Email: saish15@gmail.com

Fixed nodes in experiment (10):

nuc0-10	nuc0-11
nuc0-14	nuc0-15
nuc0-3	nuc0-4
nuc0-5	nuc0-6
nuc0-8	nuc0-9

..

Andy Van Maele
 Ghent University - imec
 IDLab
 iGent Tower - Department of Information Technology
 Technologiepark-Zwijnaarde 126, B-9002 Ghent, Belgium
 T: +32 (0)9 331 49 46
 E: andy.vanmaele@ugent.be





Saish Urumkar <saish.15@gmail.com>
to Wim, jfedbugreports@ilabt.imec.be, jfed-bugreports@intec.ugent.be

Hi wim,
It seems you last renewed your experiment to **2022-03-14T11:26:43Z**.
That date has passed, so I'm afraid that means that your experiment has been automatically deleted...
please check its month of february now not march **2022-03-14T11:26:43Z**.
The experiment is expiring next month

...



Wim Van de Meerssche <wim.vandemeerssche@ugent.be>
to Pieter, Vincent, me, jfedbugreports@ilabt.imec.be, jfed-bugreports@intec.ugent.be

Hi,

Aha, sorry, I was too fast. You're right!

In that case, it's best if the **WiLab** people have a look at this (I've put them in CC).
The experiment "urn:publicid:IDN+ilabt.imec.be:sdnfv+slice+wifiexp" should still be running.
But the WiLab1 AM is replying "No such slice here", which should only happen if it has been deleted.

Best Regards,
Wim

On Tue, Feb 15, 2022 at 12:14 PM Saish Urumkar <saish.15@gmail.com> wrote:

Hi wim,

...

Re: jFed Feedback #2593 (FEDIBBTDEV-4467): Issue recovering experiment Inbox x



Wim Van de Meerssche <wim.vandemeerssche@ugent.be>
to saish15@gmail.com, jfedbugreports@ilabt.imec.be, jfed-bugreports@intec.ugent.be

Dear Saish Pramod Urumkar,

At first glance, this seems to have been a temporary network issue.
- The call failed with: UnknownHostException: www.wall2.ilabt.iminds.be: Temporary failure in name resolution
- However, an automatic test that happens when you send the bugreport could correctly resolve that address.
- Our monitoring does not show that wall2 or it's DNS was down or unreachable

So just try again. Hopefully, this was just a temporary thing.
If not, try to enable/disable the jFed proxy server. If it's enabled, it also handles DNS (which can cause issues if the proxy malfunctions, but can fix them if you have network issues).

Best Regards,
Wim Van de Meerssche

On Tue, Feb 8, 2022 at 5:36 PM jfedbugreports@ilabt.imec.be <jfedbugreports@ilabt.imec.be> wrote:

The jFed Experimenter GUI sent feedback for user saish15@gmail.com

Feedback:

Type: Bug Report

Related testbeds: urn:publicid:IDN+wall1.ilabt.iminds.be+authority+cm

Description:

i am not able to recover my experiment it shows basic call failed

Details:

User URN: urn:publicid:IDN+ilabt.imec.be+user+saish15

****WARNING**** This is not a wall2 user, yet a mail to the public list was requested. (will not send mail to public list)

User email: saish15@gmail.com

jFed Version: 6.4.1 - build #210526-35046 - git commit #8e8a23ef9c43c69b8a926000b7e44650a224349a on master

jFed Environment: Linux 5.13.0-28-generic amd64 - Java 14.0.2 (AdoptOpenJDK)

Send to mailinglist: Yes

Included calls: 60

Includes screenshot: Yes: <https://flsmonitor-api.fed4fire.eu:9443/bugreport/2593/screenshot>

jFed admins can find additional details at: <https://flsmonitor-api.fed4fire.eu:9443/bugreport/2593> and <https://flsmonitor.fed4fire.eu/bugreport/detail/2593>

jFed admins can access the Jira issue: <https://lbcn-jira.intec.ugent.be/browse/FEDIBBTDEV-4467>

The permissions in POWDER for the nodes were revoked and needed to be re-instantiated.



Regarding Reservation of OTA-NUC External Inbox x

Saish Urumkar <urumkars@tcd.ie>
to Kirk, Sachin, Avishek ▾

Sat, 4 Jun, 21:43 ☆ ↶ ⋮

Hi Kirk,
We are again doing experiments on **POWDER** testbed involving use of 4 OTA-NUC. As these are only nodes having wireless interfaces it would be great if you can provide us the permission to perform experiments on these nodes as we are trying to perform openswitch configuration connected via wireless interface. Following is the message I get when I try to reserve the nodes.

Thanks & Regards,
Saish Urumkar

Kirk Webb <kwebb@cs.utah.edu>
to me, Kirk, Sachin, Avishek ▾

Mon, 6 Jun, 16:21 ☆ ↶ ⋮

Hello Saish,

We recently (within the last month or so) added some checks for over-the-air license acknowledgement, and a few projects have been caught up in that. I see that you clicked on the "request access" link, and that Sachin filled out the form and acknowledged the over-the-air use terms and conditions. I have re-enabled OTA rights for the ATLANTIC-eVISION project.

In POWDER for 3-day reservation also can only be done on weekends.

POWDERWIRELESS.NET: Reservation Info Request Inbox x

portal-reservations@powderwireless.net
to me ▾

Mon, Jun 13, 4:21 PM ☆ ↶ ⋮

We need information about your reservation group in project ATLANTIC-eVISION.

Hello Saish,

Is this a test you can do starting on a Friday and over the subsequent weekend? That will have less impact given the 3 day reservation length on other users. However, if you already have resources arranged at WLab for this time period, let us know and we'll go ahead and approve this reservation.

Start: 06/14/22 22:00:00 -0600
End: 06/17/22 22:00:00 -0600
Cluster Resources:
Emulab: ota-nuc1,ota-nuc2,ota-nuc3,ota-nuc4

See: <https://www.powderwireless.net/resgroup.php?edit=1&uuid=3317b62d-eb12-11ec-b318-e4434b2381fc>

Further, we did not get continuous reservations in POWDER for more than 1 day. We got reservations after some gap. This required configuration of testbeds multiple times and hence, lots of time was invested in configuring the testbed. We also made the automatic scripts. However, as there were some hardware/software issues occurred every time, we invested time in configuration multiple times. See below the email chain:



POWDERWIRELESS.NET: Reservation Info Request Inbox x

portal-reservations@powderwireless.net
to me, portal-ops

Mon, Mar 28, 4:29 PM

We need information about your reservation group in project ATLANTIC-eVISION.

Can you tell us more about this reservation? As we have said in the past, long solid blocks of time like this across all four OTA NUC devices essentially block out all other users. As the OTA lab is a popular resources, this would cause quite a burden on other people. The only way this reservation can be justified is if you plan to conduct continuous, long-running experiments. If that is the case, we need you to describe in detail the experiments you are doing, with particular emphasis on their long-running nature.

Start: 03/30/22 13:00:00 -0600
End: 04/07/22 13:00:00 -0600
Cluster Resources:
Emulab: ota-nuc1,ota-nuc2,ota-nuc3,ota-nuc4

See: <https://www.powderwireless.net/resorqup.rh?ref=16uuul=bf383ab-ae63-11ec-b318-e4434b28b1fc>

saish urumkar

Hi. Our previous excitement of wireless setup was successful. Now we are moving ahead to do automation node configuration for openflow and neighbour detection.

Kirk Webb <kwebb@cs.utah.edu>
to me, portal-reservations, portal-ops

Mon, Mar 28, 4:48 PM

Can this be broken up into a number of smaller reservations (not consecutive)? It doesn't sound like you need 24x7 operation. If you script your setup steps, it should not be difficult to get going with new experiments.

Kirk
...

Kirk Webb <kwebb@cs.utah.edu>
to me, Kirk, portal-reservations, portal-ops

Well, that is complex given that we don't have COTS WiFi access points available, and I'm not entirely sure SDR-based WiFi is possible or not (we have mainly worked with 4G/5G and custom GNU Radio communications ourselves). This is another good reason to post on the Users list and give a more complete description of what you want to do. Others there may be able to share their advice and/or experience with WiFi on POWDER.

-Kirk

On Tue, Jan 25, 2022 at 11:35 AM saish urumkar

...

saish urumkar <saish.urumkar21@gmail.com>
to Sachin

From the powder team regarding experiment

...

As reported above, it was very difficult to reserve resources on POWDER. We were only able to reserve resources for a few hours on a weekend. This has created problems in setting up integration setup with all the testbed involved with all functionality implemented. We have already performed the integration setup in the previous months with some functionality. However, the implemented IoT application is still needed to be tested with all the testbeds involved. We now leave it as future work.

Moreover, all these interactions and troubleshooting steps enhanced our knowledge about the testbeds significantly. Along with our own observations, while performing experiments and these direct interactions with the testbed owners, we were able to perform the experiments on the testbeds and able to report the results provided in this deliverable.



5 Present and Foreseen TRL

The overall current technology readiness level (TRL) of this project can be deemed between TRL 3 and TRL 4 according to the European Union's definition of TRLs. TRL 3 stands for any technology that has "Experimental proof of concept" and TRL 4 stands for "Technology validated in lab." Definitely, in this project we have performed experiments to establish a proof-of-concept of a) feasibility of inter-testbed operations between the EU and the US testbeds; b) automatic configuration of OpenFlow in wireless ad-hoc networks; c) establishment of reinforcement-learning-based path discovery in OpenFlow networks, and d) feasibility of an edge-cloud model employing an ensemble machine learning algorithm for detecting attacks on the Internet of Things (IoT). Therefore, from this perspective, our project's current TRL level is TRL 3.

However, if we want to argue that the testbeds that we have worked on in this project, e.g., wilab1 and 2, CityLab, POWDER, and COSMOS, are so much advanced in terms of their hardware and software capabilities, that they can be termed as virtual "labs" meeting certain standards. In that aspect, we can even say that the current TRL level of our project is TRL 4.

In the future, we definitely foresee our project's findings reaching at least a TRL level of TRL 5. TRL 5 relates to "Technology validated in relevant environment (industrially relevant environment in the case of key enabling technologies)." We have mentioned in the impact section later that our combined network connects us with different industries that are working in this relevant space. Therefore, there is a chance that we further work on the ideas mentioned in this project with our industrial collaborators and validate them in "industrially relevant environments".

6 Exploitation, Dissemination and Communication Status

The following papers have been accepted or published so far with the work done on the NGIAtlantic project:

1. S. Sharma, S. Urumkar, G. Fontanesi, B. Ramamurthy, and A. Nag, "Future Wireless Networking Experiments Escaping Simulations", *Future Internet*. 2022; 14(4):120. <https://doi.org/10.3390/fi14040120> (Published)
2. V. Tomer and S. Sharma, "Detecting IoT Attacks Using an Ensemble Machine Learning Model" *Future Internet*, 2022; 14(4):102. <https://doi.org/10.3390/fi14040102> (Published)
3. S. Sharma, A. Nag and B. Ramamurthy, "Cross-Atlantic Experiments on EU-US Test-beds," *IEEE Networking Letters*, doi: 10.1109/LNET.2022.317771 (Published)
4. V. Tomer and S. Sharma, "Experimenting an Edge-Cloud Computing Model on the GPULab Fed4Fire Testbed", 28th IEEE LANMAN Poster/Demo 2022 (Accepted)



5. S. Urumkar, G. Fontanesi, A. Nag, and S. Sharma," Demonstrating Configuration of Software Defined Networking in Real Wireless Testbeds", 28th IEEE LANMAN Poster/Demo Session, 2022 (Accepted)
6. S. Sharma, S. Urumkar, G. Fontanesi, V. S. Karanam, B. Hu, B. Ramamurthy and A. Nag, "Towards Emulation of Intelligent IoT Networks on EU-US Testbeds", 23rd International Seminar on Intelligent Technology and Its Applications (ISITIA), 2022 (Accepted)

We have also open-sourced our codes and demo videos as reported in the following links:

CODE or Videos Released:

1. Experimenting an Edge-Cloud Computing Model on the GPULab Fed4Fire Testbed, <https://github.com/VikasTomar32/LANMAN>
2. Demonstrating configuration of software defined networking in wireless testbeds, https://www.youtube.com/watch?v=kAkrT95tRb4&ab_channel=SaishUrumkar
3. GNN+DQN code in https://github.com/GianFont/RL_routhOpt .
4. Controller Automatic Configuration
Code:<https://bitbucket.org/saish15/olsrd2/src/master/>
5. Client node automatic configuration code:
https://bitbucket.org/o2cmf-work/olsrd_client/src/master/

We also presented our results at the Webinar of the [5th opportunity to apply the NGI Atlantic Open Call](#). Furthermore, we presented our results in the [IoT week 2022](#), held in Dublin, Ireland from June 20-23, 2022. We also plan to exploit the outcomes of this project after the project officially ends. Two journal papers are in the pipeline one led by the EU team and the other led by the US team, which we plan to publish in the next few months. Furthermore, with the foundations built from our experiments in this project, we might collaborate to participate in future NGIAtlantic calls or bigger EU-US collaborative projects, for example, the NSF-SFI joint funding program between the US and the Republic of Ireland.

7 Impacts

- Impact 1: Enhanced EU – US cooperation in Next Generation Internet, including policy cooperation.

Through our experiments performed so far, this project has educated the researchers at the University of Nebraska-Lincoln (UNL), USA to set up and operate experiments on the Fed4Fire testbeds in the EU, i.e., the CityLab and the Wilab1 and Wilab2 testbeds. Similarly, the EU team, while developing and performing their experiments on the US testbeds, developed comprehensive knowledge on the minute operational details of the POWDER and the COSMOS testbeds. The US team has provided expertise in SDN control and testbed experimentation, enabling the EU team to perform testbed



experiments. This partnership will mutually benefit and forge strong relationships that promise to drive further collaboration in the future. In fact, the EU and the US teams are already exploring participating in a joint Science Foundation Ireland (SFI) and NSF project, based on the foundations laid by this project. Some important outcomes of this project are:

1. We compared the performance of different testbeds for the same experiments and reported the compatibility of different testbeds with respect to the proposed experiments.
2. We discovered the topology for inter-testbed connectivities from the EU and the US. CityLab is connected to the Belnet network through a router at Antwerp University, which connects CityLab to the Belnet network. Belnet is a special-purpose network infrastructure that connects external educational institutions, research centres, scientific institutes, and government centres in Belgium. Belnet is directly connected to an independent US network called Internet2, which is dedicated to research and education as well. Internet2 directly connects to the POWDER testbed.
3. We demonstrated successfully the automatic deployment of OpenFlow in an ad-hoc wireless network where some wireless nodes are not directly connected to the controller.
4. We also successfully demonstrated, in a similar ad-hoc network setting as mentioned above, a reinforcement-learning-based path discovery from the OpenFlow controller to the wireless nodes and vice versa, obeying some KPIs e.g., latency and bandwidth demand
5. Experimenting with EU and US testbeds can be useful when creating edge computing-type scenarios in which some functionality can be moved closer to users to allow for faster decisions and other functionality can be kept at a faraway testbed.
6. Resource intensive functions like ML training and normal forwarding functions of a controller should not be running in the same physical node. Since we are integrating EU and US testbeds, we can use the US locations for resource-intensive functions while running normal forwarding functions in the local EU controllers.
7. Using GPULAB Fed4Fire testbed we demonstrated that an edge-cloud model employing an ensemble machine learning algorithm for detecting attacks on the Internet of Things (IoT), can be deployed seamlessly on the GPULAB testbed. We compared experimentation times and other performance metrics of our model based on different characteristics of the testbed, such as GPU model, CPU speed, and memory.

The above conclusions are a result of exemplary instances of integrating so many diverse testbeds i.e., in terms of capabilities (pure wireless, or standard TCP/IP, platforms, and functionalities, and being managed by ML-assisted SDN. Both EU and US researchers will benefit from the volume of new knowledge created through this large-scale intercontinental experimental framework.



Additionally, the European partners got the unique opportunity to perform experimentation on one of the world's largest and most advanced wireless testbeds, which complement those available to them locally within Europe. In summary, we have, through our experimentations so far, we laid a strong foundation for more advanced cross-Atlantic experiments enabled by SDN and ML. This will inspire more EU-US collaborations in terms of cross-Atlantic networking experiments and can also foster the development of intercontinental large-scale testbeds in future.

- Impact 2: Reinforced collaboration and increased synergies between the Next Generation Internet and Future Internet programmes.

This project establishes collaboration between three PI's Dr Sachin Sharma and Dr Avishek Nag from the EU and Dr Byrav Ramamurthy from the US. Dr. Sharma has worked for several EU and Flemish projects: FP7-SPARC, FP7-OFELIA, FP7- CityFlow, FP7-UNIFY, FP7-CleanSky and MECANO where he extensively used the virtual wall testbed within Fed4Fire. He is also an associate investigator at the CONNECT Centre. Dr Nag on the other hand has worked on the FP7-DISCUS project with many EU collaborators. Dr Nag was also a researcher in Ireland's biggest telecom networks research centre CONNECT where he worked on developing an LTE testbed with YouTube. CONNECT currently houses one of the biggest 5G testbeds that has links with Fed4Fire and the US testbed COSMOS. Dr Nag is also an alumnus of the same research group (i.e., Prof. Biswanath Mukherjee's networks research lab) at UC Davis as Prof Ramamurthy who is the US PI of this project. From these enriched networks of the three PIs we have identified a few teams who have already worked on NGI experimental projects from the Open Call 2, e.g., a team between CONNECT research centre, Politecnico di Milano, and the University of Arizona. We have also identified some teams from within our close networks working on NGI projects from the ongoing Open Call 3, involving UPC, Spain and UC Davis. We have reached out to some of these teams and are planning to carry forward these discussions once our results are disseminated and our intellectual properties are protected. Our plan is to first identify some future network use cases, like ML-assisted 6G networks, quantum communication networks etc. Then based on our and the potential collaborators' expertise developed from the NGI projects or otherwise, prepare a joint action plan to reinforce collaborations and foster increased synergies between the Next Generation Internet and future Internet programmes.

- Impact 3: Developing interoperable solutions and joint demonstrators, contributions to standards.

The mere philosophy of this project is based on interoperability as it establishes the feasibility of OpenFlow and SDN for unified control of wireless testbeds spanning over two continents. OpenFlow and SDN were originally suited for wide-area wired networks, and we emulated it for the first time on practical-scale networks. Different technologies and protocols need to be



interoperated with a certain scope of creating new standards. The scope of new standards lies in the fact that SDN/OpenFlow has to be redefined to support machine learning algorithms and policies as well as support automatic discovery of wireless devices. In our implementation of SDN and ML solutions for IoT networks, we used only standard protocols. For example, OpenFlow, OVS-DB and OLSR protocols are used for the implementation of automatic configuration of SDN in IoT networks. Moreover, all the data is collected using standard protocols such as OpenFlow and OVS-DB. Nevertheless, ML decisions are incorporated using the northbound API of SDN and ONF (Open Networking Foundations is currently putting efforts to standardise this API).

Our contribution is also in the application of open-source reinforcement Learning libraries, such as Tensorflow and Keras, in our testbeds for running the RL algorithm. Keras is a deep learning API written in Python, running on top of the machine learning platform Tensorflow and one of the standard API for RL.

- Impact 4: An EU - US ecosystem of top researchers, hi-tech start-ups / SMEs and Internet-related communities collaborating on the evolution of the Internet

The PIs of this project, owing to their combined professional network, has reachability to a lot of tech startups and SMEs. For example, Dr Avishek Nag and Dr Byrav Ramamurthy's PhD advisor Prof Biswanath Mukherjee is the founder of a startup based in Northern California, called Ennetix Inc. which specialises in AIOps and Network Analytics. Our automatic IoT device discovery methodologies and machine-learning-based optimum path discovery in IoT networks can help them extend their solutions beyond enterprise-wide-area networks. Further, Dr Sachin Sharma also works as an associate investigator at the CONNECT Centre, where he collaborates with many companies such as INTEL and Bosch related to the topic researched in this project. Moreover, we can also collaborate with them in future NGI calls to extend our proposal to incorporate security and advanced and accurate data collection techniques from a live operational network. While the current results reported in this deliverable are early-stage, it provides enough promise to develop a successful prototype for an SDN and ML enabled automatic discovery of IoT nodes finding applications in healthcare, sensing, and weather monitoring and can stimulate the interests of several hi-tech start-ups / SMEs and Internet-related communities. In fact, in this context, it is worth mentioning that towards the last few months of the project we have implemented some use cases on AWS cloud as AWS agreed to provide some resources to our US partner, University of Nebraska Lincoln. The findings of our experiments on AWS are not reported in this deliverable, but that will be the part of our future exploitation and dissemination as mentioned in Section 6. However, this already evidences the impact of our project to enhance an ecosystem of top-level collaborations.



8 Conclusion and Future Work

This project compared a number of EU and US testbeds based on their (1) architecture, (2) resources available, (3) IoT capabilities, (4) data that can be collected, (5) limitations, (6) Software-Defined Networking (SDN) capabilities, (7) machine learning capabilities, and (8) practical experimental results. Benchmark and failure recovery experiments are performed and results are shown. Further, issues faced from each testbed are reported. Different testbeds have different resources available for wireless experiments, as shown by the results. In addition, because nodes in different testbeds have different resources available with respect to CPU, memory, bandwidth available, we achieved different results from each testbed experimentation. Furthermore, as the selection of nodes is highly dependent on the availability of nodes at the time of experimentation, results vary depending on the type of node selected. In addition, testbed experiments provide a realistic environment for experimentation. Therefore, it makes sense to use these nodes as experimentation platforms.

Apart from that, we also achieved the objectives of this NGIAtlantic project namely:

1. We experimentally demonstrated automatic configuration of SDN/OpenFlow in Wireless Ad hoc networks. This was achieved by implementing an automatic configuration method on testbeds. The efficiency of the method is calculated by measuring the automatic discovery time and data plane latency.
2. We achieved the best data-plane latency for an e-healthcare application. This was done by applying machine learning to find the best path from an IoT device to an IoT application which meets the latency and bandwidth requirements.
3. Recover from a failure when it occurs in different network topologies (ring, grid and mesh). This was achieved by implementing a restoration scheme and calculating the failure-recovery time. The failure-recovery time was calculated after the failure is introduced in the network. Most of the results in the literature are based on simulations. The results gathered using our emulations are unique, as these are measured in a set-up emulated on real testbeds.
4. We tested inter-testbed connectivity by performing experiments on EU and US testbeds. The inter-testbed connectivity was achieved by running different modules (IoT applications and sensor nodes) on different testbeds and using the public internet for connection. For example, we ran the controller at the COSMOS testbed and an IoT application at the virtual wall testbed. Further, wireless IoT scenarios will be created on the W-iLab.t, CityLab and POWDER testbeds.
5. We tested our secure IoT application using the GPULAB testbed. Further, our e-healthcare application was tested using a setup created on POWDER, COSMOS, and AWS servers.



The RL application for route optimization can be extended in several multiple ways.

As shown in the results section for the ML, the algorithm has been tested only on w-ilab1.t, due to the many issues encountered during the deployment of the algorithm on the testbed (see Section 4.2). This leaves space to derive and get many other results in the next future, experimenting with different values of the algorithm hyperparameters. The algorithm can be tested on w-ilab2.t and POWDER, where the environment is now ready for the DQN algorithm application. The objective of future works is to compare the performance of our algorithm with State-of-art routing algorithms and/or simpler tabular reinforcement learning algorithms, such as Q-learning.

Further, due to resource limitations (as discussed in section 4.2), we could not test our IoT application with the inter-testbed environment where w-ilab1.t, w-ilab2.t and POWDER testbed are connected to each other as discussed in section 4.1.5. In the future, we will perform this experiment. Moreover, we will use the findings of this project to implement more advanced edge-computing use cases and run more robust machine learning algorithms (e.g., considering the tradeoff between algorithm accuracy vs energy efficiency) on the EU-US inter-testbed topologies. Furthermore, we would like to explore the security and trust aspect of connecting these software-controlled nodes by using the principles of Blockchain.

9 References

- [1] S. Sharma, A. Nag, P. Stynes and M. Nekovee, "Automatic Configuration of OpenFlow in Wireless Mobile Ad hoc Networks," 2019 International Conference on High Performance Computing & Simulation (HPCS), 2019, pp. 367-373, doi: 10.1109/HPCS48598.2019.9188200.
- [2] P. Dely, et. al., "OpenFlow for Wireless Mesh Networks," ICCCN 2011.
- [3] Mininet." [Online]. Available: <http://mininet.org/>
- [4] "jFed." [Online]. Available: <https://jfed.ilabt.imec.be>
- [5] "Demonstrating Configuration of Software Defined Networking in Real Wireless Testbeds" [Online]. Available: <https://www.youtube.com/watch?v=kAkrT95tRb4>
- [6] Sharma, S.; Urumkar, S.; Fontanesi, G.; Ramamurthy, B.; Nag, A. Future Wireless Networking Experiments Escaping Simulations. Future Internet 2022, 14, 120. <https://doi.org/10.3390/fi14040120>
- [7] NSL-KDD Dataset: <https://www.unb.ca/cic/datasets/nsl.html>
- [8] GPULAB Fed4Fire EU Testbed: <https://www.fed4fire.eu/testbeds/gpulab/>
- [9] Fed4fire test-bed." [Online]. Available: <https://www.fed4fire.eu/>



[10] Powder Test-bed [Online]. Available: <https://powderwireless.net/>

[11] S. Sharma, A. Nag and B. Ramamurthy, "Cross-Atlantic Experiments on EU-US Test-beds," IEEE Networking Letters, doi: 10.1109/LNET.2022.317771

[12] Paul Almasan, José Suárez-Varela, Arnau Badia-Sampera, Krzysztof Rusek, Pere Barlet-Ros, and Albert Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case", arXiv preprint arXiv:1910.07421 (2019)

10 Glossary

Acronym	Full Form
5G	Fifth Generation (mobile/cellular networks)
NGI	Next Generation Internet
SDN.	Software Defined Networking
IoT	Internet of Things
OF-Config	OpenFlow Configuration Protocol
OVS-DB	Open vSwitch Database Management Protocol
DHCP	Dynamic Host Configuration Protocol
OLSR	Optimised Link State Routing
6LoPAN	IPv6 over Low -Power Wireless Personal Area Networks
NETCONF	Network Configuration Protocol
ARP	Address Resolution Protocol
GPU	Graphics Processing Unit
CPU	Central Processing Unit



Acronym	Full Form
DQL	Deep Q Learning
RPL	Routing Protocol for Low-Power and Lossy Networks
GRE	Generic routing encapsulation
VXLAN	Virtual Extensible Local Area Networks
ML	Machine Learning
MQTT	MQ Telemetry Transport
TU	Technological University
UCD	University College Dublin
SME	Small and Medium Sized Enterprises
PI	Principal Investigator
SDR	Software-defined Radio
BS	Base Station
FPGA	Field Programmable Gate Array
C-RAN	Centralised Radio Access Network
MIMO	Multiple Input Multiple Output
mMIMO	Massive MIMO
OMF	Orbit Management Framework
LTE	Long Term Evolution



Acronym	Full Form
OML	Orbit Measurement Library
FE	Fixed Endpoint
RSSI	Received Signal Strength Indicator
ONOS	Open Network Operating System
NETCONF	Network Configuration Protocol
API	Application Programming Interface
OLSRd	Optimised Link State Protocol Daemon
IPv4/IPv6	Internet Protocol version 4/version 6
NAT	Network Address Translation
ROC	Receiver operating characteristic
RMSE	Root mean square error
RRSE	root-relative squared error
MAE	mean absolute error
RAE	relative absolute error

