X-Lab: A Federated Testbed Infrastructure to Benchmark Geographically-Distributed Low-Power Wireless Systems

Markus Schuß*, Michael Baddeley†, Monika Prakash†, Carlo Alberto Boano*, and Kay Römer*
*Graz University of Technology, AT; †Technology Innovation Institute, AE
{markus.schuss; cboano; roemer}@tugraz.at {michael.baddeley; monika.prakash}@tii.ae

Abstract

While interest in low-power mesh networks for the Internet of Things (IoT) has proliferated over recent years, much of the focus in this area has considered the deployment of a single sensor network, or brokering data to and from a cloud-based server. Yet, particularly for industrial use-cases such as distributed sensor-actuator networks, there is a need to consider fully-federated IoT networks where devices may be required to send data over a mesh-cloud continuum directly to devices at remote sites while adhering to strict latency and reliability requirements. To allow experimentation in such settings we present X-Lab, an open-source federated testbed infrastructure for *end-to-end* benchmarking of low-power wireless protocols. Using X-Lab, we establish a 4000 km *cross-continent* setup between two sites and evaluate the end-to-end performance of two low-power wireless protocols. We find that despite the large distance between sites, mesh latency often dominates the latency of the Internet – which can directly account for as little as 22% in some protocols.

Categories and Subject Descriptors

B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design, Measurement, Experimentation, Performance.

Keywords

Benchmarking, Cloud, D-Cube, Dependability, End-to-end, Geographically-distributed, IoT, Testbeds, Wireless.

1 Introduction

IoT solutions often leverage the cloud for connecting different geographically-distributed deployments. Fig. 1 shows how multiple mesh networks may be connected through the Internet by means of edge devices acting as border routers

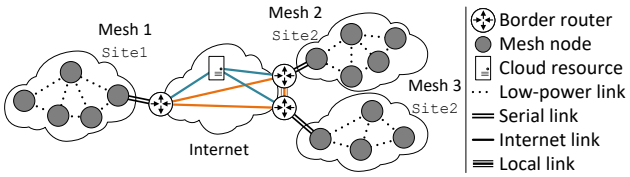


Figure 1: *Blue* links denote the classical cloud paradigm. *Orange* links allow edge devices to directly communicate.

(BRs). Links through the Internet shown in *blue* represent the *classical cloud paradigm*, where each site is connected to one or more central servers. However, such a paradigm may cause significant delays if an end-device in one instance needs to communicate with end-devices in another instance. The links in *orange* illustrate the approach of modern IoT deployments, which take advantage of *direct connections across the cloud*, allowing edge devices to exchange data directly (hence, at a lower latency) without losing access to the resources available in the cloud. Unfortunately, while the first approach has been extensively researched in the literature, few public testbeds exist that span across multiple sites in order to enable performance characterization over a mesh-cloud-continuum.

Challenges. One could naïvely assume that the latency and reliability of each part of the network is independent, with the system simply exhibiting the sum of all the parts, i.e., the sum of the latency and the product of the reliability of all links involved. However, strategies such as duty cycling which are employed by most low-power wireless protocols mean that the *latency depends on the alignment of the duty cycles of the different sites*, as a packet that arrives in a network after an active slot is delayed until the next one. As such, the uncertainty caused by congestion across the Internet link can have drastic implications for the system's performance. This calls for experimental infrastructures enabling a fine-grained performance characterization over a mesh-cloud continuum.

Lack of automation. Setting up experiments spanning multiple sites – and potentially also cloud resources – can cause significant overhead to the user, who may also introduce errors. While federated testbeds that allow the control of the devices on each site exist, all steps – from device setup to network configuration, traffic generation, and even collection of measurements – have to be performed manually.

Lack of an isolated network. While it would be possible to simply expose BRs directly to the Internet and let the user configure the necessary firewall rules, this would represent an

open call for attacks. In fact, even with a properly configured firewall, undesired traffic may still be forwarded into the low-power mesh network. Furthermore, depending on the type of Internet connection available at a site, the provision of only a single public IP utilizing network address translation (NAT) or the lack of IPv6 support (often used by low-power devices) may make the setup of end-to-end (E2E) experiments rather cumbersome.

Lack of security. While the lack of an isolated network puts the devices performing the experiment at risk, the ability to run arbitrary code on such devices may in turn expose the testbed infrastructure (and attached internal networks) to harm. While many federated testbeds include support for edge devices as virtual machines, even virtualisation does not provide perfect isolation, and attackers may compromise the underlying hypervisor. Even worse, as such devices are connected to the Internet, they may be recruited for denial of service attacks or used to share illicit materials.

Contributions. Addressing these challenges, we present `X-Lab`, an open-source[1] extension of the D-Cube testbed [16] enabling the setup of federated experiments spanning multiple D-Cube instances. With `X-Lab`, we provide the following contributions. First, we extend D-Cube with an overlay network to enable BRs to securely communicate between different testbed instances without the need for any special routing considerations on the user side. All the complexity of networking devices in different geographically-distributed sites across the Internet is hidden from the user presenting only a single layer-2 network domain. Second, we use `X-Lab` to establish a 4000 km cross-continent testbed between a site in Europe (Graz, Austria) and in Asia (Abu Dhabi, United Arab Emirates). Third, we employ this cross-continent setup to demonstrate and explore the impact of the Internet connection and necessary border routers on the performance of existing IoT protocols. The chosen protocols represent the two extremes of today's most used philosophies: carrier-sense multiple access (CSMA) utilizing only a single shared channel, which is often employed where power is not a primary concern along with the standardized routing protocol for low-power and lossy networks (RPL), and Open Synchronous Flooding (OSF), which instead relies on synchronous flooding [3] on multiple channels on a fixed periodic schedule – thus allowing a device to sleep for the remainder of the time. Our experiments highlight that delays induced by duty-cycled mesh protocols can account for up to 80% of the overall E2E delays – for example, due the unfortunate misalignment of the duty cycle schedule at different sites – and can thus significantly outweigh the delays introduced by the Internet link.

Paper outline. This paper proceeds as follows. We provide in § 2 a primer on D-Cube, the system on which `X-Lab` is based. We then present our approach to federating multiple testbed instances in § 3, detailing how we implement it in practice by extending D-Cube. We evaluate two protocols over a 4000 km cross-continent setup in § 4, shedding light on the factors affecting E2E performance. We finally provide an overview of related works in § 5 before concluding in § 6.

[1] `https://iti.tugraz.at/X-Lab`

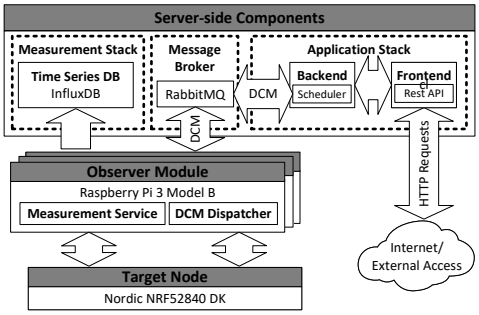


Figure 2: D-Cube architecture. Server-side components implement experiment logic, while observer modules execute the individual steps and collect the hardware-based measurements from target nodes.

2 D-Cube: A Primer

To enable E2E experimentation, we extend D-Cube: an experimental facility to benchmark low-power wireless systems [16, 15, 17] previously used to organize the EWSN dependability competition series [4]. D-Cube's architecture is shown in Fig. 2, which implements the individual steps required during an experiment as standalone components, e.g., the ability to program the low-power wireless device with the required firmware, control over the power and reset state of a device, or the collection of hardware-based measurements. The step-by-step logic required to perform an experiment is implemented in an easy-to-read script (part of the back-end's scheduler) communicating with the observer module via a common API. This API is called D-Cube messaging (DCM) and is based on Advanced Message Queuing Protocol (AMQP). Furthermore, D-Cube is split into two types of components: those running on the observer modules and those running on a central server. The former are the distributed devices forming the testbed infrastructure powering, programming, and instrumenting the low-power wireless system running the actual software to be tested. Each observer module is directly attached to one target node. The server-side components allow interaction with the outside world through a Web interface (front-end) and storage of the measurement data, as well as hosting the message broker used to relay DCM messages from and to the observer modules.

The individual steps required to perform an experiment are entirely controlled by the server-side components' scheduler, which connects to the message broker and issues commands such as downloading/flashing of the firmware, controlling the target node's power and reset state, or initializing the measurement process. A key advantage of this architecture is that the observer module *does not require any knowledge about the order of steps*, with the exception of the beginning and end of an experiment (which is used mostly to keep track of any process started during an experiment and to ensure a consistent state before and after an experiment).

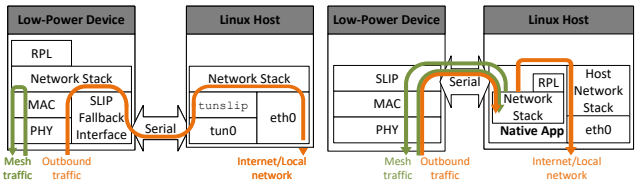
While each observer module only needs access to the server-side components, they do not need any open ports for incoming connections. As such, the underlying network architecture (including IP addresses of the observer modules) is completely hidden from the server-side components and an observer module may be on a completely different network.

3 Federating Multiple Testbed Instances

As outlined in § 2, D-Cube's observer modules connect to a message broker and await instructions without any knowledge of the order of steps to be performed for each experiment. Therefore, an observer module can be on a completely different network (even separated by the Internet) than the server-side components. We exploit this feature to connect all observer modules from the different testbeds to a single instance's message broker. On the server, a single script is in charge of running each individual experiment and is given a list of observer modules which participate (as a `json` file). Should an experiment be flagged as an *E2E* experiment, before running any commands on the nodes of the foreign testbed the script requests permission to do so from the server of that testbed instance. This means that *the observer module does not need to be aware if it is currently performing a local experiment or an E2E one*, reducing the complexity of federating experiments significantly. However, this puts the burden of coordinating the different testbeds entirely on the server-side components, and thus requires to change how experiments are scheduled.

Federated scheduler. Typically, D-Cube's scheduler only has to pick the next experiment from a list of candidates to be executed and then run the script performing the actual experiment. As the primary testbed instance (in Graz) is located in an office building, the scheduler also needs to consider the time of day – otherwise the interference caused by people in the office would negatively affect the repeatability of experiments. Further, the scheduler uses a round-robin algorithm to pick the next experiment to avoid a single researcher or group blocking the testbed for a longer duration. However, running an *E2E* experiment across *multiple testbed instances* means modifications to the scheduler are required. Namely, testbed instances now have to communicate among each other, which requires a notion of trust between instances. While a user may have an account on one site, the same might not hold true for the remaining ones, especially for private instances. To this end, we extended D-Cube with a public key which is shared with all other testbed instances that trust this specific instance and accept requests for E2E experiments. Every such request is then signed with the initiating instance's private key which can be validated with the shared public key.

Border router (BR). The observer modules in all instances are controlled from a single server, resulting in the ability to run a coordinated experiment across multiple – typically geographically distributed – sites. However, there is no way for the target nodes from one site to communicate with target nodes in another, or with any other cloud resources. As low-power wireless devices are typically not able to communicate with the Internet directly, so-called BRs are used to translate the (often compressed and fragmented) IP packets from and to an outside network. While *no singular standard for this exists*, these BRs often run entirely on a low-power device simply passing messages to an attached Linux device by serialising packets and passing them via UART or USB. Alternatively, other solutions allow the use of the low-power device only for the physical and media access layers with the rest of the protocol running on the Linux device. Fig. 3 shows the architecture



(a) embedded (b) native

Figure 3: Example of BR types: a) embedded device running most of the network stack and services such as RPL; b) the Linux host running a natively-compiled version of the embedded network stack and services along the existing host stack.

of these two approaches. The embedded solution (a) runs the entire network stack directly on the low-power device. While these devices have far fewer resources compared to the Linux host, any traffic from and to the low-power mesh network does not suffer any overhead from forwarding to/from that host. The native solution (b) delegates the time-critical MAC operations to the low-power device, but forwards the raw frames to the host for framing, fragmentation, and header compression. Either approach necessitates the ability to run code on the observer module. Since no singular standard exists, this code needs to be either provided by the user or maintained by the testbed operator. For the first release, we choose to support the SLIP protocol, which is available on operating systems such as Contiki-NG, RIOT, and Zephyr. Any experiment marked as E2E includes a list of nodes acting as BRs. These always start `tunslip`, which offers an IPv6 prefix via SLIP and then translates all packets from and to a special tunnel interface (typically `tun0`). The firmware on the low-power target devices is still in charge of handling these packets and the Linux-side code is independent of the OS and protocol running the on the target node.

Overlay network. With the integration of BRs into D-Cube's workflow and the coordinated setup and start of experiments, one could already start E2E experiments by exposing the BRs directly to the Internet. As the aim is to explore different solutions for heterogeneous IoT architectures that employ fundamentally different protocols with vastly different requirements, the setup must not impose any strict requirement beyond the use of IP. For MQTT, which uses TCP to route all data through a central message broker, this only requires a single device (which typically resides on a Linux host and must be reachable by all other devices). As a protocol optimized for the IoT, the Constraint Application Protocol (CoAP), which uses UDP, has several if not all devices acting as servers using a REST API to serve data. This necessitates incoming connections on multiple devices within the low-power mesh network. Simply exposing every single low-power device as well as the BRs directly to the Internet with a public IPv6 address would not only cause foreign traffic to affect the experiments, but also cause an unacceptable security risk to the testbed infrastructure. However, due to the vastly different requirements of protocols used for communication in IoT systems, reliance on network address translation (NAT) and users modifying their existing solutions to cope with the limitations imposed by such an approach is also not ideal. As such, we integrate a site-to-site VPN tunnel based on `wireguard` into the design of `X-Lab`, which further removes the need for all instances to

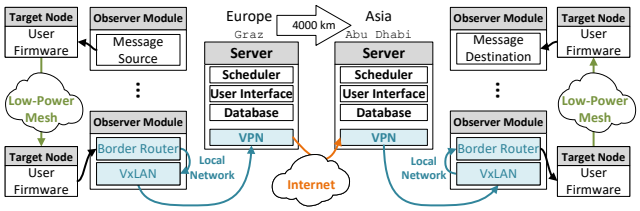


Figure 4: Two D-Cube instances extended via `X-Lab`. *Green* shows the low-power mesh. *Blue* shows the local network links. *Orange* shows the Internet site-to-site connection.

be directly reachable via a public IP. While this for the most part solves the issue of foreign traffic (some traffic from the testbed infrastructure may still be relayed to the low-power mesh), it does not solve the issue of requiring NAT on each BR. With the tunnel in place, however, we can create an isolated OSI layer 2 network between the BRs using `vxlan`. This allows encapsulation of arbitrary layer 2 frames over `wireguard's` layer 3 network, allowing the user to freely communicate directly between the BRs without the need for NAT and without any foreign traffic interfering with the experiment's traffic. Fig. 4 shows how two independent D-Cube instances located in two different continents enhanced with `X-Lab` can cooperate to execute an E2E experiment. The resulting end-to-end connection spans three types of links: low-power wireless mesh links connecting the target devices in each site with one another (green), links representing the different hops on the local area network (blue), and links across the Internet (orange).

4 Evaluation

We use `X-Lab` to evaluate the performance of different low-power wireless protocols running on a Nordic `nRF52840-DK` and establish two testbed instances in Graz (AT), and in Abu Dhabi (UAE), i.e., over 4000 km apart. Both versions are independent instances that can be used to conduct their own experiments. However, with the changes outlined in § 3, one instance can now be used to request temporary control over the nodes in the other testbed to conduct an E2E experiment.

Evaluated protocols. Based on the popular Contiki-NG operating system, we first evaluate a protocol based on RPL Lite (Contiki-NG's lightweight RPL implementation) and CSMA, as well as a second protocol, Open-SF (OSF) – a multi-PHY protocol based on synchronous transmissions [2].

RPL Lite. We evaluate CSMA+RPL with a minimal application that relays UDP messages from the source to the destination node via Contiki-NG's embedded `rpl-border-router`. This relies on serial line IP (SLIP) to pass messages from the low-power wireless mesh network to the edge device running Linux, which is connected to the other BRs via the isolated overlay network. While the goal is to show the performance of Contiki-NG out-of-the-box, due to the large and challenging setup of the Graz testbed instance, it is required to change some parameters to reliably form a network covering all devices. As such, this example increases the max link metric to 4096 (from 512) as well as the maximum number of retransmission of RPL's DAO messages to 10 (from 5).

OSF. While several synchronous flooding protocols have been open-sourced in the past [20], the latest version of OSF has the advantage of supporting IPv6, including support for SLIP-based BRs. This means that the same application logic that is used by the RPL example can be used to compare the impact of a duty-cycled but highly reliable protocol. The BR code is provided as a separate `null-border-router`, which removes RPL-specific code and uses the existing SLIP fallback interface to forward any packet not destined for the low-power wireless mesh network's prefix to the Linux part of the BR. Unless otherwise specified, we selected a period of 200 ms with NTA of 6, NTX of 12, and NSLOTS of 24[2] due to the aforementioned challenges of the Graz instance.

Table 1: Performance of the selected protocols in different setups. For the baseline performance in Graz, node 206 and 200 are source and destination, whereas node 51 and 54 are source and destination in Abu Dhabi, respectively. For the E2E experiments, node 206 and 54 are used (with 200 and 51 acting as BRs). Each experiment was repeated 10 times.

Setup	Protocol	Latency [ms]	Reliability [%]
Graz	CSMA+RPL	235.27 ± 17.13	98.13 ± 2.19
	OSF	147.00 ± 12.71	100.00 ± 0
Abu Dhabi	CSMA+RPL	40.23 ± 4.07	99.66 ± 1.10
	OSF	125.76 ± 23.24	98.97 ± 1.67
E2E	CSMA+RPL	337.70 ± 15.19	94.90 ± 2.76
	OSF	252.47 ± 6.22	99.84 ± 0.25

Testbed instances. The two testbed installations represent fundamentally different layouts, varying greatly in the number of nodes, diameter, and density.

Graz. This site has 48 nodes co-located with offices, with one side of the testbed representing a dense cluster of 20 nodes in a single room, while the remaining 28 nodes are spread across different offices.

Abu Dhabi. In this layout, 11 nodes are mounted around a $323m^2$ drone arena. Nodes are deployed along the walls of the arena, as well as mounted on the ceiling across two floors of adjacent offices. As such, the nodes are more uniformly distributed and their diameter is lower than in Graz (based on the chosen PHY, each node can be reached in one/two hops).

Baseline performance. First, we evaluate the performance of both mesh network instances individually. We only consider a single point-to-point connection within the testbed, i.e., a single source and destination.

Graz. We select a node (206) at the edge of the sparse section of the testbed to act as the source node and a geographically central node (200) as the destination. Tab. 1 (Graz) shows the mean and standard deviation for latency and reliability computed by D-Cube over 10 repetitions of a 10-minute experiment. Note that, despite using duty cycling, the flooding-based approach utilized by OSF exhibits a much lower latency with both protocols achieving a high degree of reliability.

Abu Dhabi. We again select a central location for the destination node (54) with a source node (51) at the edge of the testbed. However, as the testbed is much smaller than that in Graz, lower latency due to much fewer hops is to be expected. Tab. 1 (Abu Dhabi) shows the mean and standard deviation for the latency and reliability computed by D-Cube

[2]The OSF paper [2] provides more information on these terms.

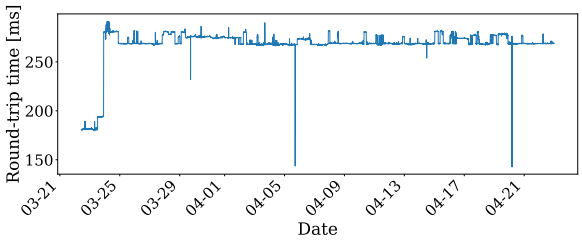
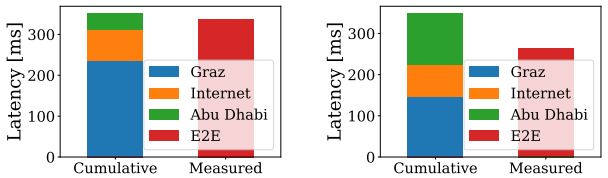


Figure 5: RTT (`ping`) across the VPN tunnel over 1 month.



(a) RPL (b) OSF

Figure 6: Comparison between the cumulative latency of the two instances plus the average Internet link latency versus the measured E2E latency for a) RPL and b) OSF.

over 10 repetitions of a 10-minute experiment. Compared to the results obtained in Graz, CSMA+RPL beats OSF in the Abu Dhabi site, despite the lower network diameter. OSF has in fact a lower latency per hop, but incurs a base latency for aperiodic traffic of half its period (100 ms).

Connection between the testbeds. To characterize the Internet link between the two D-Cube instances, we periodically run `iperf3` to determine the available bandwidth as well as `ping` to determine the round-trip time (RTT) between the two sites. Fig. 5 shows the RTT of the Internet link over the course of a month. The steep jump at the beginning of the measurements (March 23–24) is due to a significant change in Internet routing. Using `traceroute`, we noticed that – instead of going directly from Abu Dhabi to Graz (via France) – the packets were routed through the United States (based on the geolocation of the obtained IP addresses). Further, on several occasions, the measured RTT drops significantly (more than 100 ms on one occasion on April 5 and two more times on April 20) across an entire 60-packet measurement with no noticeable change in the variation between pings, only to return to its previous value in the next measurement. As Abu Dhabi employs ADSL (rather than a dedicated leased line) such variations are, however, not only expected but even (to a degree) desired. The average RTT between the sites over the course of a month was 267.35 ms (271.42 ms with the initial two days removed). As this is the average of the average RTT (each over 60 individual measurements), we will refer to the short-term standard deviation between the 60 measurements as jitter (which has a mean of 0.88 ms over the entire month). The long-term standard deviation of the measurements conducted every 10 minutes, however, is 19.20 ms (5.48 ms with the initial two days removed).

End-to-end performance. Subsequently, we can now evaluate the true E2E performance. To this end, the initial destination node 200 in Graz has been configured as BR. The new E2E benchmark suite supports multiple BRs, but as RPL-Lite no longer supports multiple DODAGs, each testbed only uses a single BR. To ensure that at least more than one hop is required, we choose a node (51) in the corner of the Abu Dhabi site as BR with a node in a second-floor office as destination. Tab. 1 (E2E) shows the E2E performance for both protocols. Unlike the baseline, these experiments are conducted over one hour, each with 10 repetitions. However, note that half the OSF results suffered from a synchronisation bug in the early version of the firmware and had to be excluded, as no network formed in one of the two instances. Importantly, these experiments were not conducted in a controlled environment but rather over the course of several hours with changes in the Internet link as well interference due to office occupancy.

More than the sum of its parts. Comparing these numbers reveals a pitfall that can occur when one considers each part of the system – the mesh networks and the Internet link – as independent. On one hand, for RPL+CSMA we observe a cumulative latency (neglecting the latency on the local Ethernet network and the BRs) of $235.26\,ms + 40.23\,ms + (267.35\,ms/2) = 409.17\,ms$, which is larger than the measured E2E-latency of 337.70 ms. For OSF, we would expect a latency of $147\,ms + 125.76\,ms + (267.35\,ms/2) = 406,435\,ms$, but observe an E2E latency of only 252.47 ms.

Obscured sources of latency. While investigating the source of this discrepancy, we took a closer look at the true Internet latency by running `tcpdump` on each of the VPN servers. With this, we can log each individual UDP packet traversing the VPN tunnel including timestamp and payload. Fig. 7 shows the RTT in blue compared to the latency computed from the PCAP file produced by `tcpdump`. While large changes in the Internet connection are for the most part reflected in both measurements, the final change in the PCAP at 2 am is not reflected in the round-trip time. With this measurement, we can improve our earlier estimate for RPL. The new mean value for the Internet of 75.79 ms (6.50 ms standard deviation) lowers the end-to-end estimate to 351.28 ms which reduces the discrepancy to well below the uncertainty of our measurements. Fig. 6 shows a comparison of the now corrected cumulative latency obtained by summing up the individual parts compared to the actually measured E2E latency. However, Fig. 6b shows that the same does not hold true for OSF, as the resulting cumulative latency of 348.55 ms is still almost 100 ms off despite the (in general) lower uncertainty observed in OSF. This is due to a property of protocols employing duty-cycling or other forms of periodically-scheduled communication, where the latency of packets can no longer be considered independent from one another. A packet that is delayed in Graz might be further delayed by an entire cycle on the Abu Dhabi side; however, in this case, while both testbeds are instructed to start at the same time, the Abu Dhabi site starts Δ_T ($\approx$ 70-85 ms) later, caused by the time it takes for the reset command to traverse the Internet link. Therefore, the OSF periodic schedule is also delayed by Δ_T. As experiment traffic from the Graz site to the Abu Dhabi site must also travel over this same Internet link (incurring the same time penalty Δ_T), it arrives at the Abu Dhabi site nicely aligned with the OSF duty cycle schedule at that location.

While these estimates now align well with our E2E measurements, they obscure an important source of latency: the BR. With the local setup, we have to receive and deliver one addi-

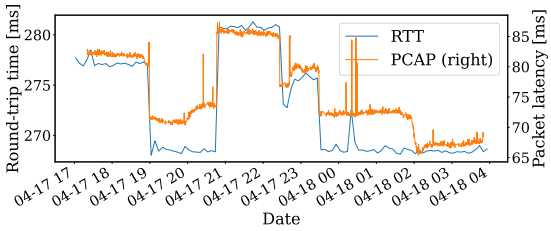


Figure 7: RTT (`ping`) of the Internet link vs. the per-packet latency obtained from a PCAP packet capture for a 1 h test.

tional message from and to D-Cube respectively. The latter employs an I2C mailbox, which introduces overhead compared to the SLIP interface. While the node will pass the packet within a few milliseconds to the SLIP interface, the HW of the Nordic `nRF52840-DK` adds significant latency to the time when the Linux side of the BR receives that packet. As a matter of fact, Contiki-NG's default UART baudrate (115200 baud) is not far from the default 100 kHz speed of the I2C peripheral. For SLIP, each UDP packet with 8 byte payload actually has a length of 70 bytes with the IP and UDP header decompressed (at least 2 bytes plus any overhead from the SLIP encoding which depends on the actual content of the IP packet). This means that the transmission of the SLIP packet takes upwards from 5 ms for a packet of only 8 bytes payload. To further investigate the impact of this overhead, we created a simplified test setup by running only the BR code on a single observer module (RPi 3B) with the node attached via the USB port used for programming and UART communication. For a 32 byte ping (84 bytes with IP and ICMP overhead) we measured an RTT from the RPi to the `nRF52840-DK` of $\approx$ 20 ms, of which only 12 ms (twice the 6 ms needed to transmit the packet via UART) can be accounted for. Using the secondary "native" USB port creates a UART interface on Linux, but does not emulate the baudrate and rather runs closer to the maximum 12 Mbps. This latency can be decreased to less than 1 ms. As such, the remaining 8 ms are lost due to overhead caused by the Segger JLink-OB acting as USB-to-UART converter or the `nRF52840-DK`'s UART peripheral driver. To verify that this is not due to a bug in Contiki-NG's SLIP or BR implementation, we test the same setup with an OpenThread BR (using `otbr` [3]), which uses its own Spinel protocol instead of SLIP: this results in a similar reduction by 20 ms when switching from UART through the Segger to the native USB implementation.

5 Related Work

`X-Lab` tackles three relevant topics for the IoT community: i) the ability to automatically run experiments on low-power wireless devices with the support for HW-based measurements [9], ii) the need for cooperation between independent testbeds, and iii) the ability to abstract underlying Internet links without altering their lossy/unpredictable nature, while enabling experimentation *without compromising security*.

Testbeds capable of measurements. When low-power wireless testbeds first emerged, they typically consisted of Linux-based embedded systems to which the target nodes were attached via USB for power and programming [10, 19]. Any metrics such as energy consumption, reliability, or latency, had to be estimated or derived from serial logs [5, 7]. Unfortunately, this often requires OS support and adds overhead affecting the measurement accuracy. Hence, testbeds started then supporting the collection of such measurements in HW [12, 16, 18]. Thanks to the accurate profiling of GPIO events, these testbeds allow measuring the performance of a solution without the need to modify the used device or OS. However, these testbeds only operate on a local scale, and while some [1, 14] do span larger areas including multiple cities, they operate mostly independently from one another without built-in support for E2E experimentation.

Federated testbeds. Federated testbeds [6, 8] often incorporate existing projects with different focuses (e.g., cellular, software-defined radios, or low-power wireless). As such, many existing testbeds for low-power devices [1, 10] have joined projects for federated testbeds: this, however, does not necessarily bring support for E2E experimentation. While such testbeds adhere to a common API, this does not mean that devices from one site can communicate with other sites. While the API often allows a user to reserve a "raw" Linux-based device to which a low-power node is attached to, it is then up to the experimenter to set up the required network and execute experiments by logging in via SSH and performing all steps manually. This requires knowledge of the internals of multiple testbeds' network, the setup of the BRs and low-power nodes, as well as the (manual) correlation of the different types of measurements collected by each individual testbed involved in the experiment.

Next-generation network research. Besides testbeds that utilize the Internet merely for connectivity between sites, there exist infrastructure projects [11, 13] to enable research into inter-domain networks that connect independent networks into an abstracted overlay network hiding the underlying topology – a field of research commonly referred to as software-defined networking. `X-Lab` builds its overlay network with the concepts and tools created by this community, but allows to extend the overlay through BRs into the domain of low-power wireless devices without the need to modify their network stack or OS.

6 Conclusions and Future work

In this paper, we have presented `X-Lab`, an extension to D-Cube that allows multiple testbed instances to cooperate and conduct E2E experiments. Further, we have used it to evaluate the impact of an unpredictable Internet connection on the E2E performance of an IoT solution consisting of multiple geographically-distributed low-power mesh networks, existing protocols, and BR solutions. We have made the entire software required for such experimentation available as open source, to enable and encourage others to test their solutions using a wide range of IoT architectures. In future work, we plan to extend `X-Lab` with support for Linux "target nodes" to allow the integration of cloud-based computing and storage resources solutions as well as edge devices.

Acknowledgements

This research work is supported by the SPiDR project funded by the Secure Systems Research Center (SSRC), Technology Innovation Institute (TII).

[3] `https://openthread.io/guides/border-router`

7 References

[1] C. Adjih et al. FIT IoT-LAB: A Large-Scale Open Experimental IoT Testbed. In *Proc. of the 2nd WF-IoT Forum*, 2015.

[2] M. Baddeley et al. OSF: An Open-Source Framework for Synchronous Flooding over Multiple PHYs. In *Proc. of the 18th EWSN Conference*, Oct. 2022.

[3] M. Baddeley et al. Understanding Concurrent Transmissions: The Impact of Carrier Frequency Offset and RF Interference on Physical Layer Performance. *ACM Transactions on Sensor Networks*, June 2023.

[4] C. A. Boano et al. EWSN Dependability Competition: Experiences and Lessons Learned. *IEEE Internet of Things Newsletter*, Mar. 2017.

[5] C. A. Boano et al. IoTBench: Towards a Benchmark for Low-power Wireless Networking. In *Proc. of the 1st CPSBench Workshop*, Apr. 2018.

[6] Crew project. Testbeds, 2010. `http://www.crew-project.eu`.

[7] A. Dunkels et al. Powertrace: Network-level Power Profiling for Low-Power Wireless Networks, 2011.

[8] Fed4FIRE+, Mar. 2017. `https://www.fed4fire.eu`.

[9] F. Foukalas et al. Dependable Wireless Industrial IoT Networks: Recent Advances and Open Challenges. In *Proc. of the 24th IEEE European Test Symp.*, May 2019.

[10] V. Handziski et al. TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks. In *Proc. of the 2nd REALMAN Workshop*, May 2006.

[11] J. Kwon et al. SCIONLab: A Next-Generation Internet Testbed. In *Proc. of the 28th ICNP Conference*, pages 1–12. IEEE, Oct. 2020.

[12] R. Lim et al. FlockLab: A Testbed for Distributed, Synchronized Tracing and Profiling of Wireless Embedded Systems. In *Proc. of the 12th IPSN Conference*, pages 153–166. IEEE, Apr. 2013.

[13] P. Lin et al. A West-East Bridge based SDN Inter-Domain Testbed. *IEEE Communications Magazine*, 53(2):190–197, Feb. 2015.

[14] L. Sanchez et al. SmartSantander: IoT Experimentation over a Smart City Testbed. *Computer Networks*, 61:217–238, Mar. 2014.

[15] M. Schuß et al. A Competition to Push the Dependability of Low-Power Wireless Protocols to the Edge. In *Proc. of the 14th EWSN Conference*, pages 54–65, Feb. 2017.

[16] M. Schuß et al. Moving Beyond Competitions: Extending D-Cube to Seamlessly Benchmark Low-Power Wireless Systems. In *Proc. of the 1st CPSBench Workshop*, Apr. 2018.

[17] M. Schuß et al. Making D-Cube an Open Low-Power Wireless Networking Benchmark. In *Proc. of the 17th EWSN Conference, poster session*, pages 164–165, Feb. 2020.

[18] R. Trüb et al. FlockLab 2: Multi-modal Testing and Validation for Wireless IoT. In *Proc. of the 3rd CPS-IoTBench Workshop*, May 2020.

[19] G. Werner-Allen et al. Motelab: A Wireless Sensor Network Testbed. In *Proc. of the 4th IPSN Conference*, pages 483–488. IEEE, Apr. 2005.

[20] M. Zimmerling et al. Synchronous Transmissions in Low-Power Wireless: A Survey of Communication Protocols and Network Services. *ACM CSUR*, Dec. 2020.