Experience: An Open Platform for Experimentation with Commercial Mobile Broadband Networks

Özgü Alay[1], Andra Lutu[1], Miguel Peón-Quirós[2], Vincenzo Mancuso[2], Thomas Hirsch[3], Kristian Evensen[3], Audun Hansen[3], Stefan Alfredsson[4], Jonas Karlsson[4], Anna Brunstrom[4], Ali Safari Khatouni[5], Marco Mellia[5], Marco Ajmone Marsan[2,5]

[1] Simula Research Laboratory, Norway [2] IMDEA Networks Institute, Spain
[3] Celerway Communications, Norway [4] Karlstad University, Sweden [5] Politecnico di Torino, Italy

ABSTRACT

Open experimentation with operational Mobile Broadband (MBB) networks in the wild is currently a fundamental requirement of the research community in its endeavor to address the need of innovative solutions for mobile communications. Even more, there is a strong need for objective data about stability and performance of MBB (e.g., 3G/4G) networks, and for tools that rigorously and scientifically assess their status. In this paper, we introduce the MONROE measurement platform: an open access and flexible hardware-based platform for measurements and custom experimentation on operational MBB networks. The MONROE platform enables accurate, realistic and meaningful assessment of the performance and reliability of 11 MBB networks in Europe. We report on our experience designing, implementing and testing the solution we propose for the platform. We detail the challenges we overcame while building and testing the MONROE testbed and argue our design and implementation choices accordingly. We describe and exemplify the capabilities of the platform and the wide variety of experiments that external users already perform using the system.

1 INTRODUCTION

Mobile broadband (MBB) networks have become the key infrastructure for people to stay connected everywhere they go and while on the move. Society's increased reliance on MBB networks motivates researchers and engineers to enhance the capabilities of mobile networks by designing new technologies to cater for a plethora of new applications and services, growth in traffic volume and a wide variety of user devices. In this dynamic ecosystem, there is a strong need for both open objective data about the performance and reliability of commercial operators, as well as open platforms for experimentation with operational MBB providers.

In this paper, we introduce ***MONROE**: the first open access hardware-based platform for independent, multihomed, large-scale experimentation in MBB heterogeneous environments.* MONROE comprises a large set of custom hardware devices, both mobile (e.g., via hardware operating aboard public transport vehicles) and stationary (e.g., volunteers hosting the equipment in their homes), all multihomed to three operators using commercial grade subscriptions.

Thorough systematic repeatable end-to-end measurements are essential for evaluating network performance, assessing the quality experienced by end users and experimenting with novel protocols. While existing experimental platforms, such as Planetlab [29], RIPE Atlas [31] or CAIDA Ark [3], meet these requirements, they are limited to fixed broadband networks and are not multihomed. MONROE is a one-of-a-kind platform that enables controlled experimentation with different commercial mobile carriers. It enables users to run custom experiments and to schedule experimental campaigns to collect data from operational MBB and WiFi networks, together with full context information (metadata). For example, MONROE can accommodate performance evaluation of different applications (e.g., web and video) over different networks or testing different protocols and solutions under the same conditions.

Objective performance data is essential for regulators to ensure transparency and the general quality level of the basic Internet access service [24]. Several regulators responded to this need with ongoing nationwide efforts [6]. Often, they do not open the solutions to the research community to allow for custom experimentation, nor do they grant free access to the measurement results and methodology. MONROE aims to fill this gap and offers free access to custom experimentation. The MONROE project selected 27 different external users to deploy their own custom experiments on the MONROE system with the purpose of testing and further improving the platform based on their feedback.

A common alternative to using controlled testbeds such as MONROE is to rely on end users and their devices to run tests by visiting a website [26] or running a special application [13]. The main advantage of such crowdsourcing techniques is scalability: it can collect millions of measurements from different regions, networks and user equipment types [10]. However, repeatability is challenging and one can only collect measurements at users' own will, with no possibility of either monitoring or controlling the measurement process. Mostly due to privacy reasons, crowd measurements do not always provide important context information (e.g., location, type of user equipment, type of subscription, and connection status (2G/3G/4G and WiFi)). MONROE is complementary to crowdsourcing approaches and the control over the measurement environment tackles the shortcomings of crowd data, though at the cost of a smaller geographical footprint [8]. Furthermore, MONROE supports the deployment of different applications and protocols, and enables benchmarking tools and methodologies.

MobiCom '17, October 16–20, 2017, Snowbird, UT, USA.
 ISBN 978-1-4503-4916-1/17/10…$15.00
DOI: https://doi.org/10.1145/3117811.3117812

In the rest of the paper, we report on our experience designing, implementing and using the platform. We detail the design considerations and demonstrate the versatility of our approach (Section 2). We explain how we cater for the requirements of experimenters and enable them to deploy myriad measurements on operational commercial MBB networks. The MONROE measurement node (hereinafter, the node or the MONROE node) sits in the center of the system and is the most important element, conditioning the proper functionality of the measurement system. We describe our experience with the MONROE system implementation and detail the hardware selection for the MONROE measurement node (Section 3). We forged the node to be flexible and powerful enough to run a wide range of measurement and experimental tasks, including demanding applications like adaptive video streaming. In the same time, we ensured that the node software design translates into a robust implementation (Section 4) that is also easily evolved and upgraded in order to sustain the most recent technological innovations. We further present the user access and scheduling solution we offer experimenters for exploiting the available resources of the platform in a fair manner (Section 5). Finally, we demonstrate that the MONROE system is a fitting solution to conduct a wide range of experiments over commercial cellular networks. To showcase its capabilities, we describe different categories of experiments MONROE supports (Section 6), which give an overview of the main categories of experiments MONROE users are conducting at the time of writing. Additionally, we expand on our experience interacting with the external users (Section 7).

2 SYSTEM DESIGN

Throughout the design process of MONROE, we interacted with the users of the platform (e.g., universities, research centers, industry and SMEs) and collected their feedback on requirements for platform functionality. This allowed us to gauge experimenters' expectations and use them to sketch the platform specifications.

2.1 Requirements

We summarize the main requirements as follows.

Large scale and Diversity: To give a representative view of the characteristics of an entire network, we need to collect measurements from a large number of vantage points. Furthermore, we should strive to collect measurements under diverse geographical settings, from major cities to remote islands.

Mobility: Mobility is what makes MBB networks unique compared to other wireless networks. To provide insight into the mobility dimension of MBB networks, it is imperative that the platform integrates a deployment under realistic mobility scenarios.

Fully programmable nodes: To accommodate the wide range of experiments users contemplate to run on the platform, we should forge measurement devices that are flexible, powerful and robust.

Multihoming support: To compare different mobile operators and/or different wireless technologies under the same conditions, the same node should connect to multiple providers at the same time (multihoming support). This further makes the platform particularly well suited for experimentation with methods that exploit aggregation of multiple connections.

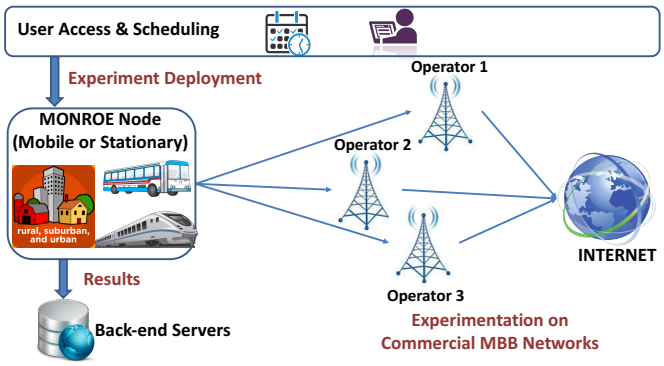

Figure 1: The MONROE platform: MONROE Nodes operate in trains, buses or inside homes and each connects to three commercial mobile operators in each country with MONROE presence. Users access the available resources and deploy their experiments via the User Access and Scheduling. Measurement results synchronize to external repositories operating in the back-end.

Rich context information: While analyzing the measurements, context information is crucial. The platform should monitor the network conditions, the time and location of the experiment, as well as the metadata from the modems, including, for example, cell ID, signal strength and connection mode.

Easy to use platform: It is crucial to make it easy for users to access the system and deploy experiments on all or a selected subset of nodes. This requires a user friendly interface together with a well managed and fair scheduling system.

2.2 Design Overview

We shaped the main building blocks of the MONROE platform such that we can meet the above-mentioned requirements. Note that while implementing different components of the platform, operational aspects also impacted the design choices, which we will discuss in detail in Sections 4-5. Next, we give an overview of the purpose and functionality of the main building blocks of the MONROE system, which we illustrate in Figure 1. All the software components of the MONROE system are open source [17].

MONROE Node: MONROE operates 150 nodes in 4 countries in Europe (Spain, Italy, Sweden and Norway). The measurement node resides at the core of our platform. Its design comprises two main notions, namely the hardware configuration, and the software ecosystem. In terms of hardware, each node has a main board that is a small programmable computer and supports (at least) 4 interfaces: three 3G/4G modems and one Wifi modem. To cover a diverse set of mobility scenarios, we customize a portion of the nodes (i.e., 95 out of 150 total nodes) to operate on public transport vehicles (buses and trains) and also in delivery trucks. In Section 3, we detail the choices for the node hardware implementation and our experience with running two node prototypes.

The node software is based on a Linux Debian "stretch" distribution to ensure compatibility with multiple hardware configurations and to enable a large set of experiments. Furthermore, especially considering the experimentation on protocols, Linux is the only operating system with sufficient hardware support for research and

implementation of transport protocols due to the accessibility of the source code, flexibility and community maintenance to ensure operability with other systems. On top of the operating system, the nodes run: (i) *the management software* that performs the normal jobs expected on any mobile device, (ii) *the maintenance software* that monitors the operational status of the nodes and diminishes manual maintenance intervention and (iii) *the experimentation enablers*, that enable experiment deployment (via the scheduler client) and feed rich context information to the experiments. To provide agile reconfiguration and access for the experimenter to different software components, the experiments run in the Docker lightweight virtualized environment. This also ensures containment of external actions in the node system. We periodically transfer the results of the experiments from the nodes to a remote repository. We further detail in Section 4 the node software ecosystem and present our evaluation of potential node internal performance overheads.

User access and scheduling: MONROE enables access to platform resources through a user-friendly web portal [19] that allows authenticated users to use the MONROE scheduler to deploy their experiments. The MONROE Scheduler facilitates exclusive access to the nodes (i.e., no two experiments run on the node at the same time) while ensuring fairness among users by accounting data quotas. We provide the details and the implementation choices for the user access and scheduling policies in Section 5.

3 HARDWARE IMPLEMENTATION

Given the requirements we drew from MONROE stakeholders (Section 2), the measurement device needs to be small, able to function in different environments (buses, trains, homes), affordable, robust, sufficiently powerful and should support the mainline Linux kernel. The size and price constraints limited us to evaluate different Single Board Computers (SBCs). There is a large amount of different SBCs available to the consumer public, with different CPU architectures and hardware configurations. However, most contain hardware requiring the use of proprietary drivers, thus restricting us to old kernels or making it impossible to compile custom kernels. We evaluated several options, including popular ones such as Raspberry Pi [30], Odroid [25], Beaglebone [1] and we selected PC Engines APU [28]. We chose the APU because it provides sufficient processing power, storage and memory for the foreseeable future at a reasonable cost. APUs integrate a 1Ghz 64 bit quad core processor, 4GB of RAM and a 16GB HDD. APUs have 3 miniPCI express slots, two of which support 3G/4G modems.

Modem Selection: To multihome to three mobile operators and a WiFi hotspot, we initially equipped the PC Engines APU board with an Yepkit self-powered USB hub [38], three USB-based CAT4 MF910 MiFis [42] and one WiFi card [4]. The reason we chose the MF910 MiFi is because, at the time we selected the hardware, it was the most modern device sold by operators we measured.

In the prototype validation phase, this implementation presented some major obstacles. While the APUs proved to be very stable, the MiFis proved more challenging than expected. First of all, in the last quarter of 2016, the MiFis' vendor issued a forced update to the firmware. The update was applied despite the fact that we took special care to configure the devices not to receive automatic updates. As a result of the forced update, all our MiFis became inaccessible for the MONROE system. Furthermore, the MiFis themselves were prone to resets or to enter a working state (transparent PPP) from which we could only restore them to normal operation by draining their batteries, or performing a manual reboot by pushing the power button. Finally, after 6 months of operation, some of the MiFis showed clear signs of swollen batteries. This problem brought serious safety concerns for the nodes operating in places other than our own (controlled) premises (e.g., public transport vehicles). We thus modified the hardware configuration to use internal modems operating in the miniPCIe slots of the APU board.

Current Node Configuration: We decided to increase the control over the MONROE node and base its implementation on a dual-APU system. One of the two APUs in each node has two MC7455 miniPCI express (USB 3.0) modems [33], while the other has one MC7455 modem and a WiFi card. We chose Sierra Wireless MC7455 as our 4G modem since, at the time of the upgrade, it was supporting the most recent category (CAT6) an industrial grade modem could provide. This design eliminates the risk brought on by the use of batteries, avoids any forced updates (the new modems are not routers), simplifies resets (no draining of battery) and increases our overall control over the system.

Takeaways: APUs showed very stable performance, while repurposing the MiFis to behave as simple modems presented major challenges (e.g., forced updates and swollen battery problems). We thus bring forward a more compact and robust node configuration that relies on internal modems operating in miniPCIe slots. This also simplifies the node since we avoid potential NAT and routing issues the MiFis might trigger.

4 NODE SOFTWARE IMPLEMENTATION

In this section, we describe in detail the node software ecosystem and present the justification for our implementation choices.

4.1 Software Ecosystem

Figure 2 presents the elements that coexist in the MONROE node software ecosystem, namely the node management software, the node maintenance software and the experimentation enablers.

The **node management software** integrates a set of core components that run continuously in the background. They perform low-level work in line with the normal jobs expected on any mobile device or computer. These include (i) a *Device Listener*, which detects, configures and connects network devices, (ii) a *Routing Daemon*, which acquires an IP address through DHCP, sets up routing tables, (iii) a *Network Monitor*, which monitors interface state, checks the connectivity of the different interfaces and configures default routes. The node operates behind a firewall, which we configure with strict rules to increase node security.

The **node maintenance software** integrates components that monitor the node status and trigger actions to repair or reinstall when malfunctioning. A *system-wide watchdog* ensures that all core components (node management) are running. However, during the first few months, we experienced loss of connection to nodes because of problems that watchdogs could not tackle, such as file system corruptions which can occur due to frequent sudden power loss in mobile nodes. Thus, we defined and implemented a robust

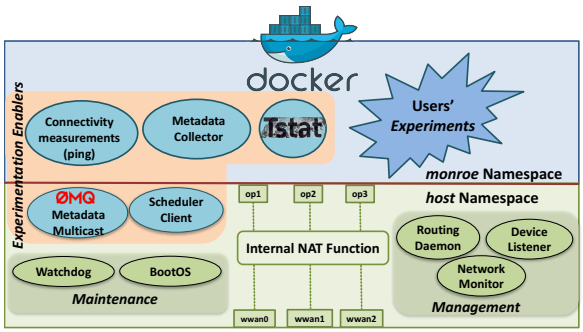

Figure 2: Node Software Ecosystem.

node recovery method, called BootOS, that enables a hard restart of the node (i.e., a reinstallation of the operating system to a known working baseline). This method allows us to recover both from file system errors that prevent system boot-ups, and software configurations that may lead to loss of connectivity. To achieve this goal, we trigger a two-stage boot loader process at node start-up. In the first stage, we start the BootOS, which resides entirely in RAM and only uses read-only hard-drive access for its normal operation. The BootOS verifies that the filesystem of the APU is not corrupt, and that no forced reinstallation has been requested. It then proceeds to boot the MainOS, which contains the MONROE system software. If the filesystem is corrupt, or in case of a forced reinstallation, the BootOS reinstalls an image of a known working installation.

The **experimentation enablers** include the scheduling client, the default experiments, and the services for external experiments. Within the node software ecosystem, we differentiate between the user experiments and the management and maintenance software by configuring a separate *monroe network namespace* where experiments run. This increases our control over the ecosystem and limits the impact external users can have on the node. This separation further allows us to account (as part of the scheduling system) the traffic volume each user consumes. We require that each experiment runs inside a virtualized environment (Docker container) to ensure separation and containment of processes. The *Scheduling Client* communicates with the Scheduler to enable experiment deployment per user request. It periodically checks for new experiment containers to run in the node and deploys them in advance to their scheduled execution time. Section 5 offers more details on the scheduling system. The *metadata broadcasting service* runs continuously in the background and relays metadata through ZeroMQ [41] in JSON [12] format to experiment containers. The nodes periodically run connectivity measurements (e.g., ping), and this together with metadata allow us to monitor the node's state and the overall health of the platform. Furthermore, the *Tstat* [9] passive probe provides insights on the traffic patterns at both the network and the transport levels, offering additional information on the traffic each interface exchanged during an experiment.

Takeaways: Containment of users activity in the node is paramount to avoid security risks, node malfunctioning events, unreliable results and, more severely, node loss. We prevent foreign unauthorized access to the node with a strict firewall. Then, continuous monitoring of the platform is crucial and we enable it by implementing monitoring functions in the node management software. Node maintenance is expensive, so it is important to forge the node as a self-healing system. We implement this functionality in the node maintenance software that takes automatic actions when the node malfunctions.

4.2 Experiment Containment

Docker Virtualization. The node design we propose mandates that MONROE users execute their experiments inside Docker containers, which provide isolation from the host node. This is true both for default monitoring measurements and external users experiments. Docker containers are based on a layered file system, where a container can reuse layers shared with other containers.

MONROE provides the default base image for the experiment containers, which integrates the base operating system installation with default tools that are potentially useful for many experiments. The lightweight containers provide just the contents that are unique for the particular experiment, significantly reducing the download and deployment time overhead and accountable traffic volume. Experiments running inside a container have access to the experimental network interfaces. They can read and write on their own file system, overlaid over that of the base MONROE image. Finally, there are specific paths (e.g., `/MONROE/results/`) where the experiments can write their results and that the node automatically transfers to the MONROE servers. Our public software repositories contain all the files necessary to build new user experiments, as well as experiment templates and examples.

Internal NAT Function. To ensure the minimum impact of user experiments gone wrong, we define the *monroe network namespace* where experiment containers run. For each physical interface that the network-listener detects as available, we create a virtualized ethernet, `veth`, interface pair, and move one end to the monroe namespace. We then add routing rules in the network namespace to allow routing by interface. In order to allow the network devices in the host namespace to communicate with the ones in the monroe network namespace, we define an internal Network Address Translation (NAT) function. We use `iptables` NAT masquerading rules in the host namespace to configure the NAT function. Finally, we add the corresponding routing rules to map each `veth` interface to the correct physical interface.

Overhead Quantification. The internal network design introduces two potential overheads that might impact performance measurements: (i) the internal NAT function that connects the network devices in the host namespace with their corresponding duplicates in the monroe namespace, and (ii) the Docker containers we use to separate the processes that correspond to a certain experiment that runs inside the container. Thus, prior to detailing the measurement results of different commercial MBB operators, we focus here on these two design overheads and aim to quantify their impact (if any) on performance measurement results. More specifically, we quantify the delay overhead by running ICMP ping measurements, and the impact on throughput by running HTTP downloads.

To instrument our system benchmarking measurements we use a single APU node running the Debian "stretch" MONROE image with a local Fast Ethernet link. Using a local link allows us to minimize the impact of the network on our measurements, and focus on the impact of the system overheads. We run http download measurements with `curl` and ICMP ping measurements with `fping`

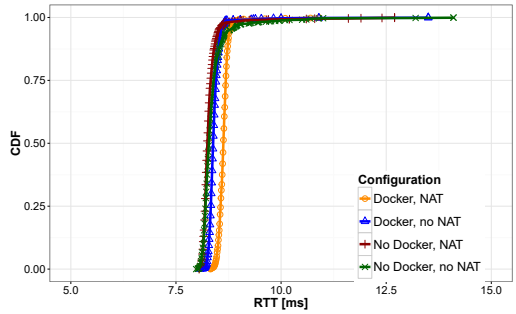

Figure 3: CDFs of ICMP RTTs [ms] measured against 8.8.8.8 per testing configuration over Fast Ethernet link.

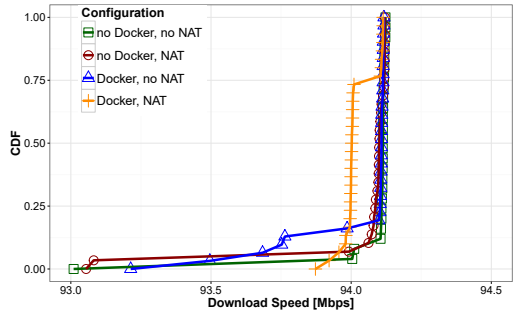

Figure 4: CDFs of Downloads Speed [Mbps] measured per testing configuration over Fast Ethernet link.

to quantify the impact of the internal NAT function and of the Docker virtualization. We focus on four configurations for our testing setup, namely: no NAT and no Docker (experiments run in host namespace), no NAT but Docker (experiments run inside a Docker container in the host namespace), internal NAT and no Docker (experiments run in the monroe namespace) and internal NAT and Docker (experiments run inside a Docker container in the monroe namespace).

To quantify the delay overhead, we collect 1,000 RTT samples against the Google DNS server 8.8.8.8 on the Ethernet connection on all four configurations. Figure 3 shows the results of the measurements. We conclude that the overhead of the NAT function internal to the node is insignificant. In average, we see a penalty in the order of 0.1ms, (i.e., in the range of clock granularity in Linux systems). We note that the Docker and NAT combination introduces a slight delay, which is not overwhelming.

For the throughput measurements, we download 1GB of data from a server configured in the local network. We collect 30 samples for each testing configuration. In Figure 4 we show the cumulative distribution of download speed per namespace and operator, for each of the different targets. We find that there is a 1% performance penalty that using the internal NAT function and the Docker virtualization introduces in average. We report no direct impact of using the Docker containers, which we expected, since the purpose of the Docker virtualization is purely for experiment containment.

Takeaways: Our priority in the node software implementation phase is keeping the nodes within normal functioning parameters for as long as possible and limiting direct maintenance intervention, while allowing external users to run a wide range of complex measurements with minimum interference. To achieve this, we separate the network namespace where users can run their experiments from the host namespace, where the monitoring and management software runs. This introduces two potential overheads in the system, which we quantify and show to have little or no impact.

5 USER ACCESS AND SCHEDULING

We provide access to the MONROE platform through a user-friendly interface consisting of an AngularJS-based web portal [19]. As part of the MONROE federation with the Fed4FIRE [7] initiative, the user access follows the Fed4FIRE specifications in terms of authentication and resource provisioning. Through the portal, experimenters interact with the scheduler and deploy their experiments without accessing directly the nodes. The scheduler API is accessible to enable experiment deployment automation. The scheduler prevents conflicts between experiments (i.e., only one user can run an experiment on a certain node at a time) and assigns resources to each user based on their requirements and resource availability.

Given the challenging scenarios we aim to cover in our testbed, nodes in MONROE have potentially unreliable connectivity and low bandwidth. This is the norm for node in buses, trains and trucks, which follow the schedule of the host vehicle. Experiment scheduling therefore accounts for two factors: (i) the node may not have connectivity at the time of the experiment and (ii) a high lead time when deploying containers means that experiments should be deployed early. Furthermore, experimenters may require to run synchronous measurements on multiple nodes. The common approach to task scheduling and decentralized computing, which deploys jobs to registered nodes based on their availability, struggles with these constraints. Therefore, for the MONROE scheduler, we follow a calendar-based approach, assigning time slots to experiments. Deployment of experiments takes place up to 24 hours in advance, as soon as the node retrieves information about the assigned task. This allows both immediate scheduling on nodes that are not otherwise occupied, and scheduling synchronous experiments on low availability nodes well in advance. It also allows synchronizing experiment runtime with vehicle schedules when available.

In addition to managing the time resource, the scheduler handles data quotas assigned by the contracts with the MBB operators. We assign each experimenter a fix data quota. In addition, we may assign users a quota on computing time (i.e., maximum time the users can run experiments on the node). We designed the quota system to provide fair usage of the available resources. An important factor to ensure fairness in day-to-day usage, is that a certain data quota is reserved by the experimenter in advance, and subtracted from the user quota for the duration of the experiment. Experimenters may subsequently refund the remaining quota. Hence, it is not possible to block large quantities of resources without having been assigned the necessary budget, even if the resources are not actually used.

From March 2016 until March 2017, the MONROE scheduler has been actively used by 30 users. A total of 75, 002 experiments have successfully ran on the platform, while 7, 972 scheduled experiments failed. There are many different reasons for failed experiments, for example that the container exits unexpectedly or the data quota is exceeded. Note that these failures are expected especially for the new users that are trying to familiarize themselves with the platform. We are running an open conversation with our users,

gathering feedback from them and updating the user access and scheduling policies accordingly.

Takeaways: Resource allocation and experiment scheduling on MONROE is challenging because nodes have potentially unreliable connectivity (e.g., nodes in mobility scenarios) and limited data quota due to commercial-grade subscriptions. A calendar-based approach for scheduling addresses these requirements by taking into account per user and per node data quota, and synchronized experiment start time.

6 OPEN EXPERIMENTATION

Starting from the platform design phase, we have been working together with our stakeholders to understand their requirements from the MONROE system and which experiments have the highest appeal (Section 2). We then took this process further and, throughout 2016 and 2017, the MONROE consortium organized two open calls for external experimenters. After a thorough selection process based on peer-review, we funded 27 different projects (12 projects in May 2016 from the first open call (OC1) and 15 projects in February 2017 from the second open call (OC2)) to be among the first users of the MONROE system.

The experiments that these projects proposed are very diverse and cover a wide range of scenarios, from simple network performance measurements to innovative protocols evaluation to application performance assessment. All experimenters were encouraged to propose SW extensions to the platform (e.g., measurement packages that can be offered to the MONROE community) as well as HW extensions to the infrastructure (e.g., deploying MONROE nodes in locations with no previous MONROE coverage, or increasing the density of MONROE nodes in locations with MONROE coverage).

6.1 MONROE Experiments

We report next on our experience accommodating the 12 OC1 measurement campaigns on the platform[1], together with the base experiments deployed by the consortium. We are currently offering to the community a series of experiments [18], which any external users can deploy on their own. This goes toward achieving our goal of shaping MONROE into an Experimentation as a Service (EaaS) platform. We group all these experiments in three main categories: Mobile Broadband Performance, Service Oriented QoE and Innovative Protocols and Services. These categories also fit to the range of measurements that our users are currently curating and have been already actively deploying. The distribution of experiment runs on the MONROE platform to the time of writing among these categories is: Mobile Broadband Performance (19%), Service Oriented QoE (36%) and Innovative Protocols and Services (45%). The volume of data that experiments in different categories consume varies, with Service Oriented QoE taking the largest quota (60%), while Innovative Protocols and Services are the least demanding (10%), despite registering the largest number of experiment runs. We further detail each category and provide examples of experiments and analysis one can perform using MONROE.

[1]We mention that, at the time of writing, only the 12 projects from OC1 are actively using the MONROE platform. Though already approved, the additional 15 projects from OC2 have not started actively using the MONROE system.

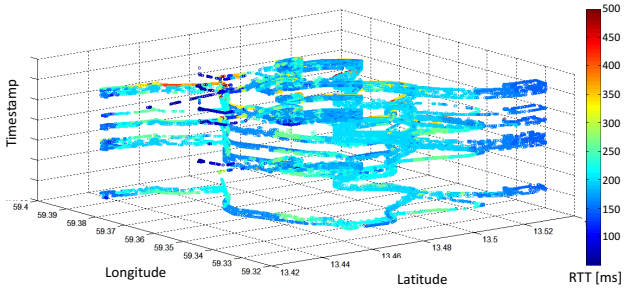

Figure 5: 3D graph average RTT for an operator in Sweden. Multiple laps are shown using the Y-axis offset based on relative timestamps to visually show the different trips.

6.1.1 Mobile Broadband Performance. To measure a mobile network in a reliable and fair way, it is important to identify the metrics that accurately capture its performance. Different stakeholders have different metrics of interest and we argue that MONROE is able to cater all of them. For example, regulators need connectivity, coverage and speed information to monitor whether operators meet their advertised services. Operators are interested in spatio-temporal data reporting the operational connectivity information to further identify instability and anomalies.

One important feature of the MONROE platform is that its deployment in public transportation vehicles allows users to evaluate MBB performance in diverse and complex urban mobility environments. A unique characteristic of this deployment is the repeatability of measurements obtained by many runs on the same route, at different hours. For example, Figure 5 shows RTT (ICMP ping) measurements for an operator in Sweden, as measured by the node operating aboard the same bus during several working days. In the figure, dot colors encode the range of values for the measured RTTs and we observe variations in RTT among different trips through the same location. Repeated measurements provide high confidence and diminish noise in the data, whereas measurement samples at the same location but at different hours allow for the analysis on the time-of-the-day effect (e.g., rush hour versus normal hours).

6.1.2 Service Oriented Quality of Experience. An important measurement dimension to explore comes from the great interest in how users perceive individual services and applications over different terminals (e.g., mobile phones, tablets, and computers). The recent proliferation of user-centric measurement tools (such as Netalyzr [13]) to complement available network centric measurements validates the increasing interest in integrating the end user layer in network performance optimization. MONROE enables experimentation with essential services and applications, including video streaming, web browsing, real-time voice and video, and file transfer services. The service oriented measurements give a good basis for investigating the mapping from Quality of Service to Quality of Experience. With such a mapping, operators can gain better understanding of how their customers perceive the services delivered by their network. From the end users and service providers perspective, they could acquire more knowledge of the performance over different MBBs and then choose the network that delivers the best quality for services that are of interest to them. Furthermore, application developers (e.g., Youtube, Netflix and Spotify) heavily

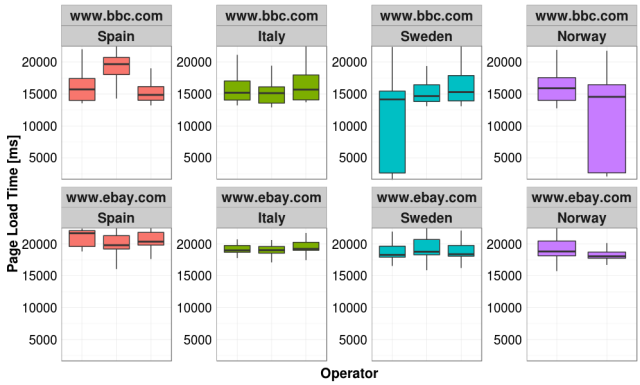

Figure 6: Page download time for 11 operators using data from 37 MONROE nodes in Spain, Italy, Sweden and Norway while fetching `www.bbc.com` and `www.ebay.com`; each subplot corresponds to a country-target pair, and each boxplot to a unique operator in each country.

rely on the underlying network characteristics while optimizing their services for the best user's experience.

To showcase the capabilities of the platform, Figure 6 reports the page load time (PLT) measured in the MONROE platform using a headless [21] browser to fetch two popular websites (www.bbc.com and www.ebay.com) from 37 nodes in four countries with MONROE coverage. If we focus on the PLT as an objective indicator for the quality of experience and track it in comparison with metadata information, we further enable the analysis of the mapping between QoS metrics to the end-user experience.

6.1.3 Innovative Protocols and Services. Another significant use case for MONROE is investigating the impact of middleboxes in the current Internet ecosystem. These range from address and port translators (NATs) to security devices to performance enhancing TCP proxies. Middleboxes are known to introduce a series of issues and hinder the evolution of protocols such as TCP. Since middleboxes are ubiquitous in MBB networks [34, 36, 37], in collaboration with the H2020 MAMI project [16] we aim to observe and characterize middlebox operations in the context of real-world MBB deployments. MONROE further enables assessment of existing protocols, paving the way for protocol innovation.

As an example in this category, we investigated whether the operators measured with MONROE use Performance Enhancement Proxy (PEP) [2] to improve end-users' quality of experience. These proxies provide higher performance and faster error recovery [5, 11, 23]. We ran throughput tests (http downloads) from all the MONROE nodes on different ports against the same responder, where we also run an instance of Tstat on server. We then cross-compared the Tstat analysis logs on the nodes (client-side) with the server-side logs to examine if the proxy splits the TCP connection. Table 1 shows the global view of operators. Yes, Yes*, and No in the table mean always, sometimes, or never. The third column indicates the usage of NAT in the operator network. For instance, op2 in Italy always uses NAT, and sometimes connections are routed through a PEP. On the contrary, op1 sometimes assigns public IP addresses, but HTTP traffic always goes through a PEP device. The forth column indicates if the performance seen on the client and server side are different. A mismatch hints for the presence of a PEP. The fifth column reports the number of public IPs seen in server-side traces (i.e., the "size" of the PEP boxes). The last column shows if the PEP changes the TCP headers (e.g., removing/adding/changing options), and on which ports. Overall, the picture varies with different PEP configurations for different operators.

Table 1: The summary of the operators and their settings.

	OP	NAT	PEP	# IP	L4 mangling
IT	op0	No	Yes	262	80
	op1	Yes*	Yes	129	80,443,8080
	op2	Yes	Yes*	1484	No
ES	op0	Yes	No	272	No
	op1	Yes	No	244	No
	op2	No	Yes	-	80
SE	op0	Yes*	Yes*	1652	No
	op1	No	Yes	3486	No
	op2	No	Yes*	4679	No
NO	op0	No	Yes*	472	No
	op1	Yes*	Yes*	46	No

6.2 External Projects Overview

The OC1 experiments focus mainly on Service Oriented Quality of Experience and assessing the QoE for popular applications, such as interactive video-conferencing (e.g., webRTC) or popular video-on-demand (VoD) application (e.g., YouTube) across multiple mobile carriers. There are 5 out of 12 such projects currently funded from OC1. Their number decreased to only one among the projects funded in OC2. A notable example in this category is a project that integrated the YoMoApp tool [35] with MONROE to monitor the QoE for YouTube. The resulting Yomo-Monroe [32] has two components: Yomo-docker and Yomo-browser-plugin. *Yomo-docker* [40] container provides features to estimate YouTube's "Quality of Experience." For this purpose, the docker container independently performs experiments and monitors the quality at the end-user side. *Yomo-browser-plugin* [39] monitors the quality of YouTube video streaming in the browser. This experiment is a software extension of MONROE and will be available as MONROE EaaS.

Three OC1 projects leverage MONROE data (from base experiments and metadata collection) to build and run anomaly detection algorithms or LTE performance benchmarking across multiple carriers. One particular OC1 project focuses on emergency communications and tests different protocol innovations. The project customized MONROE nodes with mobile, Ethernet connections, fixed point-to-point wireless links, low-band radio interfaces and satellite broadband – all classes of links that are used routinely in disaster situations to provide communications. The project proposes an application use-case for emergency communications in disaster situations. This is based on IP multi-homed support that enables resilient differentiated services, and allows an application to select the best available transport path. The users produced PATHspider-monroe [14, 27], a version of the PATHspider tool [15] adapted for running on MONROE nodes. PATHspider is a tool for A/B testing of path transparency to certain features in the Internet. We note that this experiment is also available as MONROE software extension and will be available as MONROE EaaS.

In OC2, the projects funded shifted mainly towards cloud and smart cities experimentation, with 8 out of 15 proposals active in these topics (e.g., smart city security monitoring with MONROE,

analysis of latency-critical connected vehicle applications). The OC2 external users have strong proposals for understanding the synergies between mobile carriers and popular cloud service providers (e.g., characterizing mobile content providers in the wild or tackling net neutrality in MBB networks). A couple of OC2 users aim to use the MONROE platform and the data we produce to device machine learning algorithms for informing self-organizing networks (SON).

7 EXTERNAL USERS EXPERIENCE

In this section, we present a summary of our interaction with the external users while supporting them in the process of using the platform for experimentation. The OC1 experimenters had exclusive access to the prototype platform. Thus, we relied upon their feedback to refine the platform to its current version. We collected this feedback in the form of a report the users provided, which integrated a grading system for different components of the platform. The experimenters we selected through OC2 have access to a more mature version of the MONROE platform, which we aim to further improve pending their additional feedback.

7.1 External Users Feedback

Users reported that the documentation we provided in the MONROE User Manual [20] is very useful, receiving a score of 4.5/5 and that the MONROE experiment templates and examples are easy to reuse (4.2/5). The virtualization method based on Docker containers was very well received, with a mark of 5/5. However, our users strongly suggested to create more multimedia material to complement the written user manual, particularly showing the complete life cycle of an experiment from container creation to scheduling and retrieval of results. Access to metadata was also seen as easy and useful (4.5/5).

As expected when accessing a prototype platform, our users saw some issues at the beginning, giving a mark for easiness and usability of 3.6/5. Regarding scheduling of experiments, in general our users had some troubles understanding all the details of the scheduling process, pointing towards the need for more step-by-step instructions such as the requested multimedia material. Some topics that were particularly troubling for the platform users were the binding to specific interfaces in a multihomed platform and the optimization of the size of the Docker containers.

We provided important additions based on users feedback. First, by opening the scheduler API for command-line tools, we enable submission of experiments programmatically (indeed, a tool for that purpose was released openly by one of the users). Second, we enable SSH access to containers in testing nodes for debugging purposes. The users reported that debugging in batch mode was otherwise very complex and tedious since every debug run had to wait for scheduling, execution and retrieval of results. Finally, the user access and scheduler system now supports rescheduling of the past experiments on the same or different nodes.

7.2 Lessons Learned

To achieve the goals of the external users, we learned that the large availability of experimental resources in MONROE is mandatory, while still giving experimenters strong control of the testing environment. For this reason, each user had access to a series of development nodes (for building the measurement tools in the lab) and testing nodes (for testing the measurement tools in a limited portion of the actual platform). We further decided to guarantee our external users exclusive usage of reserved resources.

Apart from offering experimenters a system ready to accommodate their measurements, we also encouraged external users to propose hardware extensions and enhancements to the platform. This allows us to grow the platform, increase its geographical footprint and engage with the stakeholders to create a relevant product. Fostering a community around the MONROE platform also means producing a rich variety of software measurement tools. Apart from the experiments we maintain withing the consortium, we also encouraged external users to bring software extensions to the platform and contribute to the MONROE EaaS initiative. Numerous of our external users responded positively to this initiative and integrated new measurements software packages with MONROE, which they offer openly to the community. Finally, the MONROE platform aims to be complementary to other measurement infrastructure. Thus, it has been an important goal for us to be able to deploy measurement tools that also run on other hardware-based platforms or in crowd-sourcing platforms. Furthermore, we collaborate with other publicly funded projects in need of mobile measurement infrastructure. An example for this is the Horizon 2020 project NEAT [22] that is planning to use MONROE to evaluate their software and API for optimized transport protocol and network selection.

8 CONCLUSIONS

In this paper, we reported on our experience designing an open large-scale measurement platform for experimentation with commercial MBB networks. MONROE is a completely open system allowing authenticated users to deploy their own custom experiments and conduct their research in the wild. The platform is crucial to understand, validate and ultimately improve how current operational MBB networks perform towards providing guidelines to the design of future 5G architectures. We described our experience with the MONROE system implementation and detailed the hardware selection for the MONROE measurement node, its software ecosystem and the user access and scheduling solution. We emphasized the versatility of the design we propose, both for the overall platform and, more specifically, for the measurement nodes. In fact, the node software design is compatible with a number of different hardware implementations, given that it can run on any Linux-compatible multihomed system. Our current hardware solution is the most fitting for the set of requirements and the predicted usage of MONROE, which we evaluated based on our discussions and interaction with the platform's users.

ACKNOWLEDGMENTS

This work is funded by the EU H2020 research and innovation programme under grant agreement No. 644399 (MONROE), and by the Norwegian Research Council RFF project No. 245698 (NIMBUS). For more information, visit https://www.monroe-project.eu/. The authors would like to express their gratitude to the reviewers and, particularly, to Aruna Balasubramanian, for their invaluable advices to improve this work.

REFERENCES

[1] Beagleboard.org. http://beagleboard.org.
[2] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. 2001. Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations. (2001).
[3] CAIDA. Archipelago (Ark) Measurement Infrastructure. http://www.caida.org/projects/ark/
[4] Compex. WLE600VX. http://www.pcengines.ch/wle600vx.htm
[5] Viktor Farkas, Balázs Héder, and Szabolcs Nováczki. 2012. A Split Connection TCP Proxy in LTE Networks. In *Information and Communication Technologies*. Springer, 263–274.
[6] FCC. 2013. *2013 Measuring Broadband America February Report*. Technical Report. FCC's Office of Engineering and Technology and Consumer and Governmental Affairs Bureau.
[7] FED4FIRE. http://www.fed4fire.eu/.
[8] Mah-Rukh Fida, Andra Lutu, Mahesh Marina, and Ozgu Alay. 2017. ZipWeave: Towards Efficient and Reliable Measurement based Mobile Coverage Maps. *Proc. IEEE INFOCOM* (May 2017).
[9] A. Finamore, M. Mellia, M. Meo, M. M. Munafo, P. D. Torino, and D. Rossi. 2011. Experiences of Internet traffic monitoring with tstat. *IEEE Network* 25, 3 (May 2011), 8–14. DOI: https://doi.org/10.1109/MNET.2011.5772055
[10] Matthias Hirth, Tobias Hobfeld, Marco Mellia, Christian Schwartz, and Frank Lehrieder. 2015. Crowdsourced network measurements: Benefits and best practices. *Computer Networks* 90 (2015), 85–98.
[11] M. Ivanovich, P. W. Bickerdike, and J. C. Li. 2008. On TCP performance enhancing proxies in a wireless environment. *IEEE Communications Magazine* 46, 9 (September 2008), 76–83. DOI: https://doi.org/10.1109/MCOM.2008.4623710
[12] JSON. http://www.json.org/.
[13] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. 2010. Netalyzr: illuminating the edge network. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 246–259.
[14] Iain R. Learmonth, Andra Lutu, Gorry Fairhurst, David Ros, and Özgü Alay. 2017. Path Transparency Measurements from the Mobile Edge with PATHspider. In *Proc. of the IEEE/IFIP Workshop on Mobile Network Measurement*.
[15] Iain R. Learmonth, Brian Trammell, Mirja Kühlewind, and Gorry Fairhurst. 2016. PATHspider: A tool for active measurement of path transparency. In *Proceedings of the 2016 Applied Networking Research Workshop*. 62–64.
[16] H2020 MAMI Project. Measurement and Architecture for a Middleboxed Internet. https://mami-project.eu/
[17] MONROE. Open Source Code. https://github.com/MONROE-PROJECT
[18] MONROE. Open-Source Experiments. https://github.com/MONROE-PROJECT/Experiments
[19] MONROE. User Access Portal. https://www.monroe-system.eu
[20] MONROE. User Manual. https://github.com/MONROE-PROJECT/UserManual
[21] MONROE. WebWorks Experiment. https://github.com/MONROE-PROJECT/Experiments/tree/master/experiments/WebWorks
[22] H2020 NEAT Project. A New, Evolutive API and Transport-Layer Architecture for the Internet. https://www.neat-project.org/
[23] Marc C. Necker, Michael Scharf, and Andreas Weber. 2005. *Performance of Different Proxy Concepts in UMTS Networks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 36–51. DOI: https://doi.org/10.1007/978-3-540-31963-4_4
[24] networld2020. 2016. Service Level Awareness and open multi-service internetworking - Principles and potentials of an evolved Internet ecosystem. (2016).
[25] Odroid. http://www.hardkernel.com.
[26] OOKLA. http://www.speedtest.net/.
[27] PATHspider-monroe. https://github.com/mami-project/pathspider-monroe.
[28] PC Engines. APU2C4. https://www.pcengines.ch/apu2c4.htm
[29] Planetlab. https://www.planet-lab.org/.
[30] RaspberryPi. http://www.raspberrypi.org.
[31] RIPE Atlas. https://atlas.ripe.net/.
[32] Anika Schwind, Michael Seufert, Özgü Alay, Pedro Casas, Phuoc Tran-Gia, and Florian Wamser. 2017. Concept and Implementation of Video QoE Measurements in a Mobile Broadband Testbed. In *Proc. of the IEEE/IFIP Workshop on Mobile Network Measurement*.
[33] Sierra-Wireless. MC7455 miniPCI express (USB 3.0) modem. https://www.sierrawireless.com/products-and-solutions/embedded-solutions/products/mc7455/
[34] Narseo Vallina-Rodriguez, Srikanth Sundaresan, Christian Kreibich, Nicholas Weaver, and Vern Paxson. 2015. Beyond the radio: Illuminating the higher layers of mobile networks. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 375–387.
[35] Florian Wamser, Michael Seufert, Pedro Casas, Ralf Irmer, Phuoc Tran-Gia, and Raimund Schatz. 2015. YoMoApp: A Tool for Analyzing QoE of YouTube HTTP Adaptive Streaming in Mobile Networks. In *European Conference on Networks and Communications (EuCNC)*.
[36] Zhaoguang Wang, Zhiyun Qian, Qiang Xu, Zhuoqing Mao, and Ming Zhang. 2011. An untold story of middleboxes in cellular networks. In *Proc. of SIGCOMM*.
[37] Xing Xu, Yurong Jiang, Tobias Flach, Ethan Katz-Bassett, David Choffnes, and Ramesh Govindan. 2015. Investigating Transparent Web Proxies in Cellular Networks. In *Proc. of Passive and Active Measurement*.
[38] Yepkit. USB Switchable Hub YKUSH Homepage. https://www.yepkit.com/products/ykush
[39] Yomo-browser-plugin. https://github.com/lsinfo3/yomo-browser-plugin.git.
[40] Yomo-docker. https://github.com/lsinfo3/yomo-docker.git.
[41] ZeroMQ. http://zeromq.org/.
[42] ZTE. USB-based CAT4 MF910 MiFi, product specification. http://www.ztemobiles.com.au/downloads/User_guides/MF910_Help1.0.pdf