MONROE

Measuring Mobile Broadband Networks in Europe

H2020-ICT-11-2014
Project number: 644399

Deliverable D5.1
User manual and initial user experiences

Editor(s): Miguel Peón-Quirós, Vincenzo Mancuso
Contributor(s): Miguel Peón-Quirós, Anna Brunstrom, Özgü Alay

Work Package:	5 / Users
Revision:	1.0
Date:	February 28, 2017
Deliverable type:	R (Report)
Dissemination level:	Confidential, only for members of the consortium (including the Commission Services)

European Commission
Horizon 2020
European Union funding
for Research & Innovation

Abstract

Report describing the user manual, the user selection process, first user workshop and the initial feedback report from the external users.

Participant organisation name	**Short name**
SIMULA RESEARCH LABORATORY AS *(Coordinator)*	SRL
CELERWAY COMMUNICATION AS	Celerway
TELENOR ASA	Telenor
NEXTWORKS	NXW
FUNDACIÓN IMDEA NETWORKS	IMDEA
KARLSTADS UNIVERSITET	KaU
POLITECNICO DI TORINO	POLITO

Contents

1 Introduction

This report describes the contents of the user manual that is provided to external users as main documentation reference. The user manual guides the users through the process of creating a successful experiment, from the creation of a user certificate to access the system, to the preparation of the containers for the experiments and the final collection of results.

Additionally, this document presents the selection process that the MONROE consortium followed to select the proposals that joined the project at the First and Second Open Calls. We further list the reviewers who served for each open call and provide a short description of the selected proposals. Then, the details of the First User Workshop, held at Oslo during June, 2016, and of the Second User Workshop, held at Madrid during October, 2016, are presented.

Finally, this document comments on the feedback received from the external users after their initial experiences with the MONROE platform. We give particular attention to the issues raised by the users about the platform usability and special needs for their own experiments. We also present the actions that we plan to take (or have already taken) to solve those issues.

2 User manual

The user manual for the MONROE platform is conceived as a live document that evolves to reflect the current situation of the platform and cover the needs of the users. It describes the characteristics of the platform, the life-cycle of a MONROE experiment, the different ways to access MONROE metadata and user experiment results and all possible logistic issues that users of the platform may encounter. The current version of the user manual is appended to the end of this deliverable and the latest version of the document can be found at:
https://github.com/MONROE-PROJECT/UserManual/raw/master/user_manual.pdf

The following paragraphs summarize the contents of the user manual.

2.1 User manual: Introduction

The user manual introduction describes briefly the hardware and software configuration of the MONROE nodes. The goal is that the users get the knowledge needed to successfully design and execute their own experiments with full knowledge of the design characteristics of the platform.

The introduction presents also the experimental workflow proposed by the MONROE consortium. Although some steps are compulsory (e.g., container certification before deployment on "deployed" nodes), users are free to adapt the rest of steps to suit their existing workflows.

2.2 User manual: Experiment preparation

This section of the user manual describes in detail the process of designing an experiment for execution on the MONROE platform. In particular, it explains how to design a Docker container for an existing application, including tips to reduce the size of the containers and minimize their use of resources (particularly, the amount of data required to deliver them to the nodes over MBBs).

The MONROE platform offers the possibility for experimenters to log into their containers while their experiments are in execution. For this, they have to provide an SSH key and the system will prepare a connection string to access the container. In this way, experimenters can monitor the execution of their experiments or even attach to their running processes for debugging purposes.

Finally, this section of the user manual describes the process to certify the containers so that they can be executed in "deployed" nodes, that is, final experimentation nodes (versus "testing" nodes used for development). The certification process, which is under ongoing evolution, is necessary to ensure that containers make a reasonable use of resources and that they do not pose an evident danger for the safety and stability of the nodes. The second consideration is important to reduce the likelihood of an experiment crashing a node that cannot be easily accessed for maintenance, even if several measures of different nature have been put into place to ensure that nodes can be recovered under most circumstances.

2.3 User manual: Resource allocation, and experiment scheduling and monitoring

This section explains how to get a user certificate to access the (web) user interface of the MONROE platform. Then, it explains how to use the user interface to create new experiments, book experiment resources (nodes, quotas, etc.) and monitor the state of ongoing, failed or finished experiments.

The user interface is designed to be easy to use and is divided into three main sections: Experiment monitoring, experiment definition and resource (node) exploration. Additionally, users can get a log of their quota usage to understand how their experiments are consuming their assigned resources.

2.4 User manual: Retrieval of metadata and experiment results

Users can get from the MONROE platform two types of results: The results of their own experiments and the metadata gathered by the MONROE default tools.

This section explains how users can retrieve the results of their experiments, either through the user interface itself or using their own means (e.g., an SCP server). Then, it also explains how to access a replica of the MONROE (Cassandra) database that they can query directly, or the daily CSV dumps that are generated by the platform.

2.5 User manual: Run-time considerations for experimenters

This section of the user manual covers several topics that affect experiments at run-time. First, it addresses how experiments can use the available network interfaces inside the containers to communicate with the outer world. Then, it explains the policies for interface naming and how to find the correlation between network interfaces and operator names at run-time using the stream of metadata that is broadcast inside the nodes. It also explains how interface binding can be achieved, possibly even for old tools that lack command-line options for configuring this aspect.

The section continues explaining the flow of metadata inside the node and how experiments can access it during their execution. Usage of Tstat-provided facilities at run-time is also addressed in this section. Finally, it covers the usage of user-owned development nodes.

2.6 User manual: Monitoring node status

This section briefly describes how the experimenters can employ the user interface to check the status of the platform nodes. This may be useful if they want to repeat an experiment in a certain set of nodes with intermittent availability, such as nodes in bus fleets.

2.7 User manual: MONROE templates, examples and default experiments

This section of the user manual describes each of the examples provided with the MONROE platform. Starting from the simplest "Hello world" test and the experiment template that is provided as a basis for new user experiments, it goes through the list of examples available at the MONROE open source repositories.

2.8 User manual: List of known bugs and issues

This section lists a few known unresolved issues that are frequently raised by users but have not been yet addressed, and presents workarounds when suitable.

2.9 User manual: Appendixes

The user manual includes a set of appendixes that provide details on topics such as the list of packages installed in the MONROE base container image, the format of the metadata fields and technical issues when using Docker containers in Windows computers.

3 External users selection process

The MONROE project has had two open calls for external projects. The calls were publicly announced via the project website, twitter and by sending the news to a list of subscribers interested in knowing more about the project. Various mailing lists and other FIRE related channels were also used for the announcements.

In both cases, the selection process followed similar criteria based on peer-review, as further detailed below for each of the calls.

3.1 First Open Call

Call coordinator: Anna Brunstrom (KaU).
Call committee: Anna Brunstrom (KAU), Özgü Alay (SRL), Håkon Lønsethagen (Telenor), Pedro Andrés Aranda Gutiérrez (Telefónica).
Substitutes: Vincenzo Mancuso (IMDEA), Nicola Ciulli (NXW), Thomas Hirsch (Celerway).
Call announcement: Wednesday, December 23, 2015.
Call deadline: Tuesday, March 15, 2016 at 17:00h CET (Brussels time).
Review deadline: Monday, April 18, 2016.
End of discussion period: Tuesday, April 26, 2016.
Notification of acceptance: Wednesday, May 4, 2016.
Starting date: Wednesday June 1, 2016.

Each proposal was reviewed by at least three reviewers, including one external reviewer. Each reviewer was asked to evaluate up to five proposals, where each proposal was up to 11 pages long. The names of the reviewers for the proposals received during the first open call are listed in Table 1. There is no budget for external reviewer's fees and, therefore, external reviewers costs represent an in-kind contribution free of charge to MONROE.

Table 1: Reviewers of the first open call.

Name	**Affiliation**	**Email**
Stefan Alfredsson	KAU	stefan.alfredsson@kau.se
Johan Garcia	KAU	johan.garcia@kau.se
Mohammad Rajiullah	KAU	mohammad.rajiullah@kau.se
Mats Björkman	Mälardalen University	mats.bjorkman@mdh.se
Ahmed Elmokashfi	SRL	ahmed@simula.no
Yan Zhang	SRL	yanzhang@simula.no
Andra Lutu	SRL	andra@simula.no
Antonios Argyriou	University of Thessaly	anargyr@gmail.com
Berna Ozbek	IYTE	bernaozbek@iyte.edu.tr
Adrian Loch	IMDEA Networks	adrian.loch@imdea.org
Miguel Peón-Quirós	IMDEA Networks	mikepeon@imdea.org
Pablo Serrano	University Carlos III of Madrid	pablo@it.uc3m.es
Marco Ajmone Marsan	POLITO	ajmone@polito.it
Marco Mellia	POLITO	mellia@polito.it
Riccardo Scopigno	ISMB	scopigno@ismb.it
Dario Sabella	TIM	dario.sabella@telecomitalia.it
Thomas Hirsch	Celerway	thomas.hirsch@celerway.com
Audun Hansen	Celerway	audunh@celerway.com
Tomasz Rozensztrauch	Radytek	t.rozensztrauch@radytek.com
Min Xie	Telenor	min.xie@telenor.com
Andres Gonzales	Telenor	andres.gonzalez@telenor.com
Juhoon Kim	Telenor external	J.Kim@telekom.de
Nicola Ciulli	NXW	n.ciulli@nextworks.it
Gino Carrozzo	NXW	g.carrozzo@nextworks.it

The following list details the twelve projects that were chosen to join MONROE as external projects:

1. **"Performance of Web RTC-enabled services on the MONROE platform Acronym: PoWeR"**

   Main goal: Industrial Innovation.
   Participants: N. Amram Technologies Ltd.
   Contact: noam.amram@gmail.com

   WebRTC is an emerging technology enabling real-time voice, video and data sharing in a web browser without the need for browser plugins. Providing a comprehensive set of standards, protocols and APIs; WebRTC's potential in transforming communications is huge. We are involved in the development of an application enabling real time collaboration and interaction of geographically distributed users connecting through a multitude of internet-connected devices. Using the WebRTC technology to develop this application is an attractive option. However, the novelty of the technology and the lack of systematic studies raise some concerns whether and how the required high level of user experience can be achieved consistently in a heterogeneous environment. In this context, to carry out experiments on a large scale measurement platform is a very attractive opportunity to mitigate technological and business risks related to our development. In the proposed experiment, we aim to measure and evaluate key performance indicators of providing WebRTC-enabled applications through MONROE; in particular, the relationship between network metrics and metrics that are directly observed by the end user. In line with our experiment, we aim to implement software extension to MONROE in order to integrate it with the NuboMedia platform. The aim of the NuboMedia is to enable the development of rich and fully-featured WebRTC-enabled applications by providing an elastic scalable cloud platform specifically designed for real-time interactive multimedia services. We believe that both our experiment and our extension will result in benefits beyond our business interests, positively impacting various stakeholders of the platform. The results of our experiment are highly relevant to network operators, regulators and other application developers; while the integration of MONROE and NuboMedia has considerable synergies that can contribute to sustaining the MONROE platform by engaging future experimenters.

2. **"Mobile Network Analytics for Apps Performance Design NAPPLYTICS"**

   Main goal: Scientific Excellence.
   Participants: EUROB CREATIVE and UNIVERSIDAD POLITÉCNICA DE VALENCIA.
   Contact: rgimenez@eurob.es

   Mobile devices and notably smartphones have become cornerstones in our daily lives. Being able to access to real time information is taken for granted in our modern societies as a key enabler for economic and social development. From a pool of 55FOR users, as already pointed out by MONROE call, is becoming popular using measurements network tools (like Netalyzr of MobiPerf) to determine problems in their connectivity. However, for Apps SW developers this approach has not yet become a standard procedure when debugging apps performance. While developers already analyse the user behaviour when using an app: usage frequency, time spent, used functionalities, etc and there are several well-known tools for this (like Yahoo Mobile, Mixpanel or Countly), these tools specifically focus in the user interaction with the app and do not provide data for an effective app tuning in terms of protocols, security, network usage, connectivity strategies, etc from a pure mobile network approach. There is thus an evident room and need for such dedicated tools and related knowledge. By NApplytics project EUROB and the Universidad Politecnica de Valencia will determine which are the KPI that govern an app User Experience considering both protocol and radio issues by means of an android library that could be easily incorporated to any Android app. In order to properly design the library and correctly identify all relevant parameters that may impact user experience the HW, SW and tools provided by MONROE will be essential in the project execution and results.

3. **"Prioritisation and Resilience for Emergency Communications PREC"**

   Main goal: Scientific Excellence.
   Participants: University of Aberdeen Court.
   Contact: gorry@erg.abdn.ac.uk

   This proposal has two components: It extends the MONROE testbed to include new types of infrastructure, and it proposes new experiments to explore survivable networking that rely upon both this new infrastructure and on the existing testbed nodes. Four new permanent nodes will be added to the MONROE platform to increase the diversity of link characteristics. They will support combinations of mobile, Ethernet connections, fixed point-to-point wireless links, low-band radio interfaces and satellite broadband. All these classes of link are used routinely in emergency preparedness exercises and have been used in disaster situations to provide communications. The use-case brings these together in a unified IP network. The performance of a radio link may be affected by environmental factors. The new nodes therefore provide an interface to query physical layer performance data. We will also deploy weather monitoring equipment to allow experimenters to determine why a link has degraded quality. 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, enabling an application to select the best available transport path. Resource-limited paths will ensure traffic is critical traffic prioritised over other traffic, so that network degradation does not halt operation of the application, although performance may be degraded. These techniques will be deployed across the distributed test infrastructure provided by the MONROE platform, allowing experiments to take advantage of the diverse capabilities and connectivity of the nodes. A distributed measurement campaign will measure performance of the developed techniques combining passive link measurements with active traffic generators to measure key performance parameters (e.g., packet loss, latency and jitter). We expect the results of this measurement campaign to inform research into new methods and their applicability across a heterogeneous network.

4. **"Quality-of-Experience of adaptive video streaming in mobile broadband networks NESTOR"**

   Main goal: Industrial Innovation.
   Participants: StreamOwl (SO) and Athens University of Economics and Business (AUEB).
   Contact: savvas@streamowl.com

   The wide deployment of multimedia services over packet networks has highlighted that the original design of the Internet as a best-effort network makes it problematic for bandwidth-intensive and delay-sensitive applications, like video streaming. Additionally, network operators typically employ centralized delivery architectures that lead to long paths between end-users and content servers, waste of network resources and increased delays. This issue becomes more important in mobile broadband networks, which have stringent resource limitations and fast-changing conditions. The NESTOR project aims at conducting an experiment campaign based on the MONROE platform to evaluate the Quality-of-Experience (QoE) of popular video streaming services (e.g., YouTube and Netflix) with active and passive measurements. Special emphasis will be given to adaptive video streaming (especially MPEG-DASH), which enables the seamless adaptation of the video client to the specific network conditions of each user and is more relevant to mobile broadband networks. The understanding of the impact of the network parameters and the media content on the human perception are key factors in optimizing the end-to-end delivery chain. More importantly, the NESTOR project will help create a sustainable experimental platform which will be able to be used after the end of the measurement campaign by third-parties (e.g., operators, video platforms, and network engineers) to test and evaluate the performance of adaptive video streaming services in a seamless, intuitive and unobtrusive manner. Thus, the experiments will assist in benchmarking mobile broadband operators in terms of reliability and in the identification of key performance indicators that could be used to improve their networks.

5. **"Monitoring and Analysis of Quality of Experience in Mobile Broadband Networks Mobi-QoE"**

   Main goal: Scientific Excellence.
   Participants: AIT Austrian Institute of Technology GmbH, Tran-Gia Informationstechnik GmbH.
   Contact: pedro.casas@ait.ac.at

   Quality of Experience (QoE) is a well-known concept in the networking research community, but its development has been traditionally limited to laboratory studies. We are witnessing a growing demand from mobile operators for more user-centric approaches to manage their networks in an increasingly competitive scenario. This need has boosted the research interest in scaling QoE out of the lab, bringing it into the management of operational networks. This combined industry/research growing interest in QoE-based solutions for network management motivates the Mobi-QoE proposal. Mobi-QoE proposes both to extend MONROE's testbed with new software (SW) tools as well as conducting innovative experiments. The first objective of Mobi-QoE is to extend MONROE's testbed to the QoE domain, by integrating novel SW-based QoE-capable measurement tools and QoE subjective-models to assess the performance of MBB networks for popular end-user services (e.g., YouTube, Facebook, Spotify, etc.) from a user-centric perspective. Such tools and models provide a multi-layer monitoring perspective, measuring QoE-relevant features at the network and application layers, and forecasting end-user experience (e.g., MOS scores). The second objective of Mobi-QoE is to evaluate MONROE's testbed visibility in terms of QoE metrics, relying on crowdsourced field trials. Mobi-QoE plans to evaluate the MONROE QoE capabilities introduced by the Mobi-QoE tools and models, comparing the MONROE-Mobi-QoE forecasted end-user experience to that reported by real MBB users through field trials, conducted with Mobi-QoE tools directly at end-user mobile devices (i.e., smartphones) in areas covered by the MONROE platform. Crowdsourced measurements would allow to further refine the integrated QoE models. To further show the benefits of the Mobi-QoE extensions to MONROE, the proposal foresees to apply the tools and models to study different MBB measurement problems from a QoE perspective, conducting several experiments. These include: Benchmarking ISPs from a QoE perspective, analyzing the impact of mobility on QoE, and detecting and diagnosing QoE-relevant degradation in MBB networks.

6. **"Software defined network based available bandwidth measurement in MONROE (SOMETIME)"**

   Main goal: Scientific Excellence.
   Participants: University of Napoli Federico II.
   Contact: pescape@unina.it

   In SOMETIME, we plan to add Available Bandwidth (ABw) estimation to the set of metrics collected by MONROE and to characterize in terms of ABw the broadband mobile networks considered in MONROE. We will implement the estimation by active measurements leveraging Software Defined Networks (SDN) paradigm, both to tune the technique considering interference with node-local processes (a more realistic scenario compared with mutually exclusive measurements), and to mitigate such interference. ABw can be considered as the spare capacity on a network path, and is an important metric with many applications; optimized routing, adaptive encoding, just to name the most common. It is related to other path capacity metrics, such as TCP bulk transfer capacity, but is different from them, with different measurement procedures and practical applications. Specifically, ABw provides an upper bound to available network capacity, dependent on network infrastructure and usage, but independent from the transport protocol and application. This makes ABw a highly valued metric to characterize operational conditions of a network path, useful to inform at the same time different kinds of applications without per-application tests. As a counterbalance of its importance, ABw estimation is not trivial, especially in wired-cum-wireless scenarios (like the MONROE testbed), that also pose constraints related to volume based billing and capping for cellular data connections. Possible applications for experiments in MONROE (e.g., multimedia streaming to smartphones, tablets, in-vehicle-infotainment systems) present the additional challenge of possible sharing of computing and communication resources during the measurements: To emulate, assess and mitigate this interference we will adopt an SDN-based approach to manage traffic in the mobile terminal (MONROE node). This approach will have the additional outcome of evaluating the possibility of running multiple active experiments concurrently on the MONROE node, guaranteeing isolation of traffic engineering or routing setups required by the different experimenters.

7. **"MOVEMENT: Extending and Experimenting upon the MONROE platform towards Voice and VidEo StreaMing AssEssmeNT in Mobile Networks"**

   Main goal: Industrial Innovation.
   Participants: COSMOTE Mobile Telecommunications S.A. and FERON TECHNOLOGIES P.C.
   Contact: glimperop@cosmote.gr

   The installation and maintenance of sustainable, large-scale platforms for monitoring commercial mobile broadband networks performance is considered a key theme within all mobile technology stakeholders, including operators, service providers, telecom SMEs, and regulatory authorities. Such platforms obtain objective information on the experience levels perceived by the end-users and correlate these levels with network performance indicators. MONROE is a promising approach towards this direction, and MOVEMENT aims to add significant value to the existing platform. This will be achieved through the provision of an open access "toolbox" of innovative components developed upon the MONROE open platform, that will support the experimentation and analysis of voice and video streaming services over 4G/Wi-Fi mobile data networks. MOVEMENT plans to: a) geographically expand the existing monitoring network with the deployment, integration, and operation of a set of new stationary and mobile nodes; b) enable the automated active testing of voice and video streaming services, including voice-over-IP (VoIP), VoLTE and on-demand video streaming (VoD) over 4G/Wi-Fi data networks, by developing a set of client and server agents for performing the tests, along with the front-end and back-end infrastructure for collecting the measurements and visualizing the QoS and QoE metrics; c) run various VoIP and VoD quality assessment campaigns over a live commercial network, as well as deploy an experimental platform based on open server tools (IMS, SIP) and live/test 4G eNBs for testing VoLTE services. The proposed extension and experiment activities are expected to enhance the MONROE project footprint, contribute towards the long-term project sustainability, and have a direct impact beyond the MONROE community, including the broader industrial and scientific communities, by providing open measurement and assessment tools and methodologies, as well as measurement data for prevailing (VoIP, VoD) and emerging VoLTE services.

8. **"Affordable LTE Network Benchmarking Based On MONROE Acronym: MONROE-LTE"**

   Main goal: Industrial Innovation.
   Participants: Allbesmart LDA.
   Contact: pmarques@allbesmart.pt

   To gain competitive advantage in today's mobile market, cellular network testing, monitoring and improving customer experience is crucial. Today independent benchmarking companies are hired by mobile operators to run drive tests in a certain geographical areas. The high cost for running these tests results in a low frequency of execution, typically this benchmarking is executed no more than 2-3 times per year, which is not sufficient to follow the dynamics of an LTE network in a dense urban area. The majority of the drive testing costs come from the car, driver, and the in-car technician. ALLBESMART, a SME specialized in smart city solutions, has already developed a Bus Tracker solution for several transportation companies in Portugal. The business idea behind this proposal is to take advantage of the existing partnerships with these companies to carry on and sell network benchmarking services to Mobile Network Operators (MNO). With this approach, unattended measurement nodes can be deployed in existing transportation fleets without the need for dedicated field personnel, reducing the cost of testing up to 70 %. This project is both an experiment and a HW/SW extension of MONROE. The experiment will use mobile nodes available in Turin (Italy) to create and validate the Allbesmart automatic LTE network benchmarking tool using the MONROE nodes placed in buses. After that, 5 additional MONROE mobile nodes will be deployed in vans circulating in Lisbon (Portugal) city centre. The "benchmarking tool" (software extension) will be developed allowing an easy comparative analyses of mobile network QoS and QoE KPIs based on MONROE raw data. To showcase the MONROE extension in Lisbon, three LTE MNOs will be considered and the results discussed with representatives of the targeted MNOs and the Portuguese regulator. Furthermore, a viable sustainability model for the proposed MONROE extension will be developed.

9. **"Software Radio for Measuring Mobile Broadband Networks SOPHIA"**

   Main goal: Scientific Excellence.
   Participants: Software Radio Systems.
   Contact: paul.sutton@softwareradiosystems.com

   A software radio combines a generic radio frequency (RF) front-end with a general-purpose processor to provide a powerful platform for wireless systems testing, measurement and experimentation. The proliferation of commodity RF front-end devices and the increasing capabilities of general-purpose processors have enabled a thriving open-source software radio ecosystem with projects focusing on every imaginable wireless technology and application. Software Radio Systems (SRS) is an Irish SME which builds on over 15 years of software radio research and development. Through the open-source srsLTE and srsUE projects, SRS provides high-performance tools for analysis and implementation of 4G LTE broadband networks and applications. Under SOPHIA, SRS will extend the MONROE platform to support software radio applications. SOPHIA will use the existing SRS open-source libraries and applications to demonstrate the benefits of software radio and to carry out experiments to perform detailed measurements of mobile broadband networks. The project will further examine the capabilities and limitations of the MONROE node as a software radio platform and provide comprehensive feedback to the MONROE consortium on performance, user experience and recommended extensions and improvements. The impact of SOPHIA will be to significantly expand the range of experiments which can be supported by the MONROE testbed, to enhance the value of existing experiments and to greatly increase the pool of potential testbed users and experimenters. SOPHIA will vastly enhance the detail with which mobile broadband networks can be measured and analyzed. It will increase the useful lifetime of the testbed by enabling over-the-air software upgrades to deploy new measurement tools and support newly deployed technologies. Finally, SOPHIA will open the MONROE testbed up to the full open-source software radio ecosystem, the huge array of existing projects, tools and applications and the vibrant community of designers, developers and users.

10. **"Utility-based Networking experiments for Improving QUality of Experience in mobile broadband environments UNIQUE"**

    Main goal: Scientific Excellence.
    Participants: Institute of Communications & Computer Systems (ICCS) and Incelligent (Incel).
    Contact: papavass@mail.ntua.gr

    In UNIQUE, we aspire to evolve, implement and experiment with a unified utility-based framework based on both QoS (network-centric) and QoE (user-centric) features and parameters, which allows to enhance, formulate and improve various advanced network operations in mobile broadband (MBB) infrastructures (network selection, offloading, resource allocation, etc.). The proposed methodology and intended MONROE software extensions are expected to be provided as services to future experimenters that desire to investigate their own mechanisms and protocols. UNIQUE targets at two key runs of experiments and respective software extensions, depicting the operation and benefit of the proposed utility-based framework, i.e., a) utility-based access network selection, and b) utility-based concurrent multipath routing and traffic control. They aim at demonstrating the power of the proposed framework for key user-desired and bandwidth intensive network services and applications, such as multimedia downloading. In order to enable these scenarios and enhance MONROE capabilities for advanced experimentation involving end-to-end performance evaluation and measurements, various software extensions enabling utility-based network study and optimization will be developed and integrated in the MONROE platform. Thus overall, UNIQUE, in addition to advancing the state-of-the-art in the research area of network utility performance optimization in wireless mobile networks, suggests long-term software extensions of the platform, and experiments that will leverage the available infrastructure and provide feedback to MONROE for improving its operation and extending the offered services. The expertise of the two participating groups (i.e., ICCS, with long research experience in wireless networks, and Incelligent, a company offering software products & services for management of wireless networks) is complementary, ensuring a full-circle approach from scientific excellence to final integration of the proposed methodology and experiments, to software and service delivery.

11. **"MARiL-in-MONROE Measurement Adaptation and Reporting in LTE"**

    Main goal: Scientific Excellence.
    Participants: University of the Basque Country.
    Contact: fidel.liberal@ehu.eus

    The rise of Mobile Broadband (MBB) usage comes together with an increasing awareness in quality of service as perceived by customers. Whilst many approaches from both the standardization bodies and the scientific environment have tried to define tools to measure and represent objective and subjective network performance, no consensus has been reached regarding the most feasible and accurate measurement methodology. Taking advantage of MONROE MBB networks measurement deployment, MARiL proposes to develop and contribute as open source multi-layer measurement tools that implement the most significative standardized Internet access measurement methods. In this regard, MARiL targets both main TCP-based and IP-layer model-based approaches. Regarding the validation of this proposal and seeking for a deep study among different contexts of use, several tests will be designed and carried out in the MONROE's Pan-European deployment. Firstly, static nodes will be used in order to establish a fair comparison between both measurement philosophies. Secondly, a large-scale measurement phase will be held under different mobility patterns, constraining developed tools in such challenging conditions. The results comparison amongst these alternatives will help determining the most adequate measurement philosophy under each scenario. Finally, helped by the metadata obtainable from MONROE node, MARiL will propose a cross-layer mapping of objective performance metrics aiming to define a more accurate relationship between performance at different layers under the aforementioned conditions. Hence, MONROE will benefit from MARiL's for its validation and addressing of MONROE as the main large-scale platform in Europe, as well as for its developments and studies for future open-calls and the further research advance. The outcome of MARiL will also input to different standardization bodies, including ITU-T, IETF and ETSI, while targeting liaison with RIPE and M-LAB initiatives. Thus, MARiL leverages the impact of the whole MONROE project in terms of transfer-to-standards and open data/open source communities.

12. **"Rapid Interpretation and Cross-Experiment Root-Cause Analysis in Network Data with Orange RICERCANDO"**

    Main goal: Scientific Excellence.
    Participants: University of Ljubljana.
    Contact: `fabio.ricciato@fri.uni-lj.si`

    The goal of the RICERCANDO project is to develop an advanced toolbox for mining MONROE data to support integrative exploration, visualization and interpretation of data and meta-data across multiple experiments. The main use of the envisioned toolbox is to facilitate the identification and interpretation of anomalies and problems within the data (e.g., clusters of measurements reporting particularly poor performances). Our aim is to (1) ease the problem discovery and troubleshooting of the MONROE monitoring system, (2) avoid erroneous accounting to the monitored broadband mobile network(s) of data anomalies caused by glitches within the monitoring system, and (3) speed up the correct identification of the source of the problem (Root-Cause Analysis). RICERCANDO does not propose new additional experiments. Instead, it will consider data and meta-data from the whole range of experiments supported by the MONROE platform, and combine them with context data about the status of the MONROE system itself (e.g., software and hardware configuration of every system component). The integration of these data with advanced data mining, visualization and interactive data exploration features will support the human expert(s) in the process of detecting and understanding the problem, narrowing down to a short list of candidate critical dimensions. A distinguishing feature of RICERCANDO is the interdisciplinary composition of the project team that includes established data mining experts working together with networking experts. The envisioned toolbox will be developed as an extension of Orange (`orange.biolab.si`), an established open-source data mining tool widely adopted by different scientific communities, including but not limited to bioinformatics and teaching of data science. The new toolbox, specifically tailored for MONROE data and use-cases, will be released as open-source.

3.2 Second Open Call

Call coordinator: Anna Brunstrom (KaU).
Call committee: Anna Brunstrom (KAU), Özgü Alay (SRL), Håkon Lønsethagen (Telenor), Pedro Andrés Aranda Gutiérrez (Telefónica).
Substitutes: Vincenzo Mancuso (IMDEA), Nicola Ciulli (NXW), Thomas Hirsch (Celerway).
Call announcement: Friday, September 23, 2016.
Call deadline: Wednesday, December 2, 2016 at 17:00h CET (Brussels time).
Review deadline: Monday, January 16, 2017.
End of discussion period: Monday, January 23, 2017.
Notification of acceptance: Wednesday, February 1, 2017.
Starting date: Wednesday March 1, 2017.

Each proposal was reviewed by at least three reviewers, including one external reviewer. Each reviewer was asked to evaluate up to six proposals, where each proposal was up to 11 pages long. The names of the reviewers for the proposals received during the second open call are listed in Table 2. There is no budget for external reviewer's fees and, therefore, external reviewers costs represent an in-kind contribution free of charge to MONROE.

The following list details the fifteen projects that were chosen to join MONROE as external projects:

1. **"Characterizing Carrier Grade NATs in Mobile Broadband Networks: CGNWatcher"**

   Main goal: Scientific Excellence.
   Participants: Universidad Carlos III de Madrid.
   Contact: marcelo@it.uc3m.es

   Deploying Carrier Grade NATs (CGNs) enables Mobile Broadband Network operators to provide Internet access services to a very large number of customers with a limited amount of public IP addresses. However, CGNs impose a number of functional and potentially performance penalties. In the CGN-watcher project we will develop a measuring tool that executes a number of active tests to fully characterize CGN deployments in Mobile Broadband Networks. The CGNwatcher tool will systematically test for over 40 behavioural requirements for NATs defined by the Internet Engineering Task Force (IETF) and also for multiple CGN performance metrics. The metrics will be designed to work in cascaded-NAT scenarios, isolating the characteristics of the different NATs along the path whenever is possible. As part of the project, we will deploy CGNwatcher in MONROE and perform large measurement campaigns to characterize the real CGN deployments of the ISPs serving the MONROE nodes. Additionally, we will deploy CGNwatcher in a crowdsourcing platform, in order to complement the data set obtained through MONROE and extract some conclusions of how representative is MONROE of other regions of the Internet in terms of CGN deployments. The measurement results obtained through CGNwatcher will be relevant to application and protocol developers to inform their design about how to overcome the limitations imposed by CGNs. The information retrieved though CGNwatcher will be also useful for experimenters using MONROE, as CGNs may have an important impact in the feasibility of experiments and can potentially bias the results if not accounted for. As part of the project, we will disseminate the results through academic publications and demonstrations and we will promote the adoption of the proposed metrics in the IETF in order to obtain a standardized set of metrics for CGN characterisation.

Table 2: Reviewers of the first open call.

Name	Affiliation	Email
Stefan Alfredsson	KAU	stefan.alfredsson@kau.se
Johan Garcia	KAU	johan.garcia@kau.se
Mohammad Rajiullah	KAU	mohammad.rajiullah@kau.se
Mats Björkman	Mälardalen University	mats.bjorkman@mdh.se
Ali Parichehreh Teroujeni	Prisma Telecom Testing	apmastermail@gmail.com
Satyanarayana Vuppala	University of Edinburgh	vsn411@gmail.com
Ahmed Elmokashfi	SRL	ahmed@simula.no
Andra Lutu	SRL	andra@simula.no
Chad Jarvis	SRL	chad@simula.no
Antonios Argyriou	University of Thessaly	anargyr@gmail.com
Berna Ozbek	IYTE	bernaozbek@iyte.edu.tr
Miguel Peón-Quirós	IMDEA	mikepeon@imdea.org
Foivos Michelinakis	IMDEA	foivos.michelinakis@imdea.org
Christian Vitale	IMDEA	christian.vitale@imdea.org
Antonio de la Oliva	UC3M	aoliva@it.uc3m.es
Vicent Cholvi	University Jaume I	vcholvi@uji.es
Marco Ajmone Marsan	POLITO	ajmone@polito.it
Marco Mellia	POLITO	mellia@polito.it
Marco Fiore	CNR - IEIIT	marco.fiore@ieiit.cnr.it
Alessandro Nordio	CNR - IEIIT	alessandro.nordio@ieiit.cnr.it
Dario Sabella	TIM	dario.sabella@telecomitalia.it
Thomas Hirsch	Celerway	thomas.hirsch@celerway.com
Audun Hansen	Celerway	audunh@celerway.com
Peyman Teymoori	UIO	peymant@ifi.uio.no
Min Xie	Telenor	min.xie@telenor.com
Andres Gonzales	Telenor	andres.gonzalez@telenor.com
Per Hjalmar Lehne	Telenor	per-hjalmar.lehne@telenor.com
Olav Østerbø	Telenor	olav-norvald.osterbo@telenor.com
Juhoon Kim	Deutsche Telekom	J.Kim@telekom.de
Nicola Ciulli	NXW	n.ciulli@nextworks.it
Gino Carrozzo	NXW	g.carrozzo@nextworks.it
Savvas Argyropoulos	StreamOwl	savvas@streamowl.com
Antonis Gotsis	Feron Technologies P.C.	antonis.gotsis@feron-tech.com

2. **"Network Neutrality in Mobile Broadband: NeutMon"**

   Main goal: Scientific Excellence.
   Participants: University of Pisa.
   Contact: alessio.vecchio@unipi.it

   EU-wide rules concerning net neutrality are one of the major achievements towards the Digital Single Market. According to these rules, blocking, throttling, and discrimination of traffic by Internet Service Providers (ISPs) is not allowed. All traffic has to be treated equally, and no form of traffic prioritization can be enforced (with few exceptions: Preserving the integrity of the network, managing temporary congestions, and compliance with legal obligations). So far, research on net neutrality focused on the wired part of the Internet. However, in recent years, smartphones and tablets have become the preferred choice for accessing a large number of networked services and applications, from social networks to video streaming. The NeutMon project is aimed at studying net neutrality in a mobile broadband context. The main goals of the project are: i) Collecting a set of measurements related to net neutrality in mobile broadband using the MONROE infrastructure; ii) developing methods for analysing such data and thus inferring the neutrality level of mobile broadband operators. Collected data will include the most important network metrics (e.g., bandwidth, delay) with traffic belonging to different classes (e.g., file sharing, HTTP). Also the path at both router and autonomous system level will be collected and analysed. As far as the analysis of data is concerned, it is important to note that currently available techniques may be not well suited to operate in a scenario with mobile broadband access. As known, wireless networks are more prone to large fluctuations than wired ones: Performance levels may vary significantly in a short interval as a consequence of interferences, mobility, and environmental factors. Effective techniques for the evaluation of neutrality in these settings have to incorporate these concepts from the beginning and must be able to distinguish deliberate traffic engineering from the peculiar variability of the considered environment.

3. **"Reconstruction of operator policies in MBB networks for improved user experience: RECON"**

   Main goal: Scientific Excellence.
   Participants: Eötvös Loránd University.
   Contact: lakis@elte.hu

   The complex Internet infrastructure plays a crucial role in determining end-to-end experiences of the users. The current Internet connects around 2 billion users through a global communication infrastructure that consists of thousands of service providers of different business types. A significant fraction of the users connect to the Internet through mobile broadband networks; using their notebooks, tablets and smart-phones. With the advent of 3G and 4G, a huge number of mobile applications have emerged with a wide range of requirements against the networking infrastructure. Ensuring high Quality of Experience has utmost importance for application developers. However, it is also well-known that network operators apply various policies in different layers of networking, affecting end-to-end traffic characteristics and eventually the observed Quality of Experience. Reconstructing and understanding these policies may help developers to make their applications compliant with the network and thus improve the provided QoE for end users. Such information can also be important for decision makers and supervisory bodies to get feedback about applied traffic management policies and practices, as well as the level of network neutrality to check their compliance with local regulations. In this project, we aim at developing a methodology and toolboxes (as extensions) for MBB networks to reconstruct operator policies applied in Network, Transport and Application Layers, following a way of reverse engineering. At the network layer, we will mainly investigate intra and inter-domain routing policies with special focus on the stability of network paths and on the topology changes of mobile end-hosts. At the transport layer, UDP and TCP functionality (e.g., UDT, MPTCP, ECN, TCP Fast Open) of different network paths will be tested through active measurements. The system allows experimenters to measure wide diversity of paths and can help determine whether a proposed solution has the required support or functionality from the MBB ecosystem. At the application layer, operators often apply traffic differentiation — giving better (or worse) performance to certain classes of Internet traffic. This also provides vital information for the net-neutrality debate.

4. **"Experimental validation of REM-based machine learning algorithms for SON using MONROE nodes: MONROE-SON"**

   Main goal: Industrial Innovation.
   Participants: RED Technologies SAS, Instituto PolitÃ©cnico de Castelo Branco.
   Contact: pjmuller@redtechnologies.fr

   SON (Self-Organizing Network) technology minimizes the lifecycle cost of running a mobile network by eliminating manual configuration of equipment and troubleshooting during operation. This can significantly reduce the cost of the mobile operator's services. Mobile operators are keen to capitalize on SON to minimize rollout delays and operational expenditures associated with their ongoing LTE deployments. The SON ecosystem is increasingly witnessing convergence with other technological trends such as LSA (Licensed Shared Access), a technology that enables mobile operators to temporarily increase the available bandwidth by using portions of the radio spectrum previously reserved for other uses. Learning and prediction of network behavior are key enablers towards the implementation of the SON paradigm. In this experiment we will use advanced machine learning algorithms to predict network congestion for a specific cell/time of the day by leveraging on *a priori* measurements from MONROE nodes, and then activate the LSA mechanism to increase LTE capacity based on a demand-driven approach. The experiment will start by using measurements from MONROE nodes in Oslo and Madrid to select and tune a machine learning algorithm able to predict cell congestion. After that, two additional MONROE nodes will be deployed in a central district in Paris, adding an extra European capital to the MONROE federation. Radio Environment Maps with a real network deployment in Paris will be used to validate the automatic activation of LSA bands in case of capacity outage. The machine learning API developed in this project will be open source and available for future MONROE use, as a software extension, able to extract useful insights from the huge amount of data generated by MONROE experiments. The outcome of this project will have a strong impact on the LSA standardization process in ETSI RRS (Reconfigurable Radio Systems) where RED Technologies is an active contributor.

5. **"Multi-homing with Ephemeral Clouds on the Move: MEC"**

   Main goal: Scientific Excellence.
   Participants: University of Macedonia, Greece.
   Contact: emamatas@uom.edu.gr

   A main research challenge in 5G Networks is an efficient synergy of the mobile network edge with nearby cloud deployments that achieves ultra-low latency and high bandwidth, while enabling innovative applications. However, in high-mobility environments it is not easy to deploy traditional clouds nearby. Furthermore, such technologies use full-scale operating systems with most of their codebase unneeded (e.g., a web server may require 50 MB but reserve a 20 GB virtual machine). We suggest that Unikernel Virtual Machines (UVMs) fit this context very well, since they have a very small size and rapid boot-up times, i.e., can be responsive to dynamic changes in the network conditions. The MONROE project offers unique experimentation capabilities utilizing both highly-mobile environments and real operational Mobile Broadband (MBB) networks. In MEC, we complement the MONROE platform with lightweight cloud capabilities residing in the mobile nodes. We plan to experiment with: (i) Intelligent orchestrated cloud resources that improve mobile communication and adapt to the conditions of the MBB networks, (ii) Multi-homing capabilities that consider resource offloading to the nearby cloud resources, and (iii) Novel forecasting mechanisms for dynamic network conditions. The above will be demonstrated with three novel scenarios utilizing the MONROE platform and our SWN test-bed (i.e., Web Load Balancing, Ephemeral VMs Orchestration and Internet of Things).

The MEC project extends the MONROE platform with the following aspects:

- The Lightweight Edge Cloud and the Multi-Homing Decision Engine realizing a Unikernel-based cloud and handling multi-homing decisions, respectively.
- The Orchestrator supporting decision engines for multi-homing strategies and UVMs orchestration along with a Prediction Engine forecasting the evolution of the network conditions.
- A number of showcase components, such as a Web Load-Balancing Controller, a UVM Repository and an IoT controller.
- Visualization tools showing both the wireless/mobile and lightweight cloud contexts, as well as performance measurements.

6. **"Programmable and Robust Smart Grid Data and Control: RASnet"**

   Main goal: Scientific Excellence.
   Participants: DAI-Labor, TU Berlin.
   Contact: thomas.huehn@dai-labor.de

   The main motivation of our project "Programmable and Robust Smart Grid Data and Control" (RASnet) is driven by our current systems research of ICT for Low-Voltage grids and its open question of how to properly make use of all available data-links between the electrical grid and its operator. To ensure a robust operation of a low-voltage grid with significant proportion of renewable energy sources within the required power-quality specifications, operators need to be able to observe the dynamic power quality state and gain control of its generation sources, both in the time domain of milliseconds.

   With our proposed RASnet project we aim to analyze the potential integration of current 3G/4G networks in a productive broadband network of a rural Wireless ISP (WISP), Evernet, with significant renewable energies. We want to extend our current research infrastructure placed in the WISP network in such a way that the current set of uplinks (DSL, VDSL, satellite) is extended to consider and integrate 3G/4G network links as additional uplink connections to increase the grid operator's main goal of a robustness monitoring and control channel to his grid.

   The goals of the RASnet research proposal are:

   - Integration of 5 MONROE nodes into the productive WISP infrastructure from Evernet, which serves about 130 households across 5 villages based on Linux OpenWrt-WiFi-routers, including 13 Photovoltaic-systems and 2 Windparks.
   - Performing long-term measurement trials to analyse the performance of rural 3G and 4G networks in order to consider them as alternative uplinks to monitor and control the electrical grid state with low latency via a robust feedback channel.
   - Analysing MONROE data-traces to generate proper input parameters for our SDN controller about available latency and data-rate in order to achieve the overall goal of providing a programmable path for ultra-robust data and control access to the Smart Grid operator.

- Extracting and disseminating meaningful and generalized insights for the 5G research community on attaining ultra-reliable and low-latency communications using SDN-based aggregation of heterogeneous access technologies.

7. **"Towards end-to-end Multipath TCP"**

   Main goal: Scientific Excellence.
   Participants: Université Catholique de Louvain.
   Contact: Olivier.Bonaventure@uclouvain.be

   Multipath TCP, defined in [RFC6824], is one of the major protocols that can really exploit the path diversity that exists in today's mobile and broadband networks. During the last years, a growing number of use cases have been developed above this new protocol. Until now, Multipath TCP has mainly been used on proxies applications that are unaware of the new capabilities brought by Multipath TCP, with one notable exception. We expect that in the coming years applications will be tuned to interact directly with Multipath TCP. In this project, our objective is to prepare this evolution by enhancing test tools to correctly interact with Multipath TCP through the enhanced socket API and by extensively testing it in the MONROE testbed. More precisely, we aim at measuring and possibly improving the Multipath TCP implementation in the Linux kernel for two use cases that are important for mobile nodes:

   - Short request/responses such as those corresponding to applications using voice recognition.
   - Audio and video streaming applications.

   We will then exploit the lessons learned from these measurements to improve the enhanced API for Multipath TCP and contribute to its standardization within the IETF.

8. **"Smart City Security Monitoring Platform: Cloud Eyes"**

   Main goal: Industrial Innovation.
   Participants: Institute of Bioorganic Chemistry of the Polish Academy of Sciences, Poznań Supercomputing and Networking Center (PSNC), and TechInnowacje sp. z o.o.
   Contact: astagor@man.poznan.pl

   Cloud Eyes is a Smart City Security Monitoring Platform enabling real-time video surveillance using WiFi and mobile broadband (MBB) networks, with dynamic adaption of transmission parameters. It is designed to provide monitoring from buses, trams, trains, robots and places where connection using a wired network is not possible. Before going to the market, we need to perform validation of the platform in a real-world MBB environment. That is why we will use the MONROE testbed to deploy Cloud Eyes in a distributed system of mobile and stationary nodes connected via various wireless networks, and to measure the key network metrics and the corresponding video quality. Our goal is to determine the correlation between Quality of Service and Quality of Experience, define encoding and transmission profiles for various network conditions and enhance our smart adaptive algorithms, making Cloud Eyes a market-ready solution running on heterogeneous wireless networks. Our product is beyond the state-of-the-art of currently available monitoring solutions as it allows for continuous, real-time surveillance, utilising both WiFi and MBB networks, whereas our competitors' solutions rely only on WiFi and are not capable of continuous, real-time monitoring. Our product takes advantage of the full capacity of MBB and overcomes problems of varying transmission quality by using dynamic adaptation. PSNC will organise and take part in dissemination and demonstration events to attract customers such as transportation companies, city authorities and law enforcement institutions. As PSNC specializes in network technologies and has been using pan-European testbeds in many projects, it can provide valuable feedback about the MONROE testbed to the consortium. PSNC envisions to negotiate with the consortium the possibilities of using the testbed to complement other laboratory facilities that we currently offer to companies in the region. Such commercial offering can help to ensure sustainability of the testbed after the project ends.

9. **"Optimization of QoE of Mobile Broadband Services through Machine Learning: OPTIMAL"**

   Main goal: Industrial Innovation.
   Participants: Modio Computing SP, Paris Descartes University (PDU).
   Contact: i.georgiadis@modio.io

   Resource allocation within a mobile broadband (MBB) environment remains a challenging topic as it faces highly dynamic user demand and volatile network conditions while it must ensure an acceptable Quality of Experience (QoE) in the provided services without over-provisioning network resources. To solve this problem, we propose the implementation and experimental validation of a novel approach to intelligent resource allocation within MBB environments. Our approach enables the dynamic adaptation of resources to users' demands effectively preventing possible degradation of the QoE of the service before this manifests to the client-side and becomes noticeable by the end users. In the proposed experiment, we will validate our approach using video streaming as the use case, which is one of the most popular and resource-demanding MBB services. Our methodology is composed of four stages:

   (a) Collection of metrics from the MONROE testbed and calculation of KPIs.
   (b) Correlation of the KPIs to a measurable form of QoE.
   (c) Prediction of future KPIs based on existing KPIs and the correlation model.
   (d) Generation of resource allocation actions both in the node (client-side) and in the server to improve the QoE.

   Our methodology includes state-of-the-art techniques for large-scale data correlation and employs machine-learning algorithms to forecast the degradation of QoE and the generation of suitable provisioning actions to prevent this degradation. Our approach shares some common aspects with autoscaling techniques employed by cloud providers to optimize the resource allocation. Furthermore, resource autoscaling using computational intelligence is currently an active research topic in 5G infrastructures. We believe that our experiment will contribute to both MBB and 5G research, thus promoting the sustainability of the MONROE testbed.

10. **"Fast and Lightweight Capacity Benchmarking of Mobile Broadband Networks in MONROE: FaLiCaB"**

    Main goal: Scientific Excellence.
    Participants: TU WIEN / Institute of Telecommunications (TUW), A1 Telekom Austria AG (A1).
    Contact: psvoboda@nt.tuwien.ac.at

    The capacity of the connecting network links mainly defines the quality of service for an end user. There are several parameters affecting network communication links, an important one being the available end-to-end bandwidth. Several different approaches for measuring the network link capacity exist in industry, standardization and the scientific community. However, their implementations are resource demanding and require a long measurement time. Both properties do not deal well with the challenges found in mobile broadband networks, especially when targeting a distributed measurement system. Bandwidth measurement in cellular wireless networks, with traffic reactive nature, nomadic end users, tariff data rate limitations and highly dynamic shared resources, is a challenging task. The goal of FaLiCaB is to integrate a packet dispersion based solution extended by parameter estimation into the MONROE platform. These new method needs validation in a nomadic setup; thus, the MONROE platform offers a unique opportunity for the FaLiCaB consortium. The project will implement an estimation method for link capacity with packet dispersion into the MONROE system. The validation will use static MONROE nodes to allow for a fair comparison between new and existing methodologies. The new module will run in a large measurement campaign on mobile MONROE nodes validating the approach for moving end users. Finally, the project will use the MONROE metadata to understand the impact of events on the results from different layers, allowing an augmentation of the capacity estimator via passively monitored parameters. The module will be cross-validated with nodes locally positioned in Austria and equipped with unlimited SIM cards from A1. The MONROE project will gain the major mobile operator in another European country with FaLiCaB. The TUW will provide its performance test setup. The MONROE project will benefit from a new module for volume lightweight and fast bandwidth estimation suitable for crowdsourced measurement systems.

11. **"Network Self-Optimization based on End-To-End measurements: eSON"**

    Main goal: Industrial Innovation.
    Participants: Universidad de Malaga (UMA).
    Contact: rbarco@uma.es

    eSON proposes an experiment that studies the effect of Self-Organizing Network (SON) functions on End-To-End (E2E) performance of Mobile Broadband (MBB) terminals. SON functions automatically adjust the configuration of the Radio Access Network (RAN) and the core of cellular networks based on network measurements and Artificial Intelligence (AI) algorithms, saving costs and reducing downtimes. The proposed experiment will take E2E measurements in the MONROE nodes in two scenarios: A baseline where no SON algorithms are present in the cellular network, and an enhanced network where self-optimization algorithms will be applied. The MONROE platform will allow to perform experiments in real conditions. However, in order to carry out the self-optimization experiments, access to the Operation, Administration & Management (OAM) of the cellular network is required in order to change network configuration parameters. To cope with this requirement, eSON also proposes a HW extension to the MONROE platform that adds an experimental LTE network to the platform. This extension will permit the execution of the current proposal, and it will also significantly increase the scope of the possible MONROE experiments from terminal-end only to full end-to-end. With this addition, experimenters can measure and control the behavior of the RAN network, core and even remote services. The proposed addition is an LTE small-cell network with 12 picocells and 14 terminals that also includes a co-located WiFi network. The installation includes the core network and management HW. On the software side, a management platform is installed in the network, with plans to integrate it into the MONROE platform. eSON complements the capabilities of the MONROE platform of performing experiments in real conditions with new capabilities in controlled scenarios. In addition, it will allow to study not only the end nodes, but also the effects of the RAN and core networks in the E2E performance.

12. **"Dynamic Pricing in HetNets: DAPHNE"**

    Main goal: Scientific Excellence.
    Participants: University of Thessaly (UTH).
    Contact: korakis@uth.gr

    Contemporary mobile devices are equipped with multiple radio access capabilities. In parallel, mobile network providers densify their deployments in urban areas, aiming to provide higher network capacity, improved QoS and enhanced end-user perceived QoE. In such environments, the role of Mobile Broadband Networks (2G/3G/4G and beyond) and WiFi technologies (IEEE 802.11 a/g/n/ac) is dominant. Nevertheless, in such highly heterogeneous environments several questions may arise, regarding which technology or combination of technologies should the end-user use and when it should handoff to another available in-range technology. Moreover, such decisions may widely affect the performance of other users operating inside the same cell. Towards addressing these questions, we propose our experiment for DynAmic Pricing in HetNEts (DAPHNE). DAPHNE will develop a framework based on pricing methods (Flat Rate Pricing, Paris Metro Pricing with dynamic pricing, Paris Metro Pricing with dynamic pricing and multiple traffic classes per UE) that will dynamically assign prices for using each network. Given the remaining cells capacities and the end-user demands, DAPHNE will appropriately let all the end clients to be distributed among all the technologies that are available inside the Heterogeneous Network (HetNet) environment. DAPHNE will be evaluated over the MONROE facilities, taking advantage of the rich MBB experimental ecosystem that they provide, as a means to experimentally derive a framework that can be directly applied to future 5G networks.

13. **"Traffic and Data Offloading in Mobile Networks: TTOff"**

    Main goal: Scientific Excellence.
    Participants: University Politehnica of Bucharest, Romania.
    Contact: ciprian.dobre@cs.pub.ro

    The growing consumption of data by mobile devices has put a strain on mobile networks. With the increasing number of Wi-Fi-capable nodes, traffic offloading between mobile broadband (MBB) and Wi-Fi has been gaining steady traction. End-users offload traffic for cost control and better throughput, while operators do it for congestion and traffic load control. However, offloading should provide users with a seamless experience while they use applications on their devices, which should make smart decisions about keeping data flows on preferred networks, while optimizing resources to improve user experience. Many solutions send all traffic to Wi-Fi when available, few looking for other metrics such as mobility-related, and quality of service (QoS) and quality of experience (QoE) parameters. Our goal is to develop mechanisms to determine the best possible use of available networks, by intelligently switching traffic between MBB and Wi-Fi (over access points), or opportunistically through devices in proximity. In this context, to carry out experiments on a large-scale measurement platform is a very attractive opportunity to mitigate technological risks associated with traffic offloading. In the experiment, we will measure and evaluate key performance indicators (KPIs) of providing offloading through MONROE. We aim to implement a software extension to MONROE, to understand the impact of mobility (derived from context information collected by mobile nodes, ranging from throughput to location or contact history) over the offloading decision, and analyze the impact of offloading over real-life situations, by observing traffic patterns, broadband and Wi-Fi network behavior (in terms of throughput, congestion, etc.). The results will help us to propose and implement solutions for offloading, that will be evaluated over the MONROE testbed, with results proving valuable to network operators and application developers. Furthermore, the integration of MONROE and TTOff can contribute to sustaining the MONROE platform by engaging future experiments.

14. **"Characterising Mobile Content Networks in the Wild: CaMCoW"**

    Main goal: Scientific Excellence.
    Participants: Queen Mary University of London (QMUL).
    Contact: gareth.tyson@qmul.ac.uk

    Mobile broadband (MBB) networks have seen dramatic increases in both capacity and uptake. Characterising user-facing network performance, however, can be challenging because it depends on the particular services being accessed by the subscriber. For example, a user accessing YouTube will receive different performance based on the provisioning and network distance between the access network and the nearest YouTube server. This is not simple — it is a product of both the MBB network and YouTube's interconnection and capacity planning strategies. Little substantive work exists on exploring the behaviour of user-facing services in MBB networks. This deficiency is particularly critical in MBB networks due to their relative dynamism, and less developed (IXP) peering models (compared to wireline infrastructures). CaMCoW will address this gap. It will exploit the MONROE testbed to measure the ways in which popular content services (e.g., Netflix, Google) have approached deploying and interconnecting their infrastructures for access by MBB subscribers. It will measure the deployment of these services and how MBB users are given access to them. We will correlate these deployment strategies with the Quality of Experience users can expect, to formulate and inform best practice. We will integrate data generated via CaMCoW with other (wireline) measurement studies we are performing in this area to gain a comprehensive view of wired and wireless behaviour. Through this, CaMCoW will:

    (a) Explain how content services are currently provisioned and interconnected for access by MBB networks;
    (b) Provide data and analysis to describe the performance of each services and how it relates to their given deployment strategies;
    (c) Offer recommendations for improving user Quality of Experience when accessing these services from MBB networks;
    (d) Generate detailed feedback for the MONROE consortium, e.g., bug fixing, improving accessibility for new users, streamlining measurement data collection, techniques for better automation of experiments.

15. **"Feasibility study of latency-critical connected vehicle applications in MONROE: FELICIA"**

    Main goal: Scientific Excellence.
    Participants: SICS Swedish ICT.
    Contact: henrik@sics.se

    Next generation connected vehicles and cooperative intelligent transport systems (C-ITS) have great potential to make road traffic safer, more efficient, greener, and available to more people. However, these systems also place extensive demands on the mobile network infrastructure. The mobile networks need to handle me-critical applications combined with uploads of large data volumes from moving vehicles. We propose a measurement study to identify possibilities and limitations in the existing mobile networks for supporting C-ITS applications. We will also study the possible gain of using multiple accesses and multi-path transport in the end-points for these type of applications, separating me-critical and bulk data into different paths. The study will be done through measurement experiments where we emulate traffic representing a range of vehicular applications with different requirements on latency and capacity, and measure their performance. The traffic models are developed in dialogue with automotive industry representatives. The MONROE platform gives great possibilities for conducting these vehicular measurements and experiments in a controlled, repeatable and vendor- and operator-independent manner, and at large scale in heterogeneous environments. The platform also supports multi-path experiments and enables comparative studies between operators and between different locations that otherwise would be difficult to achieve. We see our proposed experiment as the first step in making the MONROE platform an important experimental platform for connected vehicle applications, and so of great interest for the automotive industry, telecom operators and researchers in this area. This will contribute to the sustainability of the MONROE platform.

4 User workshops

The first MONROE User Workshop was held at Oslo during June 13-14th, 2016, in order to meet our external users from the first open call and introducte them to the MONROE platform. It included introduction sessions on general aspects of the platform and a set of parallel sessions on administrative and deeper technical issues. Table 3 details the activities carried out during the workshop.

The second MONROE User Workshop was held at Madrid during October 25-26th, 2016, and was organized as two half-days. It included technical sessions on advanced concrete aspects of the platform such as multiple interface binding and metadata processing, a set of parallel sessions on administrative issues and a special session for users interested in video streaming experiments. Table 4 details the activities carried out during the workshop.

The third MONROE User Workshop will be held at Pisa on March 30th, 2017. At the workshop we will meet our external users from the second open call and introduce them to the MONROE platform.

Table 3: First MONROE User Workshop agenda.

Time	Topic	Presenters
Monday, June 13th 2016		
9:00	Welcome and introductions	Anna Brunstrom, Özgü Alay
9:15	Overview of the MONROE project	Özgü Alay
9:30	Overview of the MONROE system	Vincenzo Mancuso
10:15	MONROE hardware	Thomas Hirsch
10:45	(Coffee break)	
11:00	Scheduling and User Access	Thomas Hirsch, Miguel Peón Quirós
11:30	MONROE Metadata and Visualization	Miguel Peón Quirós
12:00	Containers and MONROE Experiments	Stefan Alfredsson, Jonas Karlsson
12:30	(Lunch)	
13:30	Short introductions of accepted proposals	10 min + 5 min questions each
15:00	(Coffee break)	
15:15	Short introductions of accepted proposals	(cont)
16:45	Administrative issues	Özgü Alay, Renata Almeida, Anna Brunstrom
17:30	Wrap up	
18:00	Social Event/Dinner	
Tuesday, June 14th 2016		
9:00	Passive measurements and TStat	Marco Mellia
9:45	Hands-on work with the platform / Meetings with patrons	
10:45	(Coffee break)	
11:00	Hands-on work with the platform / Meetings with patrons (continued)	
12:30	(Lunch)	
13:30	Hands-on work with the platform / Meetings with patrons (continued)	
14:30	Wrap-up and planning forward	
15:00	(Official part of workshop closes)	
15:00 - 17:00	MONROE partners available for further discussions & questions as needed	

Table 4: Second MONROE User Workshop agenda.

Time	Topic	Presenters
Tuesday, October 25th 2016		
14:30	Welcome	Vincenzo Mancuso
14:40	Admin outlook	Özgü Alay
15:00	Discussion and clarifications on feedback reports	Anna Brunstrom, Miguel Peón Quirós
15:45	Extra nodes for experimenters: Status and discussion	Roberto Monno
16:00	(Coffee break)	
16:15	Tstat Presentation	Ali Safari Khatouni
16:30	Technical presentations	External users
18:00	Group discussion and coordination of experiments	All
18:30	End of meeting	
20:30	Social dinner at Madrid	(Self-funded)
Wednesday, October 26th 2016		
9:30	Nodes and containers (certification, deployment and other logistics)	Stefan Alfredsson
11:00	(Coffee break)	
11:15 (A)	Interactive coding session with users Topics: How to improve container design, common packages/libraries to be added to monroe/base, metadata issues, DB issues, networking issues, etc.	Stefan Alfredsson, Kristian Evensen, Miguel Peón Quirós
11:15 (B)	Patrons and project coordinator/manager will be available to talk to experiment leaders	Özgü Alay, Anna Brunstrom, Vincenzo Mancuso
13:30	(Lunch)	
14:30	End of workshop	

5 Initial experiences and feedback from the projects in the First Open Call

To gain input on the MONROE platform and its use, all projects selected for the first open call were asked to send in a feedback report to the consortium (see Appendix A). During the second User Workshop, the MONROE consortium gave a report on the feedback received from the projects. Figure 1 summarizes this feedback. Below, we present for each question topic a list of common issues raised by the project members in their reports and the corrective actions proposed by the MONROE consortium or agreed upon during the workshop.

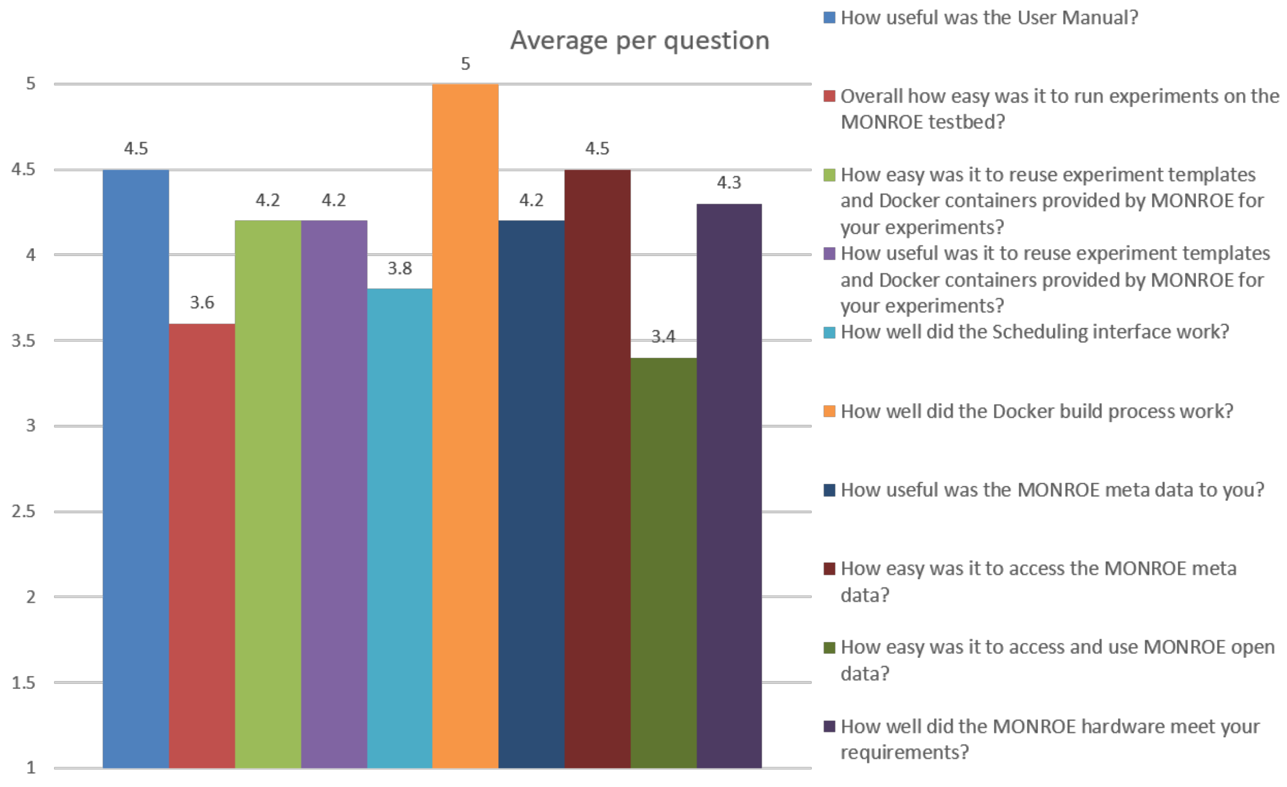


Figure 1: Average score per question in the feedback from the first open call users.

5.1 Question topic: "Most frequently used MONROE tools."

The most frequently used MONROE tools as reported by users were:

- User manual.
- Scheduler.
- Test nodes.
- Experiment templates.
- MONROE mailing list.

5.2 Question topic: "How useful was the user manual?"

Average score: $4.5(5, 5, 4.5, 4, 4.5, 3, 5, 4, 5, 4, 5, 5)$.

Issue	Proposed action
Missing info on how to activate SSL certificates.	New project members have to send the SSL ID of their certificate to a MONROE administrator. This information was already included some time ago. Improve dissemination of new versions, notifying the list when a new manual version is released. Suggest users to add a "watch" for the corresponding github repository.
Outdated procedures.	Try to keep them more updated.
Lack of a complete step-by-step example plus perhaps a multimedia (video) tutorial.	Add a complete step-by-step working example to the user manual, from container creation to results retrieval.
Need for more information to improve container building.	To be done.
Improve explanations on networking inside the containers.	To be done.
Improve explanations on metadata usage.	Add to the manual procedures to access and correlate metadata.
Inconsistent metadata descriptions.	Fixed.
"The manual says" it is better to transfer the final output files only when the task has already ended, so a continuous synchronization of this directory is avoided.	Clarify: Intermediate results are encouraged, better to have multiple files that are synchronized as the experiment progresses. The intention is to avoid updating already transferred files.

5.3 Question topic: "Overall, how easy was it to run experiments on the MONROE testbed?"

Average score: $4.3(4, 4.5, 4, 3, 5, 2, 4, 4, 4, 3, 3, 3)$.

Issue	Proposed action
Experiments get scheduled, but never advance from the "defined" state (they never run) without any explanation to why this happens.	Improve granularity of experiment states and their explanation: • Change 'defined' to 'not executed' when scheduling period is over. • Change 'defined' to 'requested' when the experiment data is first requested by the node. • Provide an explanation of the status codes in the user interface.
Improvements to the user interface.	• Implement a way to reschedule the last experiment run from the 'Status' section. • Improve ergonomy of the 'Status' page: Inline experiment status into the experiments list and show only when the user clicks on a experiment (done). Add pagination. • Implement the filters shown in the user interface (done). • Change recurrence ending date for number of repetitions. Evaluate if useful and implement if so. • Add saving and reloading of schedule options. Also, recovering scheduling options for a past experiment. • Show submission parameters and selected nodes for every experiment on the 'Status' page.
Lack of a programmatic method to schedule/consult/retrieve data.	Provide access to the backend API to bypass the web interface and access the scheduler programmatically.
Unclear error codes shown in the user interface may lead to false alarms when debugging the code, while the actual problem might be caused by the nodes themselves instead of by the experiment code.	Clarify: All status codes are node/scheduling issues — the experiment outcome itself is unknown by the scheduling system.
Offer a method to retrieve results directly via URL.	Explain how to use the results URL.
Not working with Firefox.	True, known bug. Suggestion to use Google Chrome as the browser to access MONROE.

5.4 Question topic: "How easy was it to reuse experiment templates and Docker containers provided by MONROE for your experiments?"

Average score: 4.2(5, 4, 4, 5, 5, 3, 5, 5, 3, 4, 4, 3).

Issue	Proposed action
Additional example templates would be useful (including more basic examples and more advanced applications).	
Need for examples on how to bind to a specific interface.	This topic was specifically addressed during the second workshop.
Availability of experimenter-local nodes for experimentation and development would accelerate the development process.	Develop SSH access to the containers running in testing nodes (done), development of MONROE virtual nodes. Enabling local access through SSH or serial cable to the development nodes of the projects (done).
Some templates are outdated and do not work "out of the box."	Fixed.

5.5 Question topic: "How useful was it to reuse experiment templates and Docker containers provided by MONROE for your experiments?"

Average score: 4.2(3, 3, 4.5, 5, 5, 2, 5, 5, 5, 5, 4, 4).

Issue	Proposed action
Usefulness depended on how well they covered the target use case, development language, etc.	Try to create more general examples.
Need for better examples on how to manipulate network interfaces and routes.	This topic was specifically addressed during the second workshop.
Hard to understand and debug what is happening once deployed on a node (non-interactive execution).	(See actions for previous question)
Not too useful to reuse as they had to be fixed or did not work as expected.	Ensure that they work as expected.
Need for better explanations on how to build an optimized docker container.	Address this in the user manual.

5.6 Question topic: "How well did the scheduling interface work?"

Average score: $3.8(4, 4.5, 3.5, 2, 4, 2, 5, 4, 4, 3, 4, 5)$.

Issue	Proposed action
Some nodes were not operational but the experiment was automatically scheduled on them.	Scheduler now selects only nodes that have sent a heartbeat in the last 10 min, if the experiment is scheduled within the next 20 min; sent a heartbeat in the last 24 h, otherwise. Explained during the second workshop that the scheduler may still hide some nodes the user is looking for if they have issues, and it may not entirely be able to prevent selecting a node that is afterwards unable to schedule.
Lack of feedback on scheduling status.	Implemented more comprehensive status codes to detail when an experiment has been requested by the node.
Implement container verification procedures.	A way to improve the scheduling interface would be to add a verification step for the Docker container. Maybe also decouple the testing process from certification, so that users can, at any time, send a container for verification to the certification server. Even if no certificate is issued, the server could confirm that a) the container executes and b), the deployment size on that node.
Container quota issues.	Study adding a way to warn the user when the container size exceeds the storage quota allocated. This requires size estimation of the container, which may be provided by the certification/testing procedures above.
The delete button appears to only work if an experiment finished successfully.	True. Otherwise, they are considered "aborted" and do not disappear. Added filters to hide/show completed and canceled experiments.
All three interfaces are selected by default when scheduling a new experiment.	All interfaces are always available for the containers. Remove this setting from user interface to avoid confusion.
The scheduler does not seem to take into account the whole time needed to "build" the Docker container of the experiment and only offers a naïve estimation.	Clarify: Docker containers are not "built," only downloaded and deployed. The download time is not easy to estimate, and the issue becomes worse when the size grows, and when the container has fewer layers. Discussion during the second workshop: We could implement a two-step approach (deployment/run) with a longer term reservation of download space — maybe only for certified experiments. This requires us to reserve disk space to pre-deploy, and to implement the altered scheduling process. We agree to only consider this if large experiments are unavoidable. We can also pre-deploy manually on selected nodes, e.g., "pinning" a container in some nodes for a few days.
Handling unused quota of the experiment.	Work in progress.

5.7 Question topic: "How well did the Docker build process work?"

Average score: 5(5,4.5,5,5,5,5,5,5,5,5,5,5,5).

Issue	Proposed action
Container size and quota issues.	See actions for previous question.
Include option to clean out old files.	The `makefile` for the containers will be updated.
Reduce building effort.	One of the external users provided a build script to reduce typing effort at: https://github.com/Incelligent/monroe-explorer/blob/master/build.sh

5.8 Question topic: "How useful was the MONROE metadata to you?"

Average score: 4.2(5,3,4.5,N/A,4.5,2,5,4,4,4,5,5).

Issue	Proposed action
Issues with frequency-accuracy tradeoffs.	Clarify when each metadata item is generated.
Suggestions for filtering some fields or generate on request.	Clarify and implement centrally if relevant. From the RICERCANDO project: "All the users need to preprocess and align the data from the DB. Eliminate Cassandra redundancies. Do it centralized to avoid repetitive effort for users." Clarify which procedures they had to do and implement them if needed. Maybe this is easy to do while generating the daily CSV dumps.
Redundancies in Cassandra data (e.g., modem vs. connectivity tables).	Drop the connectivity table (done).
Granularity of time stamps, GPS clock with TS.	
Not clear how to obtain the metadata from Tstat.	Explain that files are mounted at `/monroe/tstat`. Added full section to the user manual.

5.9 Question topic: "How easy was it to access the MONROE metadata?"

Average score: 4.5$(5, 4, 4.5, N/A, 4, 2, 5, 5, 5, 5, 5, 5)$.

Issue	Proposed action
Updating user manual with some examples would be helpful.	Add to the manual examples on correlating metadata.
Provide all the metadata log to users during the experiment (not only the filtered messages that are transmitted to the MONROE databases).	Study feasibility and convenience.
Tables `META.DEVICE.MODEM` and `META.CONNECTIVITY` have different values as described in https://github.com/MONROE-PROJECT/data-exporter.	Clarified that the values are consistent; they are respectively zero and one-based, as explained in the documentation.

5.10 Question topic: "How easy was it to access and use MONROE open data?"

Average score: 3.4$(N/A, 5, N/A, N/A, 4.5, N/A, N/A, 1, N/A, 3, N/A, N/A,)$.

Issue	Proposed action
No clear access method.	Explanation: Open data is not yet publicly released. We want it to be more mature. Therefore, up to now it has been delivered only upon request. Additionally, it is currently delivered as CSV dumps and access to a Cassandra database replica.
Suggestion to provide a separate, cleaned and preprocessed dataset available for offline analysis.	(See above)
Suggestion to perform centrally some alignment and cleanup operations for online analysis.	(See above)

5.11 Question topic: "How well did the MONROE hardware meet your requirements?"

Average score: $4.3(5, 4, 4, 5, 4.5, 3, 5, 5, 4, N/A, 3, N/A$.

Issue	Proposed action
Network interfaces are not always all available at the same time.	We are working on the stability of the nodes/interfaces. The second design phase of the MONROE nodes is specifically aimed towards this goal.
Doubts concerning the purpose of testing vs. deployment nodes.	Clarified during the second workshop. Address it in the user manual.
Suggestions for fine-granular metadata. SDR extensions?	To be evaluated.
Support for multi-container docker applications?	Discarded as a similar effect can be obtained by running several processes in the container.
HW can be a bottleneck for high-speed traffic generation together with cross traffic generation?	Evaluate.
Include voice equipment.	Evaluate

6 Conclusions

This report has described the contents of the user manual, which MONROE users can access freely at the project public git repository. The user manual is a live document subject to continuous improvements, particularly to address user suggestions and questions.

The selection procedure for the two Open Calls has also been explained, detailing the reviewers that took part in each of them. Additionally, a summary of the accepted proposals in both calls is also provided for reference. Users selected during the first open call have had the chance to join the first and second user workshops, help at Oslo and Madrid, respectively. Both workshops allowed us to exchange important insights with the platform users and discuss about many details of the day-to-day experience with it. During the second workshop, experimenters from the many participating groups had also the chance to discuss among them in several parallel sessions.

Finally, this document analyzes the action points that the consortium has taken or plans to take to address the different issues or doubts that the users found while using the platform. Some of them have already been addressed, whereas others will be subject to further evaluation in the future.

A First Feedback Report

Name of your project:	
Your organization(s) name(s):	
Name of contact person for report:	
Contact person telephone number:	
Contact person email:	

Part A. Project Summary Please provide a summary of the work carried out so far (1–2 pages).

Part B. Detailed description This section describes the details on the experiment/extension. It will be required for the final report.

Part C. Feedback to MONROE This section contains valuable information for the MONROE consortium and describes the associated partner's experiences while performing the experiment or while implementing the extension staring from the available software and hardware.

C.1 Resources & tools used Please list the MONROE resources and tools you have used so far and how they were used.

C.2 Feedback on experimenting in / implementing extensions within MONROE
Feedback on a number of features of MONROE is requested below. When rating the usefulness, easiness etc., please use 5 as the highest score for all questions (indicating that a feature was very useful, very easy to use, etc.). If you have no experience with a particular feature asked about, please indicate this in the open ended part of the question. Please do try to test all parts of the system as far as possible though.

How useful was the User Manual? (Rate on scale from 1-5):
Please provide any corrections / suggestions for improvement of the User Manual.

Overall how easy was it to run experiments on the MONROE testbed? (Rate on scale from 1-5):
Please provide any comments / suggestions for improvement of the overall process.

How easy was it to reuse experiment templates and Docker containers provided by MONROE for your experiments? (Rate on scale from 1-5):
Please provide any comments / suggestions for improvement of the usability of MONROE EaaS features.

How useful was it to reuse experiment templates and Docker containers provided by MONROE for your

experiments? (Rate on scale from 1-5):
Please provide any comments / suggestions for making MONROE EaaS features closer to experimenter's needs.

How well did the Scheduling interface work? (Rate on scale from 1-5):
Please provide any comments / suggestions for improvement of the scheduling interface.

How well did the Docker build process work? (Rate on scale from 1-5):
Please provide any comments / suggestions for improvement of the Docker build process.

How useful was the MONROE meta data to you? (Rate on scale from 1-5):
Please provide any comments / suggestions for improvement in the meta data provided.

How easy was it to access the MONROE meta data (Rate on scale from 1-5):
Please provide any comments / suggestions for improvement in how meta data is provided.

How easy was it to access and use MONROE open data (Rate on scale from 1-5):
Please provide any comments / suggestions for improvement in how open data is provided.

How well did the MONROE hardware meet your requirements (Rate on scale from 1-5):
Please provide any comments / suggestions for improving the hardware.

C.3 Other suggestions Please provide any other comments or suggestions you may have on how to improve the MONROE hardware or software, how to improve the interaction between MONROE and its external users or how to improve any other aspects of MONROE.

MONROE

Measuring Mobile Broadband Networks in Europe

H2020-ICT-11-2014
Project number: 644399

Deliverable User manual
MONROE Platform User Manual

Editor(s): Miguel Peón-Quirós, Özgü Alay, Vincenzo Mancuso
Contributor(s): Miguel Peón-Quirós, Thomas Hirsch, Ali Safari Khatouni

Work Package:	5.2 / User Support
Revision:	1.0
Date:	February 28, 2017
Deliverable type:	R (Report)
Dissemination level:	Public

European Commission
Horizon 2020
European Union funding
for Research & Innovation

1 Introduction

The purpose of this document is to guide MONROE experimenters through the process of creating, running and monitoring their experiments, and the subsequent collection of results. It first explains how the experiments must be prepared inside Docker containers, the testing process they must undergo before they can be deployed into MONROE's nodes, and how they must be uploaded to a repository for deployment into the nodes. Then, it explains the basics of the web interface that allows provision of resources and the scheduling of experiment executions. Finally, it shows how the experiment results can be retrieved either directly from the experiment itself or from the repository provided by MONROE.

1.1 Overview of the node configuration

MONROE nodes have been designed to have minimal impact on the experiments that run on them. Therefore, only one experiment can run at a given time in a node. Although the experiments are executed inside a Docker container, they have no quotas on CPU or memory usage, subject only to available node resources. Container image size and temporary storage in the node may be restricted, though.

Every MONROE node runs, in addition to user experiments, the following background processes:

- The experiment scheduler, which arbitrates the execution of user experiments in the node. The scheduler runs permanently in the background and contacts periodically the scheduling server, sending "heartbeats" and checking for new schedules for the node. When an experiment is not running, the scheduler may start the deployment of the containers for one or several experiments scheduled to be run in the immediate future, so that they are prepared on advance. The scheduler checks the duration of the slot assigned to an experiment; if the experiment does not finish on time, it stops the whole container.

- Synchronization (rsync) services to copy data files to the MONROE repository. This service copies user experiment results, the data collected by passive experiments and assorted metadata measurements. It runs continuously, transferring files to the server as they appear in the corresponding folders. This service uses the management interface, which is different from the interfaces available for the experiments. However, the management interface may share in some cases the same subscriber contract with one of the experiment interfaces; operators might restrict the total bandwidth available for all the SIMs linked to the same contract. Additionally, two modems (management plus experiment) using the same operator antenna may somehow affect the bandwidth available for the experiment. Therefore, experimenters should be aware of the small amount of data that can be transferred by this service in parallel to their experiments.

- Several systems run continuously in the background gathering information on the status of diverse components. Examples include a service to read the signal strength and network configuration of each of the experiment modems, the GPS data and various node parameters such as CPU load, memory usage or CPU temperature. These services run continuously in the background with a frequency that varies from one second up to several minutes. Although their impact on user experiments should be minimal, their existence must be known by the experimenters.

- In addition to the services that gather metadata, MONROE nodes keep several containers active all the time. These containers run experiments that are deemed basic for the MONROE platform and include:

  - A ping experiment. Container number 1 executes continuously an ICMP ping operation to a fixed external server (currently, Google's DNS at 8.8.8.8). The RTT values are collected and transferred to the servers. The ping experiment runs continuously with a frequency of one second, for every interface.
  - A container that runs Tstat, the passive mPlane monitoring probe that collects, for each interface, detailed flow level statistics. The Tstat container generates no traffic; flow level data is synchronized to the MONROE repository using the standard synchronization process described above.

- Finally, some built-in MONROE experiments run as scheduled containers. These experiments will not run at the same time than user experiments:
  - A bandwidth measurement test, which periodically downloads an object using the HTTP protocol to measure the achievable bandwidth. The test runs on each interface. The periodicity of this experiment and whether it can be run while user experiments are being executed are yet to be decided.
  - A container that periodically executes a full paris-traceroute to several popular websites and records information about all the intermediate hops. This container will in principle be run several times per day, but the interactions with user experiments are yet to be determined.

1.2 Overview of the experimental workflow

Experiments conducted in the MONROE platform follow the workflow shown in Figure 1, which consists of three phases: Experiment design, testing and experimentation. During the experiment design phase, the experiment goals and properties are defined and the container required to deploy it in MONROE nodes is configured. During the testing phase, the container is executed on nodes specifically devoted to testing new experiments. If the experiment passes all the safety and behavior tests, a MONROE manager will digitally sign the container image. Signed containers cannot be further modified without running again through the testing phase. Finaly, the experimenter is free to schedule the experiment container on any nodes, subject to the specific quotas assigned to their project.

2 Experiment preparation

2.1 General experiment notes

MONROE experiments run under the root user of a Docker container. Therefore, experimenters can design any kind of experiment within the security restrictions of the platform, including the configuration of routing tables, stopping or starting interfaces and executing any kind of applications. We assume the reader is familiar with the Docker technology. Otherwise, we suggest getting used to it by accessing the documentation at `https://docs.docker.com/engine/understanding-docker/`.

Creating and using containers is a two-step process. At design time, the experimenters create the image for the container in their local machine using a container-creation script. If necessary, they can install new packages (e.g., via apt-get) or copy libraries. The docker tools read the script and create the final image for the experiment, which will then have to be uploaded to a repository. At run-time, the nodes retrieve the container image from the repository and start it as scheduled.

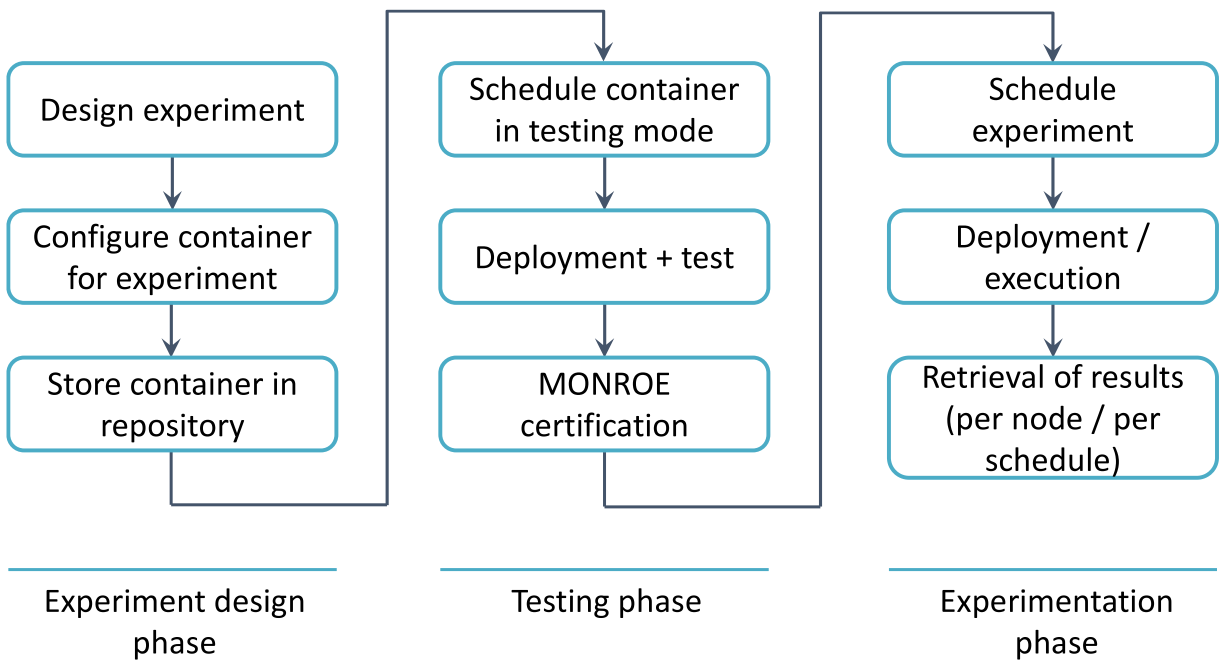


Figure 1: Experimental workflow.

During execution, the experiment should not install additional applications or download any data that is not part of the experiment itself (e.g., if the experiment uploads a file to a server to test upstream speed, either include the file to be uploaded in the container at design time or create it locally).

⇒ *Experiments will under no circumstances allow direct ssh access to the node or any other form of running interactive commands from outside the container that can pose a security risk for the platform.* ⇐

2.2 Container preparation

MONROE experiments are deployed in Docker containers (https://www.docker.com/). Preparing a new container from MONROE's base image is an easy process:

1. Install Docker in your machine. Do it preferably downloading the installation script from the web page, rather than through a package manager such as apt-get:

```
$ wget https://get.docker.com -O install.sh
$ chmod u+x install.sh
$ ./install.sh
```

   You will have to run docker as root unless you add yourself to the docker group.

   Mac users: Download and install "Docker for MAC" (https://www.docker.com/products/docker#/mac) or the "Docker Toolbox" (https://docs.docker.com/toolbox/overview/), according to your OS version.

2. Test the Docker installation with the 'Hello world!' example:

```
$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
```

```
03f4658f8b78: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:xxxxxxxxxxxxxxxxxxxxxxxx
Status: Downloaded newer image for hello-world:latest
```

If everything has worked correctly up to here, you will see a welcome message similar to the following:

```
Hello from Docker.
This message shows that your installation appears to be working correctly.
...
```

You can check which images are locally installed with:

```
$sudo docker images
REPOSITORY    TAG      IMAGE ID       CREATED       SIZE
hello-world   latest   690ed74de00f   4 months ago   960 B
```

3. Now you are ready to download the MONROE base image:

   ```
   git clone https://github.com/MONROE-PROJECT/Experiments.git
   ```

   This will fetch the repository with MONROE's example containers.

4. Head to `Experiments/experiments/template/`. Here, you will find the required files to prepare your image based on MONROE's base. You should care about four things: a) The contents of the `files/` folder, b) the `build.sh` file, c) the `push.sh` file and d) the `template.docker` script file that describes how to create your container. In the directory `files/` you can put all the files that are part of your experiment. As a simple example, we can use the following script:

   ```
   $vi files/myscript.sh
     #!/bin/bash
     ls -lah > /monroe/results/listing.txt
   ```

   Any files that your experiment creates in `/monroe/results` are delivered to the repository, where you will be able to retrieve them. *Writes to any other part of the filesystem will be lost once the experiment is finished.* In periodic schedules, no data will survive from one execution to the next (i.e., the container is loaded fresh before each execution). If result persistence is needed, the experimenter will have to supply it by downloading the needed files from the network during the experiment itself.

5. You should not need to modify the `build.sh` file. The name of the container is the name of the current directory, and it must match the name of the `.docker` file (e.g., `template.docker` as we are in a folder named `template/`).

6. The file `template.docker` is the script used to build your container. You can modify it to:

   - Define the entry point of your experiment ("ENTRYPOINT").
   - Change the base image of the container, e.g., `monroe/base`.
   - Install additional packages or libraries.

   For example:

   ```
   FROM monroe/base

   MAINTAINER your-email-address
   ```

```
COPY files/* /opt/monroe/

#Default cmd to run.
ENTRYPOINT ["dumb-init", "--", "/bin/bash", "/opt/monroe/myscript.sh"]
```

This example will copy the files in the `files/` directory to the one you specify *inside* the docker container (e.g., `/opt/monroe`).

TIP: If you need to install additional packages in the container, be sure to clean any temporary files from the image. Also, notice that the Docker creation script analyses the contents of the container filesystem after every line in the .docker script is executed. That means that, even if you delete files at the end, Docker will create intermediate "layers" that will be downloaded and applied sequentially to build the final image of your container. Consider instead using one-liners such as the following:

```
    RUN apt-get update && apt-get install -y vim && apt-get clean
```

This will ensure that the files are deleted before Docker analyses the filesystem.

7. Modify the file `push.sh` to reflect the name of your repository:

```
#!/bin/bash
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"

CONTAINER=${DIR##*/}

CONTAINERTAG=myuser/myrepo # --> Modify to your own dockerhub user/repo

docker login && docker tag ${CONTAINER} ${CONTAINERTAG} && docker push ${CONTAINERTAG} && \
    echo "Finished uploading ${CONTAINERTAG}"
```

During the development phase of your experiment, follow these steps to make your container accessible for the testing nodes:

- Create an account at Docker Hub.
- Create a private repository (you can create one container as private; no limits for public ones).
- In your development machine, run: `docker login`. It will ask you for your credentials.

8. After populating the `files/` directory, modifying the .docker file and updating the `push.sh` file, you are ready to create the image:

```
$sudo ./build.sh
Using default tag: latest
latest: Pulling from monroe/base
Digest: sha256:6df1195a3cc3da2bfe70663157fddc42e174ec88761ead7c9a3af591e80ebbd5
Status: Image is up to date for monroe/base:latest
Sending build context to Docker daemon 11.26 kB
Step 1 : FROM monroe/base
---> d1b4f4baa60d
Step 2 : MAINTAINER mikepeon@imdea.org
---> Using cache
---> 0b05b5c453c7
Step 3 : COPY files/* /opt/monroe/
---> acc2df443070
Removing intermediate container 66a666516a27
Step 4 : ENTRYPOINT dumb-init -- /bin/bash /opt/monroe/myscript.sh
---> Running in f4b7a1ee804a
```

```
---> 096c7a56ff1c
Removing intermediate container f4b7a1ee804a
Successfully built 096c7a56ff1c
Finished building template
```

9. Test that your new docker container is available:

```
$sudo docker images
REPOSITORY                                 TAG    IMAGE ID     CREATED         SIZE
hello-world                                latest 690ed74de00f 4 months ago    960 B
your_docker_account/your_experiment        latest xxxxxxxxxxxx 32 seconds ago  626.6 MB
monroe/base                                latest xxxxxxxxxxxx 12 days ago     626.6 MB
```

   Exact image ids and sizes will vary.

10. Push the container image to the repository:

```
$ sudo ./push.sh
Username (your-Docker-user-name):
Password: (type your DockerHub password)
WARNING: login credentials saved in /home/your-username/.docker/config.json
Login Succeeded
The push refers to a repository [docker.io/mikepeon/template]
5f339bfdaae2: Pushed
486ab26686cc: Layer already exists
034f70c0d9cd: Layer already exists
86b5acd8772a: Layer already exists
f03317610243: Layer already exists
50f6c1bd7ce6: Layer already exists
aec5953bffa2: Layer already exists
507169b05eea: Layer already exists
5d799297d10c: Layer already exists
759d76df9ac7: Layer already exists
5f70bf18a086: Layer already exists
12e469267d21: Layer already exists
latest: digest: sha256:c855de65307191b4832b2ec60a4401c1b63424827c29149703c5d7ef07b519f7 size: 3001
Finished uploading your-username/template
```

11. You can now test that your image runs correctly, even on your own PC (if the experiment logic and resource demands allow for it).

```
$mkdir /run/shm/myresults
$sudo docker run -v /run/shm/myresults:/monroe/results your_docker_account/your_experiment
    --> The output of your experiment will be in /run/shm/myresults/listing.txt
```

   The docker command line allows you to specify a mapping between a directory inside the docker image and one in the host system. In this case, we have mapped `/monroe/results` from the container to `/run/shm/myresults`. This is useful if you are running the container locally in a normal PC for debugging purposes.

   IMPORTANT: This process shows how to build and run a container *locally* in your workstation. However, experimenters do not have direct access to the MONROE nodes. Therefore, to execute your experiment *in* a MONROE node, you will follow the process just up to the `sudo ./push.sh` step and then use the web interface to upload and schedule the container into the nodes.

You may check the contents of `experiments/*` for more useful examples.
The following is a list of useful common Docker commands:

- To list installed/built images (and get their ids):

  ```
  docker images
  ```

- To list running containers and get their tags:

  ```
  docker ps
  ```

- To stop running containers:

  ```
  docker kill container-tag
  ```

- To delete images:

  ```
  docker rm -f image_id
  ```

- To retrieve the latest version of an image (e.g., `monroe/base`):

  ```
  docker pull monroe/base
  ```

- To attach to a running container and get an interactive shell:

  ```
  docker exec -i -t container-tag bash
  ```

2.2.1 Package and tool installation

If you have to install extra packages, libraries or tools, do it from the `my_experiment.docker` file. You should never pull repositories or download libraries from inside your experiment as this will count against your data quota (and execution slot) for every instance of your experiment. Instead, modify the container configuration file as in the following example:

```
FROM monroe/base

MAINTAINER your-email-address

RUN apt-get update && apt-get install -y \
    python \
    python-pip \
    traceroute \
    && apt-get clean
RUN pip install pygame

RUN mkdir -p /opt/yourname
COPY files/* /opt/yourname/

#Default cmd to run
ENTRYPOINT ["dumb-init", "--", "/bin/bash", "/opt/yourname/myscript.sh"]
```

You may also download any files using `wget`, but you may simply put them in the `files/` folder as well. Remember, this happens during container creation on your PC, *not* during experiment execution on the nodes.

If you find the need for big libraries that you think should go into the base image, please contact MONROE's administrators.

TIP: The easiest way to find out which packages and versions are available in the MONROE base image is to create a simple container and run an interactive batch session inside it in your workstation. For example, assuming that you have a basic container that simply waits when run, you may follow the following steps:

```
mkdir /run/shm/myresults
docker run -v /run/shm/myresults:/monroe/results repository/your_container &
docker ps --> Look for the tag of your running container
docker exec -i -t container_tag bash
--> Here you are inside the container
dpkg -l > /monroe/results/package-listing.txt
exit
--> You'll find the output at /run/shm/myresults/package-listing.txt
```

For easier reference, Table 8 in Appendix A gives a detailed listing of packages available in `monroe/base` at the time of writing this text.

2.3 Optional interactive debugging in MONROE nodes

To make the process of debugging experiments in the nodes, a small number of development nodes will be supplied. Experimenters will be able to schedule their containers to one of these nodes (even before getting their container signed) and, once the container is started, connect to it through a reverse SSH tunnel. Then, they will be able to interactively modify their scripts or applications and execute them until the allocated slot expires.

The exact technical procedures and reservation policies are still to be defined by the MONROE consortium.

2.4 Mandatory testing process

Every experiment submitted to the MONROE testbed *must* first pass through a testing process to receive manual approval by a MONROE administrator. To submit your experiment for testing, you have to use the web interface specifying “testing” as the required node type.

2.5 Deployment

MONROE’s scheduling system will automatically deploy experiments to the nodes before their execution time. The nodes will fetch the container image from the Docker repository, and the size of the download will be accounted in your data quota. Notice that in the case of periodic experiments, each time an experiment is run, the Docker container may have to be re-downloaded and its costs will be accounted in your quota.

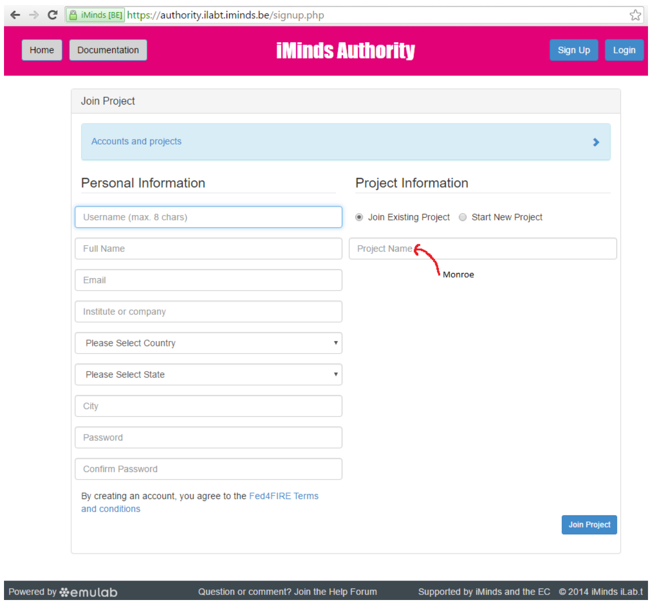
3 Resource allocation, and experiment scheduling and monitoring

Once a experiment is configured as a Docker container, it can be scheduled multiple times under different conditions using the user client web located at https://www.monroe-system.eu.

3.1 User login and certificates

User identification in MONROE is achieved through client certificates. Every experimenter has their own certificate compatible with the FED4FIRE[1] federation. User certificates are issued by iMinds through the following URL: https://authority.ilabt.iminds.be/. New users must create a new account (“sign up”). Be sure to select the option “Join Existing Project” and type the name “Monroe” in the project name

[1] http://www.fed4fire.eu/



https://authority.ilabt.iminds.be/signup.php

Home | Documentation | iMinds Authority | Sign Up | Login

Join Project

Accounts and projects

Personal Information

Username (max. 8 chars)
Full Name
Email
Institute or company
Please Select Country
Please Select State
City
Password
Confirm Password

By creating an account, you agree to the Fed4FIRE Terms and conditions

Project Information

Join Existing Project | Start New Project

Project Name

Monroe

Join Project

Powered by emulab | Question or comment? Join the Help Forum | Supported by iMinds and the EC © 2014 iMinds iLab.t

Figure 2: iMinds registration page to obtain FED4FIRE-compatible certificates for use with the MONROE platform.

field (Figure 2). The authorization process involves a manual verification step by one of the MONROE administrators, so it will probably take one or two days.

Please, notice that the current policy for MONROE is to use one user certificate per project, shared between all the experimenters belonging to that project.

Once the identity of the experimenter is approved, they will receive an informative email. They should then log into the iMinds webpage to download the certificate files (PKCS12). These files must be installed in the experimenter browser. After that, the user should be able to access the user web directly. Upon request of the main (index.html) file, the browser will contact MONROE servers to verify that the user credentials are correct. In the case of any problems, the user will be presented with instructions on how to obtain a certificate. If the client certificate is verified successfully, they will be automatically redirected to the listing of their experiments.

NOTE: User certificates are manually activated in the scheduling software. To use your certificate, please send its “ssl_id” to one of the MONROE administrators (e.g., `mailto:mikepeon@imdea.org`). You may find it in the screen after pressing the “Try me” button, once the certificate is correctly installed in your browser:

```
{
"fingerprint": "c79f1967aea17811a1ebed39b7d718430904338a",
"user": {
    "id": 3,
    "name": "MONROE Test admin",
    "quota_data": 50000000000,
    "quota_storage": 500000000,
    "quota_time": 500000000,
    "role": "admin",
    "ssl_id": "c79f1967aea17811a1ebed39b7d718430904338a"
},
"verified": "SUCCESS"
}
```

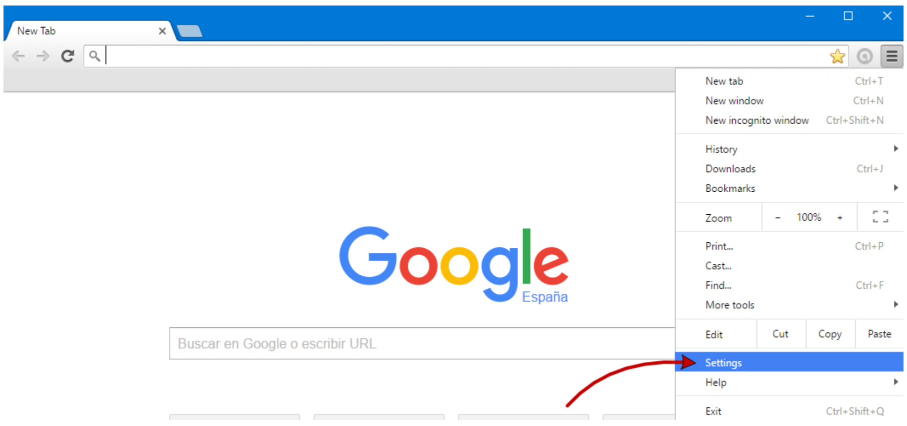
⇒ We have identified some common issues that are not yet solved. Below are some workarounds:

- For the first login, you may be asked for your user certificate and then your browser may show a security warning. This is due to the use of a self-signed server certificate. Please ask your browser to proceed. Then, you will probably see an error page from MONROE. Please, click the red button labeled “Try me” and check that you get a successful data output. Finally, please proceed again to the main page of the project. From that point, you should be able to access the system without further problems in future sessions. (Pointers on how to simplify this issue are welcome!)
- Firefox on OSX has an issue with CORS headers. Although the web and scheduling servers are running now on the same machine, you may still encounter this problem.

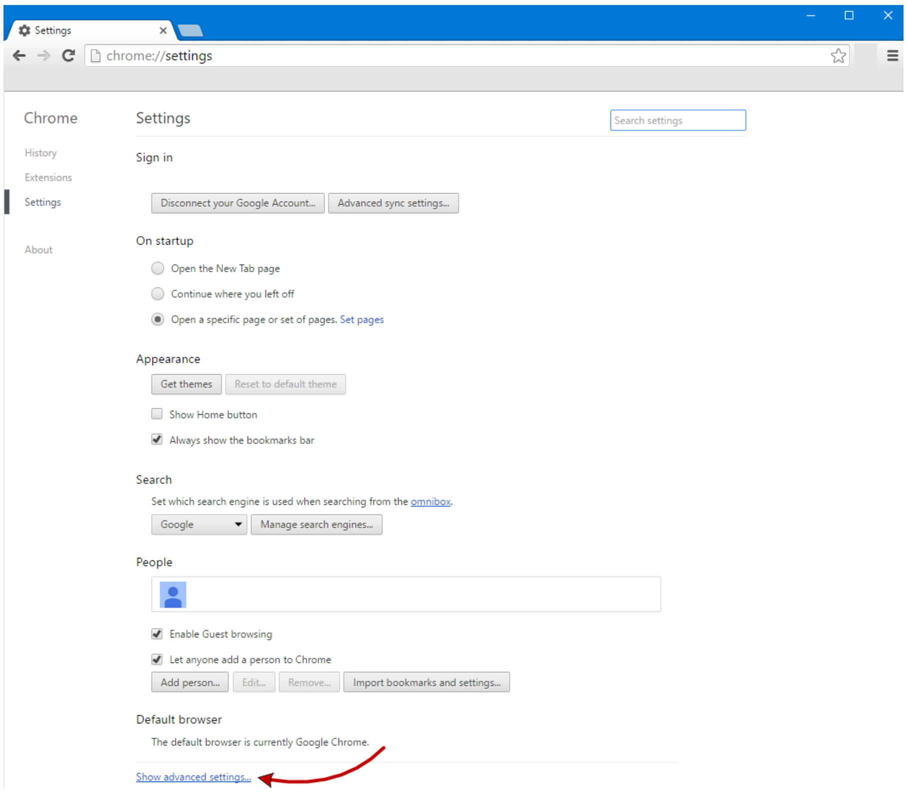
3.1.1 Installation of user certificates in Chrome

This section explains how to install the FED4FIRE-compatible user certificates used by the MONROE platform in Google Chrome for Windows. The procedure for other browsers and platforms should be similar.

1. Go to your browser settings page:



2. Display the advanced configuration settings:



Settings
chrome://settings

Chrome
History
Extensions
Settings
About

Settings

Search settings

Sign in

Disconnect your Google Account... Advanced sync settings...

On startup

- Open the New Tab page
- Continue where you left off
- Open a specific page or set of pages. Set pages

Appearance

Get themes Reset to default theme

- [ ] Show Home button
- [x] Always show the bookmarks bar

Search

Set which search engine is used when searching from the omnibox.

Google Manage search engines...

People

- [x] Enable Guest browsing
- [x] Let anyone add a person to Chrome

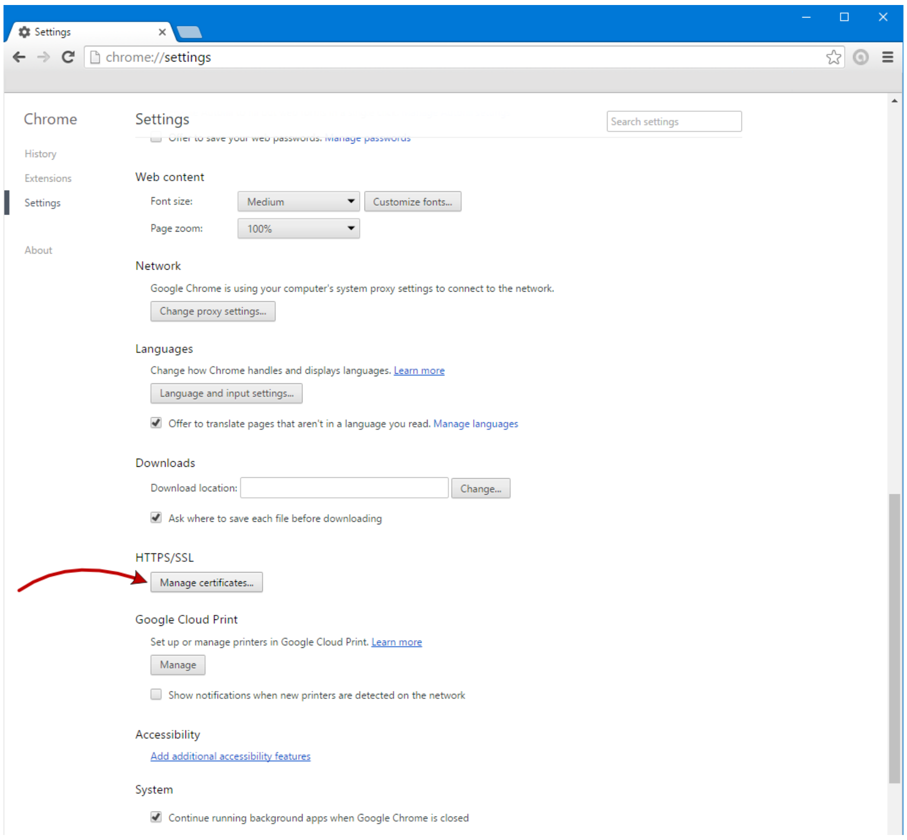
Add person... Edit... Remove... Import bookmarks and settings...

Default browser

The default browser is currently Google Chrome.

Show advanced settings...

3. Go to the section labeled "HTTPS/SSL" and click the button "Manage certificates...":



Settings
chrome://settings

Chrome
History
Extensions
Settings
About

Settings

Search settings

Offer to save your web passwords. Manage passwords

Web content

Font size: Medium Customize fonts...

Page zoom: 100%

Network

Google Chrome is using your computer's system proxy settings to connect to the network.

Change proxy settings...

Languages

Change how Chrome handles and displays languages. Learn more

Language and input settings...

- [x] Offer to translate pages that aren't in a language you read. Manage languages

Downloads

Download location: Change...

- [x] Ask where to save each file before downloading

HTTPS/SSL

Manage certificates...

Google Cloud Print

Set up or manage printers in Google Cloud Print. Learn more

Manage

- [ ] Show notifications when new printers are detected on the network

Accessibility

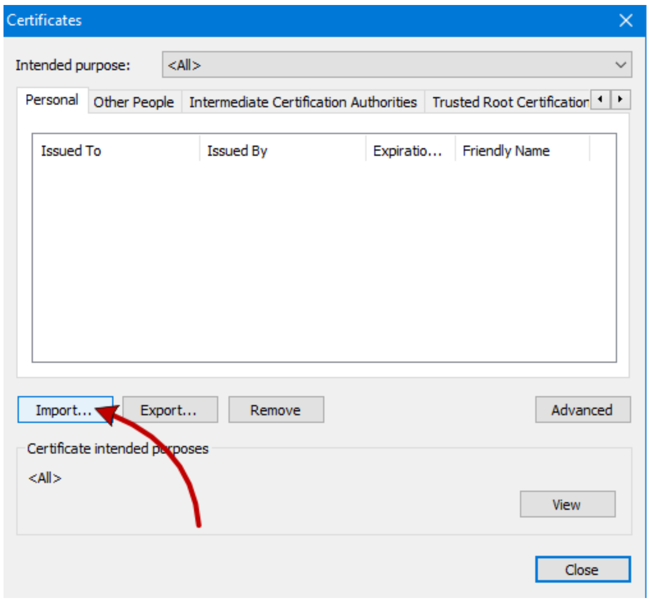
Add additional accessibility features

System

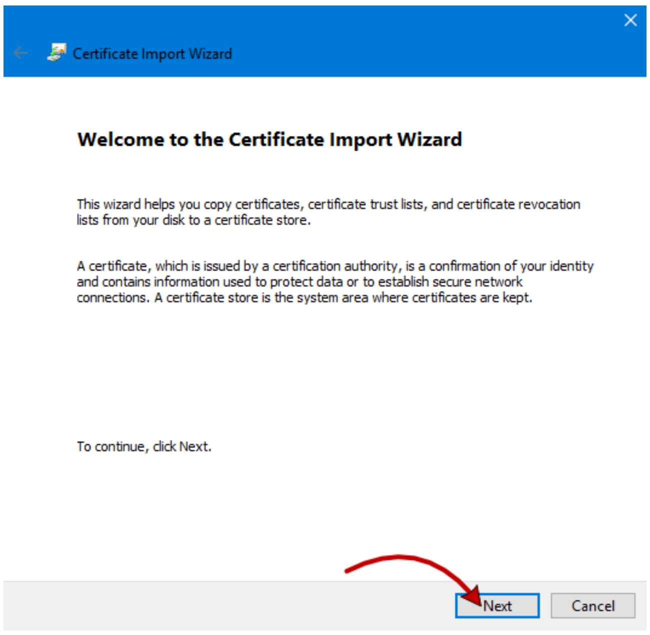
- [x] Continue running background apps when Google Chrome is closed

4. The dialog box for managing your certificates will be displayed. Press the button "Import..." to import

your certificate:



5. In the new dialog, click “Next”:



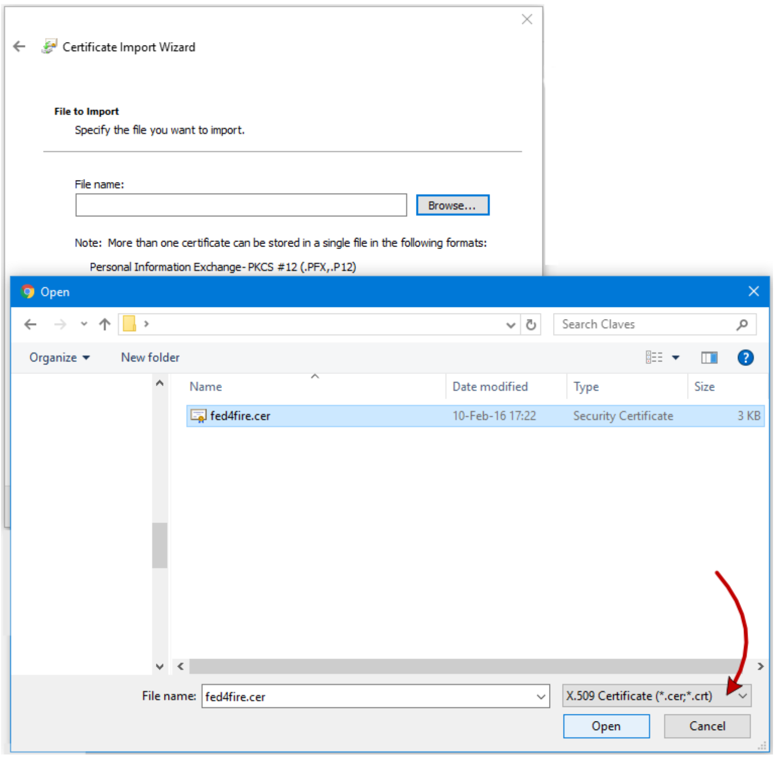
Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

6. In the file-selection dialog that appears next, change the file type from “X.509 Certificate (*,cer;*.crt)” to “Personal Information Exchange”:


Certificate Import Wizard

File to Import
Specify the file you want to import.

File name:

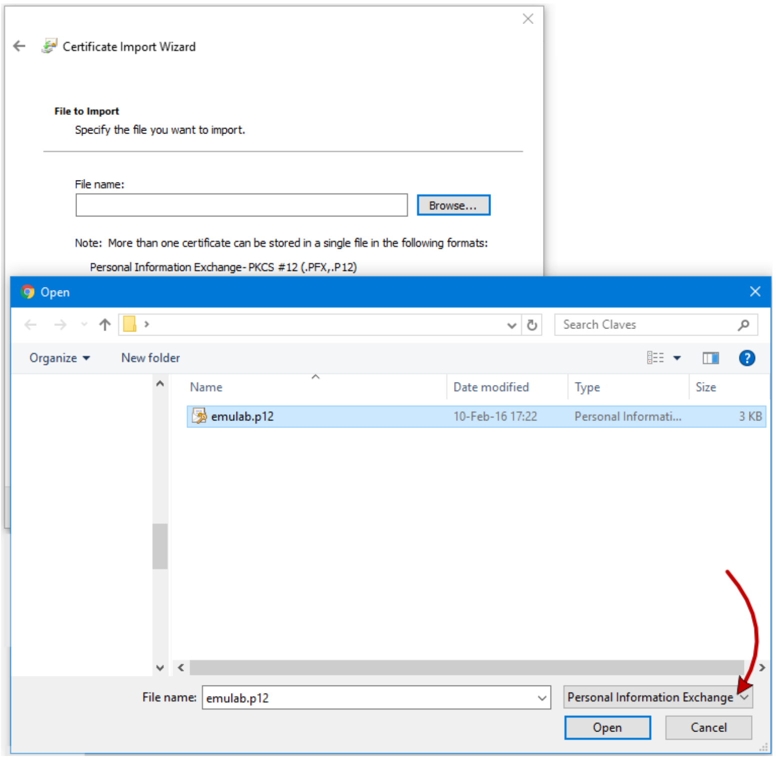
Note: More than one certificate can be stored in a single file in the following formats:
Personal Information Exchange- PKCS #12 (.PFX,.P12)

Open

Name	Date modified	Type	Size
fed4fire.cer	10-Feb-16 17:22	Security Certificate	3 KB

File name: fed4fire.cer
X.509 Certificate (*.cer;*.crt)

7. And select the file containing your certificate:


Certificate Import Wizard

File to Import
Specify the file you want to import.

File name:

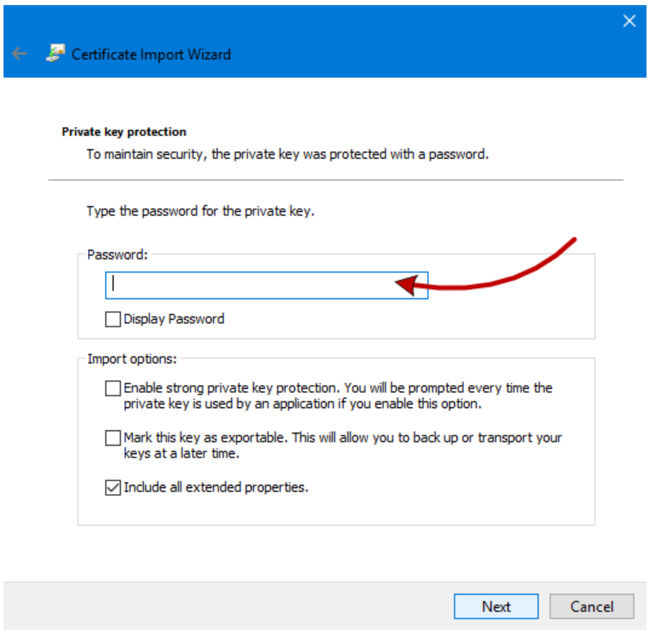
Note: More than one certificate can be stored in a single file in the following formats:
Personal Information Exchange- PKCS #12 (.PFX,.P12)

Open

Name	Date modified	Type	Size
emulab.p12	10-Feb-16 17:22	Personal Informati...	3 KB

File name: emulab.p12
Personal Information Exchange

8. In the next dialog box, enter your certificate password:



Private key protection

To maintain security, the private key was protected with a password.

Type the password for the private key.

Password:

☐ Display Password

Import options:

☐ Enable strong private key protection. You will be prompted every time the private key is used by an application if you enable this option.

☐ Mark this key as exportable. This will allow you to back up or transport your keys at a later time.

☑ Include all extended properties.

9. If the import is successful, your certificate will be imported to your "Personal Store" and you will be able to access the MONROE user interface by selecting it when prompted by your browser. Notice that you may still get a warning about the validity of the server certificate.

3.2 Resource allocation

New instances of experiments are created, assigned resources and scheduled under the tab "New." Here, the user will be presented with a page similar to Figure 3.

To create a new experiment, the user must specify at least the following parameters:

Name: A representative experiment description.

Script: A Docker hub path for the experiment container. In the previous example, it would be `your_docker_account/my_experiment`.

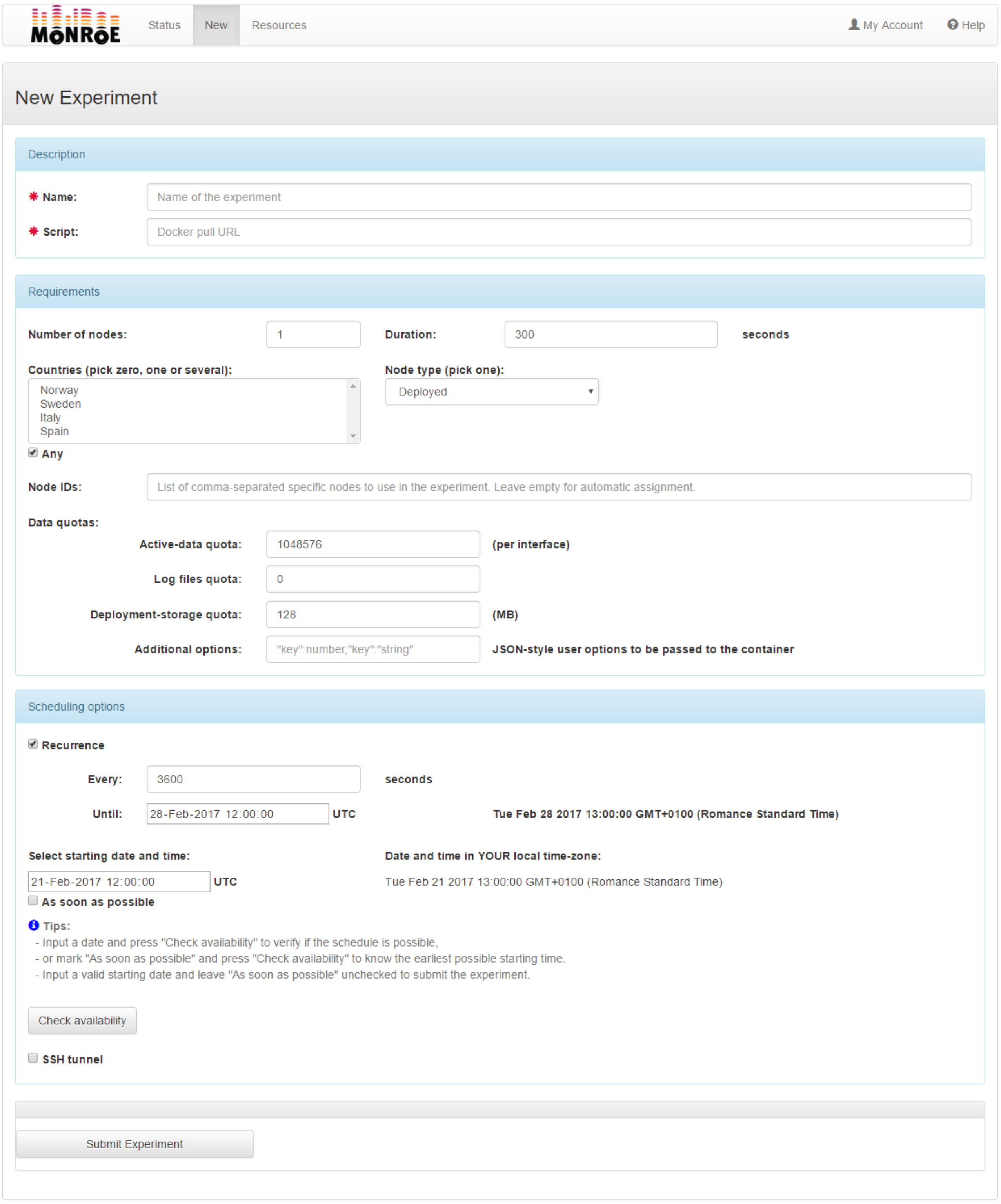
Number of nodes: The number of nodes that will execute the experiment.

Duration: Length of the experiment execution, in seconds (excluding the time required to deploy the container). The node will kill the experiment after this time. The minimum slot that can be reserved is 5 min and the maximum, 24 h.

If the starting date is fixed, the user can introduce it in the field "Start." All dates are introduced as UTC times; the interface presents alongside the corresponding local time for the user's browser. The scheduler will then try to satisfy the requirements.

Alternatively, if the starting date is not relevant, the user may leave this field empty and press the button "Check availability" to check the earliest available slot (add at least ten minutes to the proposed time to allow for container deployment into the nodes). If the user just wants to submit the experiment as soon as possible, they can just mark the option "As soon as possible" and leave the other fields empty when pressing the "Submit" button.

Additionally, the user may specify the following restrictions (Figure 4):



New Experiment

Description

* Name: Name of the experiment
* Script: Docker pull URL

Requirements

Number of nodes: 1
Duration: 300 seconds

Countries (pick zero, one or several): Norway, Sweden, Italy, Spain
[✓] Any

Node type (pick one): Deployed

Node IDs: List of comma-separated specific nodes to use in the experiment. Leave empty for automatic assignment.

Data quotas:

Active-data quota:	1048576	(per interface)
Log files quota:	0	
Deployment-storage quota:	128	(MB)
Additional options:	"key":number,"key":"string"	JSON-style user options to be passed to the container

Scheduling options

[✓] Recurrence

Every: 3600 seconds
Until: 28-Feb-2017 12:00:00 UTC   Tue Feb 28 2017 13:00:00 GMT+0100 (Romance Standard Time)

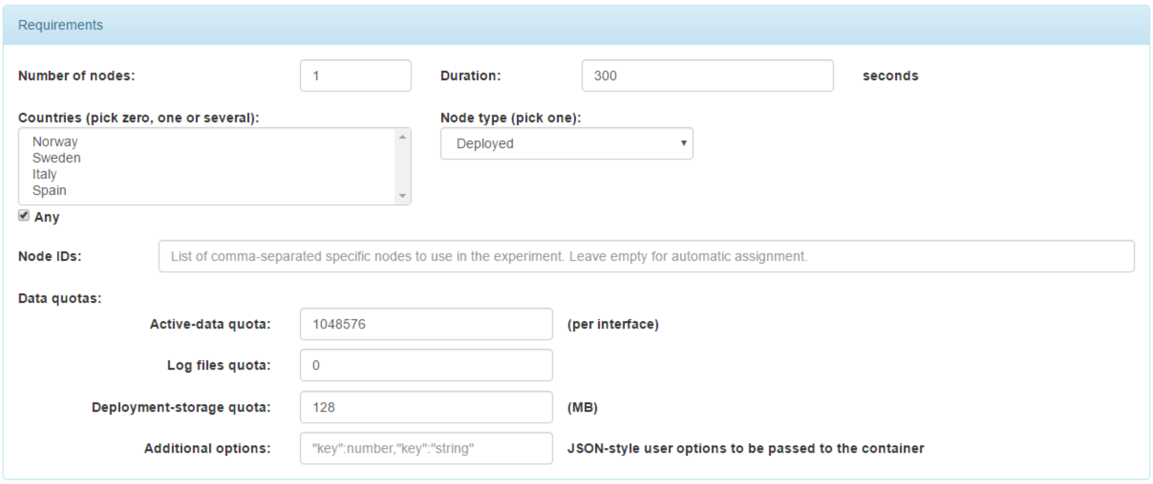
Select starting date and time: 21-Feb-2017 12:00:00 UTC
[ ] As soon as possible

Date and time in YOUR local time-zone: Tue Feb 21 2017 13:00:00 GMT+0100 (Romance Standard Time)

Tips:
- Input a date and press "Check availability" to verify if the schedule is possible,
- or mark "As soon as possible" and press "Check availability" to know the earliest possible starting time.
- Input a valid starting date and leave "As soon as possible" unchecked to submit the experiment.

[ ] SSH tunnel

Figure 3: Example for the creation of a new experiment.


Requirements

Number of nodes:	1	Duration:	300	seconds
Countries (pick zero, one or several):	Norway, Sweden, Italy, Spain	Node type (pick one):	Deployed	
✓ Any				
Node IDs:	List of comma-separated specific nodes to use in the experiment. Leave empty for automatic assignment.			
Data quotas:				
Active-data quota:	1048576	(per interface)		
Log files quota:	0			
Deployment-storage quota:	128	(MB)		
Additional options:	"key":number,"key":"string"	JSON-style user options to be passed to the container		

Figure 4: Filters for node selection.

Country filter: The user may select nodes located in one or several countries, or they may choose to use nodes from any country indistinctly.

Node type: Currently available node types are deployed or testing. The “testing” nodes are reserved for experiments that must still be verified by a MONROE administrator.

Node IDs: If the experimenter wants to use a set of specific nodes, for example, to repeat one experiment under the very same conditions, it is possible to introduce a comma-separated list of required nodes, instead of accepting any available ones.

Active-data quota: The experimenter must specify the active-data quota for each interface, that is, the maximum amount of data that each interface can use. The scheduler checks this value against the quota available for the user.

Log files quota: The user may want to place an estimate on the maximum amount of data that may be generated as result files in `/monroe/results`. This is important because the size of the results is also counted against the user quota.

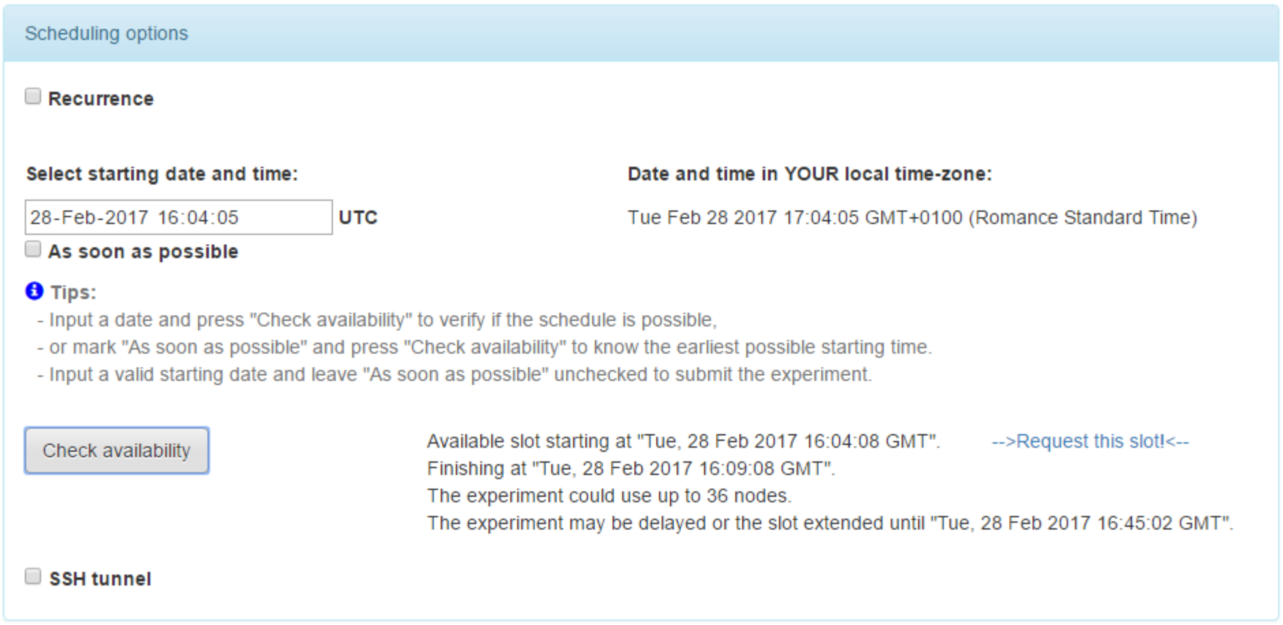
Deployment-storage quota : This is the size allocated for the container file system in the node. Bigger sizes require more time to deploy. The maximum limit is 1 GB.

Recurrence: MONROE’s scheduler allows to specify experiments that need to be repeated periodically. In that case, the user has to specify the repetition period (≥ 3600 s) and the final stopping date. The scheduler will treat each repetition as a different experiment and will try to satisfy the requirements for each of them consecutively. However, the operation is atomic: Either all the repetitions are scheduled, or none are.

3.3 Experiment scheduling

When all the requirements are specified, the user needs to click the “Submit experiment” button to submit to the scheduler. The experiments must respect several restrictions to be successfully scheduled:

- The starting time must be at least 10 min in the future, to allow time for container deployment.



Scheduling options

☐ Recurrence

Select starting date and time: 28-Feb-2017 16:04:05 UTC

☐ As soon as possible

Date and time in YOUR local time-zone: Tue Feb 28 2017 17:04:05 GMT+0100 (Romance Standard Time)

Tips:
- Input a date and press "Check availability" to verify if the schedule is possible,
- or mark "As soon as possible" and press "Check availability" to know the earliest possible starting time.
- Input a valid starting date and leave "As soon as possible" unchecked to submit the experiment.

Available slot starting at "Tue, 28 Feb 2017 16:04:08 GMT". -->Request this slot!<--
Finishing at "Tue, 28 Feb 2017 16:09:08 GMT".
The experiment could use up to 36 nodes.
The experiment may be delayed or the slot extended until "Tue, 28 Feb 2017 16:45:02 GMT".

☐ SSH tunnel

Figure 5: The scheduler may supply hints on the scheduling availability, including the earliest starting date that is possible, the end of the availability period for the required resources and the maximum number of nodes, with the specified requirements, that the experiment could reserve. In this example, the experiment can start on “2017-02-28 16:04:08 UTC” and can last until “2017-02-28 16:45:02 UTC.” The experiment can be scheduled with up to 36 nodes during this period.

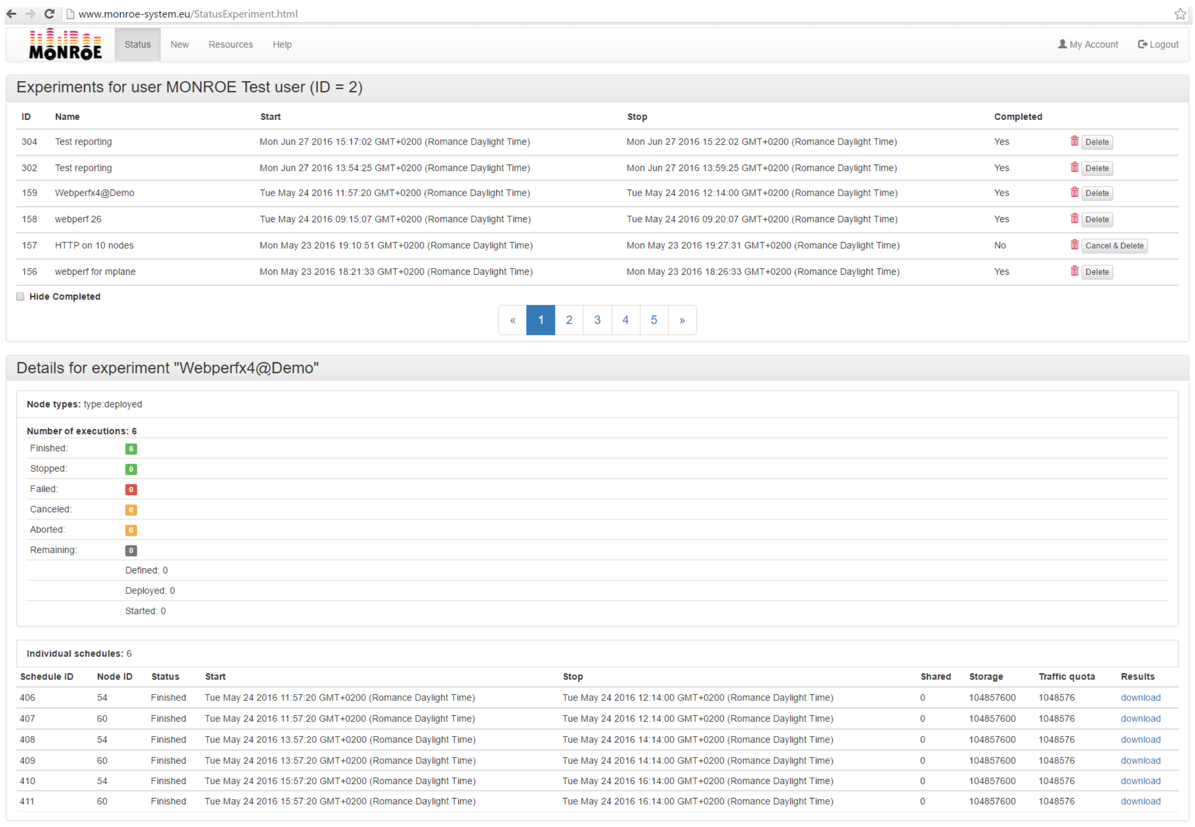
- No experiment can be scheduled more than one month in advance.
- Periodic experiments must have a period greater than 3600 s. The finishing time must also obey the previous rule, that is, the last experiment instance in the recurrence must be scheduled in less than a month from the current time.
- No experiment (or instance in a series) can last more than one day.
- If a list of specific nodes and a starting date are given, the scheduler may be unable to grant the required resources.

3.3.1 Checking availability

If the exact starting time is not relevant, the user can press the “Check availability” button. If the requirements can be satisfied, a message explaining when the experiment might be started will be displayed. Additionally, it will also inform of the maximum number of nodes that can be used during this period, and the maximum ending time. With these data, the experimenter may decide to increase the number of nodes that run the experiment, or increase its duration until the time that the scheduler is likely being able to grant. Figure 5 shows the answer of the scheduler for an availability query.

3.4 Experiment monitoring

Once an experiment is successfully submitted, the user can check its progress under the “Status” tab. Figure 6 shows an example of a list of experiments.



Experiments for user MONROE Test user (ID = 2)

ID	Name	Start	Stop	Completed	
304	Test reporting	Mon Jun 27 2016 15:17:02 GMT+0200 (Romance Daylight Time)	Mon Jun 27 2016 15:22:02 GMT+0200 (Romance Daylight Time)	Yes	Delete
302	Test reporting	Mon Jun 27 2016 13:54:25 GMT+0200 (Romance Daylight Time)	Mon Jun 27 2016 13:59:25 GMT+0200 (Romance Daylight Time)	Yes	Delete
159	Webperfx4@Demo	Tue May 24 2016 11:57:20 GMT+0200 (Romance Daylight Time)	Tue May 24 2016 12:14:00 GMT+0200 (Romance Daylight Time)	Yes	Delete
158	webperf 26	Tue May 24 2016 09:15:07 GMT+0200 (Romance Daylight Time)	Tue May 24 2016 09:20:07 GMT+0200 (Romance Daylight Time)	Yes	Delete
157	HTTP on 10 nodes	Mon May 23 2016 19:10:51 GMT+0200 (Romance Daylight Time)	Mon May 23 2016 19:27:31 GMT+0200 (Romance Daylight Time)	No	Cancel & Delete
156	webperf for mplane	Mon May 23 2016 18:21:33 GMT+0200 (Romance Daylight Time)	Mon May 23 2016 18:26:33 GMT+0200 (Romance Daylight Time)	Yes	Delete

☐ Hide Completed

« 1 2 3 4 5 »

Details for experiment "Webperfx4@Demo"

Node types: type:deployed

Number of executions: 6

Finished:	6
Stopped:	0
Failed:	0
Canceled:	0
Aborted:	0
Remaining:	0
	Defined: 0
	Deployed: 0
	Started: 0

Individual schedules: 6

Schedule ID	Node ID	Status	Start	Stop	Shared	Storage	Traffic quota	Results
406	54	Finished	Tue May 24 2016 11:57:20 GMT+0200 (Romance Daylight Time)	Tue May 24 2016 12:14:00 GMT+0200 (Romance Daylight Time)	0	104857600	1048576	download
407	60	Finished	Tue May 24 2016 11:57:20 GMT+0200 (Romance Daylight Time)	Tue May 24 2016 12:14:00 GMT+0200 (Romance Daylight Time)	0	104857600	1048576	download
408	54	Finished	Tue May 24 2016 13:57:20 GMT+0200 (Romance Daylight Time)	Tue May 24 2016 14:14:00 GMT+0200 (Romance Daylight Time)	0	104857600	1048576	download
409	60	Finished	Tue May 24 2016 13:57:20 GMT+0200 (Romance Daylight Time)	Tue May 24 2016 14:14:00 GMT+0200 (Romance Daylight Time)	0	104857600	1048576	download
410	54	Finished	Tue May 24 2016 15:57:20 GMT+0200 (Romance Daylight Time)	Tue May 24 2016 16:14:00 GMT+0200 (Romance Daylight Time)	0	104857600	1048576	download
411	60	Finished	Tue May 24 2016 15:57:20 GMT+0200 (Romance Daylight Time)	Tue May 24 2016 16:14:00 GMT+0200 (Romance Daylight Time)	0	104857600	1048576	download

Figure 6: List of user experiments.

All the active (i.e., not completed) experiments for the user are shown. Experiments that have not yet been started can be canceled and deleted. However, the scheduler will try to stop experiments that have already started, but they will not be deleted from the list.

Clicking on any experiment displays the details for its individual schedules. There, the number of schedules that are defined but not yet deployed, the ones that are deployed and ready to be started, the ones that are currently running, etc., is summarized. One line is presented for each individual schedule on each MONROE node. Table 1 explains the states in which an individual task may be.

Some experiments may be designed to finish after completion. For those ones, the correct finishing state is "Finished." If they are stopped by the scheduler, they probably exceeded the execution time foreseen by the experimenter. However, other experiments may be designed to run continuously for a period of time. In those cases, the "Stopped" state could actually be the correct ending state as intended by the experimenter.

4 Retrieval of metadata and experiment results

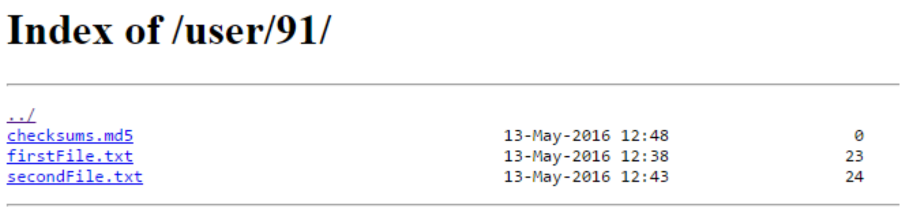
MONROE repositories contain two types of data: MONROE metadata itself and the results of user experiments.

4.1 User experiment results

Any files written during the experiment to the `/monroe/results` directory will be synchronized to the experiment repository. This operation happens continuously during experiment execution and then upon its finalization. Therefore, it is advisable that only final files ready to be transferred are copied (indeed, mv'ed)

Table 1: Experiment states

STATE	DESCRIPTION
(Ongoing states)	
Defined	The experiment is created in the scheduler. If a task remains in this state past its starting time, the node was probably shut down and the task will not be executed anymore.
Requested	The node has requested the container and is deploying it.
Deployed	The node has already deployed the container and is waiting for its starting time.
Delayed	The scheduling process failed temporarily.
Started	The container is being executed in the designated node. The “download” link for the task results is already available.
Restarted	The node has restarted the experiment after a node failure.
(Final states)	
Finished	The task was correctly executed and it finished on its own before consuming the complete time slot.
Stopped	The task was correctly executed, but it was stopped by the scheduler at the end of the execution slot (correct for tasks designed to remain in execution until the end of their time slot).
Failed	The task stopped abnormally.
Canceled	The task was canceled by the user before being started (but other tasks in the experiment were already started).
Aborted	The task was aborted by the user after being started.


Index of /user/91/

```
../
checksums.md5                                      13-May-2016 12:48                   0
firstFile.txt                                      13-May-2016 12:38                  23
secondFile.txt                                     13-May-2016 12:43                  24
```

Figure 7: Folder containing the results of an individual schedule, transferred to MONROE’s servers.

to that location to avoid the system to sync temporary files and consume your quota or produce invalid results. This recommendation means that files in the results folder should not be updated; experimenters are encouraged to copy intermediate result files as soon as they are ready so they can retrieve partial results if the experiment fails in the middle of its execution.

The result files can be accessed through the user interface: For experiments that have already been started, the interface presents a link under the column “Results” that redirects the user to the HTTP folder (Figure 7) that contains the files already synchronized from the node where the experiment runs to the repository. In this way, the experimenter can retrieve result files even for partial experiments that fail or are canceled.

In addition, the experiment may use any network functionalities to communicate with outside servers as needed (e.g., `scp` some files to an external server). In order to improve safety, private keys should be restricted to the experiments and discarded after a reasonable time. Additionally, instead of saving your keys in the container itself, you may want to pass them as additional options during experiment scheduling. The values will be available during container execution as a JSON file at `/monroe/config`. Notice that this file is created by the node scheduler. The same effect is achievable when the containers are run manually in user development nodes adding the `-v` option to the command line. To map both a locally created config file and the results folder of the container to a node folder, in development nodes without a scheduler, the following command line fragment may be used:[2]

```
-v /monroe/results:/monroe/results -v /monroe/config:/monroe/config:ro
```

4.2 MONROE metadata

MONROE metadata can be freely accessed by two means. First, a CSV dump of all database tables is generated daily. The files can be accessed at the following URL (a valid user certificate is needed): https://www.monroe-system.eu/user/dailyDumps/. The dump files should be available every day after 12:00 CET (24-hour format). Each file covers the period [00:00, 00:00) GMT.

Our servers run on CET time, but metadata timestamps use GMT. Therefore, to cover "a day" of metadata using, e.g., the local time in Norway, two CSV files need to be combined. During Winter time, the needed metadata is in the period [day_0:01:00, day_1:01:00). During Summer time, the needed metadata is in the period [day_0:02:00, day_1:02:00).

Alternatively, metadata can be accessed directly in a replica of MONROE's Cassandra database, which is updated daily approximately at noon with the data from the previous (GMT) day. Access credentials for this server will be provided as requested. MONROE repositories include several examples on how to access the database.

5 Run-time considerations for experimenters

This section discusses several considerations that experimenters must take into account when designing and running their experiments on the MONROE platform.

5.1 Communication during the experiment

During execution, the experiment is free to establish any network communications through the available interfaces. The user can choose to bind explicitly from each command or application to a specific interface, or they may define default routes during the experiment:

```
route add default gw 172.16.0.1 eth0
```

5.2 Interface naming and binding, and default route

⇒ *This section contains technical details that are currently subject to evaluation by the MONROE consortium. Comments from experimenters on this section are welcome.* ⇐

Experiments running in MONROE nodes have access to several network interfaces. By default, that is, if the experiment does not take any special configuration actions, the default route will be configured to one of the mobile broadband interfaces, if available. However, experimenters have the possibility of explicitly binding external tools or their programs to specific interfaces. For example, the standard "ping" tool can be forced to use an specific interface with the following command:

```
ping -I eth0 host_name
```

[2]Thanks to Eneko Atxutegi Narbona and Jonas Karlsson for pointing this out.

To offer a consistent view of the platform resources, whereas allowing flexibility for future changes in the platform configuration, the following naming scheme is used for each of the interfaces available for the experiments:

op0: First operator for the nodes in the given country.
op1: Second operator for the nodes in the given country.
op2: Third operator for the nodes in the given country.
eth0: Ethernet (wired) network connection, when available.

The platform guarantees that a given op$_i$ corresponds to the same operator during experiment execution. However, the assignment may change between nodes in the same country or even between successive executions in the same node. Therefore, experiments must check the metadata stream to select the correct interface.

Under some circumstances, the mobile devices used in the MONROE nodes may loose connectivity, reset themselves or undergo any other process that makes them temporarily unavailable for the experiments. To identify and tackle with these situations, experimenters are encouraged to build "robust" experiments subscribing to the corresponding metadata streams.

If the experimenter writes their own code:

1. Subscribe to the metadata broadcast.
2. Wait for a `MODEM.*.UPDATE` message for the modem(s)/operators of interest.
3. Once this information is obtained, use the desired interface and store the results with the corresponding ICCID or operator name.
4. If the interface should disappear (`ENODEV` error, "no such device"), start over at 2.

When an external tool that does not handle `ENODEV` (e.g., "fping"), replace step 4 by:

- Monitor the metadata for a `MODEM.*.CONNECTIVITY` message indicating that connectivity was lost, or monitor the interface list to check if the device disappears. Upon either event, start over at 2.

Experimenters should take notice that an interface may not only go down, but it may actually disappear from the list of available interfaces (if the modem has to be restarted, or anything similar happens in the USB stack). Even if it reappears soon after, any existing network connections on the old interface will fail with `ENODEV`.

It is also possible to skip steps 1 and 2 when reconnecting to an interface after a failure, as the interface name corresponding to the desired operator is already known. It is still necessary to keep retrying to connect to the interface, until it comes up.

5.3 Metadata at run-time

MONROE nodes retrieve constantly some metadata information concerning their own state and the network conditions. This information is continuously uploaded to the MONROE servers and stored in a database. One of the main goals of the MONROE project is to make all that information freely accessible. Therefore, experimenters may perform an off-line correlations of events in their experiment with the information in the MONROE database.

MONROE experimenters can also access all the metadata information at run-time from their experiments to achieve easy correlation of events or modify the behavior of the experiment during its execution. For example:

- Experiments that depend on external factors (location):
  - Round trip time vs. location.
  - Proactive HTTP caching according to location.
  - Round trip time vs. base station.
  - Round trip time vs. signal strength.
  - Route selection according to current conditions.
- Experiment validation:
  - Verify that node temperature is/was within limits.
  - Verify that system load is/was below threshold.

The metadata is broadcast locally using ZeroMQ. The following excerpt in Python shows how an application can subscribe to the metadata stream:

```
import zmq

context = zmq.Context()
socket = context.socket(zmq.SUB)
socket.connect ("tcp://172.17.0.1:5556")

# An empty string subscribes to everything:
topicfilter = ''   # E.g., use 'MONROE.META.DEVICE.GPS' for GPS-only metadata
socket.setsockopt(zmq.SUBSCRIBE, topicfilter)

while True:
  string = socket.recv()
  print string
```

5.3.1 Example: Correlate experiment results with metadata at run-time

The following example shows how to create an application that executes a ping to an external machine and saves the results alongside the node location:

- Pipe the ping command through a "ping formatter."
- The "ping" formatter subscribes to a zmq socket and topic:
  - Socket : 'tcp://172.17.0.1:5556'
  - Topic : 'MONROE.META.DEVICE.GPS'
- Cache the GPS position received.
- Wait for output from the ping command (stdin).
- Store experiment information including the GPS position:
  - Use the "library" `monroe_exporter` (python only).
  - Call the `monroe_exporter` script via the command line.

Below is the corresponding source code:

```
socket.connect('tcp://localhost:5557')
socket.setsockopt(zmq.SUBSCRIBE, 'MONROE.META.DEVICE.GPS')
LAST_GPS_FIX = None

monroe_exporter.initalize('MONROE.EXP.PING', 1, 5.0)

'''fork and wait for for gps messages'''
```

```
while True:
  (topic, msgdata) = socket.recv_multipart()
  LAST_GPS_FIX = json.loads(msgdata)

'''main process waits for ping experiment output '''
while line:
  exp_result = r.match(line).groupdict()
  msg = {
    'InterfaceName': interface,
    'Bytes': int(exp_result['bytes']),
    'Host': exp_result['host'],
    'Rtt': float(exp_result['rtt']),
    'SequenceNumber': int(exp_result['seq']),
    'TimeStamp': float(exp_result['ts'])
  }
  if LAST_GPS_FIX != None:
    msg.update(
      {
        'GPSTimeStamp': LAST_GPS_FIX['TimeStamp'],
        'Latitude': LAST_GPS_FIX['Latitude'],
        'Longitude': LAST_GPS_FIX['Longitude'],
        'Altitude': LAST_GPS_FIX['Altitude'],
        'NumberofSatellites': LAST_GPS_FIX['NumberofSatellites']
          })

  monroe_exporter.save_output(msg)
  line = sys.stdin.readline()
```

5.3.2 Metadata information

Currently, the collected metadata includes:

- Node GPS.
- Node sensors (CPU temp) and probes (load, memory usage).
- Modem status and events.
- Continuous and scheduled internal experiments:
  - RTT (through ping).
  - Bandwidth (through HTTP download).

The following and some examples of the information received in the metadata stream:

- RTT experiment:

{"DataId": "MONROE.EXP.PING", "Bytes": 84, "NodeId": "54", "SequenceNumber": 301, "DataVersion": 1, "Timestamp": 1465805479.747943, "Rtt": 71.2, "Host": "8.8.8.8", "Operator": "Orange", "Iccid": "89340142515410360l3", "Guid": "sha256:a9f9fb2c04bba3782ef2624e118faa18f16b08c826155cae5e1ea7e1d88832b5.0.54.3791"}

- Sensors, where each message may contain information about a different set of measurements:

{"DataId": "MONROE.META.NODE.SENSOR", "softirq": "205270", "SequenceNumber": 48581, "DataVersion": 1, "b": "1059270", "b": "4885494", "guest": "0", "NodeId": "54", "idle": "42657942", "user": "10480984", "irq": "0", "steal": "0", "Timestamp": 1465786966, "nice": "3063"}

```
{"DataId": "MONROE.META.NODE.SENSOR", "SequenceNumber": 48567, "DataVersion": 1,
"Timestamp": 1465786961, "percent": "65.98", "NodeId": "54", "current": "302234",
"start": "1465484726", "total": "5246545.72", "id": "39"}

{"DataId": "MONROE.META.NODE.SENSOR", "SequenceNumber": 48460, "DataVersion": 1,
"Timestamp": 1465786926, "apps": "3632746496", "NodeId": "54", "free": "483119104",
"swap": "0"}
```

- Modem events:

```
{"DataId": "MONROE.META.DEVICE.MODEM", "InterfaceName": "usb2", "CID": 72209509,
"DeviceState": 3, "SequenceNumber": 33548, "DataVersion": 1, "Timestamp": 1465803136,
"NWMCCMNC": 21404, "Band": 3, "RSSI": -80, "IPAddress": "10.33.101.173",
"IMSIMCCMNC": 21404, "DeviceMode": 5, "NodeId": "54", "IMEI": "864154023645179",
"RSRQ": -8, "RSRP": -85, "LAC": 28014, "Frequency": 1800,
"InternalIPAddress": "192.168.0.153", "Operator": "YOIGO",
"ICCID": "8934041514050774002", "IMSI": "214040113950108"}
```

- GPS:

```
{"DataId": "MONROE.META.DEVICE.GPS", "SequenceNumber": 34164, "DataVersion": 1,
"Timestamp": 1465805718, "Altitude": -1455.900024, "NodeId": "63",
"Longitude": -3.777019, "NMEA":
"$GPGGA,081518.0,4020.002011,N,00346.621107,W,1,02,500.0,-1455.9,M,53.0,M„*5D\r\n",
"SatelliteCount": 2, "Latitude": 40.333366}
```

5.3.3 Metadata format

Metadata and internal experiment results follow a JSON structure, as detailed in https://github.com/MONROE-PROJECT/Experiments/wiki:

- All Metadata messages have a topic according to Table 2. Appendix B gives the complete description of the meaning of all the metadata fields.
- All metadata topics are prefixed with "MONROE.META."
- All internal experiments are prefixed with "MONROE.EXP."
- Experiments receive metadata messages only for topics to which they subscribe.
- An empty string ("") subscribes to all topics.

5.4 Tstat at run-time

The Tstat (http://www.tstat.polito.it/) **runs** on all nodes in the mPlane container as one of the basic MONROE containers. The Tstat is a passive probe able to provide several insight on the traffic patterns at both the network and the transport levels. The Tstat generates two different types of logs.

5.4.1 Tstat Round Robin Database

The RRD (Round Robin Database) logs is an average of samples of each packet in 5 minutes, it imposes at least 5 minutes delay to visualize RRDs. The detail description of the RRD logs is available on Tstat

Table 2: Metadata topics.

TOPIC	DESCRIPTION
*.DEVICE.MODEM.iccid.UPDATE	
*.DEVICE.MODEM.iccid.MODE	
*.DEVICE.MODEM.iccid.SIGNAL	
*.DEVICE.MODEM.iccid.LTEBAND	
*.DEVICE.MODEM.iccid.ISPNAME	
*.DEVICE.MODEM.iccid.IPADDR	
*.DEVICE.MODEM.iccid.LOCCHANGE	
*.DEVICE.MODEM.iccid.NWMCCMNCCHANGE	
*.DEVICE.GPS	
*.NODE.SENSOR.sensor_name	Temp sensor, running experiments, quotas, . . .
*.NODE.EVENT	Power up events, etc, . . .

documentation(http://tstat.polito.it/HOWTO.shtml#RRD). RRD are available via the (http://monroe-repository.polito.it:8080/) and the Graphite GUI provides some tool to present RRD logs and save the interested plots. Fig. 8 shows the bit rate of the ICMP packet for the node #38 on interface *op0* over the last 24 hours. There is possibility to create a dashboard to monitor the experiments and interfaces' status. Fig. 9 illustrates an example of saved dashboard to monitor the volume of traffic on one node.

5.4.2 Tstat logs

Tstat generates detailed flow level logs for TCP, UDP, and HTTP flows. These are text file with more than 100 metrics, containing information about the client and server addresses, network and application level metrics, and DNS queries. The description of the metrics presents (http://tstat.polito.it/measure.shtml#LOG). In MONROE, the Tstat configure to generate 4 different logs as following:

Table 3: Tstat log types.

TYPE	DESCRIPTION
log_tcp_complete	Every TCP connection that has been tracked
log_tcp_nocomplete	All the connections for which the three way handshake is not properly seen
log_udp_complete	Every tracked UDP flow pair
log_http_complete	Information from every HTTP request and response

Logs are available in two ways,

1. Real time access on the node, logs for the last three generated are shared with MONROE experimenters on the "/monroe/tstat", it helps the MONROE users to use passive traces collected by Tstat during their experiment. The three logs can cover at most the last three hours.

2. On demand, all logs are imported into MONROE database for all node. The schema of the tables are available on github (https://github.com/MONROE-PROJECT/Database/blob/master/db_schema.cql). Three columns, (NodeId,Iccid,DataId) added to each table bring the possibility to join with metadata and collected data.

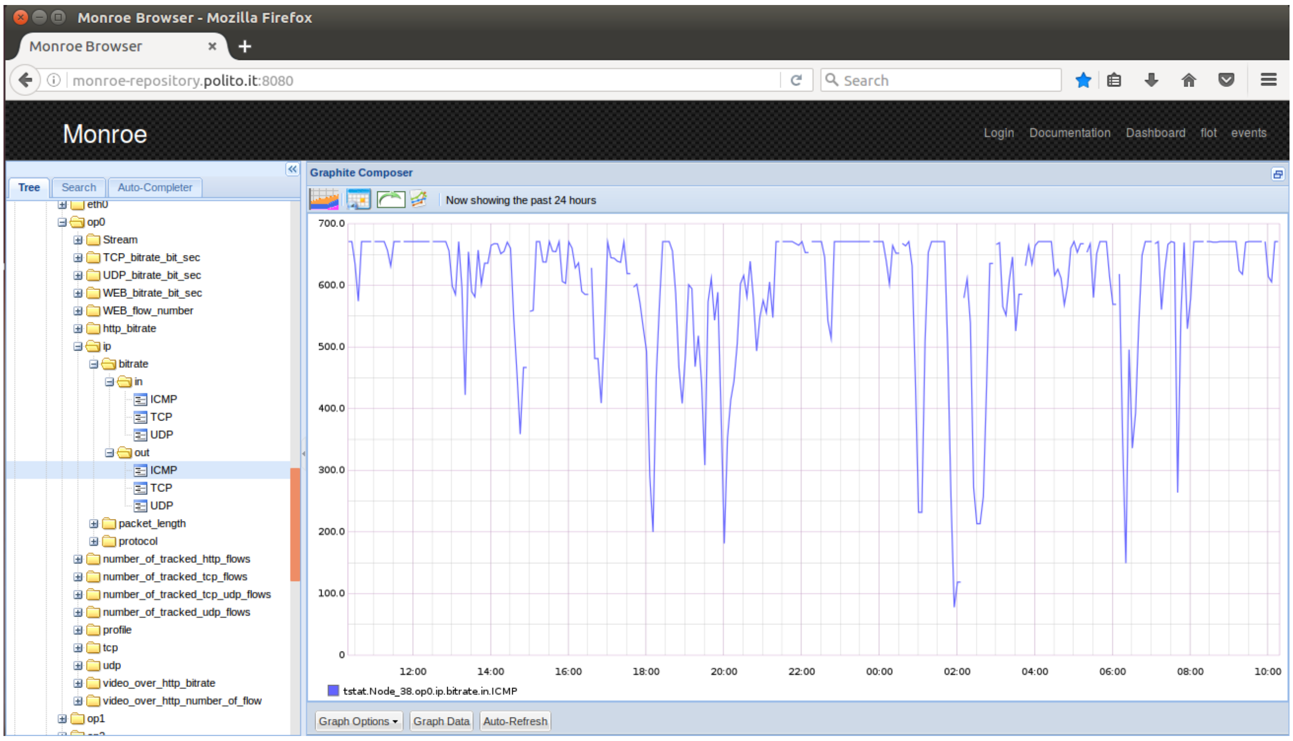


Figure 8: Graphite GUI of the Tstat RRD logs.

It is recommended to check the description of the logs on (`http://tstat.polito.it/measure.shtml#LOG`). Tables 4, 5 and 6 present the table describing of some interesting metrics in tcp and http logs.

5.5 Access to user-owned development nodes

This section refers to development nodes owned by external users under the dispositions of their specific MONROE agreement. Two options are possible for the management of those nodes:

1. The nodes join the pool of MONROE nodes. Experiments are scheduled through the MONROE scheduler and users, including the node owners, do not have direct SSH access to them. The metadata produced by these nodes will join the rest of the MONROE databases.
2. The nodes are considered "development nodes" for private use of their owners. In that case, they will not join the MONROE platform and will not be accessible through the MONROE scheduler, neither for their owners nor for other users. The nodes will be marked as "storage" or "development." Thus, users (again, only their owners) must log locally into the nodes to manually schedule their containers using Docker commands. The nodes will not run the base experiments; no metadata, or any other information produced by them will join the MONROE databases.

In essence, "managed" nodes are part of the testbed and work as any other ones, whereas "development" nodes are for private use of their owners. The following paragraphs provide relevant information for the use of development nodes.

5.5.1 Accessing user-owned development nodes

Development nodes can be accessed either through the management interface (the black wire connected to eth2) via SSH, or directly via the serial console (DB9 connector, using a null-modem cable). The necessary

Table 4: Core TCP Set.

C2S	S2C	Short description	Unit	Long description
1	15	Client/Server IP addr	–	IP addresses of the client/server
2	16	Client/Server TCP port	–	TCP port addresses for the client/server
3	17	packets	–	total number of packets observed from the client/server
4	18	RST sent	0/1	0 = no RST segment has been sent by the client/server
5	19	ACK sent	–	number of segments with the ACK field set to 1
6	20	PURE ACK sent	–	number of segments with ACK field set to 1 and no data
7	21	unique bytes	B	number of bytes sent in the payload
8	22	data pkts	–	number of segments with payload
9	23	data bytes	B	number of bytes transmitted in the payload, including retransmissions
10	24	rexmit pkts	–	number of retransmitted segments
11	25	rexmit bytes	B	number of retransmitted bytes
12	26	out seq pkts	–	number of segments observed out of sequence
13	27	SYN count	–	number of SYN segments observed (including rtx)
14	28	FIN count	–	number of FIN segments observed (including rtx)
29		First time abs	ms	Flow first packet absolute time (epoch)
30		Last time abs	ms	Flow last segment absolute time (epoch)
31		Completion time	ms	Flow duration since first packet to last packet
32		C first payload	ms	Client first segment with payload since the first flow segment
33		S first payload	ms	Server first segment with payload since the first flow segment
34		C last payload	ms	Client last segment with payload since the first flow segment
35		S last payload	ms	Server last segment with payload since the first flow segment
36		C first ack	ms	Client first ACK segment (without SYN) since the first flow segment
37		S first ack	ms	Server first ACK segment (without SYN) since the first flow segment
38		C internal	0/1	1 = client has internal IP, 0 = client has external IP
39		S internal	0/1	1 = server has internal IP, 0 = server has external IP
40		C anonymized	0/1	1 = client IP is CryptoPAn anonymized
41		S anonymized	0/1	1 = server IP is CryptoPAn anonymized
42		Connection type	–	Bitmap stating the connection type as identified by TCPL7 inspection engine (see protocol.h)
43		P2P type	–	Type of P2P protocol, as identified by the IPP2P engine (see ipp2p_tstat.h)
44		HTTP type	–	For HTTP flows, the identified Web2.0 content (see the http_content enum in struct.h)

Table 5: TCP End to End Set.

C2S	S2C	Short description	Unit	Long description
45	52	Average rtt	ms	Average RTT computed measuring the time elapsed between the data segment and the corresponding ACK
46	53	rtt min	ms	Minimum RTT observed during connection lifetime
47	54	rtt max	ms	Maximum RTT observed during connection lifetime
48	55	Stdev rtt	ms	Standard deviation of the RTT
49	56	rtt count	–	Number of valid RTT observation
50	57	ttl_min	–	Minimum Time To Live
51	58	ttl_max	–	Maximum Time To Live

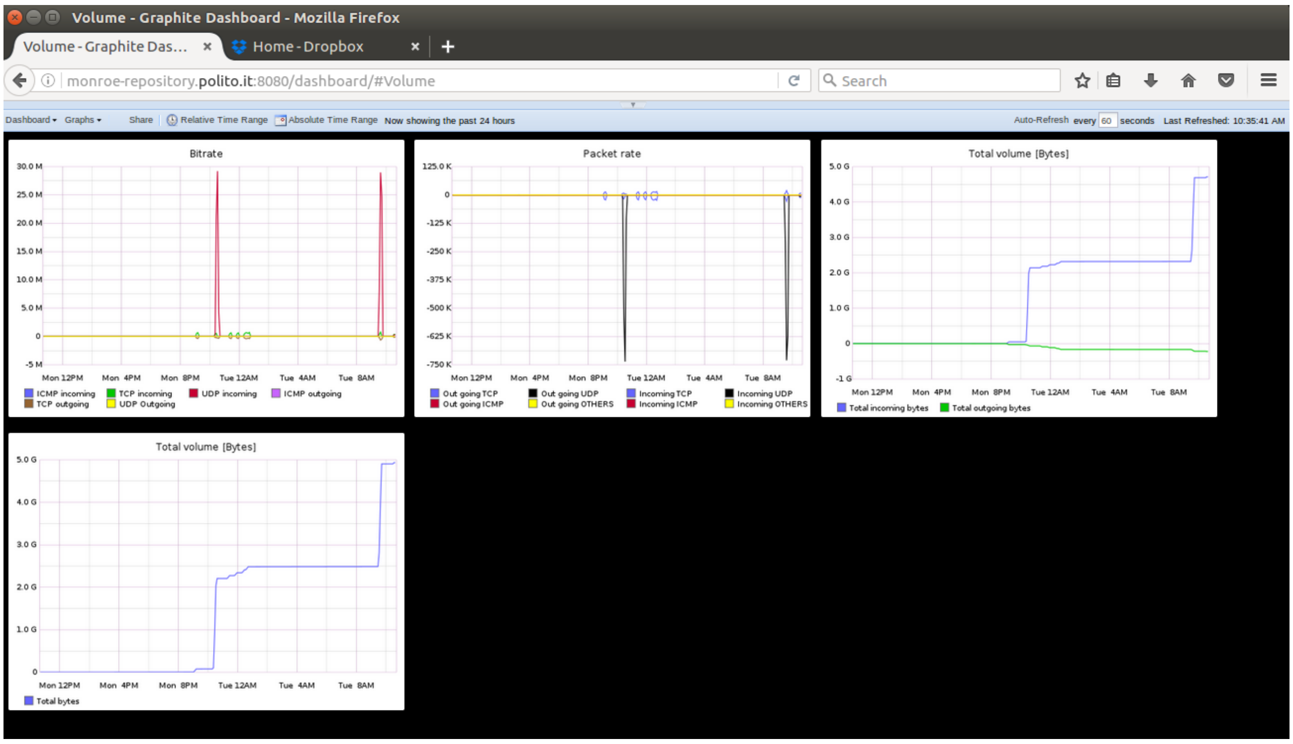

Volume - Graphite Dashboard - Mozilla Firefox
monroe-repository.polito.it:8080/dashboard/#Volume
Bitrate
Packet rate
Total volume [Bytes]
Total volume [Bytes]

Figure 9: An example of dashboard on Tstat RRD GUI.

passwords will be provided on request through a secure channel. Table 7 explains the uses of each available user.

Do not distribute passwords or keys to unauthorized personnel. Do not send passwords or keys over insecure channels. Use of the administrator user 'monroeSA' is allowed only for development on local nodes, unless granted permission to perform a specific task requiring this user. Creation of user accounts on nodes is forbidden. Modifying user accounts on nodes is forbidden. Modifying `authorized_keys` on nodes is forbidden. Be VERY careful if you change any firewall settings, and only do this on development nodes. Be smart.

Local access, which allows password authentication, can be achieved through the serial port and the management interface.

The APU's third ethernet port (eth2), nearest the USB ports, has the IP address 172.16.254.1/24. The node can be accessed setting a static IP address in the 172.16.254.0/24 network span (e.g., 172.16.254.2) on the developer side of the link and establishing an SSH connection.

The DB-9 serial port (console) allows direct terminal access. The boot process and grub menu are visible and interactive through this connection; some kernel messages will be printed as well while connected. Connecting to the console port requires a **null modem** cable. In the following examples, `/dev/ttyS0` has to be substituted with the device path for the developer's cable. A typical case for USB-to-serial adapters is `/dev/ttyUSB0`:

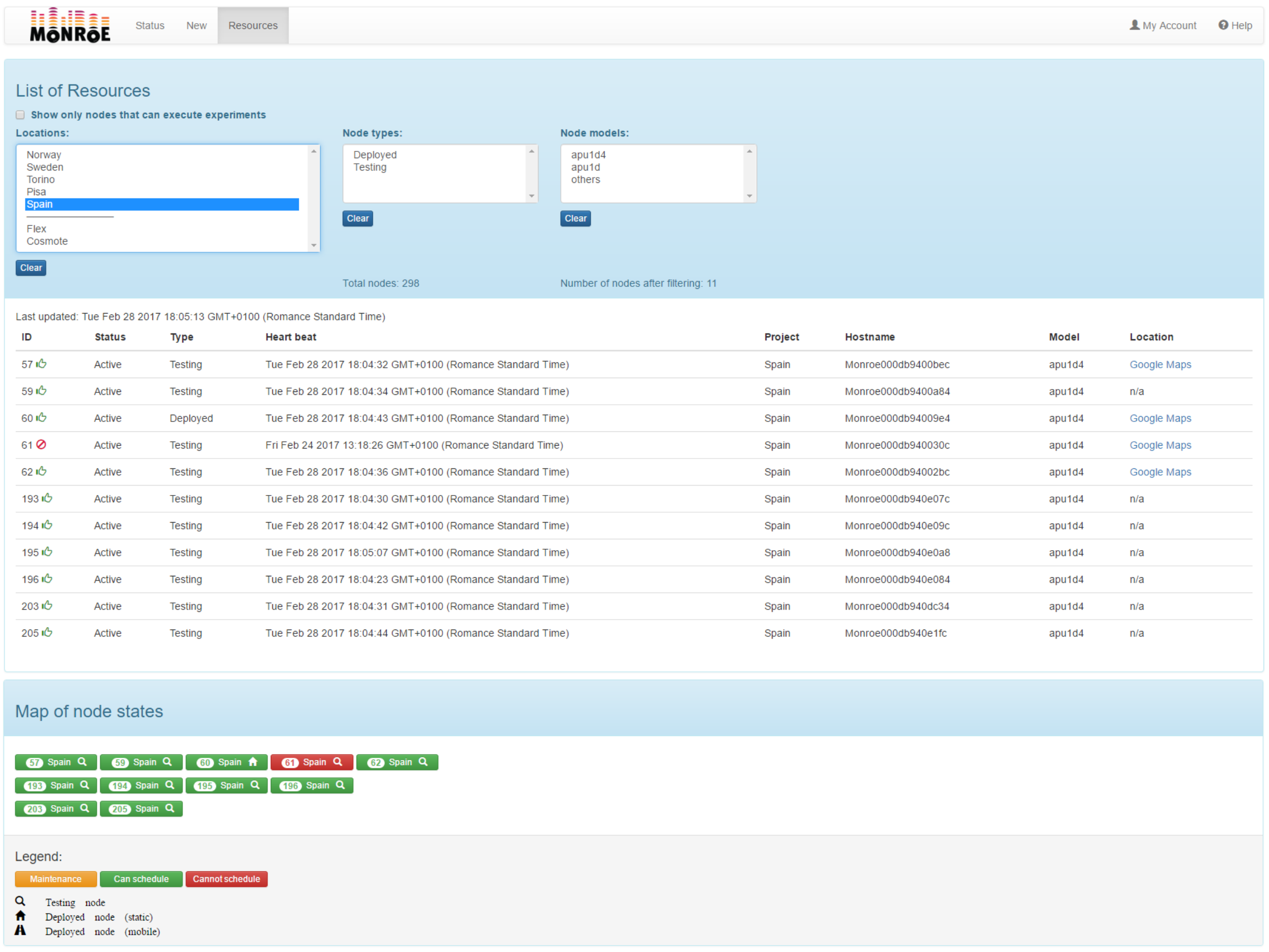
```
minicom -D /dev/ttyS0          ---or---          screen /dev/ttyS0 115200
```

Table 6: Core HTTP Set.

C2S	S2C	Short description	Unit	Long description
1	1	Client IP addr	–	IP addresses of the client (sending the request/receiving the response)
2	2	Client TCP port	–	TCP port addresses for the client
3	3	Server IP addr	–	IP addresses of the server (receiving the request/sending the response)
4	4	Server TCP port	–	TCP port addresses for the server
5	5	Segment time abs	s	Absolute time [s] (epoch) of the request/response
6		Request method	–	Request method (GET/POST/HEAD) [*]
7		Hostname	–	Value fo the “Host:” HTTP request field
8		FQDN	–	DN-Hunter cached DNS name [^]
9		URL Path	–	URL request path
10		Referer	–	Value of the “Referer:” HTTP request field
11		User agent	–	Value of the “User-Agent:” HTTP request field
12		Cookie	–	Value of the “Cookie:” HTTP request field
13		Do Not Track	–	Value of the “DNT:” HTTP request field
	6	Response string	–	Response identifier (always “HTTP”) [*]
	7	Response code	–	HTTP response code (2xx/3xx/4xx/5xx)
	8	Content len	B	Value of the “Content-Length:” HTTP response field
	9	Content type	–	Value of the “Content-Type:” HTTP response field
	10	Server	–	Value of the “Server:” HTTP response field
	11	Range	–	Value of the “Content-Range:” HTTP response field for partial content (Code 206)
	12	Location	–	Value of the “Location:” HTTP response field for redirected content (Code 302)
	13	Set Cookie	–	Value of the “Set-Cookie:” HTTP response field

Table 7: Node users

USER	PASSWORD	SUDO	USES
monroe	*[redacted]*	reboot	maintenance, troubleshooting
monroeSA	*[redacted]*	yes	administration, development



List of Resources

Show only nodes that can execute experiments

Locations: Norway, Sweden, Torino, Pisa, Spain, Flex, Cosmote

Node types: Deployed, Testing

Node models: apu1d4, apu1d, others

Total nodes: 298

Number of nodes after filtering: 11

Last updated: Tue Feb 28 2017 18:05:13 GMT+0100 (Romance Standard Time)

ID	Status	Type	Heart beat	Project	Hostname	Model	Location
57 ✓	Active	Testing	Tue Feb 28 2017 18:04:32 GMT+0100 (Romance Standard Time)	Spain	Monroe000db9400bec	apu1d4	Google Maps
59 ✓	Active	Testing	Tue Feb 28 2017 18:04:34 GMT+0100 (Romance Standard Time)	Spain	Monroe000db9400a84	apu1d4	n/a
60 ✓	Active	Deployed	Tue Feb 28 2017 18:04:43 GMT+0100 (Romance Standard Time)	Spain	Monroe000db94009e4	apu1d4	Google Maps
61 ⊘	Active	Testing	Fri Feb 24 2017 13:18:26 GMT+0100 (Romance Standard Time)	Spain	Monroe000db940030c	apu1d4	Google Maps
62 ✓	Active	Testing	Tue Feb 28 2017 18:04:36 GMT+0100 (Romance Standard Time)	Spain	Monroe000db94002bc	apu1d4	Google Maps
193 ✓	Active	Testing	Tue Feb 28 2017 18:04:30 GMT+0100 (Romance Standard Time)	Spain	Monroe000db940e07c	apu1d4	n/a
194 ✓	Active	Testing	Tue Feb 28 2017 18:04:42 GMT+0100 (Romance Standard Time)	Spain	Monroe000db940e09c	apu1d4	n/a
195 ✓	Active	Testing	Tue Feb 28 2017 18:05:07 GMT+0100 (Romance Standard Time)	Spain	Monroe000db940e0a8	apu1d4	n/a
196 ✓	Active	Testing	Tue Feb 28 2017 18:04:23 GMT+0100 (Romance Standard Time)	Spain	Monroe000db940e084	apu1d4	n/a
203 ✓	Active	Testing	Tue Feb 28 2017 18:04:31 GMT+0100 (Romance Standard Time)	Spain	Monroe000db940dc34	apu1d4	n/a
205 ✓	Active	Testing	Tue Feb 28 2017 18:04:44 GMT+0100 (Romance Standard Time)	Spain	Monroe000db940e1fc	apu1d4	n/a

Map of node states

🟩 57 Spain 🔍 🟩 59 Spain 🔍 🟩 60 Spain ⌂ 🟥 61 Spain 🔍 🟩 62 Spain 🔍
🟩 193 Spain 🔍 🟩 194 Spain 🔍 🟩 195 Spain 🔍 🟩 196 Spain 🔍
🟩 203 Spain 🔍 🟩 205 Spain 🔍

Legend:

🟧 Maintenance 🟩 Can schedule 🟥 Cannot schedule

🔍 Testing node
⌂ Deployed node (static)
Deployed node (mobile)

Figure 10: Status of the MONROE nodes. The screen capture shows all types of nodes in Spain. The green approval sign ("thumbs-up") close to the node IDs indicates that this node is capable of executing experiments. Clicking on the "Location" link for a node opens a Google Maps page showing the location of the node. Finally, the bottom part of the screen shows a "map" of nodes that allows users and MONROE administrators to quickly identify available (and problematic) nodes in the platform.

6 Monitoring node status

The user can check the state of the nodes under the tab "Resources." Figure 10 shows an example of the information that is supplied.

7 MONROE templates, examples and default experiments

This section details the template for building MONROE experiments, the experiments that run as part of the default MONROE platform and several additional examples that can be directly used or that can serve as the basis for new ones. The source code for the examples is publicly available at `https://github.com/MONROE-PROJECT/Experiments`.

7.1 Example template

This experiment template provides an extensive example to show the capabilities of the MONROE platform. The experiment will download a url (file) over http using `curl` from a configurable operator while at the same time recording the GPS positions of the node. If the operator is not available at the time of execution, the experiment will fail.

7.1.1 Usage

The configuration values can be supplied as a JSON string in the "Additional options" field of the web user interface. This allows to specify a different set of parameters for each execution of the experiment.

The values of the configuration parameters can be read by the experiment from the `/monroe/config` file. The following text shows a configuration file with per-execution ("additional options" field) options:[3]

```
{
"stop": 1486653420,
"start": 1486653120,
"traffic": 1048576,
"script": "mikepeon/mike-depurar",
"shared": 0,
"storage": 134217728,
"resultsQuota": 0,
"guid": "sha256:3796f833f55c8dbca7e9845ea06120ccebec85c2770c0de2deb57509300efa44.165695.48.1",
"option1": "value1",
"option2": "value2",
"nodeid": "48"
}
```

The default configuration values are as follows:

```
{
# The following values are specific to the monroe platform
"guid": "no.guid.in.config.file",               # Created by the scheduler
"nodeid": "no.nodeid.in.config.file",           # Created by the scheduler
"storage": 104857600,                           # Created by the scheduler
"traffic": 104857600,                           # Created by the scheduler
"script": "jonakarl/experiment-template",       # Created by the scheduler
"zmqport": "tcp://172.17.0.1:5556",
"modem_metadata_topic": "MONROE.META.DEVICE.MODEM",
"gps_metadata_topic": "MONROE.META.DEVICE.GPS",
# "dataversion": 1,                             # Version of the experiment
# "dataid": "MONROE.EXP.JONAKARL.TEMPLATE",     # Name of the experiement
"meta_grace": 120,                              # Grace period to wait for interface metadata
"exp_grace": 120,                               # Grace period before killing experiment
"meta_interval_check": 5,                       # Interval to check if interface is up
"verbosity": 2,                                 # 0="Mute", 1=error, 2=information, 3=verbose
"resultdir": "/monroe/results/",
# These values are specic for this experiment
"operator": "Telenor SE",
"url": "http://193.10.227.25/test/1000M.zip",
"size": 3*1024 - 1,                             # The maximum size in Kbytes to download
"time": 3600                                    # The maximum time in seconds for a download
}
```

[3]Entries in the `/monroe/config` file may appear in different order.

The download will abort when either size OR time OR actual size of the “url” is downloaded. All debug/error information will be printed on `stdout`, depending on the “verbosity” variable.

7.1.2 Requirements

The following directories and files must exist and have read and write permissions for the user/process running the container:

- `/monroe/config`, supplied by the scheduler in the nodes.
- “`resultdir`,” according to the values supplied in the configuration string or the default ones (Section 7.1.1).

7.1.3 Output

The experiment will execute a statement similar to running `curl` with the following command line:

```
curl -o /dev/null --raw --silent --write-out "{ remote: %{remote_ip}:%{remote_port},
size: %{size_download}, speed: %{speed_download}, time: %{time_total},
time_download: %{time_starttransfer} }" --interface eth0 --max-time 100 --range 0-100
http://193.10.227.25/test/1000M.zip
```

The experiment will produce a single-line JSON object similar to this (pretty printed to improve readability):

```
{
"Bytes": 30720000,
"DataId": "313.123213.123123.123123",
"DataVersion": 1,
"DownloadTime": 2.716,
"GPSPositions": [
{
"Altitude": 225.0,
"DataId": "MONROE.META.DEVICE.GPS",
"DataVersion": 1,
"Latitude": 59.404697,
"Longitude": 13.581558,
"NMEA": "$GPGGA,094832.0,5924.281896,N,01334.893500,E,1,05,1.6,225.0,M,35.0,M,,*5D\r\n",
"SatelliteCount": 5,
"SequenceNumber": 14,
"Timestamp": 1465551728
},
{
"DataId": "MONROE.META.DEVICE.GPS",
"DataVersion": 1,
"Latitude": 59.404697,
"Longitude": 13.581558,
"NMEA": "$GPRMC,094832.0,A,5924.281896,N,01334.893500,E,0.0,,100616,0.0,E,A*2B\r\n",
"SequenceNumber": 15,
"Timestamp": 1465551728
}
],
"Guid": "sha256:15979bc2e2449b0011826c2bb8668df980da88221af3fc7916cb2eba4f2296c1.0.45.15",
"Host": "193.10.227.25",
"Iccid": "89460850007006922138",
"InterfaceName": "usb0",
"NodeId": "45",
"Operator": "Telenor SE",
"Port": "80",
"SequenceNumber": 1,
"SetupTime": 0.004,
"Speed": 11295189.0,
"TimeStamp": 1465551458.099917,
"TotalTime": 2.72
}
```

7.1.4 Overview of the code structure

The experiment consists of one main process and two sub processes, where one process listens to modem and gps information, and the other executes the experiment. The main process supervises the execution of its two children.

Information sharing between processes. Information is shared between processes via two thread-safe data structures (i.e., a Python "Manager" object). Regarding modem information, the latest metadata update (for the specified operator) is stored in a dictionary. The GPS information is continuously appended to a list as it is received.

The metadata sub-process. This process listens to GPS and modem messages sent on the ZeroMQ bus and updates the shared data structures.

The experiment sub-process. This process reads entries from the shared data structures, runs the experiment and saves its result when finished.

7.2 Docker miscellaneous usage notes

- List running containers:

  ```
  docker ps
  ```

- Debug shell:

  ```
  docker run -i -t --entrypoint bash --net=host template
  ```

- Normal execution with output to `stdout`:

  ```
  docker run -i -t --net=host template
  ```

- Attach to a running container (with shell):

  ```
  docker exec -i -t [container runtime name] bash
  ```

- Get container logs (`stderr` and `stdout`):

  ```
  docker logs [container runtime name]
  ```

7.3 Experiment: ping

This background experiment runs continuously an RTT estimate on each MBB operator on the node (one independent experiment is run per interface). The experiments measure IP RTT by continuously sending ping packets to a configurable server (by default `8.8.8.8`, Google's public DNS server). The experiment will send one "Echo Request" (ICMP type 8) packet per second over the specified interface until aborted. RTT is measured as the time between the echo request is sent and the echo reply (ICMP type 0) is received from the server. The experiment runs on all interfaces in parallel.

7.3.1 Usage

The experiment is designed to run as a Docker container and will not attempt to do any active network configuration. If the specified interface does not exist (i.e., is not up) when the experiment starts, it will immediately exit.

The default parameter values are:

```
{
"guid": "no.guid.in.config.file",               # Created by the scheduler
"zmqport": "tcp://172.17.0.1:5556",
"nodeid": "fake.nodeid",
"modem_metadata_topic": "MONROE.META.DEVICE.MODEM",
"server": "8.8.8.8",                            # ping target
"interval": 1000,                               # time in ms between successive packets
"dataversion": 2,
"dataid": "MONROE.EXP.PING",
"meta_grace": 120,                              # Grace period to wait for interface metadata
"ifup_interval_check": 5,                       # Interval to check if interface is up
"export_interval": 5.0,
"verbosity": 2,                                 # 0="Mute", 1=error, 2=Information, 3=verbose
"resultdir": "/monroe/results/",
"modeminterfacename": "InternalInterface",
"interfacename": "eth0",                        # Interface to run the experiment on
"interfaces_without_metadata": ["eth0", "wlan0"] # Manual metadata on these interfaces
}
```

All debug/error information will be printed on `stdout` depending on the value of the "verbosity" parameter.

7.3.2 Requirements

The following directories and files must exist and have read and write permissions for the user/process running the container:

- `/monroe/config`, supplied by the scheduler in the nodes.
- "`resultdir`," according to the values supplied in the configuration string or the default ones (Section 7.1.1).

7.3.3 Output

The experiment will execute a statement similar to running `fping` with the following command line:

```
fping -I eth0 -D -c 1 -p 1000 -l 8.8.8.8
```

The experiment will produce one of the two following single-line JSON objects, depending on whether it got a reply form the server or not. If a reply was received:

```
{
"Guid": "313.123213.123123.123123", # exp_config['guid']
"Timestamp": 23123.1212, # time.time()
"Iccid": 2332323, # meta_info["ICCID"]
"Operator": "Telia", # meta_info["Operator"]
"NodeId" : "9", # exp_config['nodeid']
"DataId": "MONROE.EXP.PING",
"DataVersion": 2,
"SequenceNumber": 70,
"Rtt": 6.47,
"Bytes": 84,
"Host": "8.8.8.8",
}
```

If the reply was not received (Bytes and RRR values are not present):

```
{
"Guid": "313.123213.123123.123123", # exp_config['guid']
"Timestamp": 23123.1212, # time.time()
"Iccid": 2332323, # meta_info["ICCID"]
"Operator": "Telia", # meta_info["Operator"]
"NodeId" : "9", # exp_config['nodeid']
"DataId": "MONROE.EXP.PING",
"DataVersion": 2,
"SequenceNumber": 71,
"Host": "8.8.8.8",
}
```

7.4 Experiment: http_download

This is a periodically scheduled experiment that monitors the download speed of each MBB operator on the node. The experiment will, over each MBB operator in sequence, download the specified url (file) with `curl` (http), presenting one result per interface. The MONROE experiment template described in Section 7.1 corresponds to this experiment, therefore, it is not further detailed here.

7.5 Experiment: Tstat & mPlane

The mPlane protocol provides control and data interchange for passive and active network measurement tasks. It is built around a simple workflow that can interact with different frameworks to provide the results of the measurements. This package includes an mPlane proxy and generic configuration files for Tstat.

mPlane captures traffic flow on all interfaces by means of the Tstat (http://tstat.polito.it/) probe. The mPlane container is always running as one of the default experiments on all MONROE nodes. The Tstat passive traces are stored locally on the node and are accessible by the experimenters. A detailed description and the source code are available on github (https://github.com/MONROE-PROJECT/mPlane).

Tstat RRD logs and the compressed log are stored in the node at `/experiments/monroe/mplane`. Tstat logs are transfered to the MONROE server and imported into MONROE's (Cassandra) database. The structure of the database tables is available on github (https://github.com/MONROE-PROJECT/Database/blob/master/db_schema.cql).

During experiment execution, the last three Tstat logs are shared with the experiment at `/monroe/tstat`. Therefore, MONROE users can access the passive traces collected by Tstat during their experiments.

The data collected for a subset of the most relevant metrics for the HTTP experiments are visualized by the MONROE visualization tool. And example of the metrics contained in the Tstat logs can be seen here: http://213.182.68.136:8080/#/experiment/tstat.

7.5.1 Requirements

The script must have access to `/nodeid` and run `get_nodeid`.

7.5.2 Usage

Create your docker image normally and execute the container with the following command line:

```
docker run -i -t --net=host -d -v /mplane:/monroe/results -v /tstat:/monroe/tstat
-v /etc/nodeid:/nodeid:ro monroe/mplane
```

7.6 MONROE example: helloworld

This experiment provides an easy example for using the configuration options from the scheduler, listen to and record the metadata stream (e.g., GPS and operator information), and show the experiment log functionality on a MONROE node. The experiment listens to the metadata stream and records the `nr_of_messages` first messages. The metadata messages are saved in JSON format with a custom field ("Hello") in the output directory. Additionally, the experiment prints out some debugging messages to show how these messages are logged and later retrieved via the web user interface.

7.6.1 Usage

The experiment is configured with a JSON string introduced via the "Additional options" field in the web user interface. The configurable parameters and their default values are:

```
{
"zmqport": "tcp://172.17.0.1:5556",
"nodeid": "fake.nodeid",              # Needs to be overriden
"metadata_topic": "MONROE.META",
"verbosity": 2,                       # 0 = "Mute", 1=error, 2=Information, 3=verbose
"resultdir": "/monroe/results/",
"nr_of_messages": 3
}
```

7.6.2 Requirements

The following directories and files must exist and have read and write permissions for the user/process running the container:

- `/monroe/config`, supplied by the scheduler in the nodes.
- "`resultdir`," according to the values supplied in the configuration string or the default ones (Section 7.6.1).

7.6.3 Output

The experiment will produce a single-line JSON object similar to the following ones, depending on the metadata received ("pretty printed" here to improve readability):

```
{
"DataId": "MONROE.META.NODE.SENSOR",
"DataVersion": 1,
"SequenceNumber": 58602,
"Timestamp": 1465888420,
"NodeId": "9",
"Hello": "World"
}
```

The log file will contain records similar to these ones:

```
[2017-02-07 09:53:27.190338] Hello: Default config {
"metadata_topic": "MONROE.META",
"nodeid": "fake.nodeid",
"nr_of_messages": 3,
"resultdir": "/monroe/results/",
"verbosity": 2,
"zmqport": "tcp://172.17.0.1:5556"
}
[2017-02-07 09:53:27.20000] Hello: Start recording messages with configuration {
"metadata_topic": "MONROE.META",
"nodeid": "fake.nodeid",
"nr_of_messages": 3,
"resultdir": "/monroe/results/",
"verbosity": 2,
"zmqport": "tcp://172.17.0.1:5556"
}
[[2017-02-07 09:53:27.30000] Received message 1 with topic : MONROE.META.NODE.SENSOR
{
"DataId": "MONROE.META.NODE.SENSOR",
"DataVersion": 1,
"SequenceNumber": 58602,
"Timestamp": 1465888420,
"NodeId": "9",
"Hello": "World"
}
. # And so on for each metadata message received until the configured value of metadata messages
.
.
[2017-02-07 09:53:27.40000] Hello : Finished the experiment
```

7.7 MONROE example: paris-traceroute

This example showcases how to use the MONROE-modified version of paris-traceroute inside a container. The binary of this tool is included in the base image of MONROE.

The original version of paris-traceroute has no option to choose which interface should be used. In this version, flags to set the interface and source IP of the transmitted packets have been added. Setting the interface is obligatory; if it is not set, the program will crash (by design), since if the interface were chosen automatically, it would probably not be what the experimenter intended to use. The source IP flag is optional. Just setting the IP flag to the IP of an interface without setting the interface flag will not work either.

This is done on purpose as well, as it might be possible for multiple interfaces to have the same IP within the MONROE network namespace. If the IP flag is not set, the source IP is set to the IP of the chosen interface.

7.7.1 Usage (inside a MONROE container)

The parameters of this experiment are provided as "Additional options" in the scheduling web interface. The following JSON string is an example of the additional options that can be passed to this container:

```
"interfaces": ["op1", "op2"], "targets": ["8.8.8.8", "www.uc3m.es"],
"traceAlgos": ["exh"], "protocol": "udp"
```

Flags:

```
-C  --nodeIPArgument             Source IP
-O  --nodeInterfaceArgument      Source interface (mandatory)
```

The paris-traceroute binary can be executed (as any normal Linux command) either without specifying a traceroute algorithm to perform a "simple" traceroute (similar to the output of the ordinary traceroute command), or with the flags `-n -a exh`, to perform an exhaustive traceroute. Exhaustive traceroutes provide more detailed and accurate paths between the host (MONROE node) and the target server that are able to detect, among others, the presence of load balancers, which create multiple paths between host and target.

7.7.2 Output

The experiment output is a text file:

```
root@b59e69a56297:/# paris-traceroute -O op2 -C 192.168.1.127 8.8.8.8
traceroute [(192.168.1.127:33456) -> (8.8.8.8:33457)], protocol udp, algo hopbyhop, duration 18 s
1  192.168.1.1 (192.168.1.1)  2.946 ms   0.553 ms   0.559 ms
2  * * *
3  10.133.17.29 (10.133.17.29)  83.259 ms   136.577 ms   82.050 ms
4  10.133.17.14 (10.133.17.14)  78.783 ms   131.510 ms   79.231 ms
5  10.133.17.236 (10.133.17.236)  84.243 ms   133.024 ms   79.785 ms
6  10.133.17.3 (10.133.17.3)  81.543 ms   139.381 ms   100.263 ms
7  83.224.40.186 (83.224.40.186)  89.319 ms   188.926 ms   179.963 ms
MPLS Label 24703 TTL=254
8  83.224.40.185 (83.224.40.185)  82.710 ms   172.438 ms   147.020 ms
9  85.205.14.105 (85.205.14.105)  85.179 ms   137.514 ms   125.869 ms
10  72.14.223.169 (72.14.223.169)  85.609 ms   137.363 ms   118.063 ms
11  216.239.47.128 (216.239.47.128)  79.567 ms   146.356 ms   145.285 ms
12  209.85.243.33 (209.85.243.33)  129.615 ms   198.938 ms   269.407 ms
MPLS Label 568892 TTL=1
13  64.233.174.143 (64.233.174.143)  108.599 ms   185.810 ms   246.661 ms
MPLS Label 692130 TTL=1
14  108.170.234.47 (108.170.234.47)  111.645 ms   825.615 ms   1424.942 ms
15  * * *
16  google-public-dns-a.google.com (8.8.8.8)  103.087 ms !T2   166.279 ms !T2   224.649 ms !T2
```

7.7.3 Additional remarks

Paris-traceroute instances should be run sequentially and preferably when the node is generating little traffic in general because it uses raw packet capture to detect the replies from intermediate nodes and background traffic might interfere with this process.

7.8 MONROE example: headlessbrowsing

This experiment evaluates the performance of different HTTP protocols (HTTP1.1, HTTP1.1/TLS, HTTP2) using the headless Firefox browser. The experiment uses the Selenium browser-automation framework, which enables execution of web-browsing automation tests in different browsers such as Firefox and Chrome. The Selenium web-driver is used for Firefox. For a given url, HTTP protocol and source network interface, Selenium launches the native Firefox browser to visit that url.

7.8.1 Output

This experiment generates an HTTP ARchive (HAR) file during the download of a target url that helps to find afterwards the impact of different web-page features on its overall Page Load Time (PLT).

The experiment generates a single JSON file such as:

```
{
"DataId":"MONROE.EXP.FIREFOX.HEADLESS.BROWSING",
"ping_min":" 55.6",
"ping_max":"56.8",
"NumObjects":6,
"InterfaceName":"usb2",
"Web load time":196,
"PageSize":35641,
"DataVersion":1,
"Timestamp":1481536829.0814,
"NWMCCMNC":22210,
"Objects":[
{
"objectSize":1951,
"mimeType":"image/png",
"startedDateTime":"2016-12-12T10:00:21.293+00:00",
"url":"https://www.wikipedia.org/portal/wikipedia.org/assets/img/Wikipedia_wordmark.png",
"timings":{"receive":1, "send":0, "connect":1, "dns":0, "blocked":0, "wait":60},
"time":62
},
{
"objectSize":13196,
"mimeType":"image/png",
"startedDateTime":"2016-12-12T10:00:21.294+00:00",
"url":"https://www.wikipedia.org/portal/wikipedia.org/assets/img/Wikipedia-logo-v2.png",
"timings":{"receive":52, "send":3, "connect":59, "dns":2, "blocked":0, "wait":53 },
"time":169
},
{
"objectSize":9425,
"mimeType":"application/javascript",
"startedDateTime":"2016-12-12T10:00:21.295+00:00",
"url":"https://www.wikipedia.org/portal/wikipedia.org/assets/js/index-abc278face.js",
"timings":{"receive":3, "send":0, "connect":120, "dns":0, "blocked":0, "wait":65},
"time":188
},
{
"objectSize":1164,
"mimeType":"application/javascript",
"startedDateTime":"2016-12-12T10:00:21.296+00:00",
"url":"https://www.wikipedia.org/portal/wikipedia.org/assets/js/gt-ie9-c84bf66d33.js",
"timings":{"receive":0, "send":1, "connect":64, "dns":2, "blocked":0, "wait":70 },
"time":137
},
{
"objectSize":1590,
"mimeType":"image/png",
"startedDateTime":"2016-12-12T10:00:21.381+00:00",
"url":"https://www.wikipedia.org/portal/wikipedia.org/assets/img/sprite-icons.png?
27378e2bb51199321b32dd1ac3f5cd755adc21a5",
"timings":{"receive":1, "send":0, "connect":1, "dns":0, "blocked":0, "wait":49 },
"time":51
},
{
"objectSize":8315,
"mimeType":"image/png",
"startedDateTime":"2016-12-12T10:00:21.425+00:00",
"url":"https://www.wikipedia.org/portal/wikipedia.org/assets/img/sprite-project-logos.png?
dea6426c061216dfcba1d2d57d33f4ee315df1c2",
"timings":{"receive":2, "send":0, "connect":8, "dns":0, "blocked":0, "wait":54 },
"time":64
} ],
"IPAddress":"2.43.181.254",
"IMSIMCCMNC":22210,
"tracedRoutes":["192.168.96.1", "193.10.227.25", "xx.xx.xx.xx" ..... "192.168.96.1" ],
"InternalInterface":"op0",
"NodeId":"41",
"ping_exp":1,
"Protocol":"HTTP1.1",
"SequenceNumber":1,
"url":"www.wikipedia.org",
"ping_avg":"56.2",
"InternalIPAddress":"192.168.96.123",
"Operator":"voda IT",
"Iccid":"8939104160000392116"
}
```

7.9 MONROE example: pReplay

The pReplay experiment replays the dependency graph of a web site.

The traversal begins with the first activity: Loading the root HTML. After building the dependency graph, it acts for each task whose dependencies have already been met. For network tasks, it makes a request for the corresponding url; correspondingly, for computation activities, it waits for the amount of time mentioned in the graph. Once a particular activity is finished, pReplay checks if any activities depending on that one have already met all of their dependencies and must thus be triggered. pReplay walks through the dependency graph until all activities in the graph have been visited.

7.9.1 Usage

Execute pReplay on a command line inside a container as with any other Linux command:

```
./pReplay interface_name server testfile [http|https|http2] [max-connections] [cookie-size]
```

Parameters:

- **interface_name:** Source interface for outgoing traffic.

- **server:** DNS name or IP address.
- **testfile:** Relative path to test file in JSON format.
- **protocol:**
  - http: http 1.1
  - https: http 1.1 with SSL
  - http2: http 2
- **max-connections:** Maximum amount of concurrent connections.
- **cookie-size:** Size of cookie — works with http1 only.

7.10 MONROE example: astream

AStream is a Python based emulated video player to evaluate the performance of the DASH bitrate adaptation algorithms. The supported rate adaptation algorithms are:

- Basic adaptation.
- Segment Aware Rate Adaptation (SARA) [**?**].
- Buffer-Based Rate Adaptation (Netflix) [**?**].

7.10.1 Usage

The experimenter can choose the rate adaptation algorithm passing a JSON string to the scheduler through the user interface (e.g., `"playback":"NETFLIX"`). The default is the basic adaptation scheme. Additionally, the user can specify the target MPD file to play (e.g., `"mpd_file":"http://128.39.37.161:8080/BigBuck-Bunny_4s.mpd"`) and the number of segments to retrieve (e.g., `"segment_limit":10`).

7.10.2 Output

The astream container outputs two log files:

1. Buffer logs: Epoch time, current playback time, current buffer size (in segments), current playback state.
2. Playback logs: Epoch time, playback time, segment number, segment size, playback bitrate, segment duration, weighted harmonic mean average download rate.

7.11 MONROE example: udpbwestimator

Udpbwestimator is an experiment setup to estimate available bandwidth for a particular network interface. It consists of two applications, a receiver and a traffic generator (server). The receiver initiates connections and requests the server for traffic. Then, every second, the server sends a burst of UDP packets back to back to the receiver, which follows the packet arrival times and estimates the available bandwidth.

7.11.1 Usage

The receiver accepts the following command line parameters:

-c : Number of back-to-back packets to be sent in each second.
-b : Number of bursts to be sent.
-l : Payload length in bytes.

-s : Source IP to bind to.
-o : Source port.
-d : Destination IP.
-p : Destination port.
-w : Optional, filename for writing the packet arrival times.

7.11.2 Output

The experiment will produce a single-line JSON object similar to the following:

```
{
"CID" : 33346602,
"DataId" : "MONROE.EXP.UDPBWESTIMATOR",
"DataVersion" : 1,
"DeviceMode" : 5,
"DeviceState" : 3,
"Guid" : "sha256:872af8c8b8f1635be6936a111b5fa838071e6f42cb317e9db1d9bb0c7db31425.93321.204.1",
"IMEI" : "864154023639966",
"IMSI" : "240016025247086",
"IMSIMCCMNC" : 24001,
"IPAddress" : "78.79.63.124",
"Iccid" : "89460120151010468086",
"InterfaceName" : "usb0",
"InternalIPAddress" : "192.168.68.118",
"InternalInterface" : "op1",
"LAC" : 2806,
"NWMCCMNC" : 24202,
"NodeId" : "204",
"Operator" : "NetCom",
"RSRP" : -72,
"RSRQ" : -7,
"RSSI" : -49,
"SequenceNumber" : 1,
"Timestamp" : 1479312368.633218,
"bw" : "48.41 38.98 36.44 50.00 30.20 45.21 47.02 37.89 44.37 28.90 25.91 38.57 48.74
39.94 43.37 37.94 43.81 39.60 52.00 47.55 48.20 34.85 41.44 47.60 57.26 46.11
45.66 52.04 37.43 49.67 33.56 50.35 41.11 51.63 45.33 104.01 45.73 49.95 50.37
38.57 29.45 50.95 54.95 45.42 47.13 34.30 46.10 103.68 79.75 45.72 52.03 30.38
50.21 36.96 71.51 54.66 39.26 44.12 45.18 39.93"
}
```

7.12 MONROE example: traceroute_background_experiment

Performs traceroute periodically to various targets. This experiment is meant to be run in the background and can be run in parallel with experiments of other users. It uses the default traceroute binary distributed by the Debian repositories.

Each traceroute produces a text file that is parsed by `outputParser.py` to generate the JSON output of this experiment. The JSON file is then imported into the MONROE database.

7.12.1 Usage

To reduce experiment duration, the traceroutes can be run in parallel. The number of parallel traceroute instances is dictated by the `maxNumberOfTotalTracerouteInstances` parameter. It is possible to parallelize on a per-interface basis (i.e., `maxNumberOfTotalTracerouteInstances` per interface) or per the whole experiment (i.e., `maxNumberOfTotalTracerouteInstances` total in the experiment instance spread among all the interfaces). This behavior is controlled by the `executionMode` parameter. The available options are: serially, serialPerInterface and parallel.

Additionally, a flag can be provided to choose the protocol of the probes: default, udp, tcp and icmp.

The parameters of this experiment are provided as "Additional options" in the web user interface. An example JSON string that can be used with this container as additional options is:

```
"interfaces": ["op0", "op1", "op2"], "targets": ["www.ntua.gr", "www.uc3m.es", "Google.com",
"Facebook.com", "Youtube.com", "Baidu.com", "Yahoo.com", "Amazon.com", "Wikipedia.org",
"audio-ec.spotify.com", "mme.whatsapp.net", "sync.liverail.com", "ds.serving-sys.com",
"instagramstatic-a.akamaihd.net"], "maxNumberOfTotalTracerouteInstances": 5,
"executionMode": "parallel"
```

7.13 Other containers in the repositories

Our public repositories contain the source code for other Docker containers that perform varied tasks in the nodes. Although they are not intended as examples, users can take a look into them to gain a deeper understanding of the platform configuration.

7.13.1 Container: metadata-subscriber

The subscriber is designed to listen to ZMQ messages send out by the metadata-multicaster. The subscriber attaches to a configurable ZeroMQ socket and listens to all messages that begin with the topic "MONROE.META," except the ones whose topic ends with ".UPDATE" (rebroadcasts) and/or begins with "MONROE.META.DEVICE.CONNECTIVITY." as these are redundant. All messages are updated with NodeId, but are otherwise saved verbatim as a JSON formatted file suitable for later import in the MONROE databases.

7.13.2 Container: tunnelbox-server

This container acts as an SSH reverese tunnel endpoint that clients can use to directly connect to their experiment containers (on any MONROE node). The purpose is to provide experimenters an interactive way of accessing an experiment running on a real MONROE node during development or debugging. The client has to supply its own public SSH key to the experiment container using the web user interface. The web user interface provides further instructions (SSH command line) to connect to the experiment container using the provided key.

7.13.3 Container: monroe_base

This is the container upon which all user experiments *must* be built. The container is based on Debian "jessie" with (MONROE) common experiment tools added. For a list of the tools currently installed see the folder `monroe_base.docker` in our repositories.

8 List of known bugs and issues

- In general, Firefox does not render the date-time picker correctly. You will have to either enter the dates and times manually or use Chrome.

- Container deployment can take several minutes, particularly for nodes without an Ethernet management connection (e.g., mobile nodes in trains or buses). When scheduling an experiment, the user has to take into account the time needed for the deployment. The system will not automatically take care of this at this moment.
- Similarly, the button “Check availability” returns the earliest available slot. However, it does not account for the time needed to deploy the container. The user must manually account for that.
- Checking the option “ASAP” to schedule an experiment as soon as possible may fail due to lack of time to deploy the container. The system does add some slack in this case, but its length may need some adjustment according to the type of nodes and MBB characteristics.

A List of packages installed in monroe/base

Table 8: List of packages installed in `monroe/base`.

Name	Version	Architecture	Description
acl	2.2.52-2	amd64	Access control list utilities
adduser	3.113+nmu3	all	add and remove users and groups
apache2	2.4.10-10+deb8u4	amd64	Apache HTTP Server
apache2-bin	2.4.10-10+deb8u4	amd64	Apache HTTP Server (modules and other binary files)
apache2-data	2.4.10-10+deb8u4	all	Apache HTTP Server (common files)
apache2-utils	2.4.10-10+deb8u4	amd64	Apache HTTP Server (utility programs for web servers)
apt	1.0.9.8.3	amd64	commandline package manager
base-files	8+deb8u4	amd64	Debian base system miscellaneous files
base-passwd	3.5.37	amd64	Debian base system master password and group files
bash	4.3-11+b1	amd64	GNU Bourne Again SHell
bind9-host	1:9.9.5.dfsg-9+deb8u6	amd64	Version of ’host’ bundled with BIND 9.X
blt	2.5.3+dfsg-1	amd64	graphics extension library for Tcl/Tk - run-time
bsdutils	1:2.25.2-6	amd64	basic utilities from 4.4BSD-Lite
ca-certificates	20141019+deb8u1	all	Common CA certificates
ca-certificates-java	20140324	all	Common CA certificates (JKS keystore)
coreutils	8.23-4	amd64	GNU core utilities
curl	7.38.0-4+deb8u3	amd64	command line tool for transferring data with URL syntax
d-itg	2.8.1-r1023-3	amd64	Distributed Internet Traffic Generator
dash	0.5.7-4+b1	amd64	POSIX-compliant shell
dbus	1.8.20-0+deb8u1	amd64	simple interprocess messaging system (daemon and utilities)
debconf	1.5.56	all	Debian configuration management system
debconf-i18n	1.5.56	all	full internationalization support for debconf
debian-archive-keyring	2014.3	all	GnuPG archive keys of the Debian archive
debianutils	4.4+b1	amd64	Miscellaneous utilities specific to Debian
default-jre-headless	2:1.7-52	amd64	Standard Java or Java compatible Runtime (headless)
diffutils	1:3.3-1+b1	amd64	File comparison utilities
dmsetup	2:1.02.90-2.2	amd64	Linux Kernel Device Mapper userspace library
dnsutils	1:9.9.5.dfsg-9+deb8u6	amd64	Clients provided with BIND
dpkg	1.17.26	amd64	Debian package management system
e2fslibs:amd64	1.42.12-1.1	amd64	ext2/ext3/ext4 file system libraries
e2fsprogs	1.42.12-1.1	amd64	ext2/ext3/ext4 file system utilities
echoping	6.0.2-8	amd64	Small test tool for TCP servers
file	1:5.22+15-2+deb8u1	amd64	Determines file type using "magic" numbers
findutils	4.4.2-9+b1	amd64	utilities for finding files–find, xargs
flent	0.14.0-1	all	The FLExible Network Tester
fontconfig	2.11.0-6.3	amd64	generic font configuration library - support binaries
fontconfig-config	2.11.0-6.3	all	generic font configuration library - configuration
fonts-dejavu-core	2.34-1	all	Vera font family derivate with additional characters
fonts-lyx	2.1.2-2	all	TrueType versions of some TeX fonts used by LyX
fping	3.10-2	amd64	sends ICMP ECHO_REQUEST packets to network hosts
gcc-4.8-base:amd64	4.8.4-1	amd64	GCC, the GNU Compiler Collection (base package)
gcc-4.9-base:amd64	4.9.2-10	amd64	GCC, the GNU Compiler Collection (base package)

Table 8: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
geoip-database	20150317-1	all	IP lookup command line tools that use the GeoIP library (country database)
gnupg	1.4.18-7+deb8u1	amd64	GNU privacy guard - a free PGP replacement
gpgv	1.4.18-7+deb8u1	amd64	GNU privacy guard - signature verification tool
gpsd	3.11-3	amd64	Global Positioning System - daemon
gpslogger-oml2	2.11.0-mytestbed2	amd64	Record and store GPS measurements using OML
grep	2.20-4.1	amd64	GNU grep, egrep and fgrep
gstreamer1.0-plugins-base:amd64	1.4.4-2	amd64	GStreamer plugins from the "base" set
gzip	1.6-4	amd64	GNU compression utilities
hicolor-icon-theme	0.13-1	all	default fallback theme for FreeDesktop.org icon themes
hostname	3.15	amd64	utility to set/show the host name or domain name
httperf-oml2	2.11.0-mytestbed2	amd64	HTTP server performance tester, with OML support
httping	1.5.8-1	amd64	ping-like program for http-requests
inetutils-ping	2:1.9.2.39.3a460-3	amd64	ICMP echo tool
init	1.22	amd64	System-V-like init utilities - metapackage
init-system-helpers	1.22	all	helper tools for all init systems
initscripts	2.88dsf-59	amd64	scripts for initializing and shutting down the system
insserv	1.14.0-5	amd64	boot sequence organizer using LSB init.d script dependency information
iperf	2.0.5+dfsg1-2	amd64	Internet Protocol bandwidth measuring tool
iperf-oml2	2.11.0-mytestbed2	amd64	Internet Protocol bandwidth measuring tool, with OML support
iperf3	3.0.7-1	amd64	Internet Protocol bandwidth measuring tool
iproute2	3.16.0-2	amd64	networking and traffic control tools
iptables	1.4.21-2+b1	amd64	administration tools for packet filtering and NAT
iso-codes	3.57-1	all	ISO language, territory, currency, script codes and their translations
java-common	0.52	all	Base of all Java packages
javascript-common	11	all	Base support for JavaScript library packages
krb5-locales	1.12.1+dfsg-19+deb8u2	all	Internationalization support for MIT Kerberos
libacl1:amd64	2.2.52-2	amd64	Access control list shared library
libalgorithm-c3-perl	0.09-1	all	Perl module for merging hierarchies using the C3 algorithm
libapr1:amd64	1.5.1-3	amd64	Apache Portable Runtime Library
libaprutil1:amd64	1.5.4-1	amd64	Apache Portable Runtime Utility Library
libaprutil1-dbd-sqlite3:amd64	1.5.4-1	amd64	Apache Portable Runtime Utility Library - SQLite3 Driver
libaprutil1-ldap:amd64	1.5.4-1	amd64	Apache Portable Runtime Utility Library - LDAP Driver
libapt-pkg4.12:amd64	1.0.9.8.3	amd64	package management runtime library
libarchive-extract-perl	0.72-1	all	generic archive extracting module
libasound2:amd64	1.0.28-1	amd64	shared library for ALSA applications
libasound2-data	1.0.28-1	all	Configuration files and profiles for ALSA drivers
libasyncns0:amd64	0.8-5	amd64	Asynchronous name service query library
libatk1.0-0:amd64	2.14.0-1	amd64	ATK accessibility toolkit
libatk1.0-data	2.14.0-1	all	Common files for the ATK accessibility toolkit
libattr1:amd64	1:2.4.47-2	amd64	Extended attribute shared library
libaudio2:amd64	1.9.4-3	amd64	Network Audio System - shared libraries
libaudit-common	1:2.4-1	all	Dynamic library for security auditing - common files
libaudit1:amd64	1:2.4-1+b1	amd64	Dynamic library for security auditing
libauthen-sasl-perl	2.1600-1	all	Authen::SASL - SASL Authentication framework
libavahi-client3:amd64	0.6.31-5	amd64	Avahi client library
libavahi-common-data:amd64	0.6.31-5	amd64	Avahi common data files
libavahi-common3:amd64	0.6.31-5	amd64	Avahi common library
libbind9-90	1:9.9.5.dfsg-9+deb8u6	amd64	BIND9 Shared Library used by BIND
libblas-common	1.2.20110419-10	amd64	Dependency package for all BLAS implementations
libblas3	1.2.20110419-10	amd64	Basic Linear Algebra Reference implementations, shared library
libblkid1:amd64	2.25.2-6	amd64	block device id library
libbluetooth3:amd64	5.23-2+b1	amd64	Library to use the BlueZ Linux Bluetooth stack
libbsd0:amd64	0.7.0-2	amd64	utility functions from BSD systems - shared library
libbz2-1.0:amd64	1.0.6-7+b3	amd64	high-quality block-sorting file compressor library - runtime
libc-bin	2.19-18+deb8u4	amd64	GNU C Library: Binaries
libc6:amd64	2.19-18+deb8u4	amd64	GNU C Library: Shared libraries
libcairo2:amd64	1.14.0-2.1+deb8u1	amd64	Cairo 2D vector graphics library

Table 8: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
libcap-ng0:amd64	0.7.4-2	amd64	An alternate POSIX capabilities library
libcap2:amd64	1:2.24-8	amd64	POSIX 1003.1e capabilities (library)
libcap2-bin	1:2.24-8	amd64	POSIX 1003.1e capabilities (utilities)
libcdparanoia0:amd64	3.10.2+debian-11	amd64	audio extraction tool for sampling CDs (library)
libcgi-fast-perl	1:2.04-1	all	CGI subclass for work with FCGI
libcgi-pm-perl	4.09-1	all	module for Common Gateway Interface applications
libclass-c3-perl	0.26-1	all	pragma for using the C3 method resolution order
libclass-c3-xs-perl	0.13-2+b1	amd64	Perl module to accelerate Class::C3
libcomerr2:amd64	1.42.12-1.1	amd64	common error description library
libconfig-grammar-perl	1.10-2	all	grammar-based user-friendly config parser
libcpan-meta-perl	2.142690-1	all	Perl module to access CPAN distributions metadata
libcryptsetup4:amd64	2:1.6.6-5	amd64	disk encryption support - shared library
libcups2:amd64	1.7.5-11+deb8u1	amd64	Common UNIX Printing System(tm) - Core library
libcurl3:amd64	7.38.0-4+deb8u3	amd64	easy-to-use client-side URL transfer library (OpenSSL flavour)
libdata-optlist-perl	0.109-1	all	module to parse and validate simple name/value option pairs
libdata-section-perl	0.200006-1	all	module to read chunks of data from a module's DATA section
libdatrie1:amd64	0.2.8-1	amd64	Double-array trie library
libdb5.3:amd64	5.3.28-9	amd64	Berkeley v5.3 Database Libraries [runtime]
libdbi1:amd64	0.9.0-4	amd64	DB Independent Abstraction Layer for C – shared library
libdbus-1-3:amd64	1.8.20-0+deb8u1	amd64	simple interprocess messaging system (library)
libdebconfclient0:amd64	0.192	amd64	Debian Configuration Management System (C-implementation library)
libdevmapper1.02.1:amd64	2:1.02.90-2.2	amd64	Linux Kernel Device Mapper userspace library
libdigest-hmac-perl	1.03+dfsg-1	all	module for creating standard message integrity checks
libdns100	1:9.9.5.dfsg-9+deb8u6	amd64	DNS Shared Library used by BIND
libdrm-intel1:amd64	2.4.58-2	amd64	Userspace interface to intel-specific kernel DRM services – runtime
libdrm-nouveau2:amd64	2.4.58-2	amd64	Userspace interface to nouveau-specific kernel DRM services – runtime
libdrm-radeon1:amd64	2.4.58-2	amd64	Userspace interface to radeon-specific kernel DRM services – runtime
libdrm2:amd64	2.4.58-2	amd64	Userspace interface to kernel DRM services – runtime
libedit2:amd64	3.1-20140620-2	amd64	BSD editline and history libraries
libelf1:amd64	0.159-4.2	amd64	library to read and write ELF files
libencode-locale-perl	1.03-1	all	utility to determine the locale encoding
libexpat1:amd64	2.1.0-6+deb8u1	amd64	XML parsing C library - runtime library
libfcgi-perl	0.77-1+b1	amd64	helper module for FastCGI
libffi6:amd64	3.1-2+b2	amd64	Foreign Function Interface library runtime
libfile-listing-perl	6.04-1	all	module to parse directory listings
libflac8:amd64	1.3.0-3	amd64	Free Lossless Audio Codec - runtime C library
libfont-afm-perl	1.20-1	all	Font::AFM - Interface to Adobe Font Metrics files
libfontconfig1:amd64	2.11.0-6.3	amd64	generic font configuration library - runtime
libfreetype6:amd64	2.5.2-3+deb8u1	amd64	FreeType 2 font engine, shared library files
libgcc1:amd64	1:4.9.2-10	amd64	GCC support library
libgcrypt20:amd64	1.6.3-2+deb8u1	amd64	LGPL Crypto library - runtime library
libgdbm3:amd64	1.8.3-13.1	amd64	GNU dbm database routines (runtime version)
libgdk-pixbuf2.0-0:amd64	2.31.1-2+deb8u4	amd64	GDK Pixbuf library
libgdk-pixbuf2.0-common	2.31.1-2+deb8u4	all	GDK Pixbuf library - data files
libgeoip1:amd64	1.6.2-4	amd64	non-DNS IP-to-country resolver library
libgfortran3:amd64	4.9.2-10	amd64	Runtime library for GNU Fortran applications
libgl1-mesa-dri:amd64	10.3.2-1+deb8u1	amd64	free implementation of the OpenGL API – DRI modules
libgl1-mesa-glx:amd64	10.3.2-1+deb8u1	amd64	free implementation of the OpenGL API – GLX runtime
libglade2-0:amd64	1:2.6.4-2	amd64	library to load .glade files at runtime
libglapi-mesa:amd64	10.3.2-1+deb8u1	amd64	free implementation of the GL API – shared library
libglib2.0-0:amd64	2.42.1-1+b1	amd64	GLib library of C routines
libglib2.0-data	2.42.1-1	all	Common files for GLib library
libglu1-mesa:amd64	9.0.0-2	amd64	Mesa OpenGL utility library (GLU)
libgmp10:amd64	2:6.0.0+dfsg-6	amd64	Multiprecision arithmetic library
libgnutls-deb0-28:amd64	3.3.8-6+deb8u3	amd64	GNU TLS library - main runtime library
libgpg-error0:amd64	1.17-3	amd64	library for common error values and messages in GnuPG components
libgps21:amd64	3.11-3	amd64	Global Positioning System - library
libgraphite2-3:amd64	1.3.6-1 deb8u1	amd64	Font rendering engine for Complex Scripts – library

Table 8: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
libgssapi-krb5-2:amd64	1.12.1+dfsg-19+deb8u2	amd64	MIT Kerberos runtime libraries - krb5 GSS-API Mechanism
libgstreamer-plugins-base1.0-0:amd64	1.4.4-2	amd64	GStreamer libraries from the "base" set
libgstreamer1.0-0:amd64	1.4.4-2	amd64	Core GStreamer libraries and elements
libgtk2.0-0:amd64	2.24.25-3+deb8u1	amd64	GTK+ graphical user interface library
libgtk2.0-bin	2.24.25-3+deb8u1	amd64	programs for the GTK+ graphical user interface library
libgtk2.0-common	2.24.25-3+deb8u1	all	common files for the GTK+ graphical user interface library
libharfbuzz0b:amd64	0.9.35-2	amd64	OpenType text shaping engine (shared library)
libhogweed2:amd64	2.7.1-5+deb8u1	amd64	low level cryptographic library (public-key cryptos)
libhtml-form-perl	6.03-1	all	module that represents an HTML form element
libhtml-format-perl	2.11-1	all	module for transforming HTML into various formats
libhtml-parser-perl	3.71-1+b3	amd64	collection of modules that parse HTML text documents
libhtml-tagset-perl	3.20-2	all	Data tables pertaining to HTML
libhtml-tree-perl	5.03-1	all	Perl module to represent and create HTML syntax trees
libhttp-cookies-perl	6.01-1	all	HTTP cookie jars
libhttp-daemon-perl	6.01-1	all	simple http server class
libhttp-date-perl	6.02-1	all	module of date conversion routines
libhttp-message-perl	6.06-1	all	perl interface to HTTP style messages
libhttp-negotiate-perl	6.00-2	all	implementation of content negotiation
libice6:amd64	2:1.0.9-1+b1	amd64	X11 Inter-Client Exchange library
libidn11:amd64	1.29-1+b2	amd64	GNU Libidn library, implementation of IETF IDN specifications
libio-html-perl	1.001-1	all	open an HTML file with automatic charset detection
libio-socket-inet6-perl	2.72-1	all	object interface for AF_INET6 domain sockets
libio-socket-ssl-perl	2.002-2+deb8u1	all	Perl module implementing object oriented interface to SSL sockets
libiperf0	3.0.7-1	amd64	Internet Protocol bandwidth measuring tool (runtime files)
libisc95	1:9.9.5.dfsg-9+deb8u6	amd64	ISC Shared Library used by BIND
libisccc90	1:9.9.5.dfsg-9+deb8u6	amd64	Command Channel Library used by BIND
libisccfg90	1:9.9.5.dfsg-9+deb8u6	amd64	Config File Handling Library used by BIND
libjasper1:amd64	1.900.1-debian1-2.4+deb8u1	amd64	JasPer JPEG-2000 runtime library
libjbig0:amd64	2.1-3.1	amd64	JBIGkit libraries
libjpeg62-turbo:amd64	1:1.3.1-12	amd64	libjpeg-turbo JPEG runtime library
libjs-cropper	1.2.2-1	all	JavaScript image cropper UI
libjs-jquery	1.7.2+dfsg-3.2	all	JavaScript library for dynamic web applications
libjs-jquery-ui	1.10.1+dfsg-1	all	JavaScript UI library for dynamic web applications
libjs-prototype	1.7.1-3	all	JavaScript Framework for dynamic web applications
libjs-scriptaculous	1.9.0-2	all	JavaScript library for dynamic web applications
libjson-c2:amd64	0.11-4	amd64	JSON manipulation library - shared library
libk5crypto3:amd64	1.12.1+dfsg-19+deb8u2	amd64	MIT Kerberos runtime libraries - Crypto Library
libkeyutils1:amd64	1.5.9-5+b1	amd64	Linux Key Management Utilities (library)
libkmod2:amd64	18-3	amd64	libkmod shared library
libkrb5-3:amd64	1.12.1+dfsg-19+deb8u2	amd64	MIT Kerberos runtime libraries
libkrb5support0:amd64	1.12.1+dfsg-19+deb8u2	amd64	MIT Kerberos runtime libraries - Support library
liblapack3	3.5.0-4	amd64	Library of linear algebra routines 3 - shared version
liblcms2-2:amd64	2.6-3+b3	amd64	Little CMS 2 color management library
libldap-2.4-2:amd64	2.4.40+dfsg-1+deb8u2	amd64	OpenLDAP libraries
liblinear1:amd64	1.8+dfsg-4	amd64	Library for Large Linear Classification
libllvm3.5:amd64	1:3.5-10	amd64	Modular compiler and toolchain technologies, runtime library
liblocale-gettext-perl	1.05-8+b1	amd64	module using libc functions for internationalization in Perl
liblog-message-perl	0.8-1	all	powerful and flexible message logging mechanism
liblog-message-simple-perl	0.10-2	all	simplified interface to Log::Message
liblua5.1-0:amd64	5.1.5-7.1	amd64	Shared library for the Lua interpreter version 5.1
liblua5.2-0:amd64	5.2.3-1.1	amd64	Shared library for the Lua interpreter version 5.2
liblwp-mediatypes-perl	6.02-1	all	module to guess media type for a file or a URL
liblwp-protocol-https-perl	6.06-2	all	HTTPS driver for LWP::UserAgent
liblwres90	1:9.9.5.dfsg-9+deb8u6	amd64	Lightweight Resolver Library used by BIND
liblzma5:amd64	5.1.1alpha+20120614-2+b3	amd64	XZ-format compression library
libmagic1:amd64	1:5.22+15-2+deb8u1	amd64	File type determination library using "magic" numbers
libmailtools-perl	2.13-1	all	Manipulate email in perl programs
libmng1:amd64	1.0.10+dfsg-3.1+b3	amd64	Multiple-image Network Graphics library
libmodule-build-perl	0.421000-2	all	framework for building and installing Perl modules
libmodule-pluggable-perl	5.1-1	all	module for giving modules the ability to have plugins
libmodule-signature-perl	0.73-1+deb8u2	all	module to manipulate CPAN SIGNATURE files

Table 8: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
libmount1:amd64	2.25.2-6	amd64	device mounting library
libmro-compat-perl	0.12-1	all	mro::* interface compatibility for Perls < 5.9.5
libmysqlclient18:amd64	5.5.47-0+deb8u1	amd64	MySQL database client library
libncurses5:amd64	5.9+20140913-1+b1	amd64	shared libraries for terminal handling
libncursesw5:amd64	5.9+20140913-1+b1	amd64	shared libraries for terminal handling (wide character support)
libnet-http-perl	6.07-1	all	module providing low-level HTTP connection client
libnet-smtp-ssl-perl	1.01-3	all	Perl module providing SSL support to Net::SMTP
libnet-ssleay-perl	1.65-1+b1	amd64	Perl module for Secure Sockets Layer (SSL)
libnettle4:amd64	2.7.1-5+deb8u1	amd64	low level cryptographic library (symmetric and one-way cryptos)
libnfnetlink0:amd64	1.0.1-3	amd64	Netfilter netlink library
libnspr4:amd64	2:4.10.7-1+deb8u1	amd64	NetScape Portable Runtime Library
libnss3:amd64	2:3.17.2-1.1+deb8u2	amd64	Network Security Service libraries
libocomm	2.11.1 rc-mytestbed1	amd64	OComm: O? Communications Library (metapackage)
libocomm-dev	2.11.1 rc-mytestbed1	amd64	OML measurement library headers
libocomm1	2.11.1 rc-mytestbed1	amd64	OComm: O? Communications Library
libogg0:amd64	1.3.2-1	amd64	Ogg bitstream library
liboml2	2.11.1 rc-mytestbed1	amd64	OML: The O? Measurement Library (metapackage)
liboml2-9	2.11.1 rc-mytestbed1	amd64	OML: The O? Measurement Library
liboml2-dev	2.11.1 rc-mytestbed1	amd64	OML measurement library headers
liborc-0.4-0:amd64	1:0.4.22-1	amd64	Library of Optimized Inner Loops Runtime Compiler
libp11-kit0:amd64	0.20.7-1	amd64	Library for loading and coordinating access to PKCS#11 modules - runtime
libpackage-constants-perl	0.04-1	all	List constants defined in a package
libpam-modules:amd64	1.1.8-3.1+deb8u1+b1	amd64	Pluggable Authentication Modules for PAM
libpam-modules-bin	1.1.8-3.1+deb8u1+b1	amd64	Pluggable Authentication Modules for PAM - helper binaries
libpam-runtime	1.1.8-3.1+deb8u1	all	Runtime support for the PAM library
libpam0g:amd64	1.1.8-3.1+deb8u1+b1	amd64	Pluggable Authentication Modules library
libpango-1.0-0:amd64	1.36.8-3	amd64	Layout and rendering of internationalized text
libpangocairo-1.0-0:amd64	1.36.8-3	amd64	Layout and rendering of internationalized text
libpangoft2-1.0-0:amd64	1.36.8-3	amd64	Layout and rendering of internationalized text
libparams-util-perl	1.07-2+b1	amd64	Perl extension for simple stand-alone param checking functions
libpcap0.8:amd64	1.6.2-2	amd64	system interface for user-level packet capture
libpciaccess0:amd64	0.13.2-3+b1	amd64	Generic PCI access library for X
libpcre3:amd64	2:8.35-3.3+deb8u4	amd64	Perl 5 Compatible Regular Expression Library - runtime files
libpcsclite1:amd64	1.8.13-1	amd64	Middleware to access a smart card using PC/SC (library)
libpgm-5.1-0	5.1.118-1 dfsg-1	amd64	OpenPGM shared library
libpixman-1-0:amd64	0.32.6-3	amd64	pixel-manipulation library for X and cairo
libpng12-0:amd64	1.2.50-2+deb8u2	amd64	PNG library - runtime
libpod-latex-perl	0.61-1	all	module to convert Pod data to formatted LaTeX
libpod-readme-perl	0.11-1	all	Perl module to convert POD to README file
libpopt0:amd64	1.16-10	amd64	lib for parsing cmdline parameters
libpq5:amd64	9.4.6-0+deb8u1	amd64	PostgreSQL C client library
libprocps3:amd64	2:3.3.9-9	amd64	library for accessing process information from /proc
libpulse0:amd64	5.0-13	amd64	PulseAudio client libraries
libpython-stdlib:amd64	2.7.9-1	amd64	interactive high-level object-oriented language (default python version)
libpython2.7:amd64	2.7.9-2	amd64	Shared Python runtime library (version 2.7)
libpython2.7-minimal:amd64	2.7.9-2	amd64	Minimal subset of the Python language (version 2.7)
libpython2.7-stdlib:amd64	2.7.9-2	amd64	Interactive high-level object-oriented language (standard library, version 2.7)
libqt4-dbus:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 D-Bus module
libqt4-declarative:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 Declarative module
libqt4-designer:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 designer module
libqt4-help:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 help module
libqt4-network:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 network module
libqt4-opengl:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 OpenGL module

Table 8: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
libqt4-script:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 script module
libqt4-scripttools:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 script tools module
libqt4-sql:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 SQL module
libqt4-sql-mysql:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 MySQL database driver
libqt4-svg:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 SVG module
libqt4-test:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 test module
libqt4-xml:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 XML module
libqt4-xmlpatterns:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 XML patterns module
libqtassistantclient4:amd64	4.6.3-6	amd64	Qt Assistant client library (runtime)
libqtcore4:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 core module
libqtdbus4:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 D-Bus module library
libqtgui4:amd64	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 GUI module
libqtwebkit4:amd64	2.3.4.dfsg-3	amd64	Web content engine library for Qt
libquadmath0:amd64	4.9.2-10	amd64	GCC Quad-Precision Math Library
libreadline6:amd64	6.3-8+b3	amd64	GNU readline and history libraries, run-time libraries
libregexp-common-perl	2013031301-1	all	module with common regular expressions
librrd4	1.4.8-1.2	amd64	time-series data storage and display system (runtime library)
librrds-perl	1.4.8-1.2	amd64	time-series data storage and display system (Perl interface, shared)
librtmp1:amd64	2.4+20150115.gita107cef-1	amd64	toolkit for RTMP streams (shared library)
libruby2.1:amd64	2.1.5-2+deb8u2	amd64	Libraries necessary to run Ruby 2.1
libsasl2-2:amd64	2.1.26.dfsg1-13+deb8u1	amd64	Cyrus SASL - authentication abstraction library
libsasl2-modules:amd64	2.1.26.dfsg1-13+deb8u1	amd64	Cyrus SASL - pluggable authentication modules
libsasl2-modules-db:amd64	2.1.26.dfsg1-13+deb8u1	amd64	Cyrus SASL - pluggable authentication modules (DB)
libsctp1:amd64	1.0.16+dfsg-2	amd64	user-space access to Linux Kernel SCTP - shared library
libselinux1:amd64	2.3-2	amd64	SELinux runtime shared libraries
libsemanage-common	2.3-1	all	Common files for SELinux policy management libraries
libsemanage1:amd64	2.3-1+b1	amd64	SELinux policy management library
libsepol1:amd64	2.3-2	amd64	SELinux library for manipulating binary security policies
libsigar	1.6.5-1ppa1o	amd64	System Information Gatherer And Reporter
libslang2:amd64	2.3.0-2	amd64	S-Lang programming library - runtime version
libsm6:amd64	2:1.2.2-1+b1	amd64	X11 Session Management library
libsmartcols1:amd64	2.25.2-6	amd64	smart column output alignment library
libsndfile1:amd64	1.0.25-9.1+deb8u1	amd64	Library for reading/writing audio files
libsnmp-session-perl	1.13-1.1	all	Perl support for accessing SNMP-aware devices
libsocket6-perl	0.25-1+b1	amd64	Perl extensions for IPv6
libsodium13:amd64	1.0.0-1	amd64	Network communication, cryptography and signaturing library
libsoftware-license-perl	0.103010-3	all	module providing templated software licenses
libsqlite3-0:amd64	3.8.7.1-1+deb8u1	amd64	SQLite 3 shared library
libss2:amd64	1.42.12-1.1	amd64	command-line interface parsing library
libssh2-1:amd64	1.4.3-4.1+deb8u1	amd64	SSH2 client-side library
libssl1.0.0:amd64	1.0.1k-3+deb8u4	amd64	Secure Sockets Layer toolkit - shared libraries
libstdc++6:amd64	4.9.2-10	amd64	GNU Standard C++ Library v3
libsub-exporter-perl	0.986-1	all	sophisticated exporter for custom-built routines
libsub-install-perl	0.928-1	all	module for installing subroutines into packages easily
libsystemd0:amd64	215-17+deb8u4	amd64	systemd utility library
libtasn1-6:amd64	4.2-3+deb8u1	amd64	Manage ASN.1 structures (runtime)
libtcl8.6:amd64	8.6.2+dfsg-2	amd64	Tcl (the Tool Command Language) v8.6 - run-time library files
libterm-ui-perl	0.42-1	all	Term::ReadLine UI made easy
libtext-charwidth-perl	0.04-7+b3	amd64	get display widths of characters on the terminal
libtext-iconv-perl	1.7-5+b2	amd64	converts between character sets in Perl
libtext-soundex-perl	3.4-1+b2	amd64	implementation of the soundex algorithm

Table 8: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
libtext-template-perl	1.46-1	all	perl module to process text templates
libtext-wrapi18n-perl	0.06-7	all	internationalized substitute of Text::Wrap
libthai-data	0.1.21-1	all	Data files for Thai language support library
libthai0:amd64	0.1.21-1	amd64	Thai language support library
libtheora0:amd64	1.1.1+dfsg.1-6	amd64	Theora Video Compression Codec
libtiff5:amd64	4.0.3-12.3+deb8u1	amd64	Tag Image File Format (TIFF) library
libtimedate-perl	2.3000-2	all	collection of modules to manipulate date/time information
libtinfo5:amd64	5.9+20140913-1+b1	amd64	shared low-level terminfo library for terminal handling
libtk8.6:amd64	8.6.2-1	amd64	Tk toolkit for Tcl and X11 v8.6 - run-time files
libtrace3	3.0.21-1	amd64	network trace processing library supporting many input formats
libtxc-dxtn-s2tc0:amd64	0 git20131104-1.1	amd64	Texture compression library for Mesa
libudev1:amd64	215-17+deb8u4	amd64	libudev shared library
liburi-perl	1.64-1	all	module to manipulate and access URI strings
libusb-0.1-4:amd64	2:0.1.12-25	amd64	userspace USB programming library
libusb-1.0-0:amd64	2:1.0.19-1	amd64	userspace USB programming library
libustr-1.0-1:amd64	1.0.4-3+b2	amd64	Micro string library: shared library
libuuid1:amd64	2.25.2-6	amd64	Universally Unique ID library
libvisual-0.4-0:amd64	0.4.0-6	amd64	Audio visualization framework
libvisual-0.4-plugins:amd64	0.4.0.dfsg.1-7	amd64	Audio visualization framework plugins
libvorbis0a:amd64	1.3.4-2	amd64	decoder library for Vorbis General Audio Compression Codec
libvorbisenc2:amd64	1.3.4-2	amd64	encoder library for Vorbis General Audio Compression Codec
libwandio1	3.0.21-1	amd64	multi-threaded file compression and decompression library
libwebp5:amd64	0.4.1-1.2+b2	amd64	Lossy compression of digital photographic images.
libwebpdemux1:amd64	0.4.1-1.2+b2	amd64	Lossy compression of digital photographic images.
libwebpmux1:amd64	0.4.1-1.2+b2	amd64	Lossy compression of digital photographic images.
libwrap0:amd64	7.6.q-25	amd64	Wietse Venema's TCP wrappers library
libwww-perl	6.08-1	all	simple and consistent interface to the world-wide web
libwww-robotrules-perl	6.01-1	all	database of robots.txt-derived permissions
libx11-6:amd64	2:1.6.2-3	amd64	X11 client-side library
libx11-data	2:1.6.2-3	all	X11 client-side library
libx11-xcb1:amd64	2:1.6.2-3	amd64	Xlib/XCB interface library
libxau6:amd64	1:1.0.8-1	amd64	X11 authorisation library
libxcb-dri2-0:amd64	1.10-3+b1	amd64	X C Binding, dri2 extension
libxcb-dri3-0:amd64	1.10-3+b1	amd64	X C Binding, dri3 extension
libxcb-glx0:amd64	1.10-3+b1	amd64	X C Binding, glx extension
libxcb-present0:amd64	1.10-3+b1	amd64	X C Binding, present extension
libxcb-render0:amd64	1.10-3+b1	amd64	X C Binding, render extension
libxcb-shm0:amd64	1.10-3+b1	amd64	X C Binding, shm extension
libxcb-sync1:amd64	1.10-3+b1	amd64	X C Binding, sync extension
libxcb1:amd64	1.10-3+b1	amd64	X C Binding
libxcomposite1:amd64	1:0.4.4-1	amd64	X11 Composite extension library
libxcursor1:amd64	1:1.1.14-1+b1	amd64	X cursor management library
libxdamage1:amd64	1:1.1.4-2+b1	amd64	X11 damaged region extension library
libxdmcp6:amd64	1:1.1.1-1+b1	amd64	X11 Display Manager Control Protocol library
libxext6:amd64	2:1.3.3-1	amd64	X11 miscellaneous extension library
libxfixes3:amd64	1:5.0.1-2+b2	amd64	X11 miscellaneous 'fixes' extension library
libxft2:amd64	2.3.2-1	amd64	FreeType-based font drawing library for X
libxi6:amd64	2:1.7.4-1+b2	amd64	X11 Input extension library
libxinerama1:amd64	2:1.1.3-1+b1	amd64	X11 Xinerama extension library
libxml2:amd64	2.9.1+dfsg1-5+deb8u1	amd64	GNOME XML library
libxmuu1:amd64	2:1.1.2-1	amd64	X11 miscellaneous micro-utility library
libxrandr2:amd64	2:1.4.2-1+b1	amd64	X11 RandR extension library
libxrender1:amd64	1:0.9.8-1+b1	amd64	X Rendering Extension client library
libxshmfence1:amd64	1.1-4	amd64	X shared memory fences - shared library
libxslt1.1:amd64	1.1.28-2+b2	amd64	XSLT 1.0 processing library - runtime library
libxss1:amd64	1:1.2.2-1	amd64	X11 Screen Saver extension library
libxt6:amd64	1:1.1.4-1+b1	amd64	X11 toolkit intrinsics library
libxtables10	1.4.21-2+b1	amd64	netfilter xtables library
libxtst6:amd64	2:1.2.2-1+b1	amd64	X11 Testing – Record extension library
libxxf86vm1:amd64	1:1.1.3-1+b1	amd64	X11 XFree86 video mode extension library
libyaml-0-2:amd64	0.1.6-3	amd64	Fast YAML 1.1 parser and emitter library
libzmq3:amd64	4.0.5+dfsg-2+deb8u1	amd64	lightweight messaging kernel (shared library)

Table 8: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
lksctp-tools	1.0.16+dfsg-2	amd64	user-space access to Linux Kernel SCTP - commandline tools
login	1:4.2-3+deb8u1	amd64	system login tools
lsb-base	4.1+Debian13+nmu1	all	Linux Standard Base 4.1 init script functionality
mawk	1.3.3-17	amd64	a pattern scanning and text processing language
mgen	5.02+dfsg2-3	amd64	packet generator for IP network performance tests
mime-support	3.58	all	MIME files 'mime.types' & 'mailcap', and support programs
mount	2.25.2-6	amd64	Tools for mounting and manipulating filesystems
multiarch-support	2.19-18+deb8u4	amd64	Transitional package to ensure multiarch compatibility
mysql-common	5.5.47-0+deb8u1	all	MySQL database common files, e.g. /etc/mysql/my.cnf
ncurses-base	5.9+20140913-1	all	basic terminal type definitions
ncurses-bin	5.9+20140913-1+b1	amd64	terminal-related programs and man pages
ndiff	6.47-3	all	The Network Mapper - result compare utility
net-tools	1.60-26+b1	amd64	NET-3 networking toolkit
netbase	5.3	all	Basic TCP/IP networking system
netperf	2.7.0-1	amd64	Network performance benchmark
nmap	6.47-3+b1	amd64	The Network Mapper
nmetrics-oml2	2.11.0-mytestbed2	amd64	Measure and record system information from libsigar using OML
oml2	2.11.1 rc-mytestbed1	amd64	OML: The O? Measurement Library Suite (Metapackage)
oml2-apps	2.11.0-mytestbed2	amd64	Standalone OML2 applications (metapackage)
oml2-proxy-server	2.11.1 rc-mytestbed1	amd64	OML proxy server
oml2-proxycon	2.11.1 rc-mytestbed1	amd64	OML proxy server control script
oml2-server	2.11.1 rc-mytestbed1	amd64	OML measurement server
openjdk-7-jre-headless:amd64	7u95-2.6.4-1 deb8u1	amd64	OpenJDK Java runtime, using Hotspot JIT (headless)
openssh-client	1:6.7p1-5+deb8u2	amd64	secure shell (SSH) client, for secure access to remote machines
openssl	1.0.1k-3+deb8u4	amd64	Secure Sockets Layer toolkit - cryptographic utility
otg2-oml2	2.11.0-mytestbed2	amd64	Orbit Traffic Generator
paris-traceroute	0.92-dev-2	amd64	New version of well known tool traceroute
passwd	1:4.2-3+deb8u1	amd64	change and administer password and group data
perl	5.20.2-3+deb8u4	amd64	Larry Wall's Practical Extraction and Report Language
perl-base	5.20.2-3+deb8u4	amd64	minimal Perl system
perl-modules	5.20.2-3+deb8u4	all	Core Perl modules
procps	2:3.3.9-9	amd64	/proc file system utilities
python	2.7.9-1	amd64	interactive high-level object-oriented language (default version)
python-cairo	1.8.8-1+b2	amd64	Python bindings for the Cairo vector graphics library
python-dateutil	2.2-2	all	powerful extensions to the standard datetime module
python-glade2	2.24.0-4	amd64	GTK+ bindings: Glade support
python-gobject-2	2.28.6-12+b1	amd64	deprecated static Python bindings for the GObject library
python-gtk2	2.24.0-4	amd64	Python bindings for the GTK+ widget set
python-imaging	2.6.1-2+deb8u2	all	Python Imaging Library compatibility layer
python-lxml	3.4.0-1	amd64	pythonic binding for the libxml2 and libxslt libraries
python-matplotlib	1.4.2-3.1	amd64	Python based plotting system in a style similar to Matlab
python-matplotlib-data	1.4.2-3.1	all	Python based plotting system (data package)
python-meld3	1.0.0-1	amd64	HTML/XML templating system for Python
python-minimal	2.7.9-1	amd64	minimal subset of the Python language (default version)
python-mock	1.0.1-3	all	Mocking and Testing Library
python-nose	1.3.4-1	all	test discovery and running of Python's unittest
python-numpy	1:1.8.2-2	amd64	Numerical Python adds a fast array facility to the Python language
python-pil:amd64	2.6.1-2+deb8u2	amd64	Python Imaging Library (Pillow fork)
python-pkg-resources	5.5.1-1	all	Package Discovery and Resource Access using pkg_resources
python-pyparsing	2.0.3+dfsg1-1	all	Python parsing module
python-qt4	4.11.2+dfsg-1	amd64	Python bindings for Qt4
python-sip	4.16.4+dfsg-1	amd64	Python/C++ bindings generator runtime library
python-six	1.8.0-1	all	Python 2 and 3 compatibility library (Python 2 interface)
python-support	1.0.15	all	automated rebuilding support for Python modules
python-tk	2.7.8-2+b1	amd64	Tkinter - Writing Tk applications with Python
python-tz	2012c+dfsg-0.1	all	Python version of the Olson timezone database
python-zmq	14.4.0-1	amd64	Python bindings for 0MQ library
python2.7	2.7.9-2	amd64	Interactive high-level object-oriented language (version 2.7)
python2.7-minimal	2.7.9-2	amd64	Minimal subset of the Python language (version 2.7)

Table 8: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
qdbus	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	amd64	Qt 4 D-Bus tool
qtchooser	47-gd2b7997-2	amd64	Wrapper to select between Qt development binary versions
qtcore4-l10n	4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1	all	Qt 4 core module translations
readline-common	6.3-8	all	GNU readline and history libraries, common files
rename	0.20-3	all	Perl extension for renaming multiple files
ripwavemon-oml2	2.11.0-mytestbed2	amd64	Report statistics from a Navini RipWave modem
rsync	3.1.1-3	amd64	fast, versatile, remote (and local) file-copying tool
ruby	1:2.1.5+deb8u2	all	Interpreter of object-oriented scripting language Ruby (default version)
ruby2.1	2.1.5-2+deb8u2	amd64	Interpreter of object-oriented scripting language Ruby
rubygems-integration	1.8	all	integration of Debian Ruby packages with Rubygems
sed	4.2.2-4+b1	amd64	The GNU sed stream editor
sensible-utils	0.0.9	all	Utilities for sensible alternative selection
sgml-base	1.26+nmu4	all	SGML infrastructure and SGML catalog file support
shared-mime-info	1.3-1	amd64	FreeDesktop.org shared MIME database and spec
smokeping	2.6.9-1+deb8u1	all	latency logging and graphing system
ssl-cert	1.0.35	all	simple debconf wrapper for OpenSSL
startpar	0.59-3	amd64	run processes in parallel and multiplex their output
supervisor	3.0r1-1	all	A system for controlling process state
systemd	215-17+deb8u4	amd64	system and service manager
systemd-sysv	215-17+deb8u4	amd64	system and service manager - SysV links
sysv-rc	2.88dsf-59	all	System-V-like runlevel change mechanism
sysvinit-utils	2.88dsf-59	amd64	System-V-like utilities
tar	1.27.1-2+b1	amd64	GNU version of the tar archiving utility
tcpd	7.6.q-25	amd64	Wietse Venema's TCP wrapper utilities
tcpdump	4.6.2-5+deb8u1	amd64	command-line network traffic analyzer
tk8.6-blt2.5	2.5.3+dfsg-1	amd64	graphics extension library for Tcl/Tk - library
trace-oml2	2.11.0-mytestbed2	amd64	Measure and record libtrace data using OML
traceroute	1:2.0.20-2+b1	amd64	Traces the route taken by packets over an IPv4/IPv6 network
tzdata	2016d-0+deb8u1	all	time zone and daylight-saving time data
tzdata-java	2016d-0+deb8u1	all	time zone and daylight-saving time data for use by java run-times
ucf	3.0030	all	Update Configuration File(s): preserve user changes to config files
udev	215-17+deb8u4	amd64	/dev/ and hotplug management daemon
util-linux	2.25.2-6	amd64	Miscellaneous system utilities
x11-common	1:7.7+7	all	X Window System (X.Org) infrastructure
xauth	1:1.0.9-1	amd64	X authentication utility
xdg-user-dirs	0.15-2	amd64	tool to manage well known user directories
xml-core	0.13+nmu2	all	XML infrastructure and XML catalog file support
zlib1g:amd64	1:1.2.8.dfsg-2+b1	amd64	compression library - runtime

B Description of metadata fields

Table 9: Field description for metadata topic "MONROE.META.DEVICE.MODEM".

Name	Description
NodeId	Node numerical ID.
Timestamp	Entry timestamp (in seconds since UNIX epoch).
DataId	Metadata topic.
DataVersion	Set to 1.
SequenceNumber	Monotonically increasing message counter.
InterfaceName	Name of the interface in the MONROE node, e.g., "usb0", "usb1", "usb2", "eth0", ...
InternalInterface	Name of the interface inside the containers, e.g., "op0", "op1", "op2", "eth0", "wlan0", ... Experiments in containers have to bind to these interface names.
Cid	Cell ID.
DeviceMode	Connection mode of the modem (e.g., 2G, 3G, LTE) indicating the radio access technology the modem uses.

Table 9: Field description for metadata topic “MONROE.META.DEVICE.MODEM”. (Continued)

Name	**Description**
DeviceSubmode	Connection submode for 3G connections (e.g., CDMA, WCDMA, UMTS).
DeviceState	State of the device reported to the network: UNKNOWN (0) - Device state is unknwon; REGISTERED (1) - Device is registered to the network; UNREGISTERED (2) - Device is unregistered from the network; CONNECTED (3) - Device is connected to the network; DISCONNECTED (4) - Device is disconnected from the network.
Ecio	EC/IO, quality/cleanliness of signal from the tower to the modem (dB).
ENodebId	Evolved base station ID.
Iccid	Internationally defined integrated circuit card identifier of the SIM card.
Imsi	Internation Mobile Subscriber Identity.
ImsiMccMnc	Mobile Country Code (MCC) and Mobile Network Code (MNC).
Imei	International Mobile Station Equipment Identity.
IpAddress	IP address assigned to the modem by the operator.
InternalIpAddress	Internal IP address of the modem in the MONROE node.
MccMnc	Mobile Country Code (MCC) and Mobile Network Code (MNC).
Operator	Operator name as reported by the network for the interface in which the experiment was run.
Lac	Local Area Code for the current cell (hex).
Rsrp	Reference Signal Received Power (LTE).
Frequency	Frequency in MHz (e.g., 700, 800, 900, 1800 or 2600 in Europe).
Rsrq	Reference Signal Received Quality (valid only for LTE networks). The RSRQ measurement provides additional information when Reference Signal Received Power (RSRP) is not sufficient to make a reliable handover or cell reselection decision. RSRQ considers both the Received Signal Strength Indicator (RSSI) and the number of used Resource Blocks (N) $RSRQ = (N * RSRP)/RSSI$ measured over the same bandwidth.
Band	Band corresponding to the frequency used (e.g., 3, 7 or 20 in Europe).
Pci	Physical Cell ID.
NwMccMnc	Mobile Country Code (MCC) and Mobile Network Code (MNC) from network (read from the network). The tuple uniquely identifies a mobile network operator (carrier) that is using the GSM (including GSM-R), UMTS, and LTE public land mobile networks.
Rscp	Received Signal Code Power (UMTS).
Rssi	Received Signal Strength Indicator.

Table 10: Field description for metadata topic “MONROE.META.DEVICE.GPS”.

Name	**Description**
NodeId	Node numerical ID.
Timestamp	Entry timestamp (in seconds since UNIX epoch).
DataId	Metadata topic.
DataVersion	Set to 1.
SequenceNumber	Monotonically increasing message counter.
Longitude	Decimal degrees (WGS84).
Latitude	Decimal degrees (WGS84).
Altitude	Meters AMSL.
Speed	Speed over ground (knots; multiply by 1.852 to get $\mathrm{km\,h^{-1}}$).
SatelliteCount	Number of satellites being tracked.
Nmea	Raw NMEA string from the GPS receiver.

Table 11: Field description for metadata topic “MONROE.META.CONNECTIVITY”.

Name	**Description**
NodeId	Node numerical ID.
Timestamp	Entry timestamp (in seconds since UNIX epoch).
DataId	Metadata topic.
DataVersion	Set to 1.
SequenceNumber	Monotonically increasing message counter.
Iccid	Internationally defined integrated circuit card identifier of the SIM card.
InterfaceName	Name of the interface in the MONROE node (outside containers), e.g., “usb0”, “usb1”, “usb2”, “eth0”, ...
MccMnc	Mobile Country Code (MCC) and Mobile Network Code (MNC).
Mode	The connection mode: UNKNOWN (1), DISCONNECTED (2), NO SERVICE (3), 2G (4), 3G (5), LTE (6).
Rssi	Signal strength.

Table 12: Field description for metadata topic “MONROE.META.NODE.SENSOR”.

Name	Description
NodeId	Node numerical ID.
Timestamp	Entry timestamp (in seconds since UNIX epoch).
DataId	Metadata topic.
DataVersion	Set to 1.
SequenceNumber	Monotonically increasing message counter.
Running	Comma separated list of experiment GUIDs.
Cpu	CPU temperature (°C).
Id	Session number (boot counter).
Start	Start time (Unix timestamp).
Current	Uptime (seconds since start of the session).
Total	Uptime (cumulative uptime of the node over all sessions).
Percent	Uptime (percent of uptime vs. total lifetime of the node).
System	CPU time spent by the kernel in system activities.
Steal	The time that a virtual CPU had runnable tasks, but the virtual CPU itself was not running.
Guest	The time spent running a virtual CPU for guest operating systems under the control of the Linux kernel.
IoWait	CPU time spent waiting for I/O operations to finish when there is nothing else to do.
Irq	CPU time spent handling interrupts.
Nice	CPU time spent by nice(1)d programs.
Idle	Idle CPU time.
User	CPU time spent by normal programs and daemons.
SoftIrq	CPU time spent handling “batched” interrupts.
Apps	Memory used by user-space applications.
Free	Unused memory.
Swap	Swap space used.
usb0	Battery level for MiFi at USB0 (0-100, -1 for inactive).
usb0charging	1 if USB0 battery is charging, 0 otherwise.
usb1	Battery level for MiFi at USB1 (0-100, -1 for inactive).
usb1charging	1 if USB1 battery is charging, 0 otherwise.
usb2	Battery level for MiFi at USB2 (0-100, -1 for inactive).
usb2charging	1 if USB2 battery is charging, 0 otherwise.

Table 13: Field description for metadata topic “MONROE.META.NODE.EVENT”.

Name	Description
NodeId	Node numerical ID.
Timestamp	Entry timestamp (in seconds since UNIX epoch).
DataId	Metadata topic.
DataVersion	Set to 1.
SequenceNumber	Monotonically increasing message counter.
EventType	Watchdog.Failed: The system watchdog detected an error symptom.
	Watchdog.Repaired: The system watchdog resolved the issue.
	Watchdog.Status: Periodic status messages from the watchdog.
	Maintenance.Start: An interactive login on the node is registered.
	Maintenance.Stop: The interactive login session is closed.
	System.Halt: System halt is requested.
	Scheduling.Started: The node starts to query the scheduling server.
Message	Extra key for some event types.
User	Extra key for some event types.

Table 14: Field description for metadata topic “MONROE.EXP.PING”.

Name	Description
NodeId	Node numerical ID.
Guid	Unique experiment identifier.
Timestamp	Entry timestamp (in seconds since UNIX epoch).
SequenceNumber	Monotonically increasing message counter.
DataId	Metadata topic.
DataVersion	Set to 1.
Operator	Operator name as reported by the network for the interface in which the experiment was run.

Table 14: Field description for metadata topic "MONROE.EXP.PING". (Continued)

Name	Description
Iccid	Internationally defined integrated circuit card identifier of the SIM card.
Bytes	Size of the ping message payload.
Host	IP of the destination host of the ping probe.
Rtt	Round-Trip-Time of the ping probe.

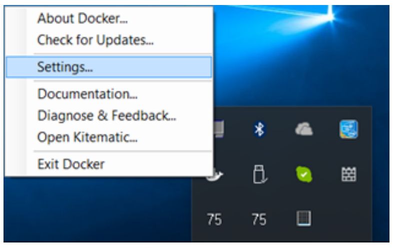
Table 15: Field description for metadata topic "MONROE.EXP.HTTP.DOWNLOAD".

Name	Description
NodeId	Node numerical ID.
Guid	Unique experiment identifier.
Timestamp	Entry timestamp (in fractional seconds since UNIX epoch).
SequenceNumber	Monotonically increasing message counter.
DataId	Metadata topic.
DataVersion	Set to 1.
Operator	Operator name as reported by the network for the interface in which the experiment was run.
Iccid	Internationally defined integrated circuit card identifier of the SIM card.
TotalTime	Total experiment execution time (in fractional seconds).
Bytes	Total number of bytes downloaded.
SetupTime	Time required to set up the HTTP connection.
DownloadTime	Time spent doing the actual download.
Host	IP address of the remote host from which data was downloaded.
Speed	Download speed in bytes/s as measured by the experiment.
Port	TCP port of the remote host from which data was downloaded.

C How to map container folders to Windows paths

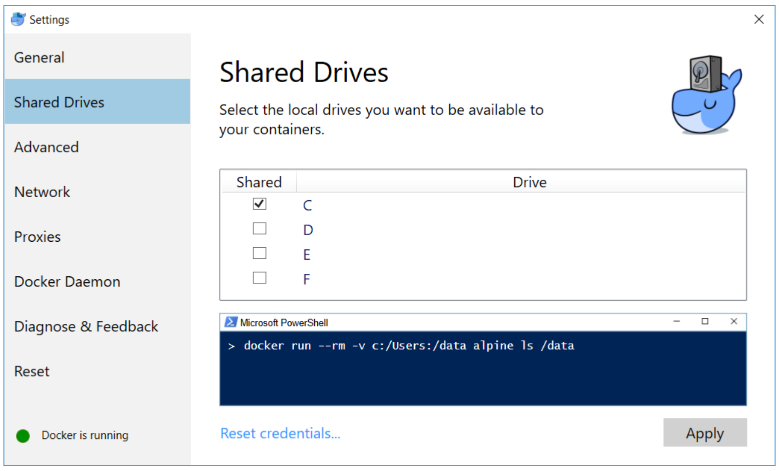
Before being able to access Windows (host) folders from a container, the drive has to be made available to the containers following these steps:[4,5]

1. Access the Docker settings dialog from its taskbar icon:



2. From the tab "Shared Drives", select the drive you want to make available to the containers, e.g., "C":

[4]https://rominirani.com/docker-on-windows-mounting-host-directories-d96f3f056a2c#.pdeuy0c4o
[5]Thanks to Lena for pointing to the solution.



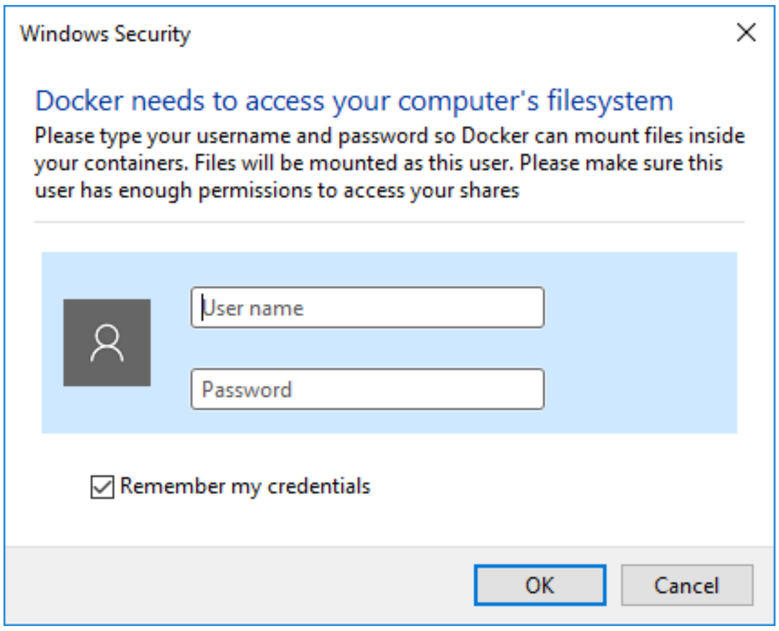
Shared Drives

Select the local drives you want to be available to your containers.

Shared	Drive
✓	C
☐	D
☐	E
☐	F

```
> docker run --rm -v c:/Users:/data alpine ls /data
```

3. You will be prompted for login credentials to access the files:



Docker needs to access your computer's filesystem

Please type your username and password so Docker can mount files inside your containers. Files will be mounted as this user. Please make sure this user has enough permissions to access your shares

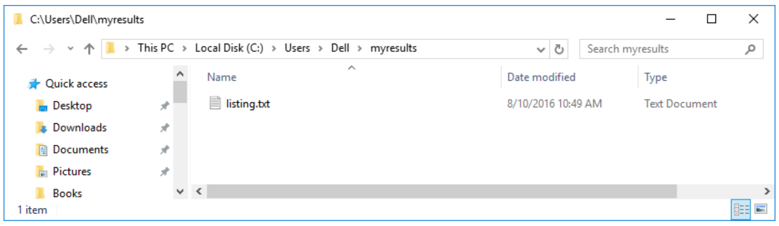
☑ Remember my credentials

4. Start the container mounting the desired folder:

```
$ docker run -v c:/Users/Dell/myresults:/data container_name ls /data
```

This command executes `ls /data` inside an instance of the container "container_name," after mounting "`C:/Users/Dell/myresults/`" into that path.

5. The folder can be accessed normally from Windows and will reflect changes to any files automatically.



Name	Date modified	Type
listing.txt	8/10/2016 10:49 AM	Text Document

Disclaimer