MONROE
Measuring Mobile Broadband Networks in Europe

H2020-ICT-11-2014
Project number: 644399

Deliverable User manual
MONROE Platform User Manual

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

Work Package: 5.2 / User Support
Revision: 1.0
Date: October 23, 2016
Deliverable type: R (Report)
Dissemination level: Public

Abstract

This document describes the processes that MONROE experimenters need to follow to create, run, monitor and collect results from their experiments.

Participant organisation name	Short name
SIMULA RESEARCH LABORATORY AS *(Coordinator)*	SRL
CELERWAY COMMUNICATION AS	Celerway
TELENOR ASA	Telenor
NETTET SVERIGE AB	NET1
NEXTWORKS	NXW
FUNDACION IMDEA NETWORKS	IMDEA
KARLSTADS UNIVERSITET	KaU
POLITECNICO DI TORINO	POLITO

Contents

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 for the 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 of 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 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 intermmediate hops. This container will in principle be run three times per day, but the interactions with user experiments are yet to be determined.

– A container that runs Tstat, the passive monitoring tool 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.

1.2   Overview of the experimental workflow

Experiments conducted in the MONROE platform follow the workflow shown in Figure 1. They consist 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 inside a Docker container under the root user of the container. Therefore, the 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. If not, we strongly 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.

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.* ⇐
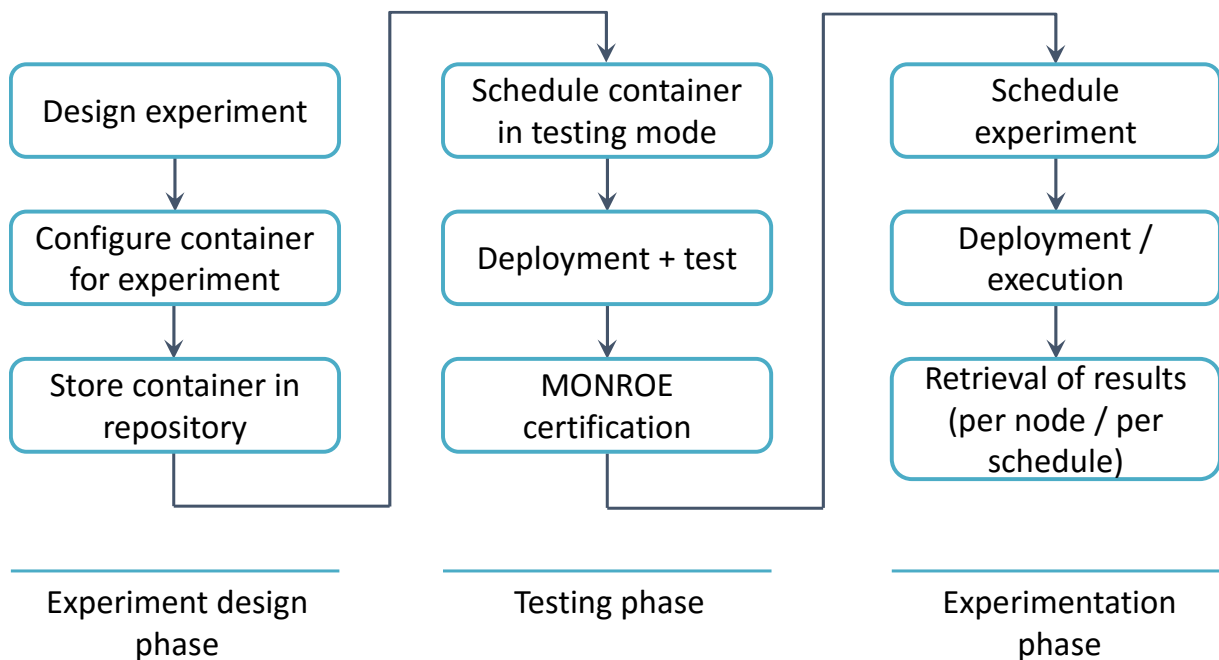
Figure 1: Experimental workflow.

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 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:xxxxxxxxxxxxxxxxxxxxxxxxx
   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` will be retrieved after its completion and delivered to the repository, where you will be able to finally 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 prototype phase of the system, please follow these additional steps to make your container available:

- 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 2 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 an inverse 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 will possibly 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 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.

---

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

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

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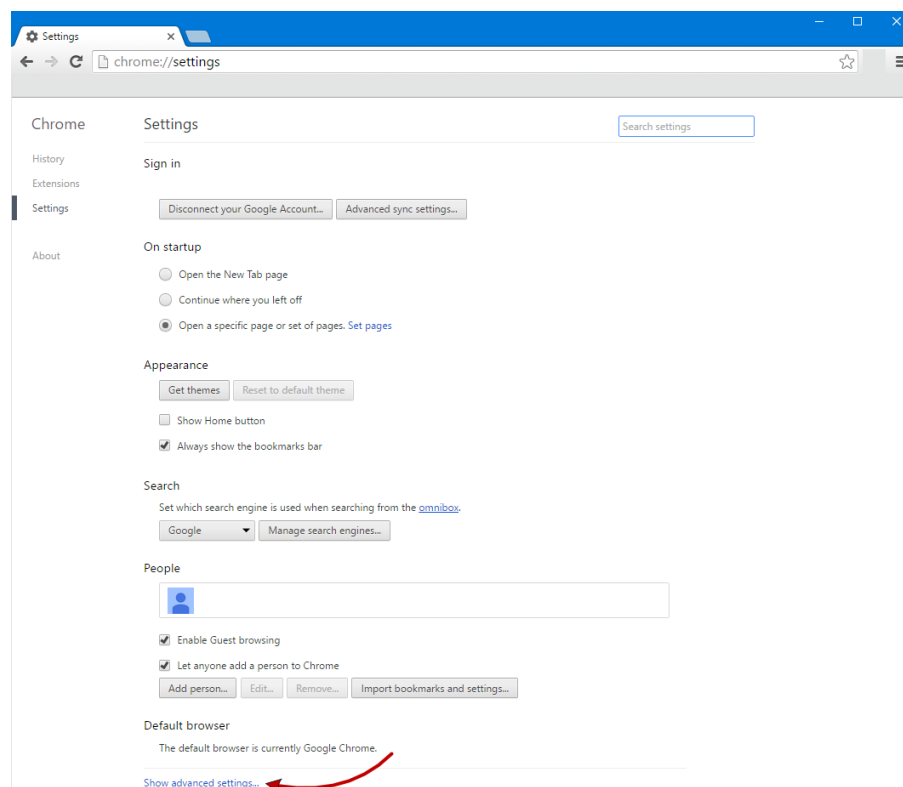
This section explains how to install the FED4FIRE-compatible user certificates used by the MONROE platform in Google Chrome for Windows. The procedure for other browsers and platforms should be similar.

1. Go to your browser settings page:
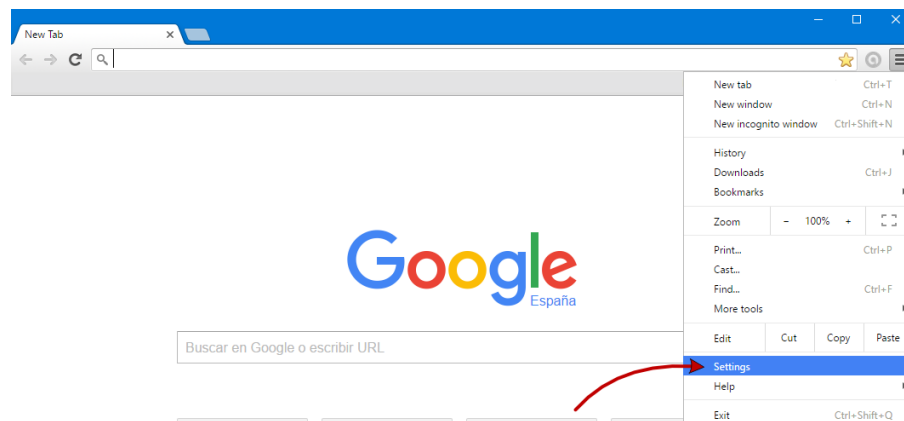
2. Display the advanced configuration settings:

3. Go to the section labeled "HTTPS/SSL" and click the button "Manage certificates...":
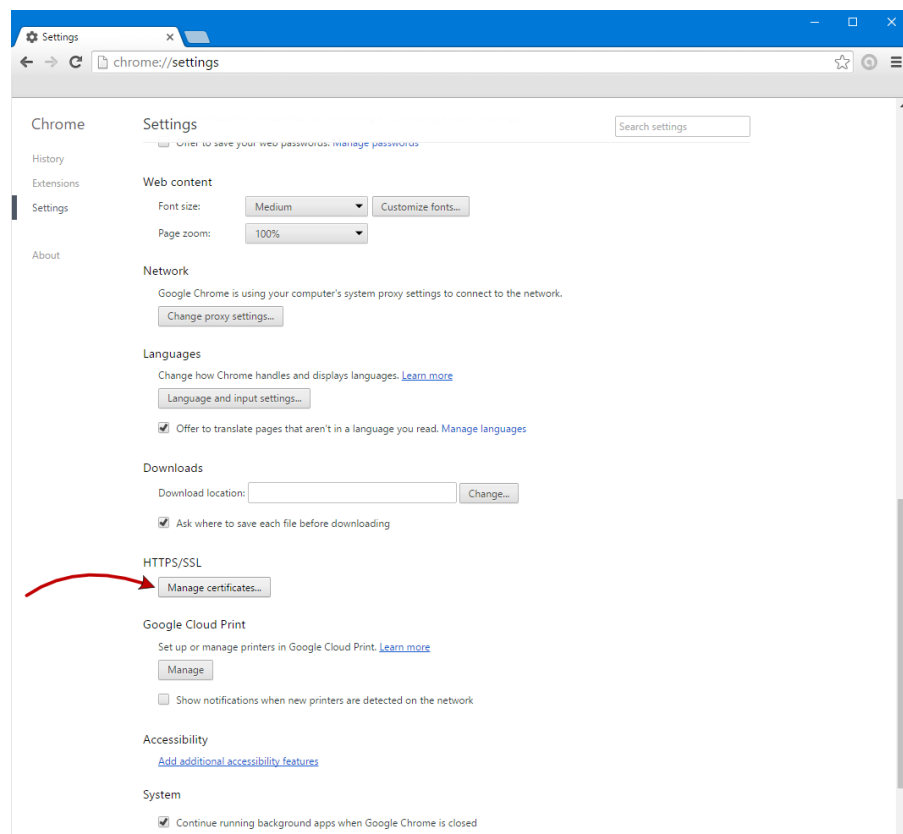
4. The dialog box for managing your certificates will be displayed. Press the button "Import..." to import your certificate:
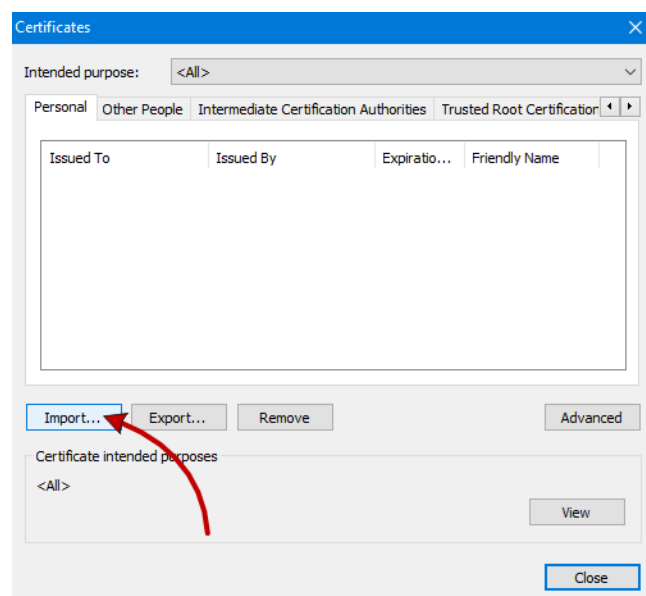
5. In the new dialog, click "Next":
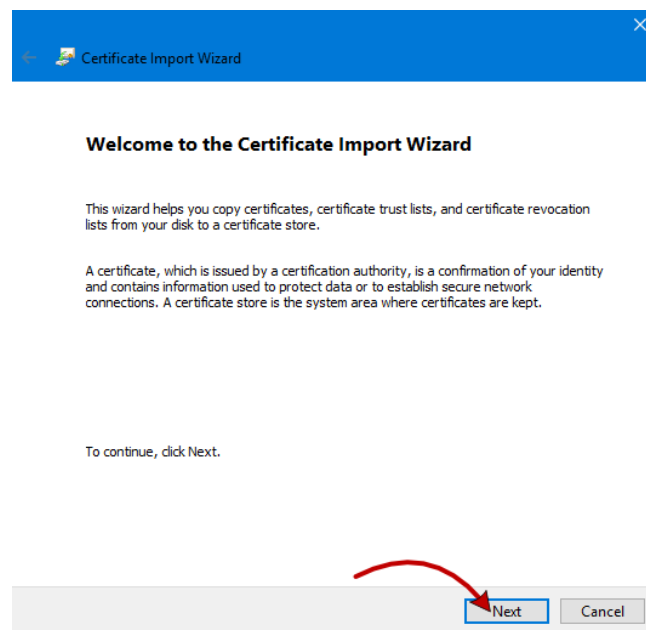
6. In the file-selection dialog that appears next, change the file type from "X.509 Certificate (*,cer;*.crt)" to "Personal Information Exchange":
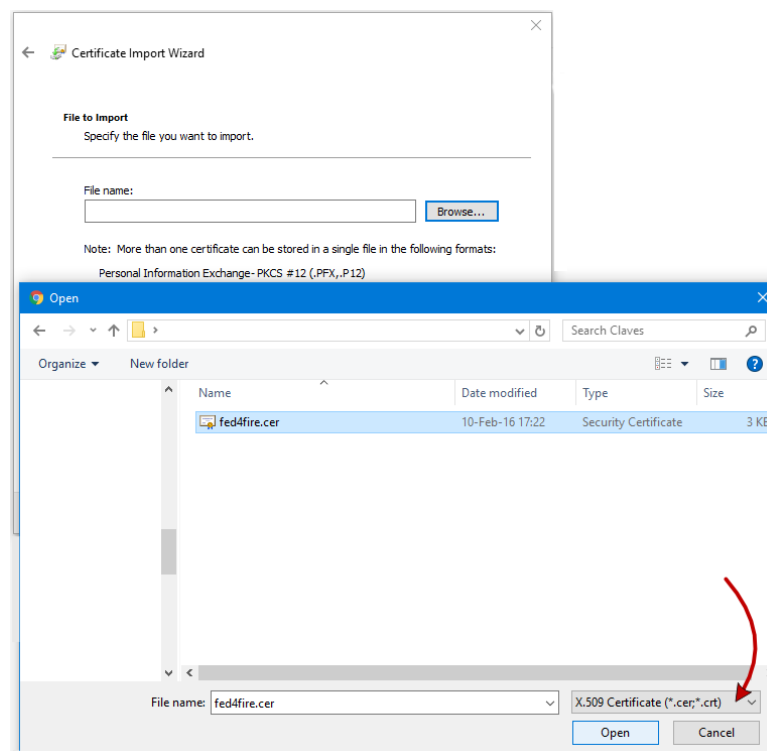
7. And select the file containing your certificate:
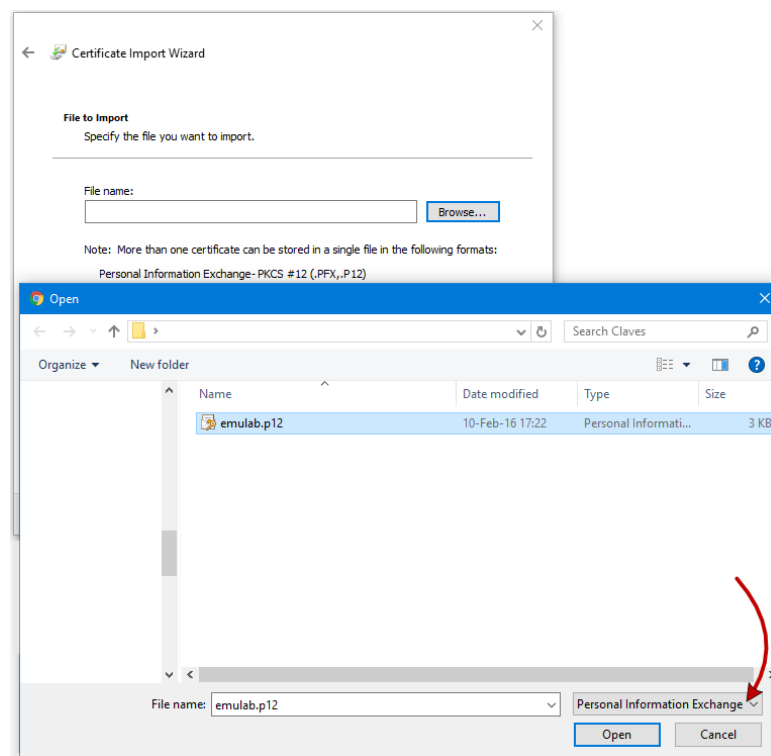
8. In the next dialog box, enter your certificate password:
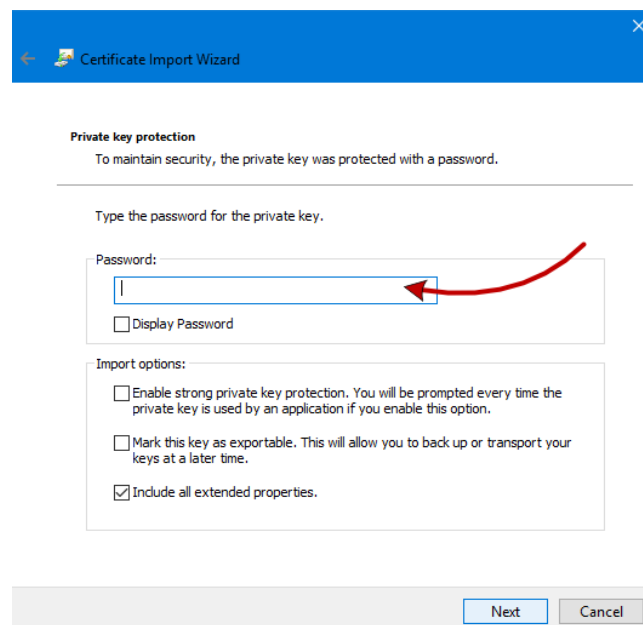
9. If the import is successful, your certificate will be imported to your "Personal Store" and you will be able to access the MONROE user interface by selecting it when prompted by your browser. Notice that you may still get a warning about the validity of the server certificate.

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 (please, 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):

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.

Required interfaces and active-data quota: By default, experiments do not have access to the node network interfaces. The experimenter must explicitly select those interfaces that they want to be available for their container. The selected interfaces will be mapped one-to-one from the physical ones. The experimenter must also specify the active-data quota for each interface, that is, the maximum amount of data that each interface can use. The scheduler multiplies this value by the number of interfaces used and checks the result 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.

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

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

Figure 4: Filters for node selection.

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.

- 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.

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 "2016-05-18 14:45:00 UTC" and can last until "2016-06-18 14:44:00 UTC." The experiment can be scheduled with up to 109 nodes during this period.

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.

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 at the bottom of the page. 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. The following list explains the meaning of the states in which an individual task may be:

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 (see note below).

Failed: The task stopped abnormally.

Canceled: The task was cancelled by the user before being started.

Aborted: The task was aborted by the user after being started.

Defined: The task is known to the scheduler.

Deployed: The corresponding node has already deployed the container and is waiting for its starting time.

Started: The container is running in the designated node. The "download" link for the task results is already available.

Some experiments may be designed to finish after completion. For those ones, the correct finishing state

Figure 6: List of user experiments.

is "Finished." If they are stopped by the scheduler, they probably exceeded the execution time foreseen by the experimenter. However, some 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 experiment results from MONROE's repository

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 to that location to avoid the system to sync temporary files and consume your quota or produce invalid results.

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 that have already been 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 the network functionalities to communicate with any outside servers as needed (e.g., scp some files to an external server).

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.

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
```

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.

For a given country, the platform guarantees that a given op_i does always correspond to the same operator.

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 events.
- Modem connectivity status.
- 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":  "8934014251541036013", "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"}
```

Table 1: Metadata topics.

TOPIC	DESCRIPTION
*.CONNECTIVITY.iccid	Interface, connection events, IP
*.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, …

- Connectivity:

```
{"DataId": "MONROE.META.CONNECTIVITY", "InterfaceName": "usb2", "SequenceNumber": 41412,
"DataVersion": 1, "Timestamp": 1465805641, "NodeId": "54", "MCCMNC": 21404,
"Mode": 6, "RSSI": -80, "ICCID": "8934041514050774002"}
```

- 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://secure.mlab.no/monroewiki/doku.php?id=dataformat`:

- All Metadata messages have a topic according to Table 1 (`https://secure.mlab.no/monroewiki/doku.php?id=metadataformat`). 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.

6 Node status

The user can check the state of the nodes under the tab "Resources." Figure 8 shows an example of the information that is supplied.

Figure 8: Status of the MONROE nodes.

7   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 2: 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

Table 2: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
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)
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

Table 2: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
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
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

Table 2: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
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
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

Table 2: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
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
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

Table 2: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
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
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

Table 2: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
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
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

Table 2: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
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)
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)

Table 2: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
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)
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 runtimes
ucf	3.0030	all	Update Configuration File(s): preserve user changes to config files

Table 2: List of packages installed in `monroe/base`. (Continued)

Name	Version	Architecture	Description
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 3: Field description for metadata topic "MONROE.META.DEVICE.MODEM".

Name	Description
NodeId	Node numerical ID.
Timestamp	Entry timestamp (in milliseconds 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., "sim0", "sim1", "sim2", "eth0", …
Cid	Cell ID.
DeviceMode	Connection mode of the modem (e.g., 2G, 3G, LTE) indicating the radio access technology the modem uses.
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 4: Field description for metadata topic "MONROE.META.DEVICE.GPS".

Name	Description
NodeId	Node numerical ID.
Timestamp	Entry timestamp (in milliseconds since UNIX epoch).
DataId	Metadata topic.

Table 4: Field description for metadata topic "MONROE.META.DEVICE.GPS". (Continued)

Name	Description
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).
SatelliteCount	Number of satellites being tracked.
Nmea	Raw NMEA string from the GPS receiver.

Table 5: Field description for metadata topic "MONROE.META.CONNECTIVITY".

Name	Description
NodeId	Node numerical ID.
Timestamp	Entry timestamp (in milliseconds 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, e.g., "sim0", "sim1", "sim2", "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 6: Field description for metadata topic "MONROE.META.NODE.SENSOR".

Name	Description
NodeId	Node numerical ID.
Timestamp	Entry timestamp (in milliseconds 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 7: Field description for metadata topic "MONROE.META.NODE.EVENT".

Name	Description
NodeId	Node numerical ID.
Timestamp	Entry timestamp (in milliseconds 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 8: Field description for metadata topic "MONROE.EXP.PING".

Name	Description
NodeId	Node numerical ID.
Guid	Unique experiment identifier.
Timestamp	Entry timestamp (in milliseconds 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.
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 9: Field description for metadata topic "MONROE.EXP.HTTP.DOWNLOAD".

| Name	Description
NodeId	Node numerical ID.
Guid	Unique experiment identifier.
Timestamp	Entry timestamp (in milliseconds 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:[2,3]
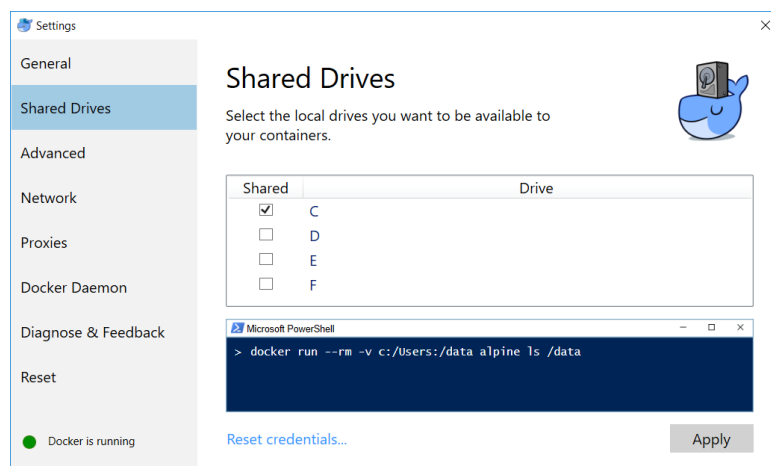
---

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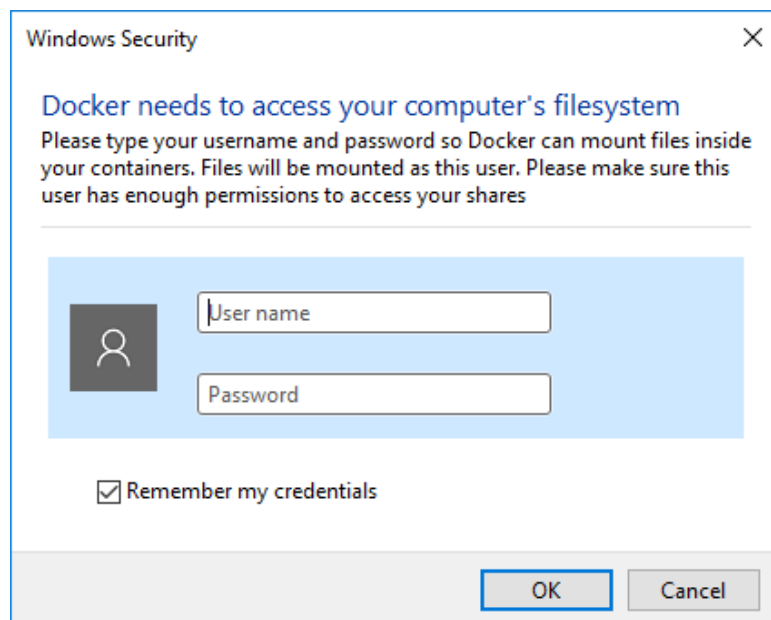[3] Thanks to Lena for pointing to the solution.

1. Access the Docker settings dialog from its taskbar icon:
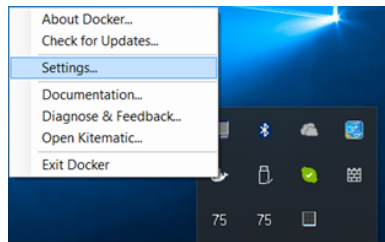
2. From the tab "Shared Drives", select the drive you want to make available to the containers, e.g., "C":

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

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.
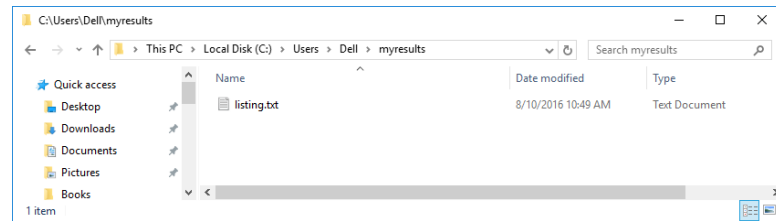
Disclaimer

The views expressed in this document are solely those of the author(s). The European Commission is not responsible for any use that may be made of the information it contains.

All information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.