

OPENBACH, OPEN METROLOGY TESTING FRAMEWORK

Detailed technical presentation
E. Dubois (CNES), D. Pradas (Viveris Technologies)



SUMMARY



- How to get/install OpenBACH
- Auditorium presentation
- How to use the web/scripts interface.
 - Example: Simple ping
- Scenario “strengths” and scenario builder
 - Example: MP-TCP test
- Job development tips

OPENBACH PLATFORM INSTALL



Core Controller and Collector install

- Must be installed on **Ubuntu 16.04** (64 bits)
- **Ansible** installation
- Add the Controller, the Collector and the Auditorium to the **SSH** "known_hosts" file of the host from which you install OpenBACH.



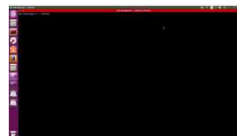
➤ Sources

```
git clone --recursive https://forge.net4sat.org/openbach/openbach.git
```



➤ Install

```
./openbach_installer.py --controller-ip *ip_address* --controller-name  
Openbach-Controller --controller-username *username* --controller-password  
*password* install
```



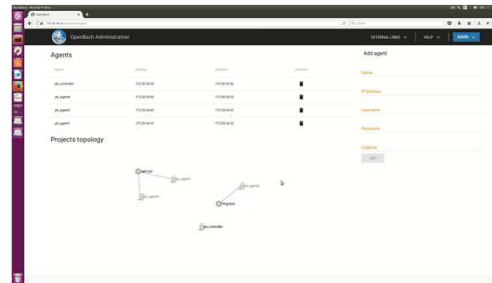
OPENBACH PLATFORM INSTALL



Agent install

- Requirement: Python 2.7, tested on Ubuntu 14.04 and 16.04.
- For SSH: You have to add the host in the known_hosts list of the Controller.

- From web interface:



- From scripts:

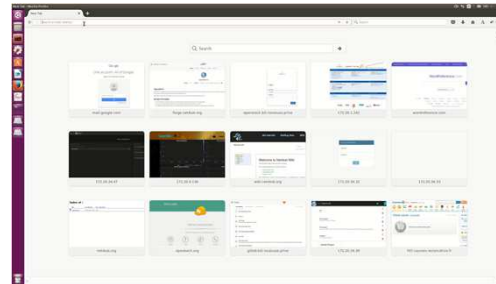
```
./install_agent.py *agent_ip* *collector_ip* *username* *password*  
*name*
```

Auditorium allows to control OpenBACH:

- Create/delete/modify projects/scenarios
- Install/uninstall agents/jobs
- Launch/stop scenarios instances (and check status)
- Launch/stop job instances (and check status)
- List entities/agents/jobs/scenarios and their status
- Etc.

Two ways to control OpenBACH:

- Via the Web interface (auditorium-web)
 - User friendly



- Via the Python scripts (auditorium-scripts)
 - More flexibility

*thanks to command line
execution of python scripts*



HOW TO USE OPENBACH: WEB INTERFACE

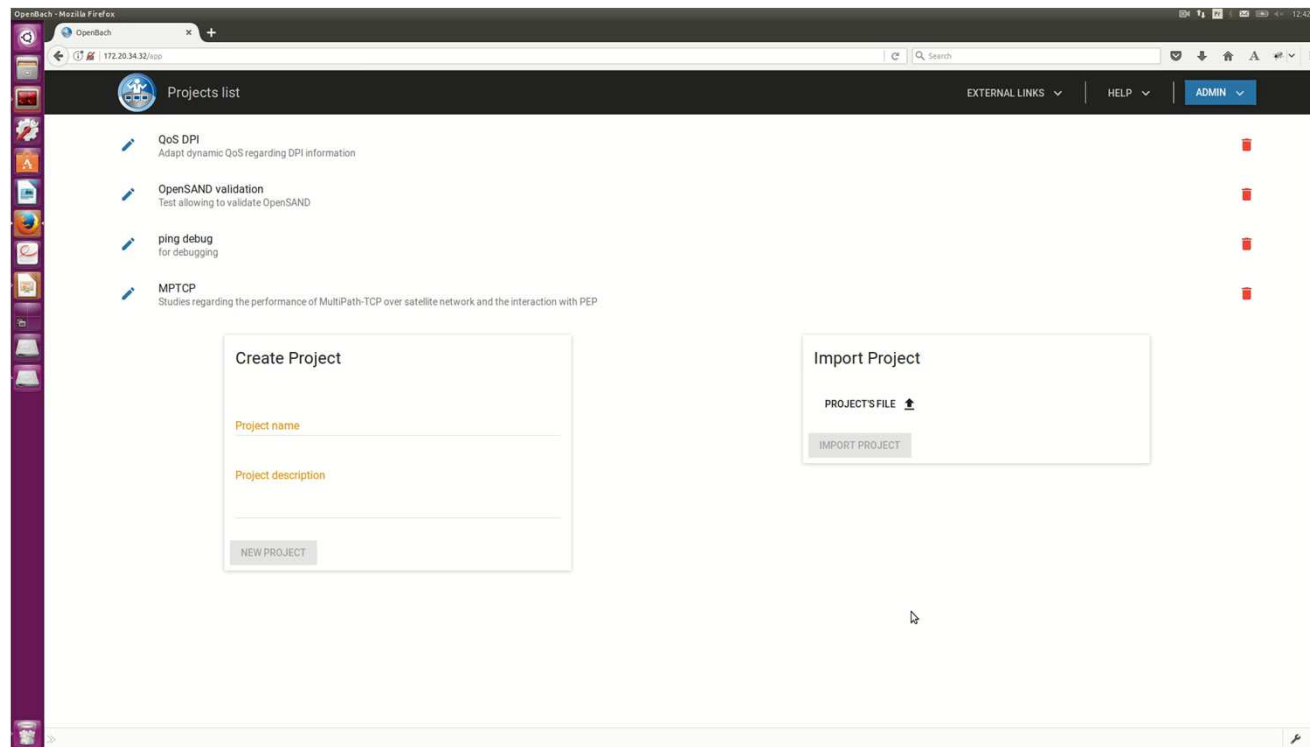


- Create project
- Add a job to OpenBACH (on developer tips)
- Install jobs on Agents
- Create scenario
- Launch scenario instance
- Show results (stats)

Watch "how to ..." on video screencast

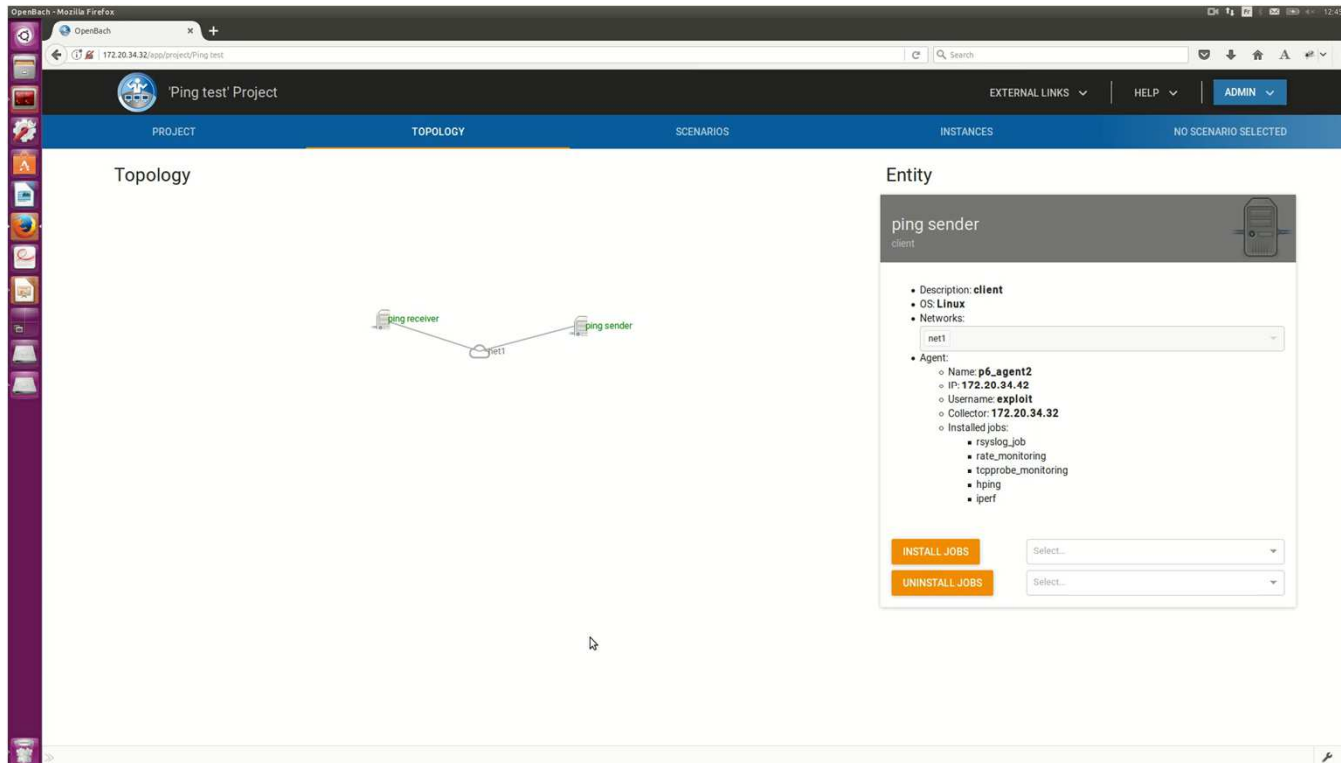
CREATE A PROJECT

- New project and description
 - Add a network topology
 - Associate agents to entities



JOB INSTALL

- Deploy a job fping on an Agent



The screenshot displays the OpenBACH web interface in a Mozilla Firefox browser window. The browser address bar shows the URL `172.20.34.32/app/project/Ping test`. The page title is "Ping test' Project". The navigation bar includes tabs for "PROJECT", "TOPOLOGY", "SCENARIOS", "INSTANCES", and "NO SCENARIO SELECTED". The "TOPOLOGY" tab is active, showing a network diagram with two nodes: "ping receiver" and "ping sender", connected to a central node labeled "net1".

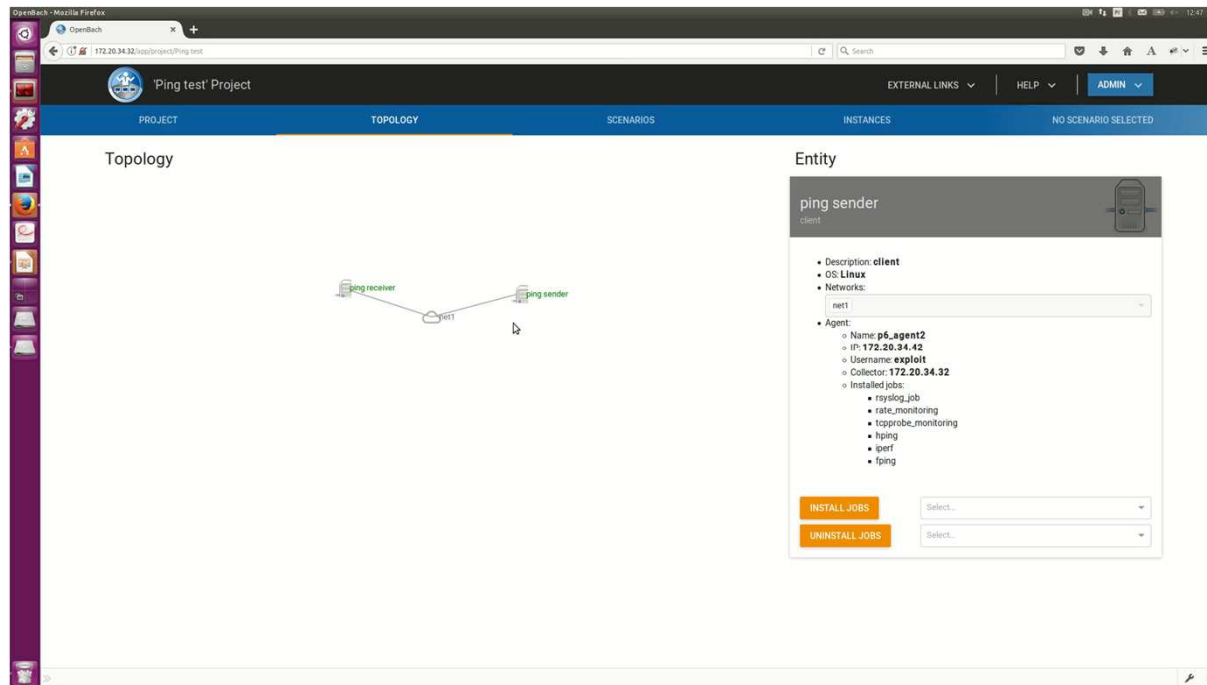
On the right side, the "Entity" panel is open for the "ping sender" client. The details are as follows:

- Description: client
- OS: Linux
- Networks: net1
- Agent:
 - Name: p6_agent2
 - IP: 172.20.34.42
 - Username: exploit
 - Collector: 172.20.34.32
 - Installed jobs:
 - rsyslog_job
 - rate_monitoring
 - topprobe_monitoring
 - fping
 - iperf

At the bottom of the entity panel, there are two orange buttons: "INSTALL JOBS" and "UNINSTALL JOBS", each followed by a "Select..." dropdown menu.

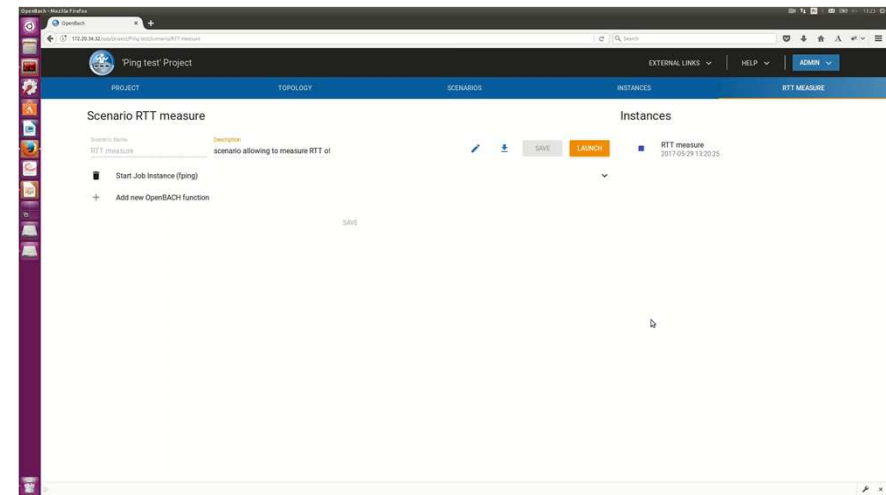
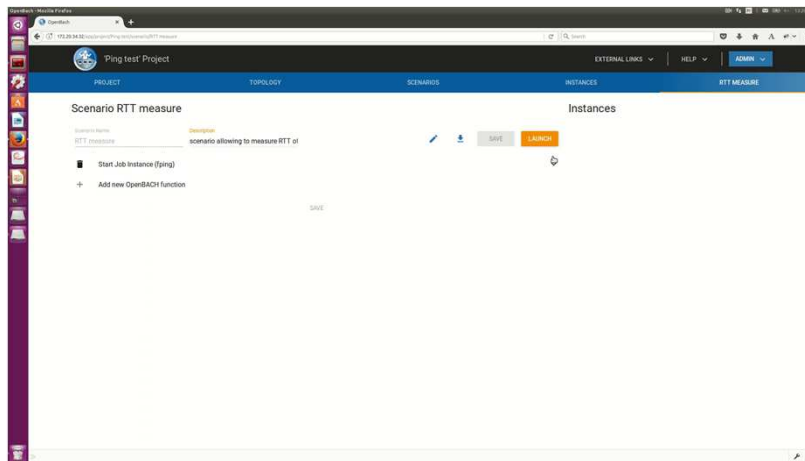
CREATE SCENARIO

- New scenario and description
 - Add openbach functions allowing to start/stop job instances (and subscenarios)
 - Example: fping



LAUNCH SCENARIO INSTANCE

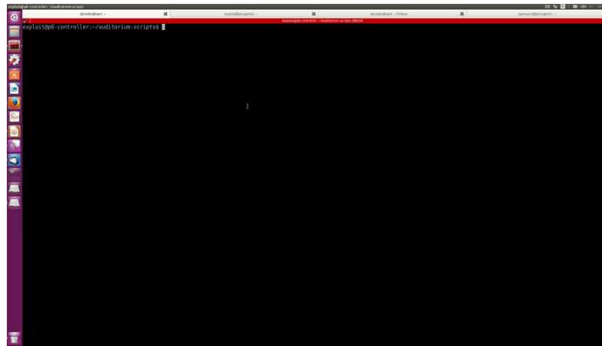
- Start scenario instance
 - Visualize status of instance
 - Show results/statistics on Grafana
 - Show Log messages



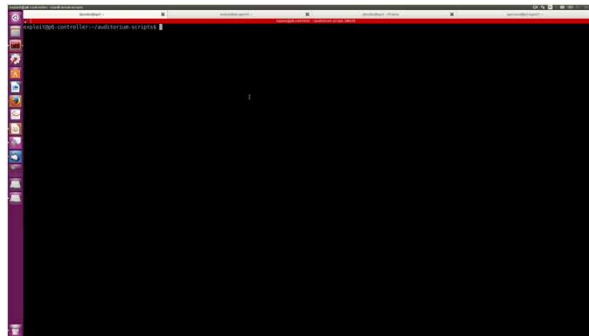
HOW TO USE OPENBACH: SCRIPTS INTERFACE



- List agents



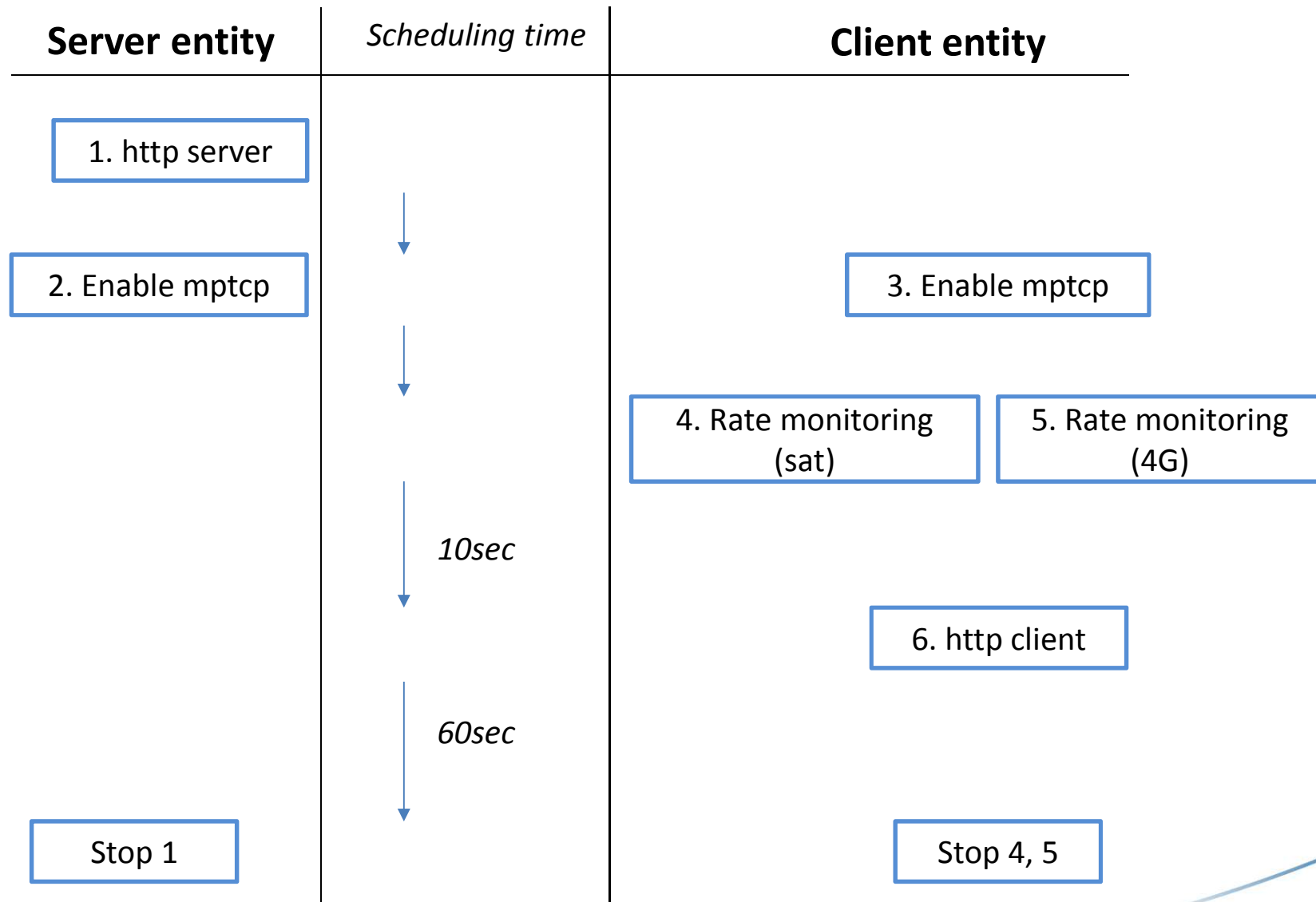
- Install a job (on correct/wrong agent)



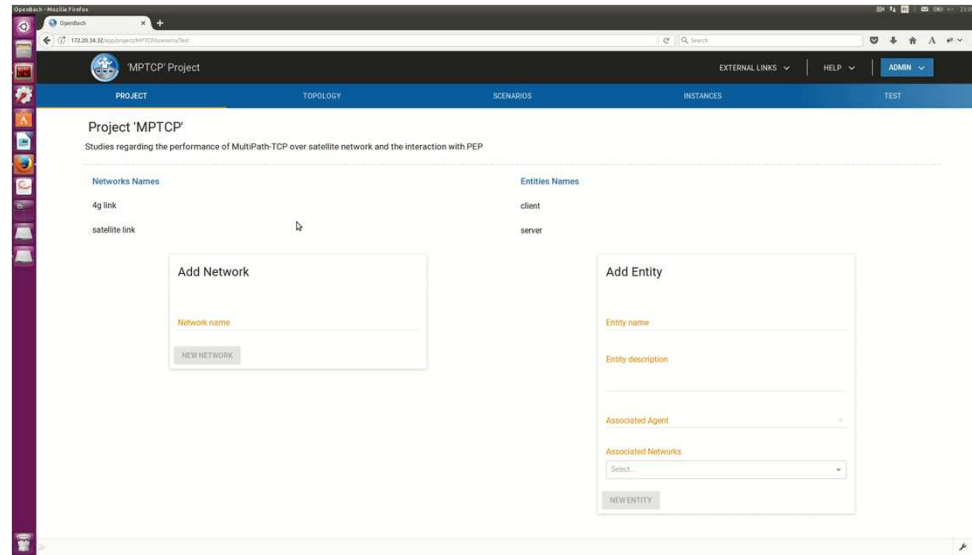
THE « STRENGTHS » OF THE SCENARIO

- Able to create dependencies between openbach-functions and job instances
 - “wait for finished”
 - “Wait for launch”
- Able to create sub scenarios
- Able to launch jobs with accurate scheduling time
- Able to use if/while functions
- Able to pass arguments to the scenario
- Different ways of creating scenarios → based on JSON (able to export to Web and Python interfaces)

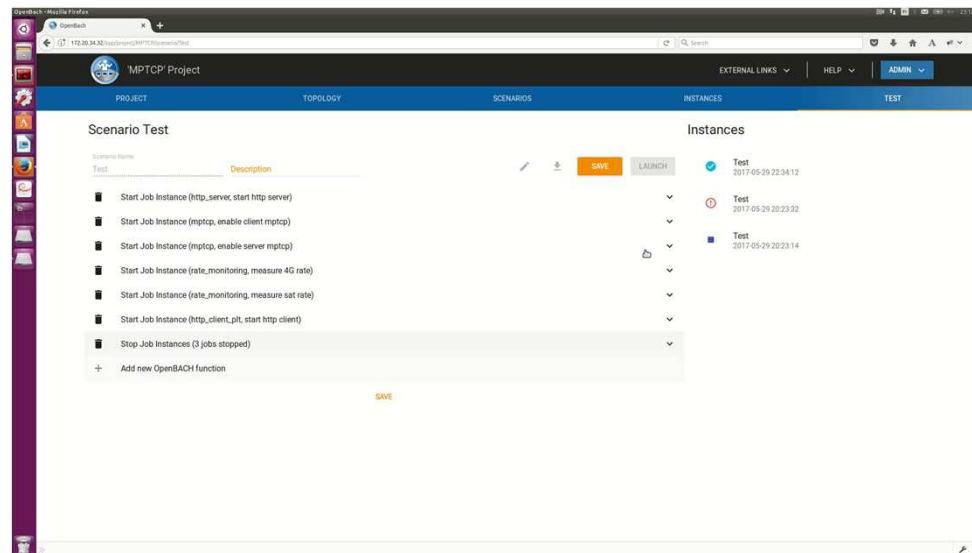
MP-TCP SCENARIO EXAMPLE



MP-TCP SCENARIO EXAMPLE

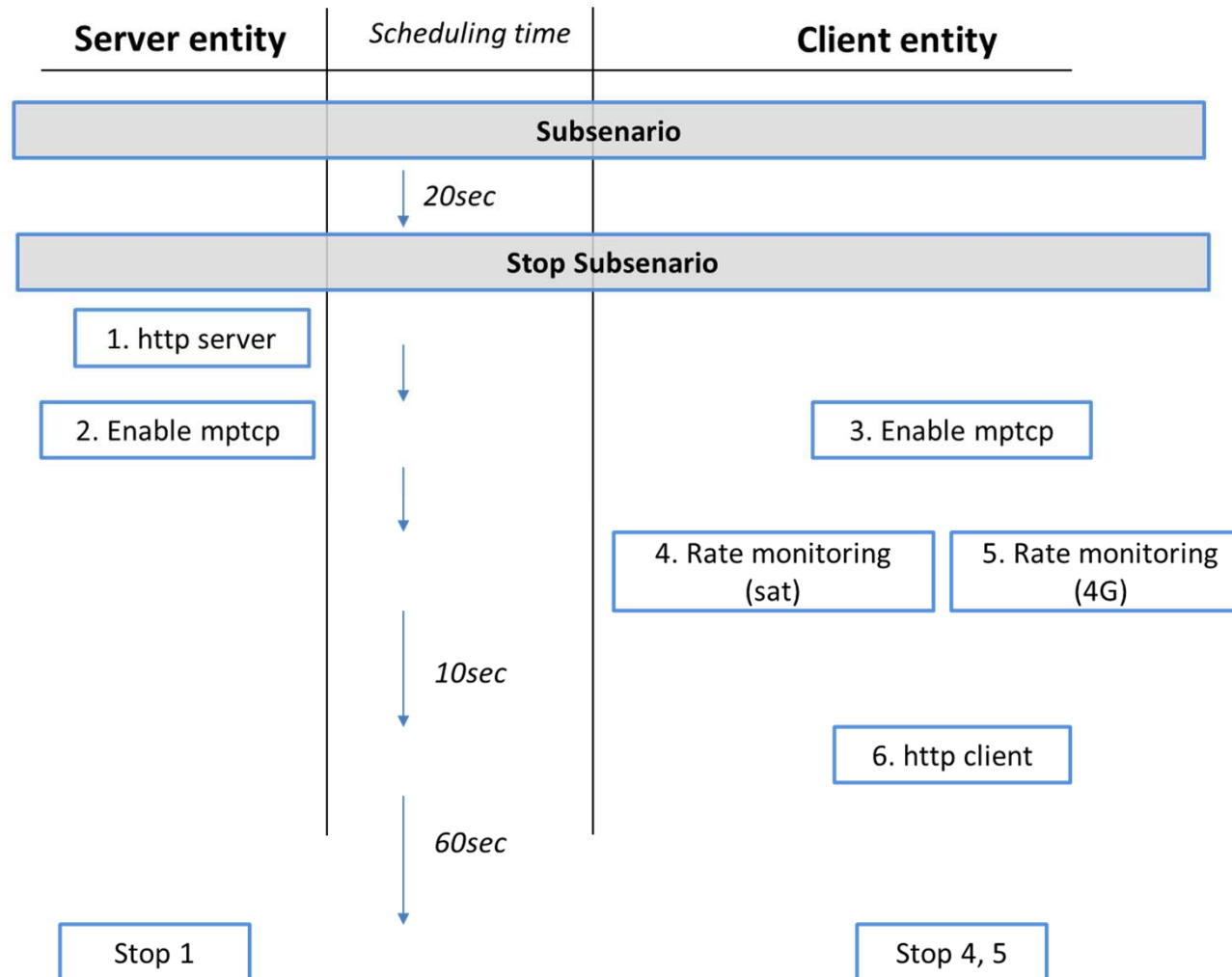


The screenshot shows the OpenBACH web interface for a project named 'MPTCP'. The page title is 'Project 'MPTCP'' and the subtitle is 'Studies regarding the performance of MultiPath-TCP over satellite network and the interaction with PEP'. The interface has a navigation bar with tabs for PROJECT, TOPOLOGY, SCENARIOS, INSTANCES, and TEST. The main content area is divided into two columns: 'Networks Names' and 'Entities Names'. Under 'Networks Names', there are two entries: '4g link' and 'satellite link'. Under 'Entities Names', there are two entries: 'client' and 'server'. Below these lists are two form boxes: 'Add Network' and 'Add Entity'. The 'Add Network' form has a 'Network name' input field and a 'NEW NETWORK' button. The 'Add Entity' form has an 'Entity name' input field, an 'Entity description' input field, an 'Associated Agent' dropdown menu, an 'Associated Networks' dropdown menu, and a 'NEW ENTITY' button.



The screenshot shows the OpenBACH web interface for a scenario named 'Test'. The page title is 'Scenario Test' and the subtitle is 'Description'. The interface has a navigation bar with tabs for PROJECT, TOPOLOGY, SCENARIOS, INSTANCES, and TEST. The main content area is divided into two columns: 'Scenario Test' and 'Instances'. Under 'Scenario Test', there is a list of job instances with a 'SAVE' button and a 'LAUNCH' button. The list includes: 'Start Job Instance (http_server, start http server)', 'Start Job Instance (mptcp, enable client mptcp)', 'Start Job Instance (mptcp, enable server mptcp)', 'Start Job Instance (rate_monitoring, measure 4G rate)', 'Start Job Instance (rate_monitoring, measure sat rate)', 'Start Job Instance (http_client_pt, start http client)', and 'Stop Job Instances (3 jobs stopped)'. There is also a '+ Add new OpenBACH function' button. Under 'Instances', there is a list of instances with a 'SAVE' button and a 'LAUNCH' button. The list includes: 'Test 2017-05-29 22:04:12', 'Test 2017-05-29 20:23:32', and 'Test 2017-05-29 20:23:14'.

MP-TCP SUBSCENARIO EXAMPLE



16

MP-TCP SUBSCENARIO EXAMPLE



A screenshot of the OpenBACH web interface in a Mozilla Firefox browser. The browser address bar shows the URL "172.20.34.32/app/project/MPTCP/scenario/Test". The page title is "'MPTCP' Project". The navigation menu includes "PROJECT", "TOPOLOGY", "SCENARIOS" (highlighted), "INSTANCES", and "TEST". In the top right, there are links for "EXTERNAL LINKS", "HELP", and "ADMIN". The main content area shows two status indicators: "links validation" with "2 openbach functions" and "Test" with "9 openbach functions". Below these are two panels: "Create Scenario" with a text input field labeled "Scenario name" and a "NEW SCENARIO" button; and "Import Scenario" with a "SCENARIO'S FILE" upload icon and an "IMPORT SCENARIO" button. A vertical sidebar with various icons is visible on the left side of the browser window.

THE SCENARIO BUILDER (PYTHON)



- **Objectives: Python API that makes the creation of scenarios easier and programmable.**
- *“Interface between Python code and JSON scenario definition”*
- Exports scenario in JSON
- Use of Python tools and loops/conditions

DATA ACCESS API



- **Objectives: Python module allowing to access the Collector database (logs and stats)**
- For Post-processing tasks
- A job is able to access the desired data (classified by scenario id, job instance id, agent name, job name, etc) and process the data.

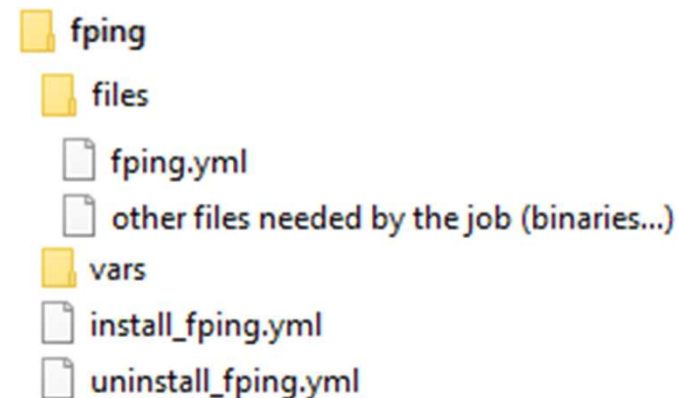
HOW TO WRITE A JOB



- Get your favorite language to do the task you are willing to execute on an Agent;
- Use collect-agent to store logs and data in the collector;
- collect-agent library is written in C++ but provides C and Python wrappers, so these languages are favored when writing jobs;
- collect-agent formats messages to an inner daemon (rstats) on the agent, so manually sending logs and data is still possible (albeit tedious).

HOW TO PACKAGE A JOB

- Jobs deployment to an Agent is done through Ansible
- Most of Ansible configuration is performed by OpenBACH but you still have to provide (un)installation instructions by the mean of 2 playbooks: `install_<job_name>.yml` and `uninstall_<job_name>.yml`
- Instructions about how to launch a job have to be provided by the mean of a configuration file (`<job_name>.yml`): command to launch, accepted args, metadata...
- Regular Ansible rules apply so the expected layout of the files is:



HOW TO ADD A JOB

- Aim: send the job's folder to the controller and register it in the backend.
- Using the auditorium-scripts: upload the folder somewhere on the collector and run, from your install machine:

```
python3 add_job.py <job_name> <uploaded_path>
```
- Using the Frontend: archive (tar.gz) the content of the folder and use the administration tools to send it.

ADDING A JOB FROM THE FRONTEND



OpenBach Administration

EXTERNAL LINKS ▾ | HELP ▾ | ADMIN ▾

- Agents
- Jobs
- Settings
- Logout

Available jobs

Add a new job

Please provide a tar.gz file containing the top-level directory of the job's definition files

New Job Name

DEFINITIONS FILE

ADD JOB

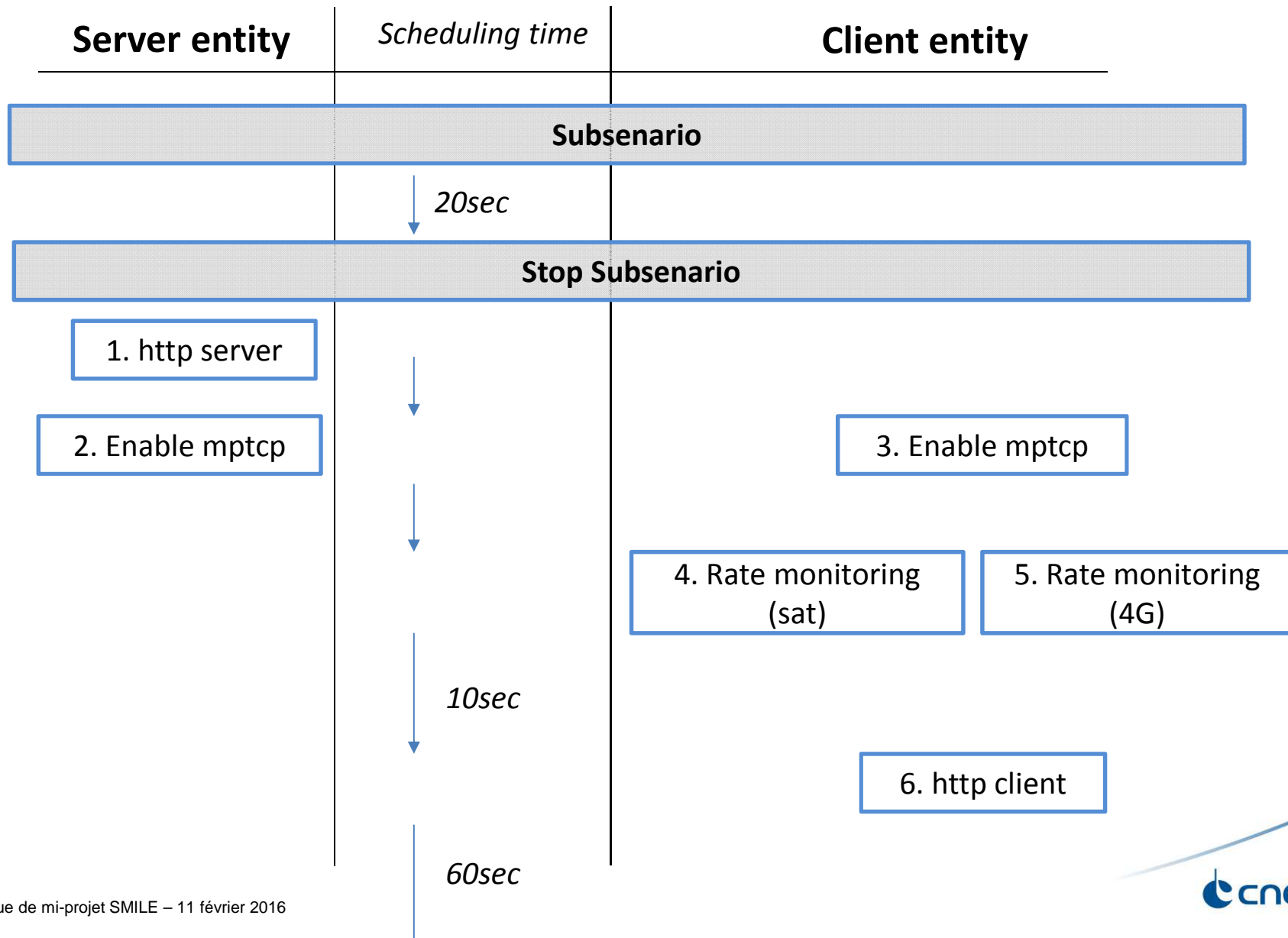
Filter Jobs

d-itg_log		fping		rate_monitoring	
d-itg_rcv		hping		rstats_job	
d-itg_send		http_server		rsyslog_job	
empty_elasticsearch_db		iperf		synchronization	
empty_influxdb_db		netcat		tcp_buffer	
		opensand_gw_deploy		tcpprobe_monitoring	
		opgw			

QUESTIONS ?



Thank you



MP-TCP SCENARIO EXAMPLE

- **Of1:** launch MPTCP-job on client
- **Of2 :** launch MPTCP-job on server
- Wait for of1 and of2 to be finished
- **Of3:** launch http_server (job)
- **Of4:** launch rate_monitoring (job) → measures rate b/s (stat)
- **Of5:** launch http_client (job) during 60s→ measures PLT (stat)
- Wait for http_client to be finished
- **Of6:** stop job http_server / rate_monitoring
- Wait for http_client to be finished
- **Of7:** stop MPTCP-job on server/client (to backup initial configuration of system)