H2020 i**P** **O**ver **I**c**N**- the bet**T**er IP (POINT)

Examples

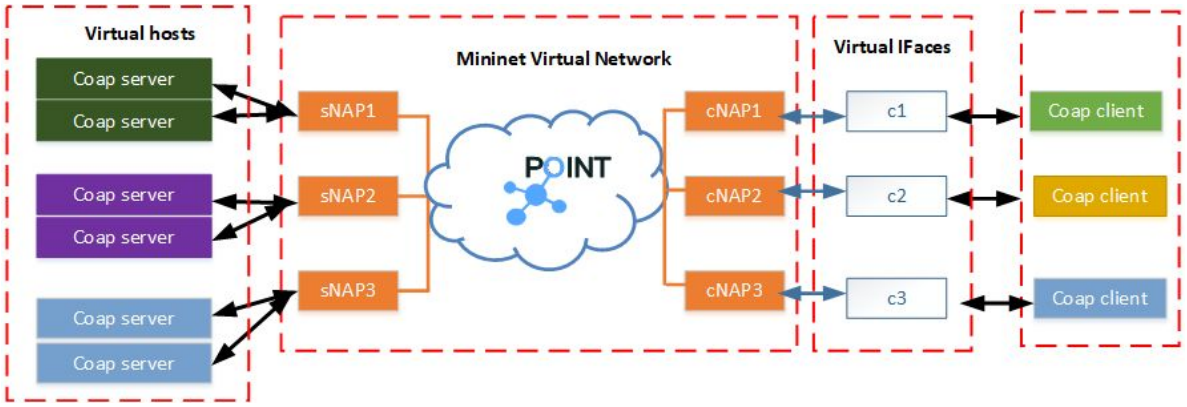
CoAP over ICN demo

List of Authors: Dmitrij Lagutin, Nikos Fotiou

1. Overview

This document provides instructions on how to run a mininet based demo of the POINT CoAP-over-ICN architecture. This demo is included in *apps/coap/tools/mininet* folder. Technical details of this demo are published in "N. Fotiou et al., *ICN enabling CoAP extensions for IP based IoT devices*, Proceedings of the 4th ACM Conference on Information-Centric Networking, pages 218-219, 2017"[1].

The demo setup is illustrated in the following figure.



2. Execution

Firstly, make sure that the dummy CoAP server *tools/coap-apps* has been compiled (`make`). The mininet python script should be invoked with root privileges.

```
$ sudo python CoAP-Over-Point_demo.py
```

With this command:

- The POINT network is set up
- CoAP handlers are executed with the proper configuration files
- CoAP servers are set up and executed

Furthermore, the script creates 3 virtual interfaces in the host machine with the following IP addresses:

```
c1: 172.16.191.1/24
c2: 172.16.192.1/24
c3: 172.16.193.1/24
```

These interfaces can be used as a CoAP proxy by a CoAP client running on the host machine, in order to access the CoAP servers running in the virtual hosts over the (virtualized) POINT network.

[1] https://icn17posters.hotcrp.com/paper/31?cap=031a5Uu5ZDNZous

The demo emulates a building management system with 6 IoT devices. Each device implements two (virtual) thermometers, (virtual) light sensor, (virtual) fan switch, and two (virtual) light switches. IoT devices can be accessed using the following URIs:

- `sensor1.floor1.west.building6`
- `sensor1.floor1.east.building6`
- `sensor1.floor2.west.building6`
- `sensor1.floor2.east.building6`
- `sensor1.floor3.west.building6`
- `sensor1.floor3.east.building6`

Groups of devices can be accessed by using more coarse-grained URIs e.g., a request to "*west.building6*" will delivered to *sensor1.floor1.west.building6*, *sensor1.floor2.west.building6*, and *sensor1.floor3.west.building6*.

In order, for example to access a light measurement sensor using c2 interface as a proxy, the following command can be used (it is assumed that the *libcoap* library has been installed):

```
$ coap-client -P 172.16.192.2 -T abcd
coap://sensor1.floor1.west.building6/a/A2
```

Whereas, a (virtual) light switch can be set on or off using the following command (0=on, 1=off):

```
$ coap-client -P 172.16.192.2 -T abcd -m PUT
coap://floor1.building6/a/D4 -e 1
```

Dummy CoAP servers contain virtual light switches on resource names "/a/D4" and "/a/D5". Values of these switches affect the virtual light sensor ("/a/A2") value. Just like in many real-world CoAP devices, the value reported by the server decreases as the raw sensor value increases. Therefore value 0 denotes the maximum light level while value 1000 is the minimum light level.

Therefore if both lights are on (value of D4 and D5 = 0), the A2 value will be randomized in the range of 0...300. Turning one of the light off, will increase the A2 reading, and turning both of them off, will set the A2 value to the range of 700...1000. The value of temperature sensors is completely random in the range of 0...1000.

The *CoAP-Over-Point_demo.py* script has the following configuration options:

- emulateDemo = True/False, if this is set to True, dummy CoAP servers are started, allowing testing of the setup without real hardware.
- configuration_folder = "Demo", this points to the folders under *tools/mininet/click_conf*, which contains Blackadder topology configuration files and *tools/mininet/config* that contains configuration files of server-side NAPs.

2.1 Using real CoAP devices

The *CoAP-Over-Point_demo.py* script can also be used with real CoAP devices. In order to have a realistic setup to test the group communication and CoAP observe, a separate network interface should be used for each server-side NAP. This can achieved either by the computer having multiple physical network interfaces, or using a single interface with multiple VLANs (Virtual LAN) and a VLAN-enabled network switch. The mininet script provides a built-in support for using VLANs.

An example of using real CoAP devices with VLANs is provided below:

1. Configure VLAN switch, and connect CoAP devices to it. In this example VLAN names are .100, .200, and .300.
2. Setup the VLANs (in this case eth0 interface is used):
   a. `$ sudo modprobe 8021q`
   b. `$ sudo vconfig add eth0 100`
   c. `$ sudo ip link set up eth0.100`
   d. Repeat steps b-c for other VLANs (200, 300)
3. Configure addresses of CoAP devices in the *mininet/config/Demo* folder (snap1.conf, snap2.conf, and snap3.conf) as explained in the *HowTo-CoAP* document.
4. Make sure that *emulateDemo* is set to *False* in the *CoAP-Over-Point_demo.py* script.
5. If some other network interface name is used, the interface name should be updated on the following lines in the *CoAP-Over-Point_demo.py* script:

```
call(["ovs-vsctl", "add-port", "s1", "eth0.100"])
call(["ovs-vsctl", "add-port", "s1", "eth0.200"])
call(["ovs-vsctl", "add-port", "s1", "eth0.300"])
```

6. Run the script normally:

```
$ sudo python CoAP-Over-Point_demo.py
```

2.2 Debugging

The debugging output of the Blackadder, NAPs, and dummy CoAP servers will be stored in the *tools/mininet/output* folder. It contains the following files:

- cnapXCLICK.out and snapXCLICK.out - Blackadder debugging output for various client- and server-side NAPs.
- FW.out, RV.out and TM.out - Blackadder debugging output for Forwarding node, Rendezvous node, and Topology Manager.
- cnapXNAP.out and snapXNAP.out - CoAP handler debugging output for various client- and server-side NAPs.
- eastXSERVER.out and westXSERVER.out - Dummy CoAP server debugging output for various dummy servers. These files can be used to check whether the request did reach the servers.

3. Using the Node-RED based GUI

The demo can also be controlled using a Node-RED based GUI. In order to view the GUI, first copy *apps/toos/nodered/flows_linux.json* into the Node-RED home directory (*~/.node-red*). Then run Node-RED. The UI is available through a browser at: http://127.0.0.1:1880/ui.

The GUI contains sensor readings and toggles for switches for all of the six demo devices. There are also toggles to switch fans and lights based on the wing, floor, or in the whole building using the group communication.

On another tab, there is example of using CoAP observe to observe a single resource simultaneously with two client-side NAPs. In this example a raw light value (coap://floor3.west.building6/a/A2) is shown, therefore if the lights are turned on, the raw observe value will decrease. In a case the tab switch isn't visible, the observe tab can be accessed through http://127.0.0.1:1880/ui/#/0 while the main tab is accessible at http://127.0.0.1:1880/ui/#/1.

Documentation about Node-RED can be found here: https://nodered.org/docs/.