DISTRIBUTED MESSAGE BROKER

ADVANCED NETWORK ARCHITECTURES AND WIRELESS SYSTEMS

LEONARDO POGGIANI

CLARISSA POLIDORI

BRUNO AUGUSTO CASU PEREIRA DE SOUSA

SPECIFICATIONS

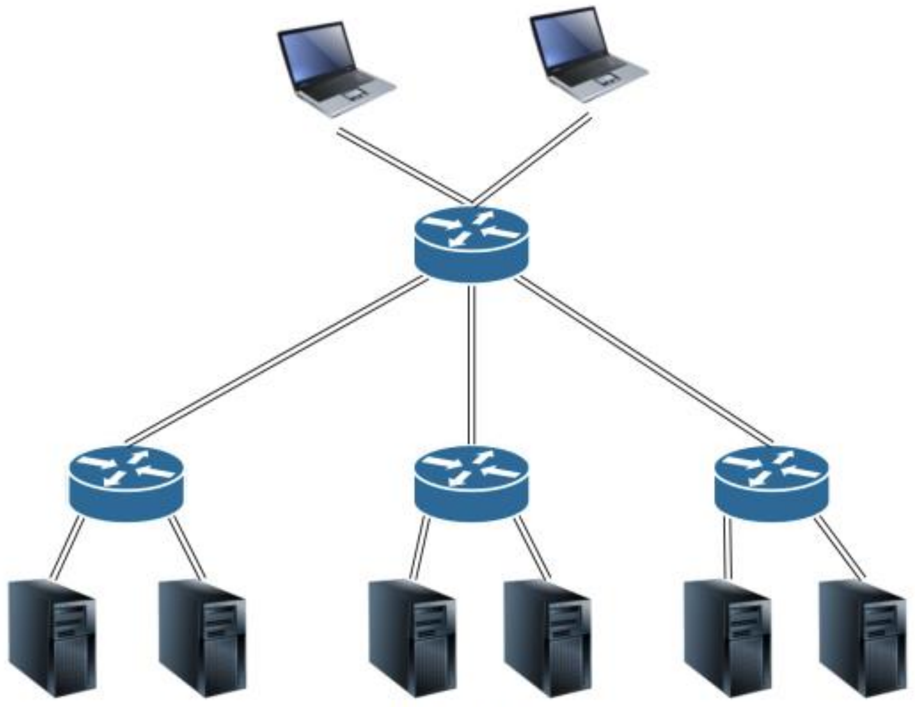


Figure 1 -

Computing nodes (either clients or servers) are connected through an **SDN-based network**.

The network acts as **a distributed message broker**, allowing computing nodes to *subscribe* or *publish* messages to resources.

Each *resource* is represented through a (virtual) IP address. Users can configure the system through a **REST-based Northbound interface**.

FUNCTIONALITIES

The system should:

- allow a user to **create** a resource;
- allow a user to **subscribe** a computing node to a resource;
- allow a computing node to **publish** a message on a resource;
- whenever a message is published on a resource, **deliver** it to every subscriber of the corresponding resource.

REST INTERFACE

http://controller_ip:8080/db/subscribers/RESOURCE_IP/json

http://controller_ip:8080/db/resources/json

REST INTERFACE

- Endpoint */resources*:
  - **GET**: Retrieves the list of registered resources.
  - **POST**: Creates a resource assigning to it a virtual address.
  - **DELETE**: Removes a resource from the registered resources.
- Endpoint */subscribers*:
  - **GET**: Retrieves the list of users subscribed to a resource.
  - **POST**: Adds a user to the list of subscribed users of a resource.
  - **DELETE**: Removes a user from the list of subscribed users to a resource.

DISTRIBUTED MESSAGE BROKER

DESIGN CHOICE

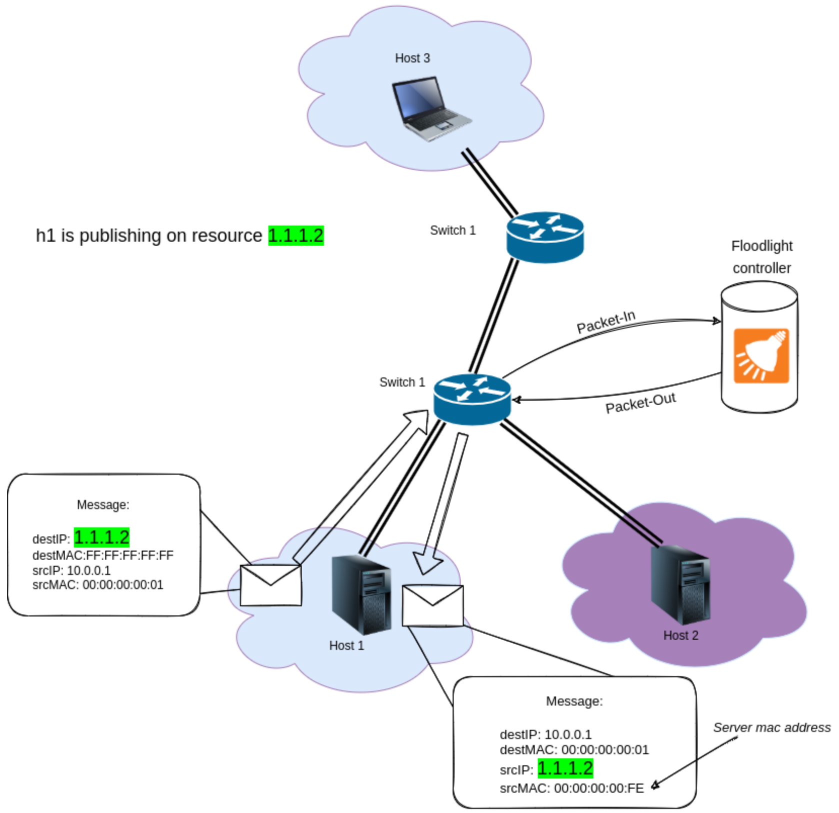
- **Only Packet-In and Packet-Out** instead of *flow-mods*
- **Only UDP traffic**
- ARP packets addressed to **virtual resources**
- **Custom forwarding module** instead of the *default one.*

UNWANTED BEHAVIOR

In our network we don't want to be possible that:

- A user send a message to another user using the **real (physical) destination address**.
- A user receive a message from a resource to which it is **not subscribed**.
- A user publish messages on a resource to **which he is subscribed**.

ARP REQUESTS

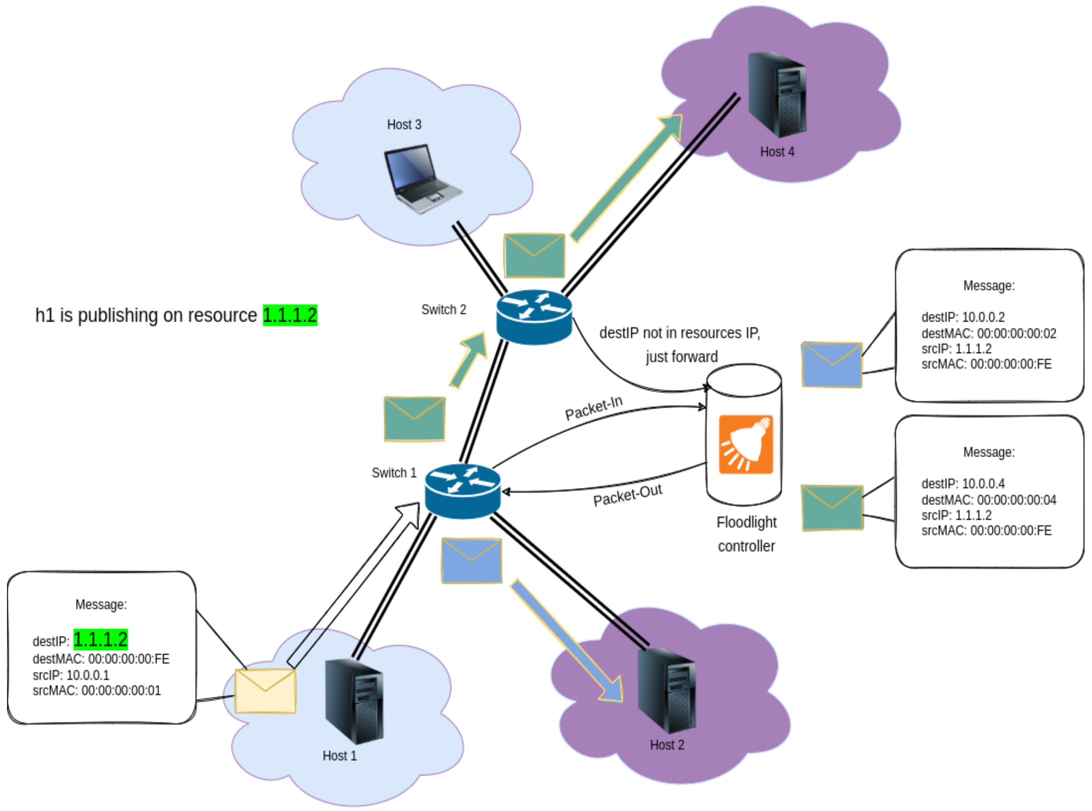


Example: h1 try to publish a message on resource 1.1.1.2

This originate a *packet-in* to the **Floodlight controller** who generates a *packet-out* containing an ARP REPLY message.

This ARP REPLY contains the source MAC address corresponding to the *SERVER_MAC address* and the source IP address equal to the resource address.

IPV4 PACKETS



Message:

destIP: 10.0.0.2
destMAC: 00:00:00:00:02
srcIP: 1.1.1.2
srcMAC: 00:00:00:00:FE

Message:

destIP: 10.0.0.4
destMAC: 00:00:00:00:04
srcIP: 1.1.1.2
srcMAC: 00:00:00:00:FE

Message:

destIP: 1.1.1.2
destMAC: 00:00:00:00:FE
srcIP: 10.0.0.1
srcMAC: 00:00:00:00:01

Example: h1 try to publish a message on resource 1.1.1.2

This originate a *packet-in* to the **Floodlight controller** who generates a *packet-out* performing *two output actions*.

The output actions are redirected on the output port where they can reach (with the shortest path possible) their destinations.

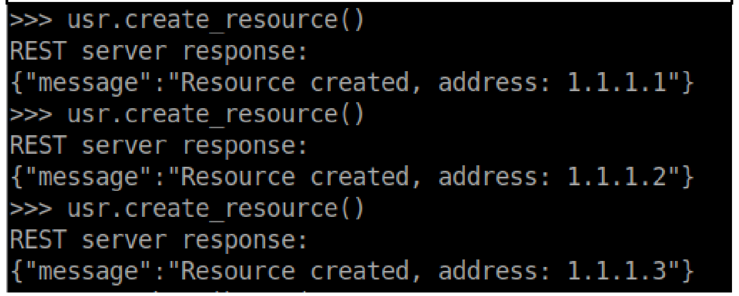
TESTING

TESTING OBJECTIVES

- Tests will evaluate the system implemented features and performance.
- Two scenarios were developed, using the **mininet** software to provide a virtual network topology. For tracing the sent packets, **Wireshark** was used to capture the messages.
- From the virtual network, the available hosts will be used to run simple python scripts to test the nodes communication and message flow, using the developed Broker system.

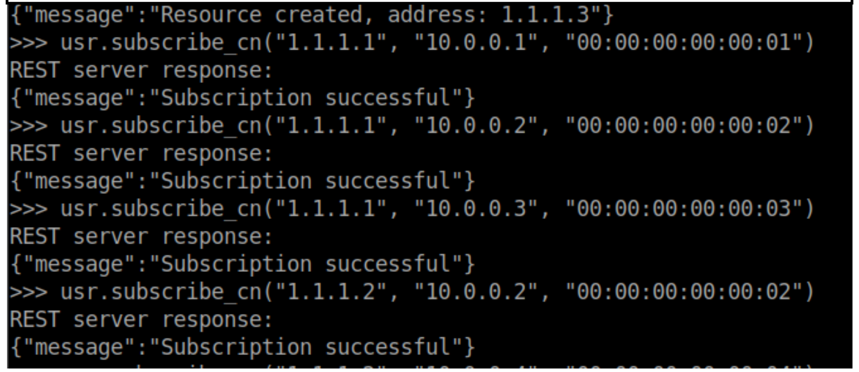
TEST TOPOLOGY

- Creating resources using the Broker REST interface

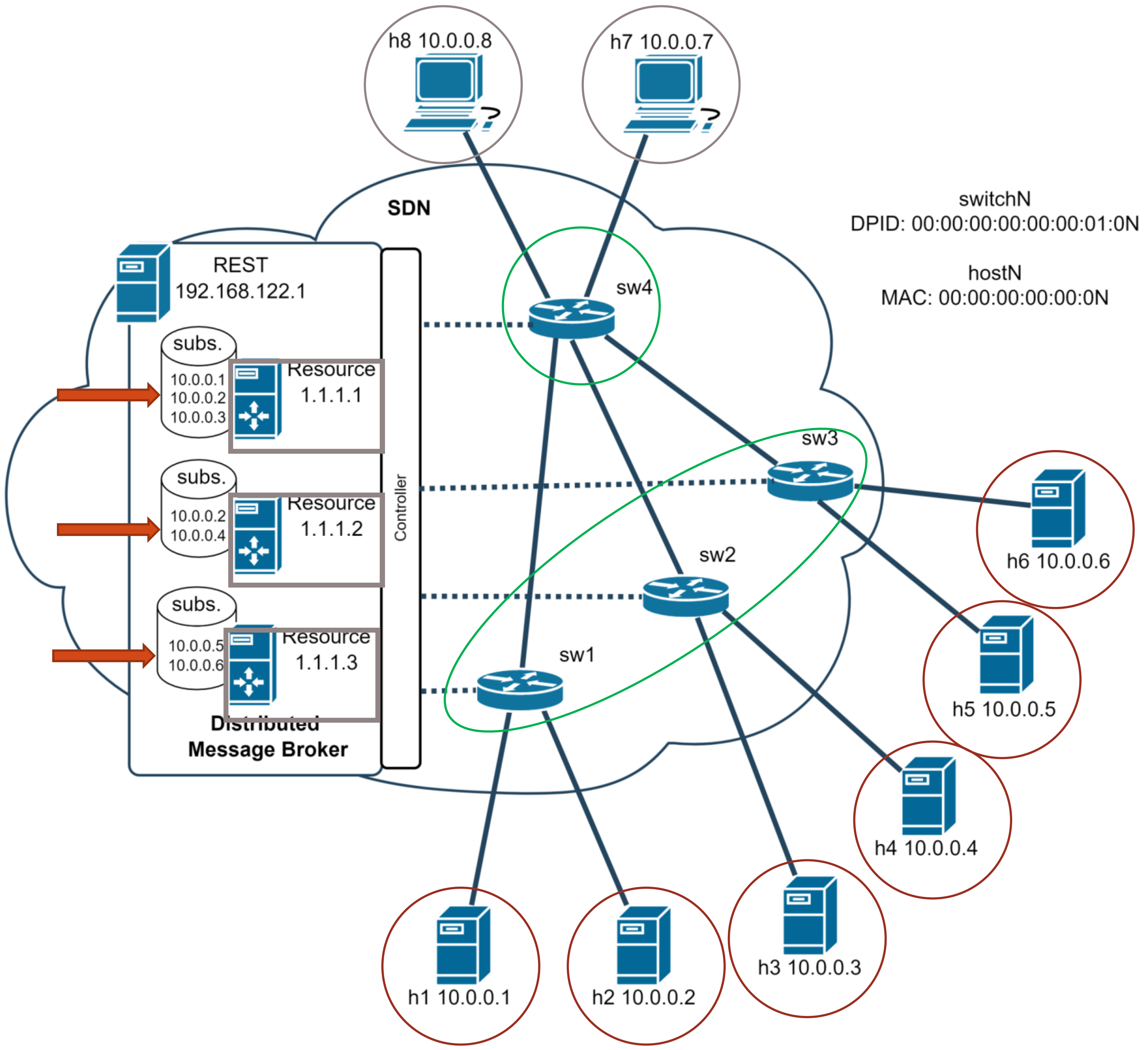


```
>>> usr.create_resource()
REST server response:
{"message":"Resource created, address: 1.1.1.1"}
>>> usr.create_resource()
REST server response:
{"message":"Resource created, address: 1.1.1.2"}
>>> usr.create_resource()
REST server response:
{"message":"Resource created, address: 1.1.1.3"}
```

- Subscribing the computing nodes using the Broker REST interface

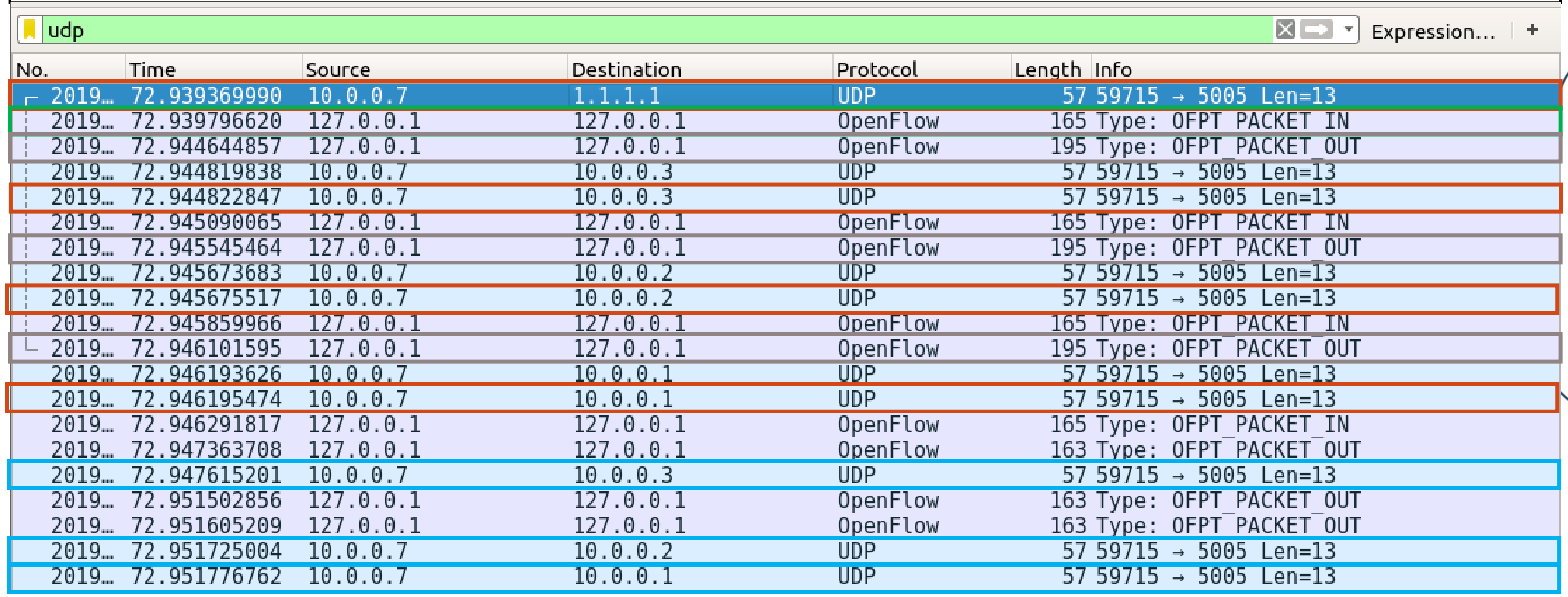


```
{"message":"Resource created, address: 1.1.1.3"}
>>> usr.subscribe_cn("1.1.1.1", "10.0.0.1", "00:00:00:00:00:01")
REST server response:
{"message":"Subscription successful"}
>>> usr.subscribe_cn("1.1.1.1", "10.0.0.2", "00:00:00:00:00:02")
REST server response:
{"message":"Subscription successful"}
>>> usr.subscribe_cn("1.1.1.1", "10.0.0.3", "00:00:00:00:00:03")
REST server response:
{"message":"Subscription successful"}
>>> usr.subscribe_cn("1.1.1.2", "10.0.0.2", "00:00:00:00:00:02")
REST server response:
{"message":"Subscription successful"}
```

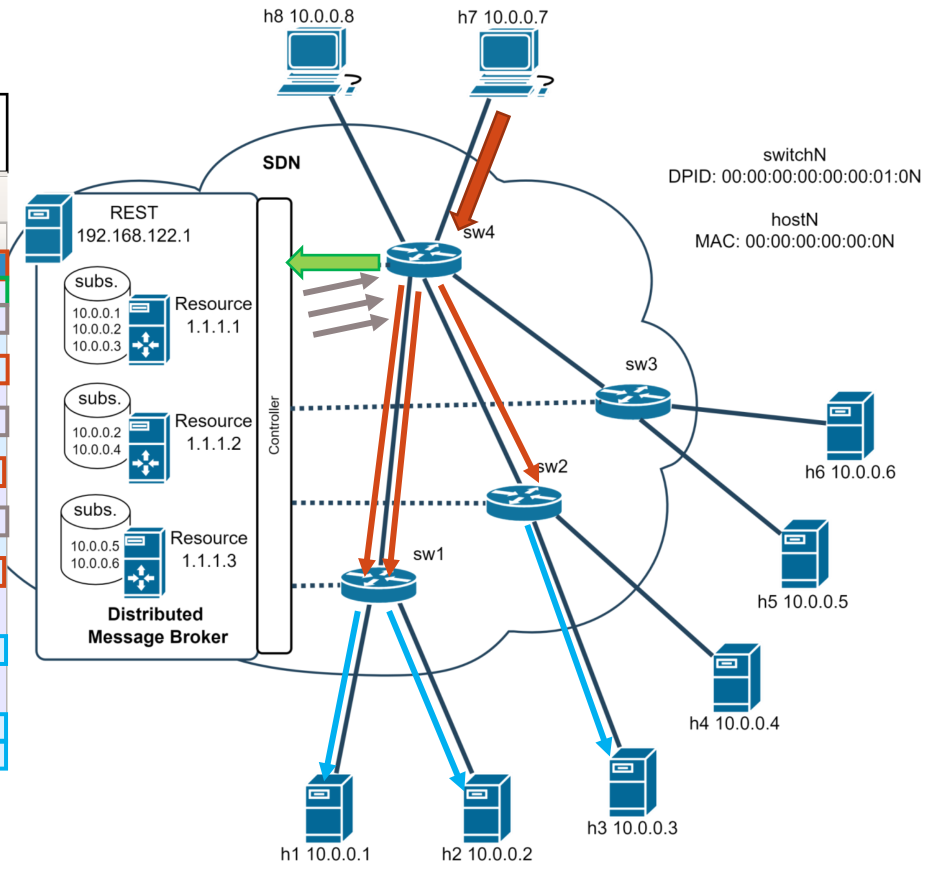


TESTING: PUBLISHING MESSAGES IN THE NETWORK

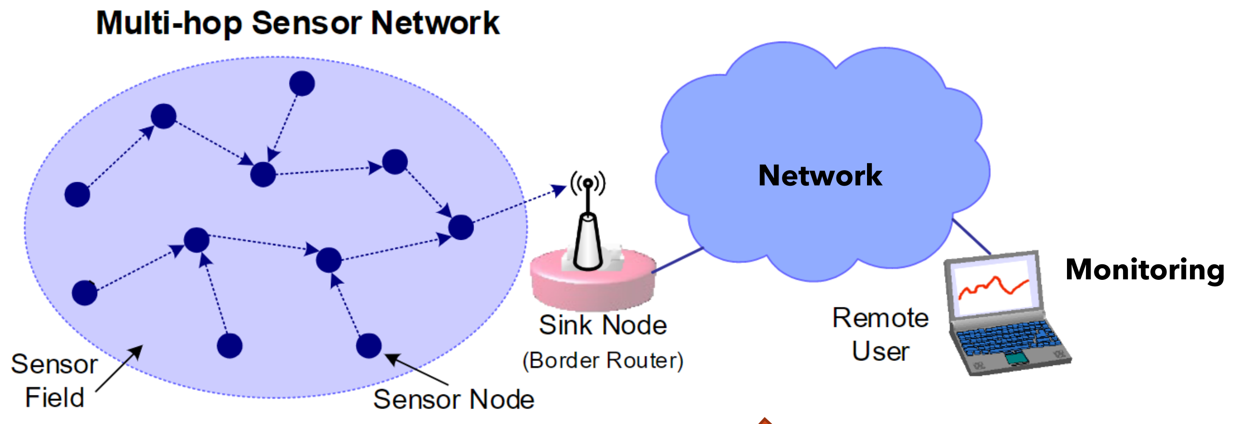
- Node h7 publishing on resource 1.1.1.1

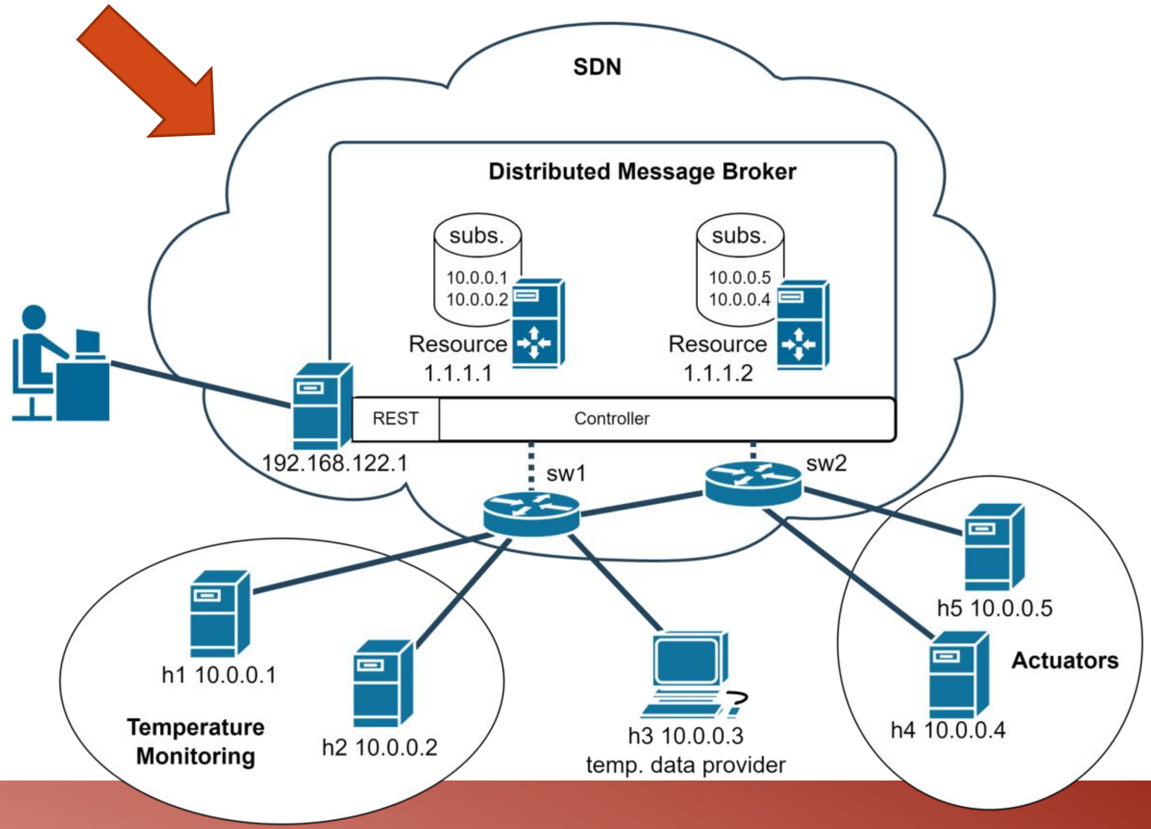


```
udp                                                                          Expression…  +
No.       Time           Source      Destination  Protocol   Length Info
2019…  72.939369990  10.0.0.7    1.1.1.1      UDP            57 59715 → 5005 Len=13
2019…  72.939796620  127.0.0.1   127.0.0.1    OpenFlow      165 Type: OFPT_PACKET_IN
2019…  72.944644857  127.0.0.1   127.0.0.1    OpenFlow      195 Type: OFPT_PACKET_OUT
2019…  72.944819838  10.0.0.7    10.0.0.3     UDP            57 59715 → 5005 Len=13
2019…  72.944822847  10.0.0.7    10.0.0.3     UDP            57 59715 → 5005 Len=13
2019…  72.945090065  127.0.0.1   127.0.0.1    OpenFlow      165 Type: OFPT_PACKET_IN
2019…  72.945545464  127.0.0.1   127.0.0.1    OpenFlow      195 Type: OFPT_PACKET_OUT
2019…  72.945673683  10.0.0.7    10.0.0.2     UDP            57 59715 → 5005 Len=13
2019…  72.945675517  10.0.0.7    10.0.0.2     UDP            57 59715 → 5005 Len=13
2019…  72.945859966  127.0.0.1   127.0.0.1    OpenFlow      165 Type: OFPT_PACKET_IN
2019…  72.946101595  127.0.0.1   127.0.0.1    OpenFlow      195 Type: OFPT_PACKET_OUT
2019…  72.946193626  10.0.0.7    10.0.0.1     UDP            57 59715 → 5005 Len=13
2019…  72.946195474  10.0.0.7    10.0.0.1     UDP            57 59715 → 5005 Len=13
2019…  72.946291817  127.0.0.1   127.0.0.1    OpenFlow      165 Type: OFPT_PACKET_IN
2019…  72.947363708  127.0.0.1   127.0.0.1    OpenFlow      163 Type: OFPT_PACKET_OUT
2019…  72.947615201  10.0.0.7    10.0.0.3     UDP            57 59715 → 5005 Len=13
2019…  72.951502856  127.0.0.1   127.0.0.1    OpenFlow      163 Type: OFPT_PACKET_OUT
2019…  72.951605209  127.0.0.1   127.0.0.1    OpenFlow      163 Type: OFPT_PACKET_OUT
2019…  72.951725004  10.0.0.7    10.0.0.2     UDP            57 59715 → 5005 Len=13
2019…  72.951776762  10.0.0.7    10.0.0.1     UDP            57 59715 → 5005 Len=13
```



SENSOR APPLICATION IN LLN



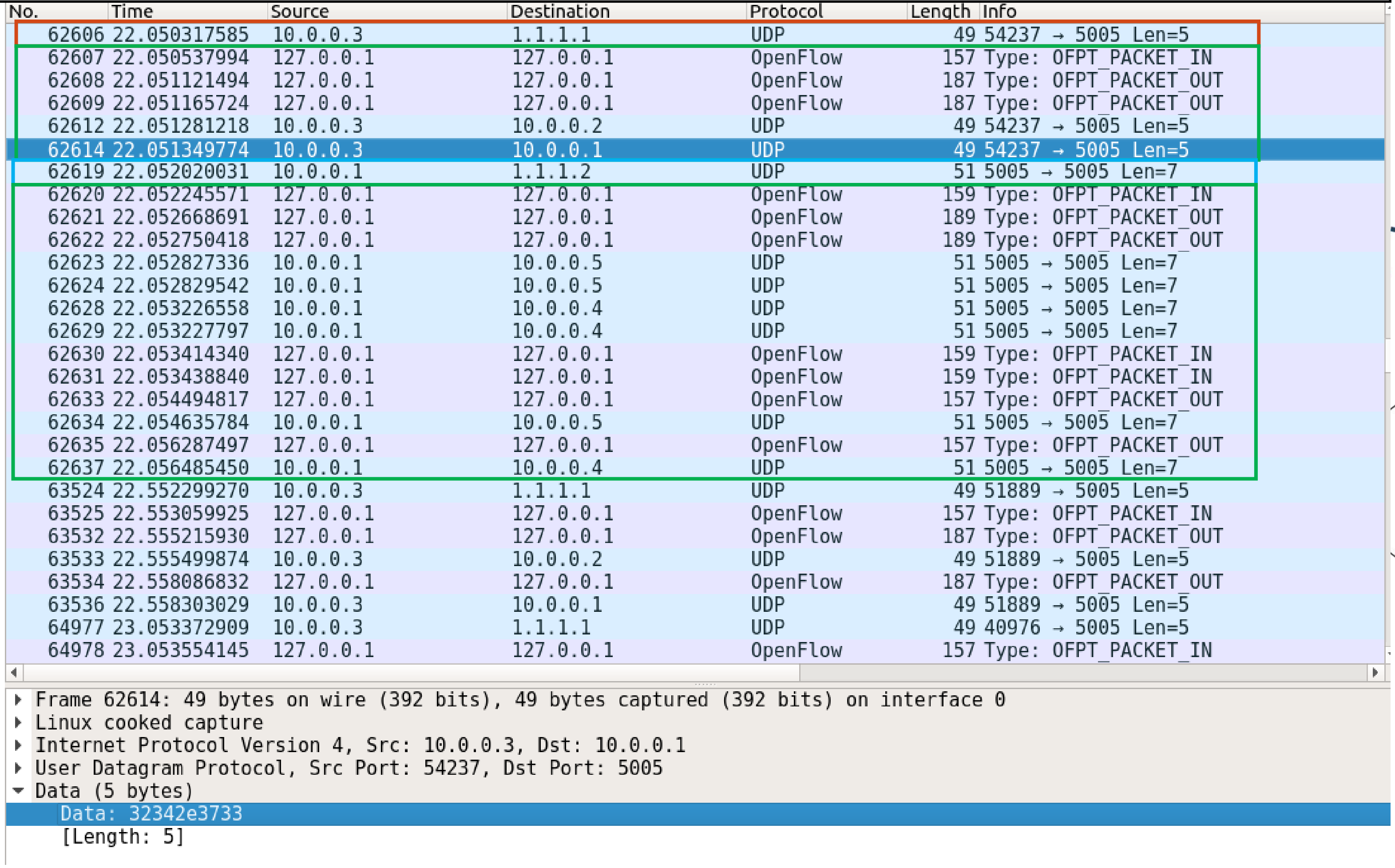


Temperature Monitoring System

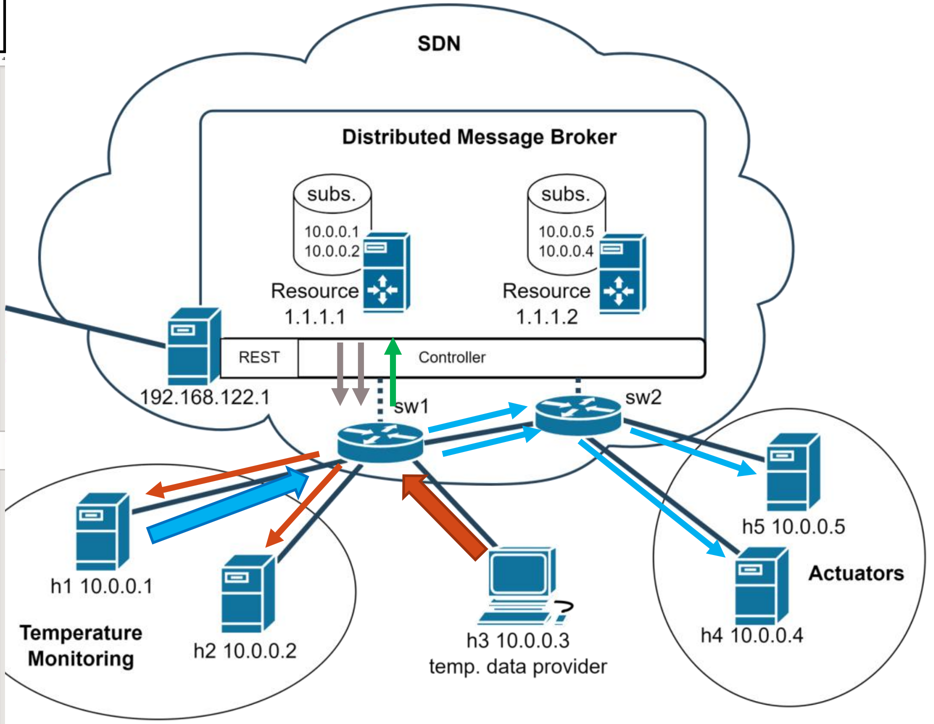
- A second **mininet** topology was implemented, and a simple python application was introduced in the available computing nodes, providing a simulated temperature monitoring system
- The system components are:
  - Data Provider node (acting as a publisher)
  - Monitoring Nodes (pool of servers subscribed to the monitoring topic)
  - Actuator Nodes (pool of server subscribed to the actuators topic)

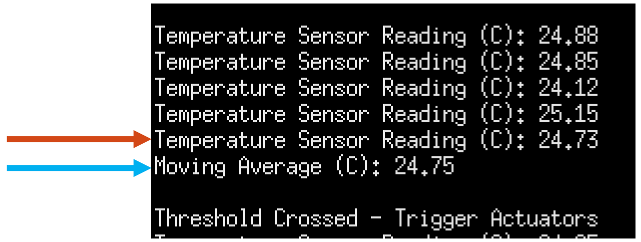
TESTING: SENSOR THRESHOLD CROSSED

- Node h3 (data source) publishing on resource 1.1.1.1 and h1 (monitoring node) publishing on resource 1.1.1.2



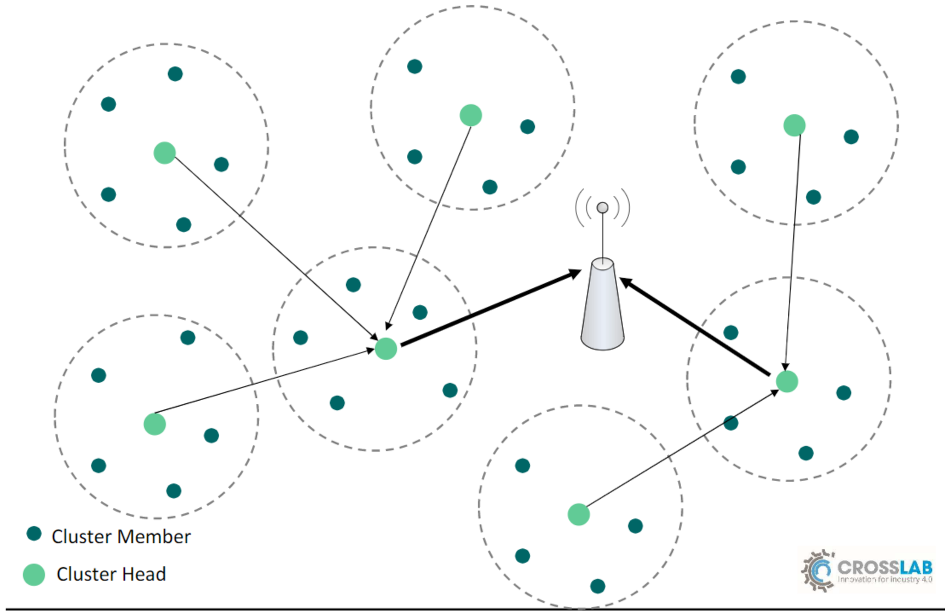
```
No.    Time          Source     Destination  Protocol  Length Info
 62606 22.050317585  10.0.0.3   1.1.1.1      UDP          49 54237 → 5005 Len=5
 62607 22.050537994  127.0.0.1  127.0.0.1    OpenFlow    157 Type: OFPT_PACKET_IN
 62608 22.051121494  127.0.0.1  127.0.0.1    OpenFlow    187 Type: OFPT_PACKET_OUT
 62609 22.051165724  127.0.0.1  127.0.0.1    OpenFlow    187 Type: OFPT_PACKET_OUT
 62612 22.051281218  10.0.0.3   10.0.0.2     UDP          49 54237 → 5005 Len=5
 62614 22.051349774  10.0.0.3   10.0.0.1     UDP          49 54237 → 5005 Len=5
 62619 22.052020031  10.0.0.1   1.1.1.2      UDP          51 5005 → 5005 Len=7
 62620 22.052245571  127.0.0.1  127.0.0.1    OpenFlow    159 Type: OFPT_PACKET_IN
 62621 22.052668691  127.0.0.1  127.0.0.1    OpenFlow    189 Type: OFPT_PACKET_OUT
 62622 22.052750418  127.0.0.1  127.0.0.1    OpenFlow    189 Type: OFPT_PACKET_OUT
 62623 22.052827336  10.0.0.1   10.0.0.5     UDP          51 5005 → 5005 Len=7
 62624 22.052829542  10.0.0.1   10.0.0.5     UDP          51 5005 → 5005 Len=7
 62628 22.053226558  10.0.0.1   10.0.0.4     UDP          51 5005 → 5005 Len=7
 62629 22.053227797  10.0.0.1   10.0.0.4     UDP          51 5005 → 5005 Len=7
 62630 22.053414340  127.0.0.1  127.0.0.1    OpenFlow    159 Type: OFPT_PACKET_IN
 62631 22.053438840  127.0.0.1  127.0.0.1    OpenFlow    159 Type: OFPT_PACKET_IN
 62633 22.054494817  127.0.0.1  127.0.0.1    OpenFlow    157 Type: OFPT_PACKET_OUT
 62634 22.054635784  10.0.0.1   10.0.0.5     UDP          51 5005 → 5005 Len=7
 62635 22.056287497  127.0.0.1  127.0.0.1    OpenFlow    157 Type: OFPT_PACKET_OUT
 62637 22.056485450  10.0.0.1   10.0.0.4     UDP          51 5005 → 5005 Len=7
 63524 22.552299270  10.0.0.3   1.1.1.1      UDP          49 51889 → 5005 Len=5
 63525 22.553059925  127.0.0.1  127.0.0.1    OpenFlow    157 Type: OFPT_PACKET_IN
 63532 22.555215930  127.0.0.1  127.0.0.1    OpenFlow    187 Type: OFPT_PACKET_OUT
 63533 22.555499874  10.0.0.3   10.0.0.2     UDP          49 51889 → 5005 Len=5
 63534 22.558086832  127.0.0.1  127.0.0.1    OpenFlow    187 Type: OFPT_PACKET_OUT
 63536 22.558303029  10.0.0.3   10.0.0.1     UDP          49 51889 → 5005 Len=5
 64977 23.053372909  10.0.0.3   1.1.1.1      UDP          49 40976 → 5005 Len=5
 64978 23.053554145  127.0.0.1  127.0.0.1    OpenFlow    157 Type: OFPT_PACKET_IN

▸ Frame 62614: 49 bytes on wire (392 bits), 49 bytes captured (392 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.0.0.3, Dst: 10.0.0.1
▸ User Datagram Protocol, Src Port: 54237, Dst Port: 5005
▾ Data (5 bytes)
    Data: 32342e3733
    [Length: 5]
```

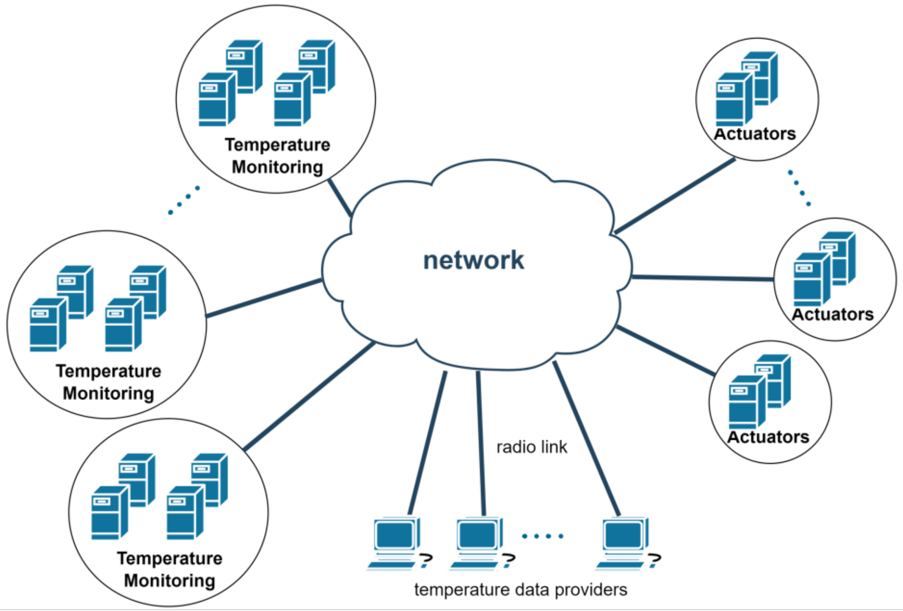




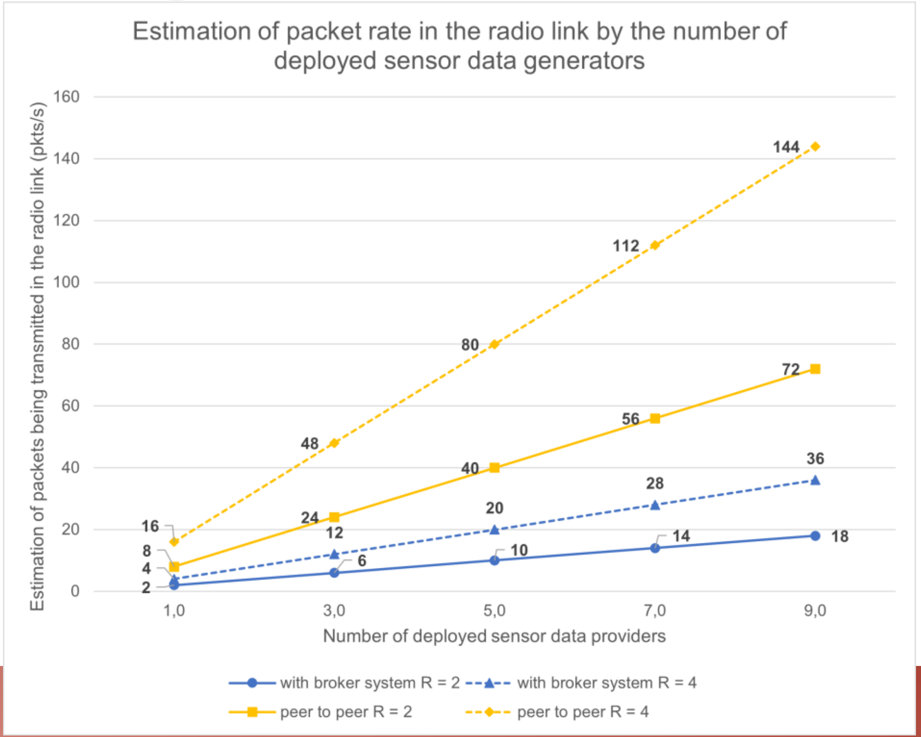
```
Temperature Sensor Reading (C): 24.88
Temperature Sensor Reading (C): 24.85
Temperature Sensor Reading (C): 24.12
Temperature Sensor Reading (C): 25.15
Temperature Sensor Reading (C): 24.73
Moving Average (C): 24.75

Threshold Crossed - Trigger Actuators
```

CONCLUSIONS AND PERFORMANCE EVALUATION





- **System scalability**
  - IoT systems can be designed with multiple clusters of sensor nodes, that transmits its local readings to the monitoring centers.
  - The Broker system then provides a better way to handle an increased volume of messages that needs to reach the aggregator node, reducing the traffic in the Radio Link.
  - The plot shows the expected number of packets transmitted in the Radio Link when using the Broker system and when using P2P communication, with different system workloads.



Rate **R** is the amount of pkts generated by the Data Providers (500 ms temp. Reading interval results in R=2pkts/s)

THANKS FOR THE ATTENTION!

- Project github repo: https://github.com/leonardopoggiani/ANAWS-project
- IoT images extracted from Prof. Anastasi lecturing material

BACKUP SLIDES

DETAILED TESTING OBJECTIVES

- All the subscribers to a resource receive the messages published on it
- Non-subscribers do not see message published on the resource
- A host can subscribe to multiple resources
- A host that unsubscribes no longer receives messages published to that resource
- When a resource is deleted, no host is subscriber anymore and receives nothing
- A host can not send a message to another host directly
- A host cannot publish to a resource to which it is subscribed

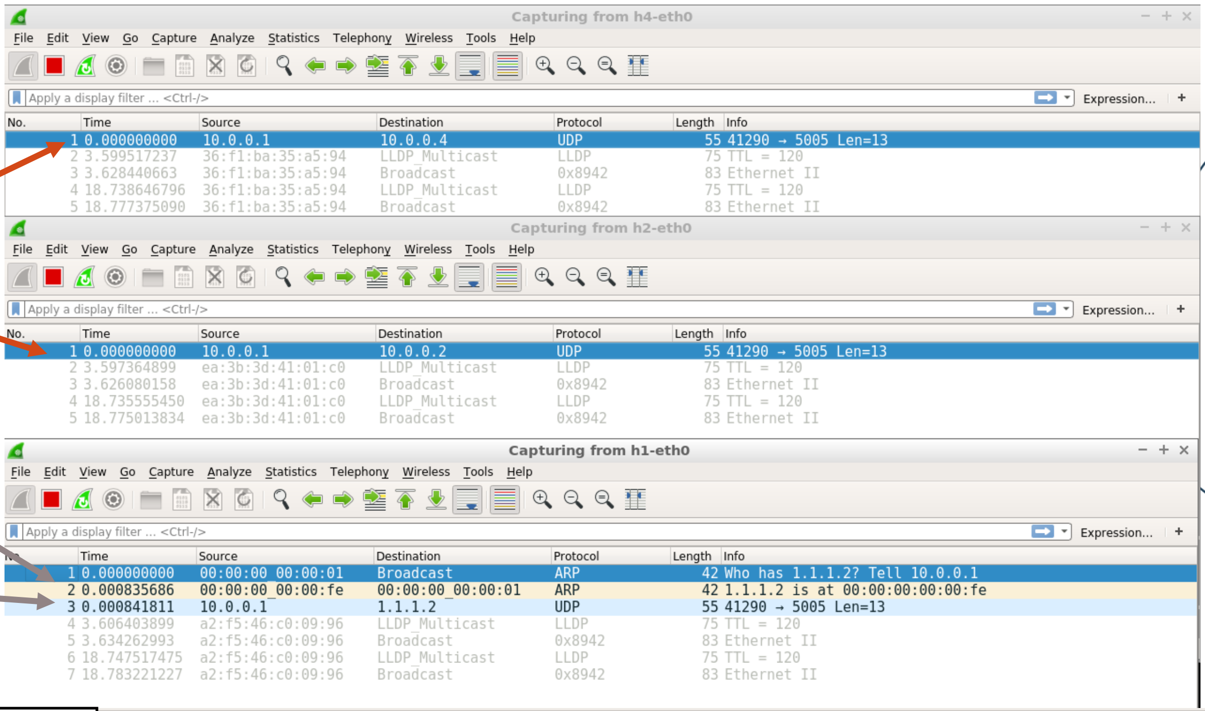
TESTING: PUBLISHING MESSAGES IN THE NETWORK

- Node h1 publishing on resource 1.1.1.2

Received UDP msg (h4 and h2 interfaces - subscribers)

ARP resolution provided by the Broker module

Sent UDP msg (h1 interface - publisher)



Capturing from h4-eth0

```
No.  Time         Source             Destination     Protocol  Length Info
   1 0.000000000  10.0.0.1           10.0.0.4        UDP           55 41290 → 5005 Len=13
   2 3.599517237  36:f1:ba:35:a5:94  LLDP_Multicast  LLDP          75 TTL = 120
   3 3.628440663  36:f1:ba:35:a5:94  Broadcast       0x8942        83 Ethernet II
   4 18.738646796 36:f1:ba:35:a5:94  LLDP_Multicast  LLDP          75 TTL = 120
   5 18.777375090 36:f1:ba:35:a5:94  Broadcast       0x8942        83 Ethernet II
```

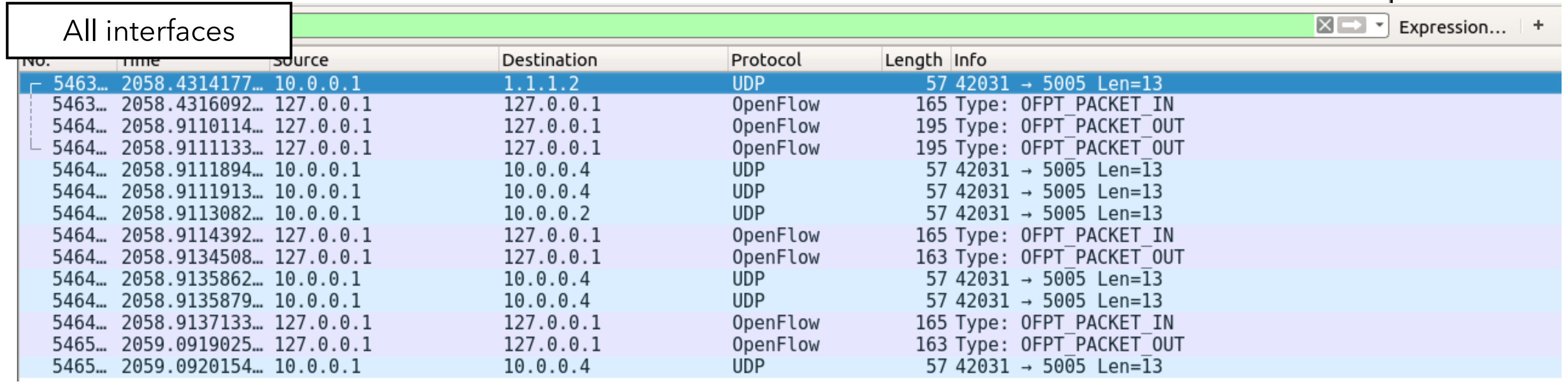
Capturing from h2-eth0

```
No.  Time         Source             Destination     Protocol  Length Info
   1 0.000000000  10.0.0.1           10.0.0.2        UDP           55 41290 → 5005 Len=13
   2 3.597364899  ea:3b:3d:41:01:c0  LLDP_Multicast  LLDP          75 TTL = 120
   3 3.626080158  ea:3b:3d:41:01:c0  Broadcast       0x8942        83 Ethernet II
   4 18.735555450 ea:3b:3d:41:01:c0  LLDP_Multicast  LLDP          75 TTL = 120
   5 18.775013834 ea:3b:3d:41:01:c0  Broadcast       0x8942        83 Ethernet II
```

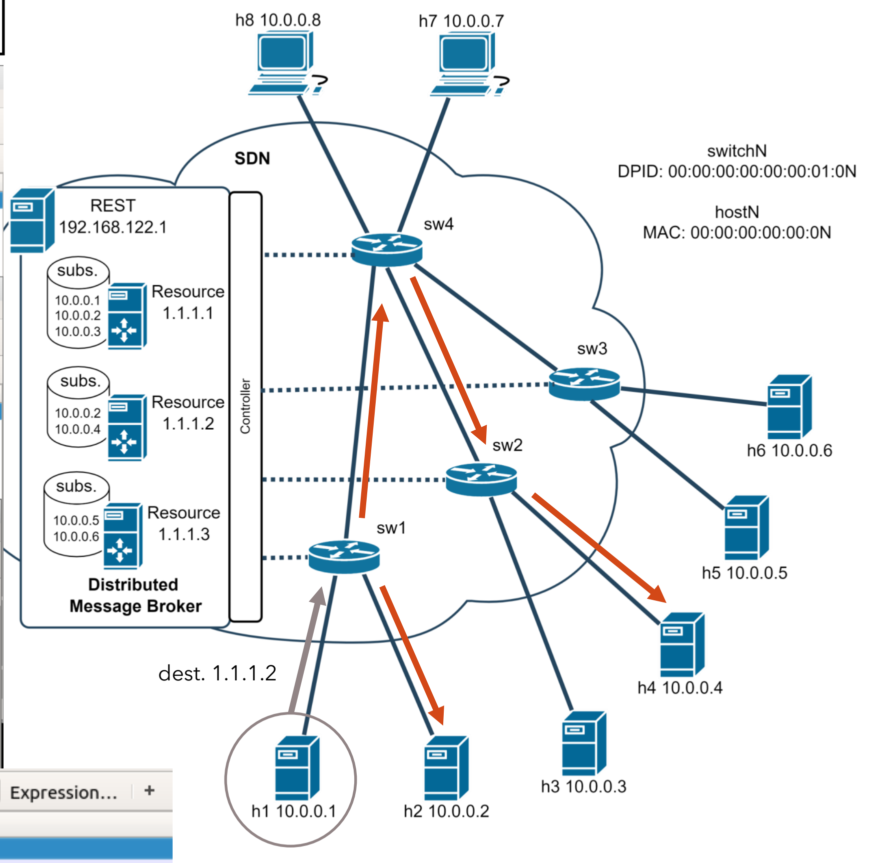
Capturing from h1-eth0

```
No.  Time         Source             Destination        Protocol  Length Info
   1 0.000000000  00:00:00_00:00:01  Broadcast          ARP           42 Who has 1.1.1.2? Tell 10.0.0.1
   2 0.000835686  00:00:00_00:00:fe  00:00:00_00:00:01  ARP           42 1.1.1.2 is at 00:00:00:00:00:fe
   3 0.000841811  10.0.0.1           1.1.1.2            UDP           55 41290 → 5005 Len=13
   4 3.606403899  a2:f5:46:c0:09:96  LLDP_Multicast     LLDP          75 TTL = 120
   5 3.634262993  a2:f5:46:c0:09:96  Broadcast          0x8942        83 Ethernet II
   6 18.747517475 a2:f5:46:c0:09:96  LLDP_Multicast     LLDP          75 TTL = 120
   7 18.783221227 a2:f5:46:c0:09:96  Broadcast          0x8942        83 Ethernet II
```

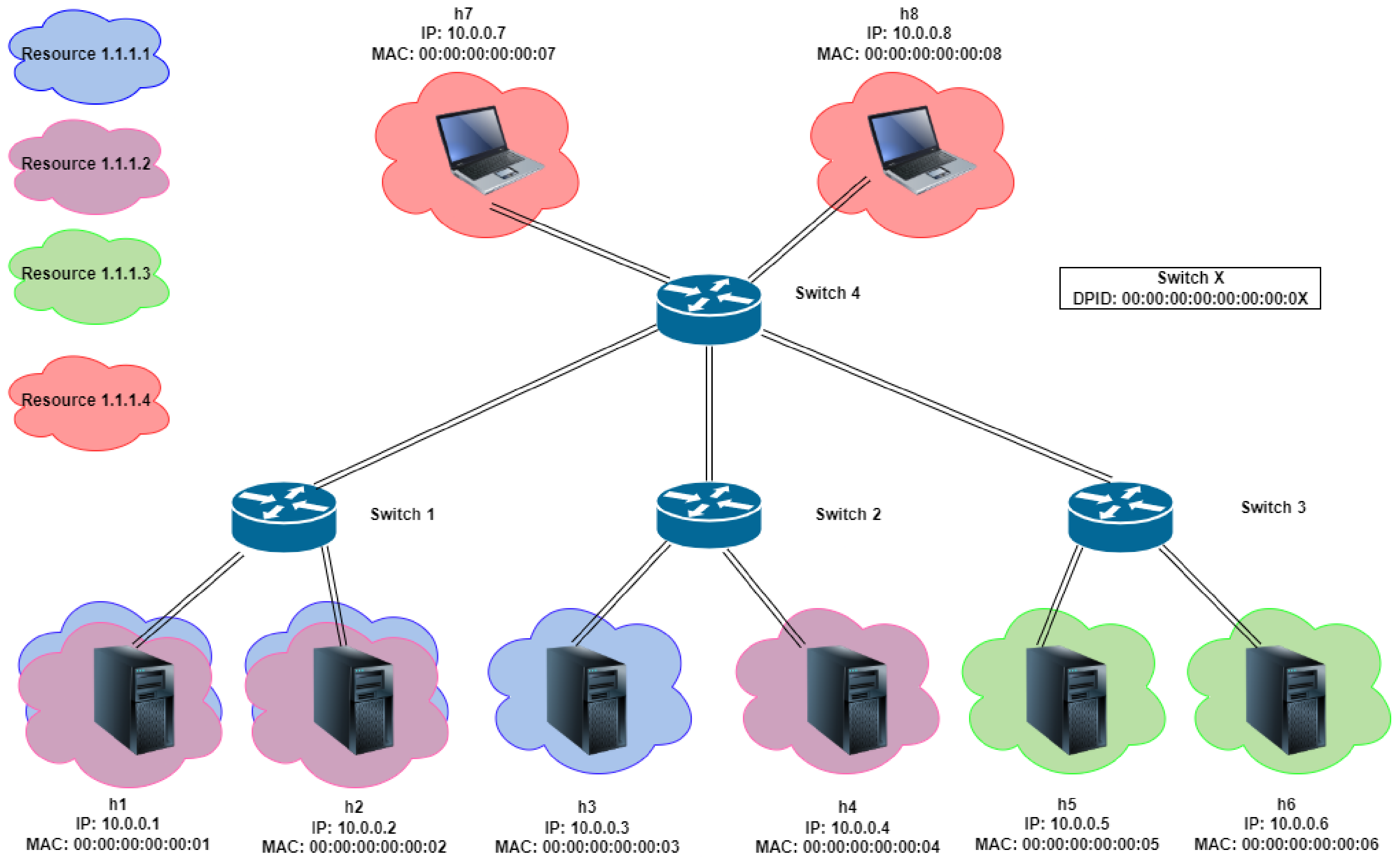
All interfaces



```
No.    Time           Source     Destination  Protocol  Length Info
5463…  2058.4314177…  10.0.0.1   1.1.1.2      UDP           57 42031 → 5005 Len=13
5463…  2058.4316092…  127.0.0.1  127.0.0.1    OpenFlow     165 Type: OFPT_PACKET_IN
5464…  2058.9110114…  127.0.0.1  127.0.0.1    OpenFlow     195 Type: OFPT_PACKET_OUT
5464…  2058.9111133…  127.0.0.1  127.0.0.1    OpenFlow     195 Type: OFPT_PACKET_OUT
5464…  2058.9111894…  10.0.0.1   10.0.0.4     UDP           57 42031 → 5005 Len=13
5464…  2058.9111913…  10.0.0.1   10.0.0.4     UDP           57 42031 → 5005 Len=13
5464…  2058.9113082…  10.0.0.1   10.0.0.2     UDP           57 42031 → 5005 Len=13
5464…  2058.9114392…  127.0.0.1  127.0.0.1    OpenFlow     165 Type: OFPT_PACKET_IN
5464…  2058.9134508…  127.0.0.1  127.0.0.1    OpenFlow     163 Type: OFPT_PACKET_OUT
5464…  2058.9135862…  10.0.0.1   10.0.0.4     UDP           57 42031 → 5005 Len=13
5464…  2058.9135879…  10.0.0.1   10.0.0.4     UDP           57 42031 → 5005 Len=13
5464…  2058.9137133…  127.0.0.1  127.0.0.1    OpenFlow     165 Type: OFPT_PACKET_IN
5465…  2059.0919025…  127.0.0.1  127.0.0.1    OpenFlow     163 Type: OFPT_PACKET_OUT
5465…  2059.0920154…  10.0.0.1   10.0.0.4     UDP           57 42031 → 5005 Len=13
```



CONFIG OTHER TESTS


Resource 1.1.1.1
Resource 1.1.1.2
Resource 1.1.1.3
Resource 1.1.1.4
h7
IP: 10.0.0.7
MAC: 00:00:00:00:00:07
h8
IP: 10.0.0.8
MAC: 00:00:00:00:00:08
Switch 4
Switch X
DPID: 00:00:00:00:00:00:00:0X
Switch 1
Switch 2
Switch 3
h1
IP: 10.0.0.1
MAC: 00:00:00:00:00:01
h2
IP: 10.0.0.2
MAC: 00:00:00:00:00:02
h3
IP: 10.0.0.3
MAC: 00:00:00:00:00:03
h4
IP: 10.0.0.4
MAC: 00:00:00:00:00:04
h5
IP: 10.0.0.5
MAC: 00:00:00:00:00:05
h6
IP: 10.0.0.6
MAC: 00:00:00:00:00:06