batman-adv: distributed handling of non-mesh clients in a layer 2 mesh

Marek Lindner & Antonio Quartulli

B.A.T.M.A.N.-Advanced
www.open-mesh.org

March 29th, 2012
WirelessBattleMeshv5 - Athens

Overview

- How the clients get into the mesh
- Locating non-mesh clients - old mechanism
- Locating non-mesh clients - improving the mechanism

Where the mesh begins

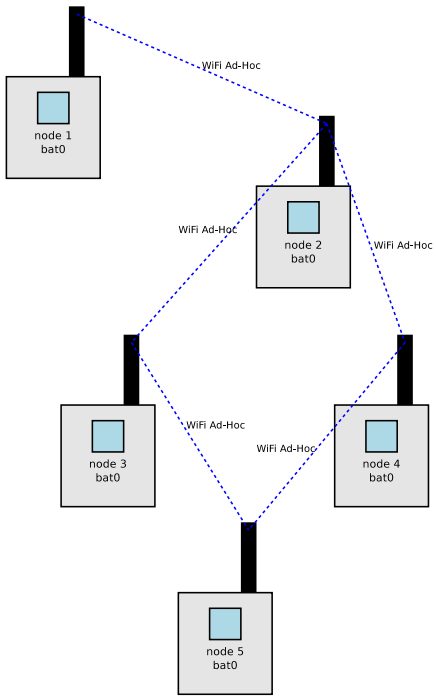
Mesh participants:

- mesh nodes
- non-mesh clients

Where the mesh begins

Mesh participants:

- mesh nodes
- non-mesh clients

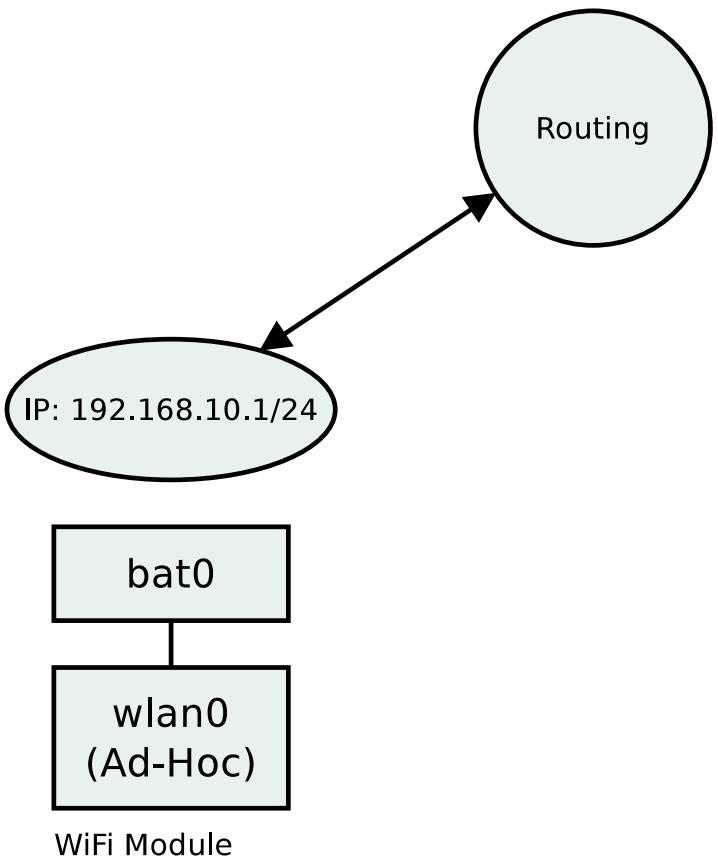


Single interface setup

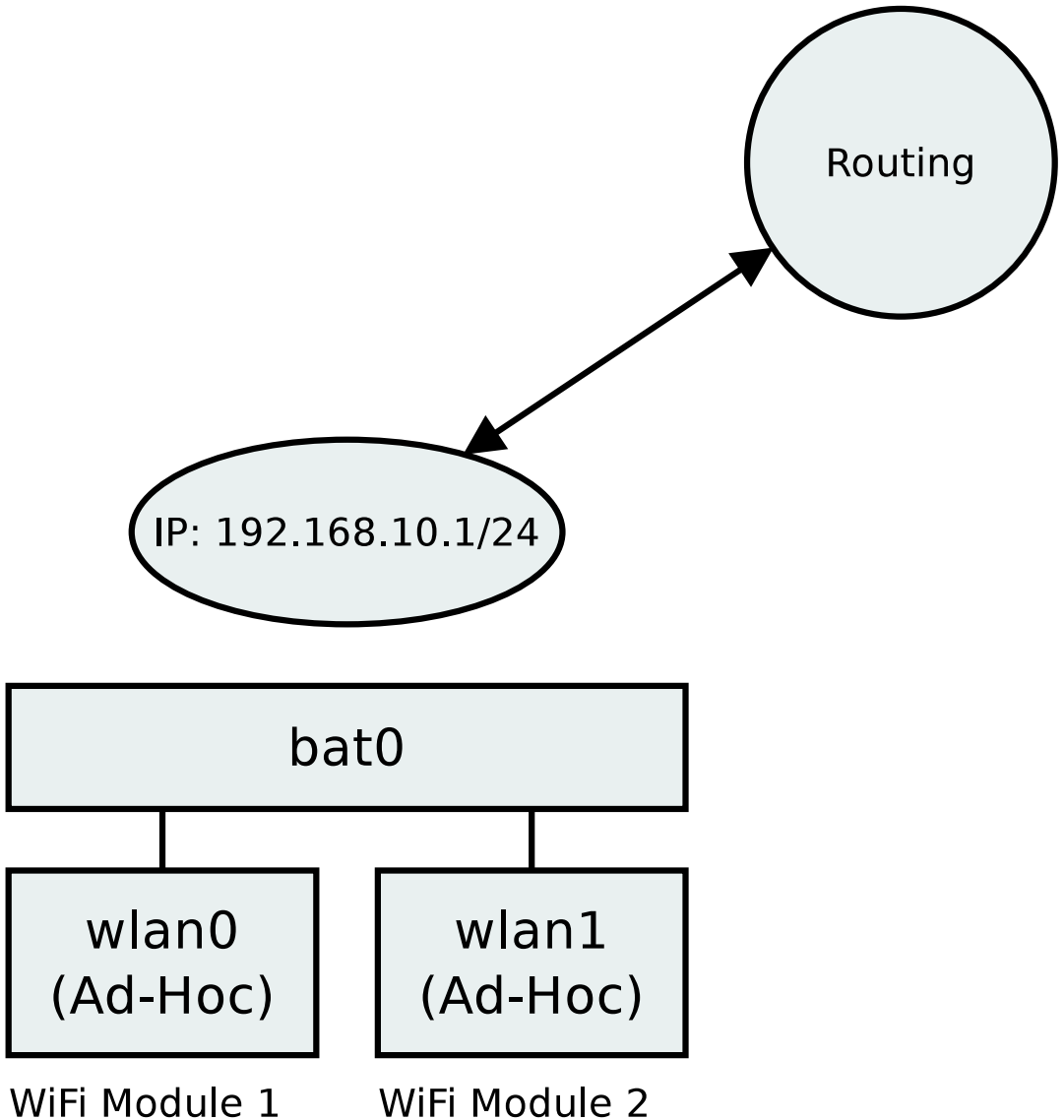
- give control of the interface to B.A.T.M.A.N.-Adv
- B.A.T.M.A.N.-Adv creates “an access to the mesh” interface (bat0)

Single interface setup

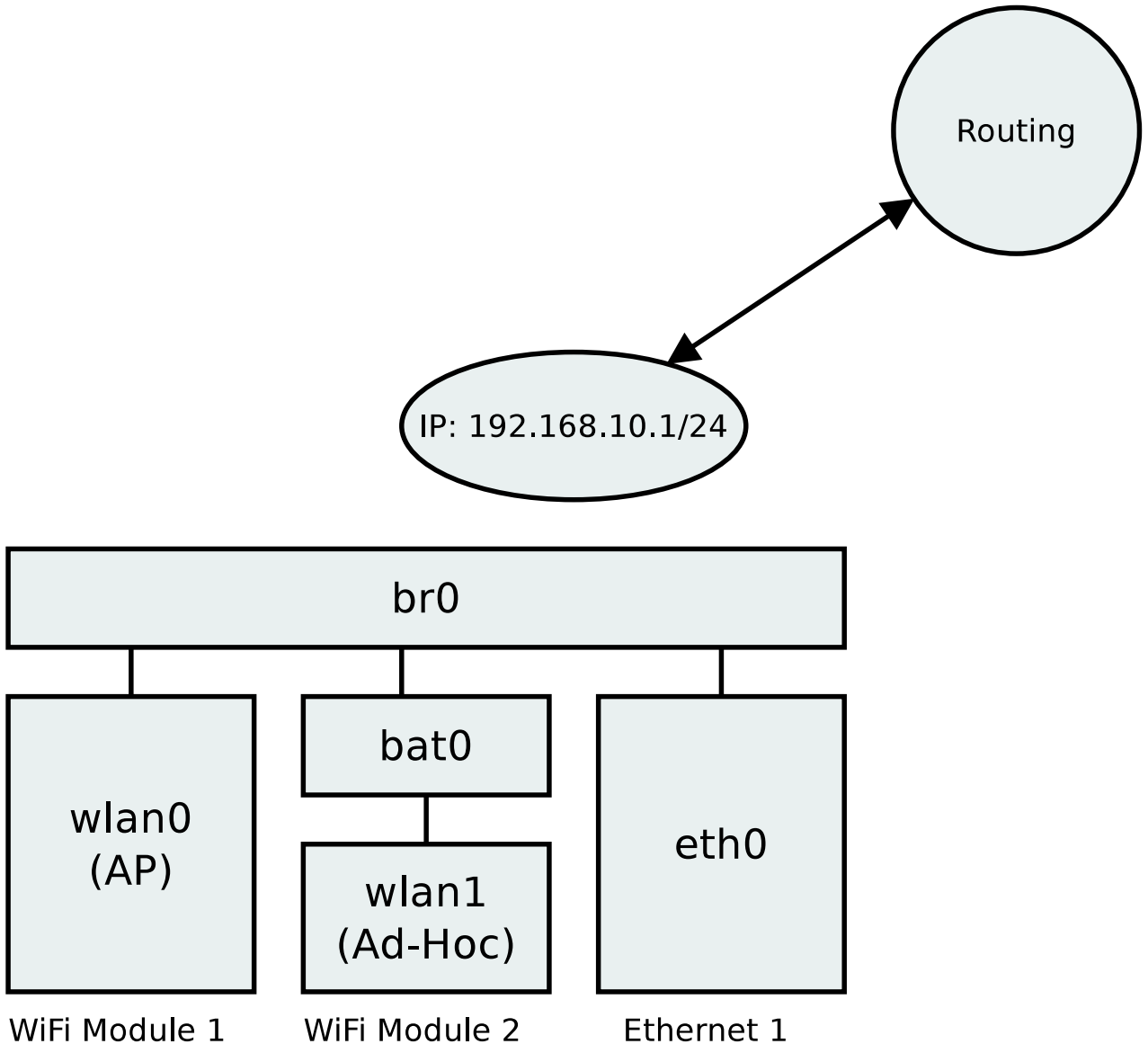
- give control of the interface to B.A.T.M.A.N.-Adv
- B.A.T.M.A.N.-Adv creates “an access to the mesh” interface (bat0)



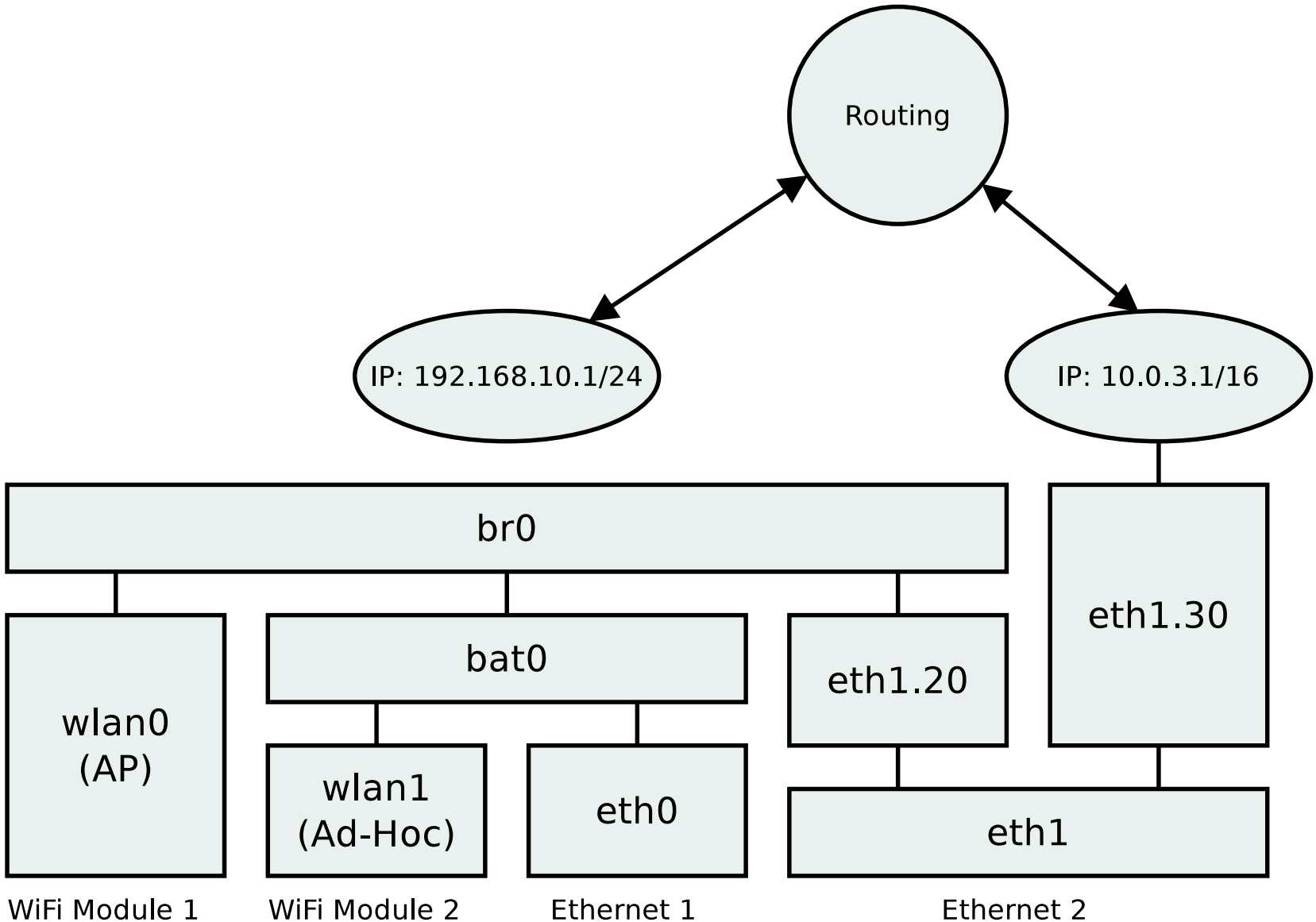
Multi-interface setup



Bridging


Routing
IP: 192.168.10.1/24
br0
wlan0
(AP)
bat0
wlan1
(Ad-Hoc)
eth0
WiFi Module 1
WiFi Module 2
Ethernet 1

Make your live harder



Locating non-mesh clients - old mechanism

Local Translation Table:

- Stores the MAC address of each of the clients "bridged" with the local B.A.T.M.A.N. soft-interface
- A client is added when its first packet goes through the soft-interface

Locating non-mesh clients - old mechanism

Local Translation Table:

- Stores the MAC address of each of the clients "bridged" with the local B.A.T.M.A.N. soft-interface
- A client is added when its first packet goes through the soft-interface

Global Translation Table:

- Stores the addresses of clients announced by other nodes in the network
- For each client it keeps the originator announcing it

Syncing the global table - old mechanism

- Each node attaches its local translation table to its own OGMs
- Nodes receiving the messages extract the table and store it by overwriting the previous one

Pros and cons

Pros:

- very simple
- it works!

Pros and cons

Pros:

- very simple
- it works!

Cons:

- overhead (OGM size)
- roaming works, but leaves room for improvement
- not extensible

New mechanism

- We discussed the topic (non-mesh clients) at WBMv3 (in Rome)
- A couple of weeks later it was decided to start a research project with the University of Trento (Italy) in order to develop a new mechanism
- The project aimed to produce a theoretical basis on top of which we developed a working implementation
- After several review cycles the code has been merged and released

Local translation table versioning

We introduced a new concept: **the versioning**

Each node:

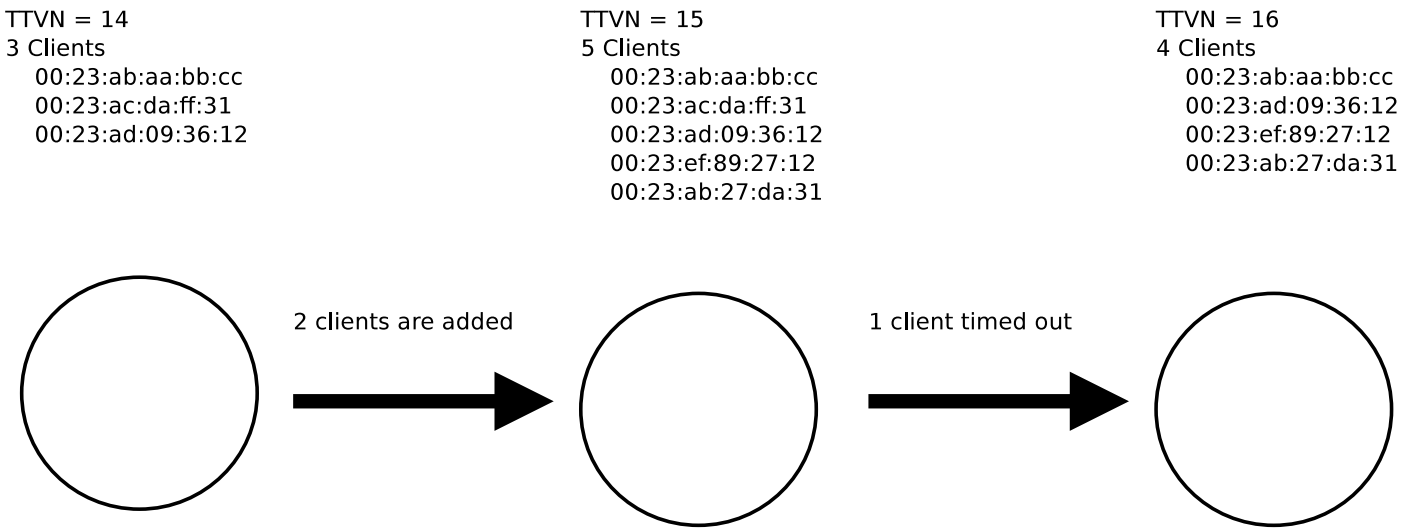
- has a number associated to its local table that identifies its "version": the **Translation Table Version Number (TTVN)**
- increases its TTVN by 1 each time changes happened on its local translation table within the last originator interval
- floods the network with its TTVN (within the OGMs)

Local translation table versioning

We introduced a new concept: **the versioning**

Each node:

- has a number associated to its local table that identifies its "version": the **Translation Table Version Number (TTVN)**
- increases its TTVN by 1 each time changes happened on its local translation table within the last originator interval
- floods the network with its TTVN (within the OGMs)



Local table updates exchange

Given the versioning concept it is now possible to exchange **changes** only instead of the whole local translation table.

Therefore, each time a change (or more) happens, the next OGM will be in charge of carrying **the TTVN increased by one 1** and the set of changes that brought to this new version.

- With this information, nodes can so easily update their Global Translation Table accordingly

Inconsistency detection

Of course, something can go wrong. . .

1. OGMs can get lost
2. wrong updates can get in

The node ends up in a inconsistent state (wrong global table!).

Inconsistency detection

Of course, something can go wrong. . .

1. OGMs can get lost
2. wrong updates can get in

The node ends up in a inconsistent state (wrong global table!).

Inconsistency can be detected by:

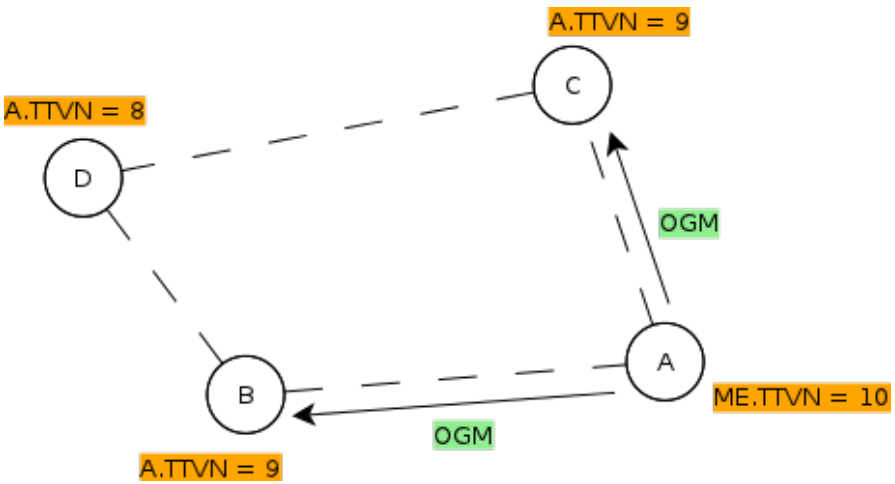
1. receiving an unexpected TTVN within an OGM
2. computing a CRC that doesn't match the one received within the OGM

Table recovery: TT Request / Response

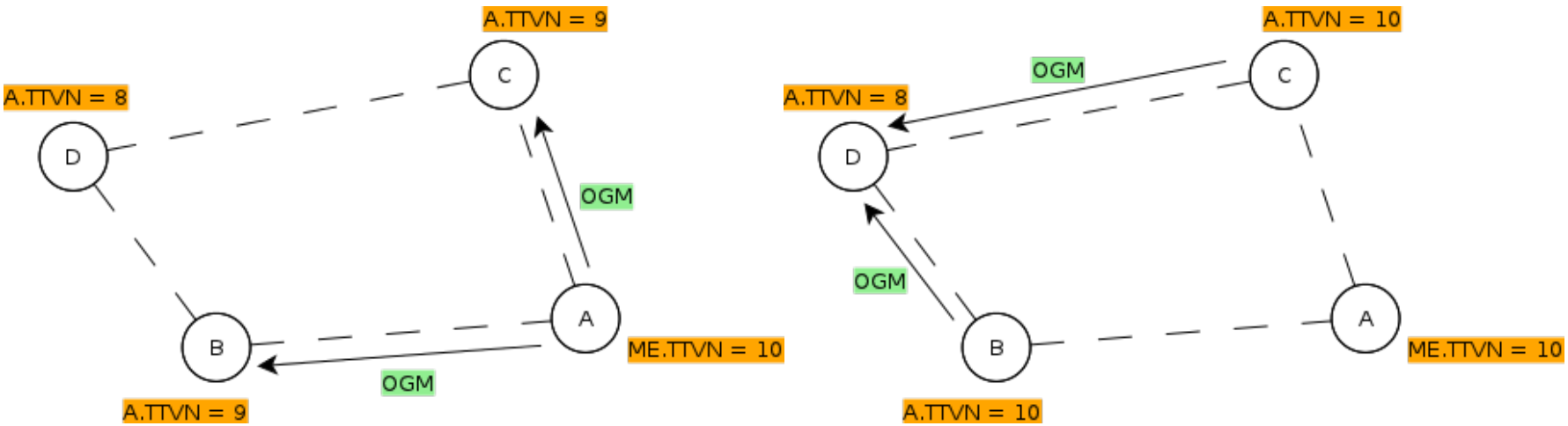
A node can recover its global table state by means of the **TT Request/Response** messages

- It is possible to recover the last changes set only or the full table

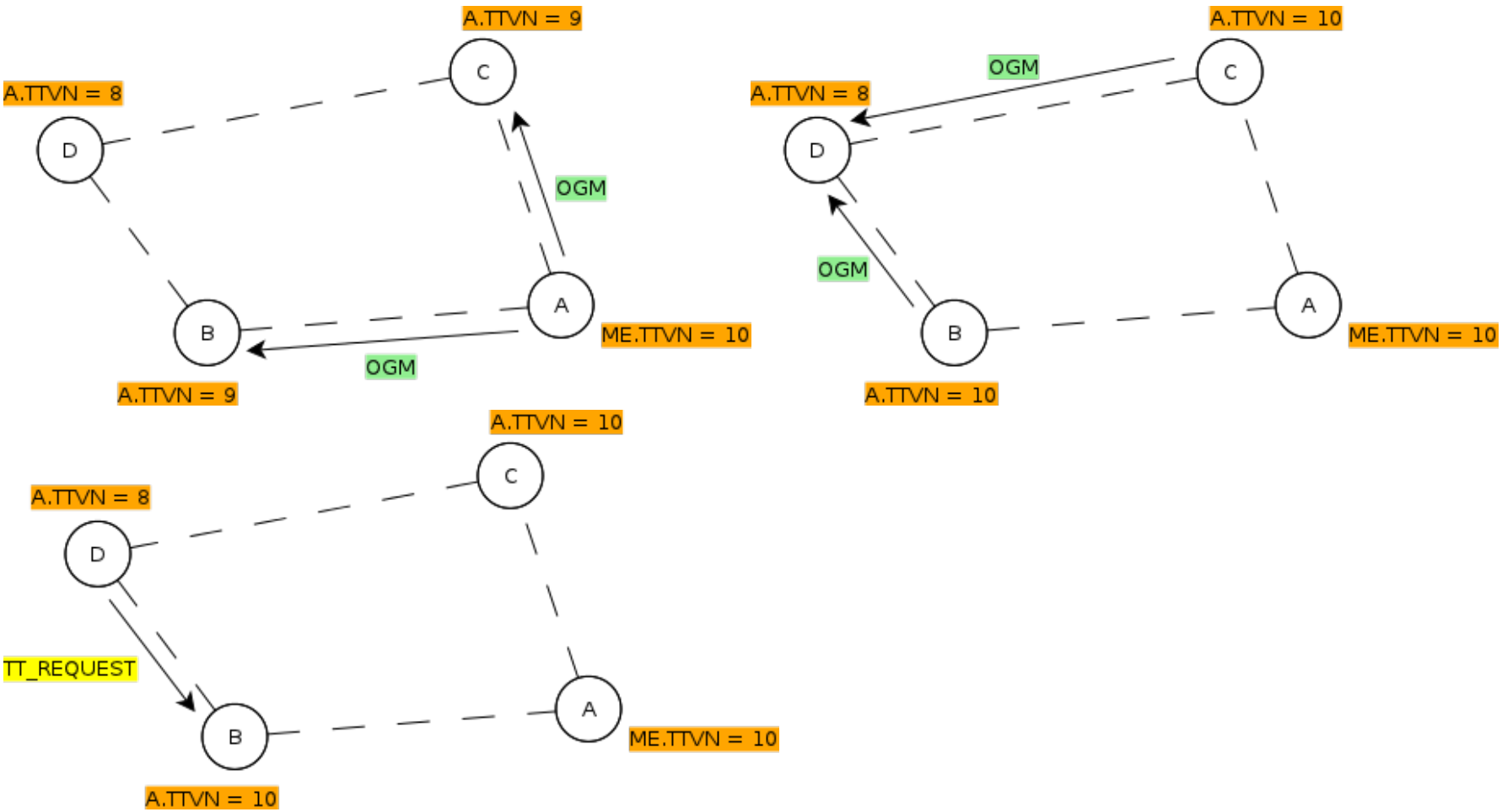
An example



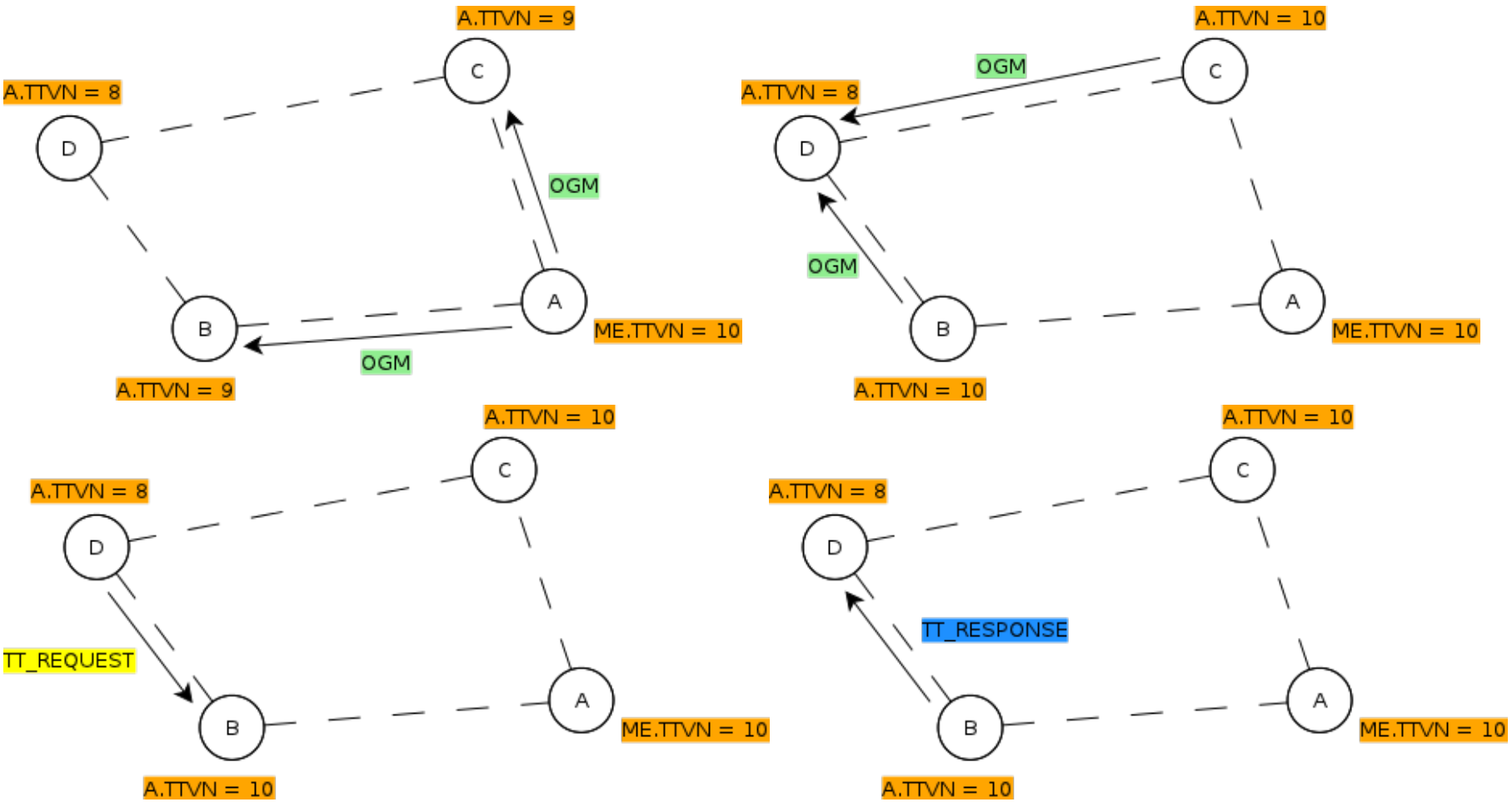
An example


A.TTVN = 9
C
A.TTVN = 8
D
OGM
A
B
ME.TTVN = 10
OGM
A.TTVN = 9
A.TTVN = 10
OGM
C
A.TTVN = 8
D
OGM
A
B
ME.TTVN = 10
A.TTVN = 10

An example


A.TTVN = 9
C
A.TTVN = 8
D
OGM
A
B
ME.TTVN = 10
OGM
A.TTVN = 9
A.TTVN = 10
C
OGM
A.TTVN = 8
D
OGM
A
B
ME.TTVN = 10
A.TTVN = 10
A.TTVN = 10
C
A.TTVN = 8
D
TT_REQUEST
A
B
ME.TTVN = 10
A.TTVN = 10

An example


A.TTVN = 9
C
A.TTVN = 8
D
OGM
A
B
ME.TTVN = 10
OGM
A.TTVN = 9
A.TTVN = 10
C
OGM
A.TTVN = 8
D
OGM
A
B
ME.TTVN = 10
A.TTVN = 10
A.TTVN = 10
C
A.TTVN = 8
D
TT_REQUEST
A
B
ME.TTVN = 10
A.TTVN = 10
A.TTVN = 10
C
A.TTVN = 8
D
TT_RESPONSE
A
B
ME.TTVN = 10
A.TTVN = 10

Benefits

- Reduced overhead
  - OGM are smaller
  - Local recovery mode
  - Updating the network only when needed
  - Support for very large LANs
- Extensible: we have a new framework on top of which we can build new (cool!) features

New features

Something we have already developed:

- AP-Isolation
- Seamless Roaming

Scenario: batman-adv is a layer2 routing protocol. Our use case is a mesh network where each node acts either as node and as bridge to allow mesh un-aware clients to get connected

AP-Isolation

Objective:

- Distributed AP-Isolation feature
- If enabled, different WiFi clients, connected to different mesh nodes should not be able to communicate

How:

- Thanks to the new mechanism, batman-adv nodes can exchange meta information about the clients

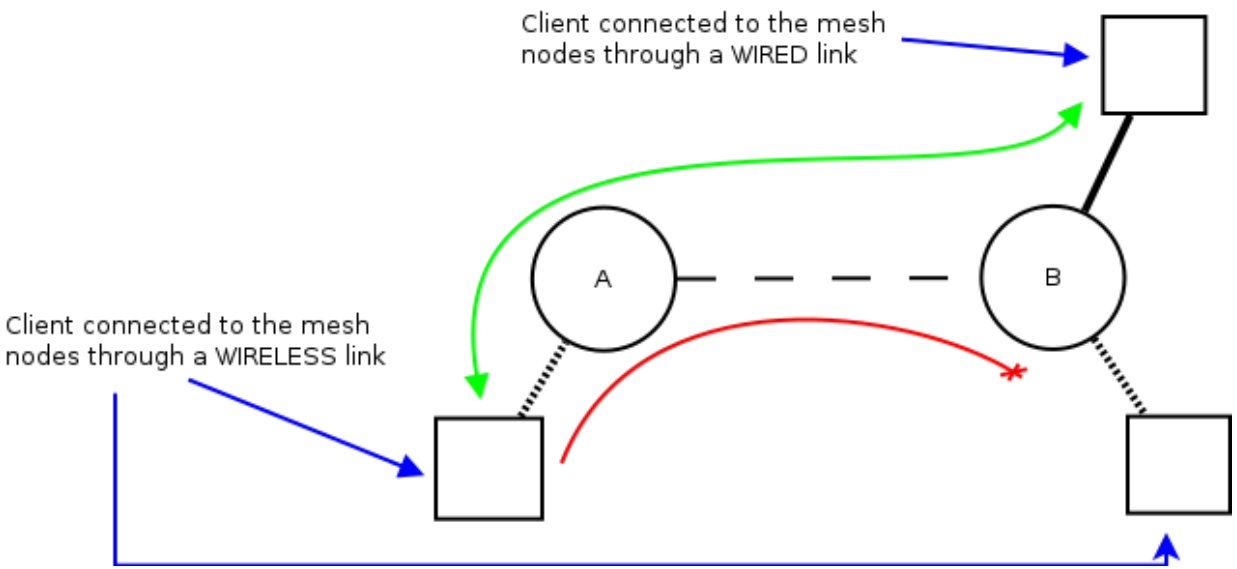
AP-Isolation

Objective:

- Distributed AP-Isolation feature
- If enabled, different WiFi clients, connected to different mesh nodes should not be able to communicate

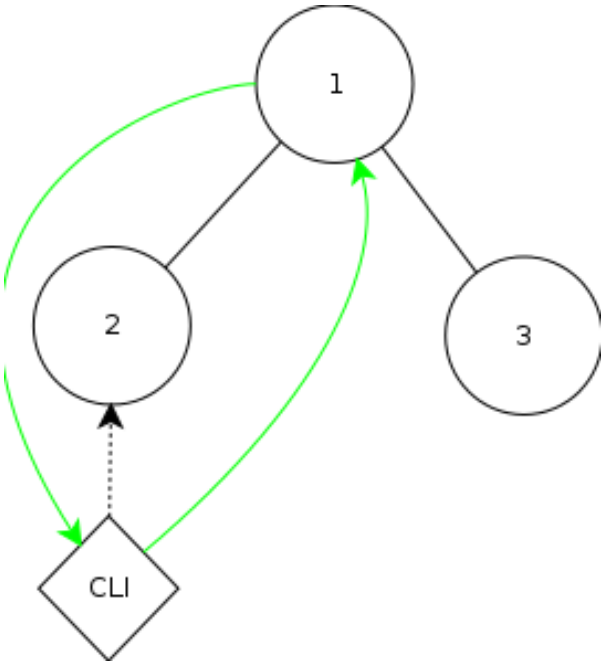
How:

- Thanks to the new mechanism, batman-adv nodes can exchange meta information about the clients



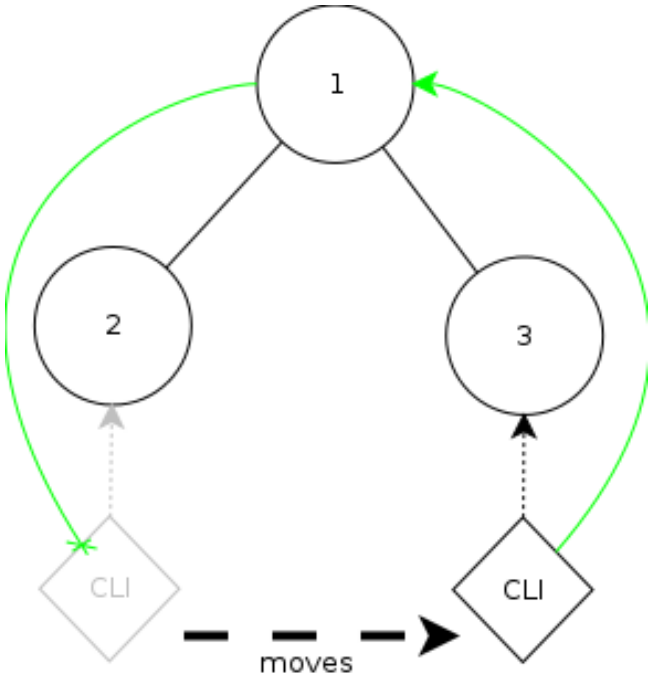
Seamless Roaming

Problem:



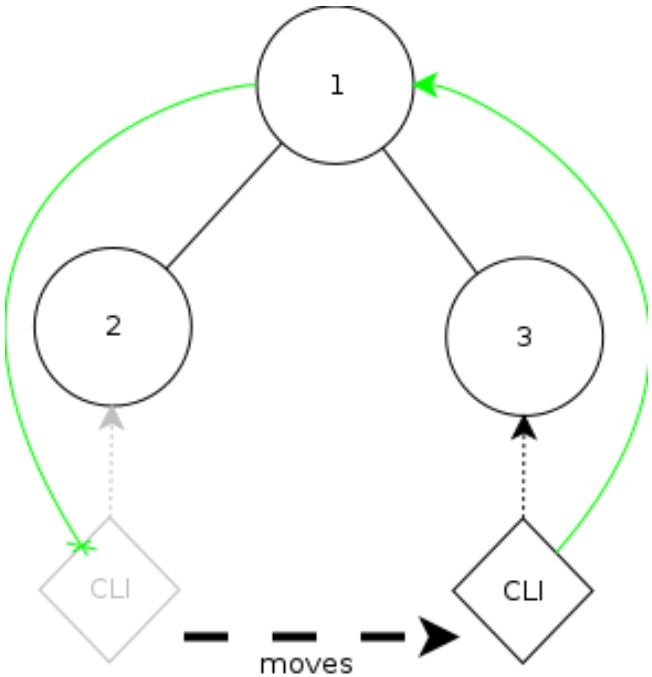
Seamless Roaming

Problem:



Seamless Roaming

Problem:



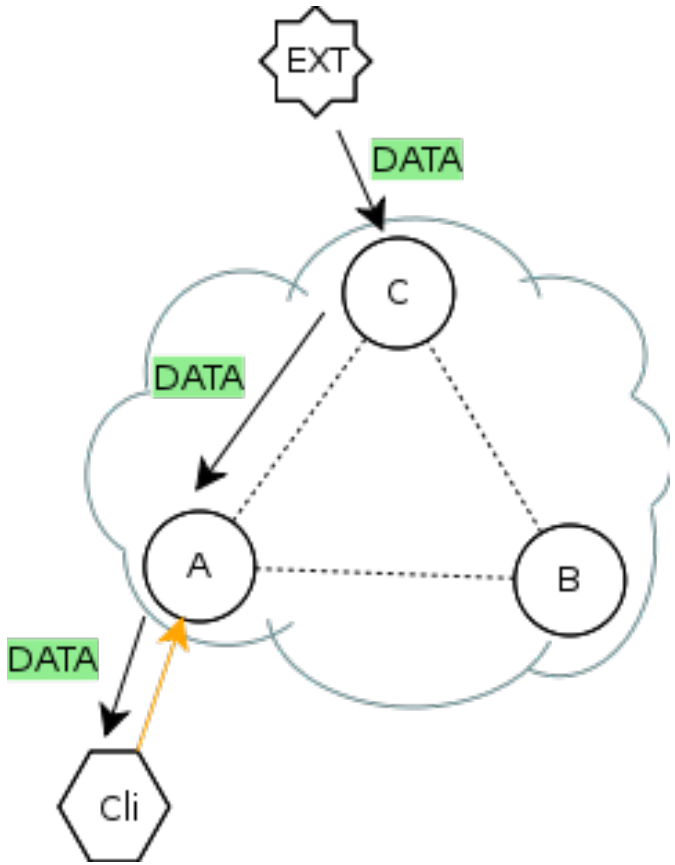
Objective:

- Enable clients to roam and keep connections alive

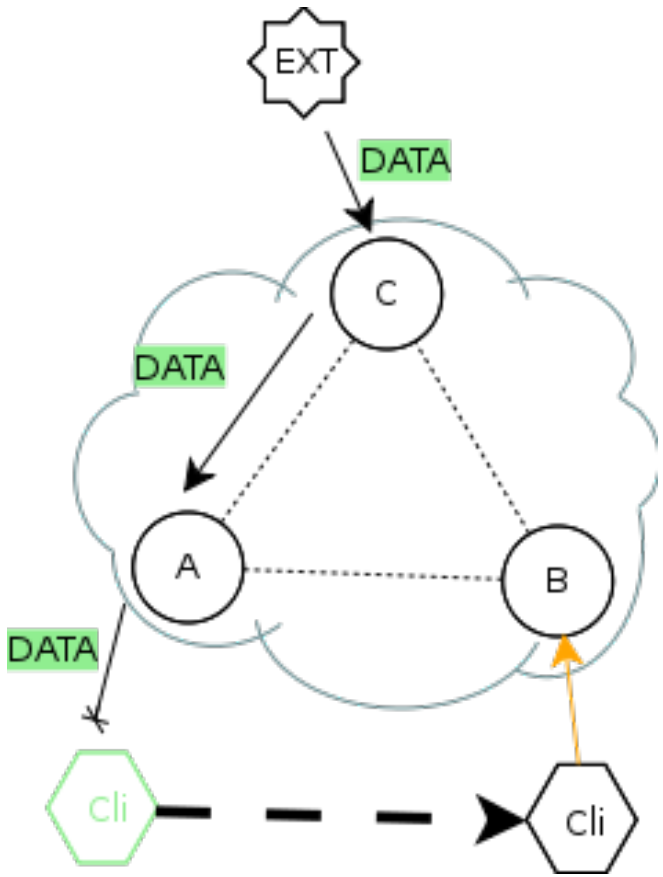
How:

- The new mechanism provides a way to “inject” a single change

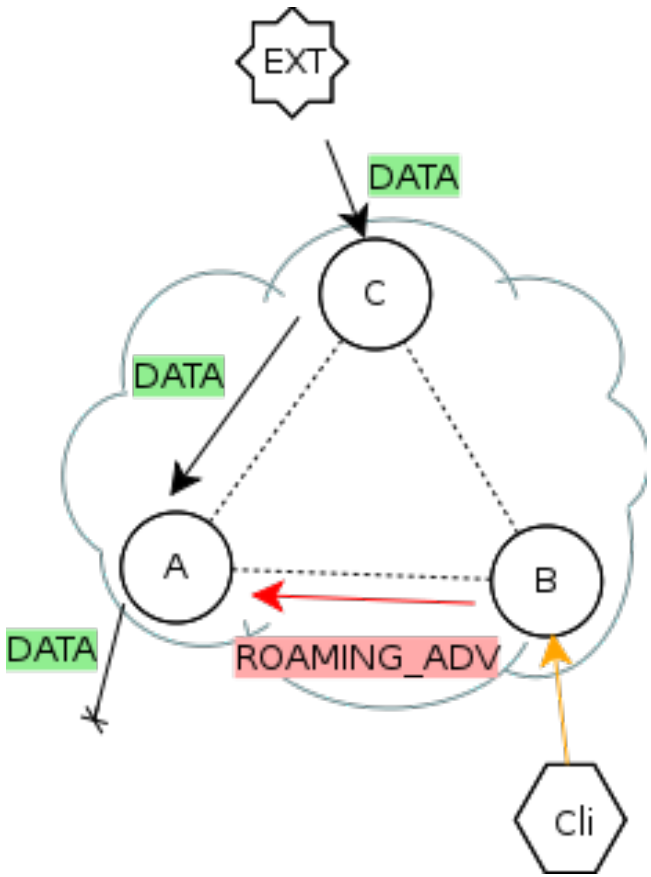
How does it really works? (High level concept)



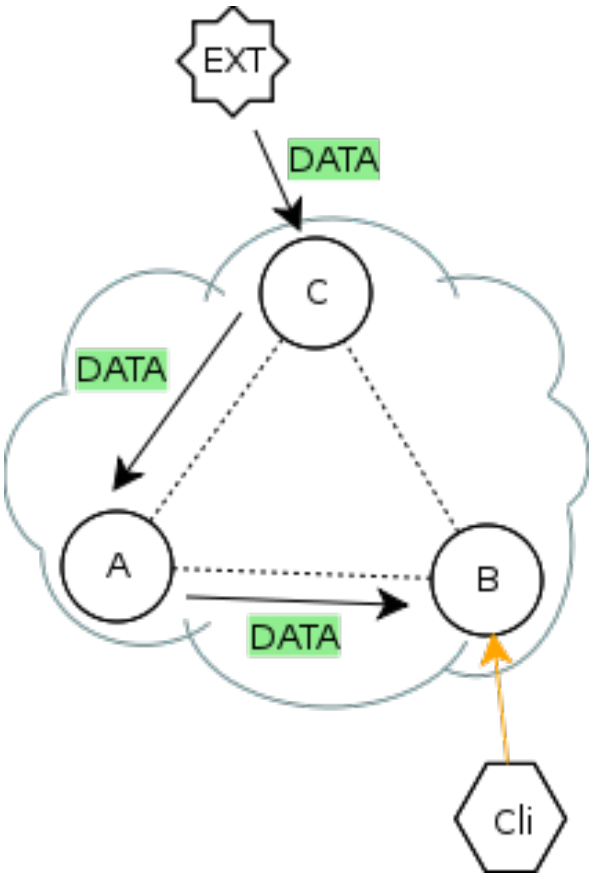
How does it really works? (High level concept)



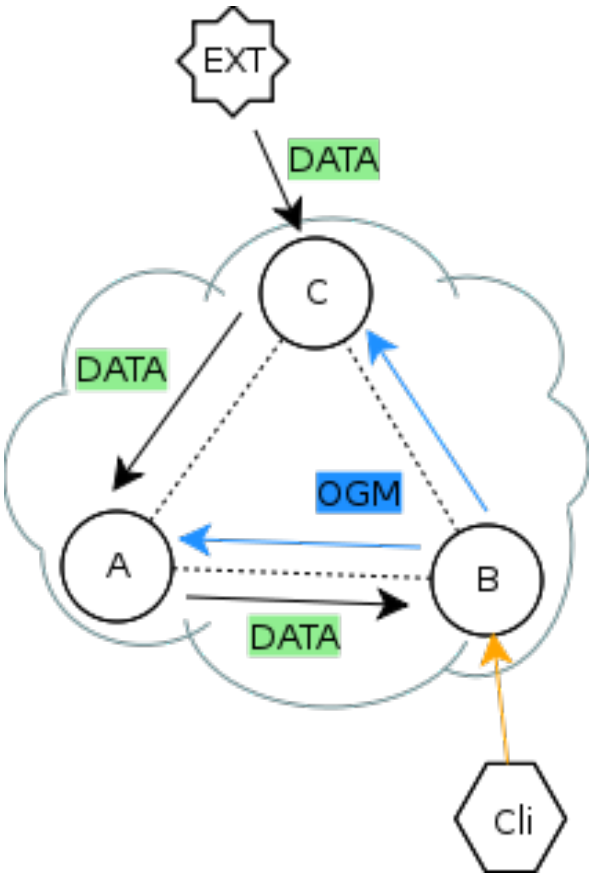
How does it really works? (High level concept)



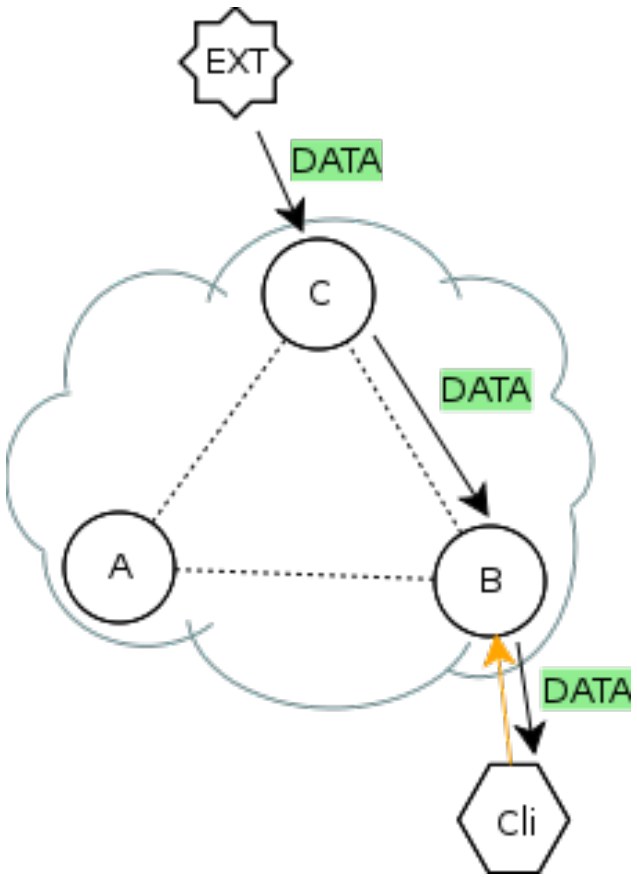
How does it really works? (High level concept)



How does it really works? (High level concept)



How does it really works? (High level concept)

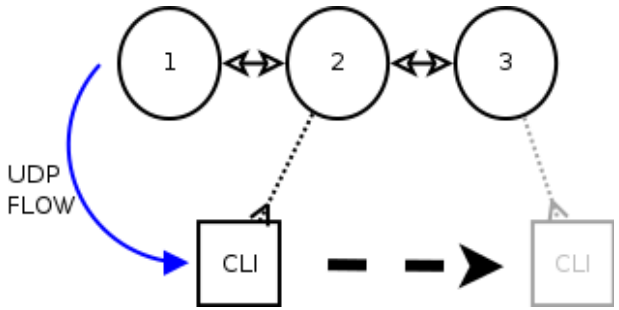


Tests description

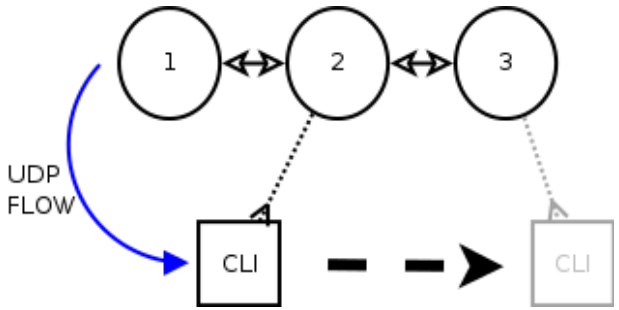
Testbed: made up by a bunch of VMs interconnected using vde_switch/wirefilter.
Topology: We chose to recreate a chain topology in order to point out the problems of the old implementation.
Parameters:

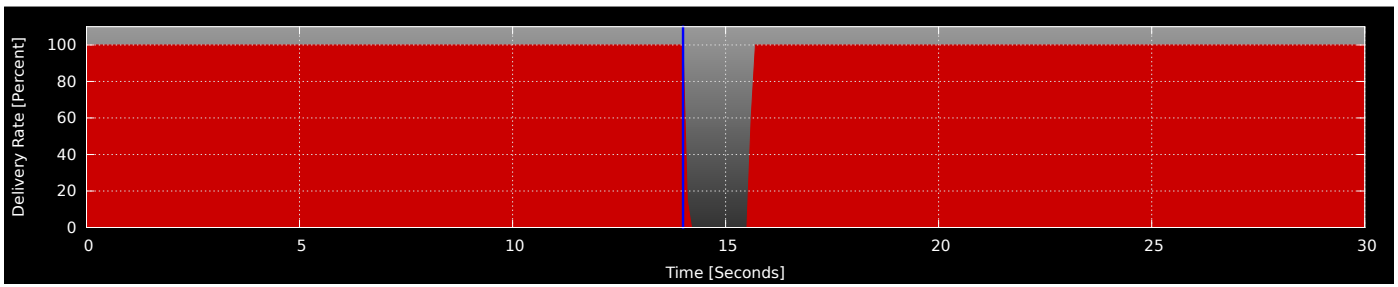
- Originator Interval (amount of time between two different OGM sendings): 1sec and 5secs
- OGM loss probability per link: 0% and 25%
- Number of nodes into the chain (we will see why it is interesting): 3 and 10

Short chain - 0% OGM loss prob - 1s orig.int.

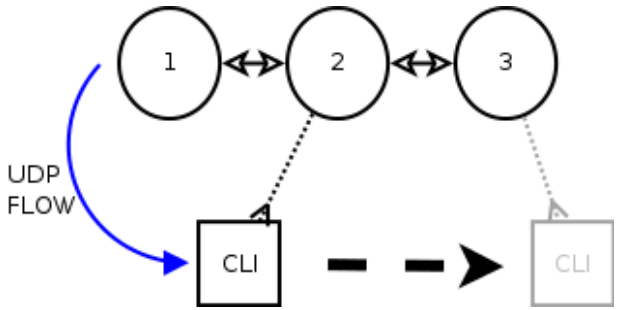


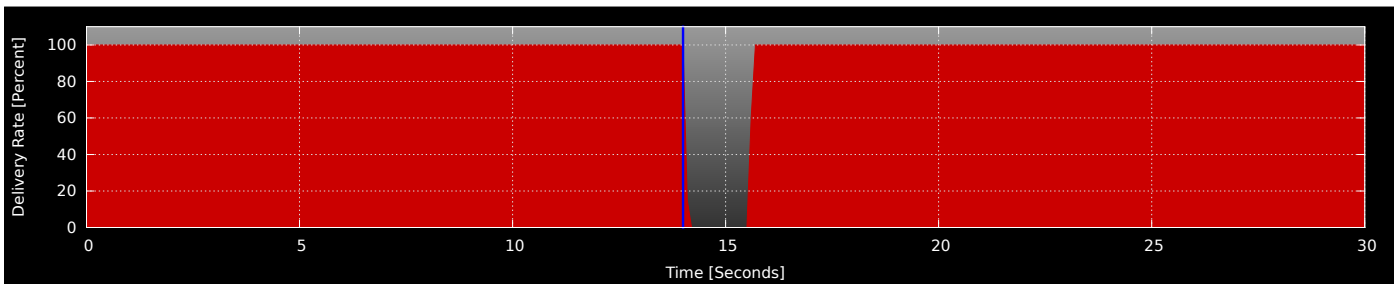
Short chain - 0% OGM loss prob - 1s orig.int.

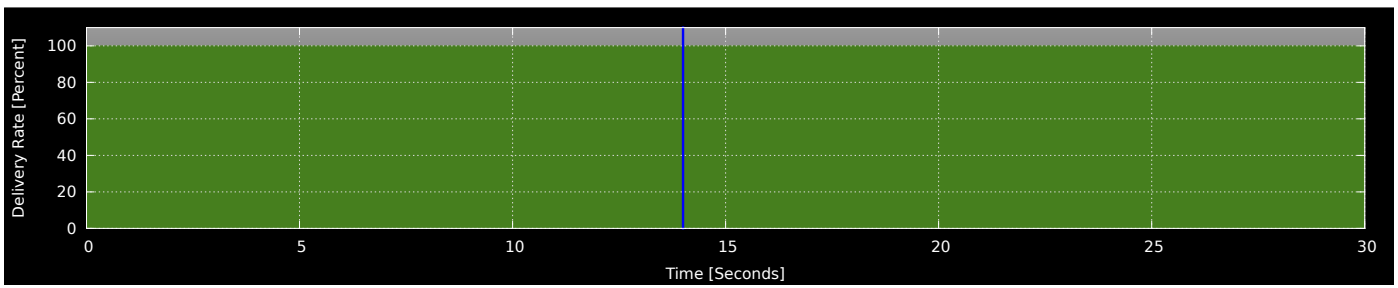




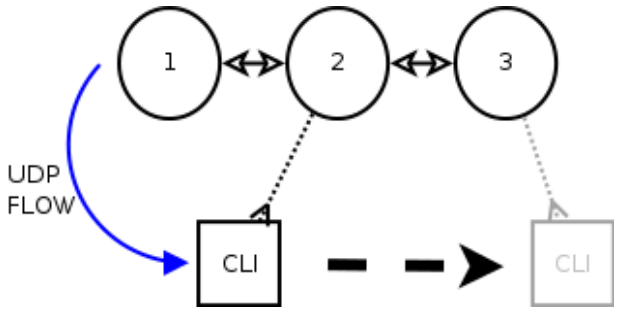
Short chain - 0% OGM loss prob - 1s orig.int.



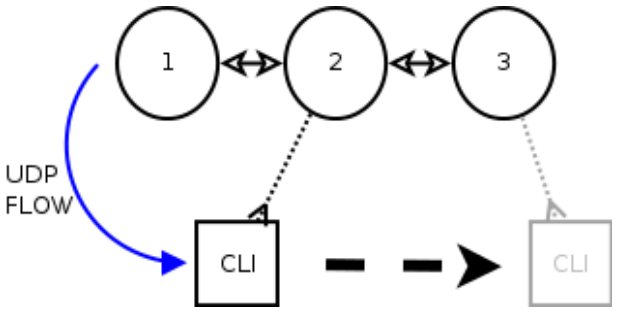


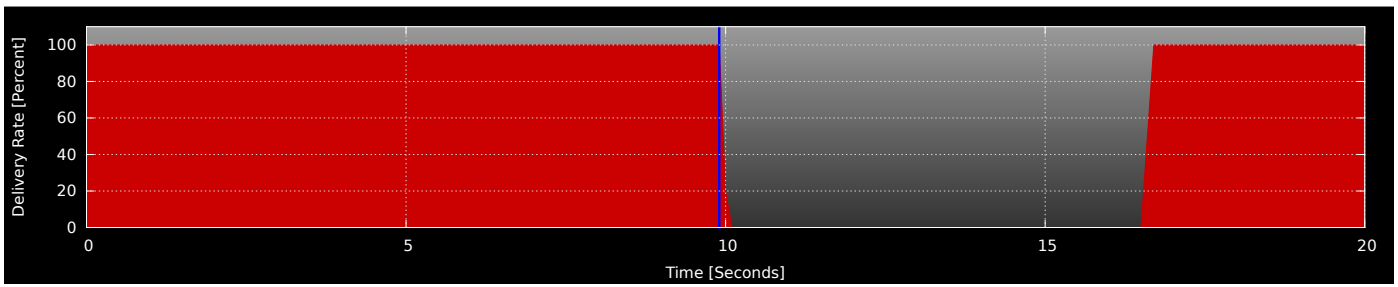


Short chain - 0% OGM loss prob - 5s orig.int.

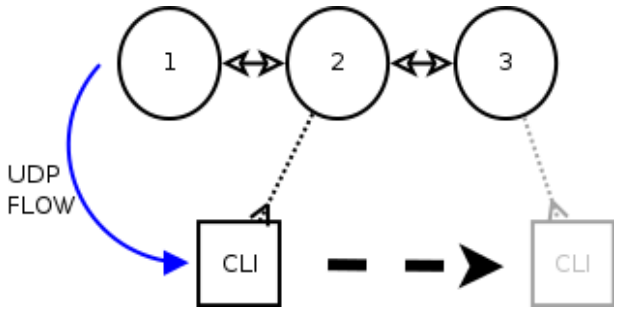


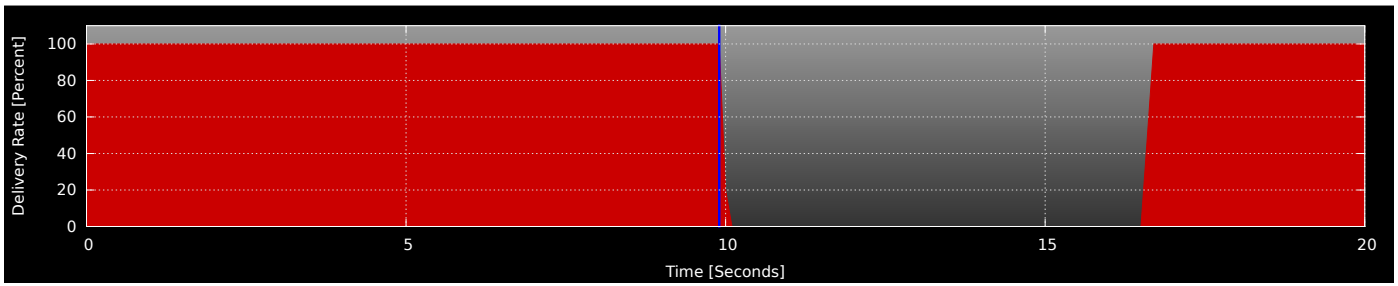
Short chain - 0% OGM loss prob - 5s orig.int.

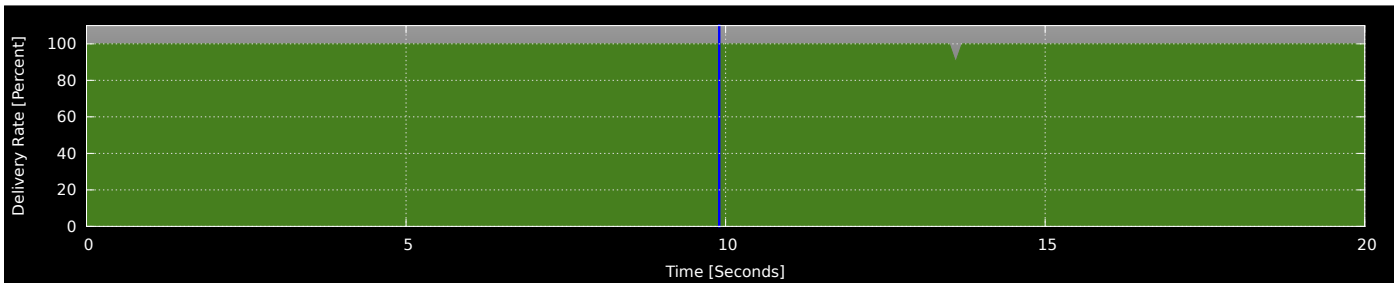




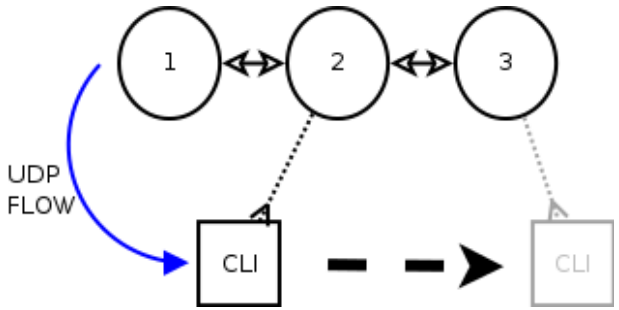
Short chain - 0% OGM loss prob - 5s orig.int.



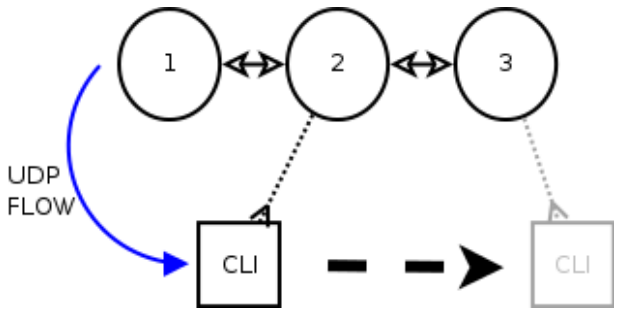


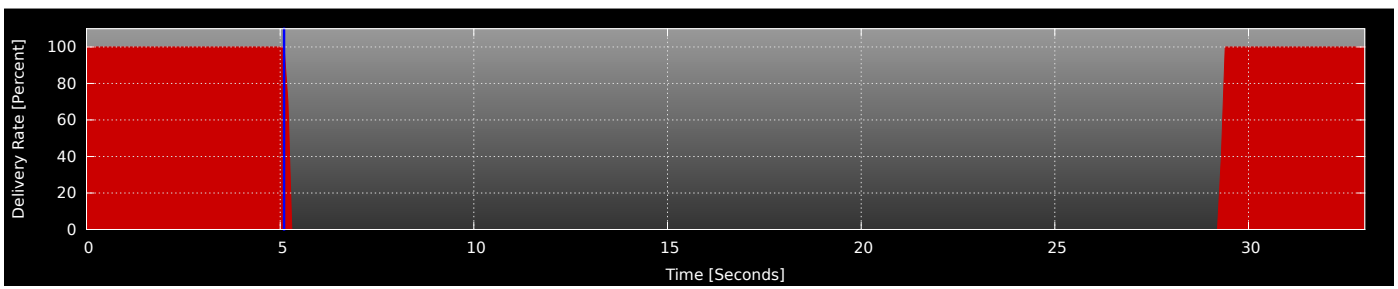


Short chain - 25% OGM loss prob - 5s orig.int.

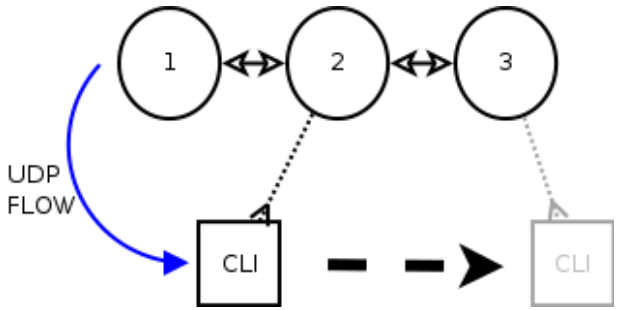


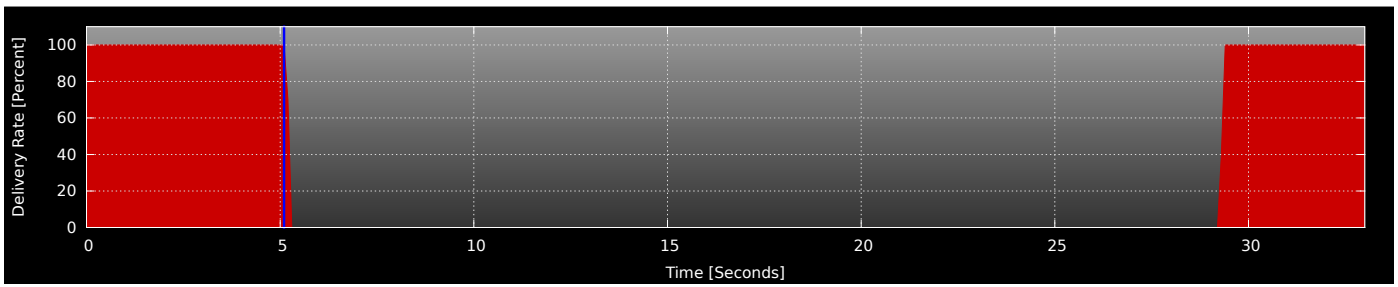
Short chain - 25% OGM loss prob - 5s orig.int.

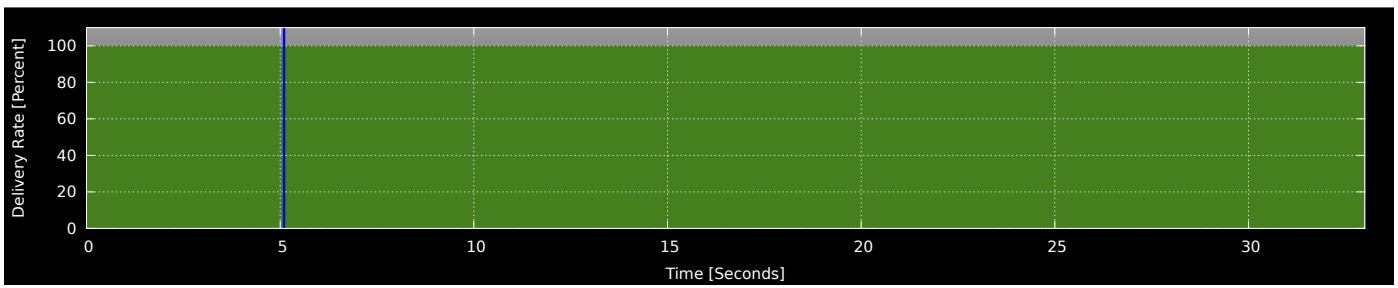




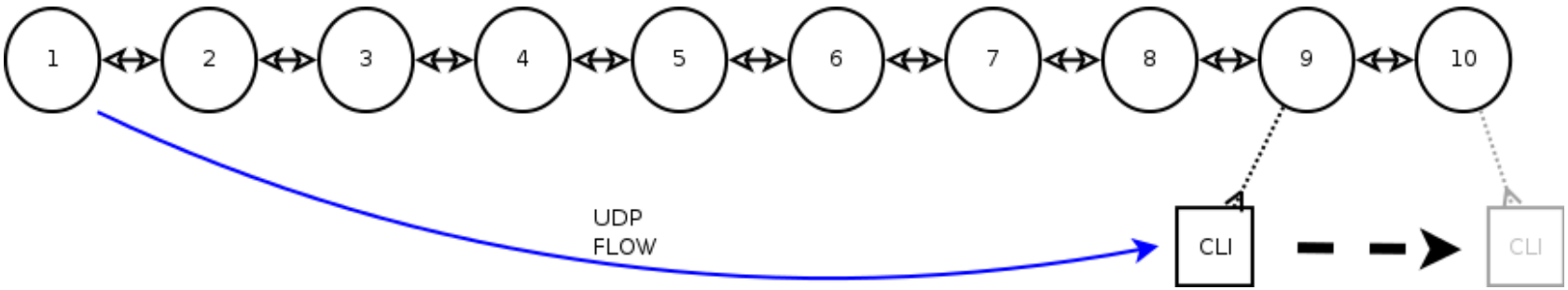
Short chain - 25% OGM loss prob - 5s orig.int.



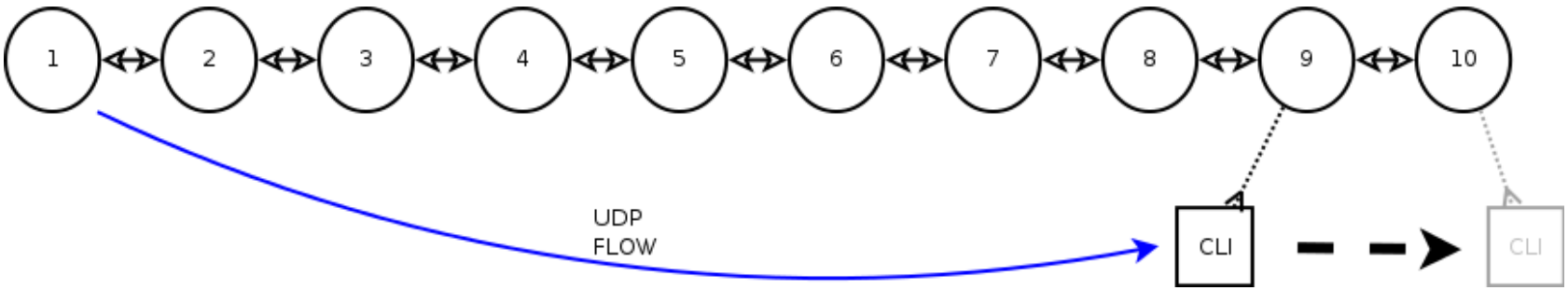


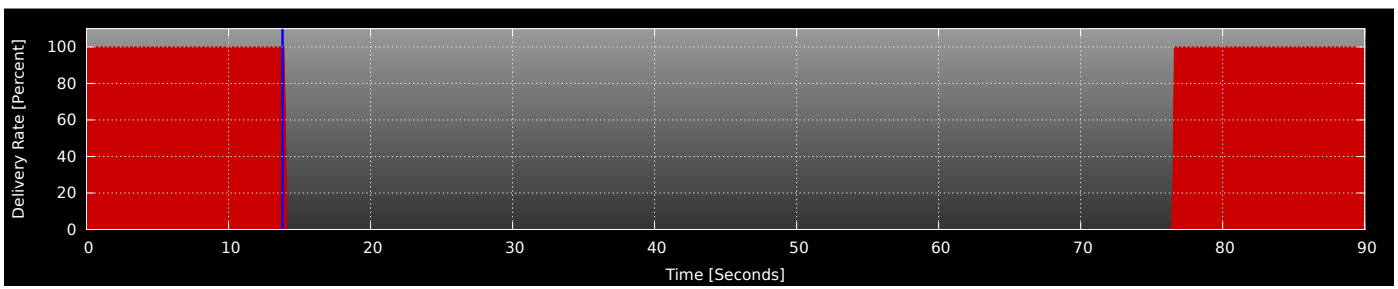


Long chain - 25% OGM loss prob - 5s orig.int.

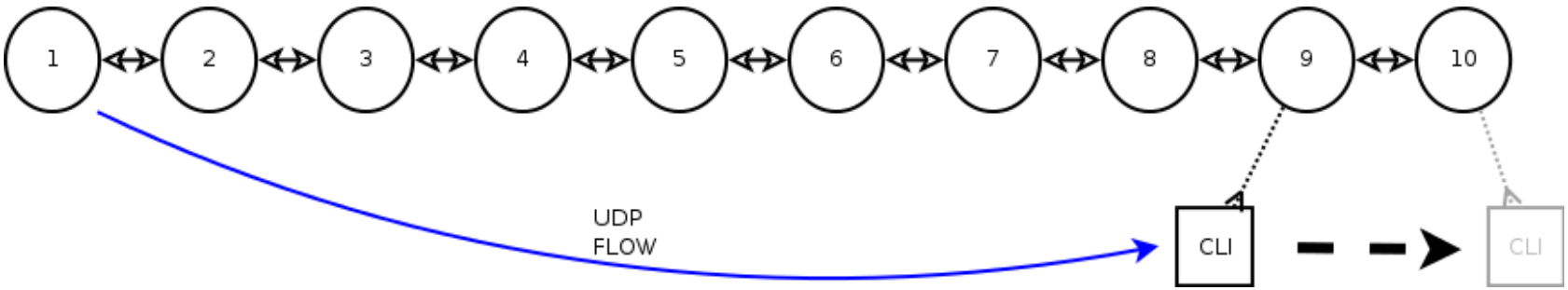


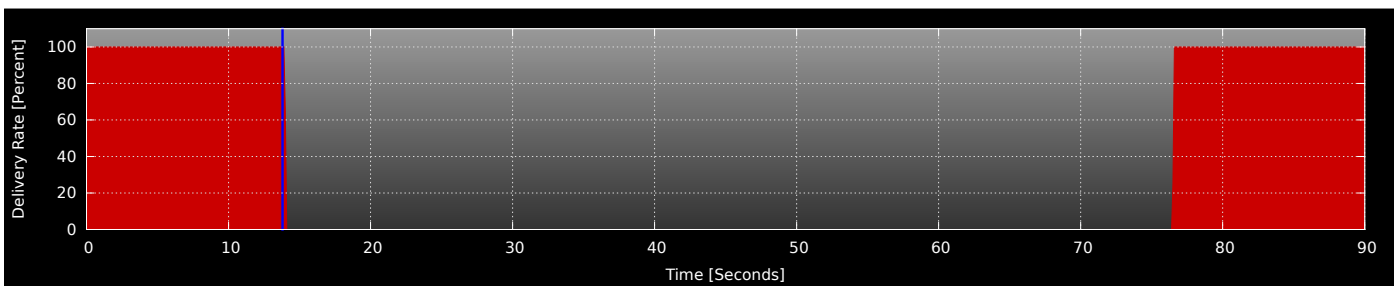
Long chain - 25% OGM loss prob - 5s orig.int.

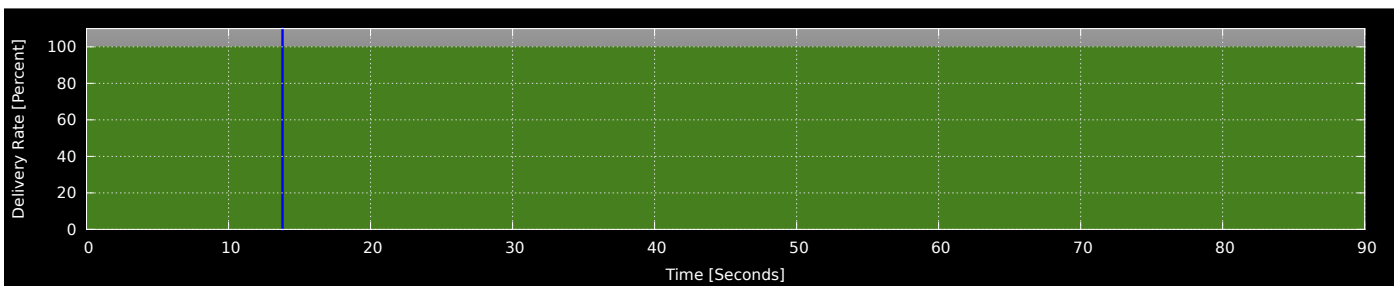




Long chain - 25% OGM loss prob - 5s orig.int.







References

The new client handling code and the roaming mechanism code have been written during a one-year **Research Project** at the University of Trento (Italy)

A technical report has been written out and is freely available online (not really updated):
http://eprints.biblio.unitn.it/2269/1/report.pdf

Wiki:
http://www.open-mesh.org

Thank you for your attention

Question?