A practical experience in designing an OpenFlow controller

Roberto Bifulco, Roberto Canonico
Universita’ degli studi di Napoli “Federico II”
{roberto.bifulco2, roberto.canonico}@unina.it

Marcus Brunner, Peer Hasselmeyer, Faisal Mir
NEC Laboratories Europe
{brunner, faisal.mir, peer.hasselmeyer}@neclab.eu

Presented by: Itzik Malkiel
tcp mobility limitation

• TCP/IP networks were originally designed for communications between fixed devices.
• when a Mobile Node (MN) moves from one link to another without changing its IP address, it cannot receive packets in its new address.
• when MN changes its IP address when it moves, it must terminate and restart any communications.
Migrating Service

**Step 1**
- Access NW 1
- Access NW 2
- "Internet"
- Data Center A
- Data Center B

**Step 2**
- Access NW 1
- Access NW 2
- "Internet"
- Data Center A
- Data Center B

**Step 3**
- Access NW 1
- Access NW 2
- "Internet"
- Data Center A
- Data Center B
FOLLOW-ME CLOUD

• Overcome TCP/IP architecture mobility limitations
  – ability to migrate network end-points
  – reactively relocate network services depending on users’ locations

• performance for the client-server communication
FMC example scenario

Correspondent Node (CN)
Correspondent Switch (CS)
Home Switch (HS)
Foreign Switch (FS)
Migrating Node (MN)

identifier /locator
FMC distributed architecture

Correspondent Controller

Foreign Controller

HH

MN moved

identifier / locator

MN is moving

locator / identifier

Id

CN

MN
How To Make FMC Usable

• FMC must scale with the number of users and migrations
• must be easily deployable in traditional networks
CONTROLLER DESIGN

• Data model to describe the network and its state
• Control logic programming model to interact with such data model

• Distributed
  – different parts of the controller should be able to be moved to different computing nodes

• Extensible
  – providing the ability to combine different network functions

• Performance
• Scalability
Data model

• Network contains a globally unique identifier and a set of OFSwitch objects
• OFSwitch is the base class used to represent and manage an OpenFlow switch
Hierarchical control

Global Level
(e.g., all operator’s networks)

North-bound Interface (E.g., for external systems)

Domain Level
(e.g., single network)

Network

cordinate the local levels to provide the required
network functions (e.g. network addresses mobility)

Local Level
(e.g., single switch)

Learning Switch

Learning Switch

Learning Switch

FMC Switch

handles OFSes directly, providing FTEs and handling
network events
Hierarchical control

Global Level
(e.g., all operator’s networks)

Domain Level
(e.g., single network)

North-bound Interface (E.g., for external systems)
Hierarchical control

- **Global Level**
  - E.g., all operator's networks

- **Domain Level**
  - E.g., single network

- **Local Level**
  - E.g., single switch

Diagram:

- North-bound Interface (E.g., for external systems)
- Learning Switch
- Learning Switch
- FMC Switch
- Network

Network representation:

- A
- B
Hierarchical control
Extensibility

- provided using OO paradigm
- OFSwitch class can be extended to provide new functions
- But... the addition of a FTE can have unexpected effects on the behavior of the switch
- a subclass that inherits from the OFSwitch can use the methods from the superclass
Extensibility using inheritance

- OFSwitch
  - Control logic:
    - Flow table entries handling
    - Packet in processing
    - Dynamic processing state
    - ...

- LearningSwitch
  - Eth Learning Switch Control logic:
    - ARP handling
    - MAC based FTEs
    - MAC-Port table
    - ...

- FMCSwitch
  - FMC Control logic:
    - IP address “migration” function
    - Locator addresses handling
    - Advanced ARP handling

...there is a paradigm mismatch between OO programming and OF programming
Scalability: distribution of networks
Scalability: distribution of networks
DISCUSSION

• Network class implement global control logic
• OFSwitch objects are in charge of handling the local control logic
• all the local events are kept local, requiring no interactions among different Controller nodes
• separating the development of low level FTEs programming from the development of high-level network functions
I should update network C identifier / locator.

Installing FTE...

Network A

Network C

Network B
CONCLUSIONS

• Appling OO modeling for an OpenFlow controller design is good to organize the network data and to structure the application for scalability

• it does not help in providing extensibility, that still requires a direct handling of FTEs
  – the function that is going to be extended must be well known by the programmer
What is missing?

• Security issues
• Migrating service
• Implementation
• Actual results
Questions?