Towards an SDN-based Mobile Core Network

Zoran Despotovic
Future Network Technologies
Huawei Technologies Co. Ltd.

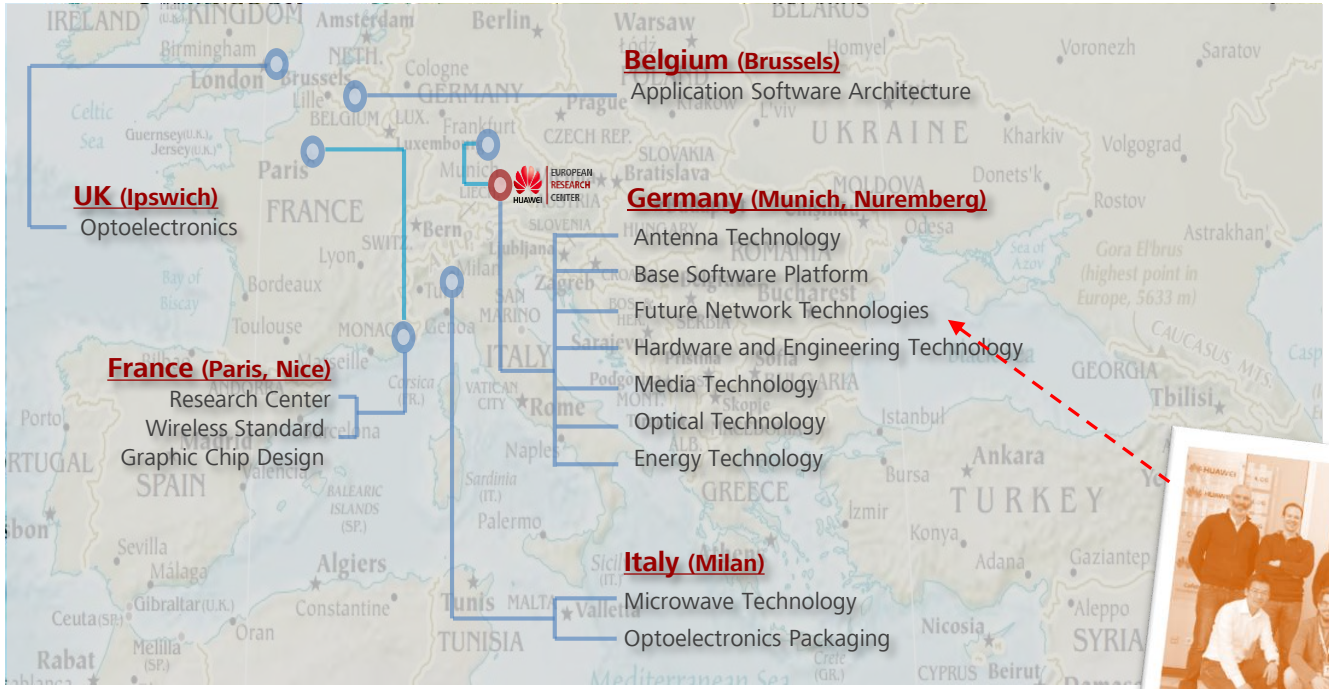
Contributors: Artur Hecker, Xueli An,
Clarissa Marquezan, Ramin Khalili, David Perez

BUILDING A BETTER CONNECTED WORLD

Contents

- About Huawei ERC
- SDN background
- Mobile network architecture background
  - Possible architecture/infrastructure evolutions
- Our research on SDN & Mobile network architecture
  - Our testbed
  - Mobility management
  - Packet_IN context interpretation
- Can we improve the current SDN model
  - Proxy and Switch2Switch communication

Competence Centers in Huawei ERC



Yes, we are hiring

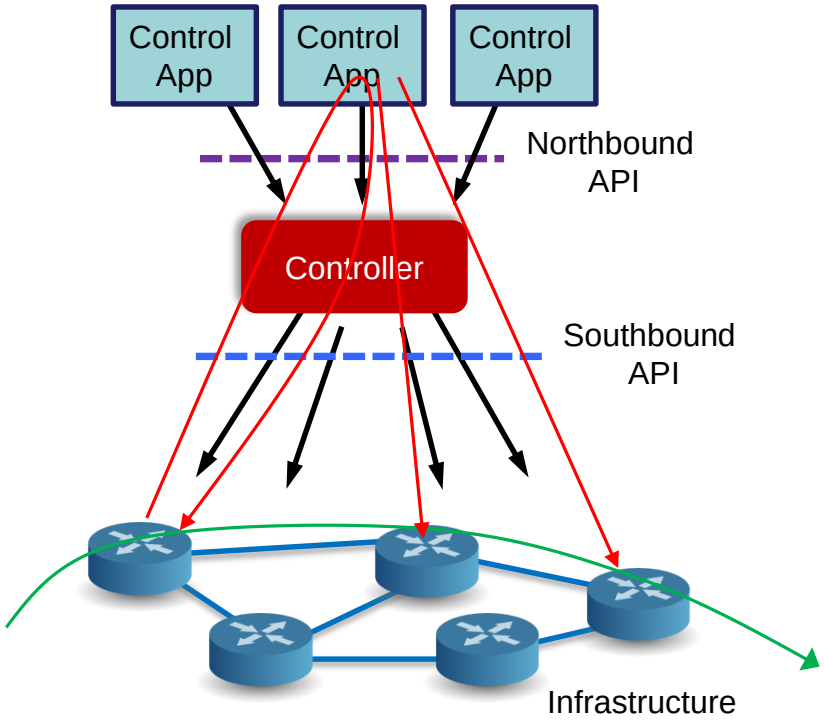
- Great news: jobs for you!

- Internships, master theses, PhD, regular employment
- Profiles:
  - Any **networking** competence is interesting
    - PhD is great *but* industry experience can compensate
  - Need: **hands on** experience and ***can do*** **attitude**

- Check our ‘Careers’ page or send emails to zoran.despotovic@huawei.com

A bit about SDN

SDN model



- Controller (control apps) sets up the flows
- Switches only forward

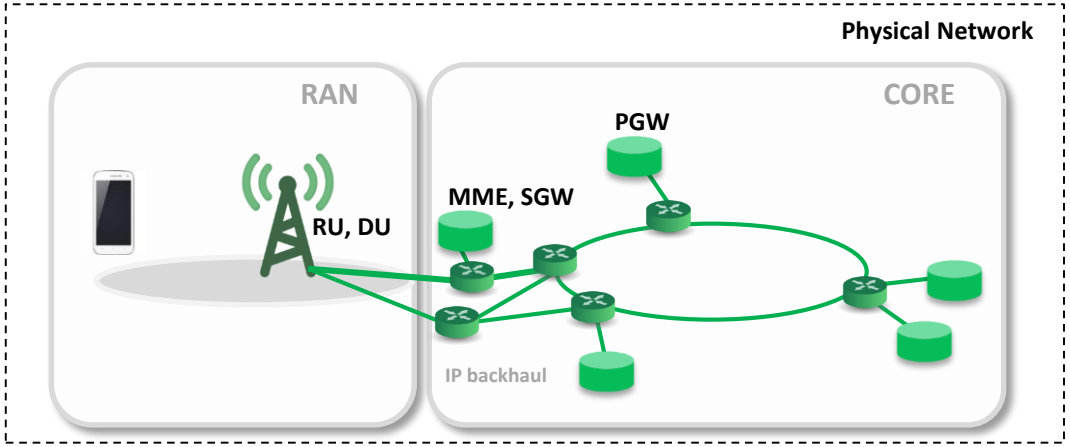
SDN today

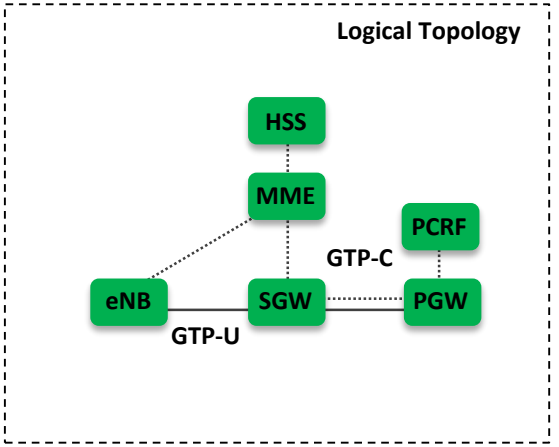
- Mainly driven through ONF's OpenFlow
- So far used as a new enabler technology in *computer networks*
  - LAN/MAN: academic environments, enterprise networks
  - Data Centers (DC) – part of Cloud Technologies
- A very active area of research and development
  - Several open source controller projects
  - Several vendors active
  - Expected outcomes
    - Network programming: network languages & compilers, static analysis
- How to bring that emerging know-how to Carriers?

Current Mobile System (LTE/EPC) and its evolutions

Introduction of SDN

Current Generation (4G): LTE / EPC



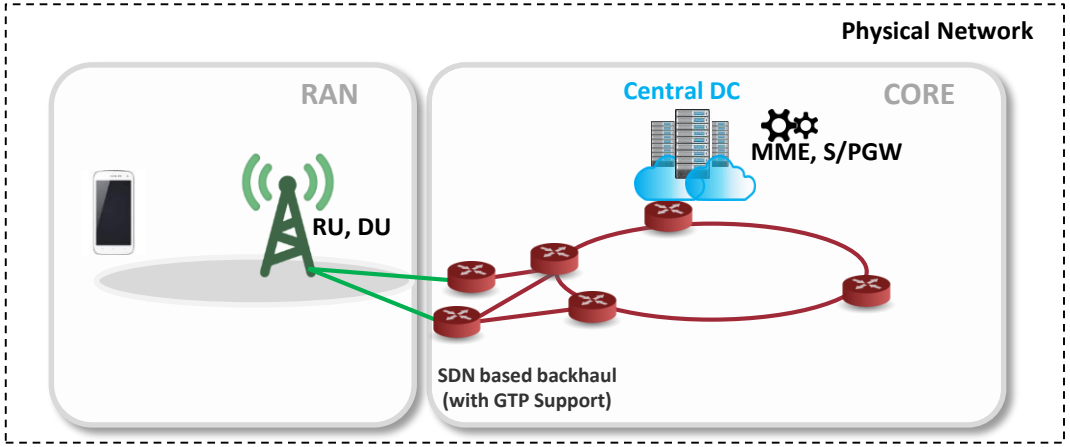


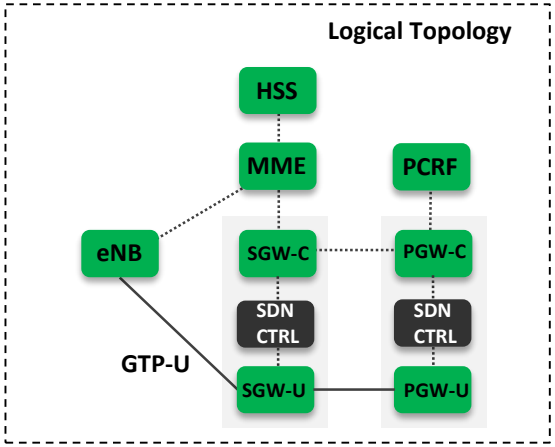
System Architecture:

- Pre-configuration: fixed infrastructure, fixed architecture, manual setup
- Single architecture: one size fits all
- LTE entities becoming bottlenecks (MME in C-plane, PGW in D-plane)

- Non-native transport: overlaying in both D-Plane and C-Plane
- Non-optimized service provisioning: packets are touched by many entities, user states are spread in multiple network elements
- Cost: All dedicated hardware, high CAPEX and high OPEX

Evolution (1) SDN as an Interface to the Infrastructure

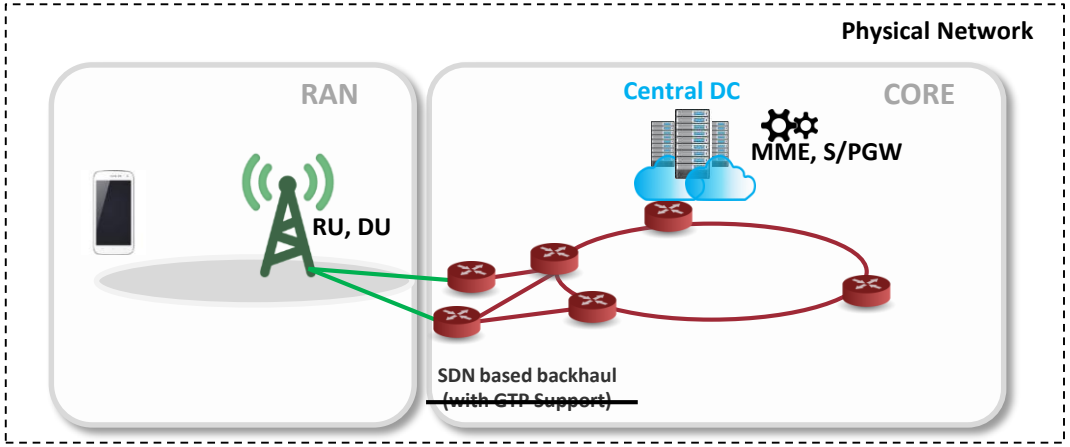


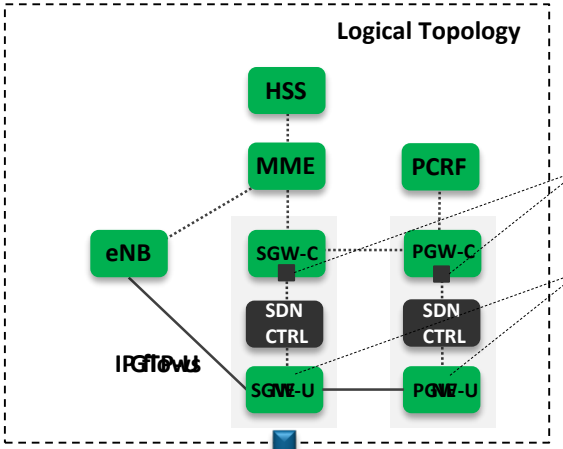


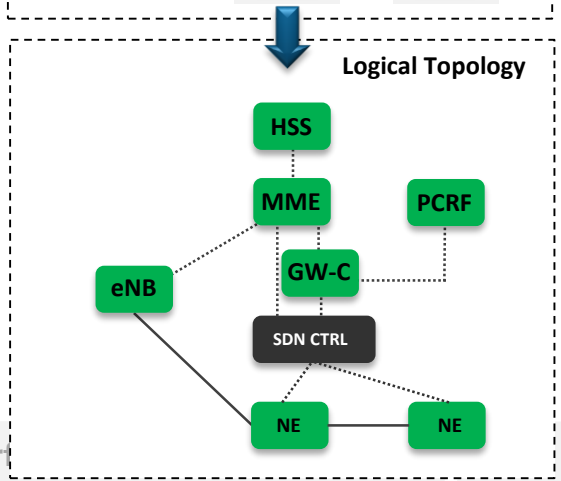
- Virtualized network function (VNF) approach
  - Access function like DU, Core network function like MME, S/PGW are implemented as software and run on Commercial-Off-The-Shelf (COTS) hardware.
- SDN is used to provision network connection among VNFs.
  - More dynamic flow distribution over the infrastructure

- SDN's advantages are not fully leveraged.
  - SDN only sets flow paths for GTP-U data bearers
- Logical topology is untouched. Same as conventional LTE.
  - Directly map from "physical box" to "virtual box"
  - Pre-configuration
  - Single architecture: one size fits all

Evolution (2) SDN as D-Plane: Get rid of GTP Tunnels



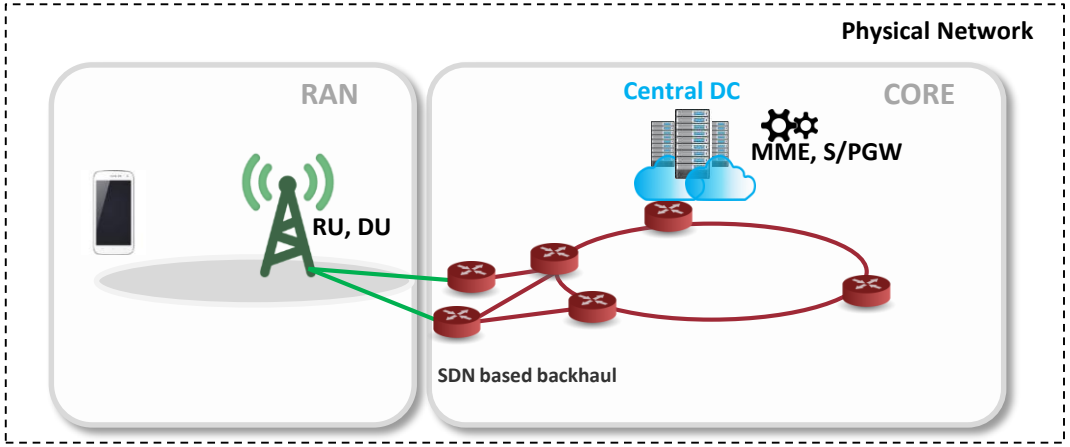


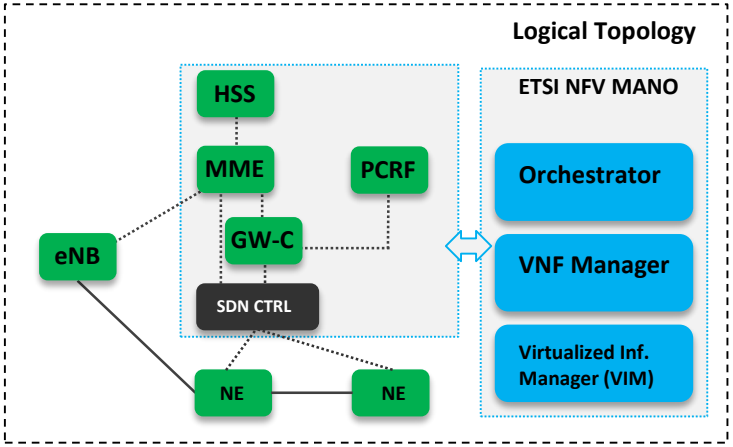


- GTP-U is replaced by flow entries directly
- Native data plane transport: less overhead on the data plane

- Logical topology is untouched. Same as conventional LTE

Evolution (3): ETSI NFV + SDN

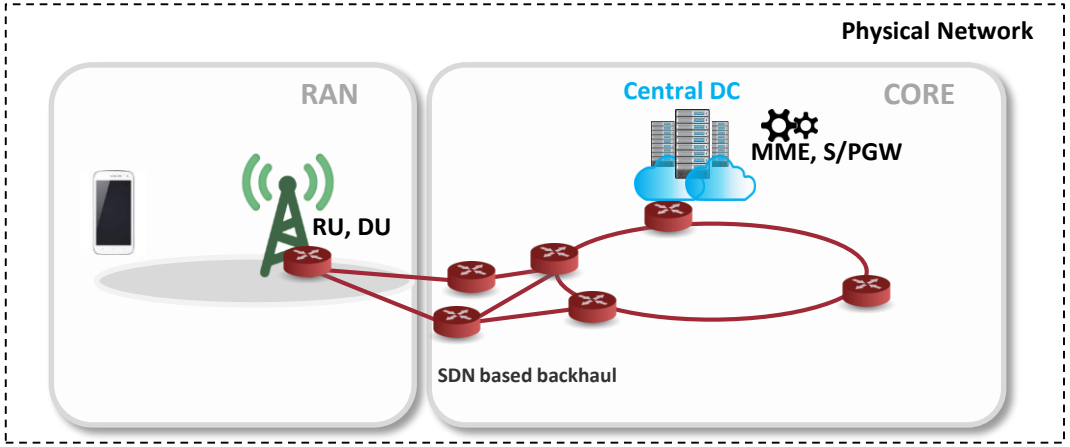


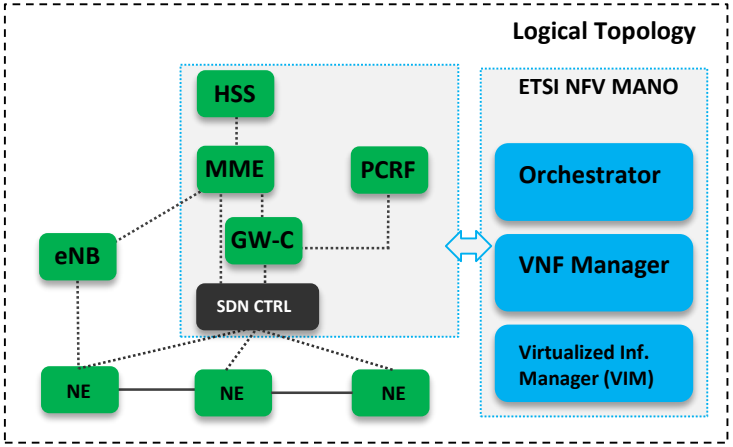


- Bring system agility
- Respond on-demand to the dynamic needs of the traffic and services running over it.
- Allows overcoming bottlenecks more easily

- Logical topology is untouched. Same as conventional LTE
- Extra management effort

Evolution (4): ETSI NFV + SDN + SDN enabled eNB

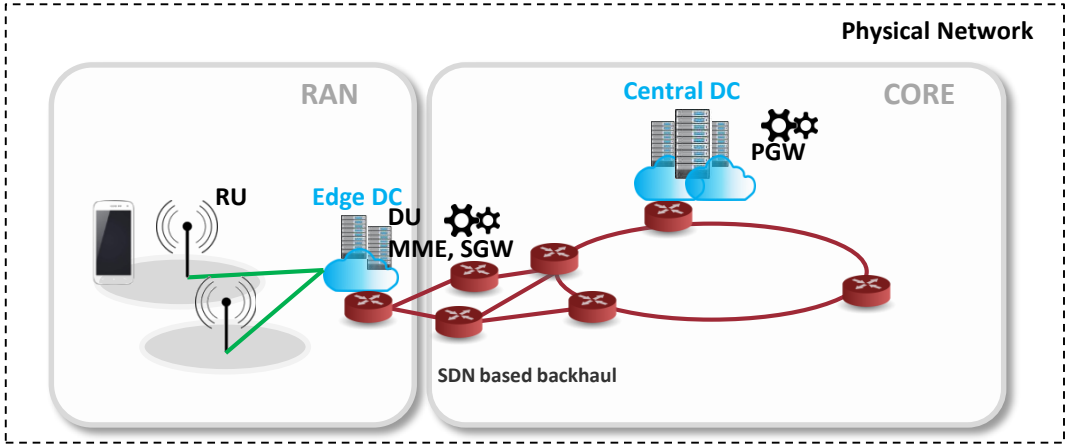


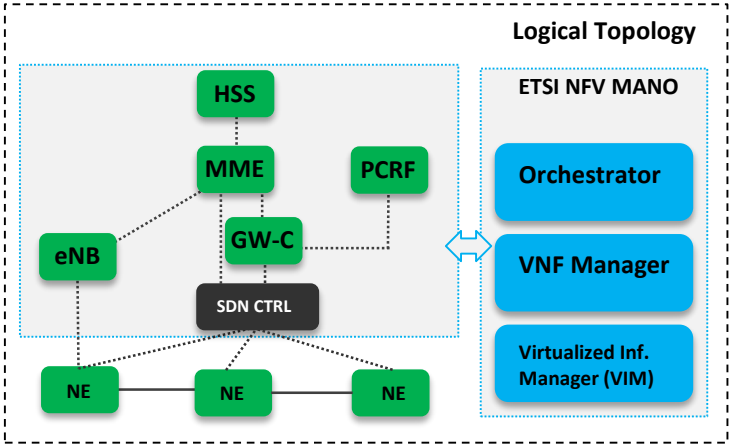


- A new functional split, flow-related parts as SDN control apps.
- Native traffic distribution in the infrastructure, depending on the SDN capabilities

- Logical topology is untouched. Same as conventional LTE
- Extra management effort

Evolution (5): ETSI NFV + SDN + MEC

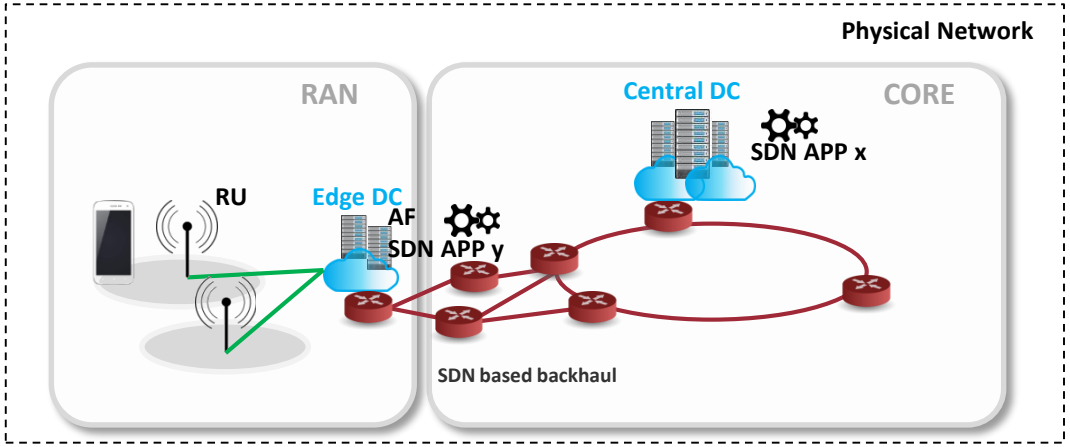


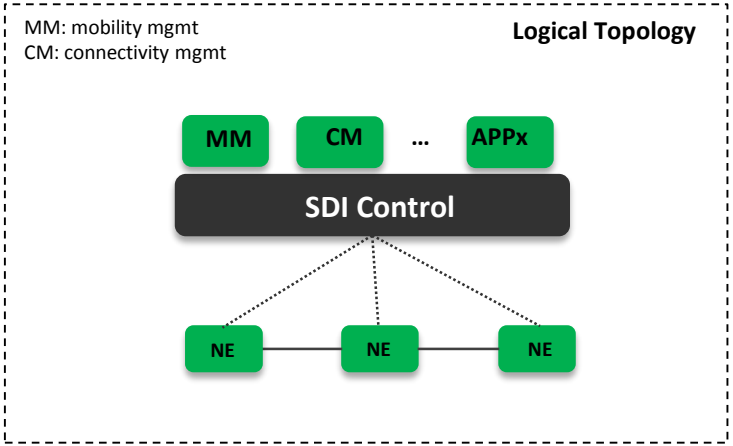


- Resource pooling at the edge.
- Enable C-RAN model.
- Enable core network functions to be placed at the edge -> reduced C-plane latency

- Logical topology is untouched. Same as conventional LTE
- Extra management effort

Evolution (6): Programmable infrastructure - Full-SDN



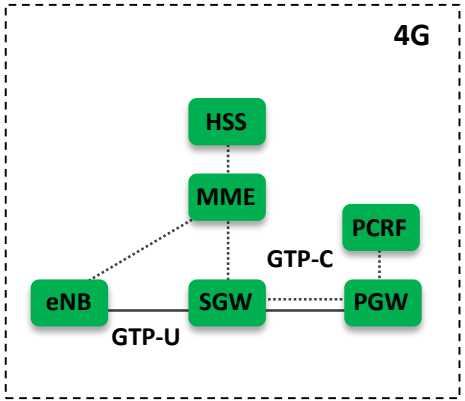


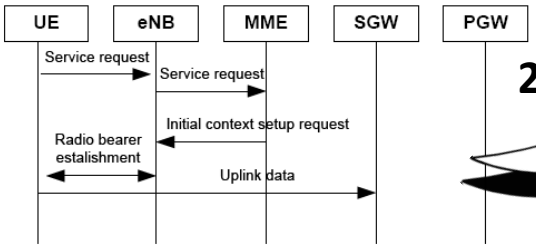
No fixed architectures anymore - architecture is a programme.

- A programmable functional split
- One unique control plane
- Programmable infrastructure: no bottlenecks, functions from the infrastructure, on the fly

- Distributed system limits, potentially too complex to realize?

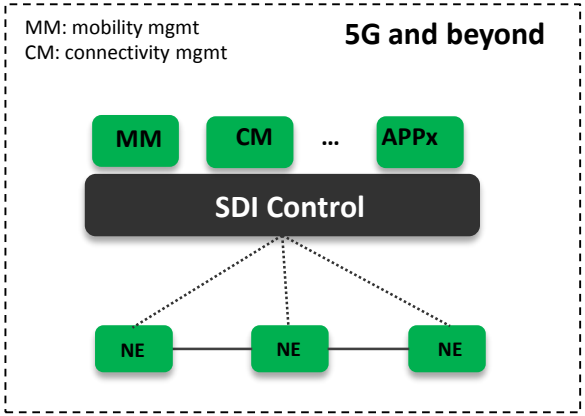
EPC vs. Full SDN MCN

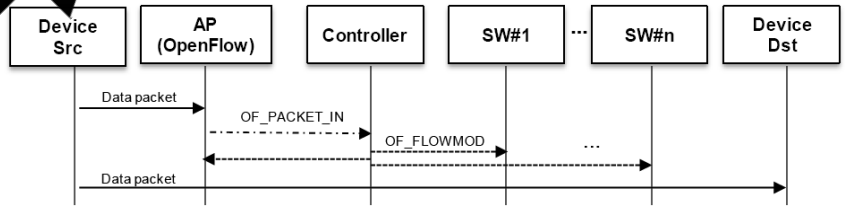




2015

2020+



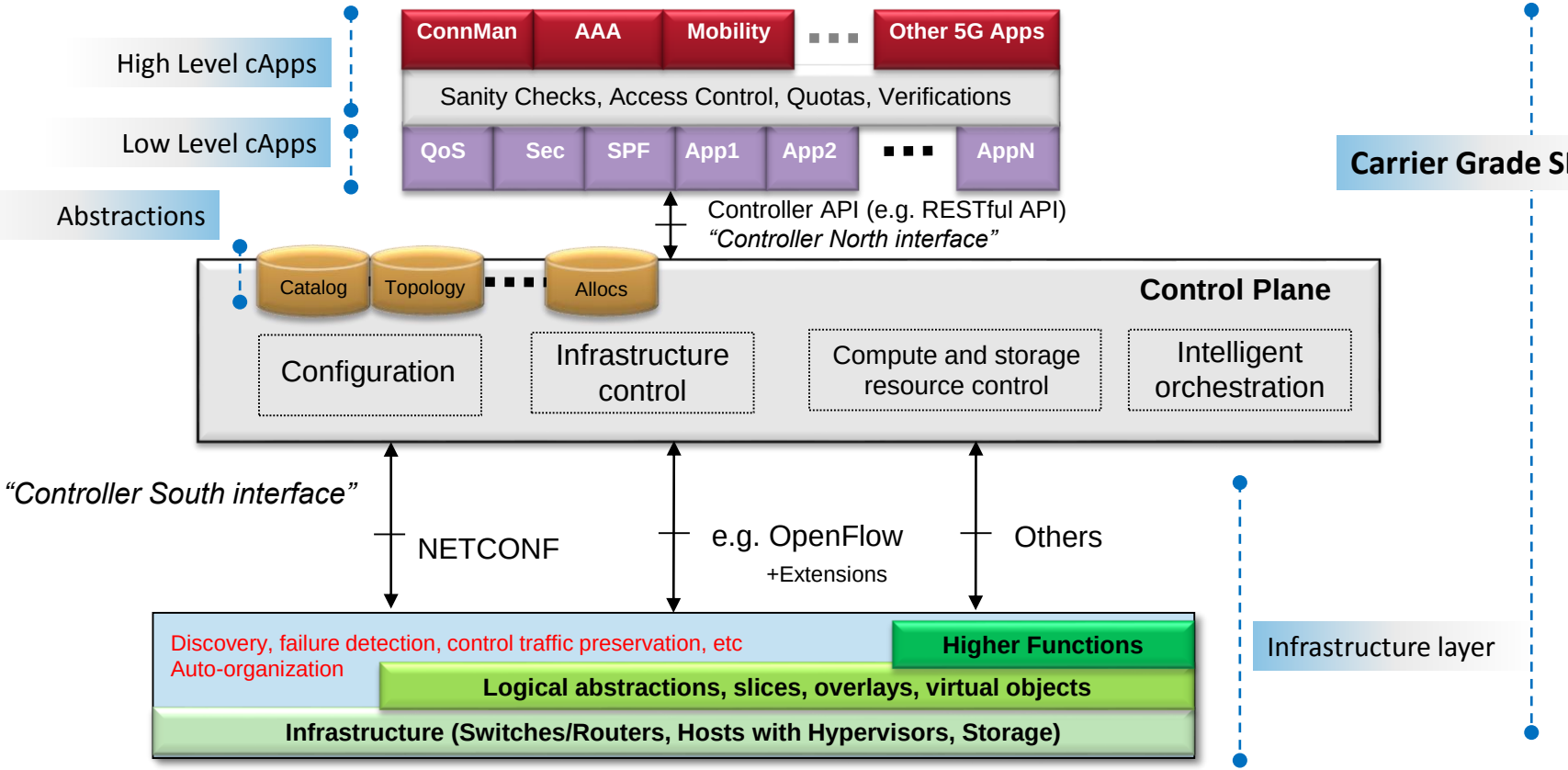


- Pre-provisioned exchanges in both C- and D-Planes, realized as overlays

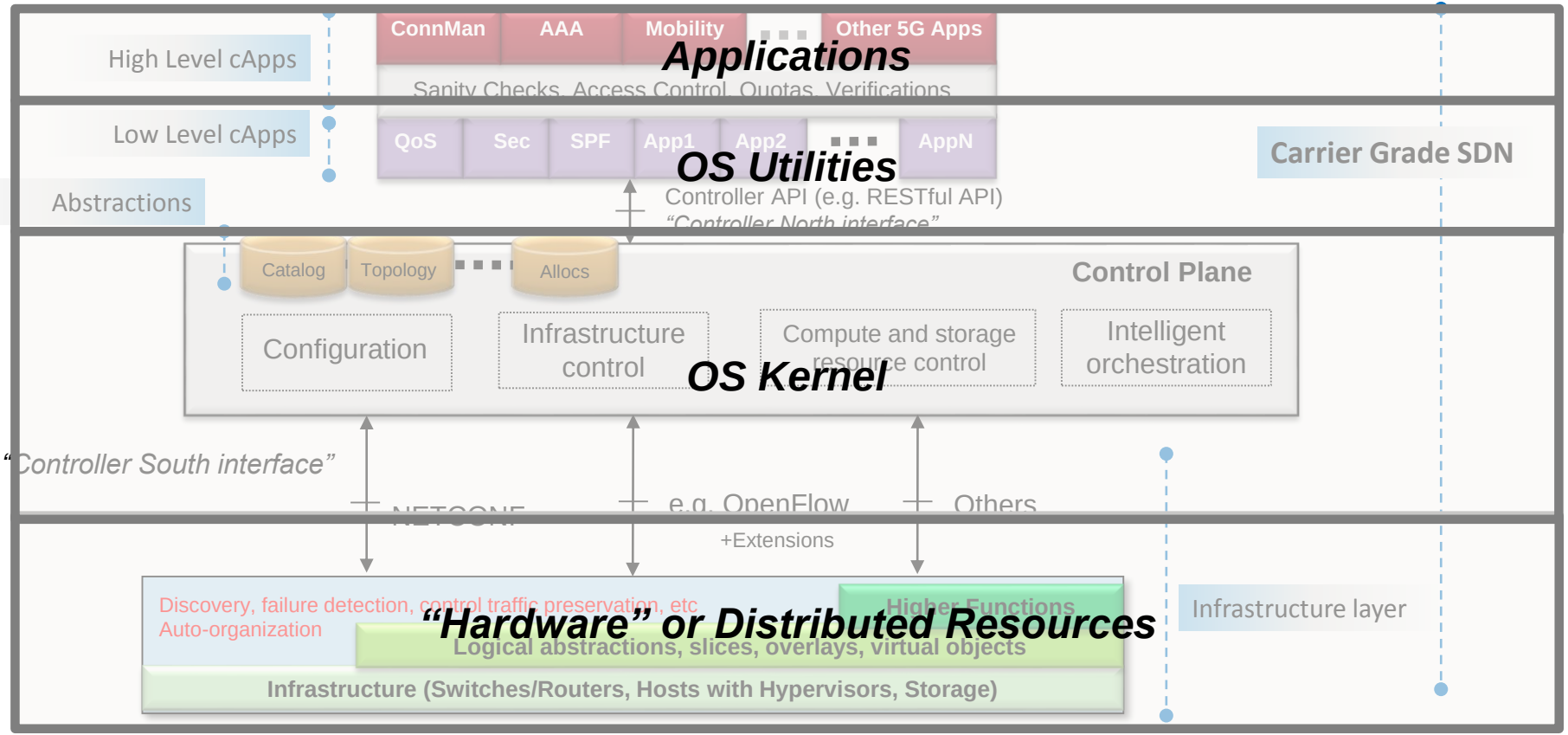
- Everything is event-based; events are dispatched from the infrastructure elements to cApps.
- CP runs on the same infrastructure (in-resource ctrl). cApps dynamically decide what to do.

What are we doing?

MCN as Full-SDN

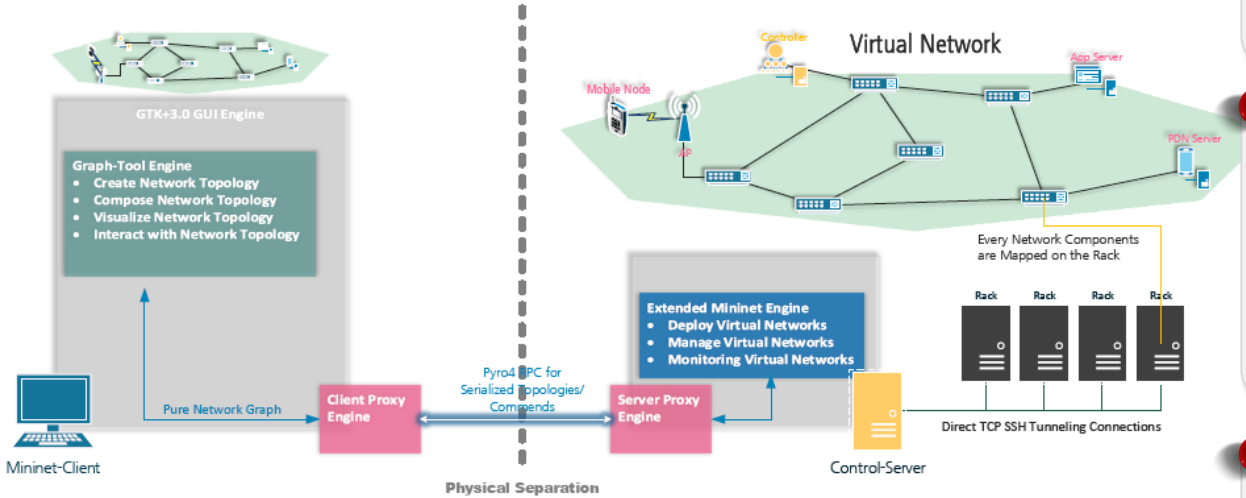


MCN as Full-SDN



Our Testbed

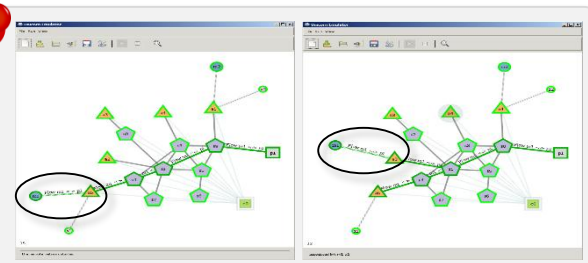
Product of
Future Network Technologies
ERC in Munich, Huawei Technologies



Client Specifications:
- OS Platform: Linux Ubuntu
- GUI Engine: GTK+3.0
- Client Proxy Engine: Pyro4 RPC

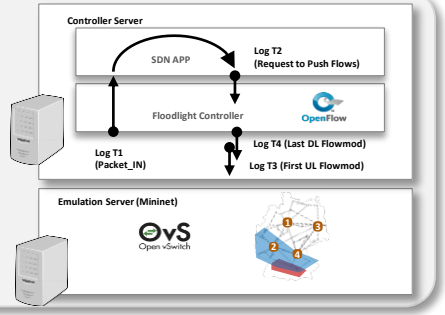
Server Specifications:
- OS Platform: Linux Ubuntu
- Mininet: Customized Mininet 2.2.1
- Server Proxy Engine: Pyro4 RPC

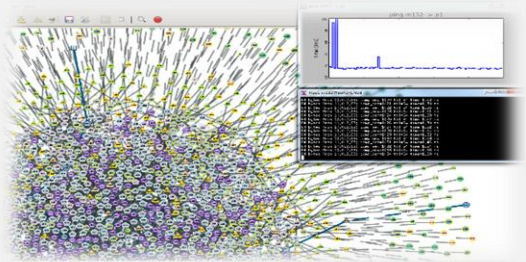
We pursue a radical goal of tuneless, anchor-less and NFV-less mobility support in SDN



MCN support: mobility, paging, etc.

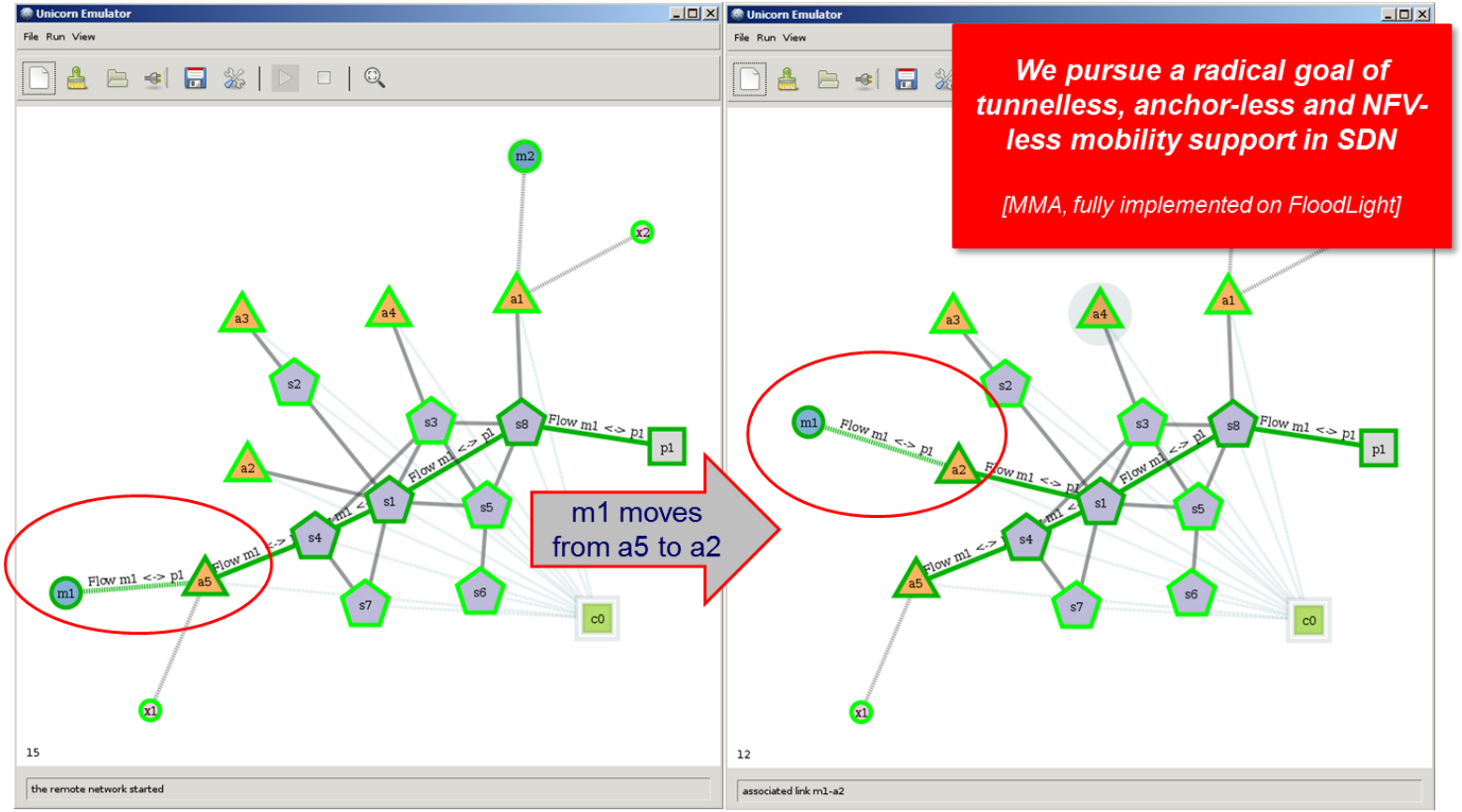
It could measure!



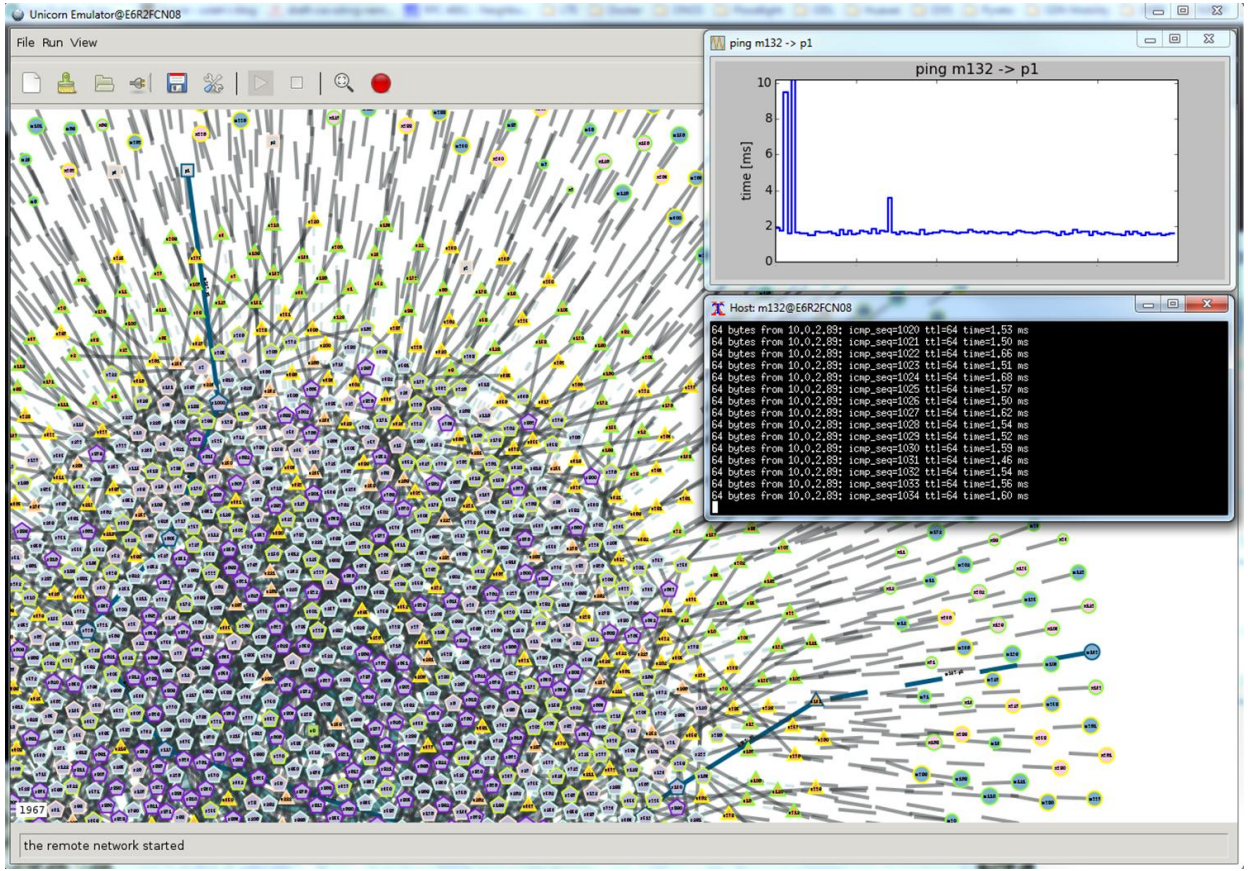


It also scales!

Nice GUI and 5G as control apps



BTW, it scales

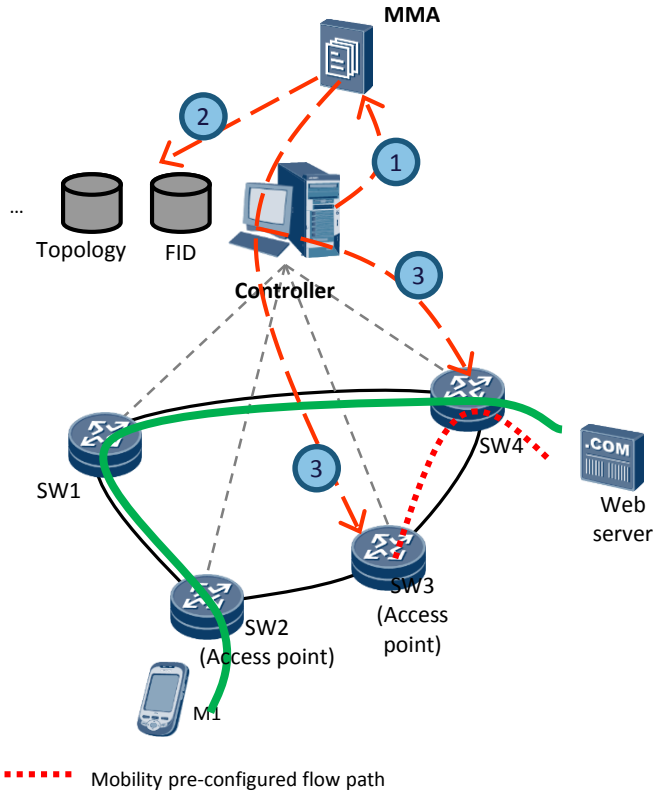


What can we do with SDN as it is now?

Mobility management

Interface (re)design

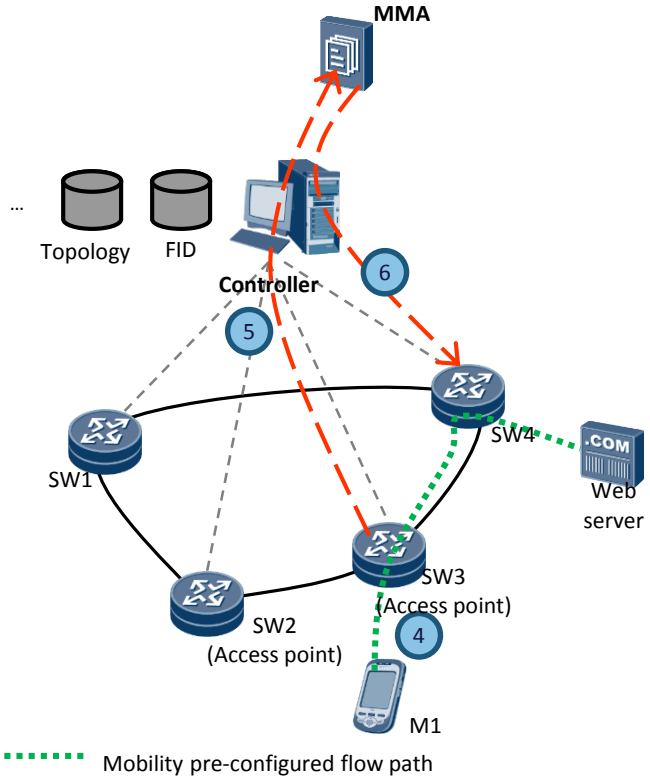
Proactive Mobility Management (1/2)



Sequence of exchanged messages:

1. Controller maintains Flow Information Database (FID), which stores the information about existing flows of the nodes in the network.
2. While M1 is at SW2, MMA retrieves the flows of M1 and selects APs to which M1 might move (e.g. SW3). It then calculates the routes from these APs to the M1's flows end-points and splits them into UL and DL parts.
3. MMA adds UL and DL flows along the calculated route(s) from each new AP (e.g. SW3) up to the intersecting switch (anchoring point, SW4) between the old and new path. In each neighbour AP two actions are associated with each flow of M1: (a) forward the packet to the next hop of the new route (e.g. SW4); (b) PACKET_IN to the controller to indicate that the new route is active.
4. When M1 moves to SW3, M1 transmissions will trigger the preconfigured M1 flow entries in SW3. The UL traffic from M1 will be forwarded to SW4 without interruption.
5. The first UL packet of M1 will generate a PACKET_IN. The controller checks the FID and determines the context of the PI and calls the MMA.
6. MMA reconfigures the rules of the intersecting switch (anchoring point, switch SW4 in the figure) so that the DL traffic is now sent down the new route.

Proactive Mobility Management (2/2)



Sequence of exchanged messages:

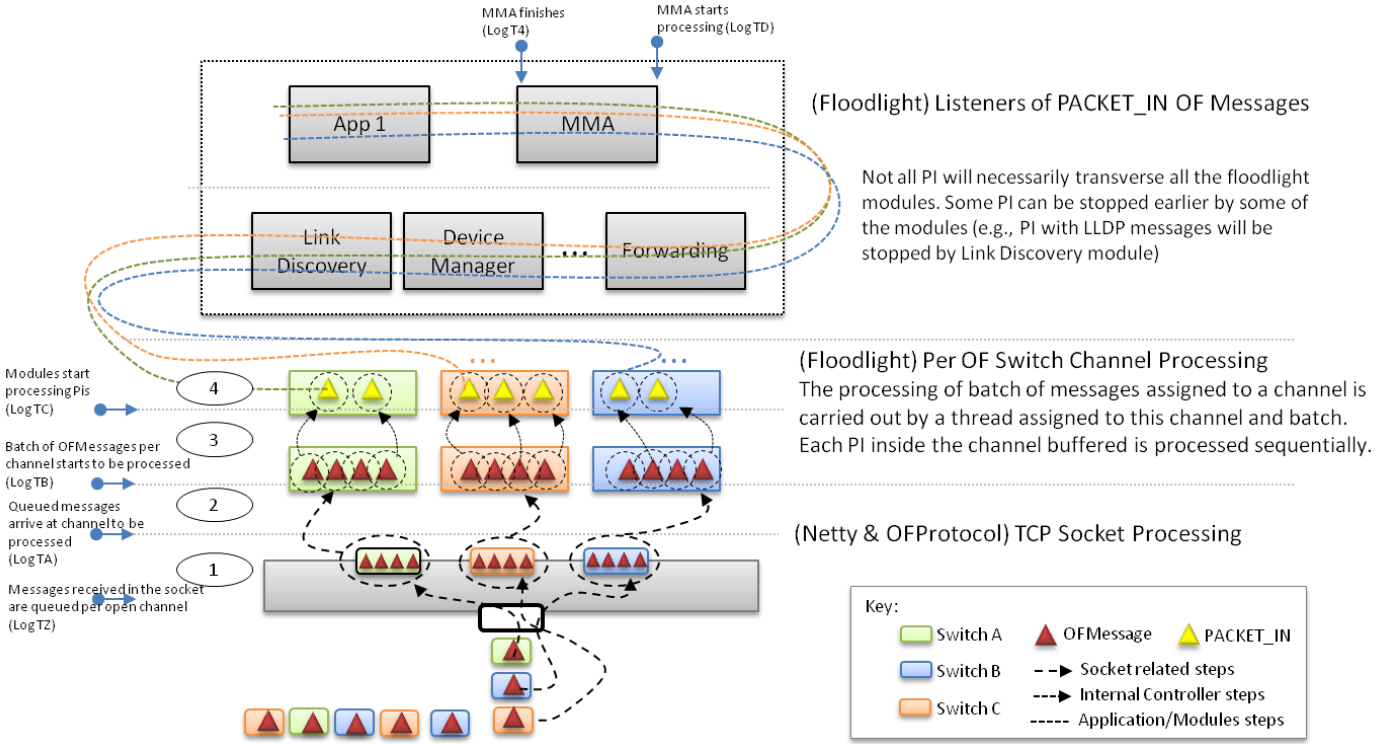
1. Controller maintains Flow Information Database (FID), which stores the information about existing flows of the nodes in the network.
2. While M1 is at SW2, MMA retrieves the flows of M1 and selects APs to which M1 might move (e.g. SW3). It then calculates the routes from these APs to the M1's flows end-points and splits them into UL and DL parts.
3. MMA adds UL and DL flows along the calculated route(s) from each new AP (e.g. SW3) up to the intersecting switch (anchoring point, SW4) between the old and new path. In each neighbour AP two actions are associated with each flow of M1: (a) forward the packet to the next hop of the new route (e.g. SW4); (b) PACKET_IN to the controller to indicate that the new route is active.
4. When M1 moves to SW3, M1 transmissions will trigger the preconfigured M1 flow entries in SW3. The UL traffic from M1 will be forwarded to SW4 without interruption.
5. The first UL packet of M1 will generate a PACKET_IN. The controller checks the FID and determines the context of the PI and calls the MMA.
6. MMA reconfigures the rules of the intersecting switch (anchoring point, switch SW4 in the figure) so that the DL traffic is now sent down the new route.

Proactive Mobility Management – Algorithm

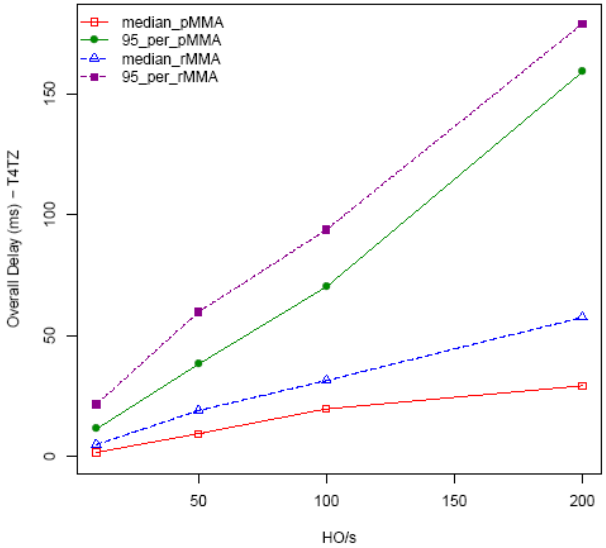
```
1   Algorithm proactiveMMA
    input :
      PI: Packet-in
      Sw: the switch generating PI
      S;D: Source and destination nodes IP addresses (obtained
    from PI)
      activeList: list of active flow paths
2     passiveList: list of passive flow paths.
3   if ((id = activeList.getID(S,D,Sw)) == 0) and
4     (id = passiveList.getID(S,D,Sw)) == 0) then
5     setActiveRules
6   else
7     activatePassiveRules
8   end
9   for n ∈ getNeighbors(Sw) do
10    calculatePassiveRules
11  end
12
13  Procedure setActiveRules
14    id = generateNewID(S,D)
15    newPath = computePath(S,D)
16    activeList.add(id, f1 = newFlow(DL,newPath))
17    activeList.add(id, f2 = newFlow(UL,newPath))
18    pushFlows(newPath, f1,DL, f2,UL)
19  Procedure activatePassiveRules
20    currPath = activeList.get(id,DL).getPath()
21    newPath = passiveList.get(id,DL).getPath()
22    crossSwitch = newPath.getLastSwitch()
23    delFlowEntry(crossSwitch, currentPath.getMatch(crossSwitch))
24    firstMatch = passiveList.get(id,UL).getPath().getFirstMatch()
25    delDoubleAction( f irstMatch)
26    delObsoleteFlows(currPath, crossSwitch)
27    aFlow = activeList.get(id)
28    aFlow.setPath(mergePaths(id,newPath,currPath))
29    pFlow = passiveList.get(id)
30    pFlow.setPath(mergePaths(id,newPath,currPath))
31
32  Procedure calculatePassiveRules
33    path = computeMobilityPath(n,D,newPath)
34    setDoubleAction(path.getFirstSwitch())
35    pushFlows(path,DL,UL)
36    decreasePriority(path.getLastSwitch(),DL)
37    passiveList.add(id,n,DL, path)
38    passiveList.add(id,n,UL, path)
```

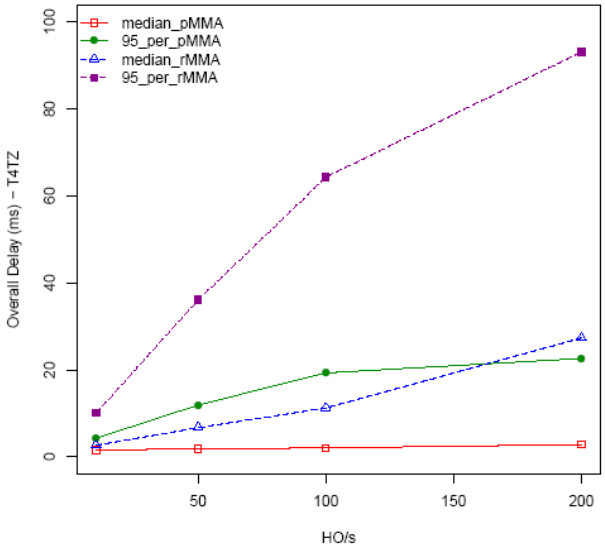
Implementation and logging

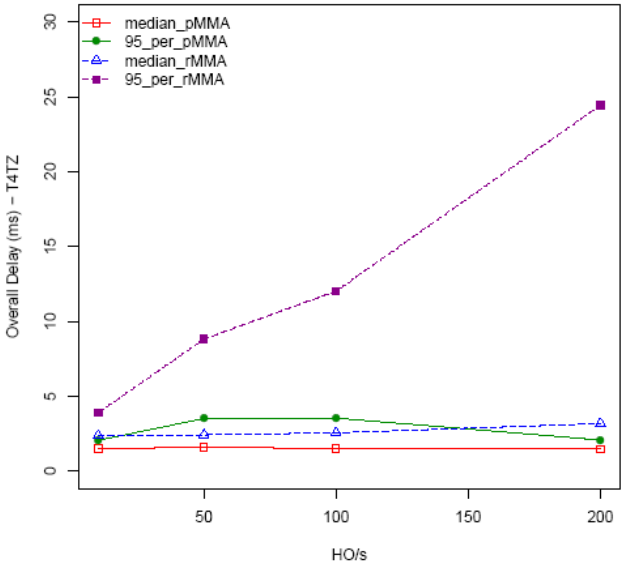
System Model and Logging points



Proactive vs. reactive mobility - results

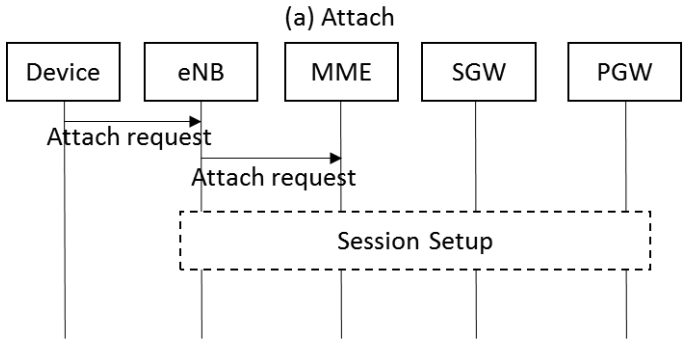


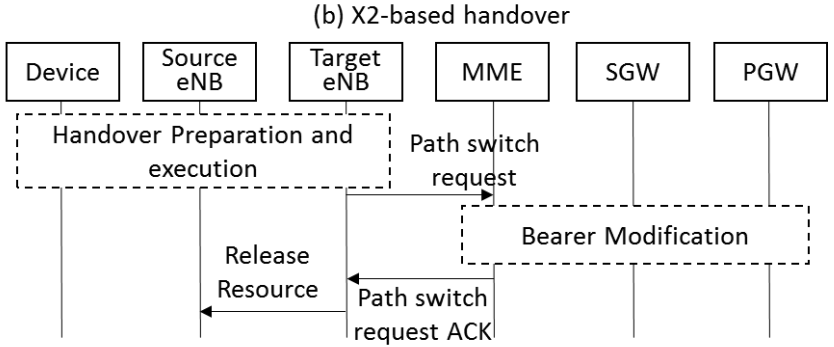




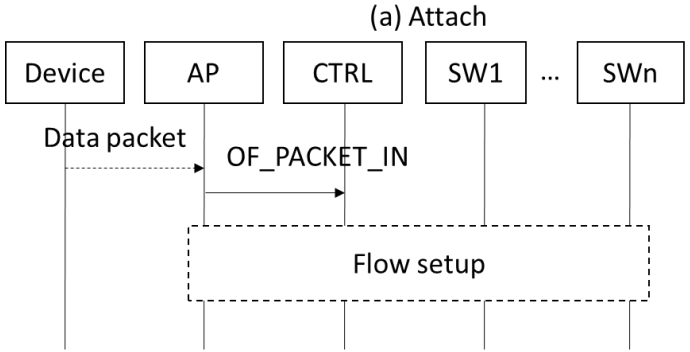
Interfaces in EPC and OF

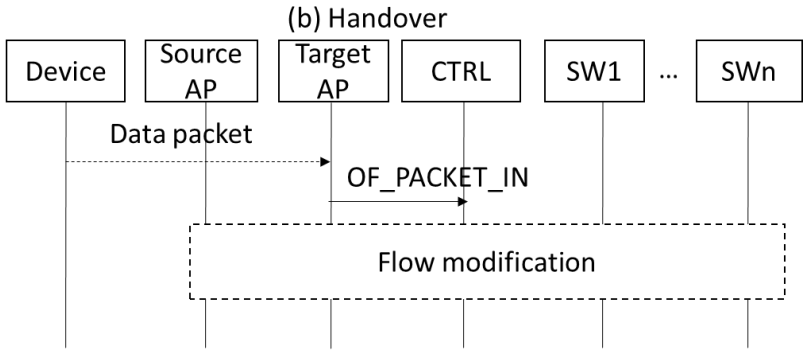
EPC





OF

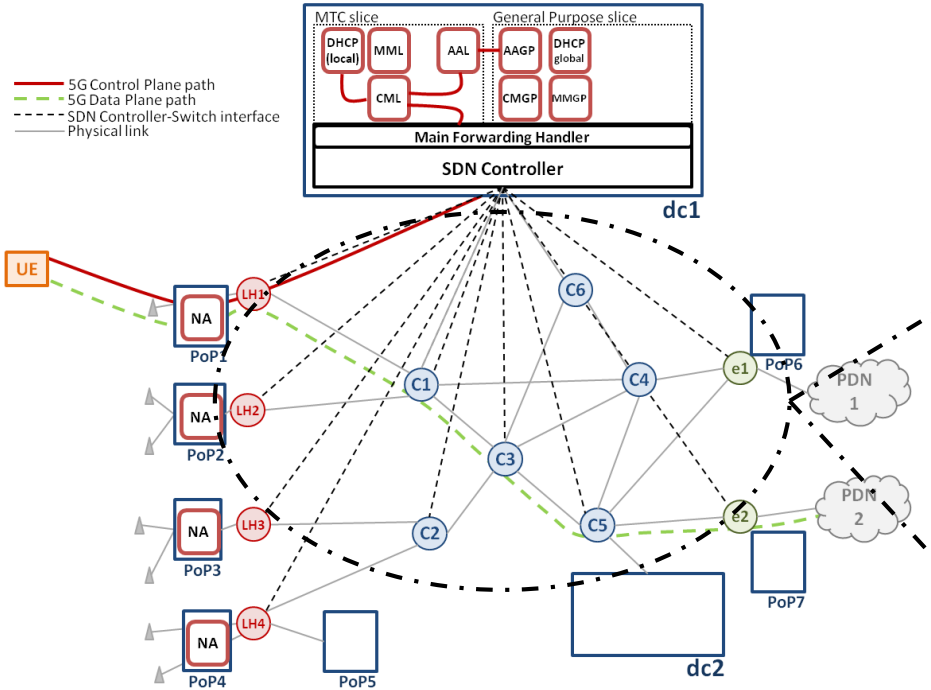


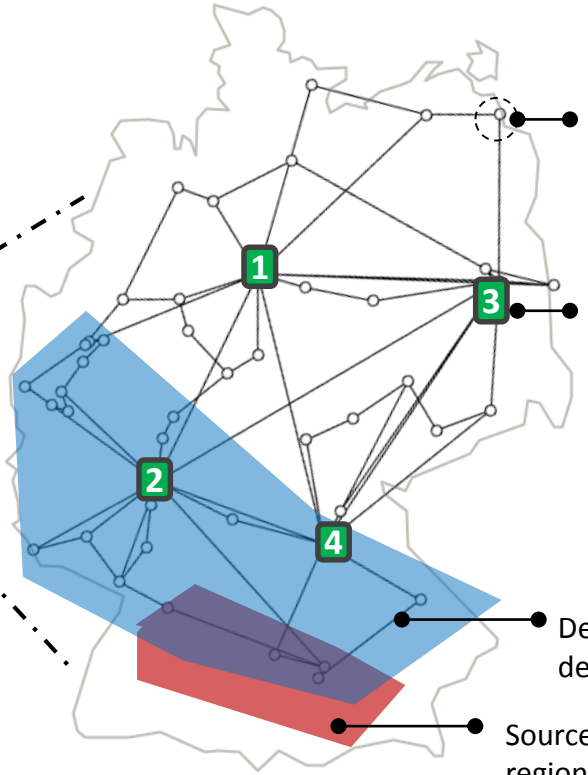


Context Interpretation

- Maintain network state info (NSI):
  - (*src*, *dst*, *attP*, *fm*, *fp*, *status*)
- Retrieve relevant info from PACKET_IN and check against NSI
- Draw the conclusion from the result
- *Examples*
  - NSI does not contain any entry with the same source address field of the incoming PACKET_IN → *attachment*
  - NSI contains one entry that has the same src, dst and matching field as the incoming PACKET_IN AND the switch that sent the PACKET_IN is different from the attP of the above entry

Testbed





Core switch

Location for SGW/PGW

Destination device region

Source device region

C-Plane/U-Plane functions:
AAL, AAGP: Authentication Authorization Local, AA General Purpose
CML, CMGP: Connectivity Management Local, CM General Purpose
MML, MMGP: Mobility Management Local, MM General Purpose
AN: Access Node
GW: Gateway

Forwarding elements:
LH Last Hop Forwarding Element
C Core Forwarding Element
e Network Entry Point

Results

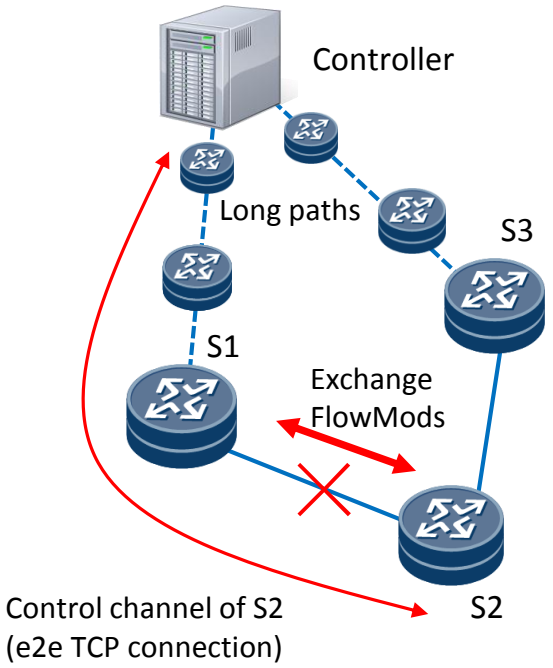
TABLE I: Interpretation of PIs associated with Attachment, Mobility and Renew Flow events (unit: millisecond)

	Attach				Mobility				New Flow			
	1 CPU		4 CPU		1 CPU		4 CPU		1 CPU		4 CPU	
	Median	95th %	Median	95th %	Median	95th%	Median	95th%	Median	95th%	Median	95th%
50	0.0526	0.1249	0.0531	0.1281	0.0502	2.3312	0.0542	0.1125	0.0484	0.0651	0.0511	0.0829
100	0.0536	0.1208	0.0531	0.1101	0.0528	3.8006	0.0530	0.0986	0.0453	0.0625	0.0519	0.0960
200	0.0517	0.1134	0.0433	0.1082	0.0537	4.3648	0.0525	0.1018	0.0461	0.0593	0.0494	0.0851
300	0.0498	0.1118	0.0407	0.1070	0.0538	4.4773	0.0519	0.1100	0.0462	0.0608	0.0494	0.0755
400	0.0418	0.1053	0.0409	0.1033	0.0521	3.9133	0.0529	0.6581	0.0457	0.0585	0.0495	0.0806

Is this the last word of SDN?

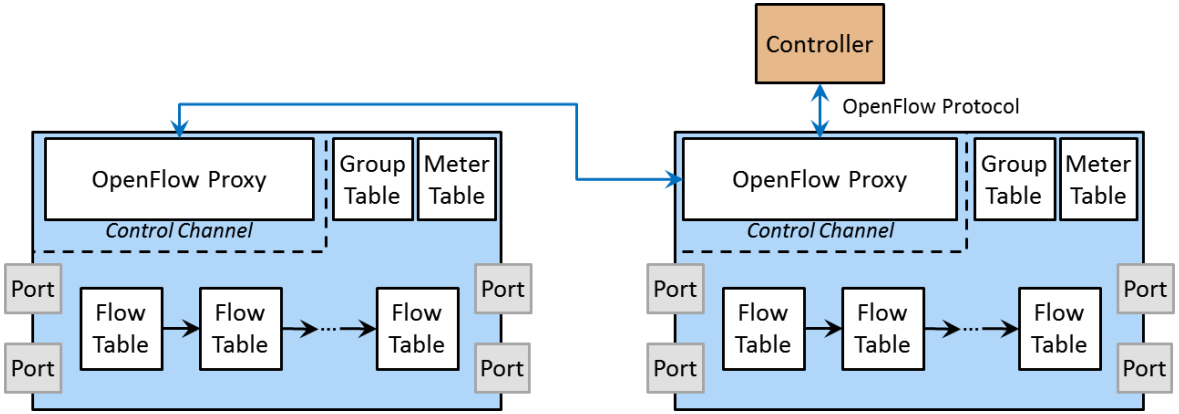
Can it do better?

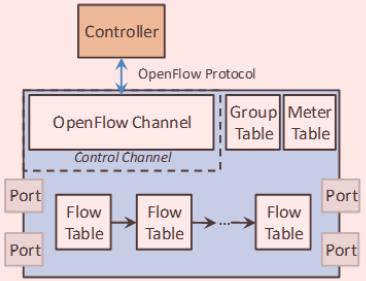
OpenFlow is still at the beginning



Problem description	Problem type
All switches must have IP configured to forward the control traffic	Bad design
Slow failure recovery (only end-to-end)	Performance issue
Local, switch-to-switch, interactions to implement control applications impossible	Performance issue
…	…

OpenFlow Proxy

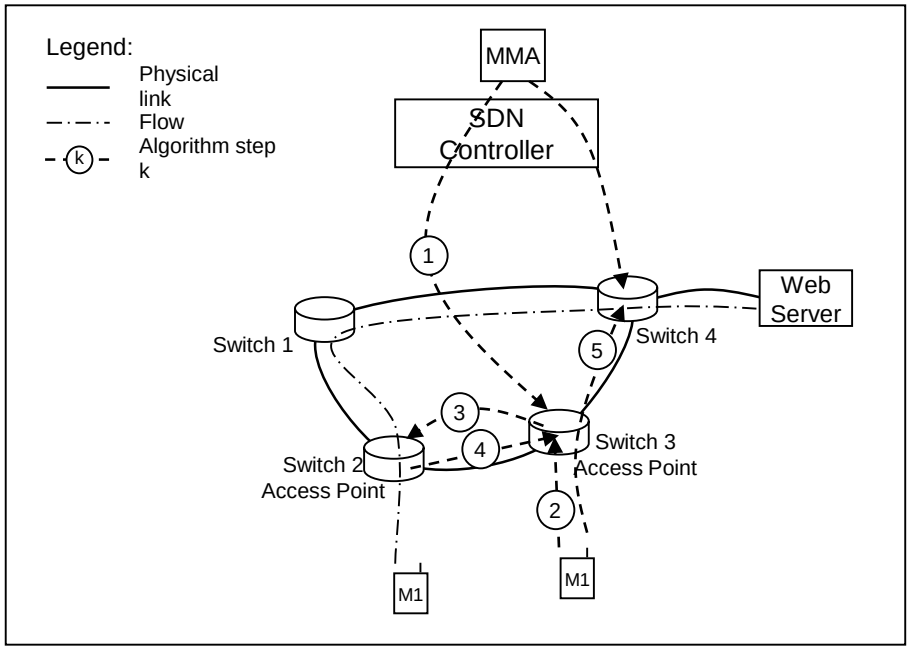




The OpenFlow channel is usually instantiated as a single network connection between the switch and the controller, using TLS or plain TCP (see 6.3.5). Alternatively, the OpenFlow channel may be composed of multiple network connections to exploit parallelism (see 6.3.7). The OpenFlow switch must be able to create an OpenFlow channel by initiating a connection to an OpenFlow controller (see 6.3.2). Some switch implementations may optionally allow an OpenFlow controller to connect to the OpenFlow switch, in this case the switch usually should restrict itself to secured connections (see 6.3.5) to prevent unauthorised connections.

To contrast with this

Mobility, once again



1. While a mobile node M1 is attached to switch 2 (operating also as an access point), MMA pre-configures a flow rule at a neighboring switch 3 to send a FlowMod to switch 2. This FlowMod instructs switch 2 to forward the DL traffic of M1 to switch 3. MMA also installs a rule at switch 3 that forwards M1 uplink traffic to switch 4. In switch 2 MMA installs a rule that matches on the packets from switch 3 and as action installs the requested rule.
2. M1 moves to switch 3 and triggers the FlowMod transfer to switch 2.
3. The FlowMod is sent to switch 2. Switch 2 stores it in its flow table.
4. Further downlink packets for M1 flow from switch 2 to switch 3 (due to the rule just installed).
5. Uplink traffic of M1 is forwarded to switch 4. Thus it follows the new path, which it should follow after the handover.

Is OF-SDN the last word of SDN?

- SDN: Software Defined Networking
  - But what exactly is ***defined by software*** and what is not?
- Three Different Aspects of Networking:
  1. Network is graph: topology construction, maintenance
  2. Network is exchange: protocols, communication stacks
  3. Network is sharing: resource sharing by traffic distribution
- Do we have an SDN?
  - OpenFlow: Software Defined Traffic Steering
    - What about other types of resources?
  - What about 1 and 2?

Summary

- Wide range of reasons why SDN is a useful technology
- Plenty of strong reasons why we need more flexible mobile architecture
  - Many believe that SDN can bring the needed flexibility
- How can the current SDN model be utilized
  - Mobility can work; bearers and QoS can work
- But there is still long way to go to find the right role of SDN in mobile networks
- Besides, it makes sense to find first the right SDN model

JOIN US IN

BUILDING A BETTER CONNECTED WORLD

THANK YOU

HUAWEI EUROPEAN RESEARCH CENTER