5G Crosshaul

CREATING THE LIVING NETWORK™

Ethernet OAM and SDN: a matching opportunity

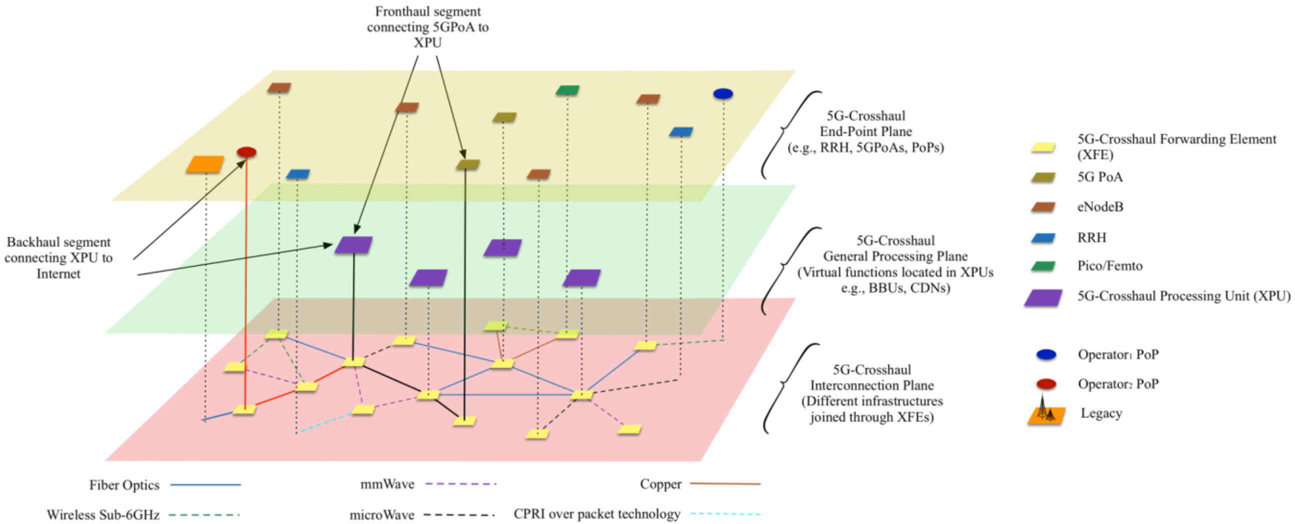
Luca Cominardi

27/06/2016 – EuCNC'16 – Athens

INTERDIGITAL

5G-Crosshaul in 30 seconds

- The 5G-Crosshaul project aims at developing a 5G **integrated _Ethernet-based_ backhaul** and **fronthaul** transport network enabling a flexible and **software-defined reconfiguration** of all networking elements in a multi-tenant and service-oriented ***unified management environment***



Problem statement

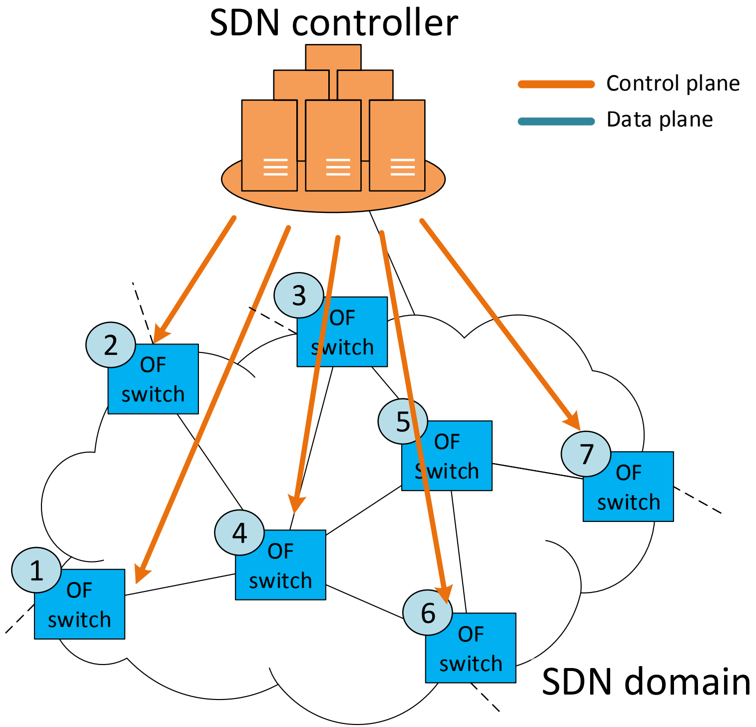
5G Crosshaul

- Ethernet Operation, Administration, and Mantainence (OAM) defines complex and **stateful** mechanisms
  - Fault management and performance monitoring [ITU-T Y.1731]
  - Connectivity fault management [IEEE 802.1ag]
  - Link layer discovery [IEEE 802.1ab]
  - Ethernet protection switching [ITU G.8031]
- Pure SDN switches performs **stateless** packet lookups and forwarding
  - E.g., OpenFlow does not provide any mechanisms for:
    - Counting events, timers, packet generation, react to a specific set of events
      - E.g., it cannot implement "send a notification back to the controller if I haven't received 2 heartbeat messages in the last 50ms / reply to a Link Trace Message with a Trace Route Reply"
- OAM protocols are not standardized in some parts of IMT networks such as the fronthaul network [ITU FG IMT-2020]

OpenFlow – LLDP example (0)

5GXCrosshaul

- Assuming the OpenFlow controller wants to discover the network topology
  - E.g., OpenDaylight already implements this procedure
- Link Layer Discovery Protocol (LLDP) provides the means for topology discovery
- Preliminary procedure:
  - The SDN controller configures the rules for matching LLDP on each OpenFlow switch:
    - Match ingress packets on LLDP multicast address and LLDP EtherType
    - Send the matched packets to the controller

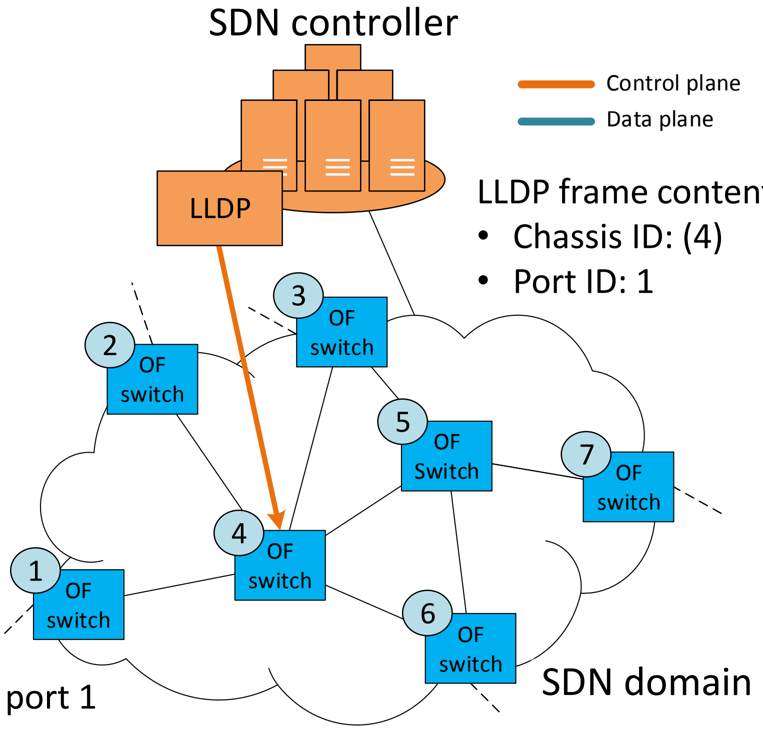


INTERDIGITAL

OpenFlow – LLDP example (1)

5GXCrosshaul

- The topology discovery is done by the OpenFlow controller:
  1. Forge an LLDP packet for every combination of (switch, port)
  2. Send the LLDP packet as payload of an OpenFlow message to the switch via the control channel
- The procedure is executed every *e.g.* ***30 seconds***



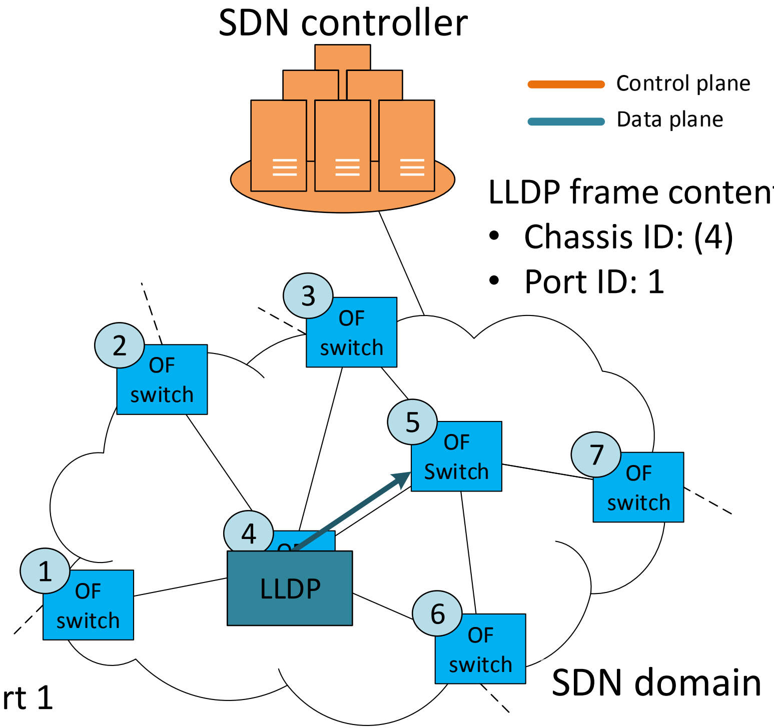
LLDP frame content:
- Chassis ID: (4)
- Port ID: 1

OpenFlow message content:
- Destination: (4)
- Send encapsulated frame on port 1

INTERDIGITAL

OpenFlow – LLDP example (2)

- The switch (4) receives over the control channel the OpenFlow message containing the LLDP frame
  - The switch (4) extracts the LLDP frame
  - The switch (4) sends the LLDP frame on the data plane port 1 as indicated in the OpenFlow message

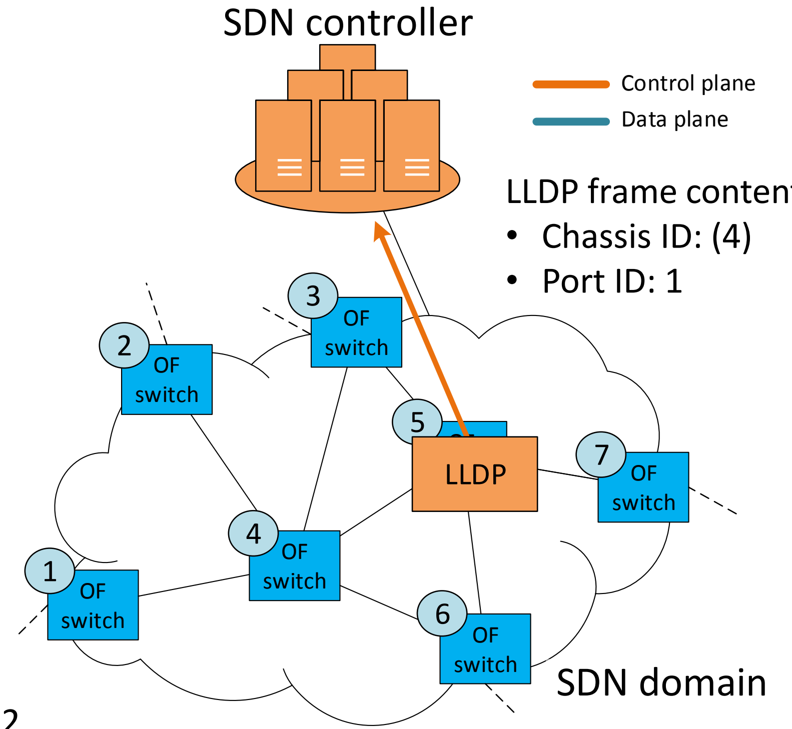


LLDP frame content:
- Chassis ID: (4)
- Port ID: 1

OpenFlow message content:
- Destination: (4)
- Send encapsulated frame on port 1

OpenFlow – LLDP example (3)

- The switch (5) matches the incoming LLDP frame on port 2
  - The switch (5) encapsulates the LLDP frame into an OpenFlow message
  - The switch (5) sends the OpenFlow message to the SDN controller
- The controller now knows that there is connectivity between (4, port 1) and (5, port 2)

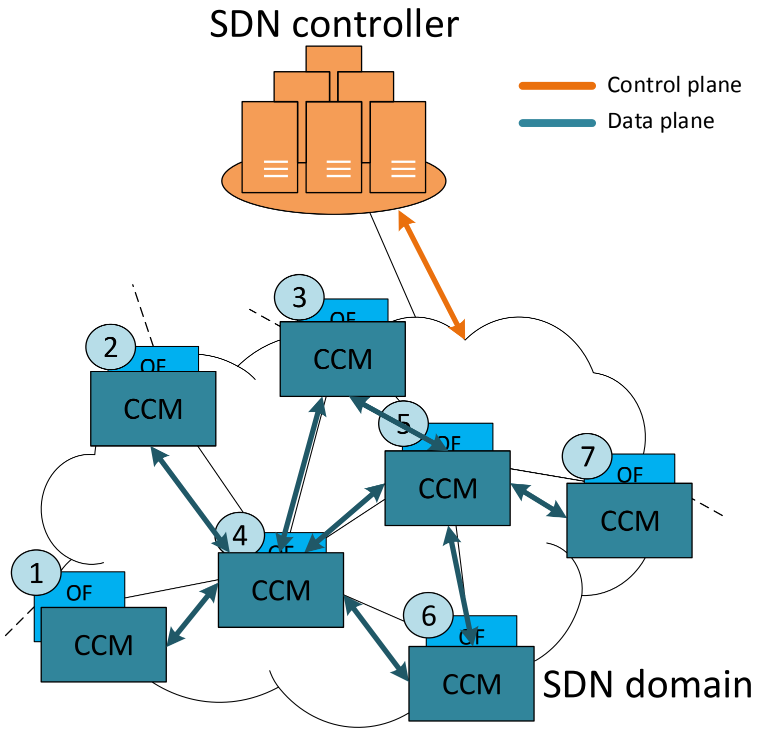


LLDP frame content:
- Chassis ID: (4)
- Port ID: 1

OpenFlow message content:
- Source: (5)
- Destination: (SDN Controller)
- Received LLDP frame on port 2

OpenFlow – CCM example

- The Continuity Check Message (CCM) provides a means to detect connectivity failures
- Akin to LLDP, the CCM is done by the OpenFlow controller:
  - Forge a CCM packet for every combination of (switch, port)
  - Send the CCM packet as payload of an OpenFlow message to the switch via the control channel
  - Send the CCM packets back to the OpenFlow controller upon reception
- The procedures is executed every *e.g.* ***10 milliseconds***



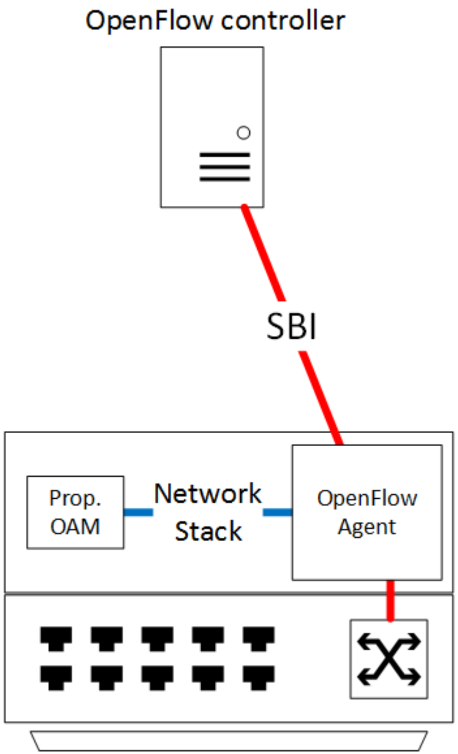
Stateful operations

5GXCrosshaul

- Stateful logic can be implemented in two ways:
  - On the **control plane** (OpenFlow controller – Previous examples: LLDP/CCM)
    - Pro: There is no need to change anything on the switch
    - Cons: All the logic implemented on the controller, overload of the control channel, reduced reactiveness due to the remote controller
      - ***Think about CCM messages generated and sent every 10/50ms, or link failure notification***
  - On the **data plane** (OpenFlow switch) via external software interacting directly with the OpenFlow agent
    - Pro: Offload of the controller and remote control channel, more reactive due locality.
      - ***Keep local at the switch the execution of procedures having local scope without the need of involving a remote controller every time. The controller is still in charge of the configuration.***
    - Cons: Switches need to implement complex and stateful mechanisms, no standard interface to do that.
      - ***Who configures (or install) those pieces of software? What kind of interface?***

Data plane options

5G Crosshaul



Option 1

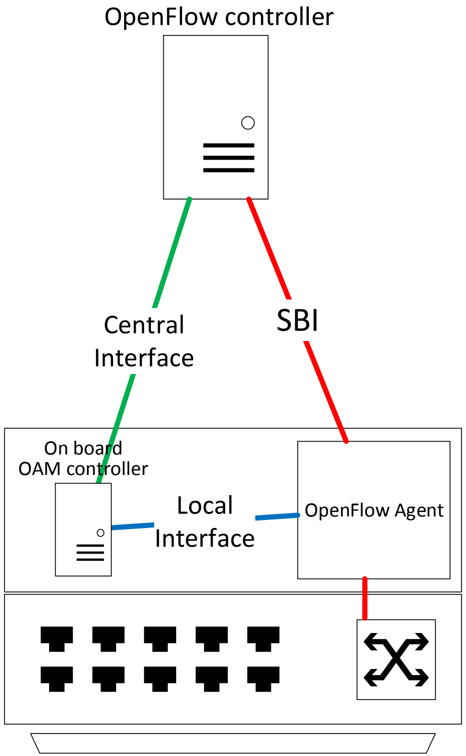
- No need to specify any new interface
- Static configuration
- Proprietary interface
- No interaction with the controller
- Binary blob generating and consuming OAM packets
- Proprietary hooks required to interact with the OpenFlow agent

State of art

Option 2

- OAM mechanisms can be dynamically (re)configured via a *Central Interface* according to network and controller needs
- Need to define OAM *primitives* for:
  - OAM packet gen.
  - Consumption of OAM packets
  - Counters/Timers
  - Associated actions
- ***Unified OAM management for fronthaul and backhaul***

Area of interest



INTERDIGITAL

5G-Crosshaul consortium

5GXCrosshaul

NEC
ERICSSON
Atos
NOKIA
INTERDIGITAL EUROPE

Telefónica
Telefónica I+D
TELECOM ITALIA
orange™
Universidad Carlos III de Madrid

visiona
EBlink
NEXTWORKS ENGINEERING FORWARD
cnd core network dynamics
TELNET Redes Inteligentes

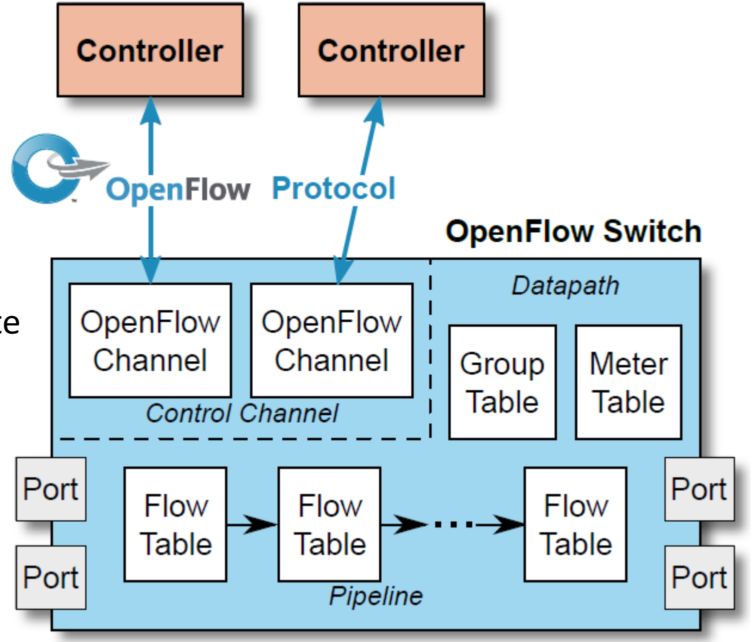
Fraunhofer HHI
CTTC
CREATE-NET
POLITECNICO DI TORINO
LUND UNIVERSITY
ITRI Industrial Technology Research Institute

INTERDIGITAL

Backup slides

OpenFlow 1.5 – Overview

OpenFlow 1.5 – Overview

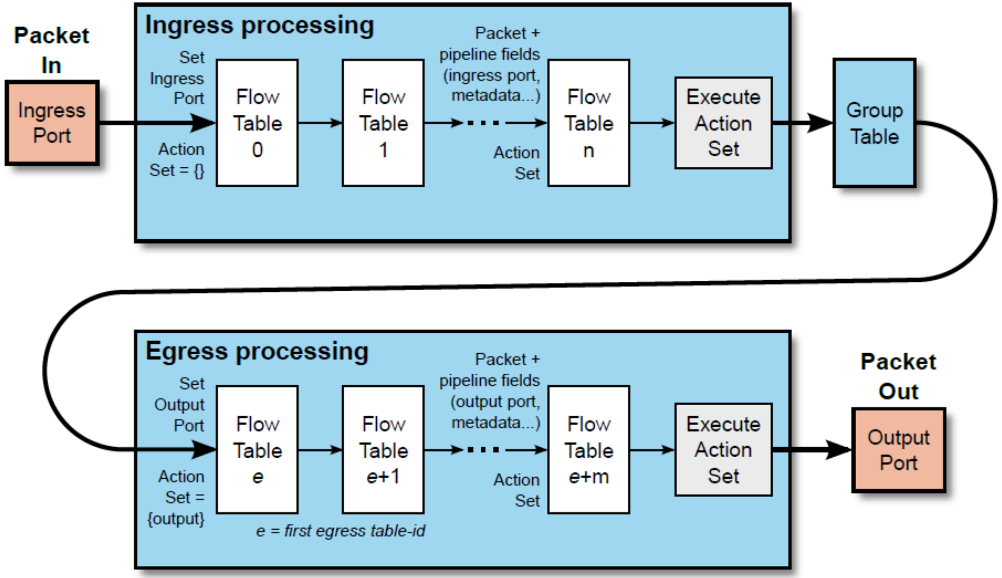


- An **OpenFlow** Logical Switch consists of one or more *flow tables* and a *group table*, which perform **stateless** packet lookups and forwarding, and one or more *OpenFlow channels* to a controller
- The controller is typically external and can add, update, and delete flow entries in flow tables, both reactively and proactively
- Matching starts at the first flow table and may continue to additional flow tables of the pipeline
- Instructions associated with each flow entry either contain actions or modify pipeline processing
- Flow entries may forward to a physical, logical, or reserved port
- Actions associated with flow entries may direct packets to a group for additional processing
- The group table contains group entries; each group entry contains a list of action buckets with specific semantics dependent on group type (e.g., multicast, flooding, multipath).
- Switch designers are free to implement the internals in any way convenient, provided that correct match and instruction semantics are preserved

INTERDIGITAL

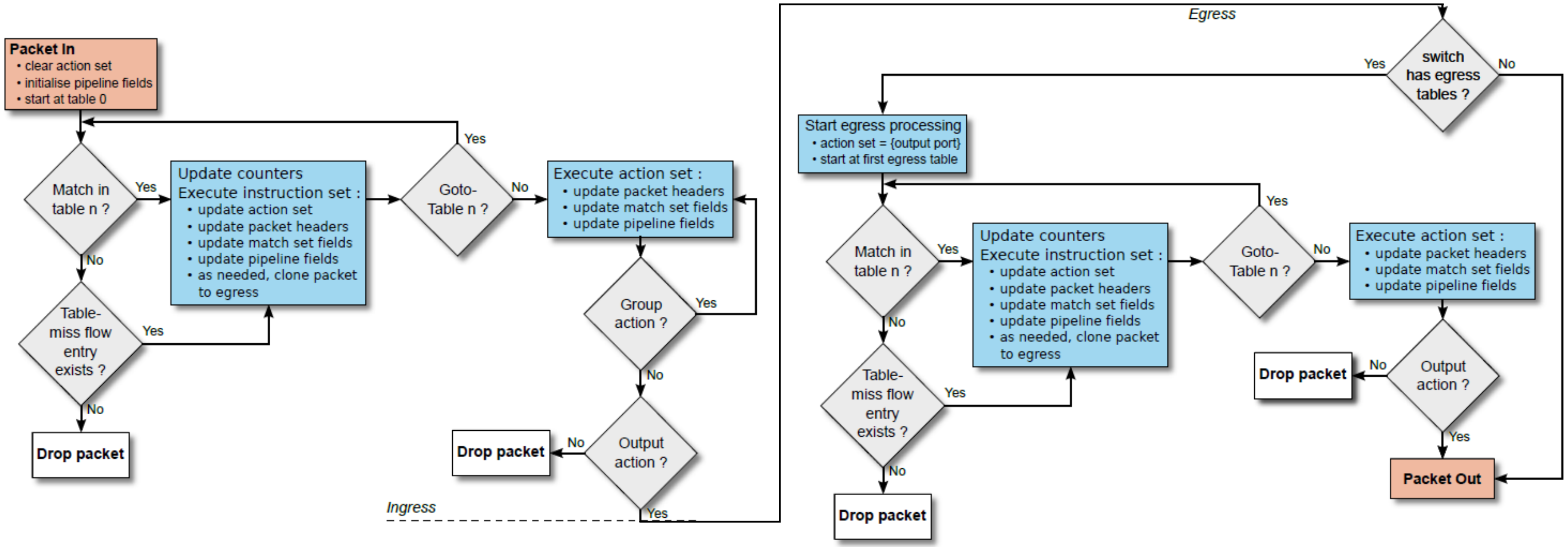
OpenFlow 1.5 – Pipeline processing (1)

- OpenFlow switches come in two types: *OpenFlow-only*, and *OpenFlow-hybrid*
- The OpenFlow pipeline is an **abstraction** that is **mapped** to the switch hardware
- A flow table entry is identified by its match fields and priority and is stateless
- A flow entry instruction may contain **actions** to be performed on the packet at **some point of the pipeline**
- Packet match fields used for table lookups depend on the packet type



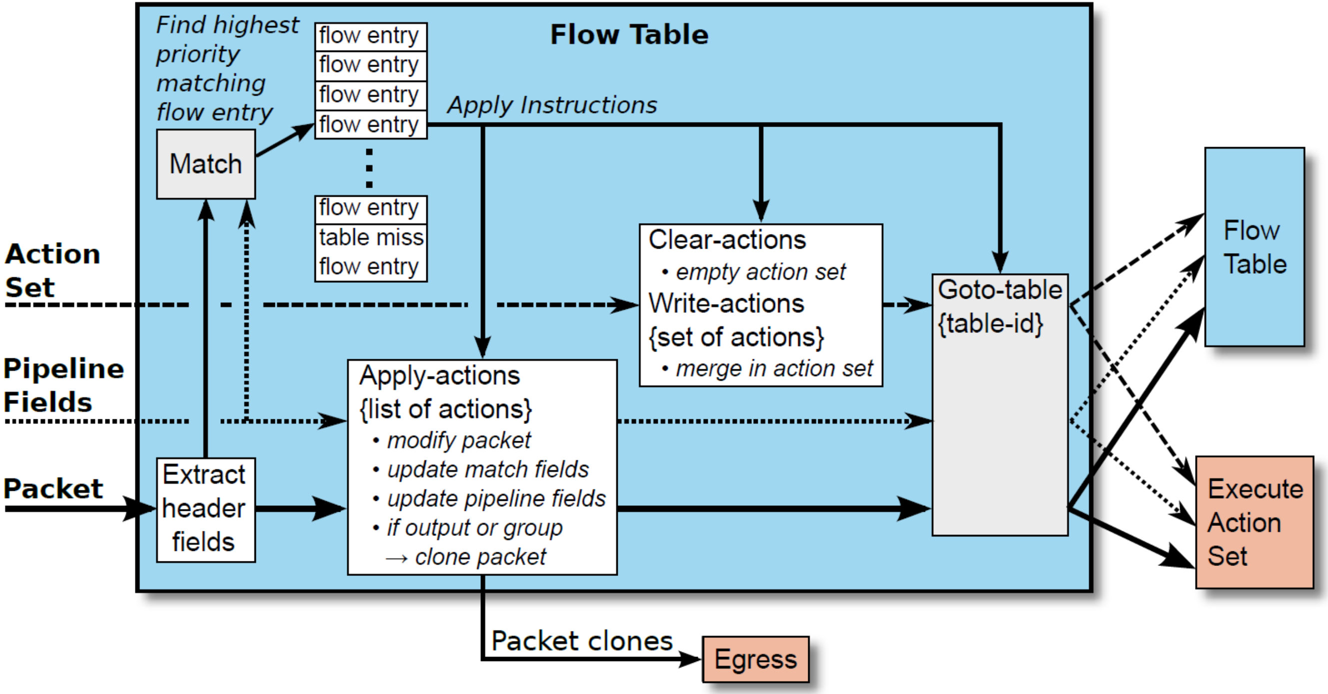
- Every flow table must support a table-miss flow entry to process table misses
- *Ingress processing* happens when the **packet enters** the OpenFlow switch
- *Egress processing* is the processing that happens after the determination of the output port (**output port context**)

OpenFlow 1.5 – Pipeline processing (2)



OpenFlow 1.5 – Flow Table

5GXCrosshaul



INTERDIGITAL

OpenFlow 1.5 – Ports

- OpenFlow ports are the network interfaces for passing packets between OpenFlow processing and the rest of the network
- OpenFlow packets are received on an **ingress port** and processed by the **OpenFlow pipeline** which may forward them to an **output port**
  - OpenFlow **physical ports** are switch ports that correspond to a hardware interface. OpenFlow switch may be virtualised. In those cases, an OpenFlow physical port may represent a virtual slice of the hardware interface
  - OpenFlow **logical ports** are higher level abstractions that may be defined in the switch using non-OpenFlow methods (e.g. link aggregation, tunnels) that must be transparent to OpenFlow processing
  - OpenFlow **reserved ports** specify generic forwarding actions such as sending to the controller, flooding, or forwarding using non-OpenFlow methods, such as “normal” switch processing
- **OpenFlow logical ports can be used to insert a complex and stateful processing in the switch**
  - Packet recirculation via logical ports is optional, OpenFlow supports multiple types of port recirculation
    - A packet is sent on a logical port and returns back via the same logical port (unidirectional processing)
    - A packet is sent on a logical port and returns back via another logical port. This could be used to represent tunnel endpoints (port pairs) or bidirectional packet processing
  - A switch should protect itself from packet infinite loops when using port recirculation

Link Layer Discovery Protocol – Overview

LLDP – Overview (1)

5G Crosshaul

- The Link Layer Discovery Protocol (LLDP) is a vendor-neutral link layer protocol used by network devices for advertising their identity, capabilities, and neighbors on an IEEE 802 local area network
- LLDP information is sent by devices from each of their interfaces at a fixed interval (**about seconds**), in the form of an Ethernet frame
  - Each frame contains one LLDP Data Unit (LLDPDU)
  - Each LLDPDU is a sequence of type-length-value (TLV) structures
- Information gathered with LLDP is stored in the device as a management information database (MIB) and can be queried with SNMP
- The topology of an LLDP-enabled network can be discovered by crawling the hosts and querying this database

LLDP – Overview (2)

- The LLDP destination MAC address is typically set to a special multicast address that 802.1D-compliant bridges do not forward
  - Other multicast and unicast destination addresses are permitted
  - The EtherType field is set to 0x88cc
- Each LLDP frame starts with the following mandatory TLVs:
  - Chassis ID, Port ID, and Time-to-Live.
  - The mandatory TLVs are followed by any number of optional TLVs
- The frame ends with a special TLV, named end of LLDPDU in which both the type and length fields are 0

Destination Address	Source Address	EtherType	Chassis ID TLV	Port ID TLV	Time to live TLV	Optional TLVs	End of LLDPDU TLV	
01:80:c2:00:00:0e 01:80:c2:00:00:03 01:80:c2:00:00:00	Station's address	0x88CC	Type=1	Type=2	Type=3	Zero or more complete TLVs	Type=0, Length=0	CRC / FCS

OpenState

- *"OpenState is a research effort focused in the development of a stateful data plane API for Software-Defined Networking. We propose an extension to current OpenFlow abstraction that use state machines implemented inside switches to reduce the need to rely on remote controllers."*
- Stateful operations implemented only at the receiver switch
- No mention about OAM and packet generation/timestamping
- To know more about OpenState, refer to the paper:
  - G. Bianchi, M. Bonola, A. Capone, and C. Cascone, "OpenState: Programming Platform-independent Stateful OpenFlow Applications Inside the Switch", ACM SIGCOMM Computer Communication Review, vol. 44, no. 2, pp. 44–51, 2014.