Layer 2 Openflow

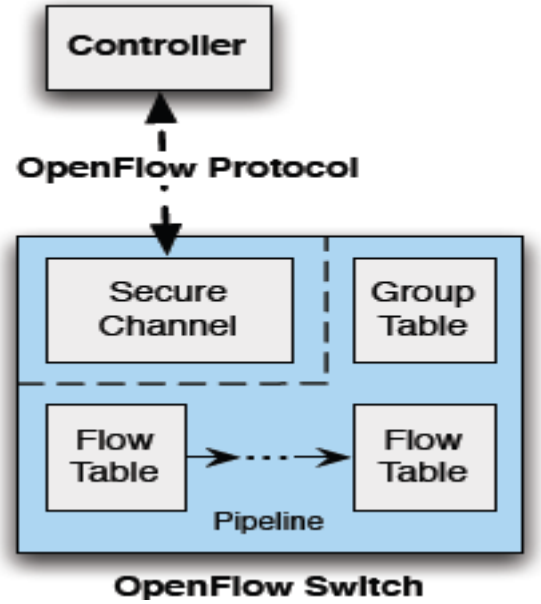
Authors: **Wenle Yang; Ruobin Zheng, Hesham ElBakoury**
Version: 1.0

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Introduction to OpenFlow

openflow-spec-v1.3.3



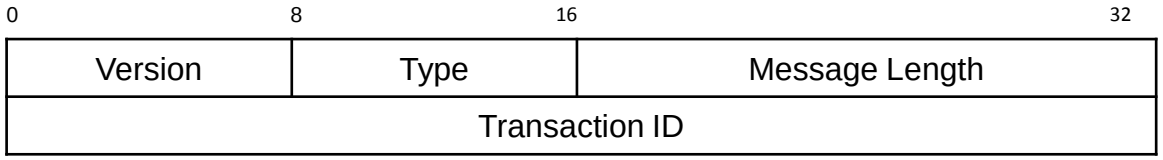
- OpenFlow Channel
  - connects each OpenFlow Switch to an OpenFlow controller
  - is used to exchange OpenFlow message between an OpenFlow switch and an OpenFlow controller
  - uses TLS or plain TCP

OpenFlow Protocol

- **Three message types**
  - Controller-to-Switch
    - **initiated by the controller**
    - **used to directly manage or inspect the state of the switch**
  - Asynchronous
    - **initiated by the switch**
    - **used to update the controller of network events and changes to the switch state**
  - Symmetric
    - **initiated by either the switch or the controller.**
    - **Sent without solicitation**

OpenFlow Message Format

Header on all OpenFlow packets

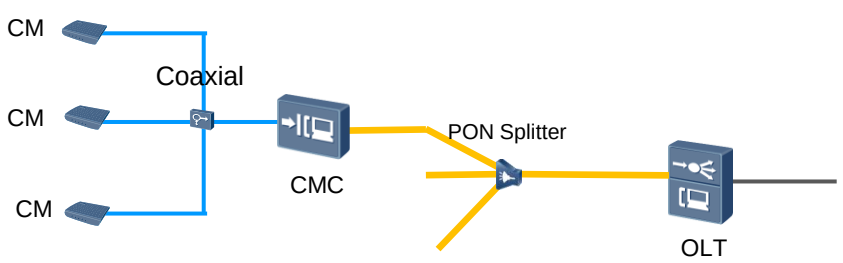


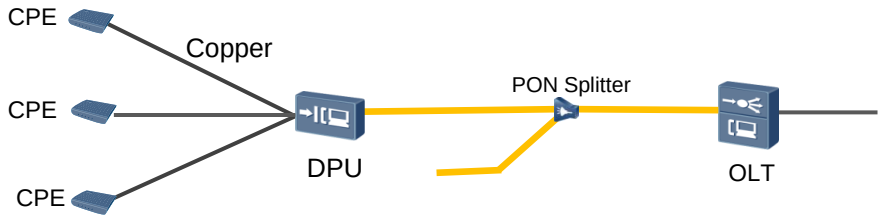
Payload is defined for each type of message

Modify Flow Entry (Controller-to-Switch Message)

```
/* Flow setup and teardown (controller -> datapath). */
struct ofp_flow_mod {
    struct ofp_header header;
    uint64_t cookie;        /* Opaque controller-issued identifier. */
    uint64_t cookie_mask;   /* Mask used to restrict the cookie bits
                               that must match when the command is
                               OFPFC_MODIFY* or OFPFC_DELETE*. A value
                               of 0 indicates no restriction. */
    /* Flow actions. */
    uint8_t table_id;       /* ID of the table to put the flow in.
                               For OFPFC_DELETE_* commands, OFPTT_ALL
                               can also be used to delete matching
                               flows from all tables. */
    uint8_t command;        /* One of OFPFC_*. */
    uint16_t idle_timeout;  /* Idle time before discarding (seconds). */
    uint16_t hard_timeout;  /* Max time before discarding (seconds). */
    uint16_t priority;      /* Priority level of flow entry. */
    uint32_t buffer_id;     /* Buffered packet to apply to, or
                               OFP_NO_BUFFER.
                               Not meaningful for OFPFC_DELETE*. */
    uint32_t out_port;      /* For OFPFC_DELETE* commands, require
                               matching entries to include this as an
                               output port. A value of OFPP_ANY
                               indicates no restriction. */
    uint32_t out_group;     /* For OFPFC_DELETE* commands, require
                               matching entries to include this as an
                               output group. A value of OFPG_ANY
                               indicates no restriction. */
    uint16_t flags;         /* One of OFPFF_*. */
    uint8_t pad[2];
    struct ofp_match match; /* Fields to match. Variable size. */
    //struct ofp_instruction instructions[0]; /* Instruction set */
};
OFP_ASSERT(sizeof(struct ofp_flow_mod) == 56);
```

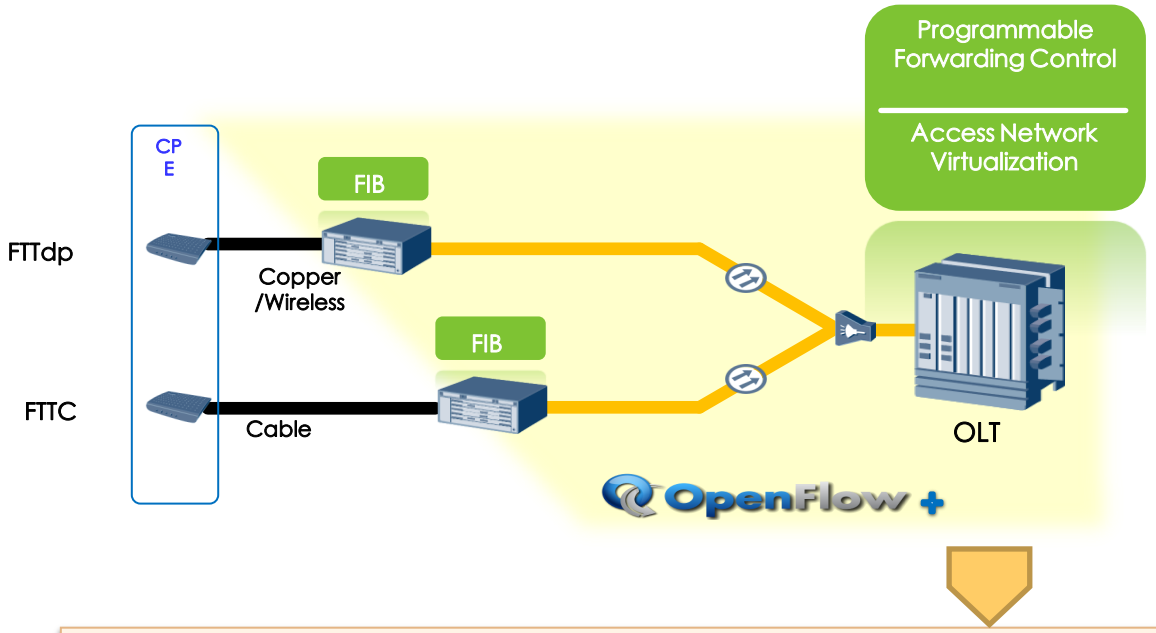
Current Challenges for Remote Nodes Management





- Distributed solution challenges the management of remote nodes:
  - FTTDp/EPoC FCU/Remote CMTS/Remote CCAP brings 10-100 times more nodes
- Remote node supports multiple-services
  - residents, enterprises, mobile backhaul, and wholesale services.
- Remote nodes are expected to **be simple to manage**

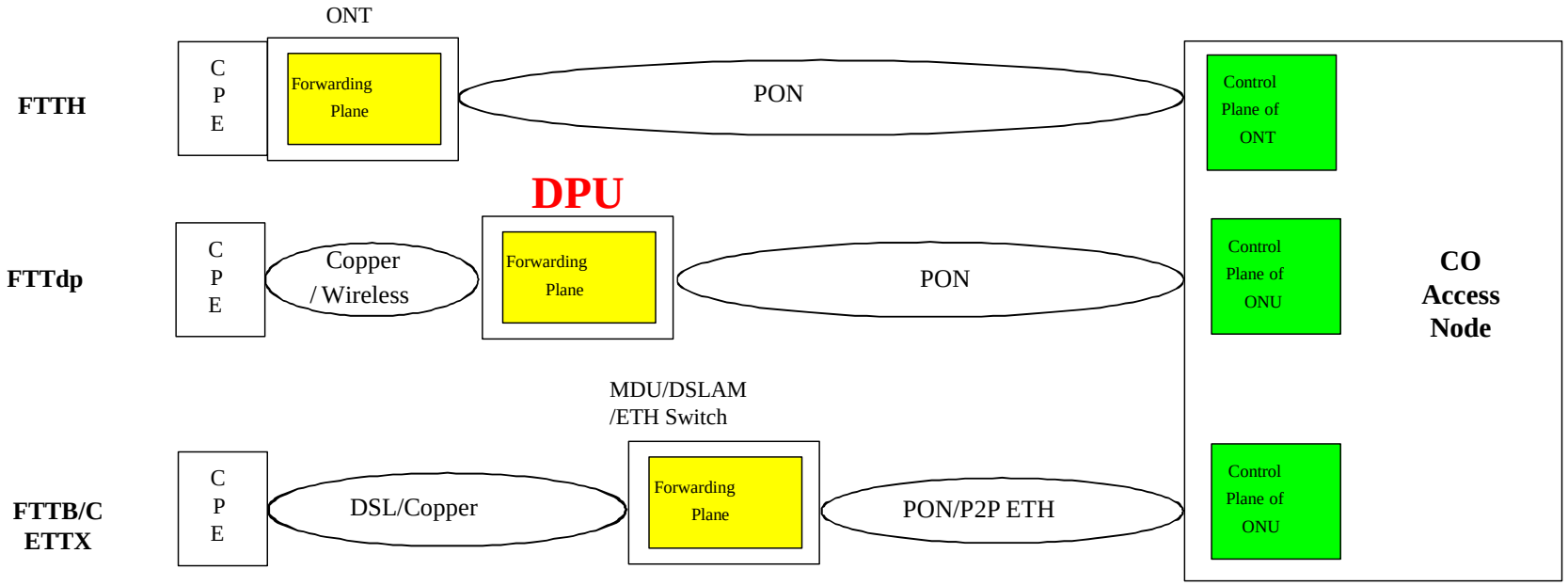
Software Defined Access Network & AN Virtualization



- Based on the separation of the forwarding and control planes , the control plane for remote nodes are relocated and centralized to OLT.
- OLT and its remote nodes are virtualized as one access node with one management IP address.

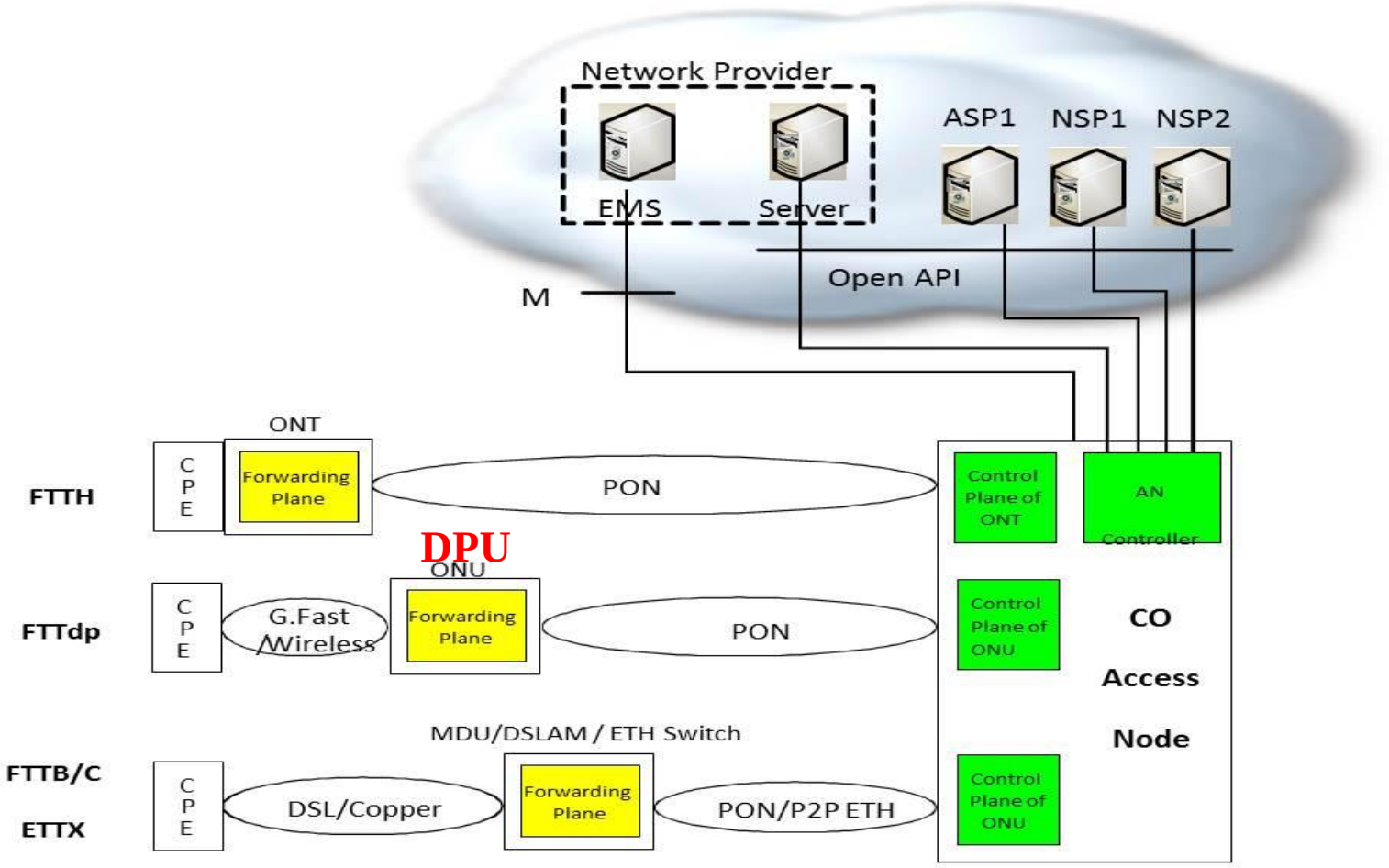
- **Benefits**
  - Remote nodes are simplified to programmable devices.
  - Decouple access devices from services.
  - Simplified O&M
  - No IP address is needed to configure remote nodes.

Use Case Defined by BBF SD-313: Remote Nodes Plug & Play in Access Networks

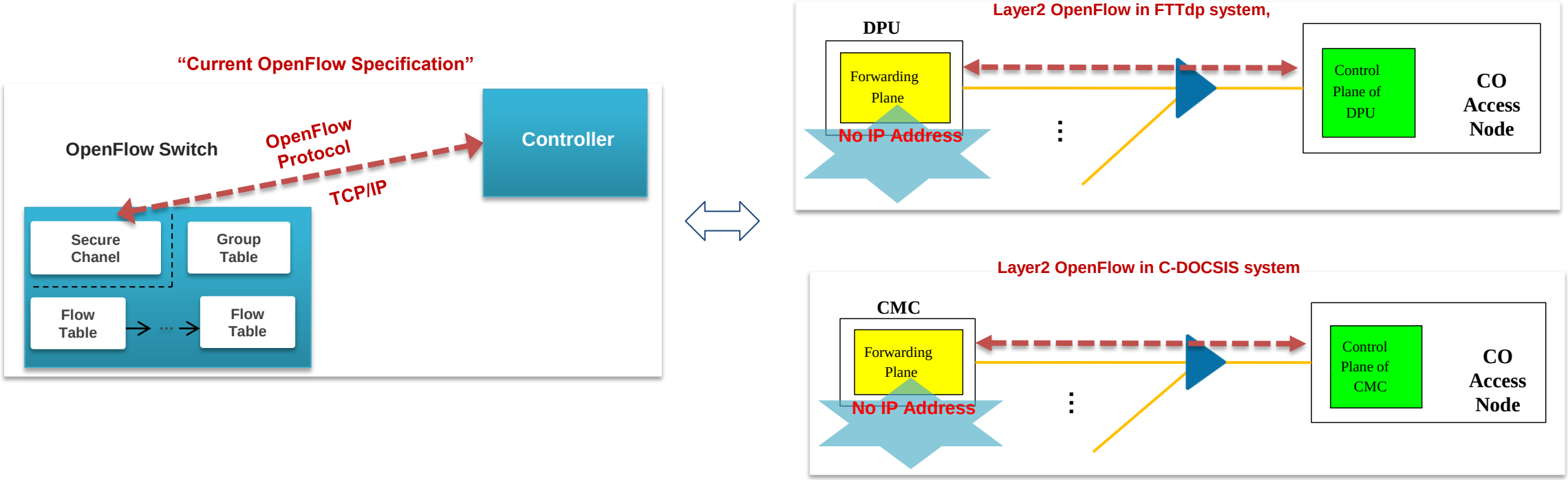


- **Control planes of remote nodes are moved to the aggregation OLT, while remote nodes implements forwarding plane.**

Use Case Defined by ETSI NFV: Access Network Virtualization



Why Layer 2 OpenFlow is Needed ?

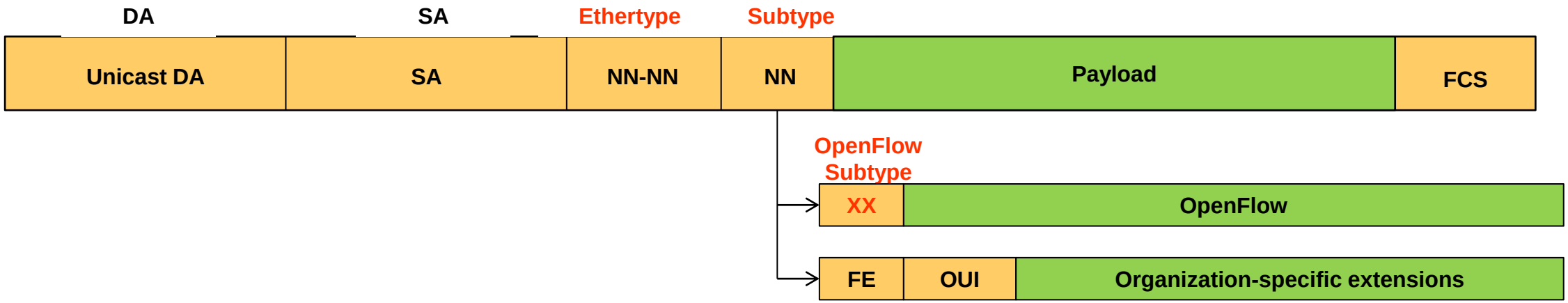


- In the current OpenFlow specification, OpenFlow channel is encrypted using TLS or run directly over TCP, which is not suitable for the access network where DPUs and CMCs work on layer 2 without IP addresses.
- Layer2 OpenFlow allows OpenFlow messages to be exchanged over layer2 network between Controller and DPUs or CMCs, to support programmability of these devices.

Layer 2 OpenFlow

- **Allow OpenFlow to run over layer2 connections in access network.**
  - OpenFlow over OMCI for GPON system,
  - Openflow over Ethernet OAM
  - IEEE1904.2 management channel for EPON or P2P Ethernet.
- **Openflow specification has been rapidly growing over the past five years.**
  - Extending OAM or OMCI to support L2 Openflow may not be a pragmatic approach.
  - We recommend using IEEE 1904.2 management channel to provide transport for L2 Openflow messages.

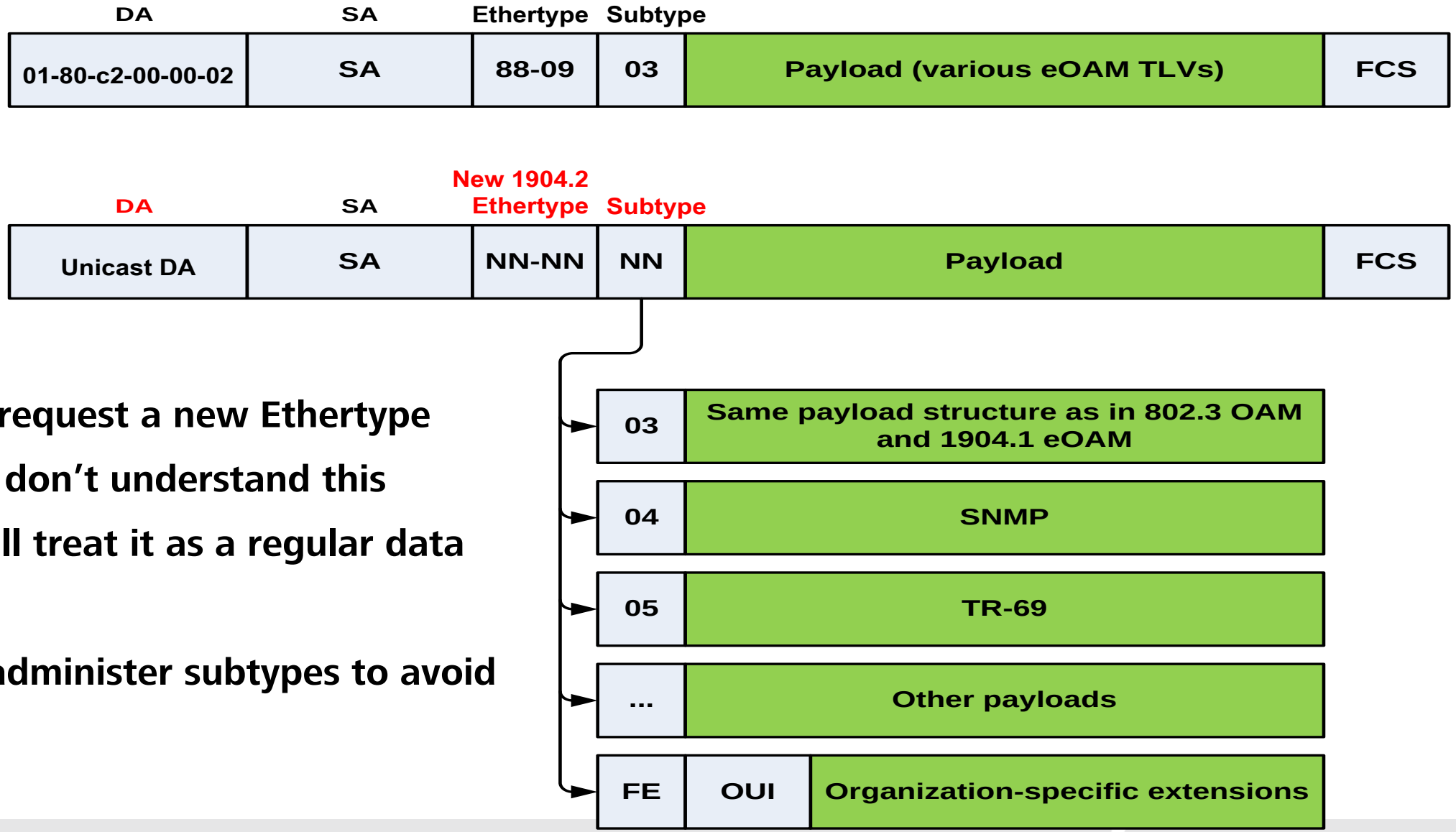
Using IEEE1904.2 to Support Layer 2 OpenFlow



- Two possible approaches to support OpenFlow:
  - Define a new Subtype for OpenFlow protocol
  - Use OUI (Organization-specific extensions)
- **We recommend the 1st approach.**

THANK YOU

Possible approach for IEEE 1904.2



- 1904.2 may request a new Ethertype
- Devices that don't understand this Ethertype will treat it as a regular data frame
- 1904.2 will administer subtypes to avoid conflicts