NTWRK SLICING:

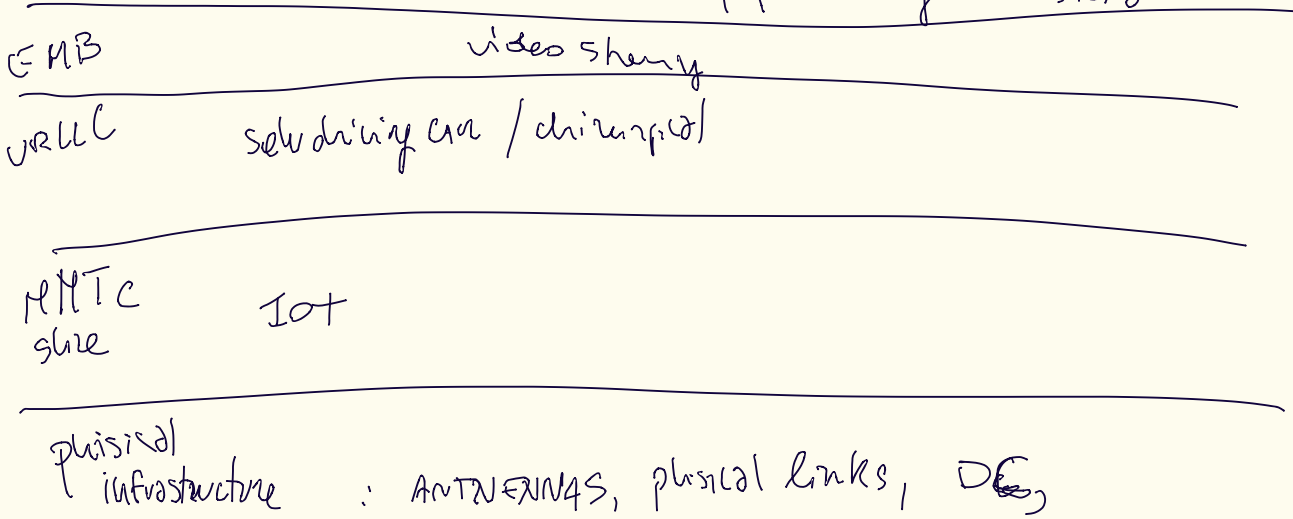
- Importante nelle Mobile NETWORKS → evoluzione al 5G => requisiti + complessi
  considera requirements

1) Massive Machine type Communications: IOT distributed, perdere dati, comunicazione, controllare... SAVE BATTERY
2) Comunicazione affidabile e low-latency: low RESPONSE TIME (operazioni chirurgiche) (Ultra Reliable low latency comunication)
3) AUMENTARE MOBILE BROADBAND: video, streaming traffic

=> Bisogna ridisegnare l'architettura di rete per soddisfare

NETWORK SLICING

- Da un'architettura fisica => creare diverse architetture logiche al disopra di quella fisica

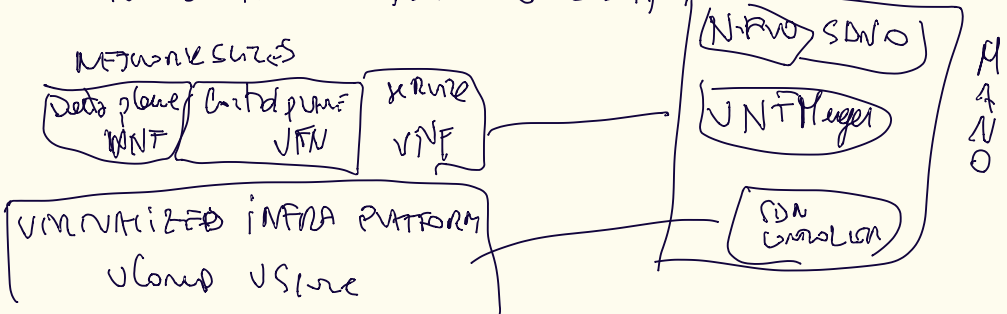


use NFV to create virtual resources to VNF?

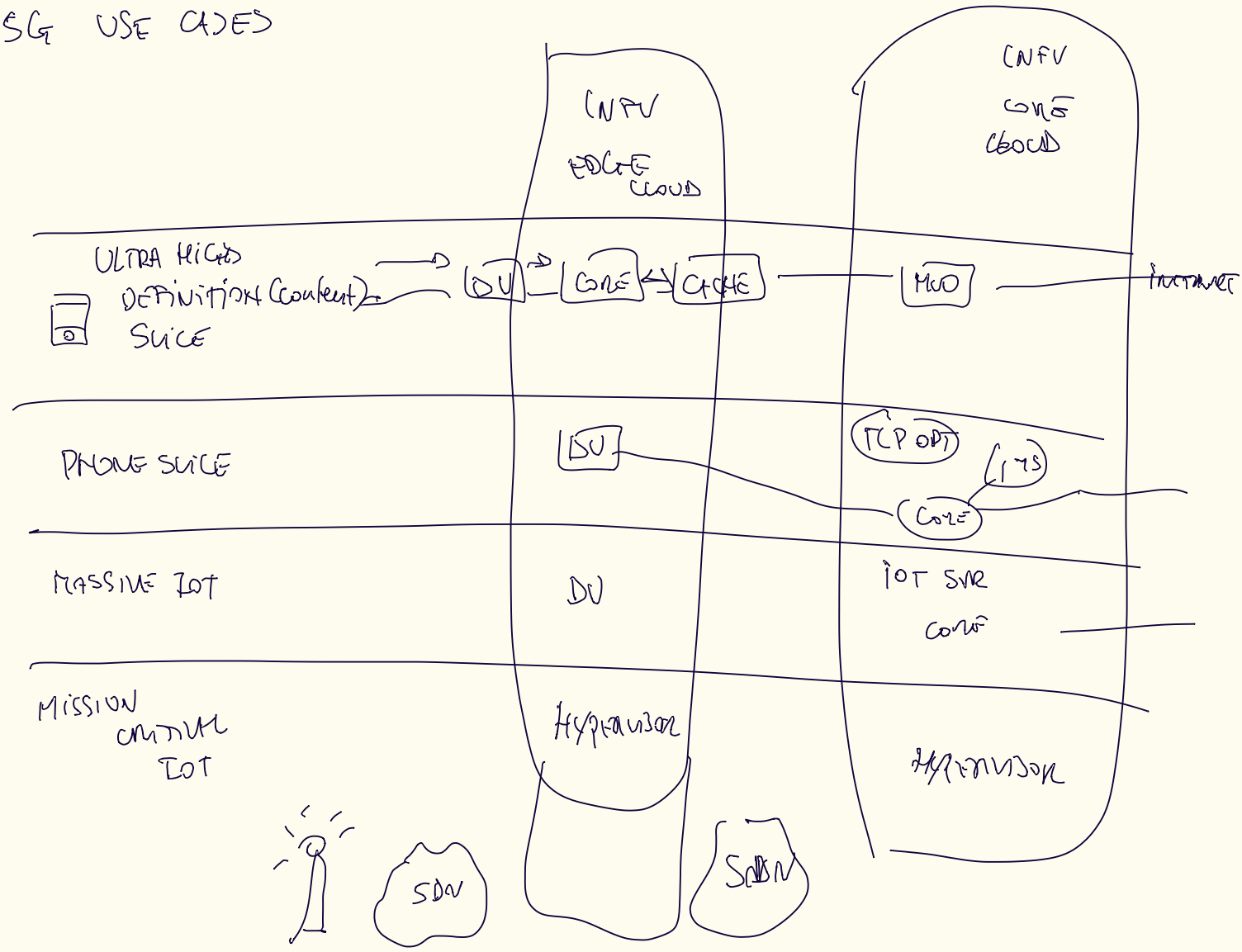
NETWORK SLICE: VIRTUAL END-TO-END NETWORK BUILD on top of a physical network

-> each network slice: ≠ Requirements
≠ functionalities
≠ QoS

KEY TECNOLOGIE TO REALIZE IT: NFV & SDN



5G USE CASES



- CON SDN & NFV => posso creare + network slices

- in UHD slice content can have + VNF like Distributed Unit: located part of RAN, can be close to physical antenna within a DC

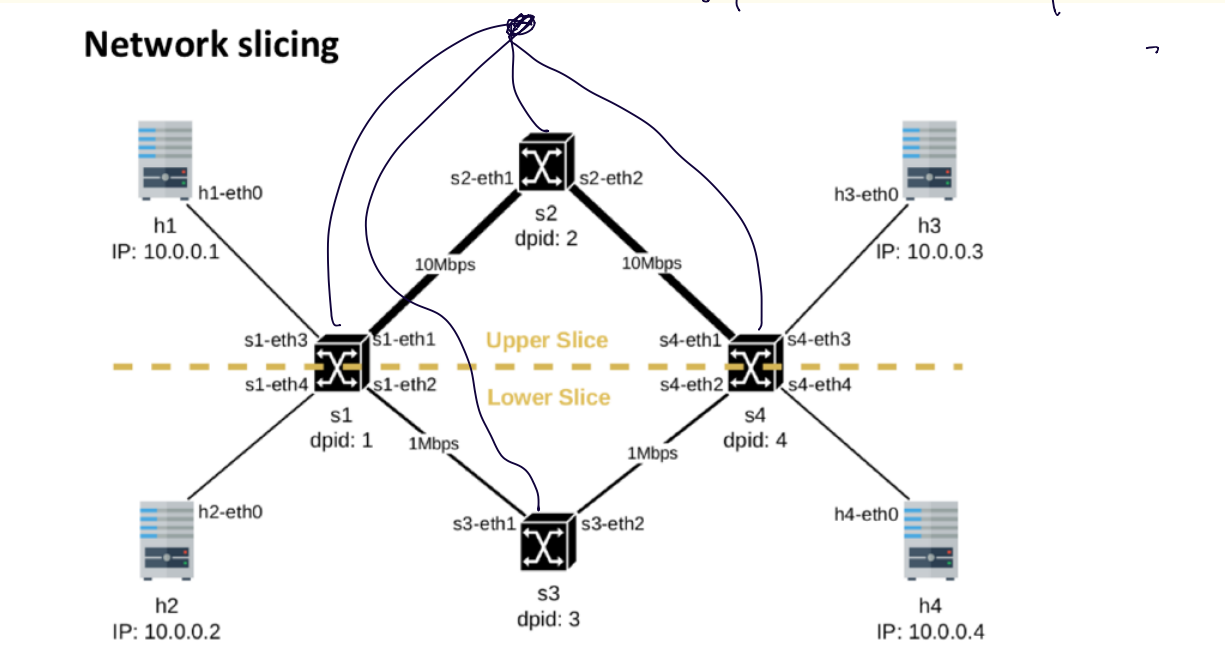
=> can move CORE FUNCTIONS (close to the edge)

=> 1 edge cloud, 1 core cloud

=> DC close to the RAN (antennas) and or remote

3 of DC ('in CORE, remote) netw of network provider

CONTROLLER: provides control plane



- In normal topology: all hosts could communicate the switches to other hosts

=> we have to 2 slices on this topology, isolate the resources to have the slice

- UPPER SLICE: $h_1 \rightarrow S_1 \rightarrow S_2 \rightarrow S_4 \rightarrow h_3$ 10 Mb/s
- Lower Slice: $h_2 \rightarrow S_1 \rightarrow S_3 \rightarrow S_4 \rightarrow h_4$

TOPOLOGY SLICING: .py => istruzioni al controller per regole gli eventi:

```
def __init__(self, *args, **kwargs):
    super(TrafficSlicing, self).__init__(*args, **kwargs)

    # out_port = slice_to_port[dpid][in_port]
    self.slice_to_port = {
        1: {1: 3, 3: 1, 2: 4, 4: 2},
        4: {1: 3, 3: 1, 2: 4, 4: 2},
        2: {1: 2, 2: 1},
        3: {1: 2, 2: 1},
    }

@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    msg = ev.msg
    datapath = msg.datapath
    in_port = msg.match["in_port"]
    dpid = datapath.id
    out_port = self.slice_to_port[dpid][in_port]

    actions = [datapath.ofproto_parser.OFPActionOutput(out_port)]
    match = datapath.ofproto_parser.OFPMatch(in_port=in_port)
    # add_flow(self, datapath, priority, match, actions)
    self.add_flow(datapath, 1, match, actions)
    self._send_package(msg, datapath, in_port, actions)
```

Switch

S1 riceve PACCHETTO DATA PORTA 3 & lo MANDA VIA DALLA PORTA 1

1) packet_in:
- Switch will send to Controller packets they don't have rules for them
- instruct the controller to perform the slicing for packet in events

here is defined how e packet IN should managed by switch

=> here the switch enforce the slice: take message within packet in, take input port where this msg is taken, provide output port

- Con mininet, prima abbiamo creato la topologia, host switch e links

2) SWITCH FEATURE reply msg to controller:
- at startup they exchange feature req / resp msg
=> its completed by controller say: "send the packet you cannot handle to me via packet in

```python
@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
def switch_features_handler(self, ev):
    datapath = ev.msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    # install the table-miss flow entry.
    match = parser.OFPMatch()
    actions = [
        parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
        ofproto.OFPCML_NO_BUFFER)]
    self.add_flow(datapath, 0, match, actions)

def add_flow(self, datapath, priority, match, actions):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    # construct flow_mod message and send it.
    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
        actions)]
    mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
          match=match, instructions=inst)
    datapath.send_msg(mod)
```

I'm registering the application to this event OFPSwitchFeature

is feature reply message

elements to instruct the Controller

=> Fields of the flow we want to INSTALL:

MATCH: any packet for you don't have entries

ACTION: send to controller

method add_flow that prepare a FLOW-MOD message to be send to C -> S

=> MOD FLOW MODE MESSAGE

If controller receive come datapath to switch & in input port

priority, match-field: IN-PORT or other field

EFFECTED BY the rule

INSTRUCTIONS: in this case => send to the controller

=> all this for the INITIAL EVENT: any packet send to controller

=> incapsulate a packet in packet IN option msg

(A) DS: allowing to have the pair of PORTS COUPLED

=> ∀ input port we find the output port & prepare actions for the switch

& prepare flow with priority = 1 (A)

(F) then send up to the switch