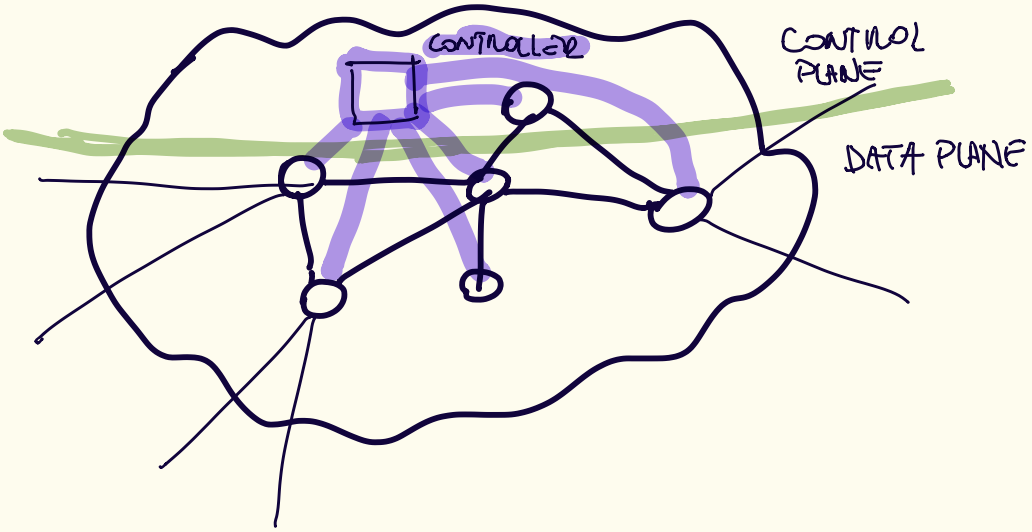
SDN: software defined networking



Data plane: forwarding traffic

Control plane: compute STATE of routers that allows to make decisions and forward traffic to the destination: state = routing tables
Network control plane computes the ROUTING TABLE

- In today's networking: Data & Control planes run on routers → distributed in do network
- In SDN: CONTROL PLANE runs in a LOGICALLY CENTRALIZED CONTROLLER
  - (1) controls routers across the network
  - (2) control program exerts control over the routers in the network
    - ↳ Facilitates NETWORK-WISE CONTROL

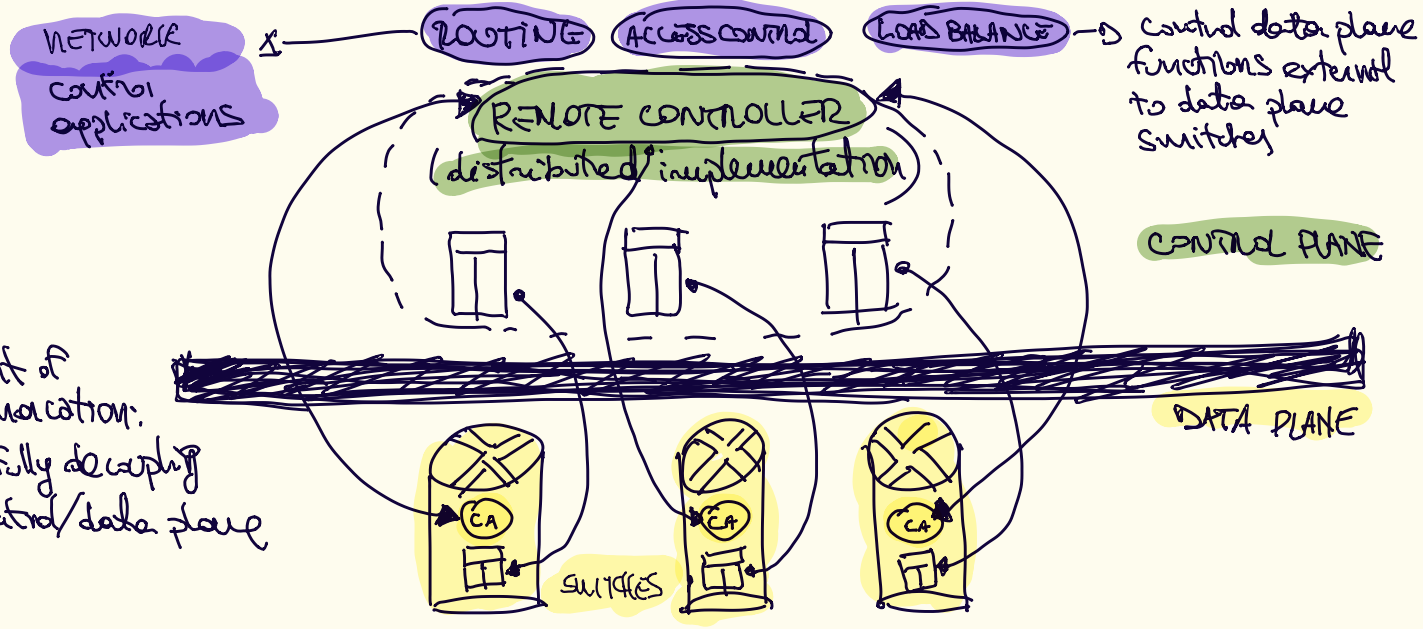
=> - separation of control plane & data plane
=> allows to build a network with COMMODITY DEVICES where control resides in a SEPARATE control programs

=> this refactoring allow to move from a network where devices are VERTICALLY integrated to a network where devices have OPEN INTERFACES that can be controlled by SW

- **Data plane** => forwarding: moving packets from router's input port to routers' output port, according to FT.
- **Control plane** => compute FT: determine route taken by packet from source to destination

↓

- Classic: per-router control plane: individual routing algo in each router interact in control plane to compute routing tables.
- SDN: logically centralized control plane: remote controller computes/installs FT in routers



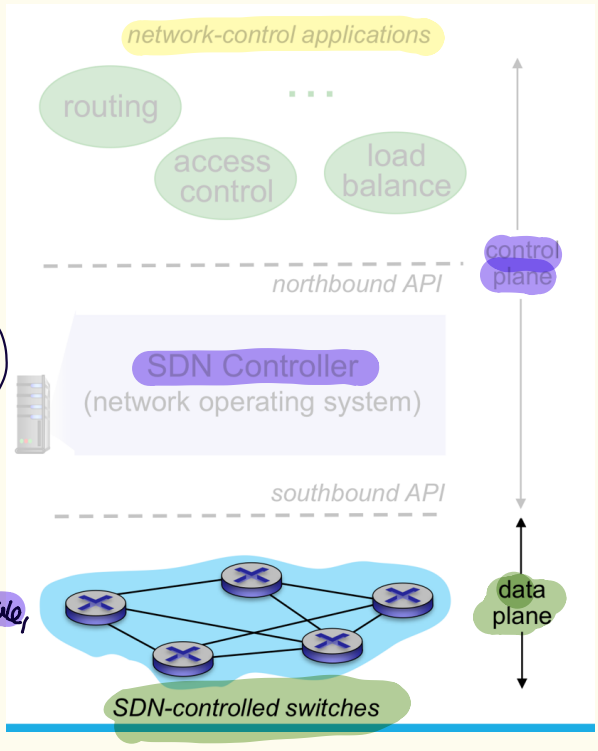
- Remote controllers and replicas to manage false positives
- Dedicated devices (servers) to compute FT
- Receive a packet, look into FT, deliver to destination
- Remote controller collects infos to get NETWORK STATE, sending instructions to routers
- Routers talk to RC, infos on paths/links/statistics

CONTROL PLANE

☐ Why at logically CONTROL PLANE?

1) Easier NTWORK management, traffic flows flexibility
2) More easy defining applications to handle the traffic
3) Possibility to set different routing strategies
4) CENTRAL CONCEPT: table-based forwarding (OPENFLOW API) allows PROGRAMMING ROUTERS:
   - centralised programming easier: compute tables centrally and distribute
   - distributed programming harder: compute tables as result of distributed algo implemented in each and every router

1) SDN controlled switches: DATA PLANE
   1) Switches implementing data-plane forwarding in hardware
   2) Flow table computed, installed by controller
   3) API for table-based switch control (OpenFlow)
      - receive FT by controller and exploit API so table can be updated
   4) Protocol communicating with controller (SOUTHBOUND API)
2) SDN CONTROLLER (NETWORK OS)
   1) Builds topology on underlying level
   2) Decides routes (FT)
   2) Maintain NTWORK STATE informations
   3) Implemented as distributed systems for performance, scalability, robustness, fault-tolerance
   4) Interacts with NTWORK-CONTROL APPS via northbound API and with SDN-CONTROLLED SWITCHES via southbound API



3) NETWORK-CONTROL APPLICATIONS

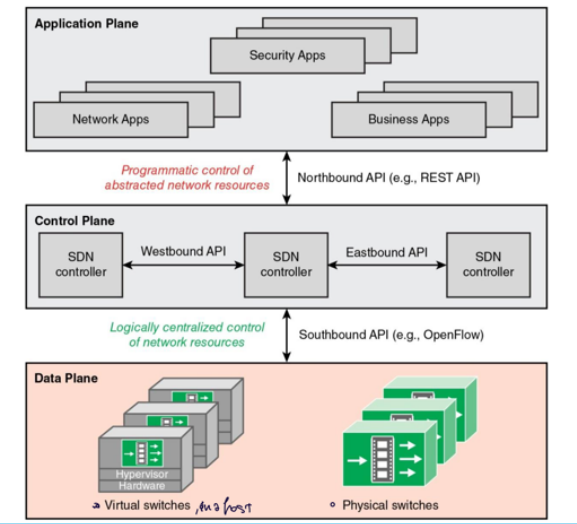
1) Brains of control:
- implements control functions using lower-level services (API provided by controller)
- northbound API: high level of abstraction than southbound API

2) "UNBOUNDED": can be provided by 3rd-parts: distinct from router vendors or SDN Controller

---

DATA PLANE

- FORWARDING ABSTRACTION
- Resource or infrastructure layer made by forwarding devices (forwarding switches)
  - perform transport and processing of data according to DECISIONS MADE BY THE SDN CONTROL PLANE
  - without embedded SW to make autonomous decisions
- functions



1) DATA FORWARDING: look in the FT to match the traffic flow along data forwarding paths, computed and established according to rules by SDN apps.

2) CONTROL SUPPORT:
- interactions with SDN controller for managing forwarding rules
- openflow switch control: module to interact with controller

GENERALIZED FORWARDING: MATCH + ACTIONS

- "match + action" abstraction: performed on packets, match bits in arriving packets, take actions:

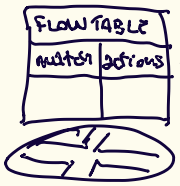
1) DESTINATION-BASED FORWARDING: forward based on IP-address

2) GENERALIZED FORWARDING:
- many header fields can determine actions (not necessary IP/PORT match, other fields, like MAC address)
- Many actions possible: DROP / COPY / MODIFY / LOG packets (different actions in FT)

FLOW TABLE ABSTRACTION

1) FLOW: defined by header fields values
- Router's FT defines router's match + action rules

Fields:



- should keep FT short as possible

src = *.*.*.*, dest=3.4.*.*	forward(2)
src=1.2.*.*, dest=*.*.*.*	drop
src=10.1.2.3, dest=*.*.*.*	send to controller

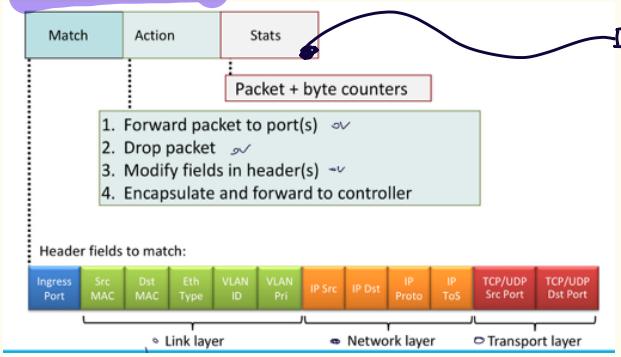
(priority)

- if you don't have a specific behaviour, switches send packet to CONTROLLER, it will decide actions on that, sending back to the switch instructions to perform

2) GENERALIZED FORWARDING: simple packet handling rules

1) MATCH: pattern values in packet header fields
2) ACTIONS: for matched packets: drop, forward, modify matched packet or send matched packets to controller that tells actions to be performed on that packet
3) PRIORITY: disambiguate overlapping patterns: can have more matches, wins higher priority
4) COUNTERS: #bytes and # of packets for statistics

OPENFLOW: FLOW TABLE ENTRIES



→ relevant to controller, patterns for a given traffic flow, recompute FT...

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	*	*	*	*	22	drop

Block (do not forward) all datagrams destined to TCP port 22 (ssh port #)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	drop

Block (do not forward) all datagrams sent by host 128.119.1.1 (suspicious host)

- CVB specify dest address to forward at port x

- BEHAVIOR OF A SWITCH:

Layer 2 destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	*	port3

layer 2 frames with destination MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3

OPENFLOW ABSTRACTIONS: the implemented behaviors

Router
- *match:* longest destination IP prefix
- *action:* forward out a link

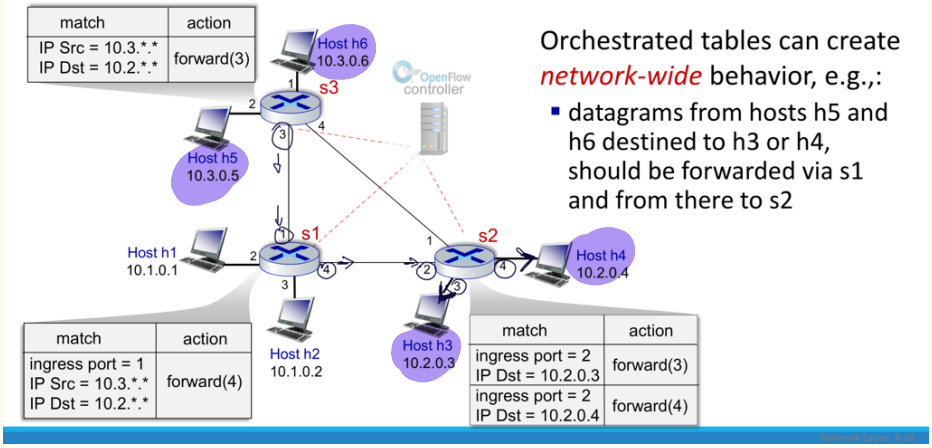
Switch
- *match:* destination MAC address
- *action:* forward or flood (to port / more ports)

Firewall
- *match:* IP addresses and TCP/UDP port numbers
- *action:* permit or deny

NAT
- *match:* IP address and port
- *action:* rewrite address and port

→ device can modify the header
- 1st: match to a given private IP/PORT
- 2nd: rewrite/substitute IP address with public one

OPENFLOW EXAMPLE



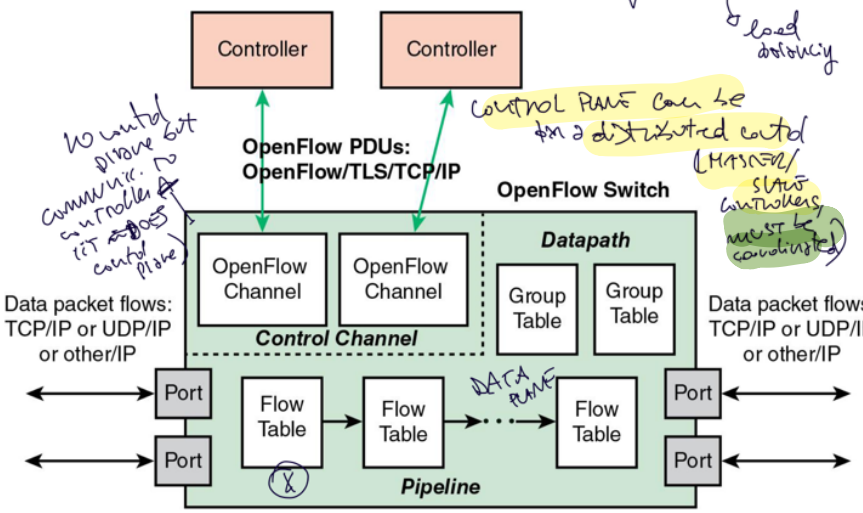
Orchestrated tables can create *network-wide* behavior, e.g.,:
- datagrams from hosts h5 and h6 destined to h3 or h4, should be forwarded via s1 and from there to s2

- CONTROLLER orchestrates rules to implement the behavior of the network topology

SUMMARY ON GENERALIZED FORWARDING

- "match plus action" abstraction: match bits in arriving packet header(s) in any layers, take action
  - matching over many fields (link-, network-, transport-layer)
  - local actions: drop, forward, modify, or send matched packet to controller
  - "program" *network-wide* behaviors
- simple form of "network programmability"
  - programmable, per-packet "processing"
  - *historical roots:* active networking
  - *today:* more generalized programming:
    P4 (see p4.org):

OpenFlow Switch



OpenFlow channel: through OpenFlow, the controller can add, update, and delete flow entries in tables, both reactively (in response to packets) and proactively.

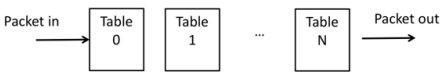
Tables for packet flow processing, i.e. forwarding

multiple I/O ports for packet flows into and out of the device.

Flow Table Pipeline

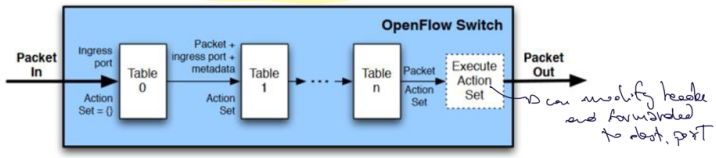
- A switch includes one or more flow tables
- If there is more than one flow table, they are organized as a pipeline
  - The flow tables of an OpenFlow switch are sequentially numbered, starting at 0.
  - The packet is first matched against flow entries of flow table 0. Other flow tables may be used depending on the outcome of the match in the first table.

The use of multiple tables in a pipeline, rather than a single flow table, provides the SDN controller with considerable flexibility



Processing pipeline

- When processed by a flow table, the packet is matched against the flow entries of the flow table.
- If there is a matching, any instructions associated with this entry is executed. This may include updating the action set, updating the metadata value, and performing actions.
- those instructions may explicitly direct the packet to another flow table (Goto Instruction), where the same process is repeated again, to the group table, to the meter table, or, finally, **to an output port for forwarding**



Processing pipeline

If there is a match on one or more entries in a table the match is defined to be with the **highest-priority** matching entry

If there is a match only on a table-miss entry, the table entry may contain instructions, as with any other entry. In practice, the table-miss entry specifies one of three actions:

a) **Send packet to controller**. This will enable the controller to define a new flow for this and similar packets, or decide to drop the packet.

b) **Direct packet to another flow table** further down the pipeline.

c) **Drop the packet**.

If there is no match on any entry and there is no table-miss entry, the packet is dropped.