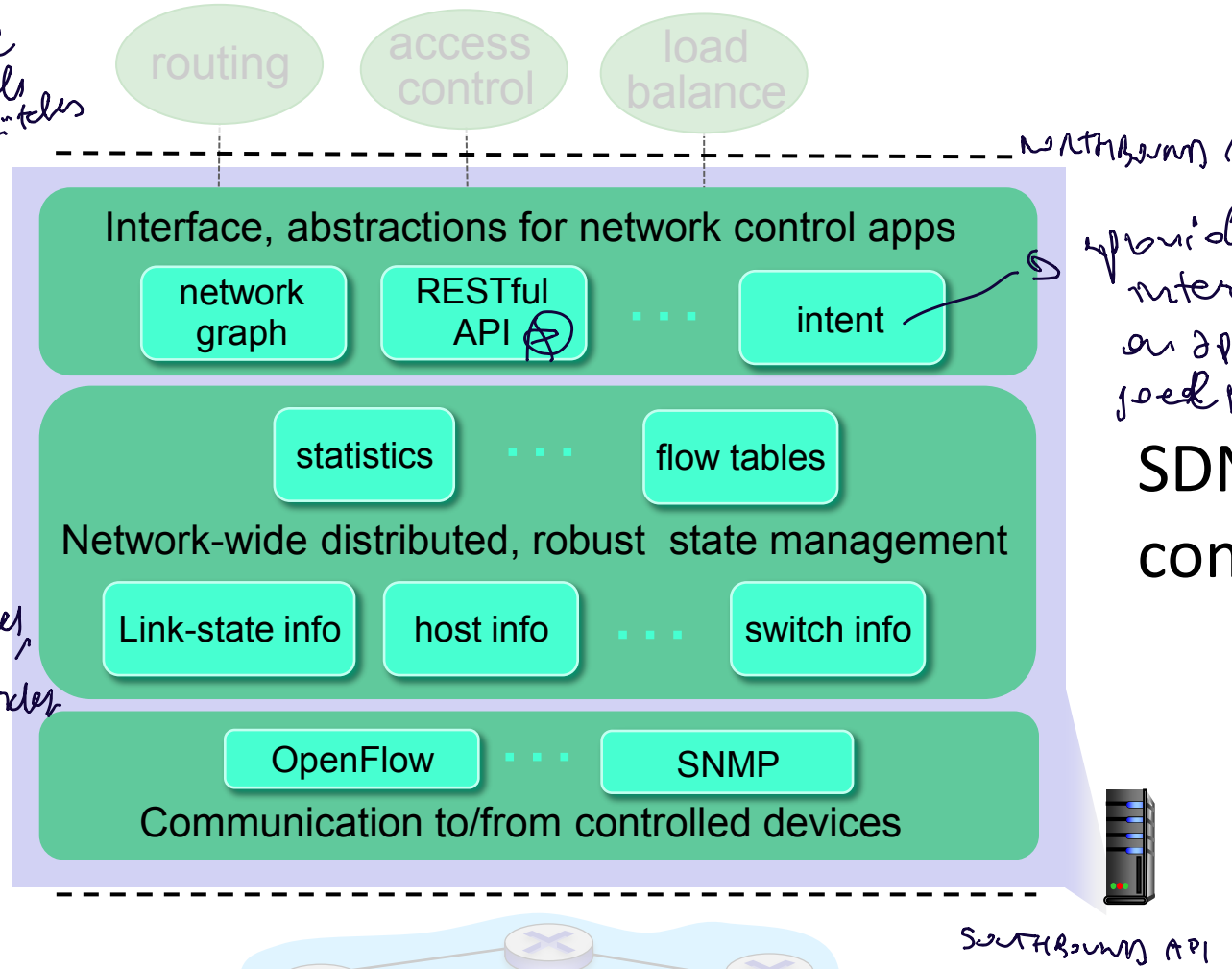
SDN Control Plane

Components of SDN controller



(1) interface layer to network control apps: abstractions API

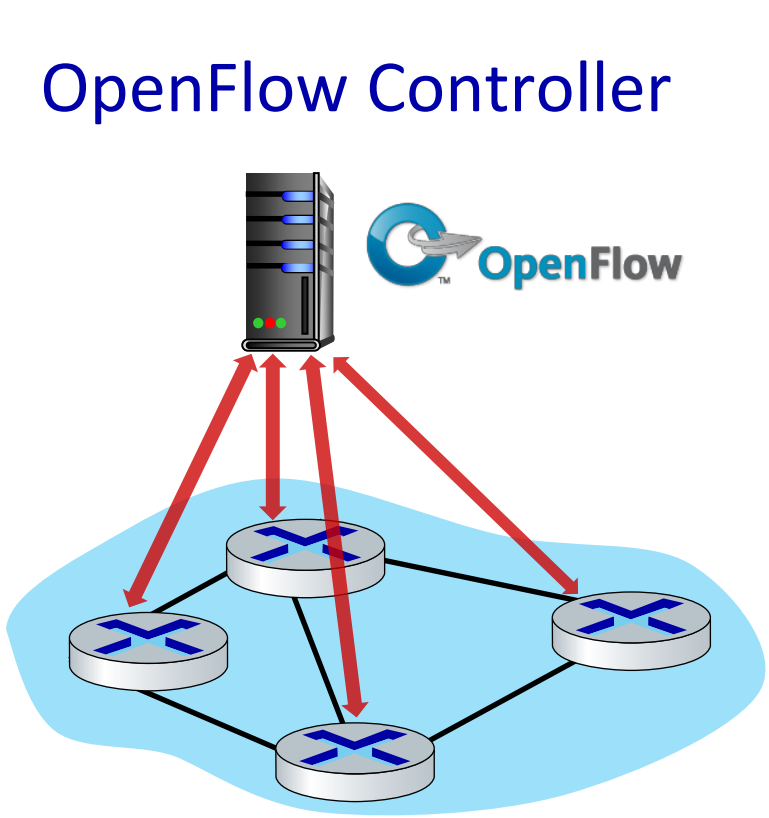
(2) network-wide state management : state of networks links, switches, services: a *distributed database*

(3) *communication*: communicate between SDN controller and controlled switches

SDN controller

OpenFlow protocol

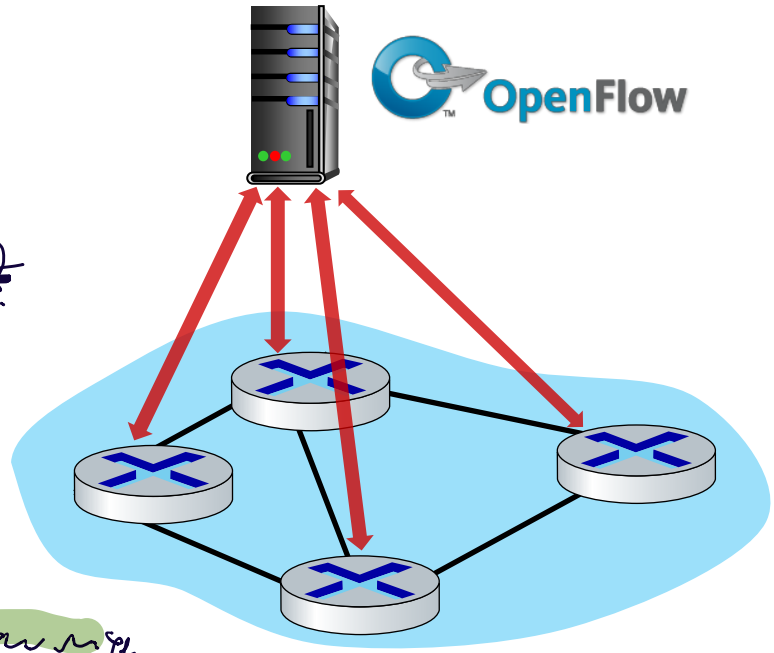
- operates between controller/switch
- TCP used to exchange messages
  - optional encryption (TLS)
- three classes of OpenFlow messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc.)
- distinct from OpenFlow API
  - API used to specify generalized forwarding actions



OpenFlow: controller-to-switch messages

Key controller-to-switch messages

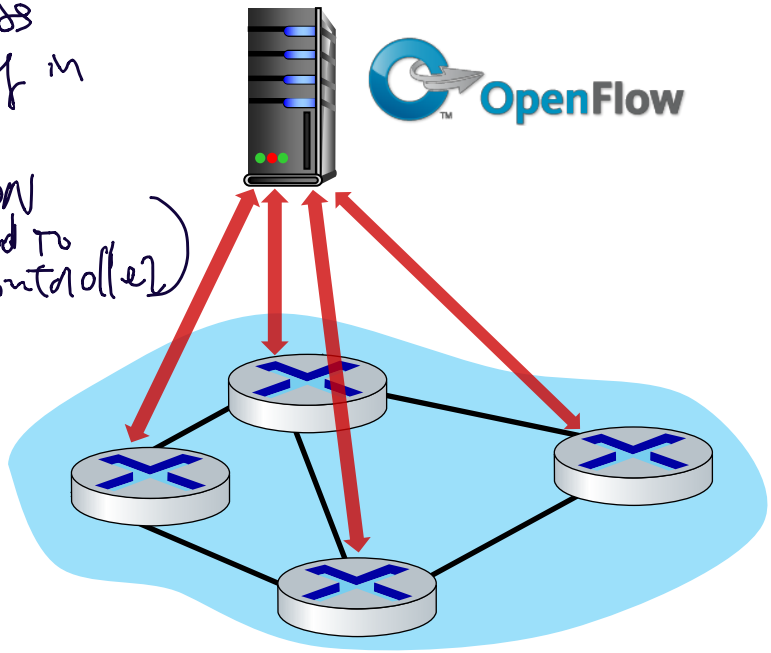
- ***features:*** controller queries switch features, switch replies
- ***configure:*** controller queries/sets switch configuration parameters
- ***modify-state (FlowMod):*** add, delete, modify flow entries in the OpenFlow tables
- ***packet-out:*** controller can send this packet out of specific switch port



OpenFlow: switch-to-controller messages

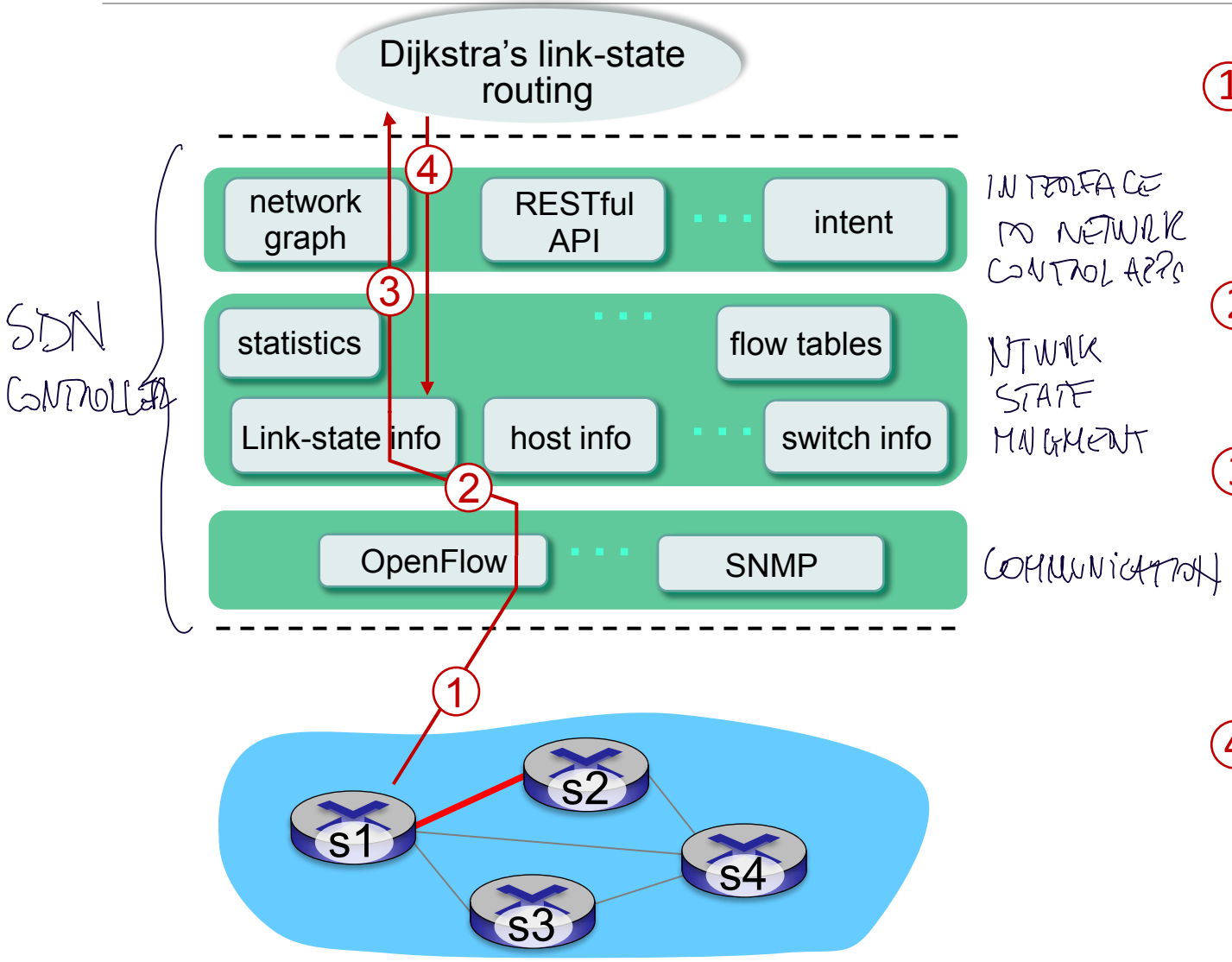
Key switch-to-controller messages (ASYNC)

- *packet-in:* transfer packet (and its control) to controller. See packet-out message from controller (NO PACKET FLOW MATCH || SPECIFIC ACTION (TO send TO controller))
- *flow-removed:* flow table entry deleted at switch (if not matches for a time and informs the controller)
- *port status:* inform controller of a change on a port. (STATE OF A PORT)



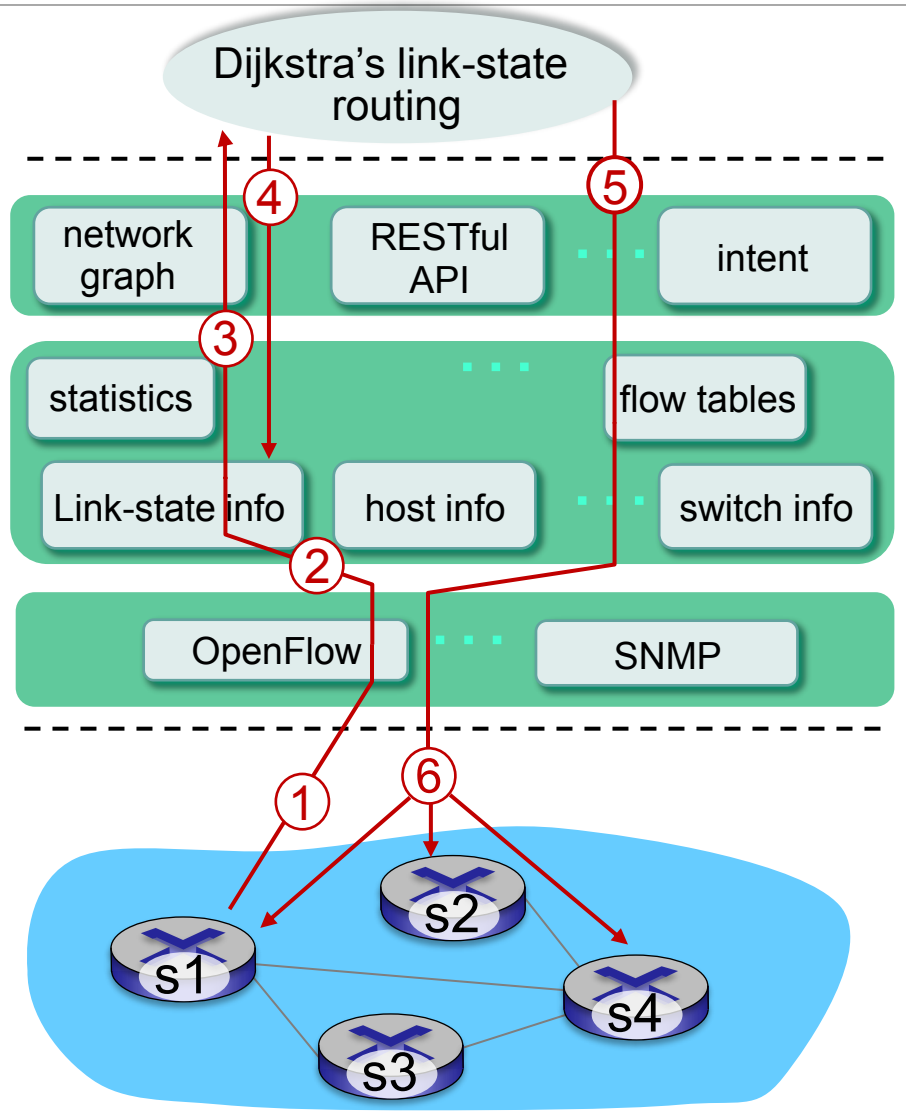
Fortunately, network operators don't "program" switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller

SDN: control/data plane interaction example



1. S1, experiencing link failure uses OpenFlow **port status** message to notify controller
2. SDN controller receives OpenFlow message, updates link status info
3. Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.
4. Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes

SDN: control/data plane interaction example



⑤ link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed

⑥ controller uses OpenFlow to install new tables in switches that need updating

SDN Controller: implementations

- **NOX**:
  - first SDN Controller
  - https://github.com/noxrepo/nox

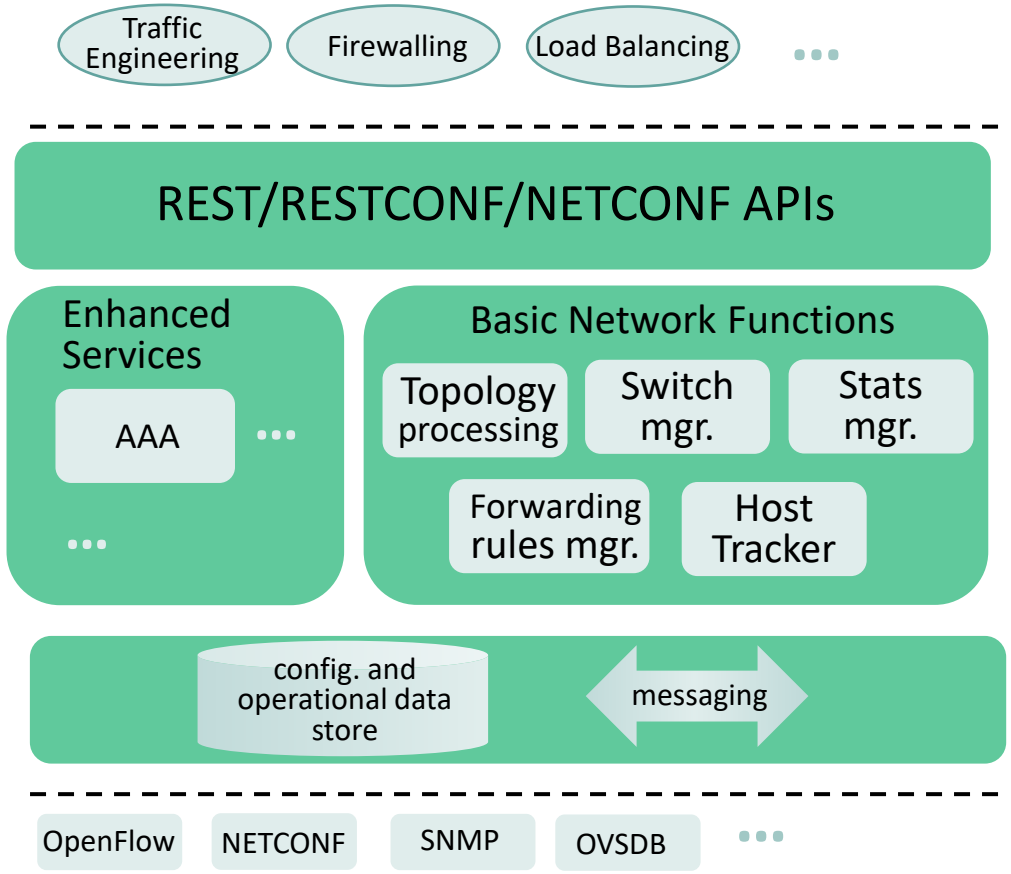
- **Ryu**
  - https://ryu-sdn.org/

More recent Open source SDN Controllers

- **OpenDayLight (ODL)**
  - https://www.opendaylight.org/

- **Open Network Operating System (ONOS)**
  - https://opennetworking.org/onos/

OpenDaylight (ODL) controller

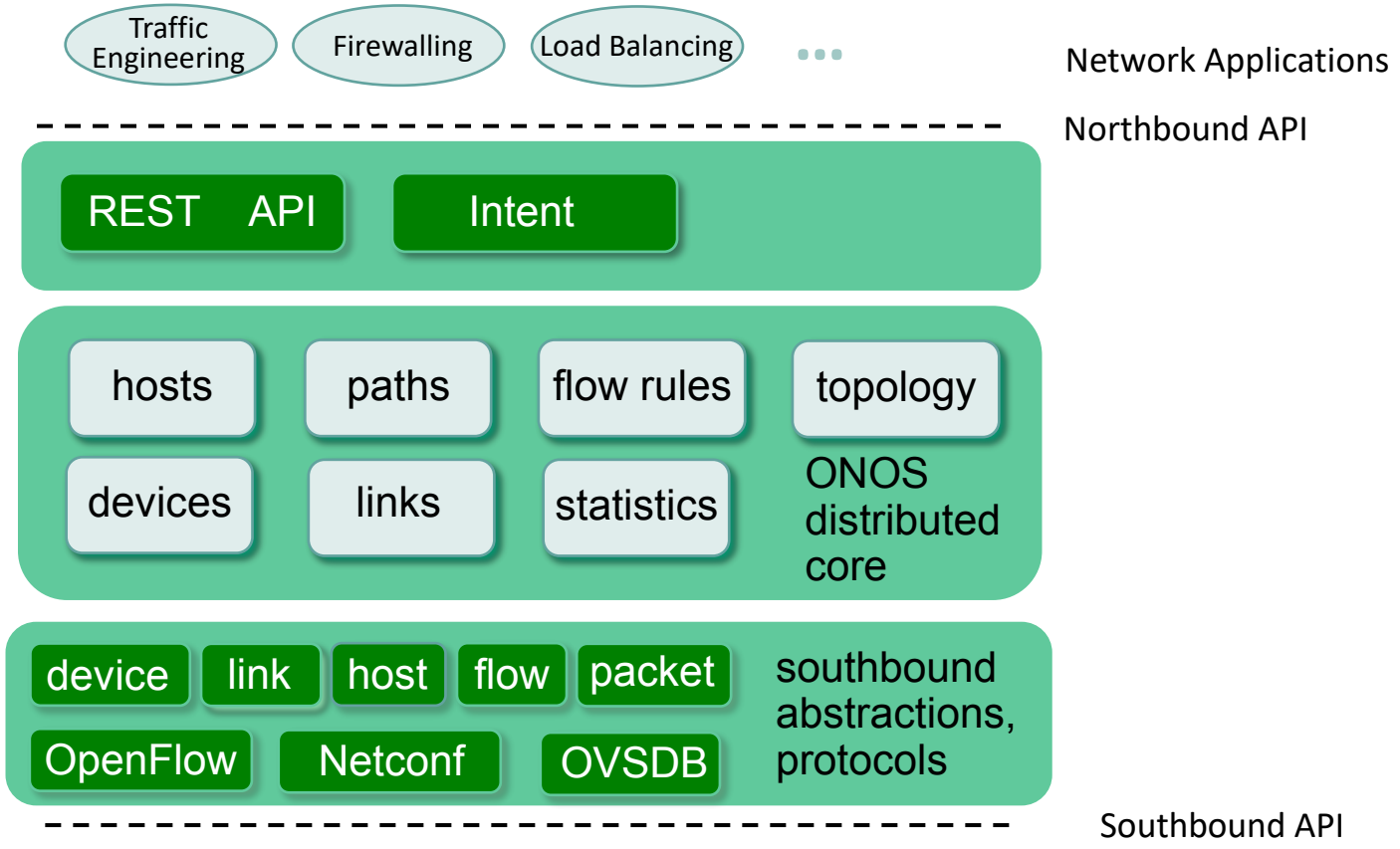


- designed to be fault tolerant (distributed)

Service Abstraction Layer:

- interconnects internal, external applications and services

ONOS controller



- control apps separate from controller
- intent framework: high-level specification of service: what rather than how
- considerable emphasis on distributed core: service reliability, replication performance scaling

Topology discovery and forwarding in SDNs

Routing

Traditionally the **routing** function is **distributed** among the routers in a network

In an **SDN** controlled network, it makes sense to **centralize the routing function within the SDN controller**

The controller can develop a **consistent view** of the network state for calculating shortest paths and can implement application-aware routing policies

Data plane switches are relieved of the processing and storage burden associated with routing, leading to **improved performance**

- path comp → controller
- device → forward info

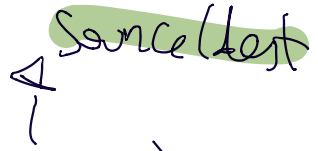
1) Discover topology: link, switch connection, capacity

Routing

The centralized routing application performs two distinct functions:

1) **Link/Topology discovery**
   - The routing function needs to be aware of links between data plane switches
   - The topology discovery in OpenFlow domains currently is not standardized

2) **Topology manager**
   - Maintains the topology information for the network
   - calculates routes in the network (the shortest path between two data plane nodes or between a data plane node and a host



controller should merge different requirements
- short path for a given flow
- if a flow has req. incompatible with maybe if also same src/dest

flexibility <-

Topology Discovery

The OpenFlow (OF) switches have two major initial configurations that enable discovering the topology:

1. every OF switch has initially set the IP address and TCP port of a controller to establish a connection as soon as the device is turned on.
2. switches have preinstalled flow rules to route directly to the controller via a Packet-In message any message of the Link Layer Discovery Protocol (LLDP).

Link Layer Discovery Protocol (LLDP)

- Standardized for the first time in 2005 and 2009 by IEEE
- it is a neighbor discovery protocol of a single jump, i.e. it advertises its identity and capabilities and receives the same information from the adjacent switches.
- vendor neutral protocol
- It works on Layer 2 of the OSI model
- In traditional networks, every LLDP frame is sent by the switches that support and have activated this functionality at fixed intervals of time, configured by the network administrator.
- In OpenFlow-based networks, switches send the LLDP messages to discover the underlying topology by request of the controller.

Reference: Ochoa Aday L, Cervelló Pastor C, Fernández Fernández A. Current trends of topology discovery in OpenFlow-based software defined networks. https://upcommons.upc.edu/bitstream/handle/2117/77672/Current%20Trends%20of%20Discovery%20Topology%20in%20SDN.pdf

LLDP frame

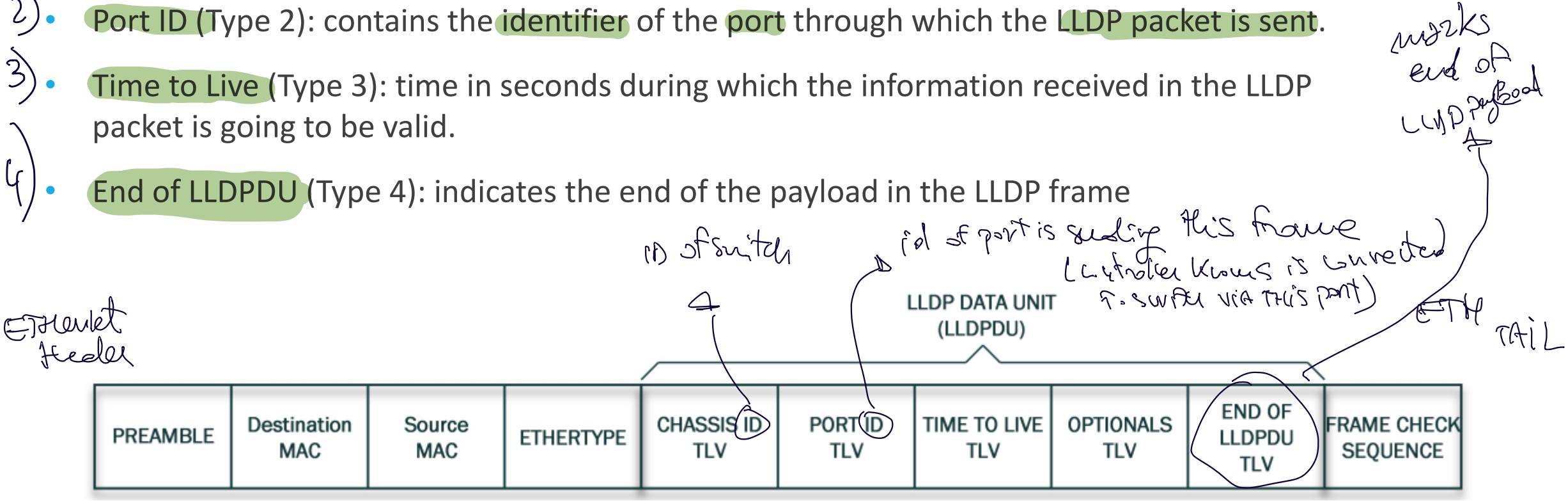
Typically, LLDP uses Ethernet as its "transport" protocol. The Ethernet type for LLDP is 0x88cc.

1) • Chassis ID (Type 1): contains the identifier of the switch that sends the LLDP packet.

2) • Port ID (Type 2): contains the identifier of the port through which the LLDP packet is sent.

3) • Time to Live (Type 3): time in seconds during which the information received in the LLDP packet is going to be valid.

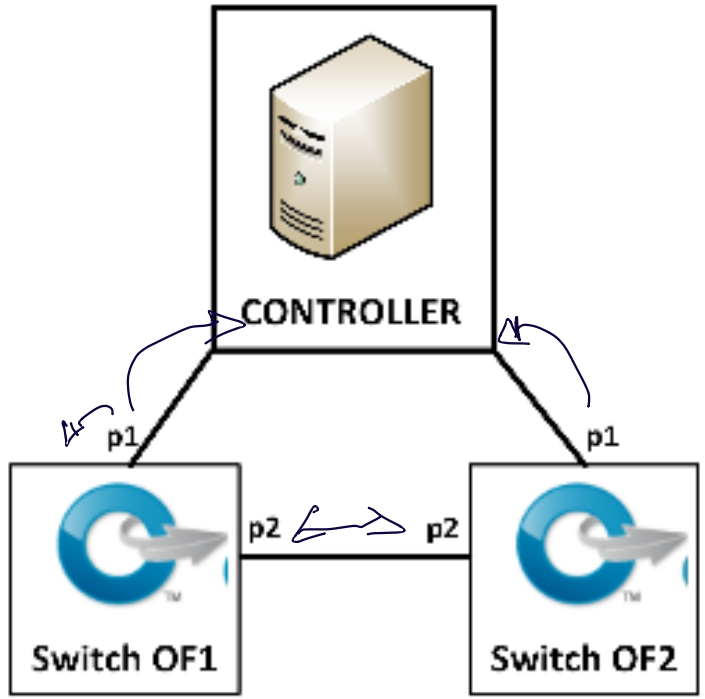
4) • End of LLDPDU (Type 4): indicates the end of the payload in the LLDP frame



Switch Initialization

When the switch is initialized:

- it attempts to establish a connection with the controller
- the controller sends a message **(FEATURE REQUEST MESSAGE)** to the switch
- the switch responds with a message (**FEATURE REPLY MESSAGE**)
- it informs the controller of relevant parameters for the discovery of the links like the **Switch ID**, a **list of active ports** with their respective MAC associates, among others.



At the end of the initial handshake, the controller knows the exact number of active ports on the OF switches

But it doesn't know the physical connections between switches, so it needs to build it -> Topology discovery

Topology Discovery

Based on the information obtained from the initial handshake, the **controller knows the exact number of active ports**

1. The controller generates a **Packet-Out message per active port** on each switch discovered on the network and encapsulates a **LLDP packet** inside each generated message.
2. When an OF switch receives a LLDP message sent by the controller, it **forwards the message by the appropriate Port ID** included in the message to adjacent switches
3. Upon receiving the messages by a port that are not the controller port, **the adjacent switches encapsulate the packet within a Packet-In** message addressed to the controller. Meta-data is included in the message such as Switch ID, Port ID where the LLDP packet is received, among others
4. When the packet comes back to the controller from an adjacent switch, the controller extracts this information, and then it knows **a link exists between those switches**
   - **Switch ID and Port ID in the LLDP message and Switch ID and Port ID in metadata**

Topology Discovery

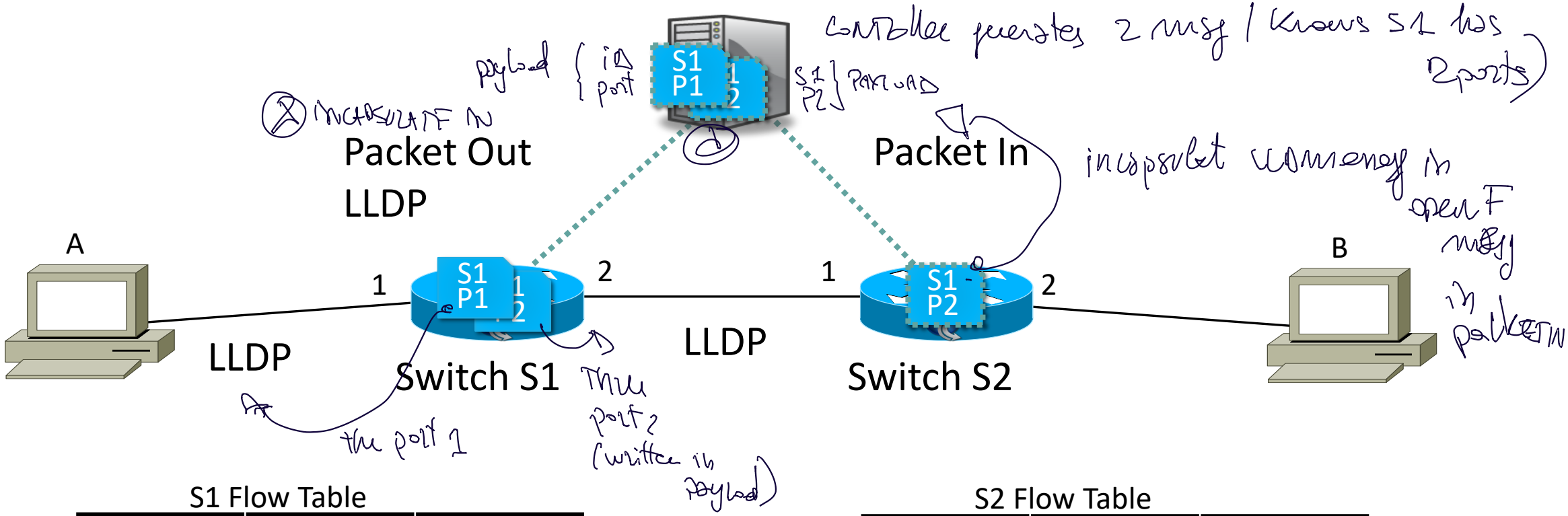
- This process is repeated for every OF switch available on the network.
- The entire process of the topology discovery is performed periodically.
- In a network comprising *S* OF switches interconnected by a set of links *L*, the total of Packet-Out messages that the controller sends out to the network to discover all existing links between OF switches with P active ports is:

$$\text{TOTAL}_{\text{PACKET-OUT}} = \sum_{i=1}^{S} P_i$$

Topology Discovery

Links

Src Switch	Src Port	Dst Switch	Dst Port
S1	2	S2	1



S1 Flow Table

Eth Src	Eth Dst	Out Port

S2 Flow Table

Eth Src	Eth Dst	Out Port

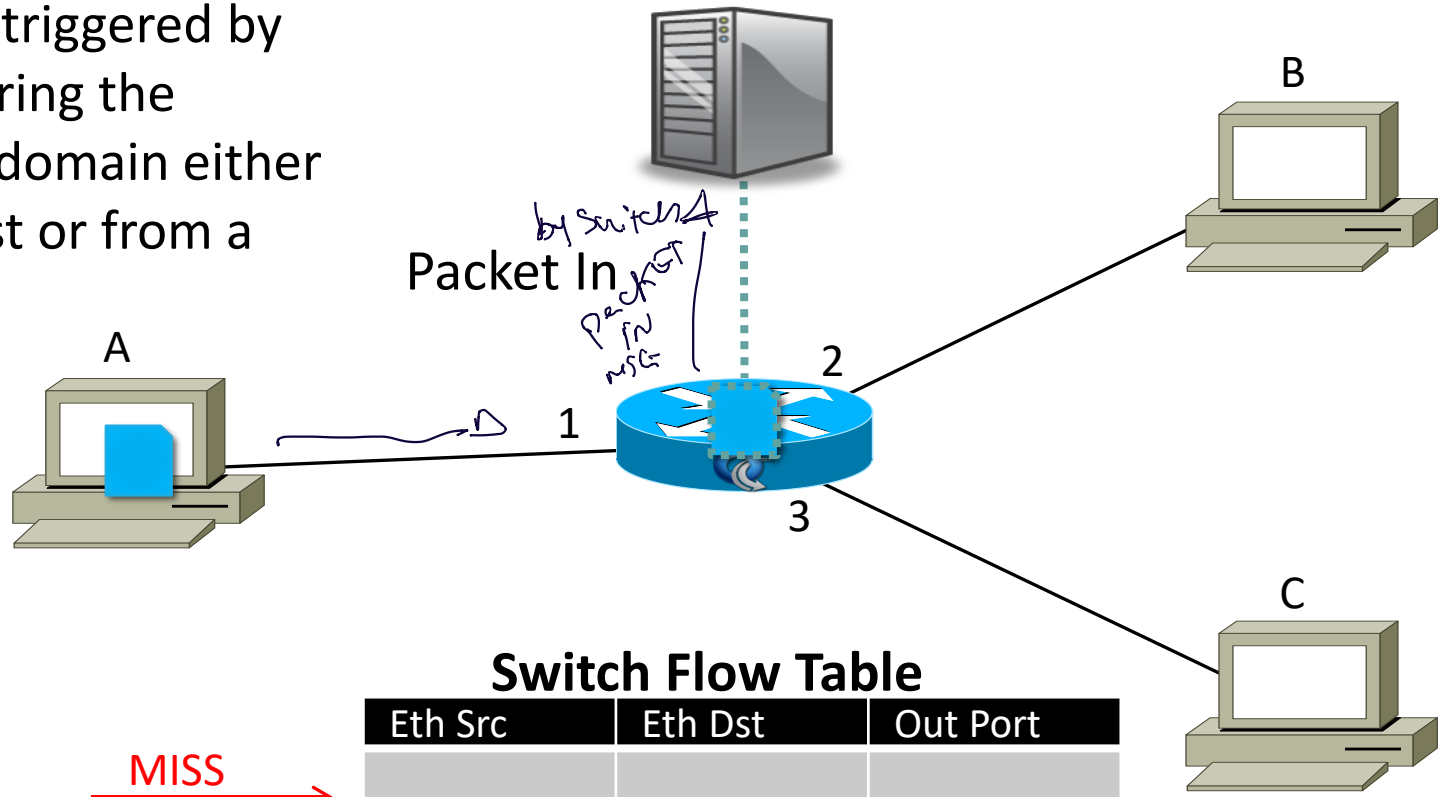
Host Reachability

Packet A -> B

Host Table

Eth Address	Switch	Port
A	1	1

Discovery is typically triggered by unknown traffic entering the controller's network domain either from an attached host or from a neighboring router



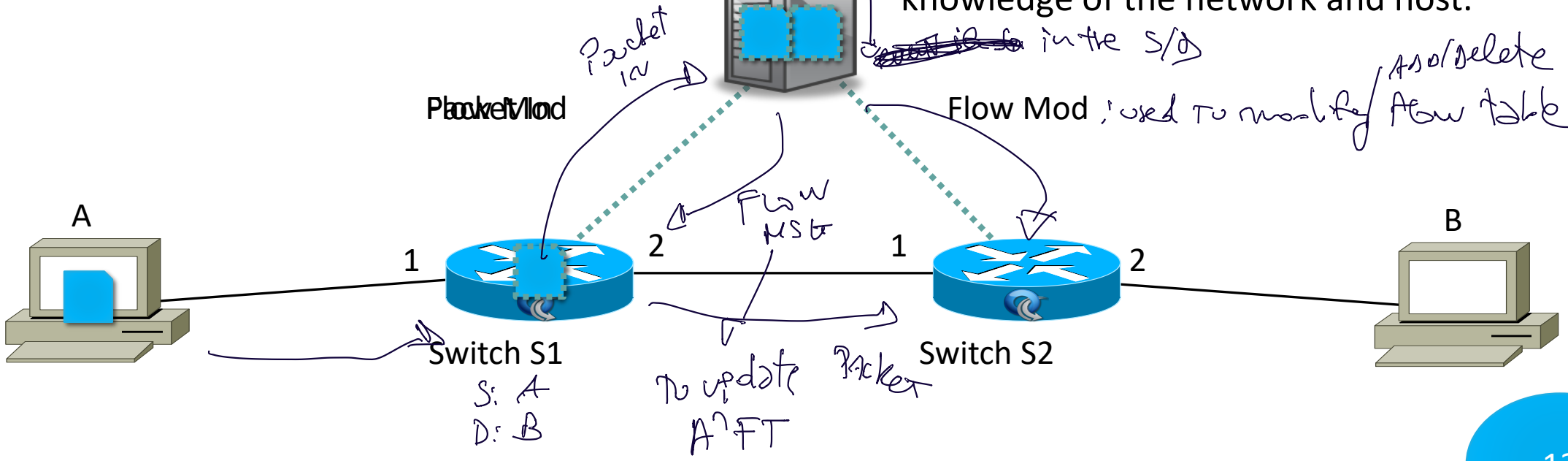
Switch Flow Table

Eth Src	Eth Dst	Out Port

Path Computation

Packet A -> B

Initially flow tables on all switches are empty, but assume that Host table and Topology at the controller are fully populated with the necessary knowledge of the network and host.



S1 Flow Table

Eth Src	Eth Dst	Out Port
A	B	2

S2 Flow Table

Eth Src	Eth Dst	Out Port
A	B	2

13

Path Computation

- When the controller receives a Packet In it will build a route between the source and destination. (compute FT of switch involved in communication)
- This route will be advertised to switches through Flow Modification message
- Upon Flow Mod reception, each switch along the path adds the necessary flow entries to match the packets from this flow.
- These Flow Mod packets will be sent to each switch in **reverse order,** from the switch closest to the destination and so on until the source switch => 1ST TO S2 then TO S1
  - Why? CZ S2 doesn't know how to handle the packet
    CAN happen that S1 has instruction ed S1 send TO S2 BUT get 22->S2 packet NOT ABLE TO HANDLE => RULE: 1ST final switch 2ND source switch
- All future packets from this flow will match directly at each switch and no longer need to take a trip up to the controller.

Exercise

Describe the source, destination and semantics of the following OF messages:

- packet-in

- packet-out

- FlowMod: