Introduction to Network Layer Control Plane: SDN

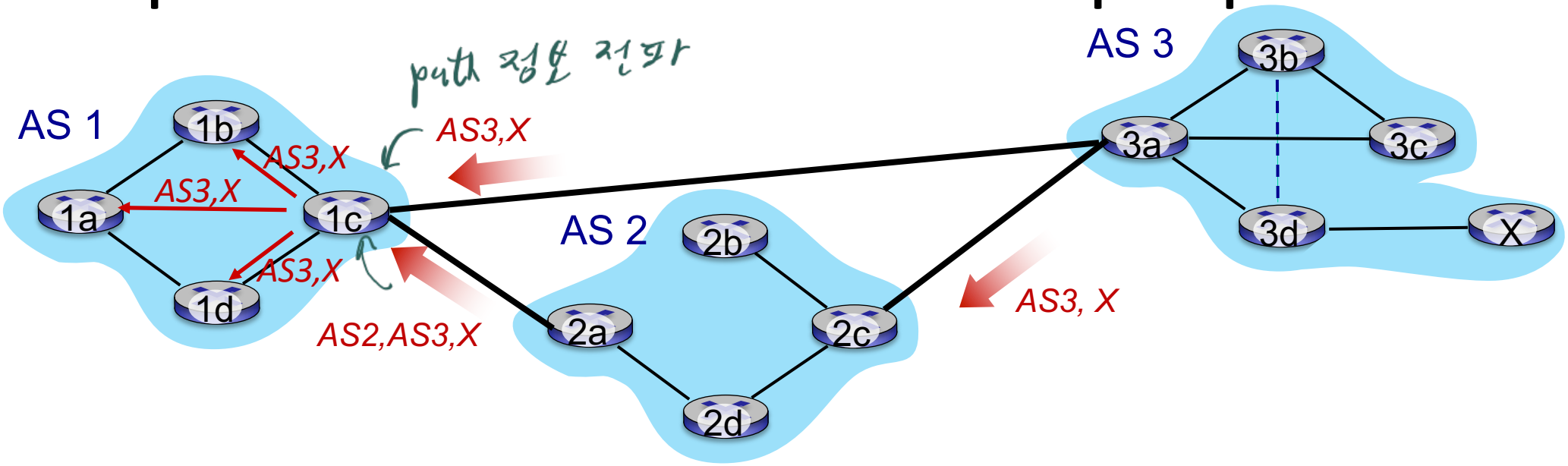
Nov 14, 2024

Min Suk Kang
Associate Professor
School of Computing/Graduate School of Information Security

NetS&P
Research Lab @ KAIST

KAIST

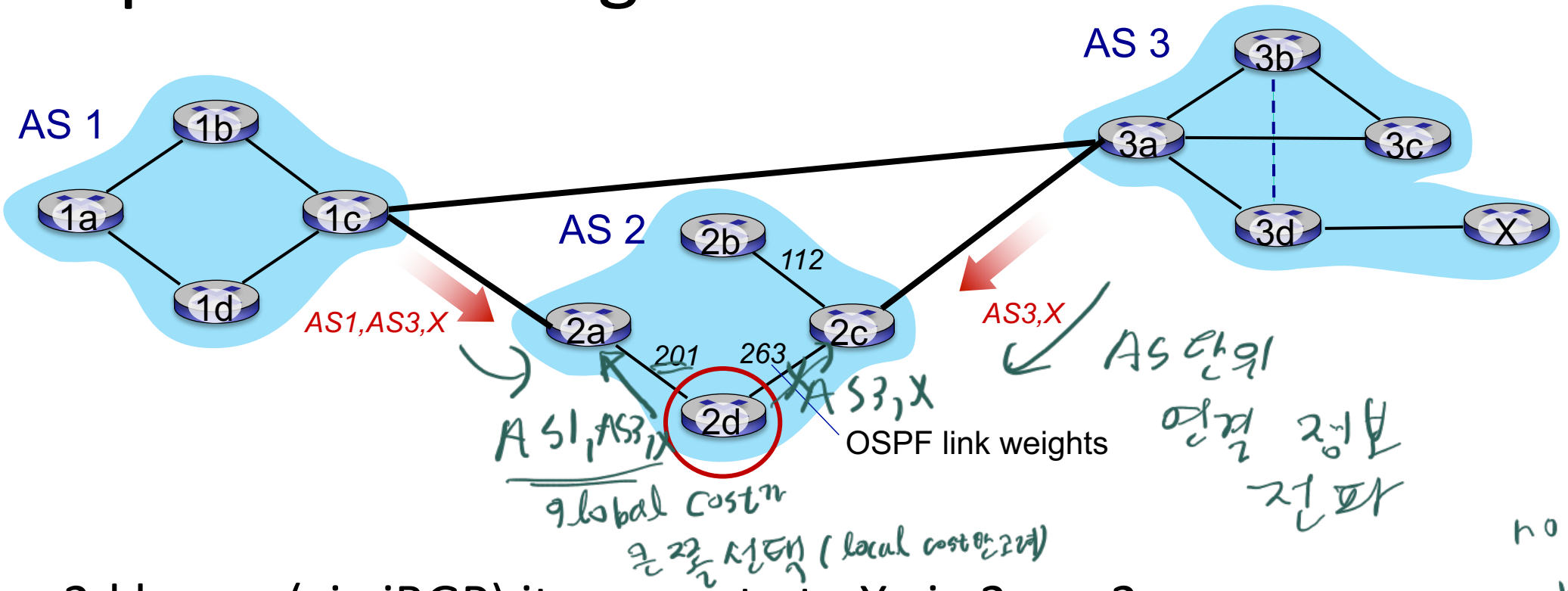
BGP path advertisement: multiple paths



gateway router may learn about multiple paths to destination:

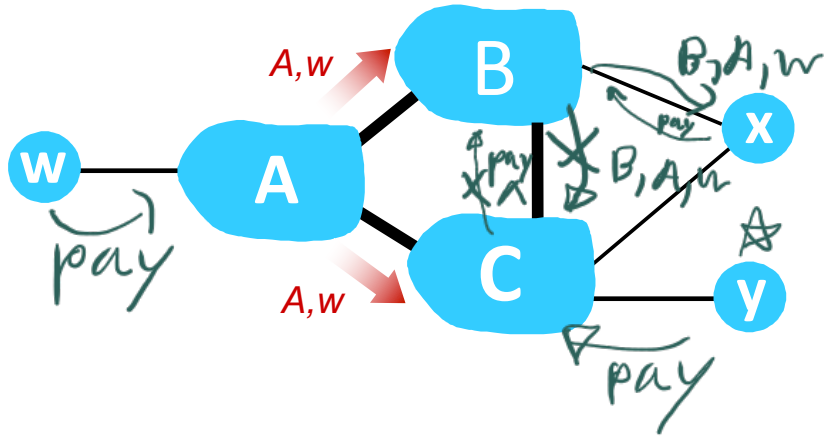
- AS1 gateway router 1c learns path *AS2,AS3,X* from 2a
- AS1 gateway router 1c learns path *AS3,X* from 3a
- based on *policy,* AS1 gateway router 1c chooses path *AS3,X* and advertises path within AS1 via iBGP

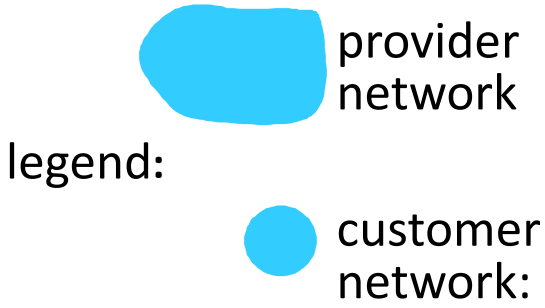
Hot potato routing



- 2d learns (via iBGP) it can route to X via 2a or 2c
- hot potato routing: choose local gateway that has least *intra-domain* cost (e.g., 2d chooses 2a, even though more AS hops to *X*): don't worry about inter-domain cost!

BGP: achieving policy via advertisements

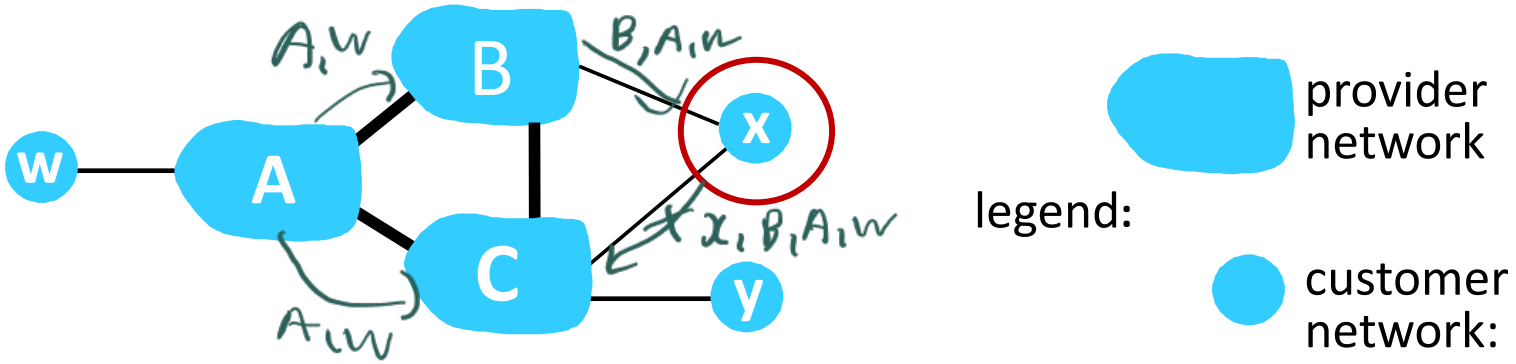




ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs – a typical "real world" policy)

- A advertises path Aw to B and to C
- B *chooses not to advertise* BAw to C!
  - B gets no "revenue" for routing CBAw, since none of C, A, w are B's customers
  - C does *not* learn about CBAw path
- C will route CAw (not using B) to get to w

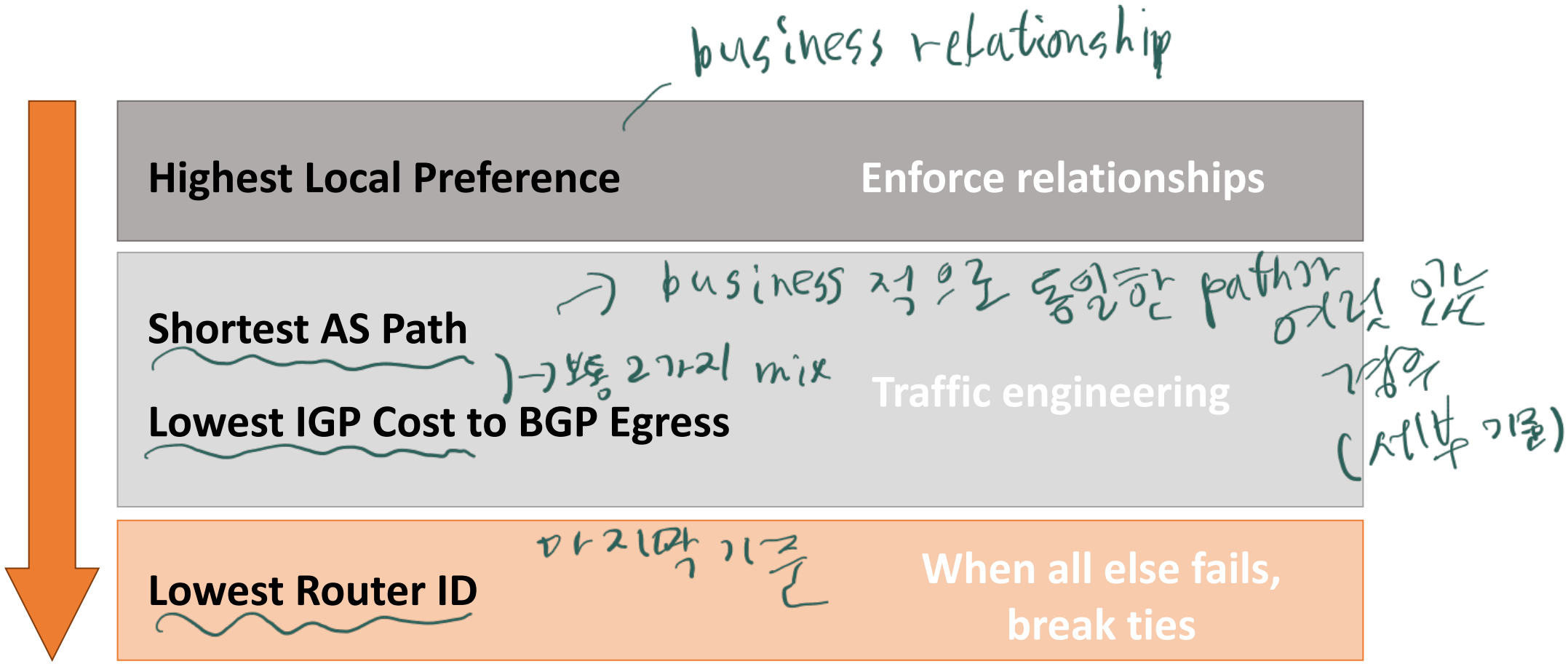
BGP: achieving policy via advertisements (more)



ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs – a typical "real world" policy)

- A,B,C are provider networks
- x,w,y are customer (of provider networks)
- x is dual-homed: attached to two networks
- *policy to enforce:* x does not want to route from B to C via x
  - .. so x will not advertise to B a route to C

Route Selection Summary



Interception in real world

S2W

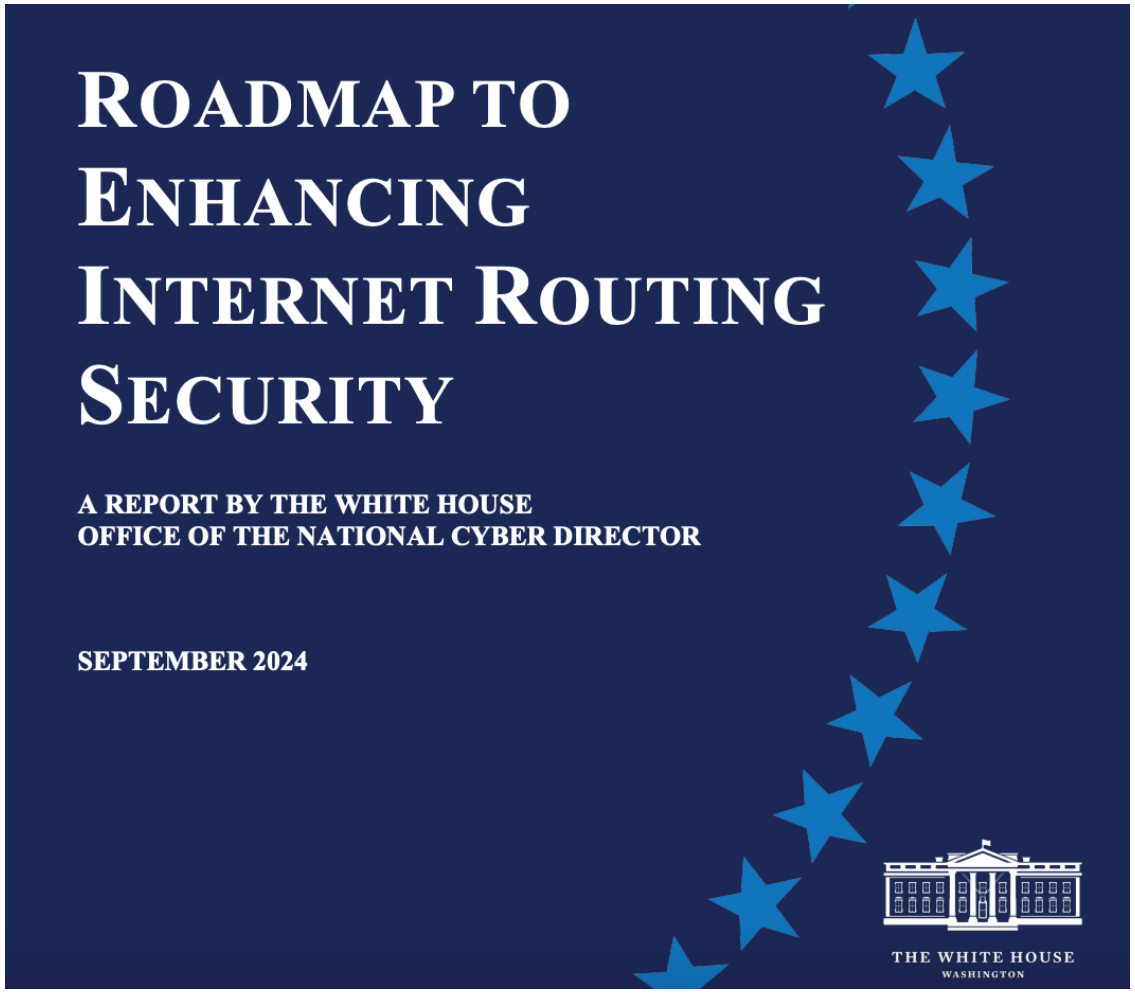
Feb 17 · 18 min read · Listen

Post Mortem of KlaySwap Incident through BGP Hijacking | EN

Author: S2W TALON with eyez (Lead by Sojun Ryu)

> *Last Modified : 2022.02.16.*

Increased attention to BGP security



- https://www.whitehouse.gov/wp-content/uploads/2024/09/Roadmap-to-Enhancing-Internet-Routing-Security.pdf

↓

백악관 보고서

(routing security sumerize)

Network layer: “control plane” roadmap

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol

- network management, configuration
  - SNMP
  - NETCONF/YANG

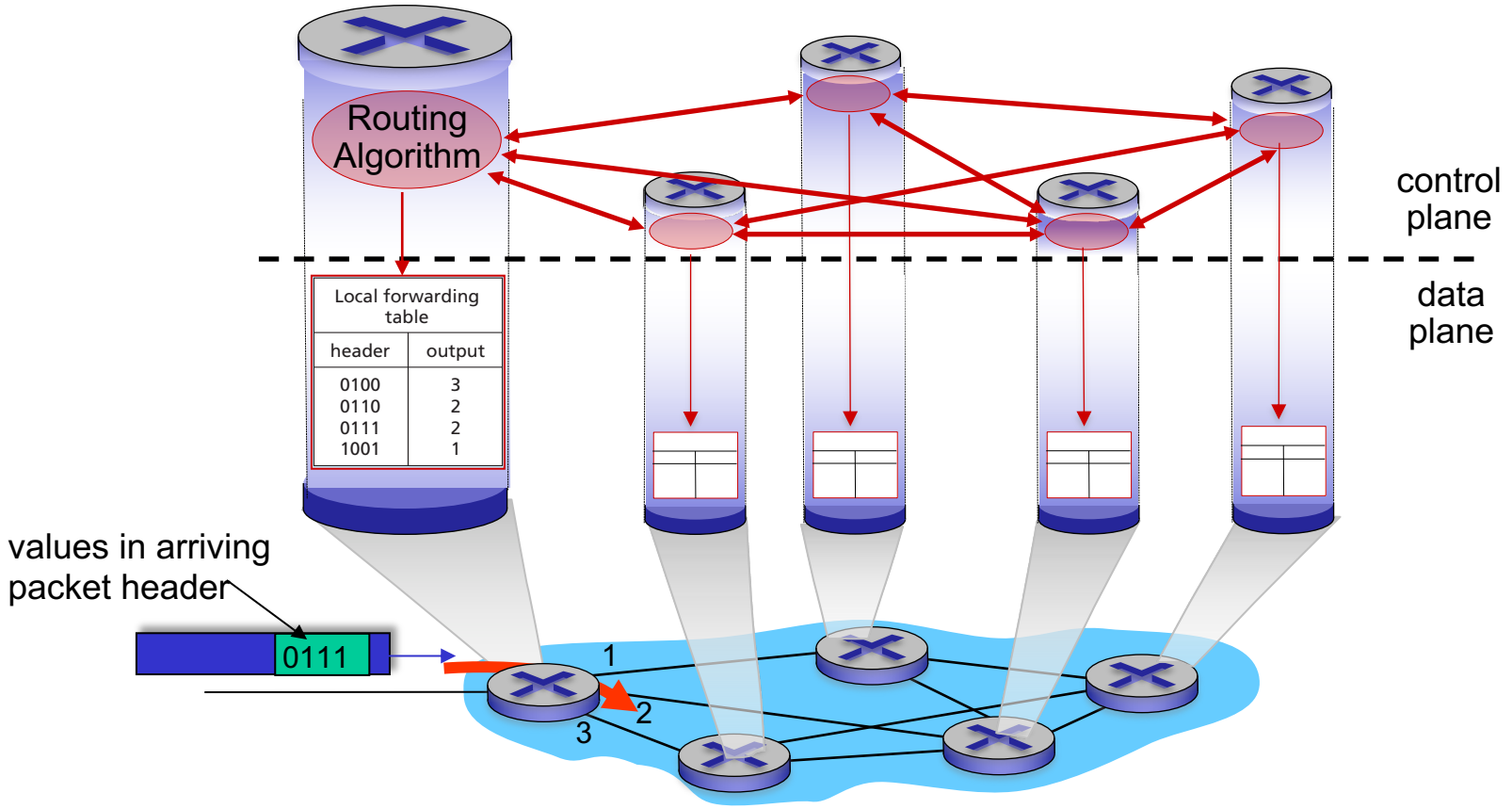
Software defined networking (SDN)

- Internet network layer: historically implemented via distributed, per-router control approach:
  - *monolithic* router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, IS-IS, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)
  - different "middleboxes" for different network layer functions: firewalls, load balancers, NAT boxes, ..
- ~2005: renewed interest in rethinking network control plane

Per-router control plane

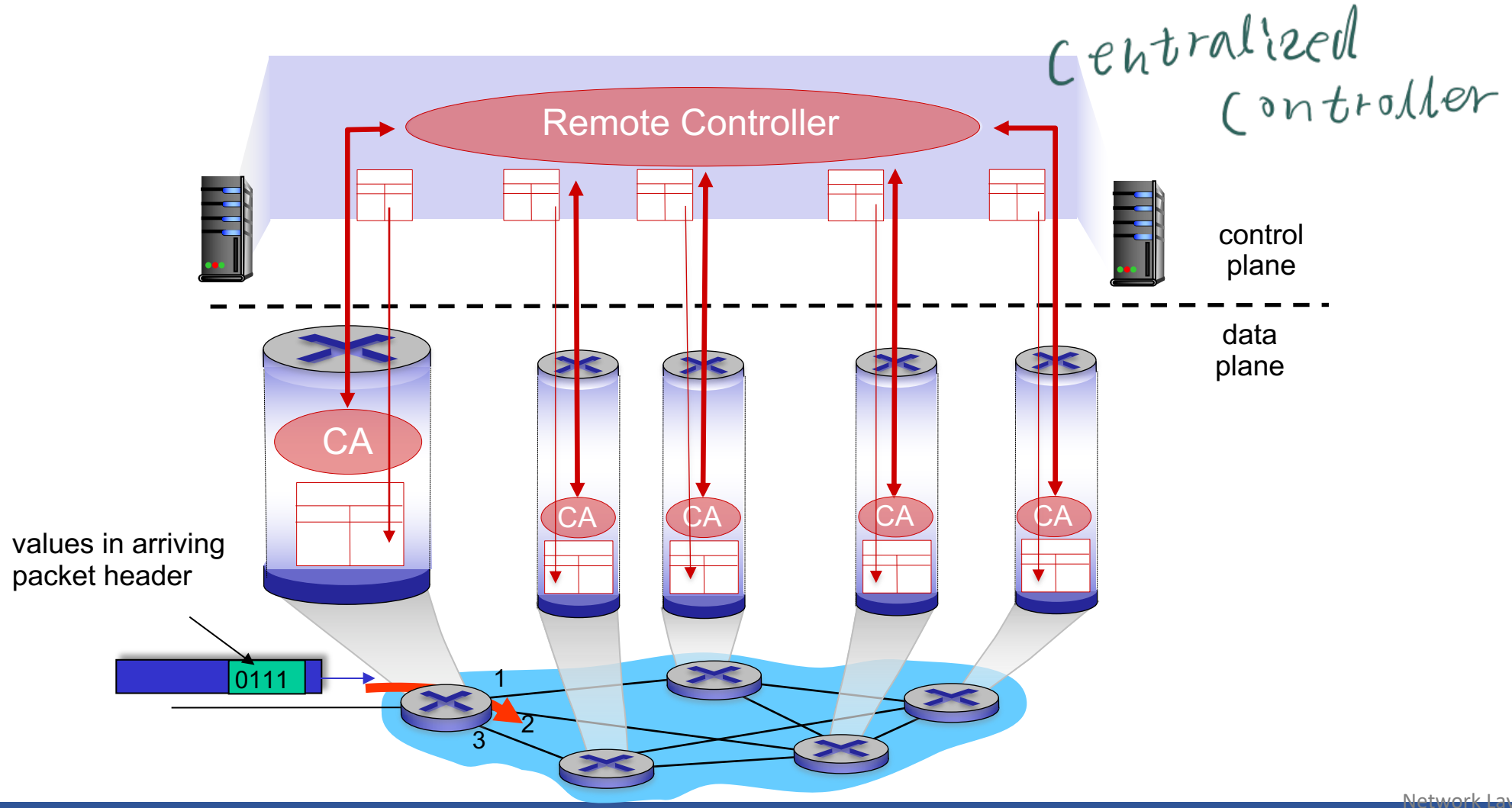
개별 router의 algorithm

Individual routing algorithm components *in each and every router* interact in the control plane to computer forwarding tables



Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



Software defined networking (SDN)

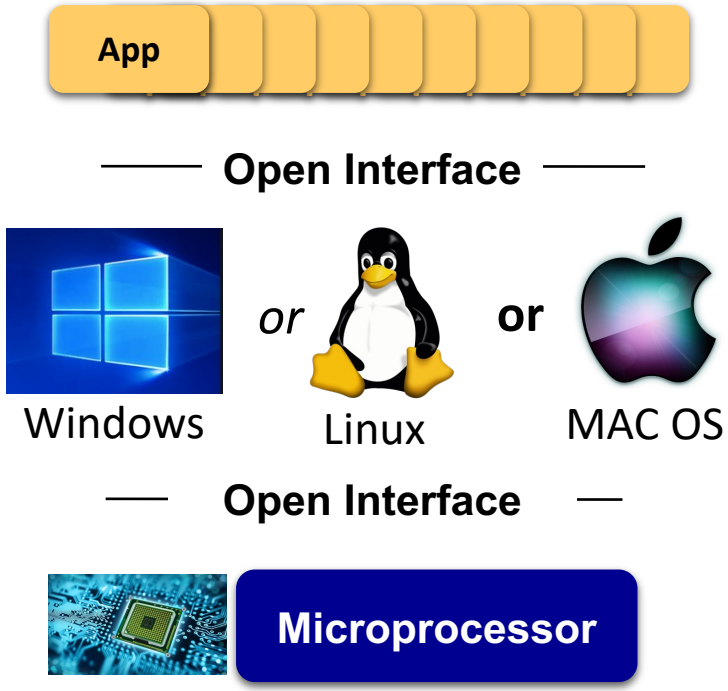
Why a *logically centralized* control plane?

- easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- table-based forwarding (recall OpenFlow API) allows "programming" routers
  - centralized "programming" easier: compute tables centrally and distribute
  - distributed "programming" more difficult: compute tables as result of distributed algorithm (protocol) implemented in each-and-every router
- open (non-proprietary) implementation of control plane
  - foster innovation: let 1000 flowers bloom

SDN analogy: mainframe to PC revolution

ex) openflow

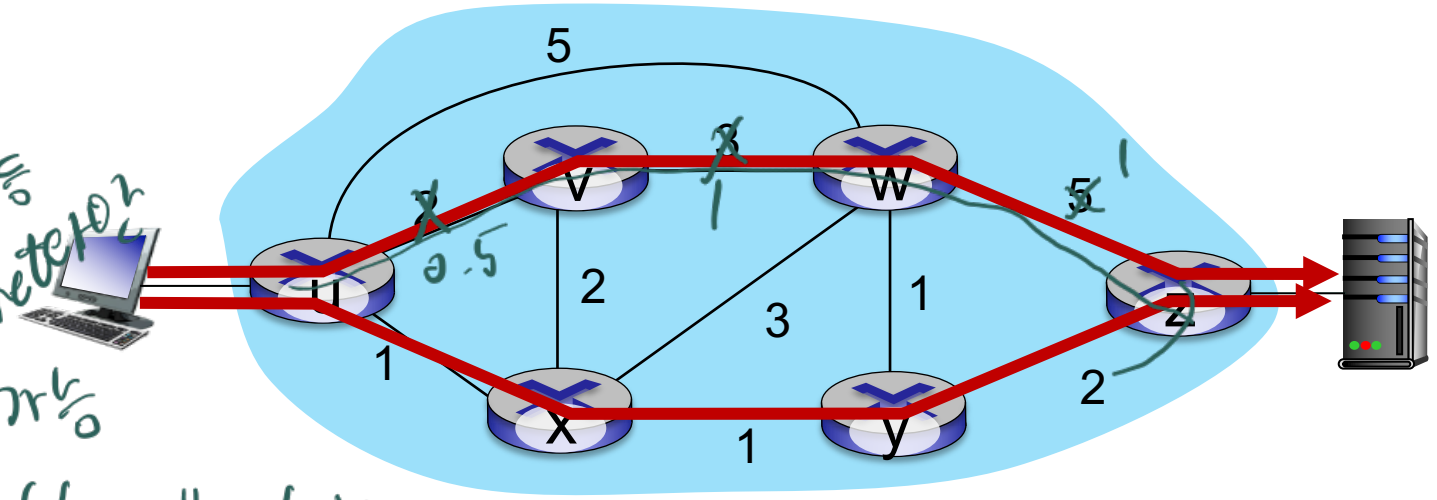
Vertically integrated
Closed, proprietary
Slow innovation
Small industry



Horizontal
Open interfaces → 다양한 app개발로 급격한 혁신
Rapid innovation
Huge industry

* Slide courtesy: N. McKeown

Traffic engineering: difficult with traditional routing

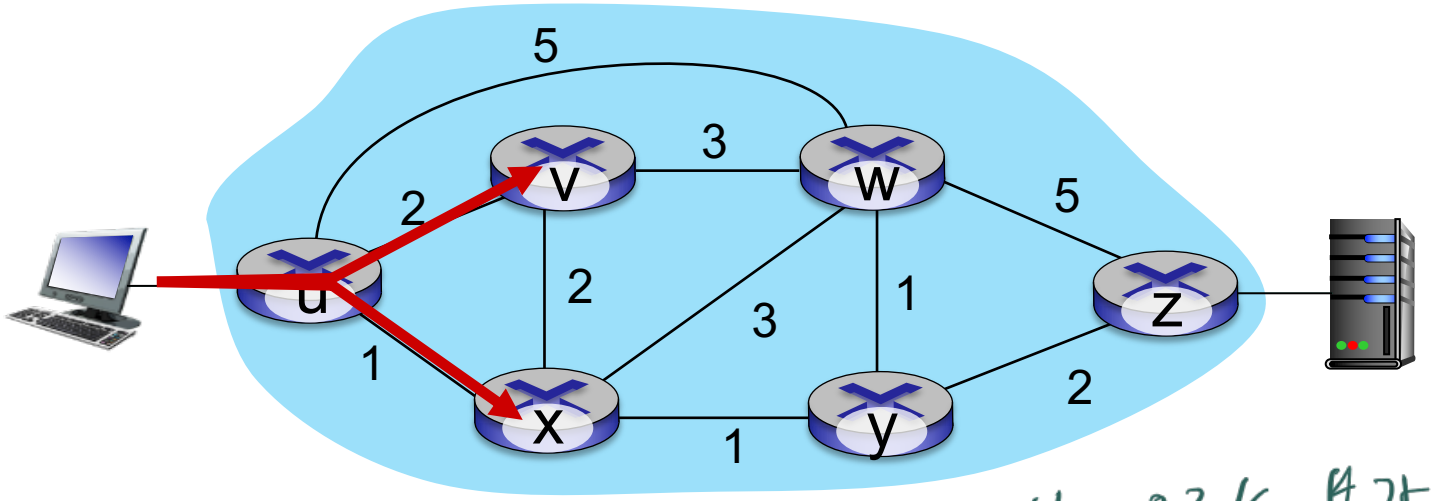


Q: what if network operator wants u-to-z traffic to flow along *uvwz*, rather than *uxyz*?

A: need to re-define link weights so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!

link weights are only control "knobs": not much control!

Traffic engineering: difficult with traditional routing

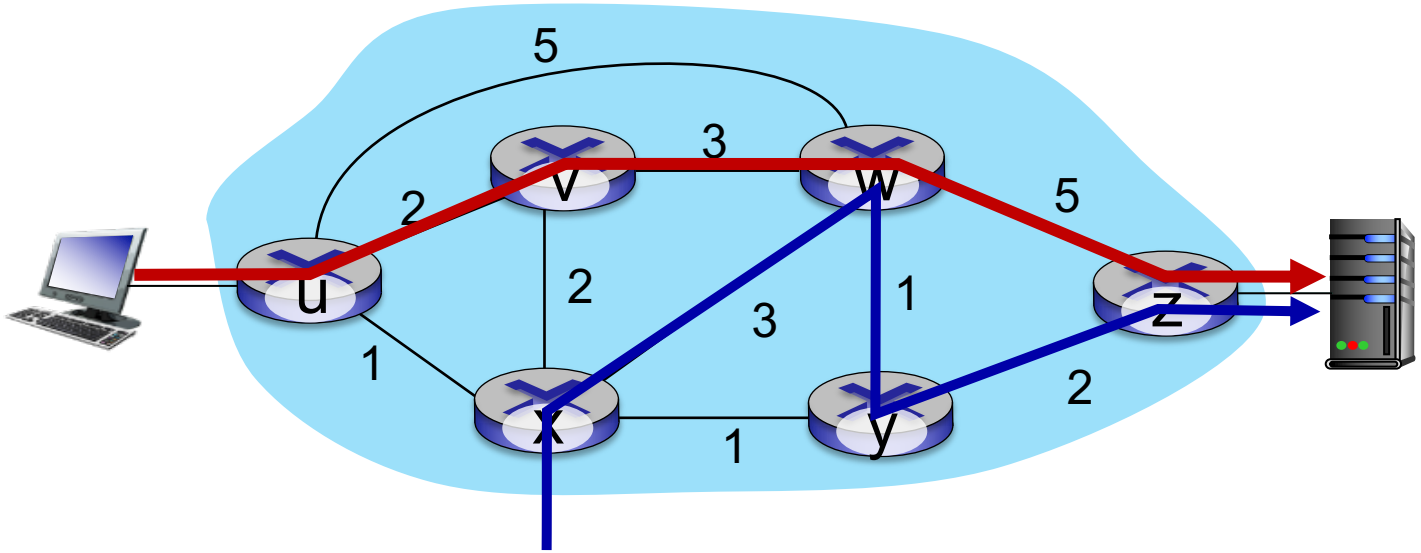


↳ dijkstra algorithm으로는 불가
⇒ 새로운 algorithm update 필요
↓
router는 routing algorithm이 내장되어 있으므로 교체필요

Q: what if network operator wants to split u-to-z traffic along uvwz *and* uxyz (load balancing)?

A: can't do it (or need a new routing algorithm)

Traffic engineering: difficult with traditional routing



Q: what if w wants to route blue and red traffic differently from w to z?

A: can't do it (with destination-based forwarding, and LS, DV routing)

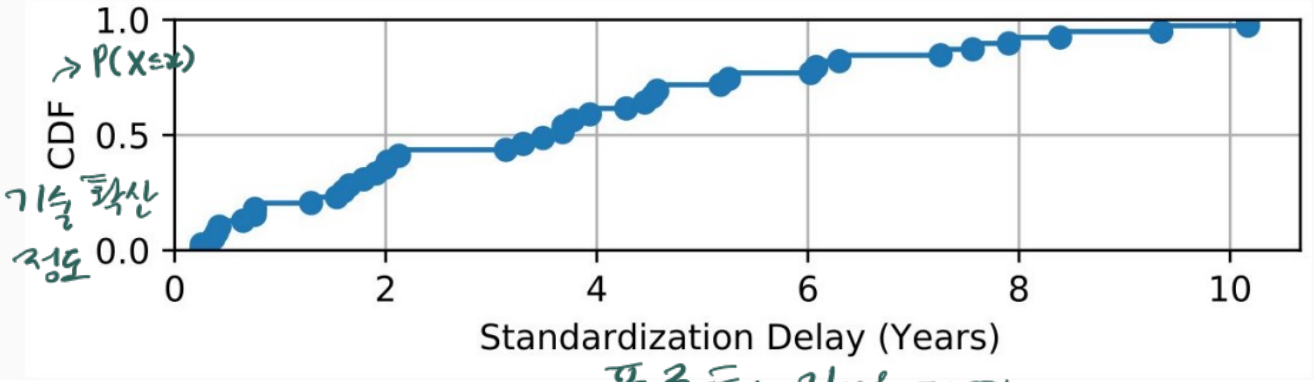
We learned in Chapter 4 that generalized forwarding and SDN can be used to achieve *any* routing desired → match-action 일반화로 다양한 routing 가능

Some more reasons

- Evolution of network protocols

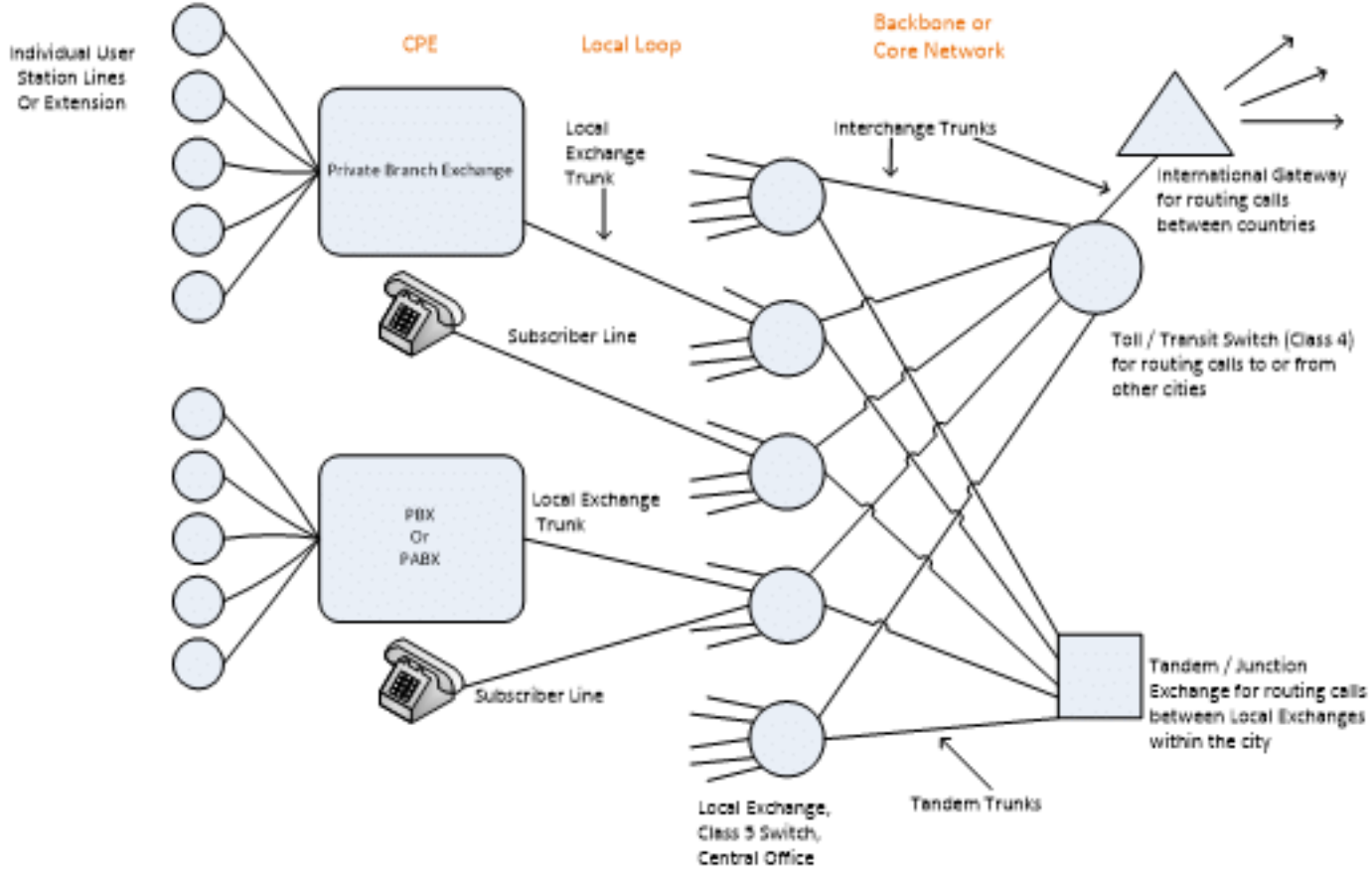
The evolution is complex:

1. Standardization by the IETF (3.5 years in average for BGP)
2. Implementation on the vendor OS
3. Update routers of networks



Trends repeat

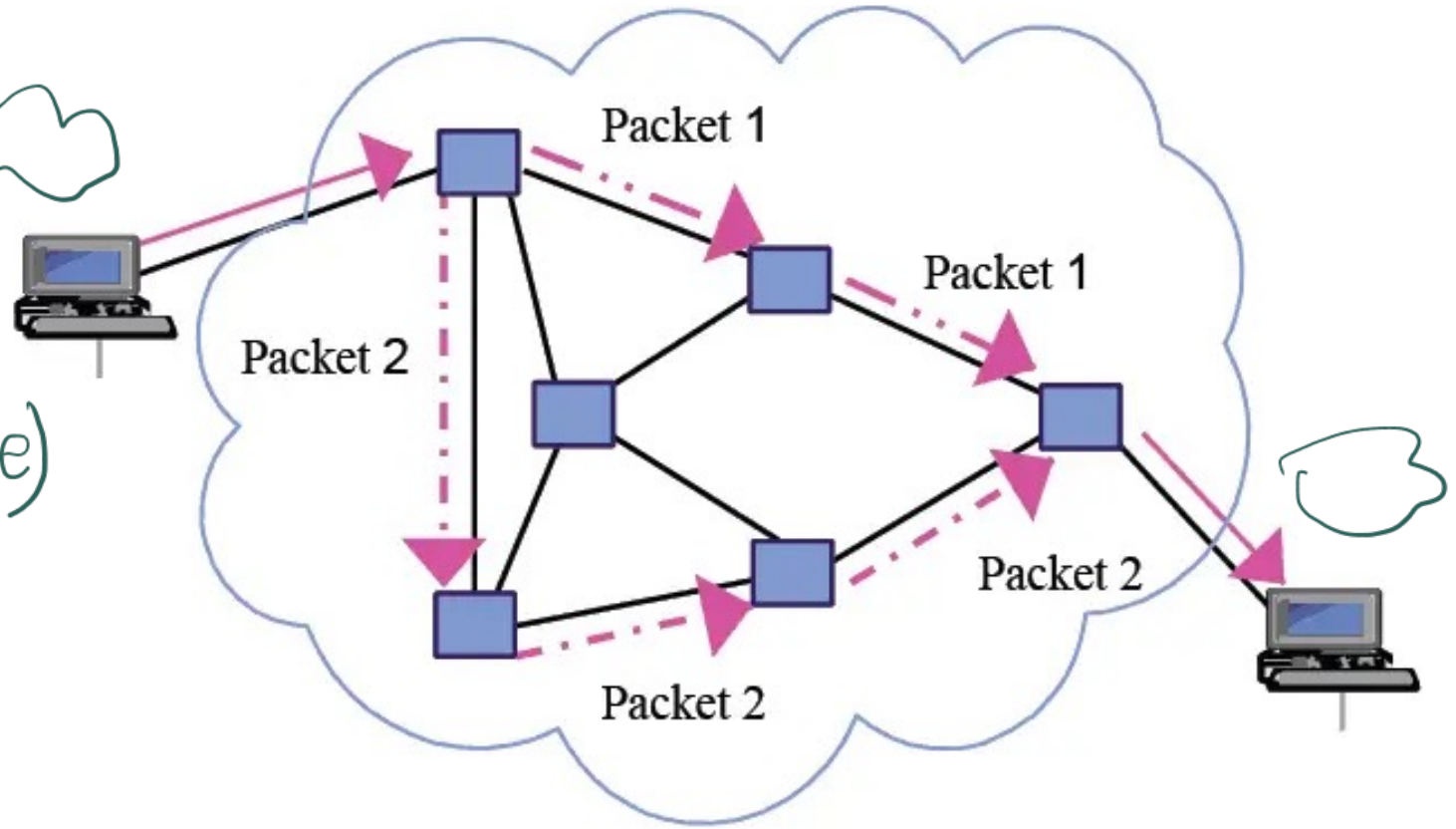
- Circuit switching (-90s)



(Public Switched Telephone Network)

Trends repeat

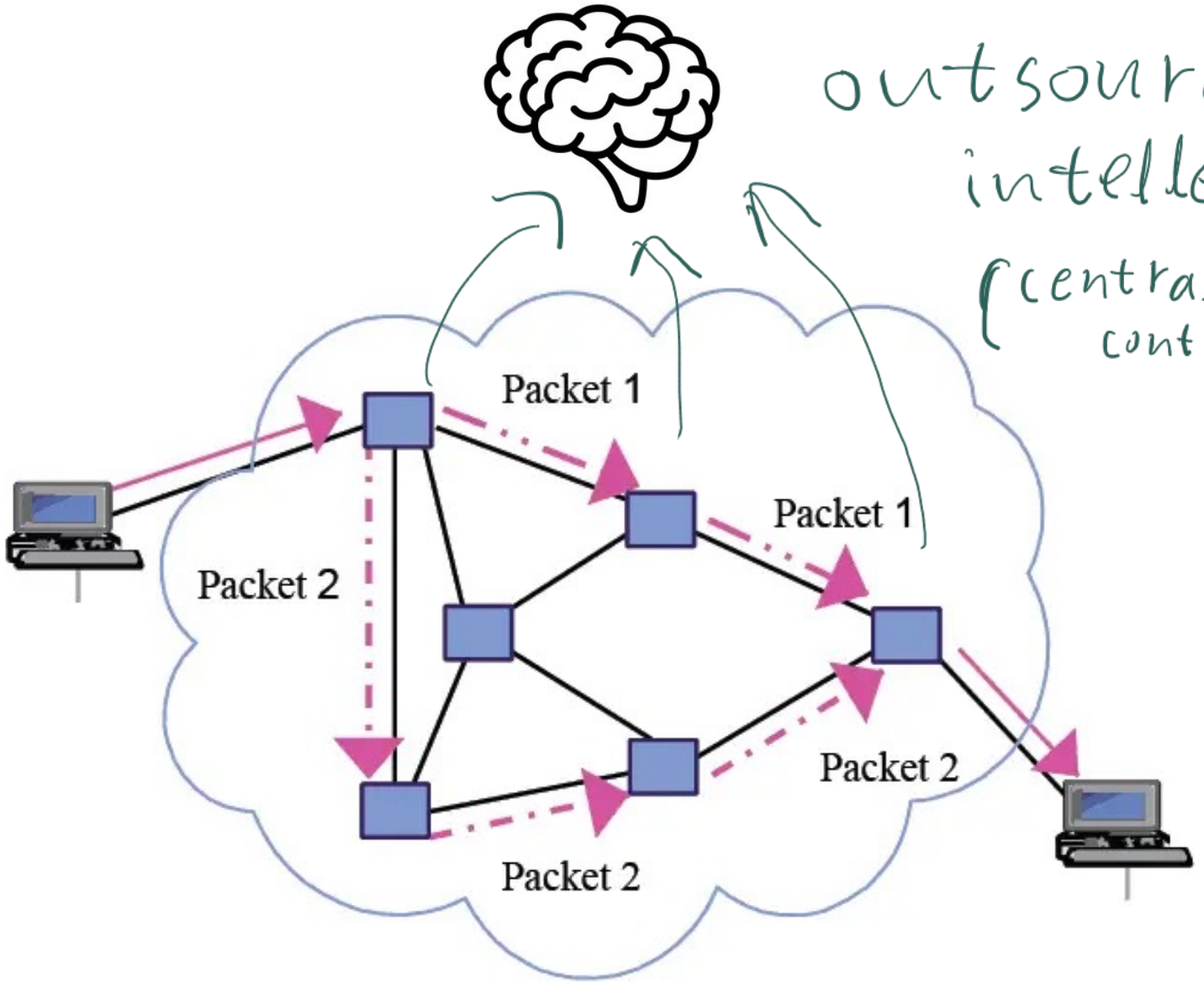
- Packet switching (90s-)



Trends repeat

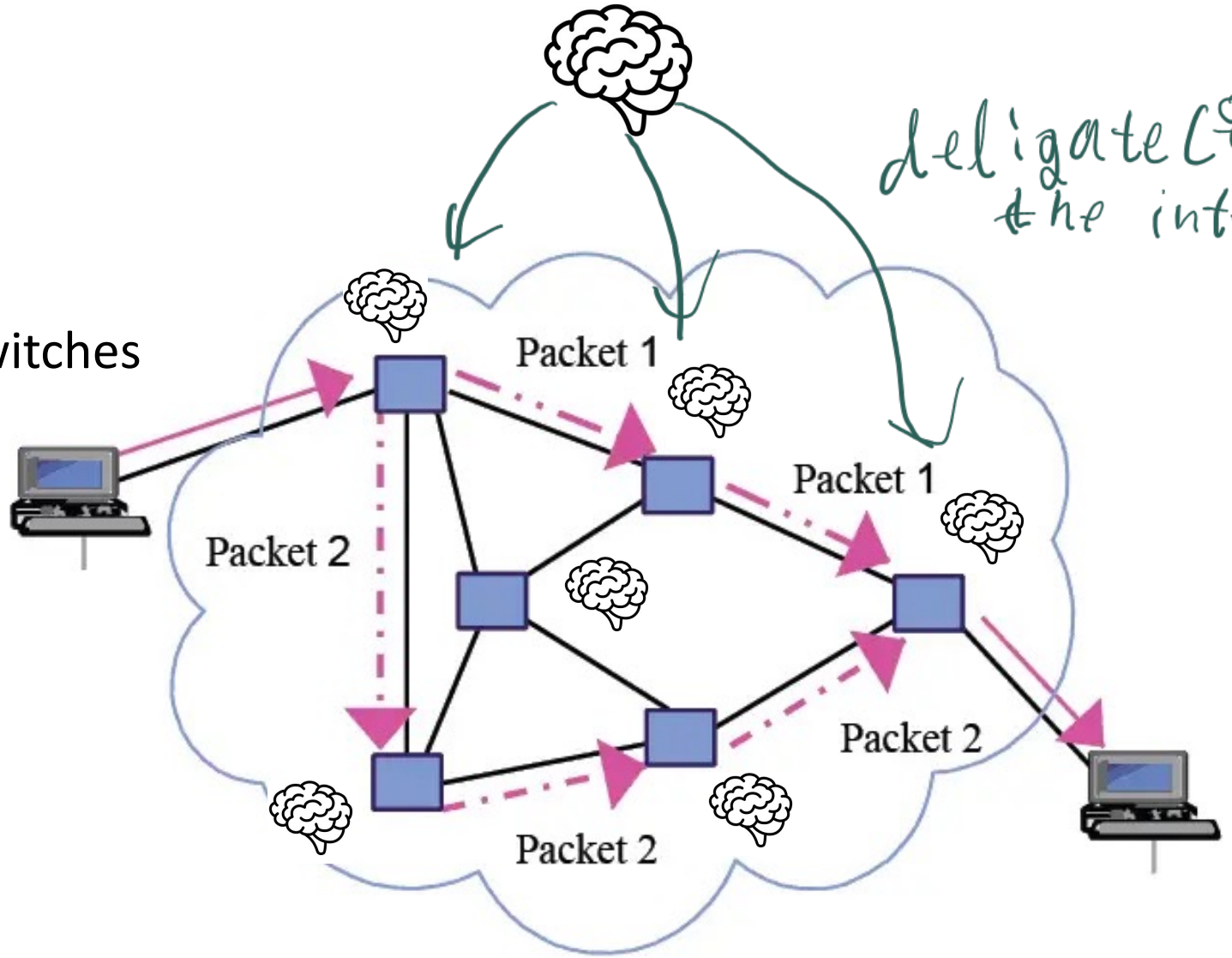
- Packet switching
  with SDN (10s-)

기능의 집약화



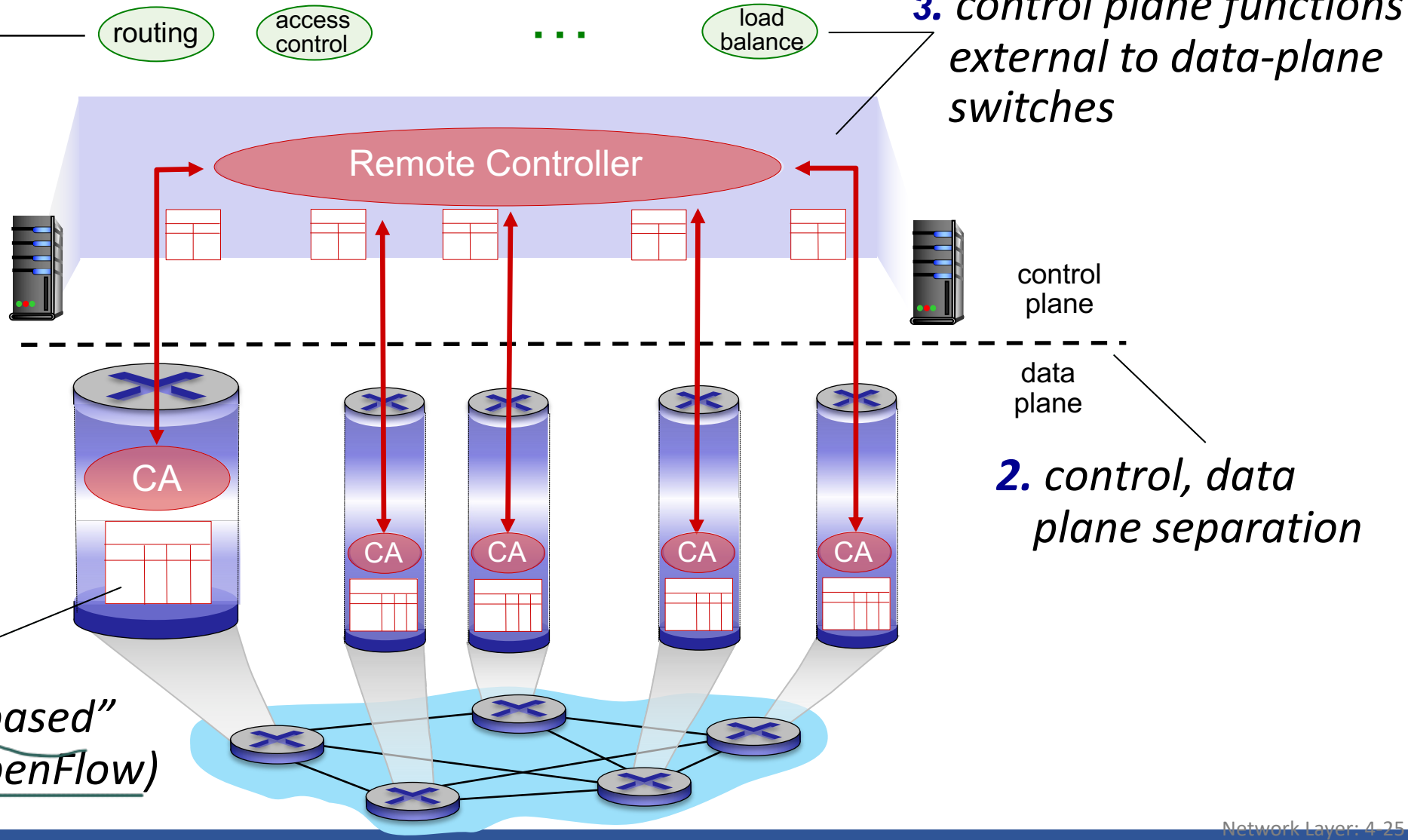
Trends repeat

- Packet switching
  with programmable switches
  (around 2015~)



Software defined networking (SDN)

4. *programmable control applications*



3. *control plane functions external to data-plane switches*

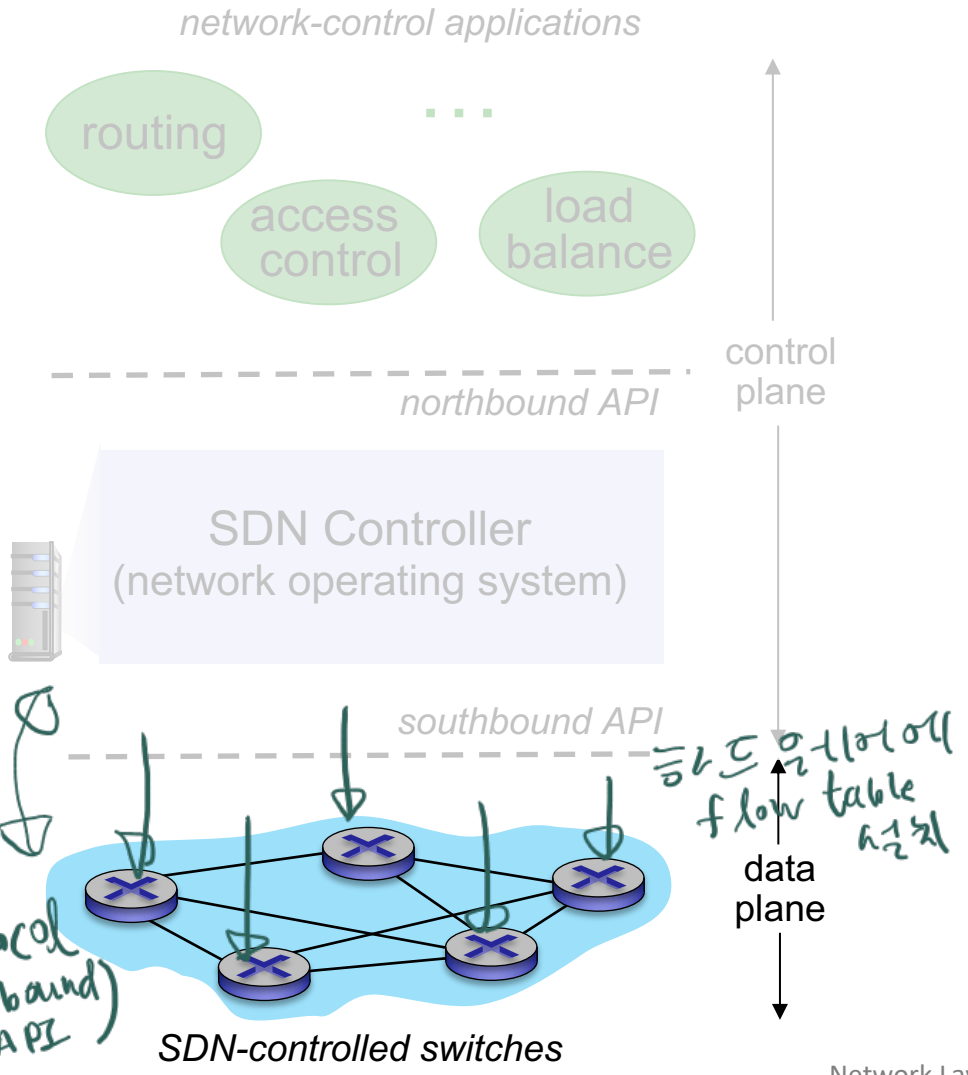
2. *control, data plane separation*

1: *generalized "flow-based" forwarding (e.g., OpenFlow)*

Software defined networking (SDN)

Data-plane switches:

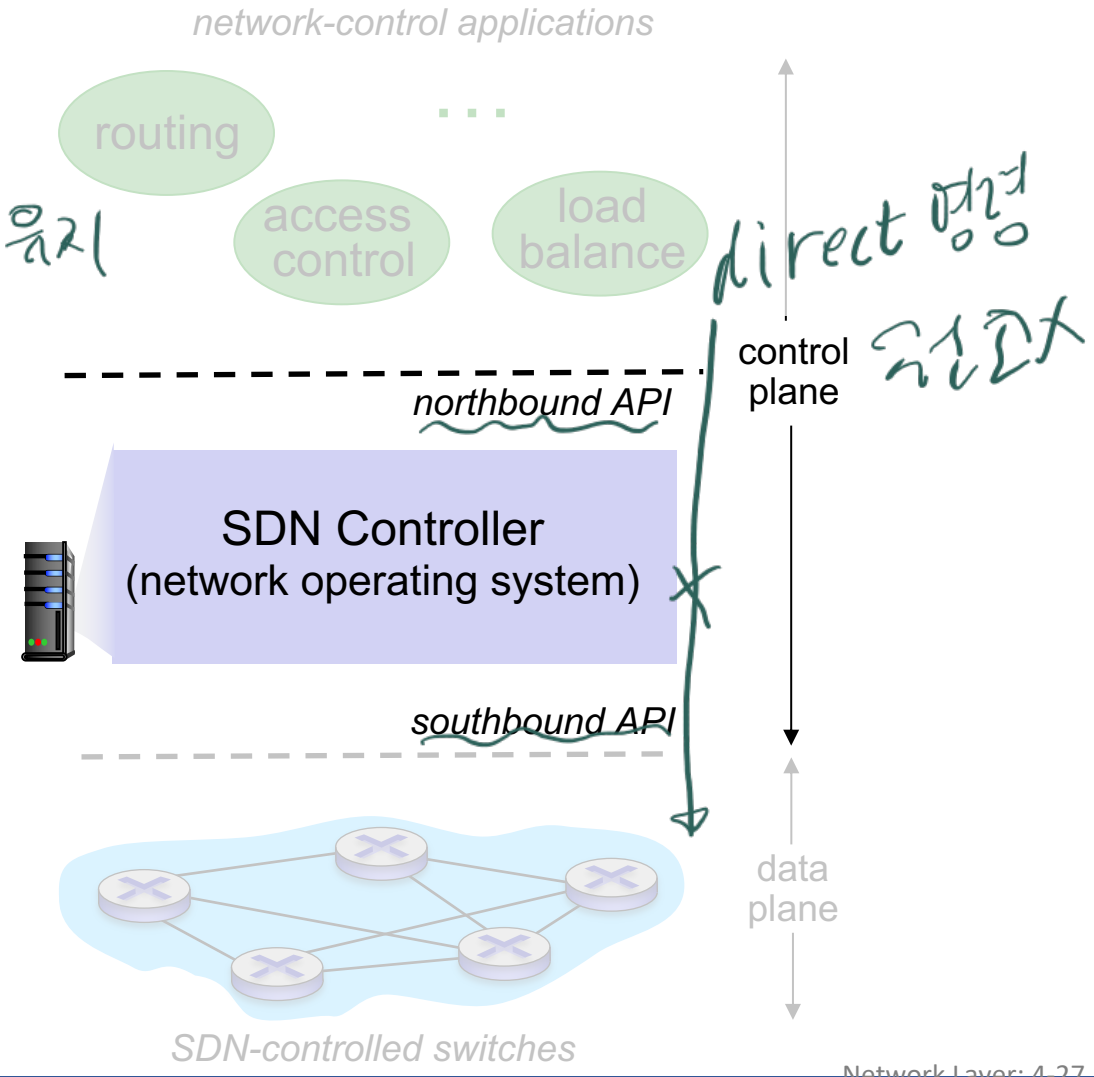
- fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) in hardware
- flow (forwarding) table computed, installed under controller supervision
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable, what is not
- protocol for communicating with controller (e.g., OpenFlow)



Software defined networking (SDN)

SDN controller (network OS):

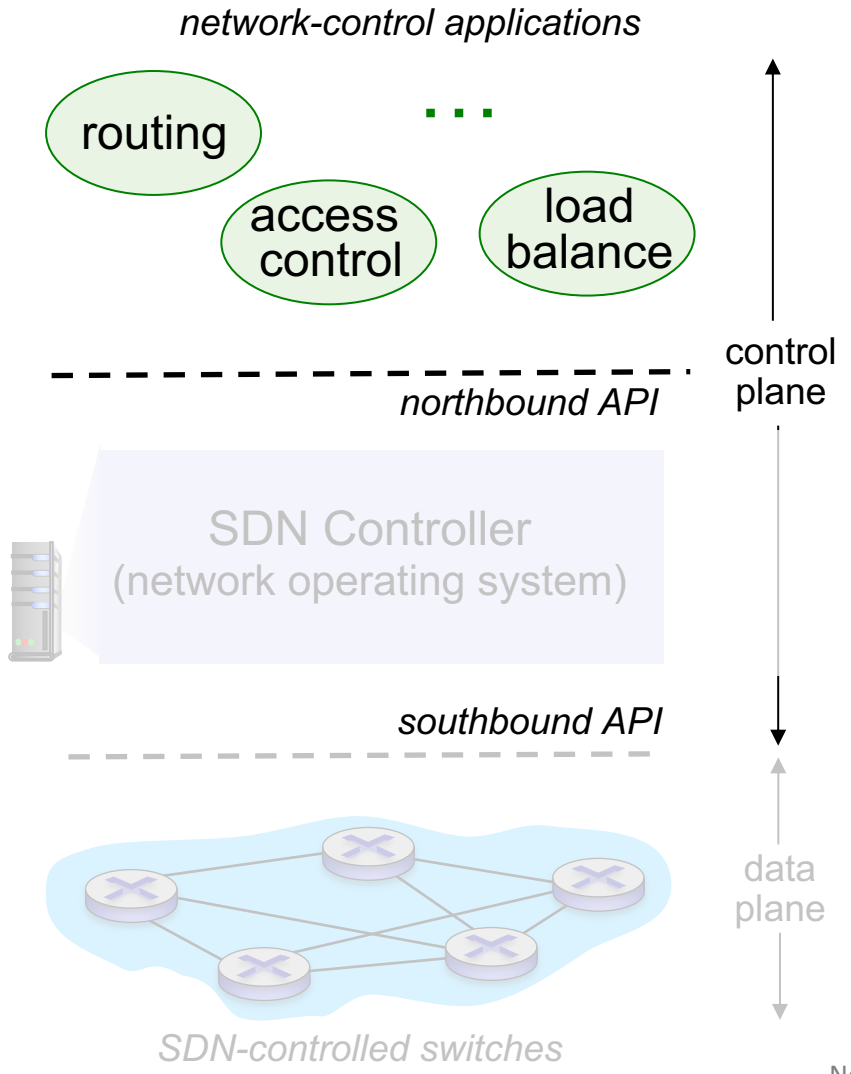
- maintain network state information 네트워크 상태 정보 유지
- interacts with network control applications "above" via northbound API
- interacts with network switches "below" via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness 고장 오류



Software defined networking (SDN)

network-control apps:

- "brains" of control: implement control functions using lower-level services, API provided by SDN controller
- *unbundled:* can be provided by 3rd party: distinct from routing vendor, or SDN controller

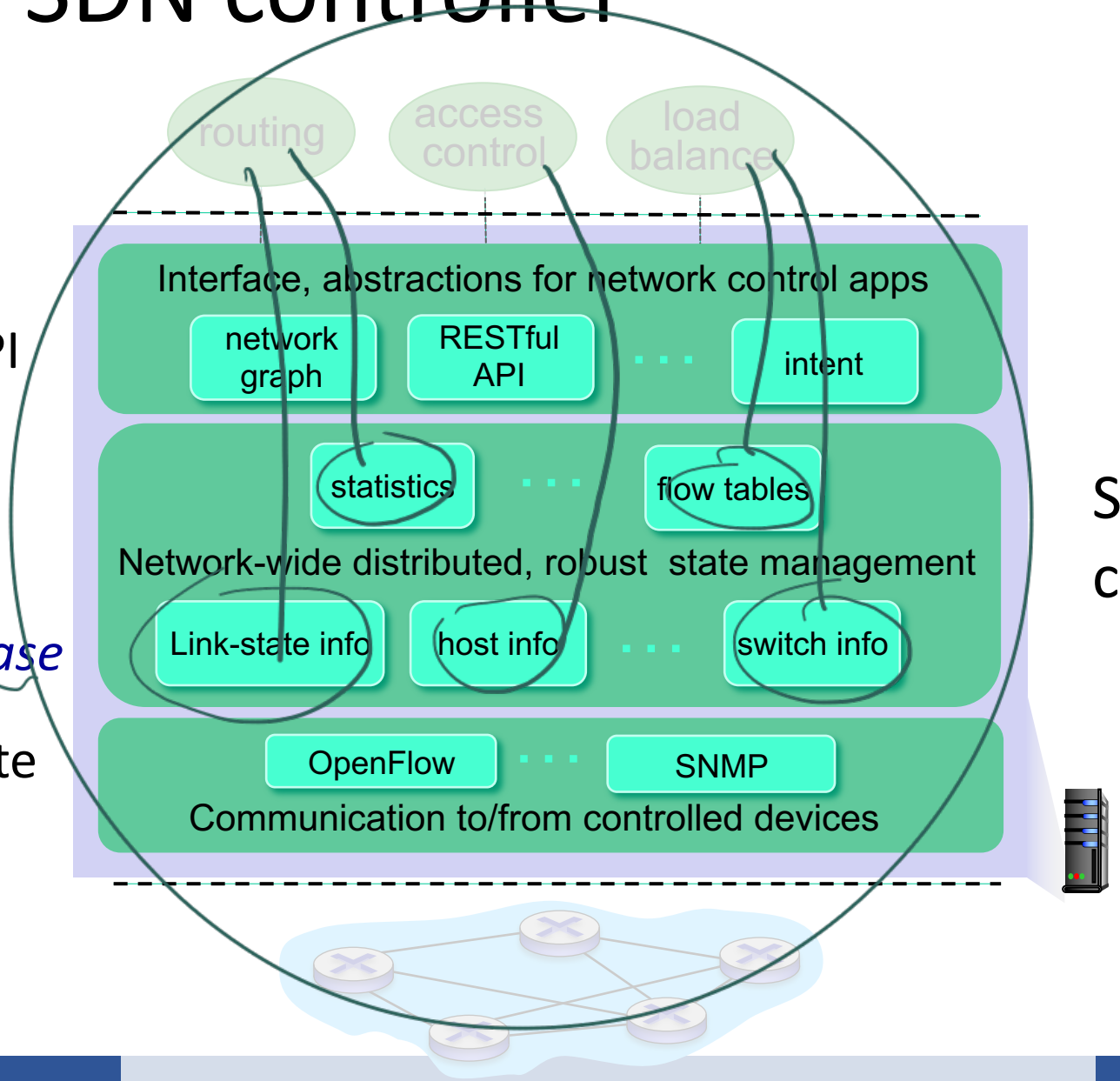


Components of SDN controller

interface layer to network control apps: abstractions API

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

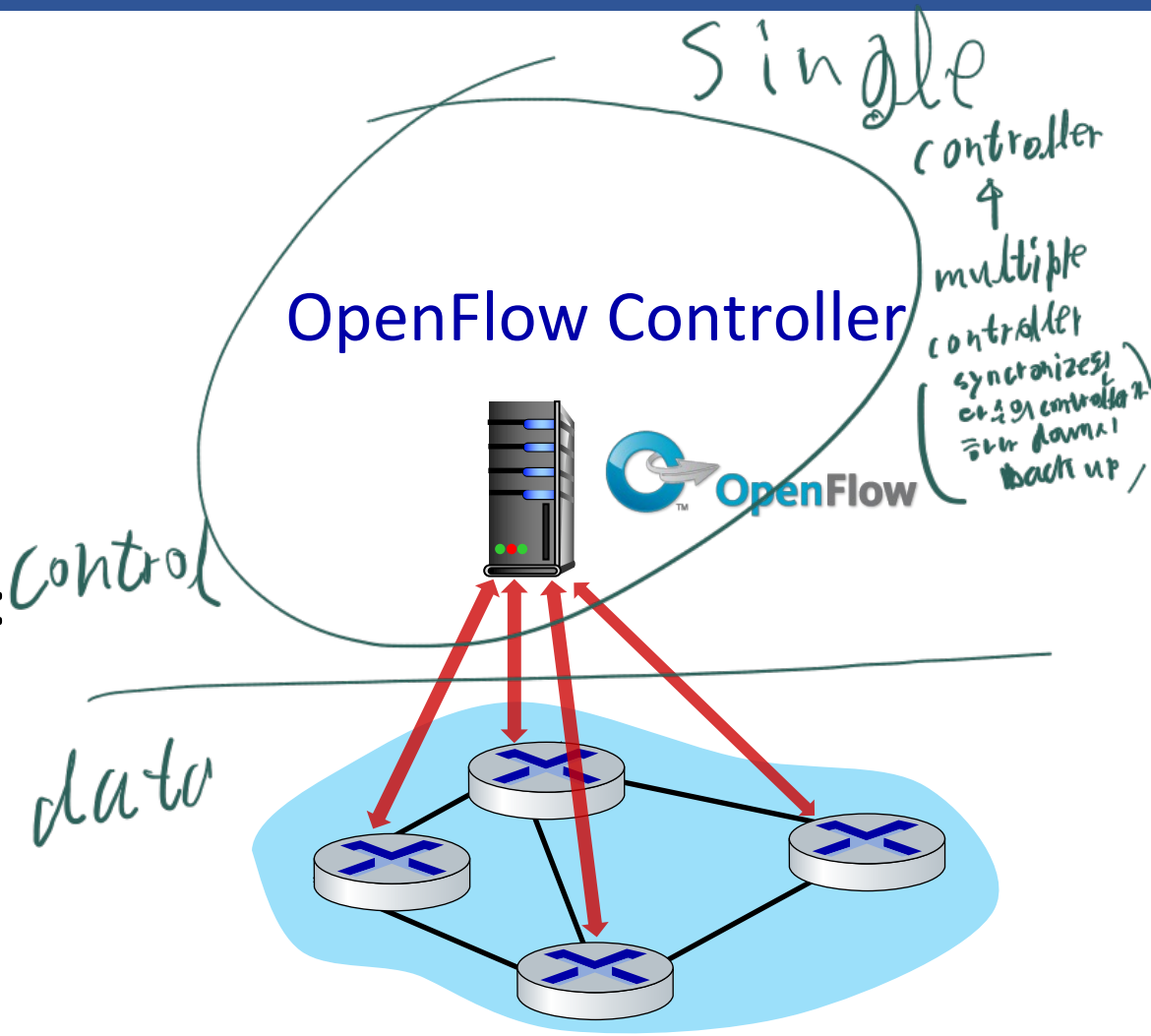
communication: communicate between SDN controller and controlled switches



SDN controller

OpenFlow protocol

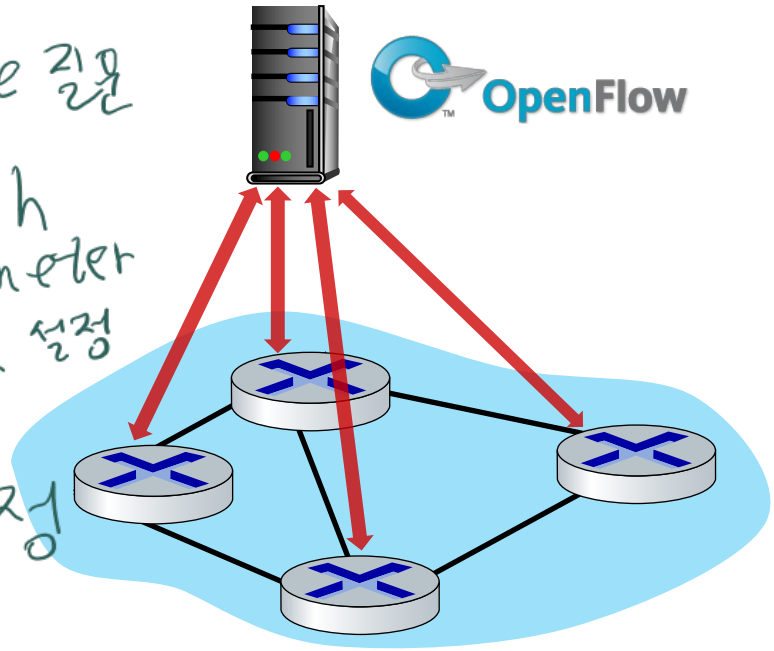
- operates between controller, switch
- TCP used to exchange messages
  - optional encryption
- three classes of OpenFlow messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc.)
- distinct from OpenFlow API
  - API used to specify generalized forwarding actions



OpenFlow: controller-to-switch messages

Key controller-to-switch messages

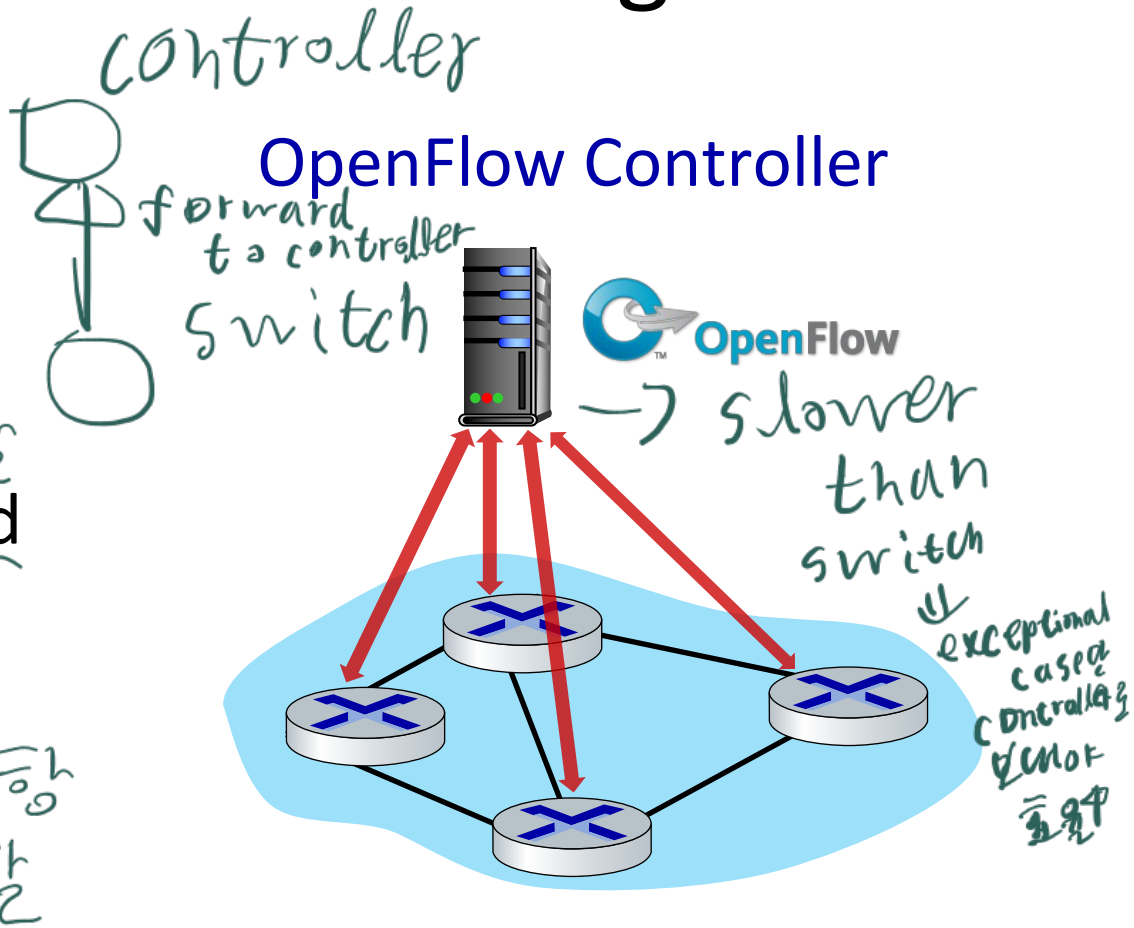
- *features:* controller queries switch features, switch replies
- *configure:* controller queries/sets switch configuration parameters
- *modify-state:* add, delete, modify flow entries in the OpenFlow tables
- *packet-out:* controller can send this packet out of specific switch port



OpenFlow: switch-to-controller messages

Key switch-to-controller messages

- *packet-in:* transfer packet (and its control) to controller. See packet-out message from controller → packet 전달
- *flow-removed:* flow table entry deleted at switch → flow table 삭제
- *port status:* inform controller of a change on a port. → port 변경 사항 전달

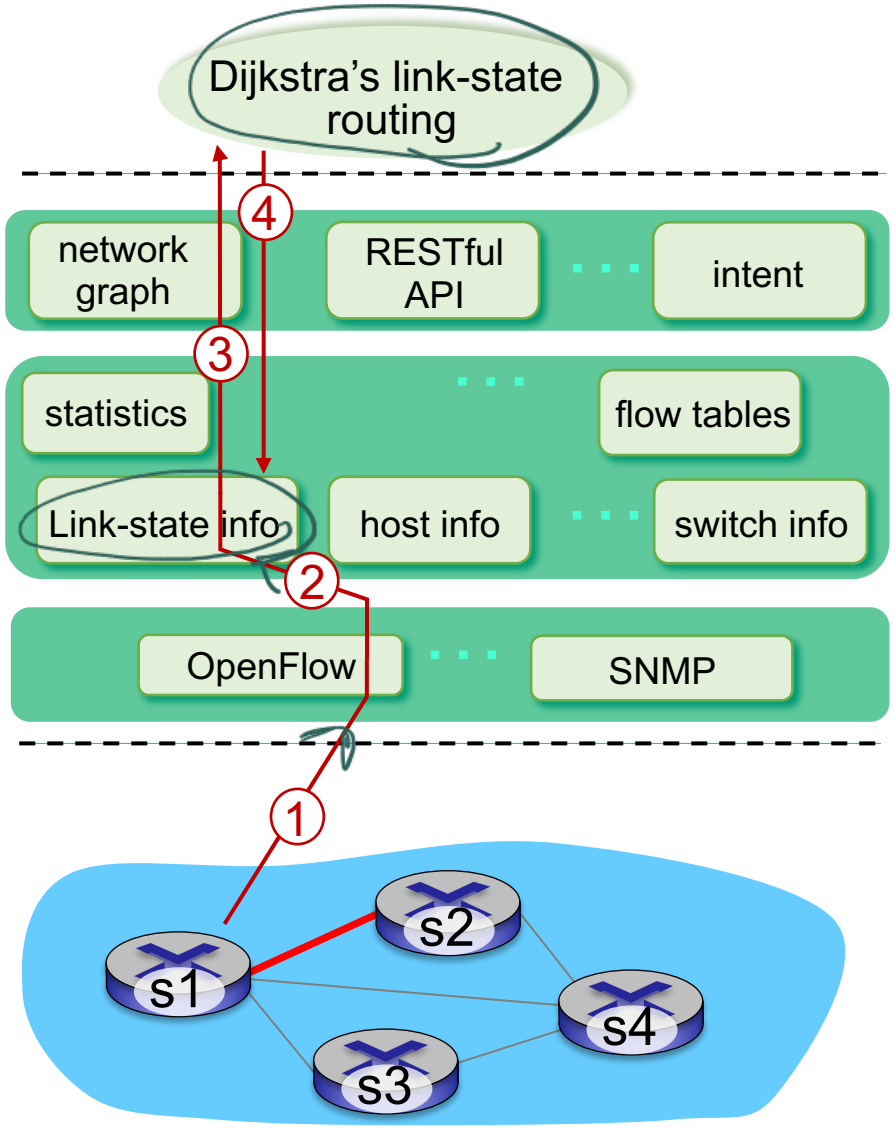


OpenFlow Controller

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

→ 메시지를 직접 생성하는 대신 더 추상화 된 계에서 프로그래밍

SDN: control/data plane interaction example



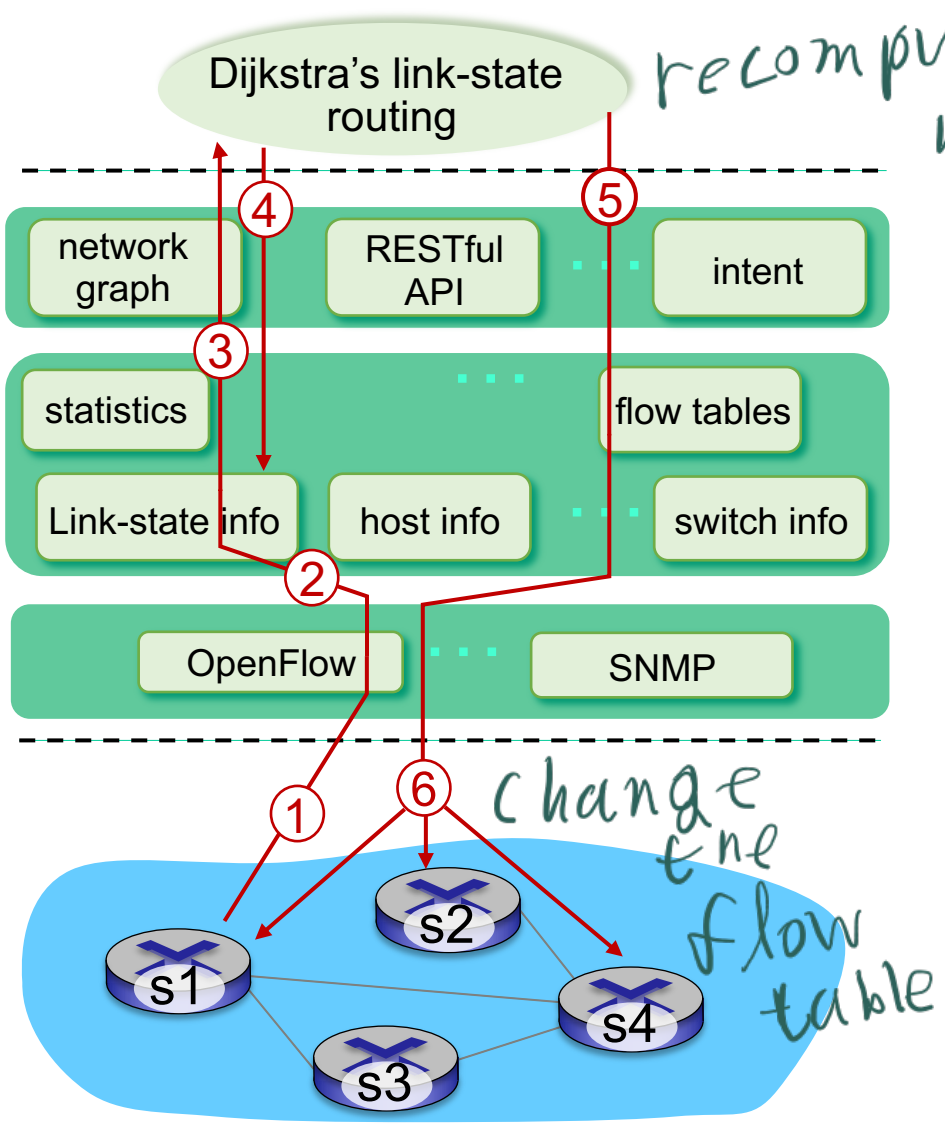
① S1, experiencing link failure uses OpenFlow port status message to notify controller link 끊김 통보

② SDN controller receives OpenFlow message, updates link status info SDN에서 받은 후 link 상태 업데이트

③ Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called. link 상태가 변경되었으므로 Dijkstra 호출

④ Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes Dijkstra에서 network 정보에 접근해 새로운 경로 계산

SDN: control/data plane interaction example



recompute new route

change the flow table

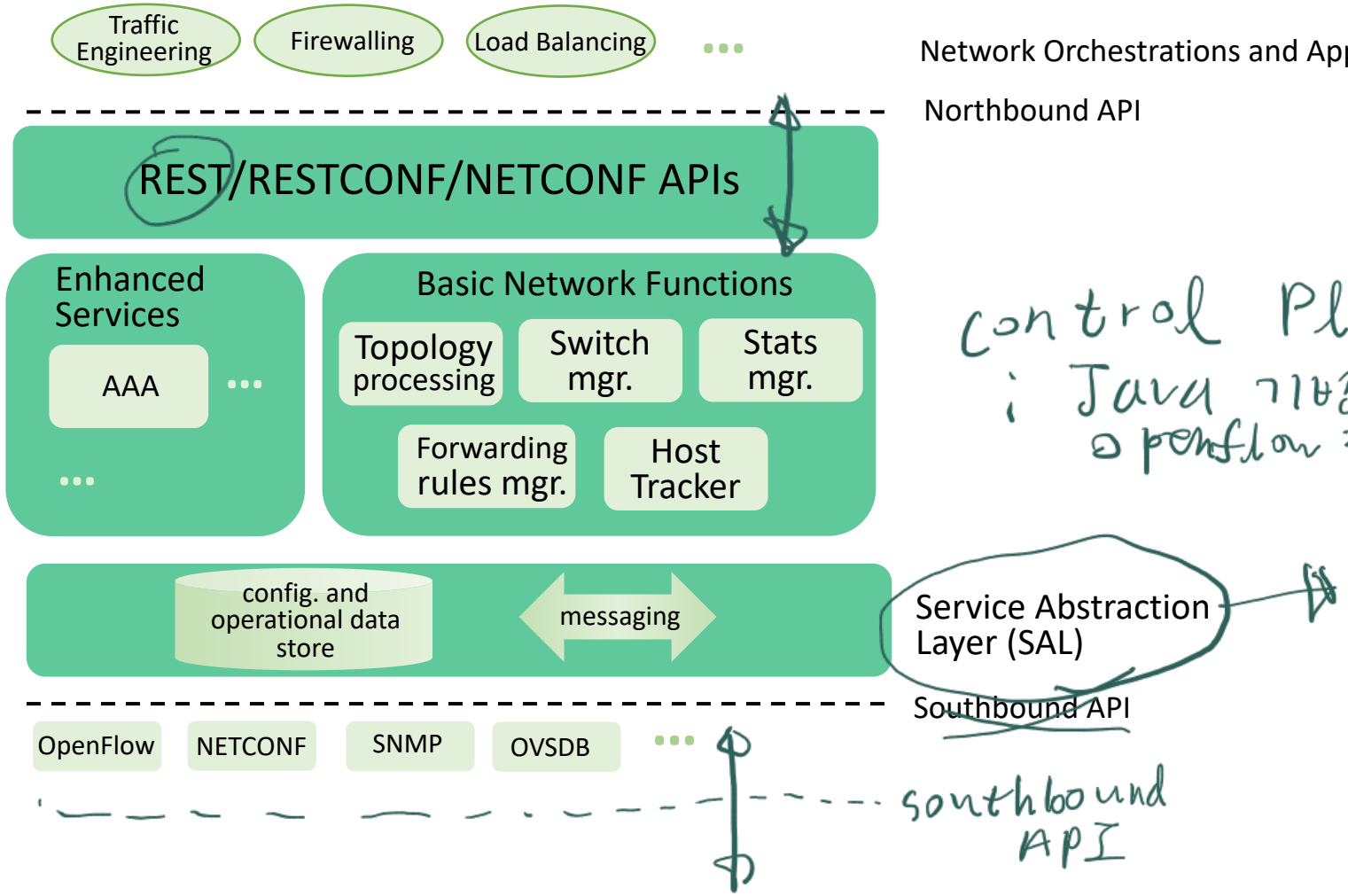
⑤ link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed

routing app이 SDN의 flow-table과 상호작용해서 새로운 flow-table 생성

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

openflow를 이용해서 새로운 table을 라우터에 update

OpenDaylight (ODL) controller



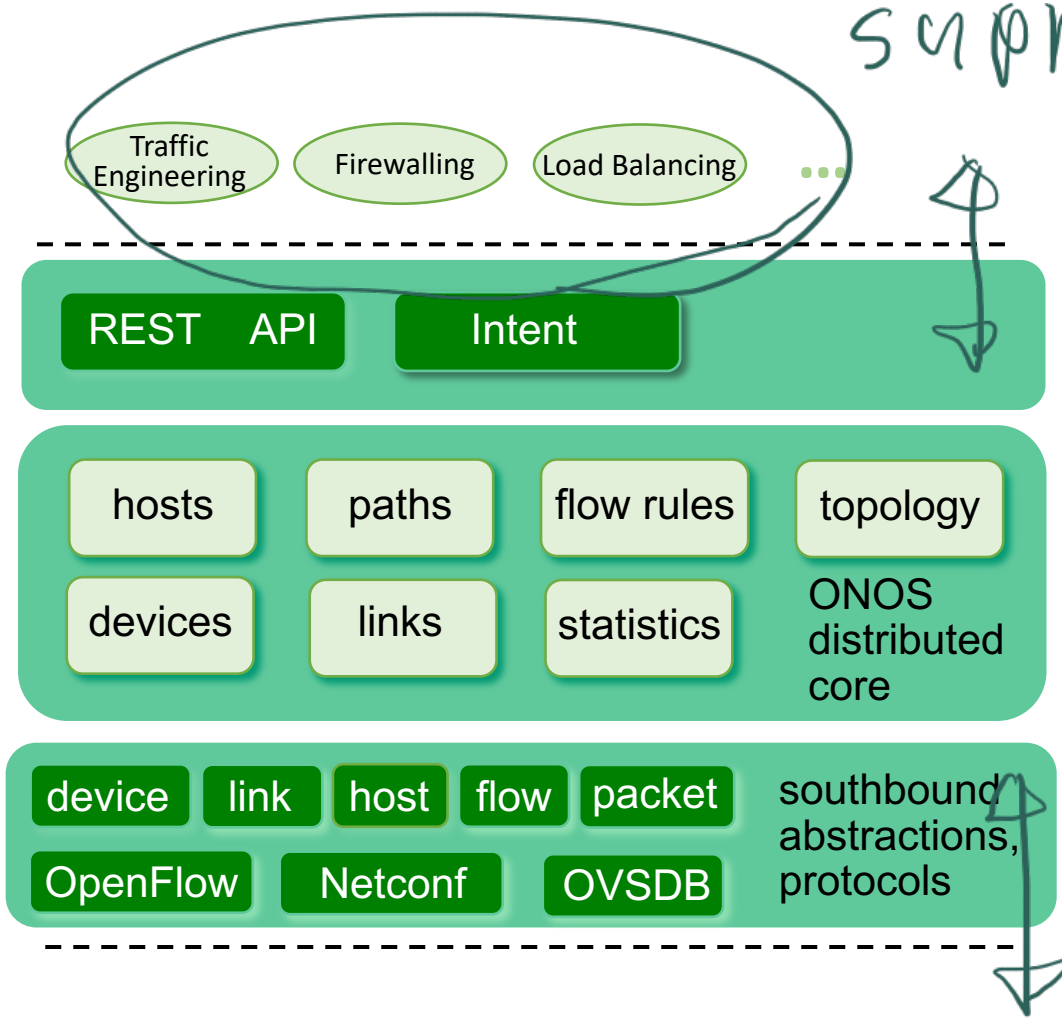
control Platform
: Java 기반,
openflow 지원

Service Abstraction Layer:

- interconnects internal, external applications and services

내부-외부 app 및 서비스 연결

ONOS controller



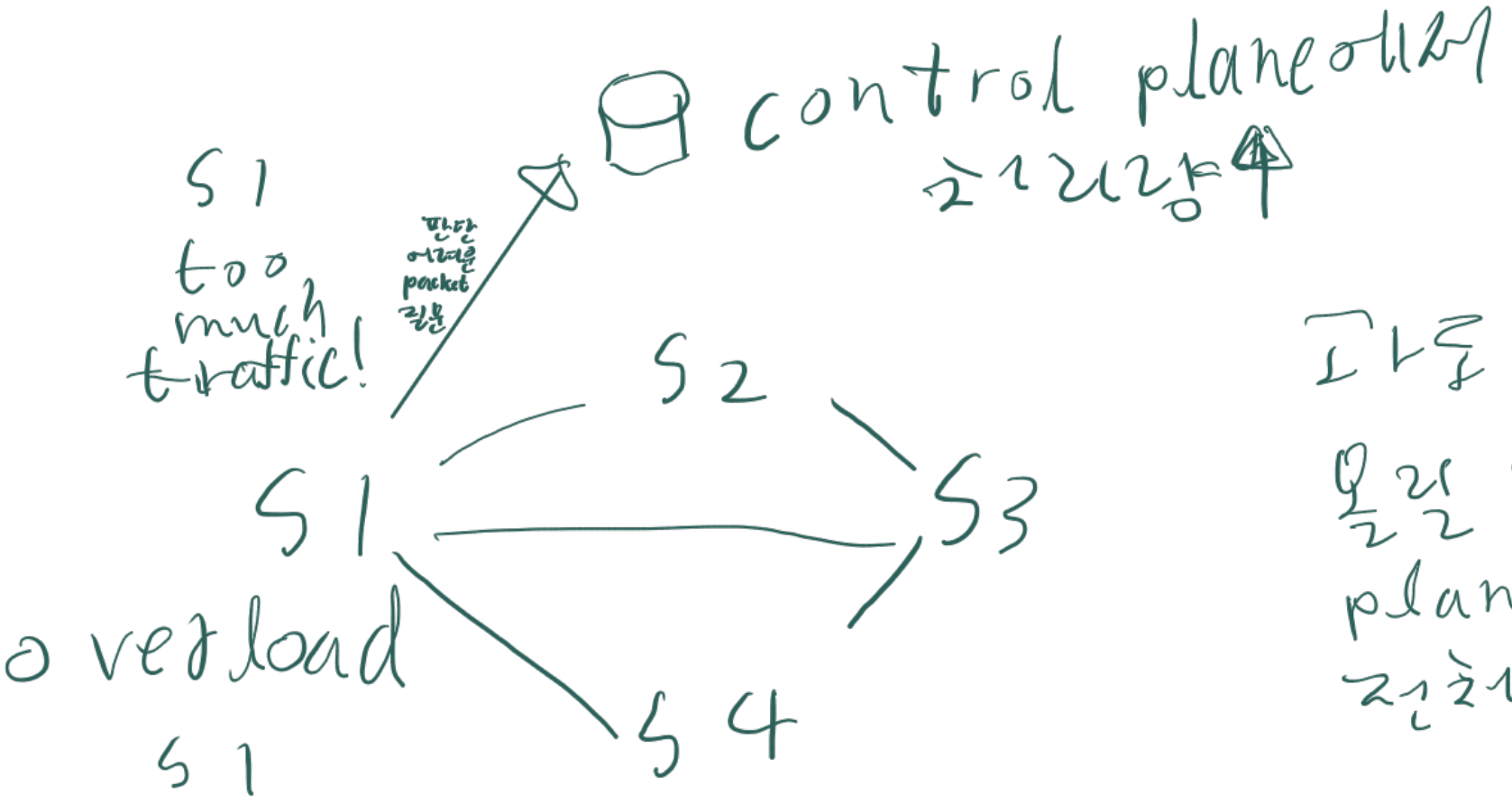
support network admin (make own network)

- 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

Is OpenFlow-based SDN silver bullet?

만경 ex) DoS → 고도화된 처리량

- What about denial-of-service attacks?



고도화된 처리량이
필요 경우, control
plane이 아닌 data
전송 레이어에

Programmable switches

P4 is control plane 프로그램 언어

by fast evolution of dataplane

패킷에서 헤더 추출

Parser Program

```
parser parse_ethernet {
    extract(ethernet);
    return switch(ethernet.ethertype) {
        0x8100 : parse_vlan_tag;
        0x0800 : parse_ipv4;
        0x8847 : parse_mpls;
        default: ingress;
    }
}
```

header

헤더 및 메타데이터 선언

Header and Data Declarations

```
header_type  ethernet_t     { … }
header_type  l2_metadata_t  { … }

header       ethernet_t     ethernet;
header       vlan_tag_t
vlan_tag[2];
metadata     l2_metadata_t  l2_meta;
```

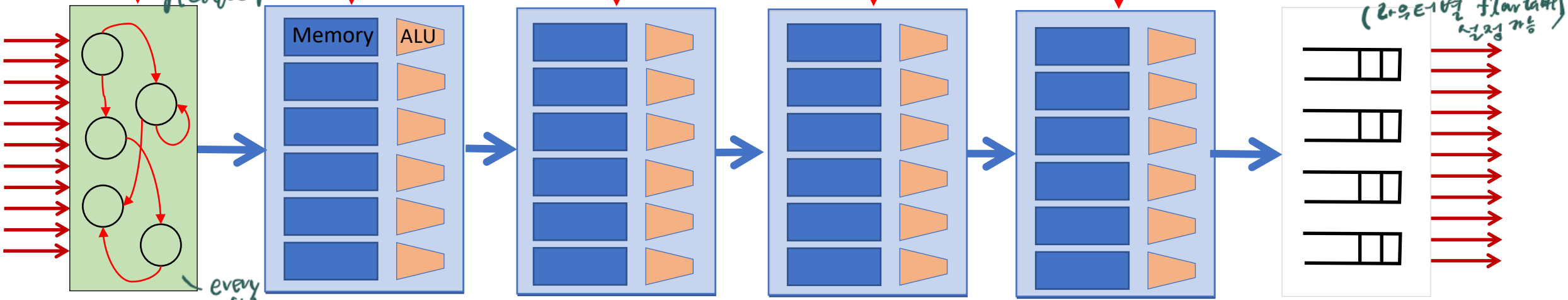
Tables and Control Flow

```
table port_table { … }

control ingress {
    apply(port_table);
    if (l2_meta.vlan_tags == 0) {
        process_assign_vlan();
    }
}
```

↳ 선언한 메타데이터와 header로 프로그래밍 (라우터별 flow table 설정 가능)

header



Programmable Parser

every thing in parser is dataplane

Programmable Match-Action Pipeline

SDN: selected challenges

- hardening the control plane: dependable, reliable, performance-scalable, secure distributed system
  - robustness to failures: leverage strong theory of reliable distributed system for control plane
  - dependability, security: "baked in" from day one?
- networks, protocols meeting mission-specific requirements
  - e.g., real-time, ultra-reliable, ultra-secure
- Internet-scaling: beyond a single AS
- SDN critical in 5G cellular networks

SDN and the future of traditional network protocols

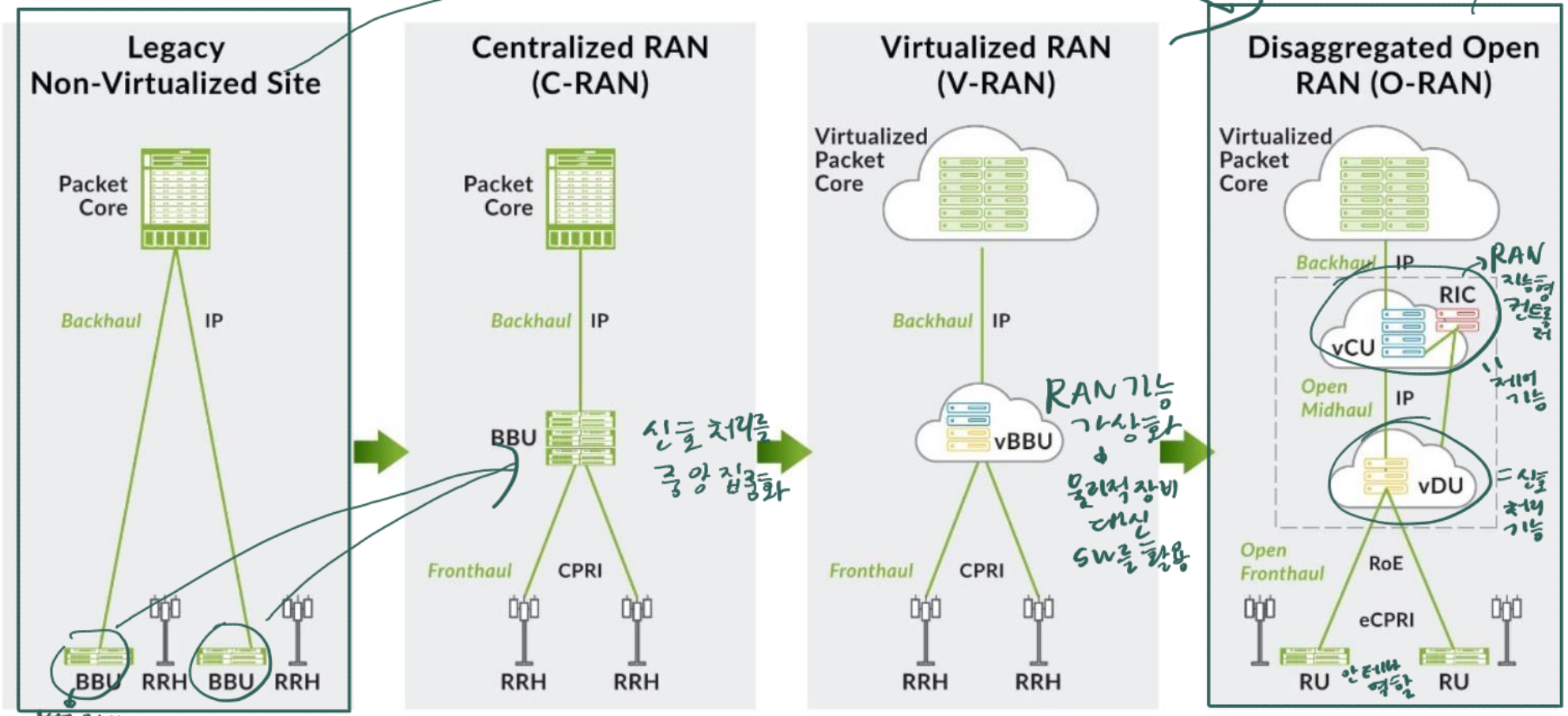
- SDN-computed versus router-computer forwarding tables:
  - just one example of logically-centralized-computed versus protocol computed
- one could imagine SDN-computed congestion control:
  - controller sets sender rates based on router-reported (to controller) congestion levels

How will implementation of network functionality (SDN versus protocols) evolve?

new trands in cellular network

Example: Open RAN (O-RAN) in 5G/6G

다양한 공급업체 선택가능



RRH = Remote Radio Head
BBU = Baseband Unit
CPRI = Common Private Radio Interface

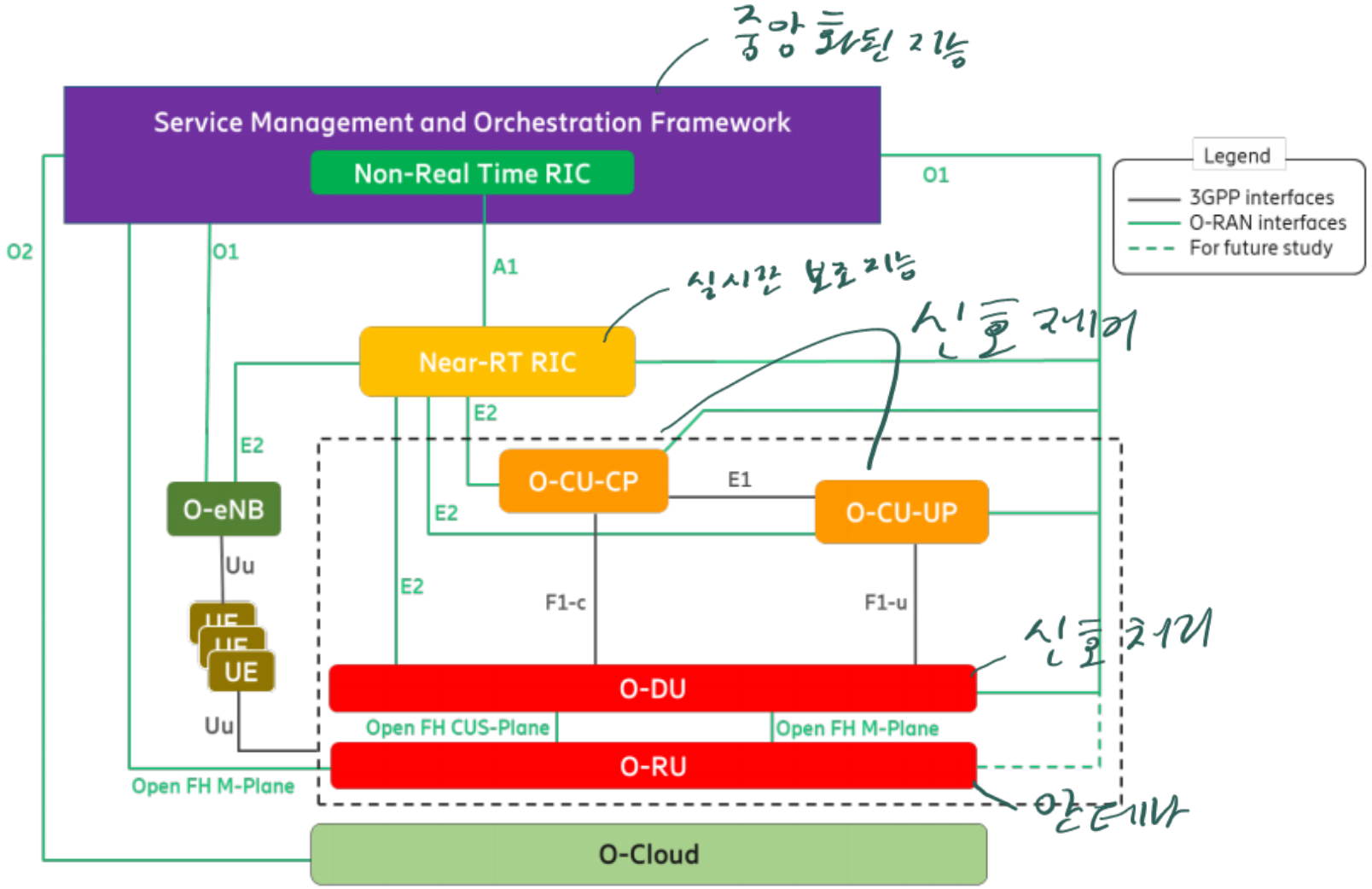
RIC = RAN Intelligent Controller
CU = Centralized Unit
DU = Distributed Unit

RU = Remote Unit
RoE = Radio over Ethernet
eCPRI = Ethernet CPRI

독점적 인터페이스인 RRH, BBU 대신 RC, DU, CU 등 여러기능과 이들간 개방형 interface로 세분화 RIC로 인텔리전스 추가

[source: : https://www.juniper.net/kr/ko/research-topics/what-is-open-ran.html]

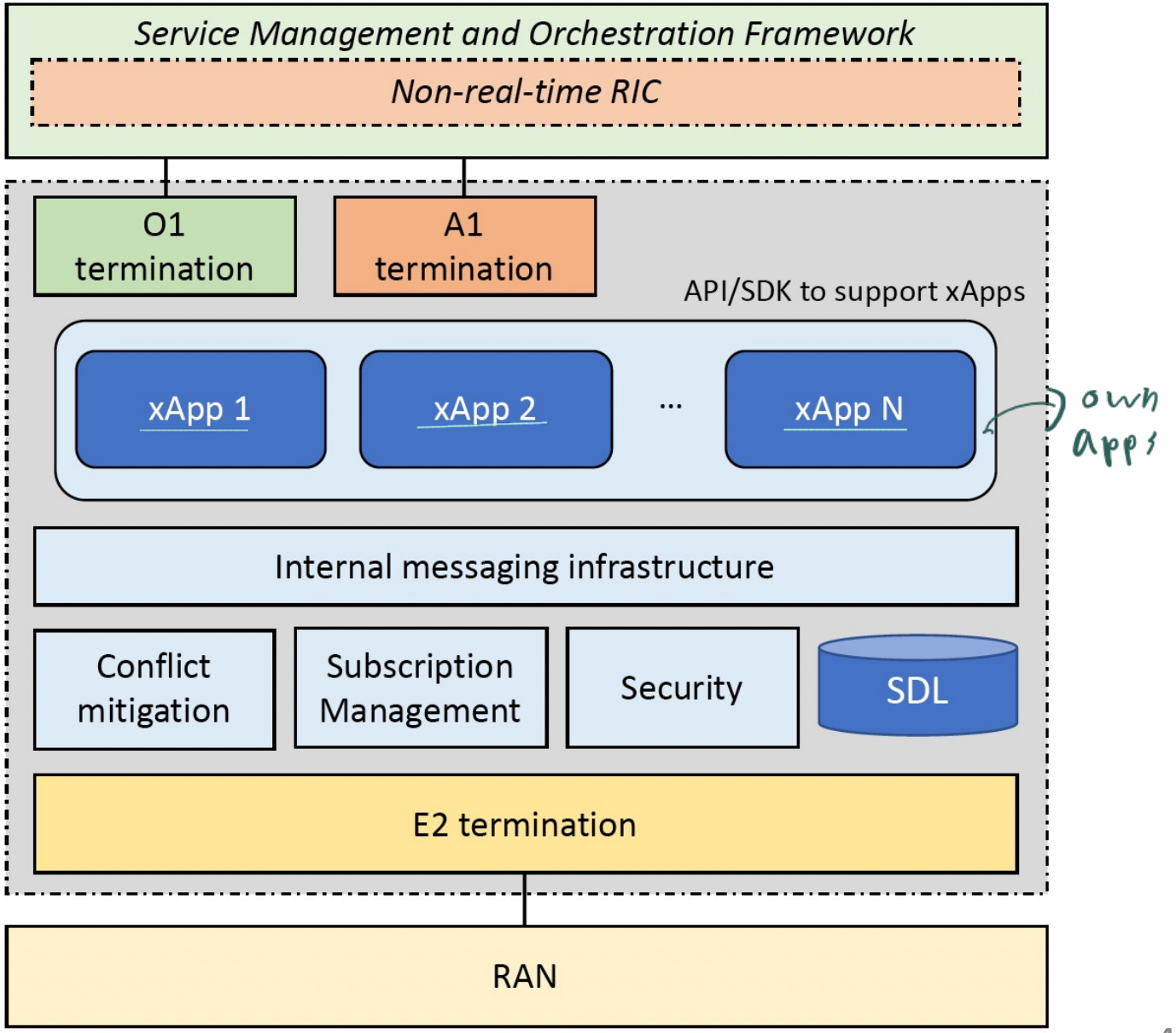
O-RAN Architecture



[source: : O-RRAN Alliance, O-RAN.WG1.O-RAN-Architecture-Description-v06.00]

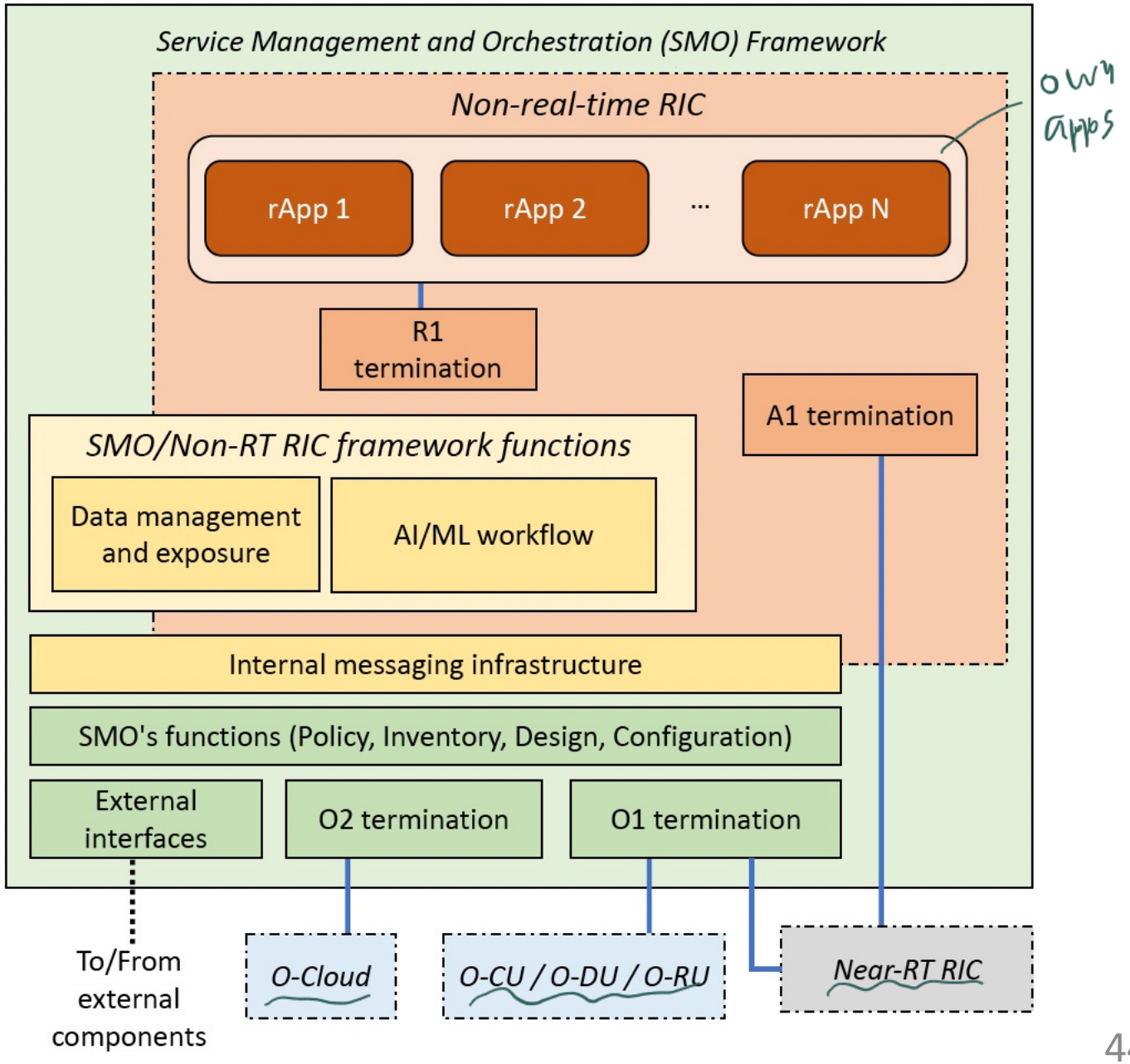
O-RAN

- Near-RT RIC architecture



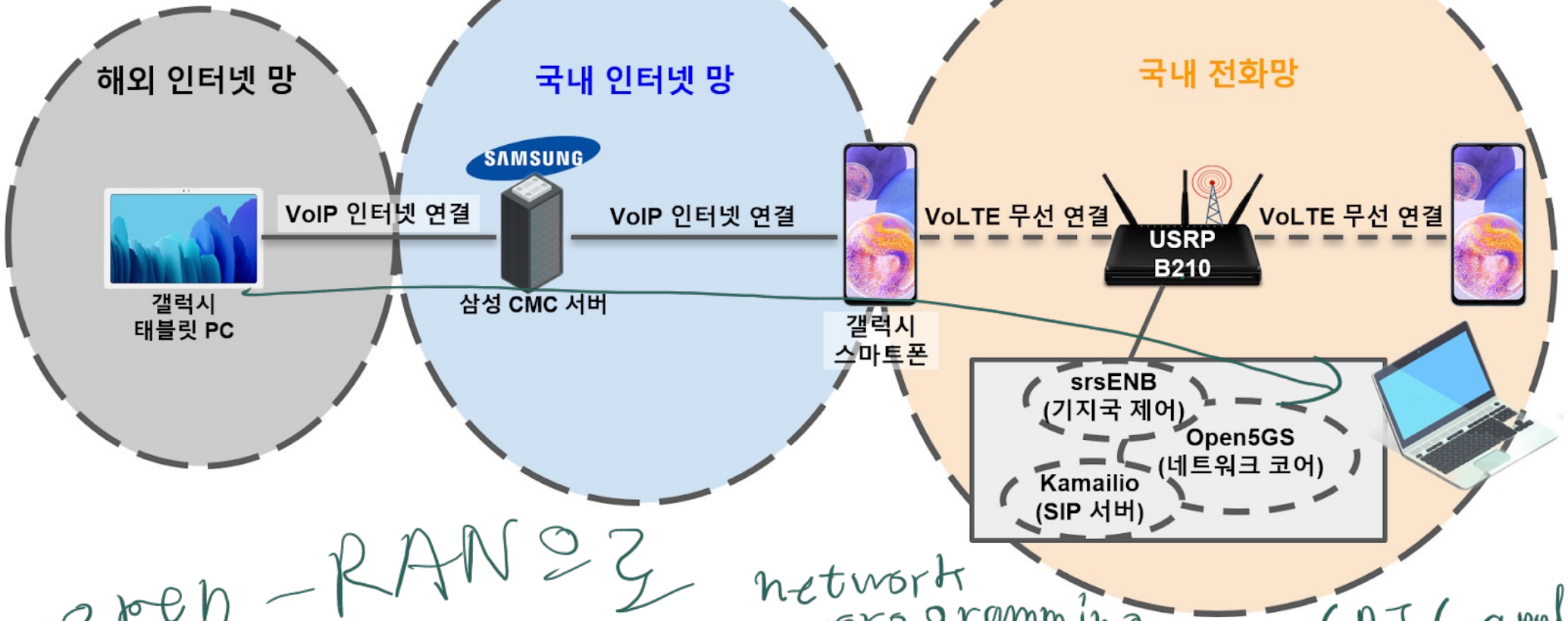
O-RAN

- Non-RT RIC architecture



Another example: tracking voice phishers

보이스 피싱 역추적



open-RAN 오픈

network programming 기능 (RIC application)

Next…

- *Chapter 6.1 Introduction to the Link Layer*
- *Chapter 6.2 Error-Detection and -Correction Techniques*