

# Computer Networks

## COL 334/672

Software Defined Networking

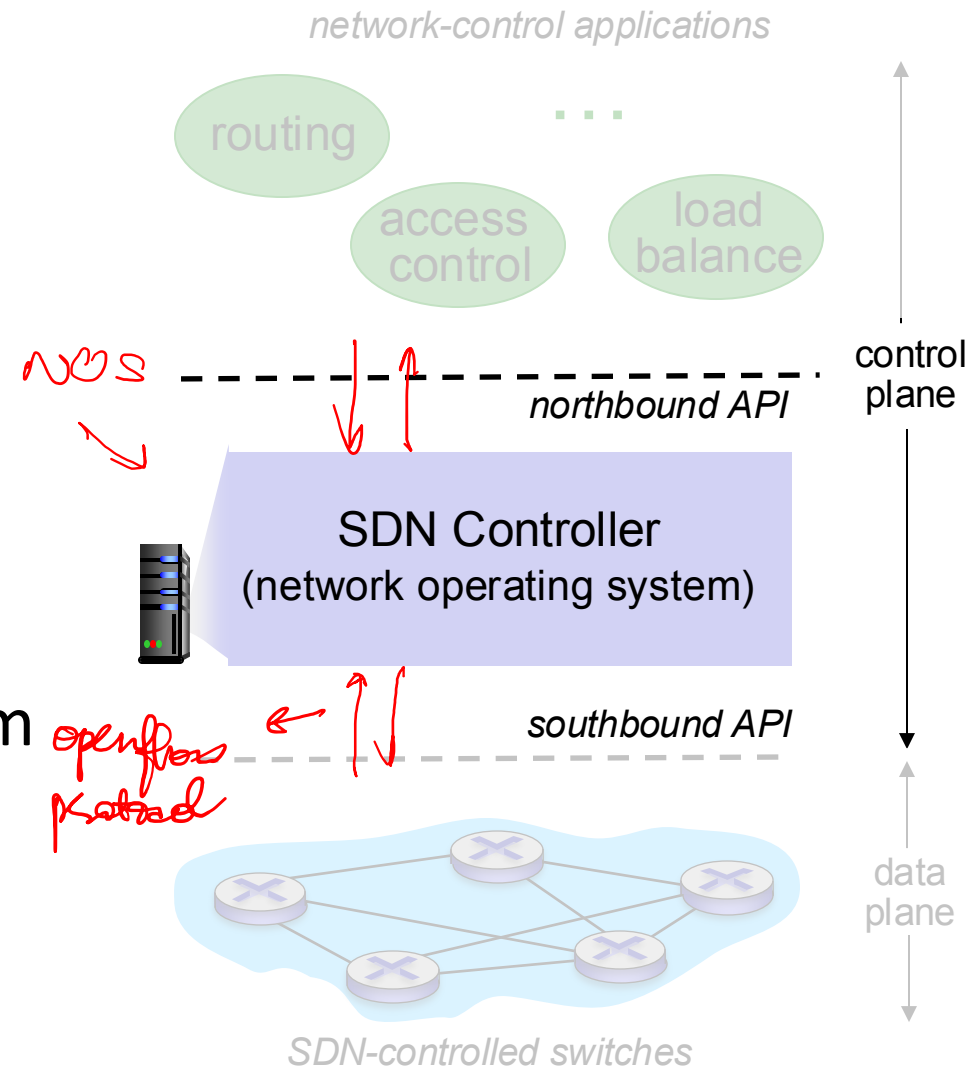
Tarun Mangla

*Slides adapted from KR*

Sem 1, 2024-25

# Recap: SDN Controller

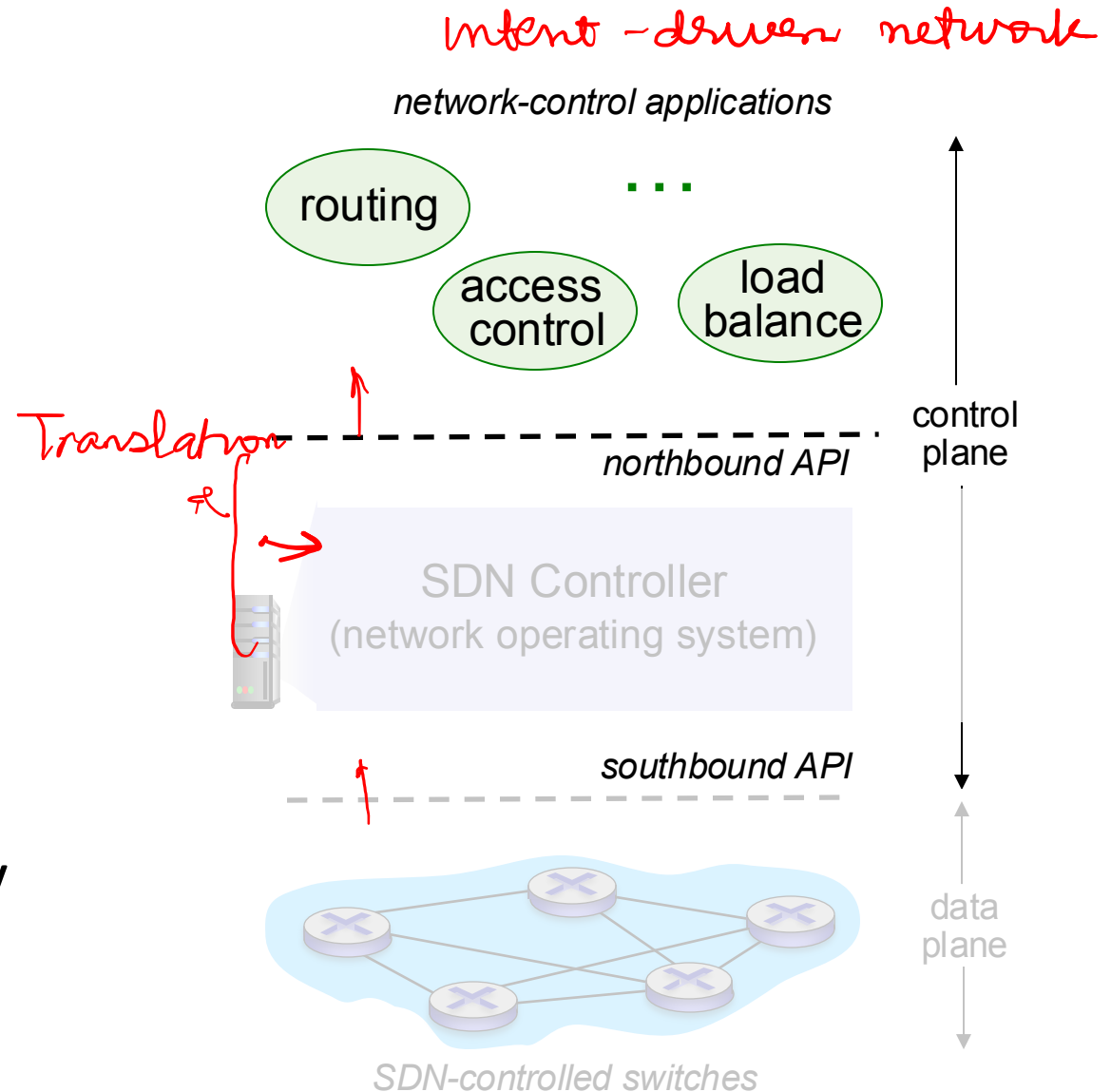
- maintains network state information
- interacts with network control applications “above” via northbound API
- interacts with network switches “below” via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



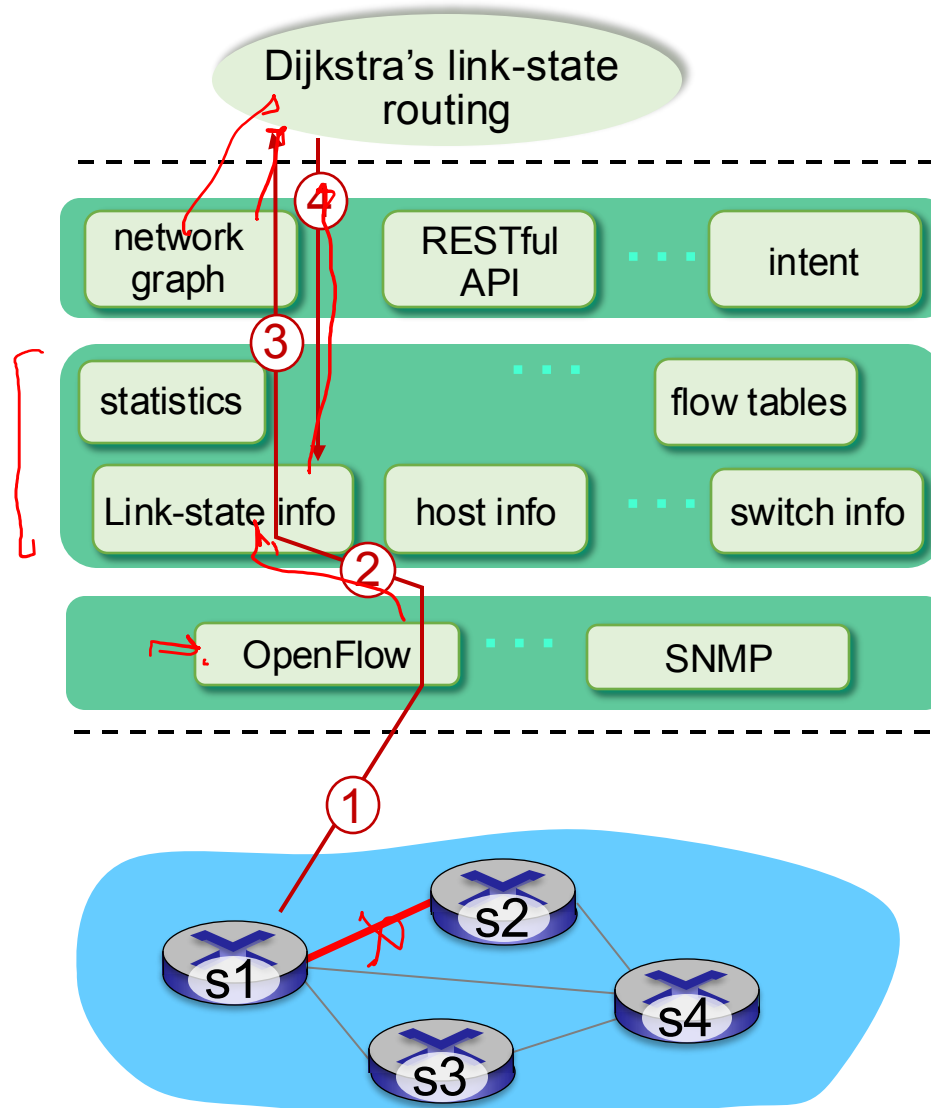
protocol → application

# Software defined networking (SDN)

- operators don't "program" switches by creating/sending OpenFlow messages directly.
- Instead use higher-level abstraction at controller
- "brains" of control: implement control functions using lower-level services, API provided by SDN controller
- *unbundled*: can be provided by 3<sup>rd</sup> party: distinct from routing vendor, or SDN controller

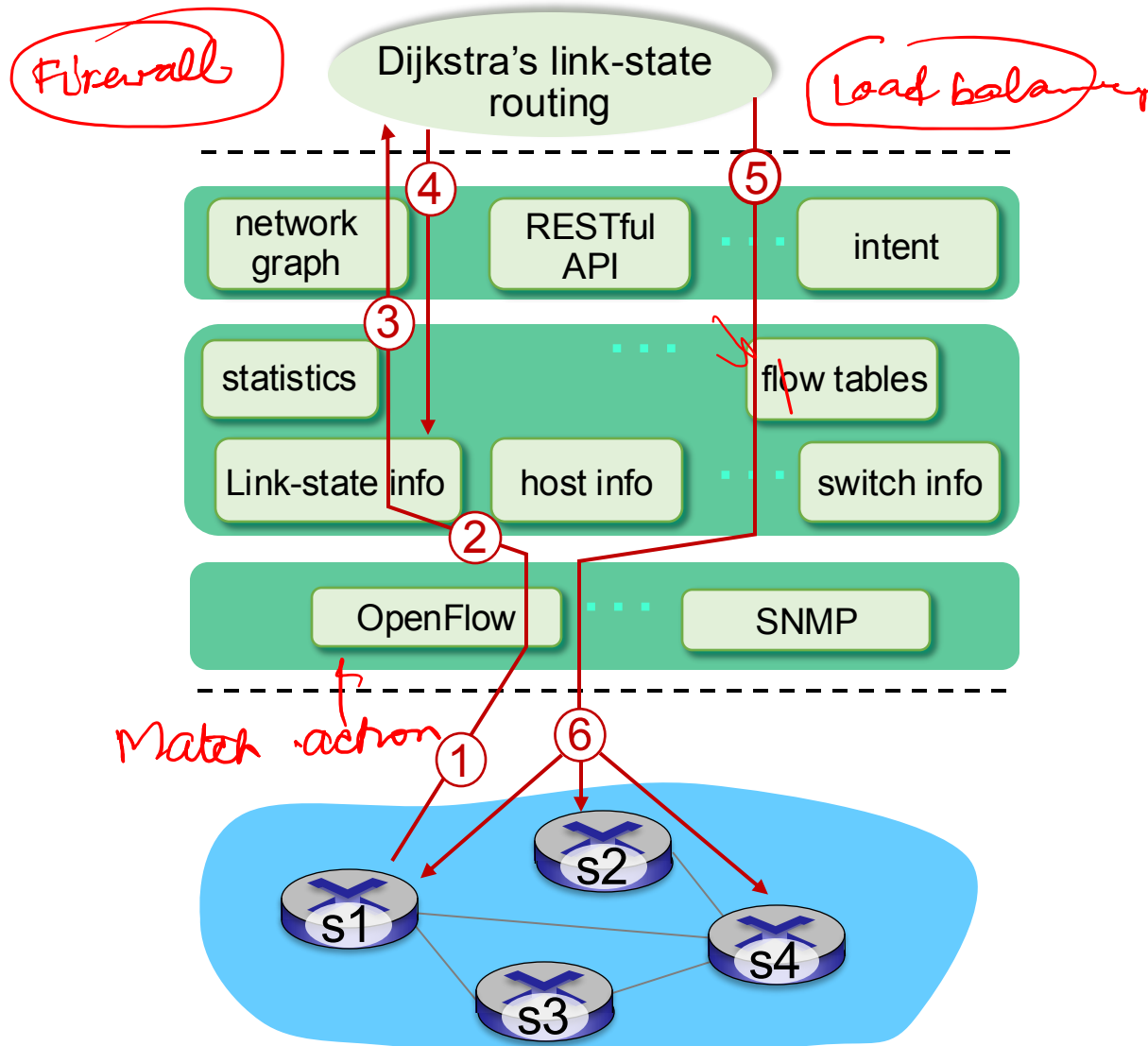


# SDN: control/data plane interaction example



- 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

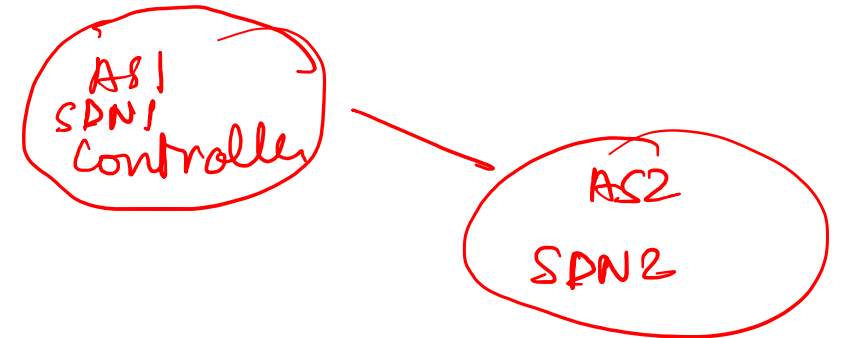
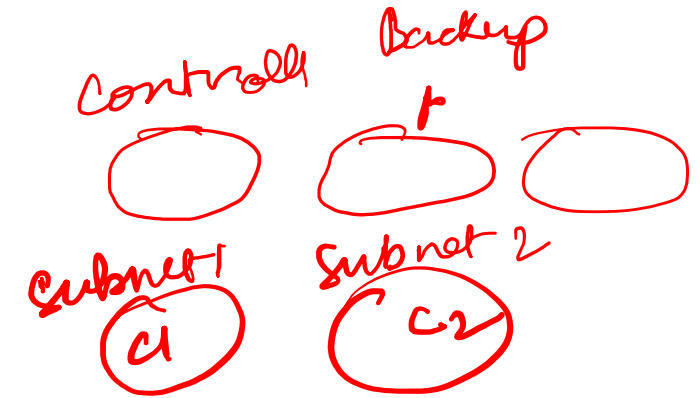


- ⑤ link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed
- ⑥ controller uses OpenFlow to install new tables in switches that need updating

# SDN: Key Challenges

## ■ Hardening the control plane

- Scalability
  - Reliability
  - Consistency
  - Security
- Real-time → logically centralized
- Distributed systems



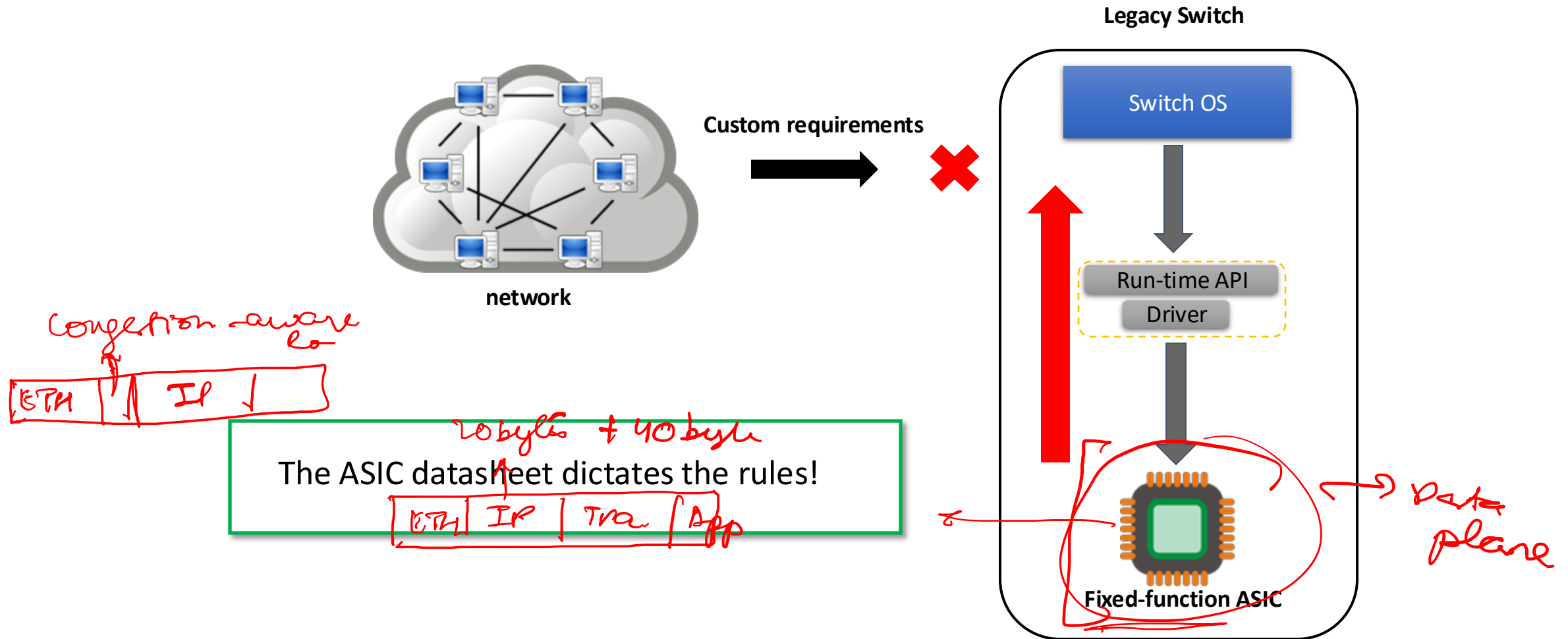
## ■ Internet-scaling: beyond a single AS (?)

# What else is programmable in the network?

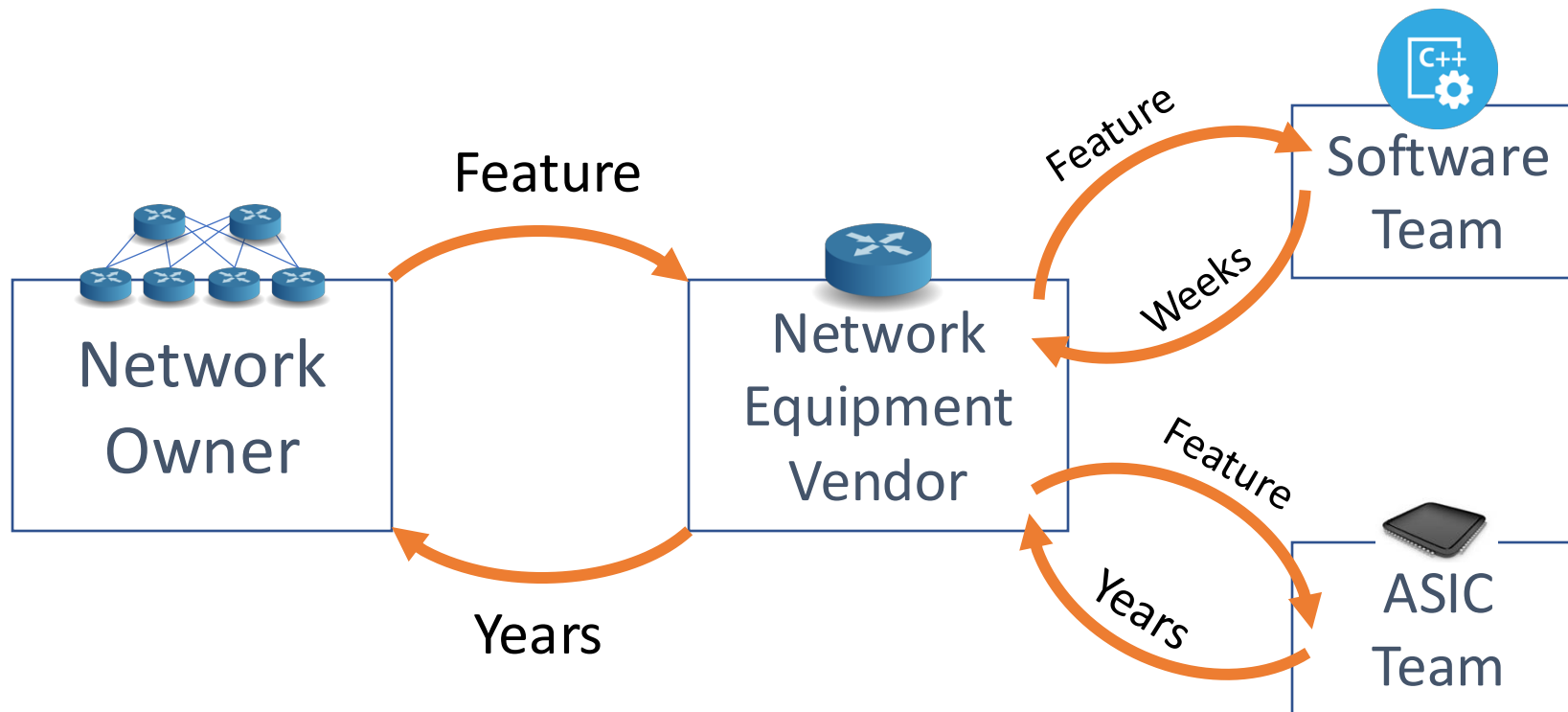
- Programmable data plane }
- Network function virtualization (NFV)

↓  
making middleboxes programmable

# Status Quo: Bottom-up design



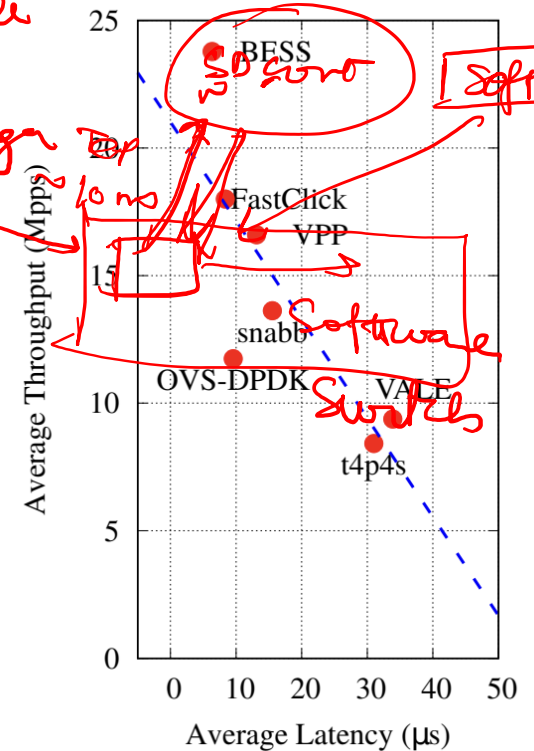
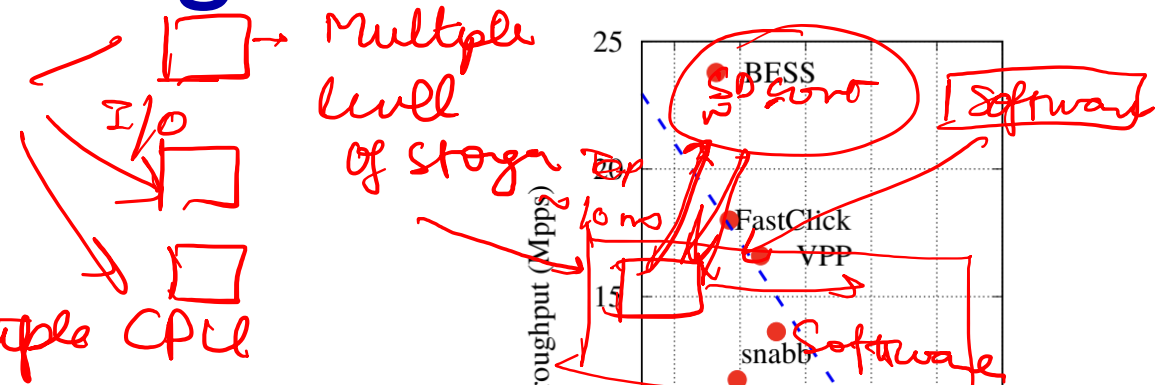
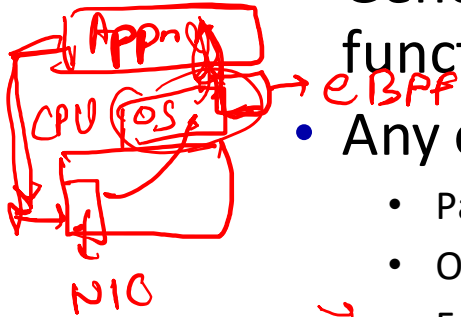




# How to make Data Plane Programmable?

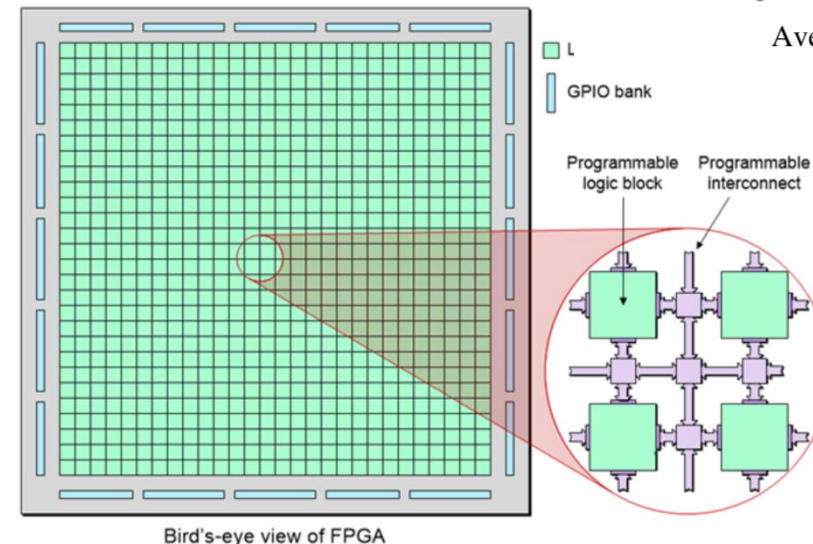
## ■ Move the data plane to software

- Generally, too slow for data plane functions
- Any optimization techniques?
  - Parallelism across multiple servers and cores
  - Optimizations in NUMA
  - Fast I/O ...



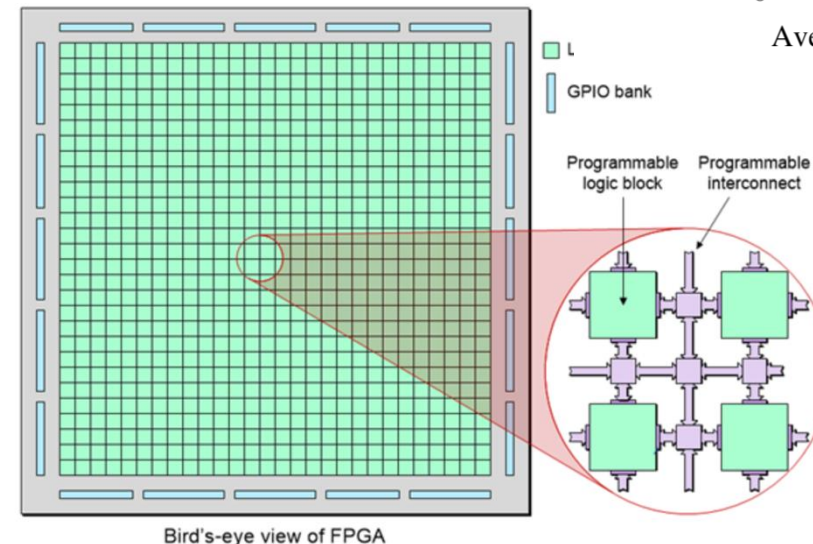
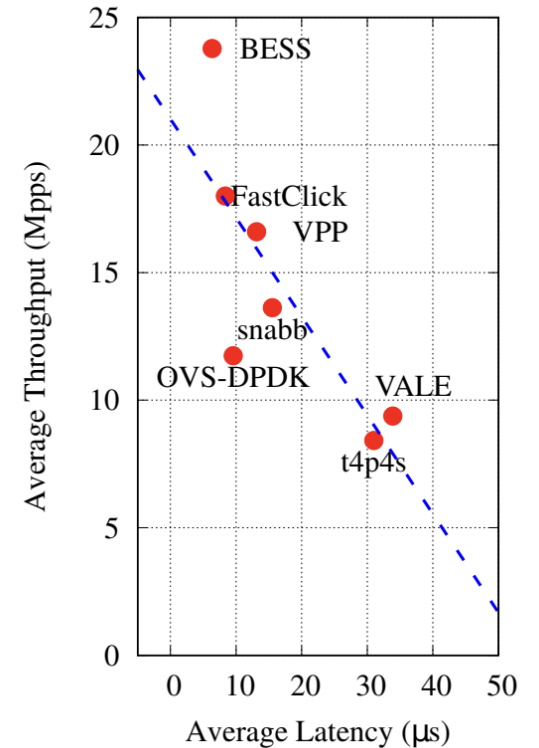
## ■ What about programmable hardware?

- FPGA: but costly, power-hungry, slower



# How to make Data Plane Programmable?

- Move the data plane to software
  - Generally, too slow for data plane functions
  - Any optimization techniques?
    - Parallelism across multiple servers and cores
    - Optimizations in NUMA
    - Fast I/O ...
- What about programmable hardware?
  - FPGA: but costly, power-hungry, slower

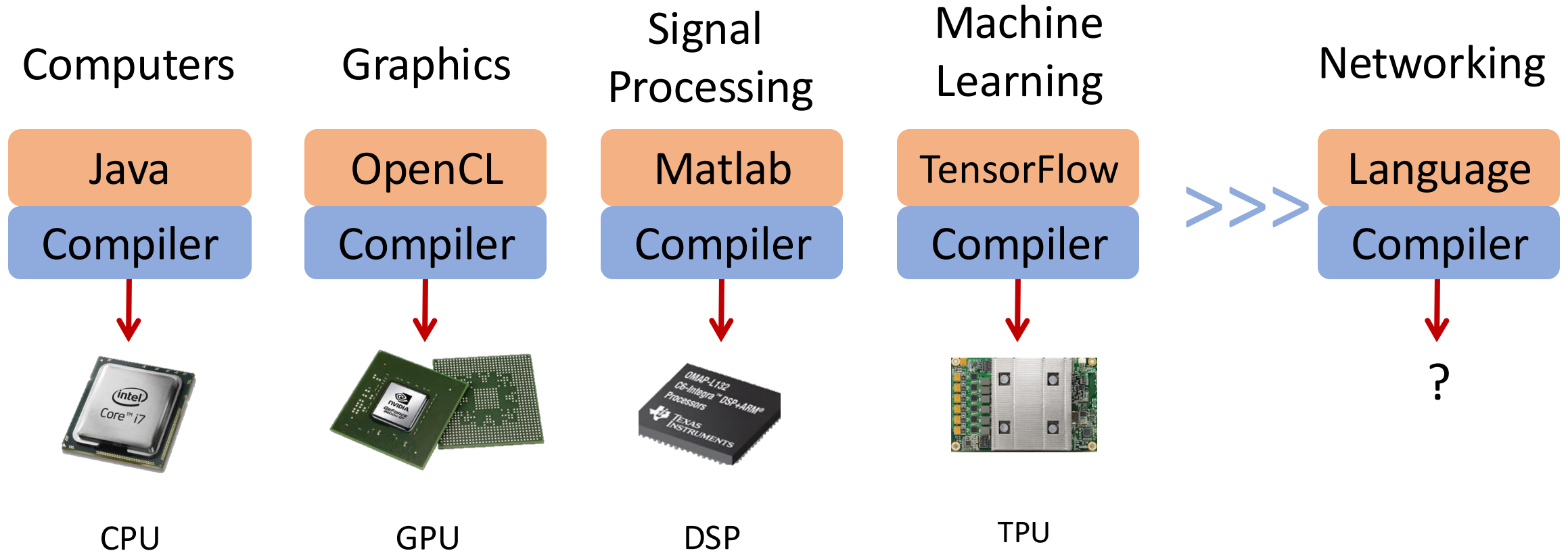


# What about Programmable Hardware?

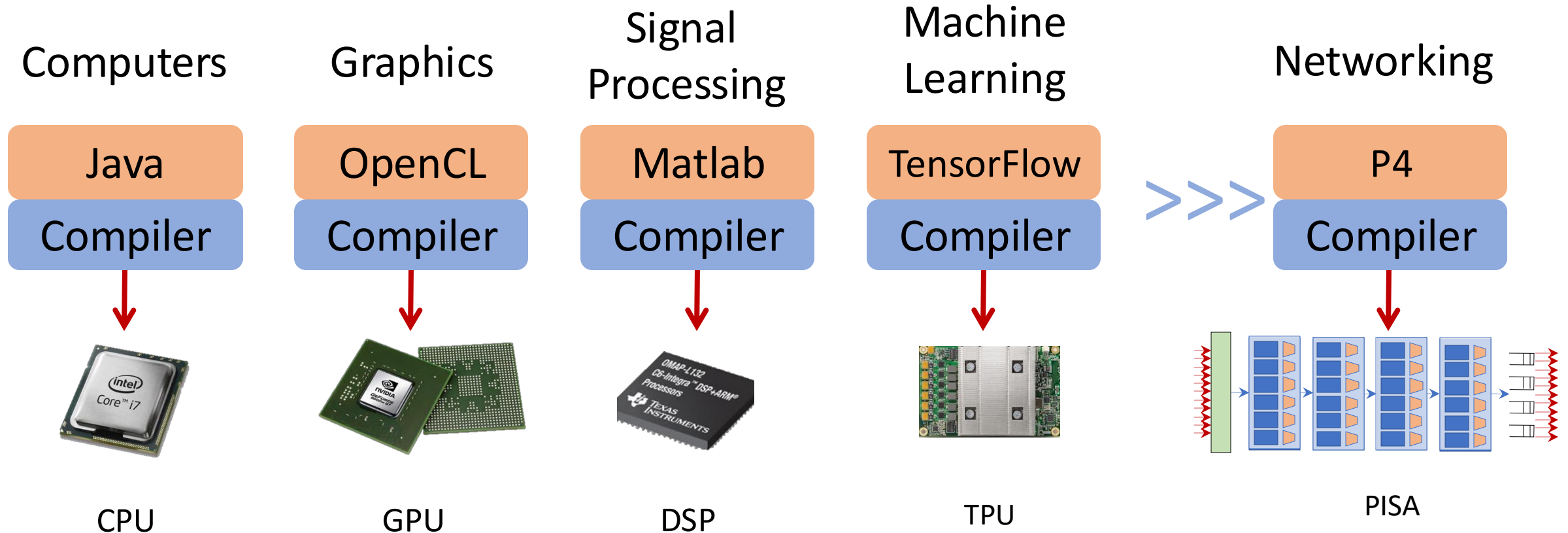
**“Programmable switches are 10-100x slower than fixed-function switches. They are more expensive and consume more power.”**

Conventional wisdom in networking

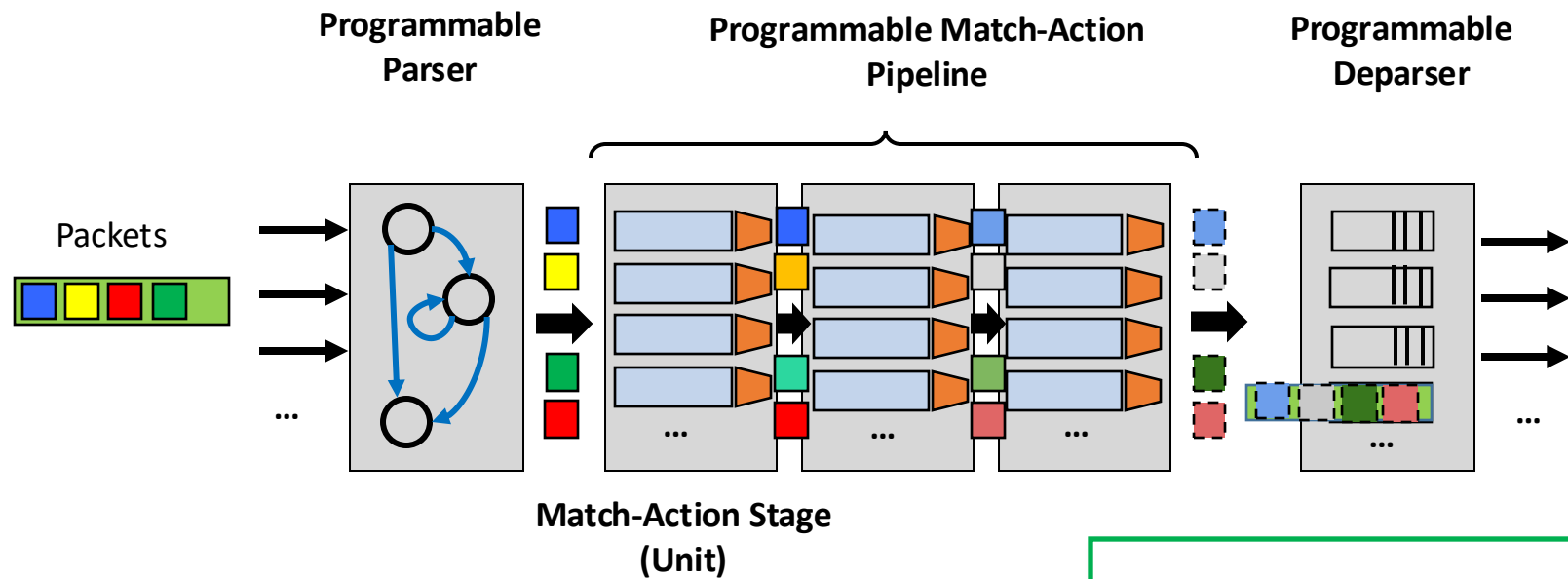
# Domain Specific Processors



# Domain Specific Processors



# Protocol Independent Switch Architecture



*P4 describes PISA's behavior*

# Reducing complexity

switch.p4

Switch OS

## IPv4 and IPv6 routing

- Unicast Routing
  - Routed Ports & SVI
  - VRF
- Unicast RPF
  - Strict and Loose

## ~~Multicast~~

- ~~- PIM-SM/DM & PIM-Bidir~~

## Ethernet switching

- ~~- VLAN Flooding~~
- MAC Learning & Aging
- STP state
- ~~- VLAN Translation~~

## Load balancing

- ~~- LAG~~
- ECMP & WCMP
- Resilient Hashing
- ~~- Flowlet Switching~~

## Fast Failover

- LAG & ECMP

## Tunneling

- IPv4 and IPv6 Routing & Switching
  - ~~- IP in IP (Gin4, 4in4)~~
  - VXLAN, NVGRE, GENEVE & GRE
  - ~~- Segment Routing, ILA~~

## ~~MPLS~~

- ~~- LER and LSR~~
- ~~- IPv4/v6 routing (L3VPN)~~
- ~~- L2 switching (EoMPLS, VPLS)~~
- ~~- MPLS over UDP/GRE~~

## ACL

- MAC ACL, IPv4/v6 ACL, RACL
- ~~- QoS ACL, System ACL, PBR~~
- Port Range lookups in ACLs

## QoS

- QoS Classification & marking
- ~~- Drop profiles/WRED~~
- ~~- RoCE v2 & FCoE~~
- CoPP (Control plane policing)

## ~~NAT and L4 Load Balancing~~

## Security Features

- ~~- Storm Control, IP Source Guard~~

## Monitoring & Telemetry

- ~~- Ingress Mirroring and Egress Mirroring~~
- Negative Mirroring
- ~~- Sflow~~
- INT

## Counters

- Route Table Entry Counters
- ~~- VLAN/Bridge Domain Counters~~
- Port/Interface Counters

## Protocol Offload

- BFD, OAM

## Multi-chip Fabric Support

- ~~- Forwarding, QoS~~

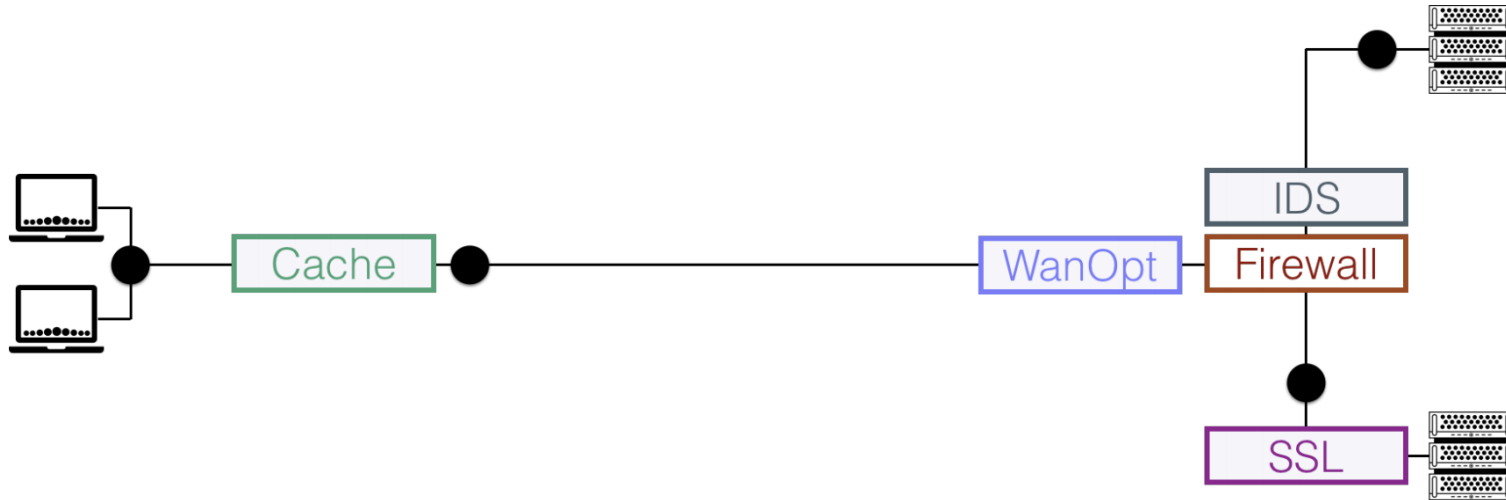


# What else is programmable in the network?

- Programmable data plane
- **Network function virtualization**

# Middleboxes

Data delivery is not the only required functionality.

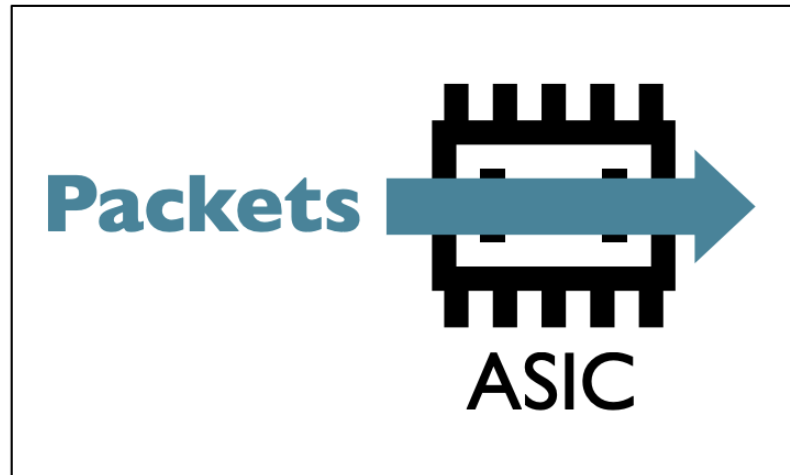


Elements in the network path for security, performance enhancements etc.

One-third of all network devices in enterprises are middleboxes!  
Sherry et al., SIGCOMM'12


# Evolution of Middleboxes

Dedicated hardware

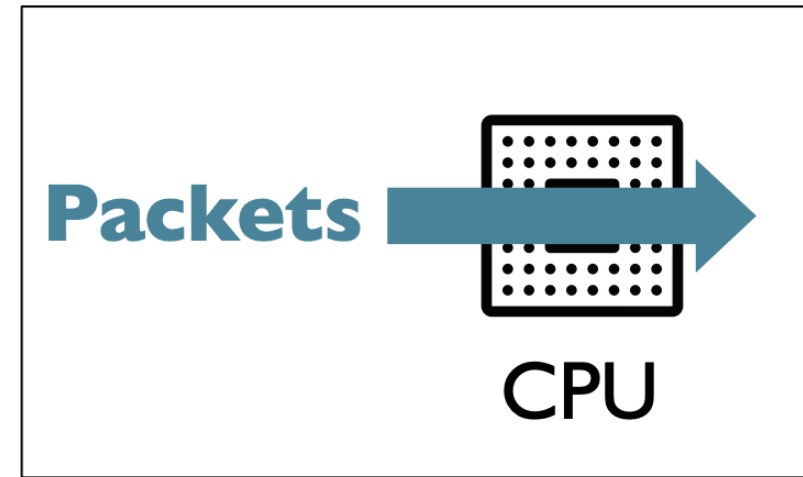


Middleboxes

*Need for  
flexibility*

A black arrow pointing from the left diagram to the right diagram, indicating the transition from dedicated hardware to software.

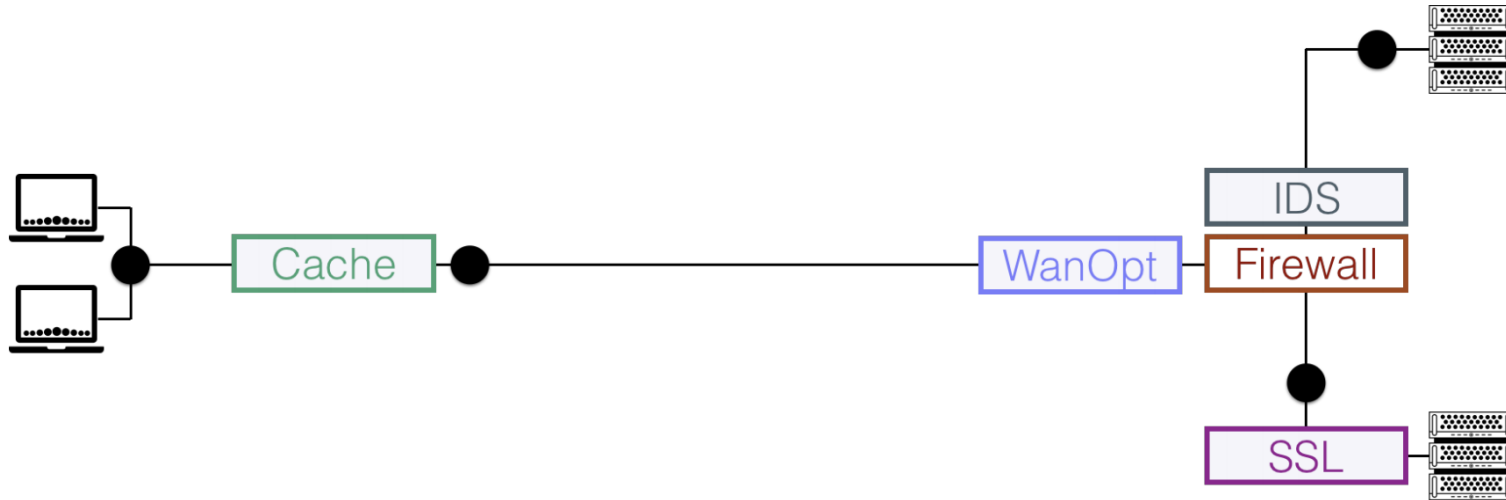
Software



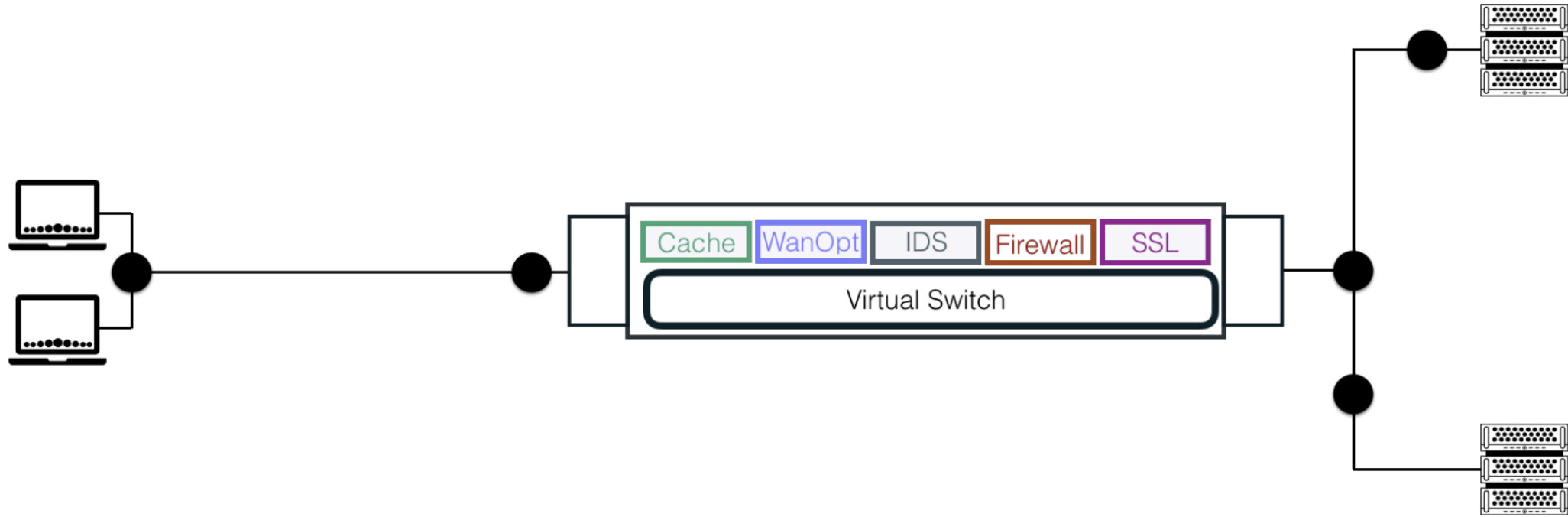
Network functions

# From Hardware Middleboxes..

Data delivery is not the only required functionality.



# To Software Network Functions (NF)



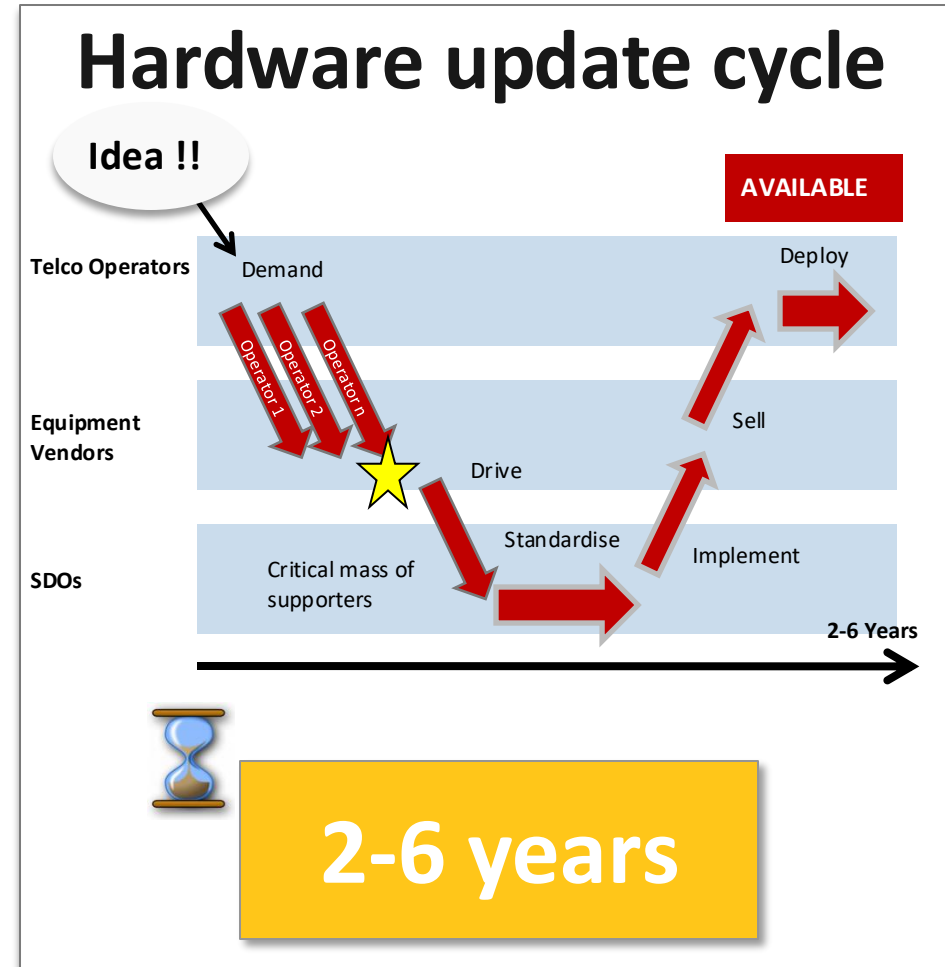
Primarily deployed in a VM  
**(Network Function Virtualization or NFV)**

# Functional Elements, not Middleboxes

- **WAN Optimizer** = Caching + Deduplication + Compression + Encryption + Forward Error Correction + Rate Limiter
- **Application Firewall** = IP Defragmenter + Application Detection Engine + Logger + Blocker
- **IDS** = IP Defragmenter + Preprocessing + Misuse Detection Engine + Logger

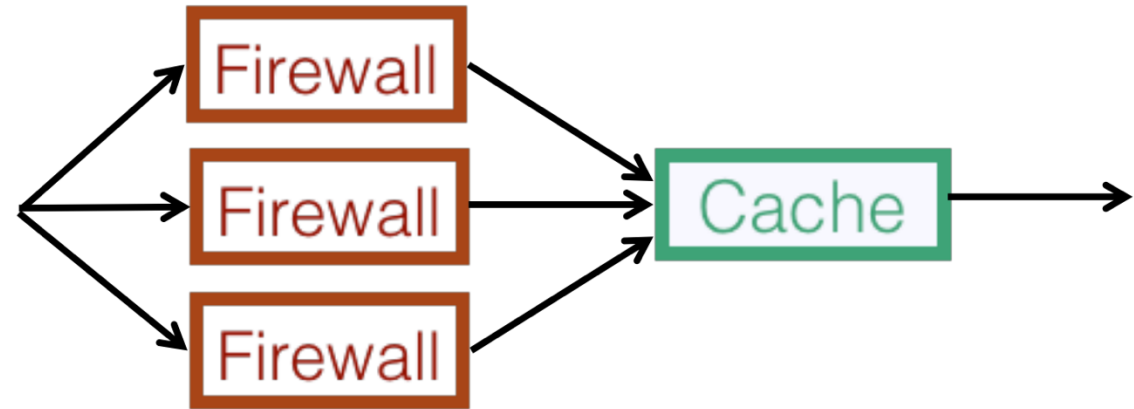
# Why NFV?

- Softwarization leads to faster innovation



# Why NFV?

- Softwarization leads to faster innovation
- Ease of deployment, configuration, and management
- Consolidation: Reduce number of hardware boxes in the network



Being adopted by both carriers and cloud providers



# NFV Challenges

- Virtual network function management
  - Where and how to install network functions?
- Unpredictable (low) Performance
  - How to mitigate the overheads of virtualization?
- Fault Tolerance
  - How to handle recovery in case of faults?

# Summary

- Increased programmability in the networks
  - Greater flexibility → Faster innovation
- Programmable control plane
  - SDN
- Programmable data plane
  - SDN-2 / P4
- Software middleboxes implemented in VMs
  - Network function virtualization (NFV)

# Attendance

