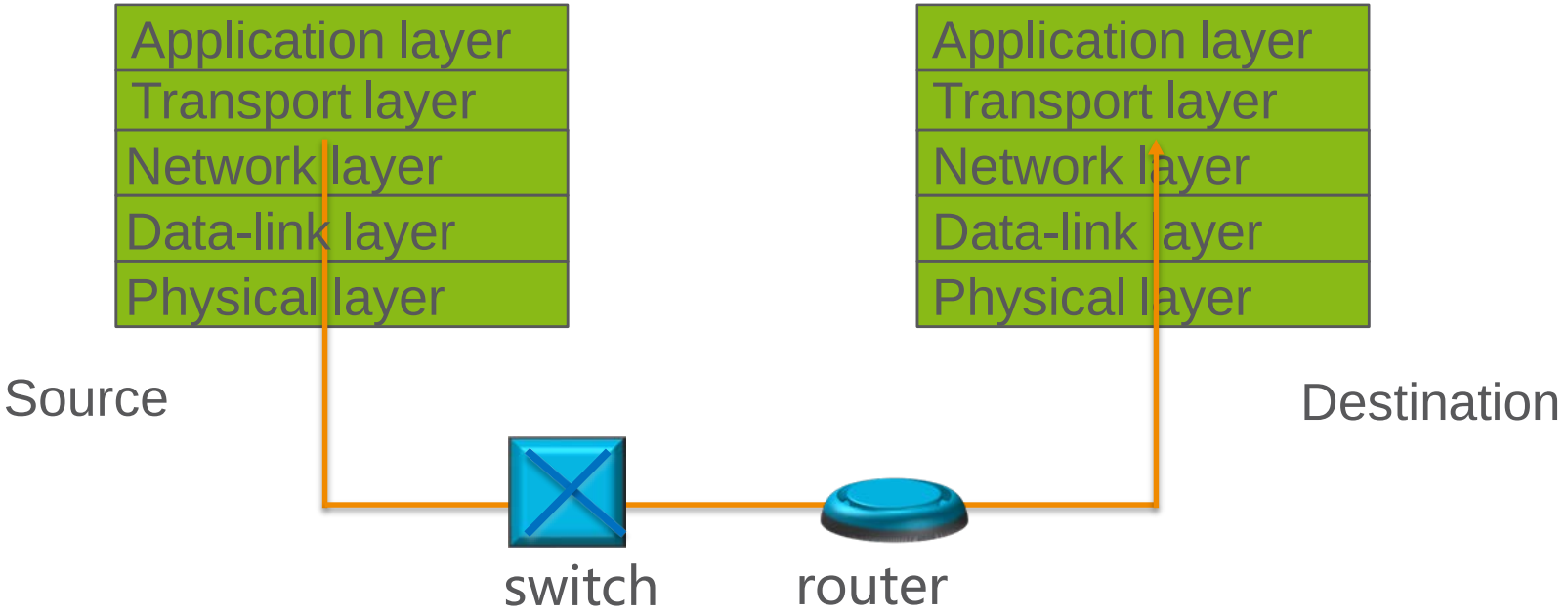
浙江大學
ZHEJIANG UNIVERSITY

An Introduction to Software-Defined Networking(SDN)

蒋骁翀

Reviewing traditional networking



Why layers? Good abstraction, transparency…

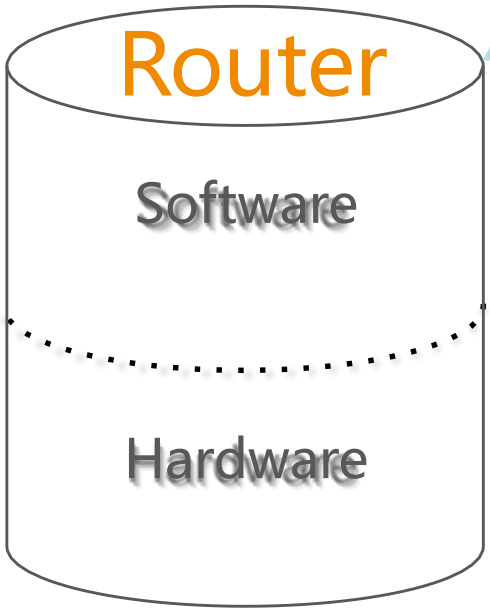
Pros & Cons

- Design principles of traditional Internet
  - Simple
  - Intelligent end-points
  - Distributed control*
- Resulting in huge complex network and hard to manage
  - Billions of computers
  - Tens of thousands of Ases
  - Great business for selling routers

Once Upon a Time…

- *AT&T Eyes Flexibility, Cost Savings With New Network Design*, Wall Street journal, 2014.
  - Upgrade their internal network infrastructure (routers and switches) every 18 months to keep up with the current demands for network.
  - Cost Billions USD to upgrade.
    - Cisco top of the line switch cost $27K USD
  - Other high cost: Involved many men power to upgrade the network.
  - *In Summary*: AT&T was eyeing for SDN capable switches (only **$11K USD** each).

Complex routers



Routing, management, mobility management, access control, VPNs, …

Million of lines of source code

Hard to extend

Tens of billions of transistors

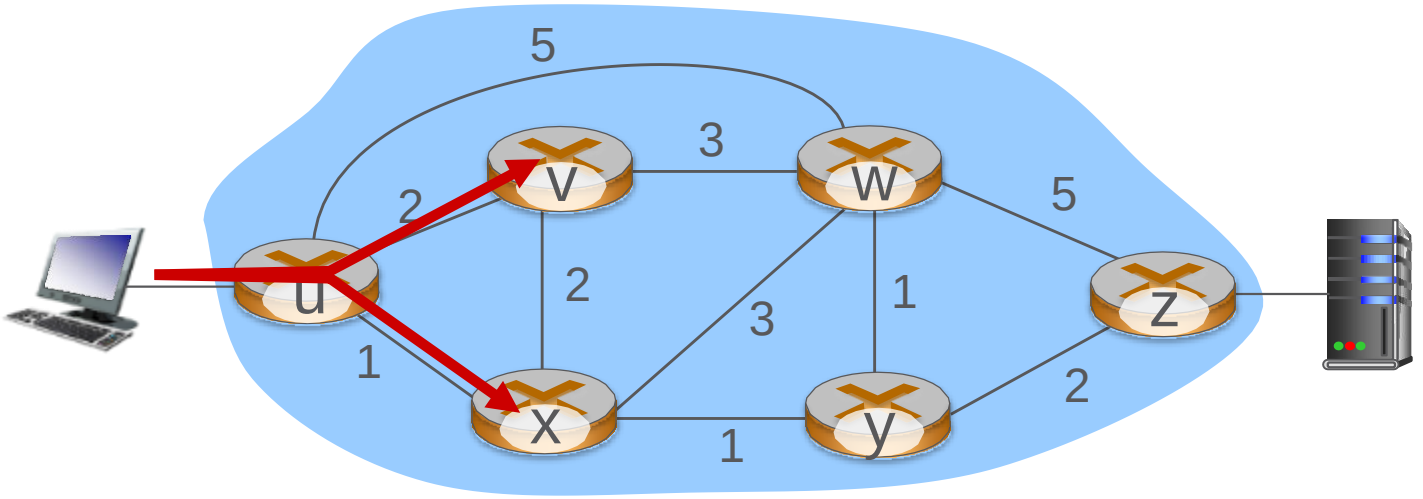
Power Hungry

10Gbytes RAM

- Many complex functions baked into the infrastructure

  OSPF, BGP, multicast, differentiated services, Traffic Engineering, NAT, firewalls, MPLS, redundant layers, …

Traffic engineering: difficult

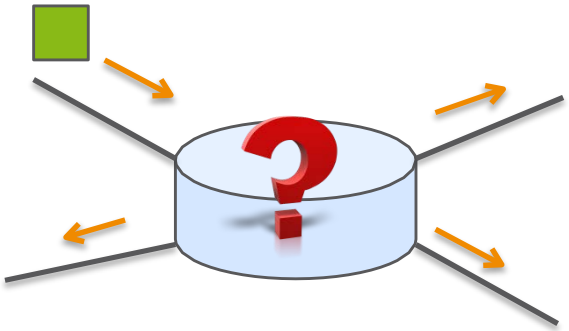


Q: What if network operator wants to split u-to-z traffic along uvwz *and* uxyz (load balancing)?

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

Relieve router’ s stress!

Router Example

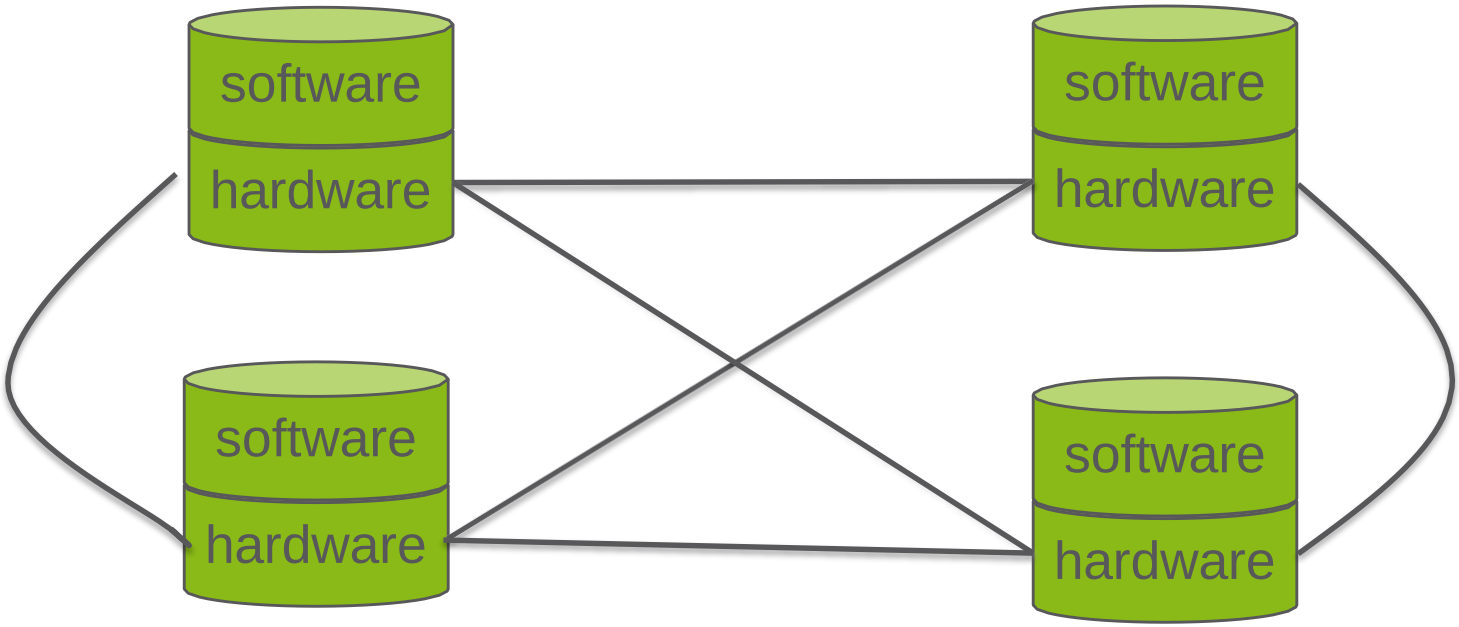


Basic job of the router:

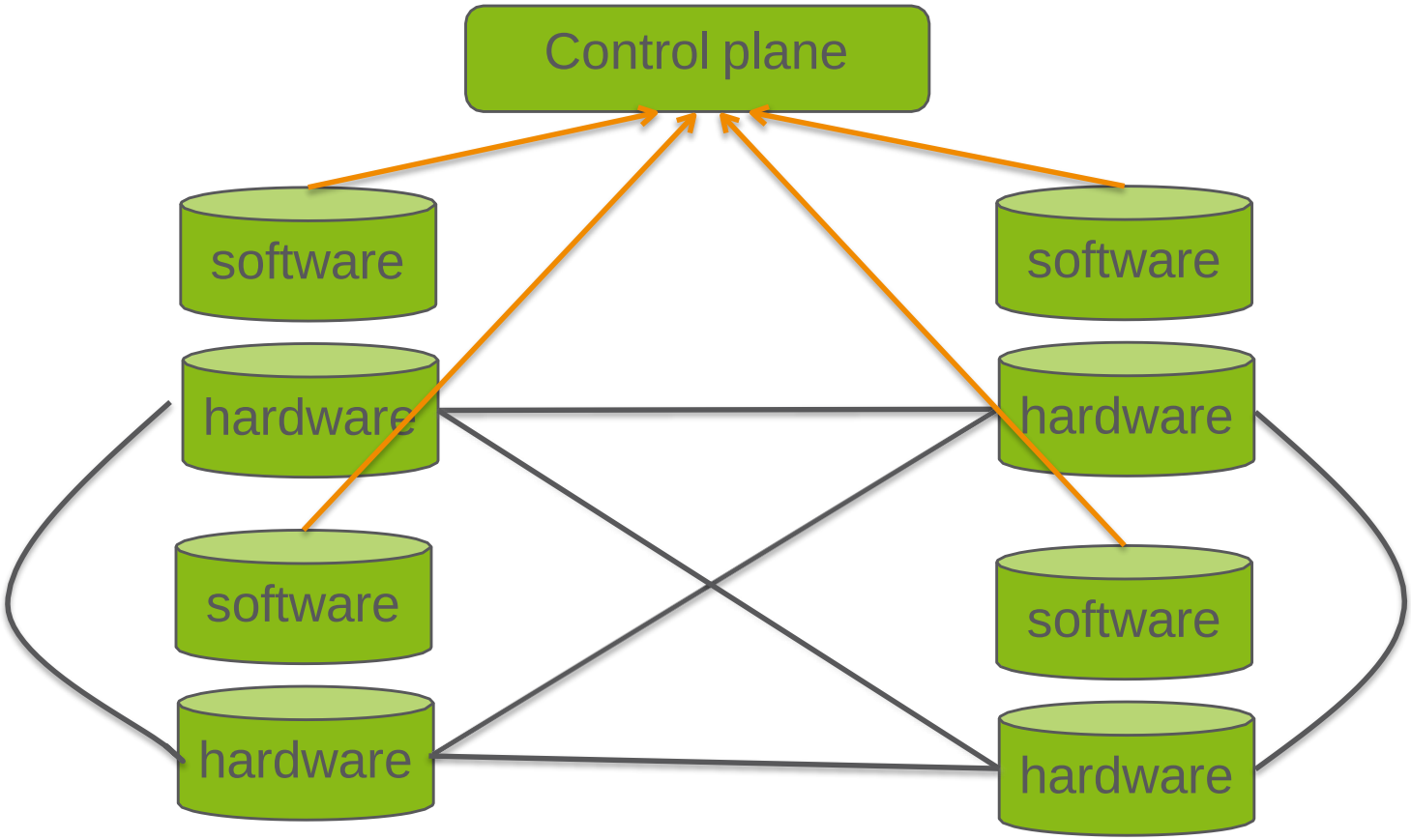
- *receiving packets, checking the routing table, forwarding the packets out*

- In order to build the routing table, the router has to understand BGP, OSPF, RIP, etc.
- What about getting the routing table from somewhere else!?

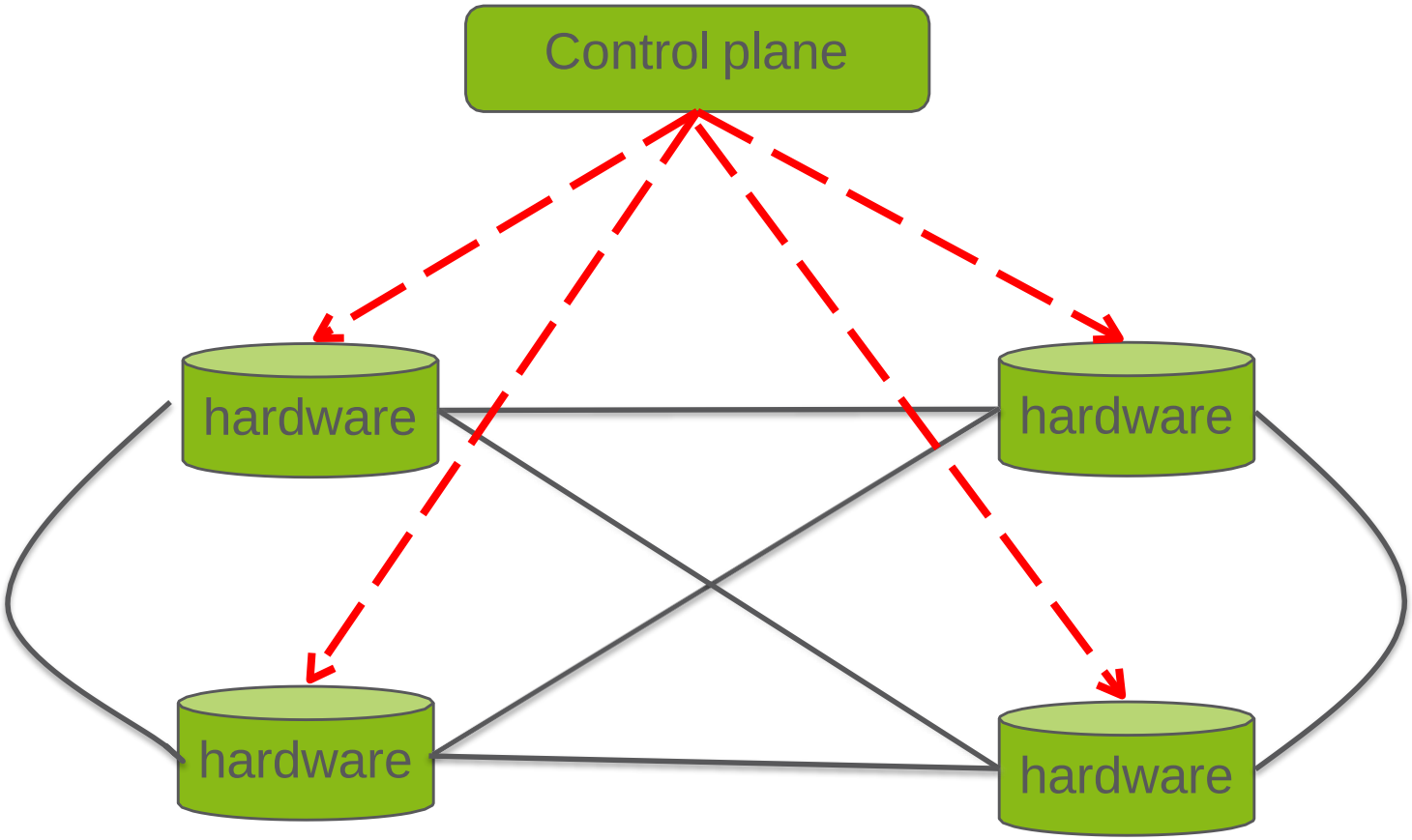
Traditional Network Topology



Relieve router’s stress!



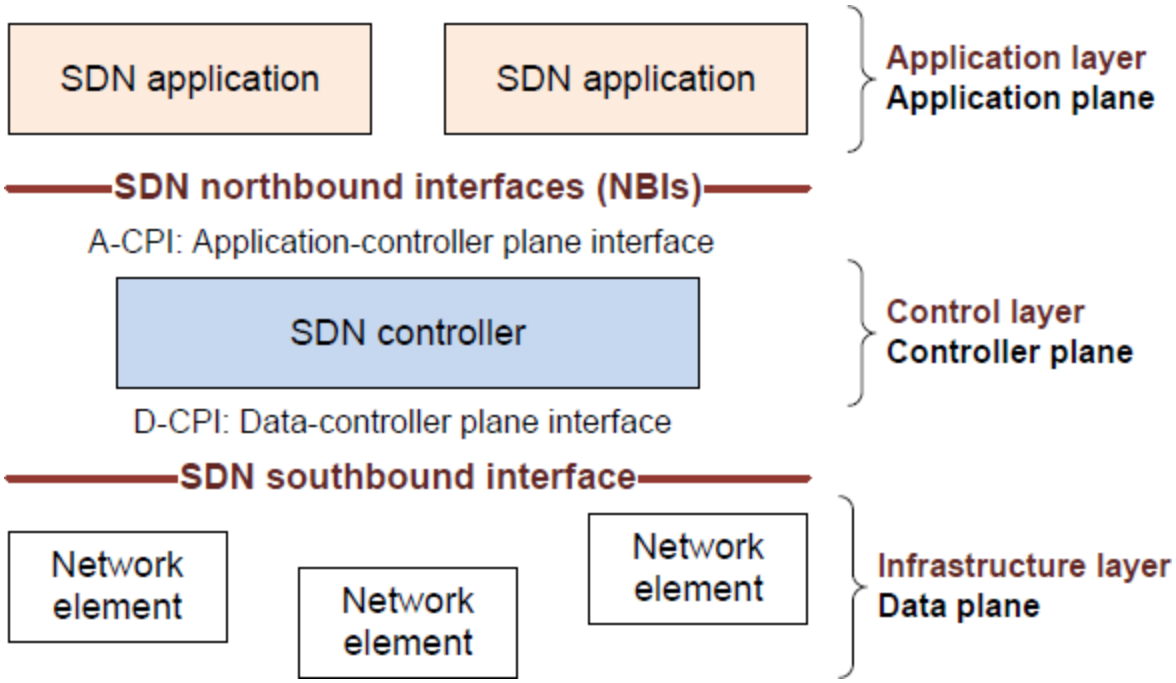
Relieve router’s stress!



SDN

- A network in which the control plane is physically separate from the data plane.
- A single control plane controls several forwarding devices.

SDN Architecture



SDN History

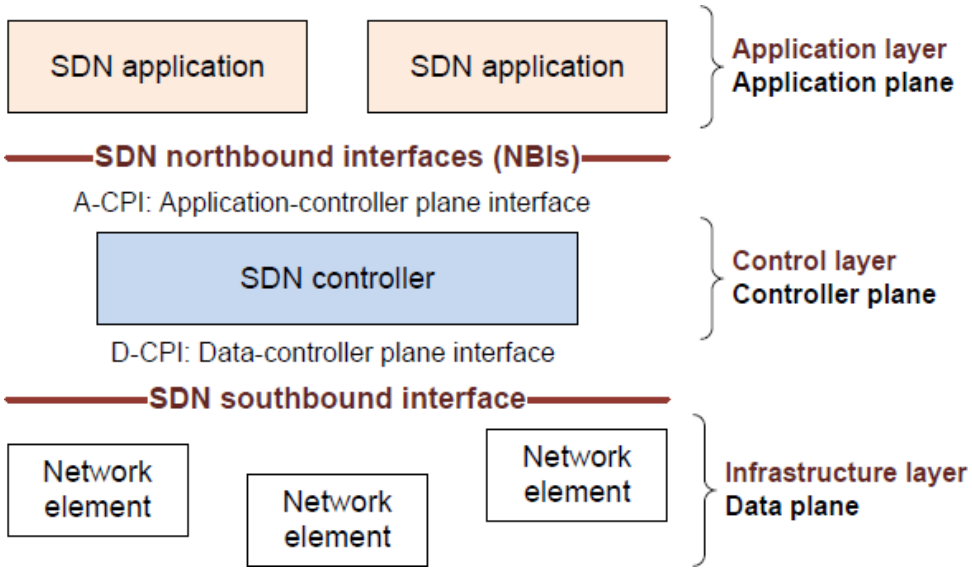
- 2006: **Martin Casado**, a PhD student at Stanford and team propose a clean-slate security architecture (SANE) which defines a centralized control of security (in stead of at the edge as normally done). Ethane generalizes it to all access policies.
- The idea of *Software Defined Network* is originated from OpenFlow project (*ACM SIGCOMM 2008*).
- 2009: Stanford publishes OpenFlow V1.0.0 specs.
- June 2009: Martin Casado co-founds Nicira.
- March 2011: Open Networking Foundation is formed.
- Oct 2011: First Open Networking Summit. Many Industries (Juniper, Cisco announced to incorporate.
- July 2012: VMware buys Nicira for **$1.26B**.
- Lesson Learned: Imagination is the key to *unlock* the power of possibilities.

SDN History

- The industries were **skeptical** whether SDN was possible.
- Google had big problems:
  - **High financial cost** managing their datacenters: Hardware and software upgrade, over provisioning (fault tolerant), manage large backup traffic, time to manage individual switch, and a lot of men power to manage the infrastructure.
  - **Delay** caused by rebuilding connections after link failure.
    - Slow to rebuild the routing tables after link failure.
    - Difficult to predict what the new network may perform.
- Google went a head and implemented SDN.
  - Built their hardware and wrote their own software for their internal datacenters.
  - Surprised the industries when Google announced SDN was possible in production.
- How did they do it?
  - ✓ Read *B4: Experience with a Globally-Deployed Software Defined WAN*, ACM SIGCOMM 2013.

Data-plane

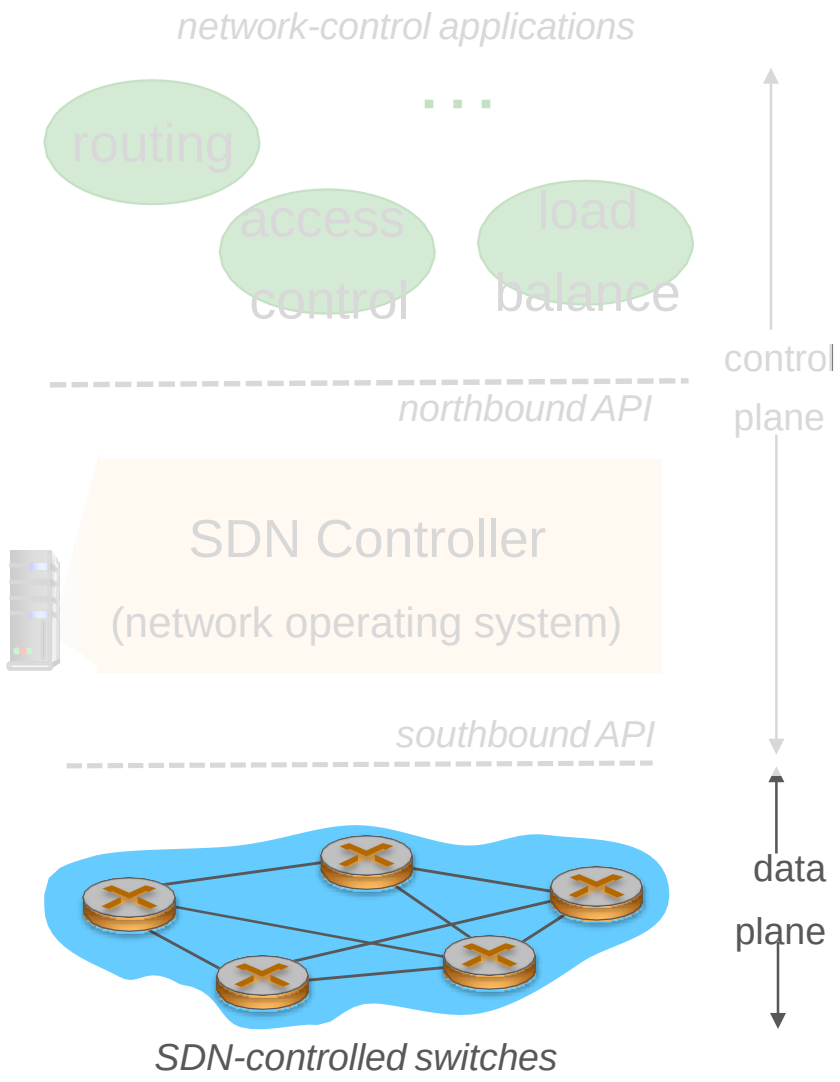
- Data sources and sinks
- Traffic forwarding/processing engine
  - May have the ability to handle some types of protocol, e.g. ARP, LLDP.
- Provide interfaces communicating to the control plane
  - Programmatic control of all functions offered by the network element
  - Capability advertisement
  - Event notification



Data plane switches

Data plane switches

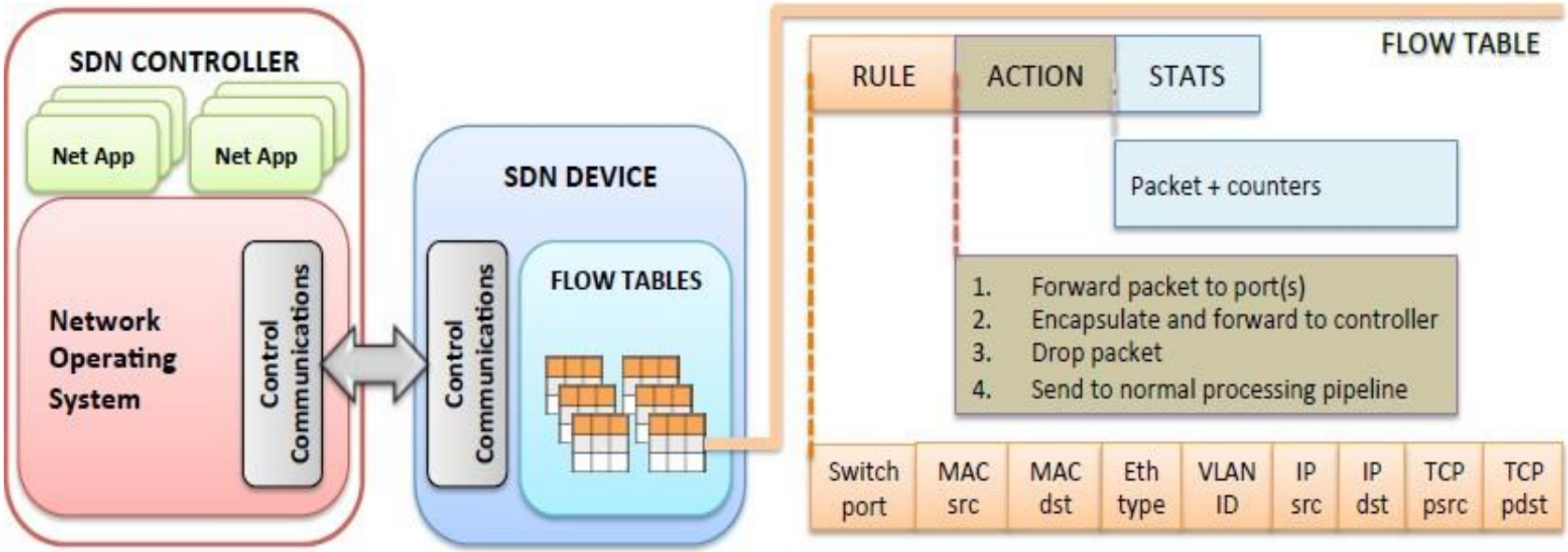
- Fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
- Switch flow table computed, installed by controller
- API for table-based switch control (e.g., OpenFlow)
  - Defines what is controllable and what is not
- Protocol for communicating with controller (e.g., OpenFlow)



What is OpenFlow?

An southbound interface standard!

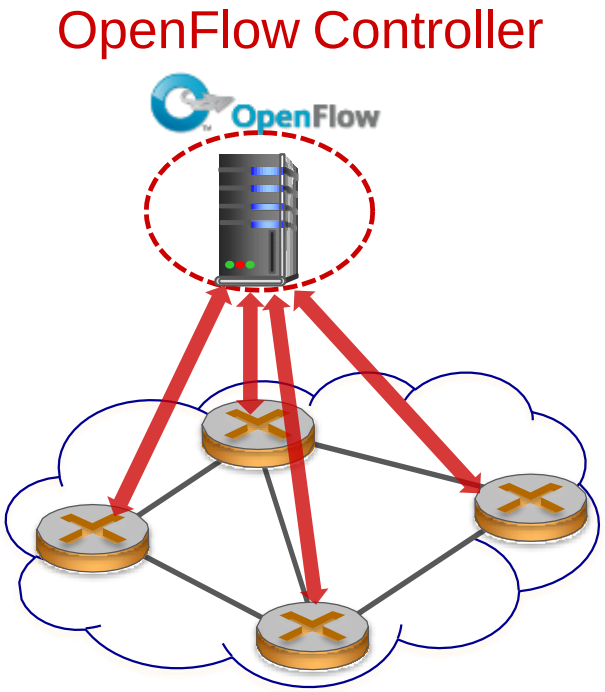
- Provide specification to implement OpenFlow-enabled forwarding devices
- Communication channel between data and control plane
  - TCP used to exchange messages



Controller-to-switch Messages

Key controller-to-switch messages

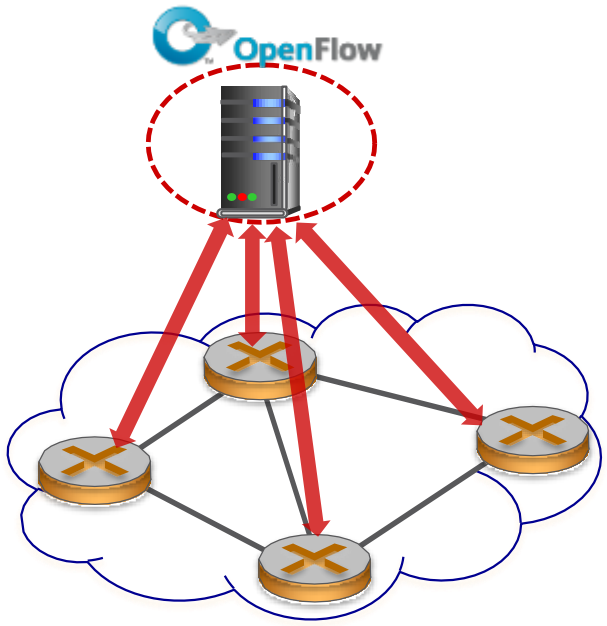
- *features:* controller queries switch features, switch replies
- *configure:* controller queries/sets switch configuration parameters
- *modify-state:* add, delete, modify flow entries in the OpenFlow tables
- *packet-out:* controller can send this packet out of specific switch port



Switch-to-controller Messages

Key switch-to-controller messages

- *packet-in:* Transfer packet (and its control) to controller. See packet-out message from controller
- *flow-removed:* Flow table entry deleted at switch
- *port status:* Inform controller of a change on a port.



The Actual Flow Table Looks Like…

Port	Src MAC	Dst MAC	VLAN ID	Priority	EtherType	Src IP	Dst IP	IP Proto	IP ToS	Src L4 Port ICMP Type	Dst L4 Port ICMP Code	Action	Counter
*	*	0A:C8:*	*	*	*	*	*	*	*	*	*	Port 1	102
*	*	*	*	*	*	*	192.168.*.*	*	*	*	*	Port 2	202
*	*	*	*	*	*	*	*	*	*	21	21	Drop	420
*	*	*	*	*	*	*	*	0x806	*	*	*	Local	444
*	*	*	*	*	*	*	*	0x1*	*	*	*	Controller	1

OpenFlow Actions

- All: To all interfaces except incoming interface.
- Controller: Encapsulate and send to controller.
- Local: send to its local networking stack.
- Table: Perform actions in the next flow table (table chaining or multiple table instructions).
- In_port: Send back to input port.
- Normal: Forward using traditional Ethernet.
- Flood: Send along minimum spanning tree except the incoming interface.
- Drop: Discard packet.

Devices That Support OpenFlow

HP Procurve 5400

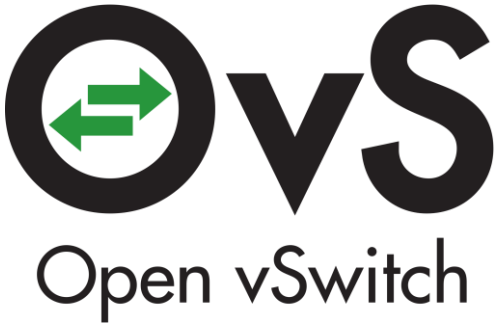
Netgear 7324

Pronto 3240/3290

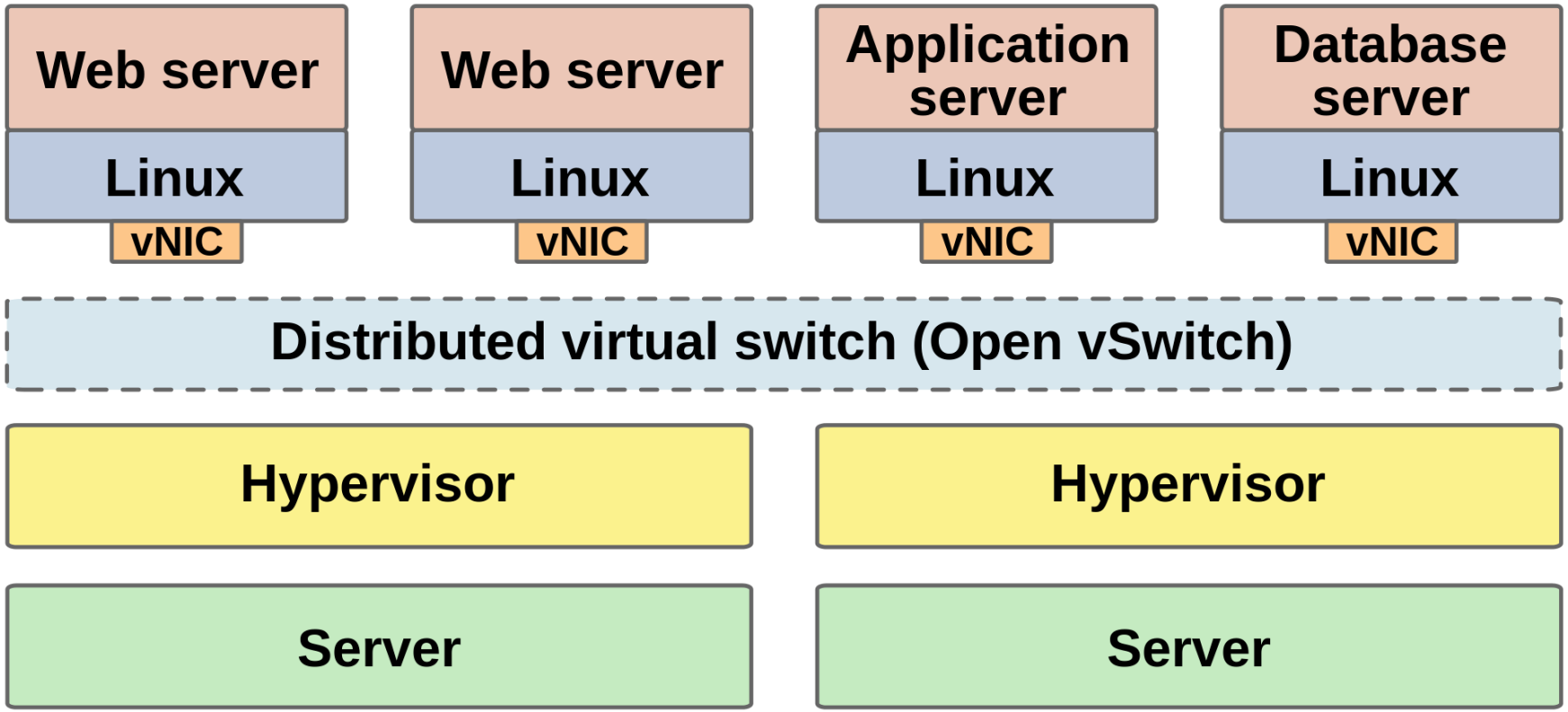
Ciena Coredirector

Commercial Products

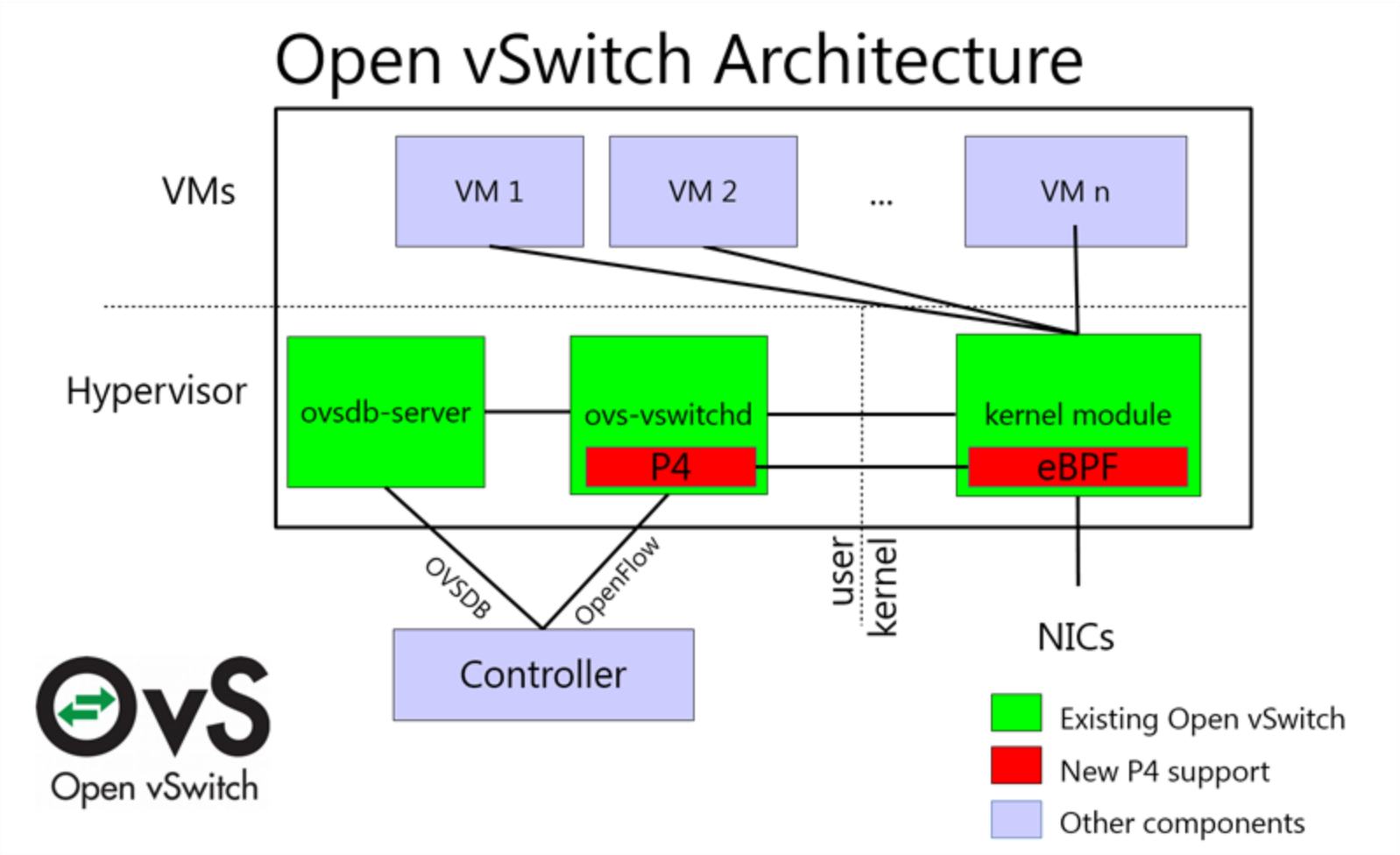
Devices That Support OpenFlow



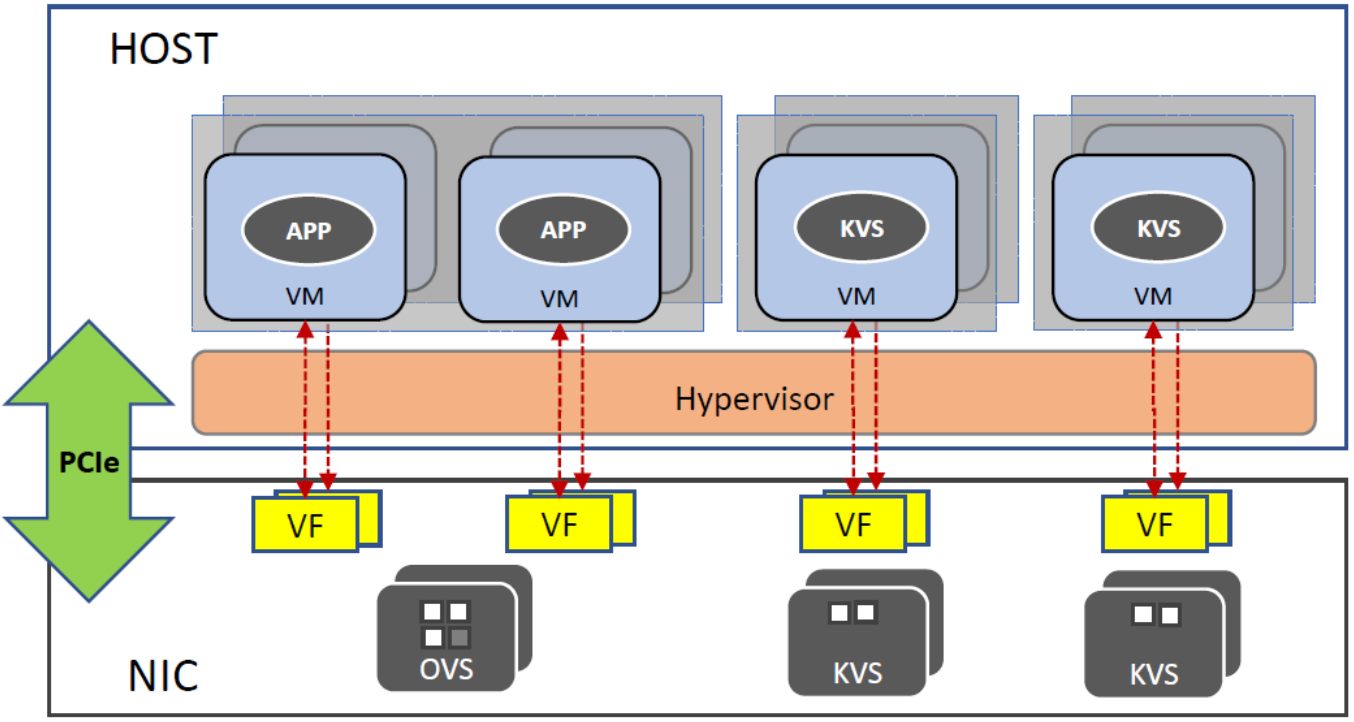
Use virtual switches on servers!



Open vSwitch



Smart NIC

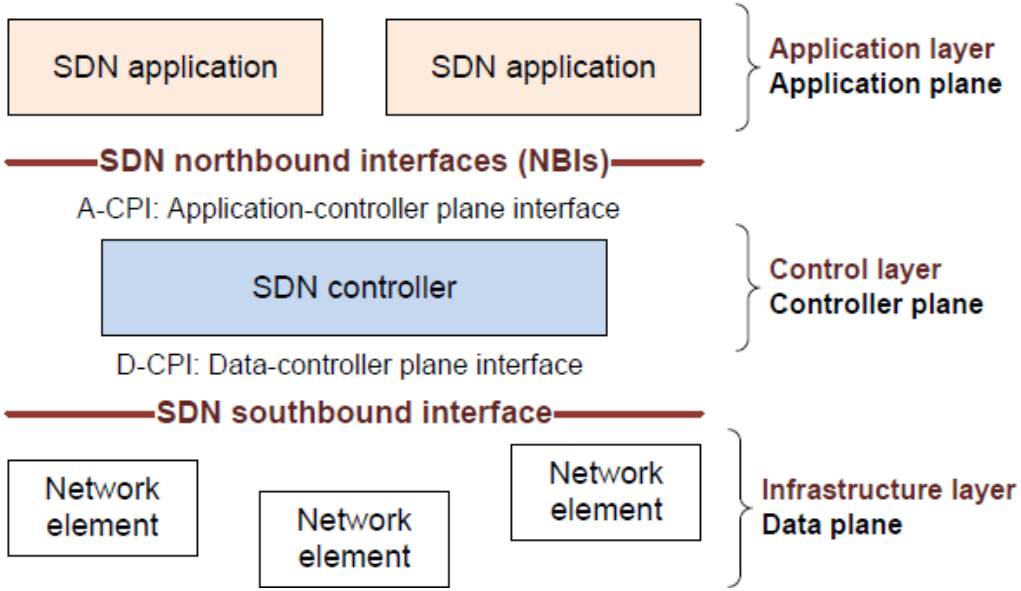


KVS VMs can communicate with SmartNIC applications through SR-IOV

- 4 × 100 Gigabit Ethernet
- PCIe Gen4 Interface
- Intel Stratix DX210 FPGA

Application-plane

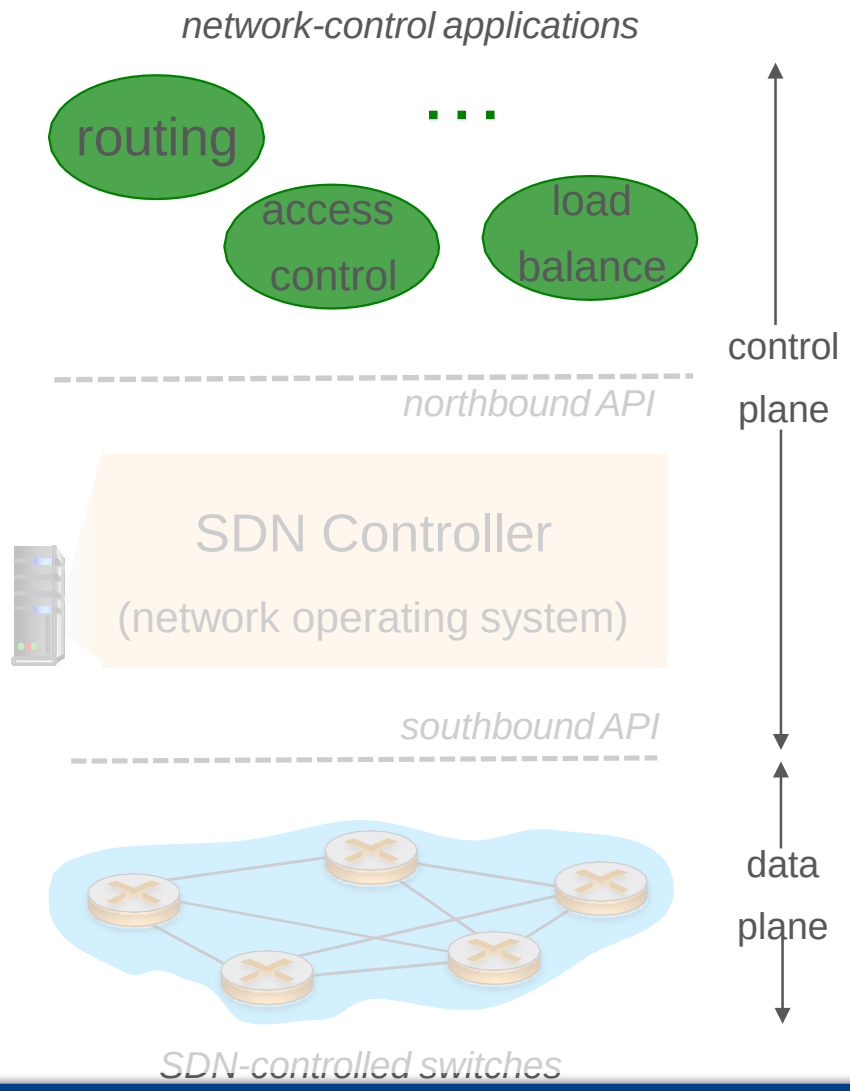
- Applications specify the resources and behaviors required from the network, with the context of business and policy agreement
- It may need to orchestrate multiple-controllers to achieve the objectives, (Cloudify, Unify)
- Programming languages help developing applications.



Control Applications

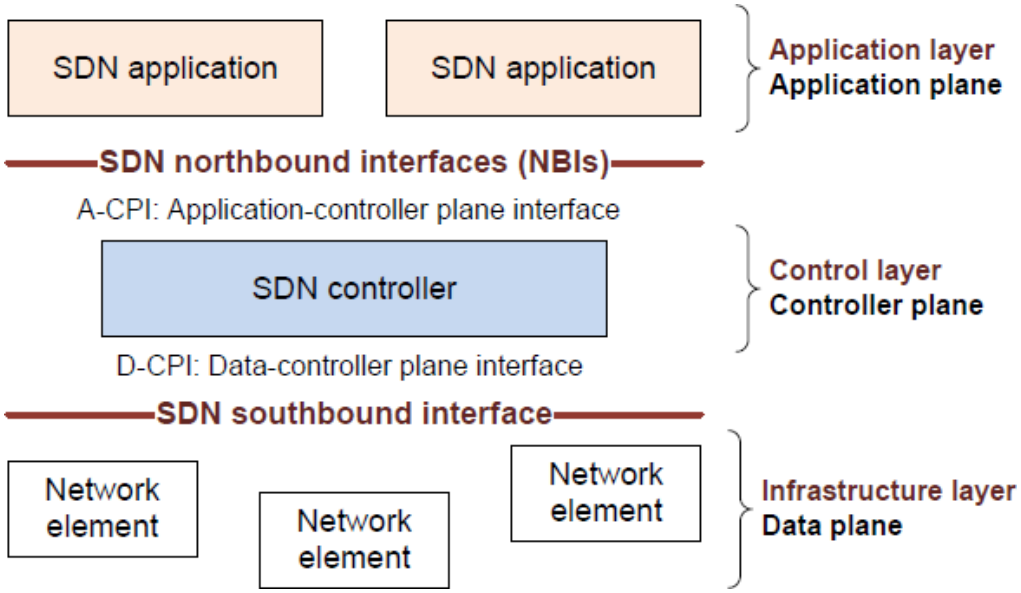
Network-control apps:

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



What about Control-plane?

- Logically centralized
- Core functionality
  - Topology and network state information
  - Device discovery
  - Path computation
  - Security mechanism
- Coordination among different controllers
- Interfaces to the application plane

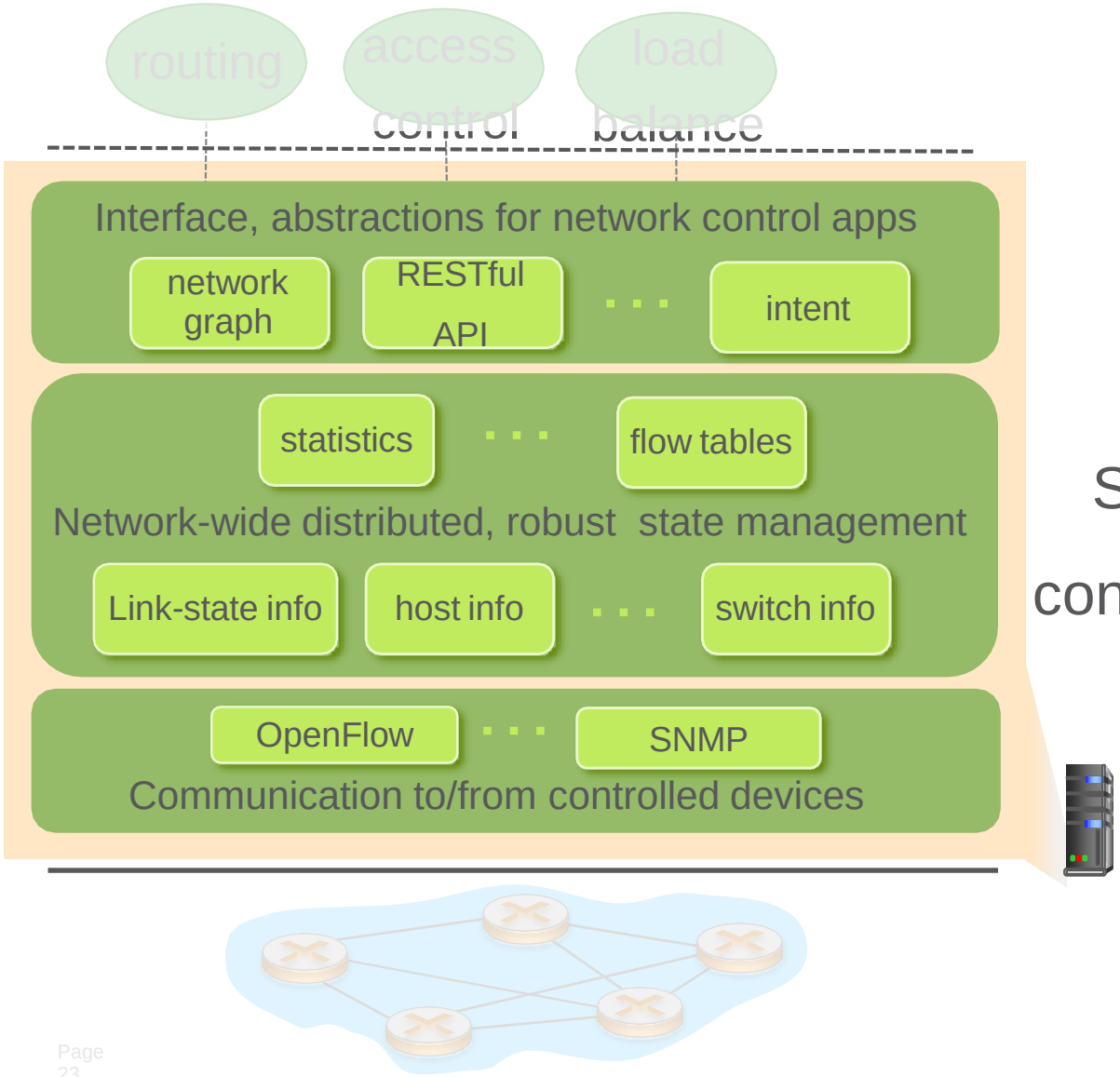


Components of SDN Controller

Interface layer to network control apps: abstractions API

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

Communication layer: communicate between SDN controller and controlled switches



Some SDN Controllers

- **POX**: (Python) Out of Date.
- **IRIS**: (Java) Scalability and High Availability
- **MUL**: (C) MūL, is an OpenFlow (SDN) controller. It has a C based multi-threaded infrastructure at its core.
- **NOX**: (C++/Python) NOX was the first OpenFlow controller.
- **Jaxon**: (Java) Jaxon is a NOX-dependent Java-based OpenFlow Controller.
- **Trema**: (C/Ruby) Trema is a full-stack framework for developing OpenFlow controllers in Ruby and C.
- **Beacon**: (Java) Beacon supports both event-based and threaded operation.
- **Floodlight**: (Java) It was forked from the Beacon controller, originally developed by David Erickson at Stanford.
- And many more.

浙江大學
ZHEJIANG UNIVERSITY

Orion: Google’s Software-Defined Networking Control Plane

Google, NSDI’2021

What is Orion?

- Orion is…
  - Google’s 2nd generation SDN control plane
  - Responsible for configuration, management, real-time network control
  - In all Google’s data center (Jupiter), campus, and private Wide Area (B4) networks
  - In production for over 4 years
  - An evolving platform
    - Biweekly releases, over 30 new significant capabilities, improved scale by 16x, improved availability by 50x (Jupiter) / 100x (B4), and improved convergence time by a factor of 40x

Challenges

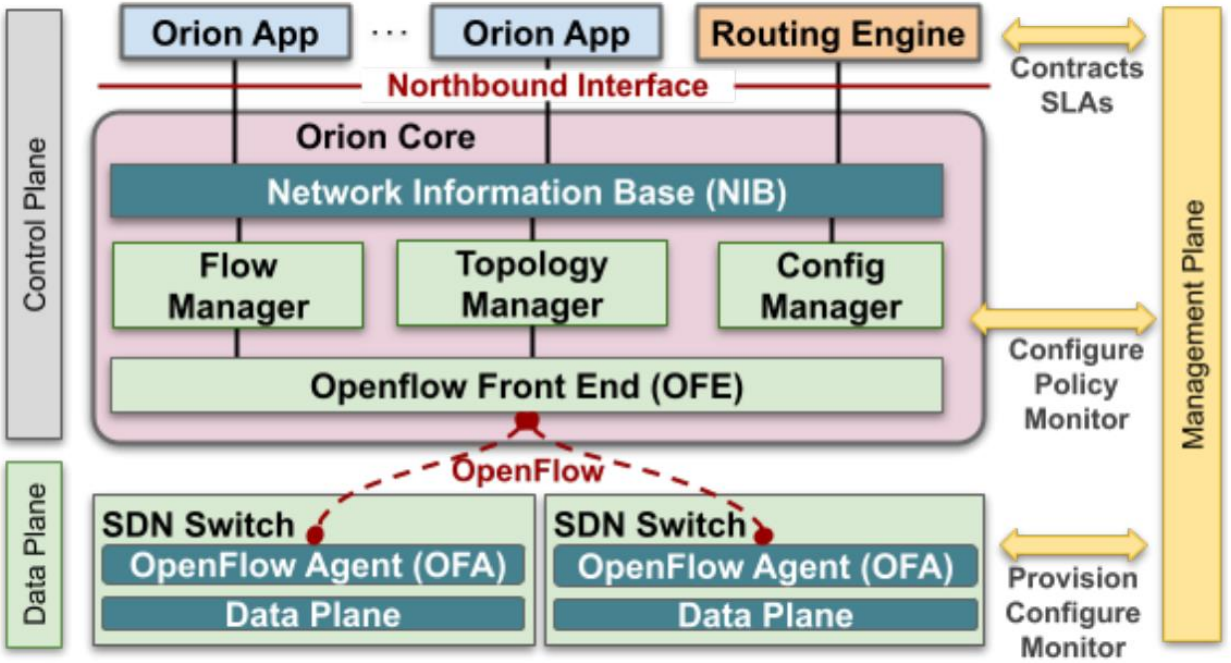
#1: Logically centralized control require fundamentally high performance for updates, in-memory representation of state, and appropriate consistency levels among loosely-coordinating micro-service SDN applications.

#2: The decoupling of control from hardware elements breaks fate sharing in ways that make corner-case failure handling more complex.

#3: Managing the tension between centralization and fault isolation must be balanced carefully.

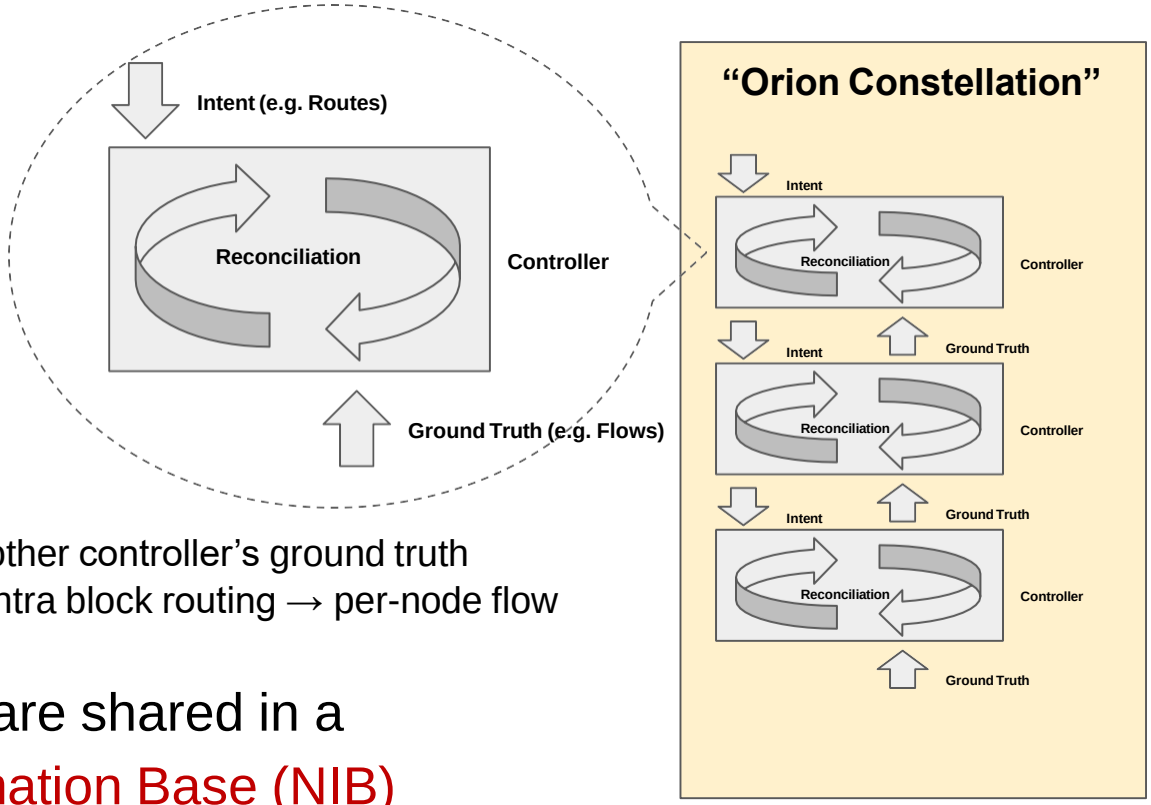
#4: In a global network setting, we must integrate existing routing protocols, primarily BGP, into Orion to allow inter-operation with non-SDN peer networks

Orion’ s Architecture



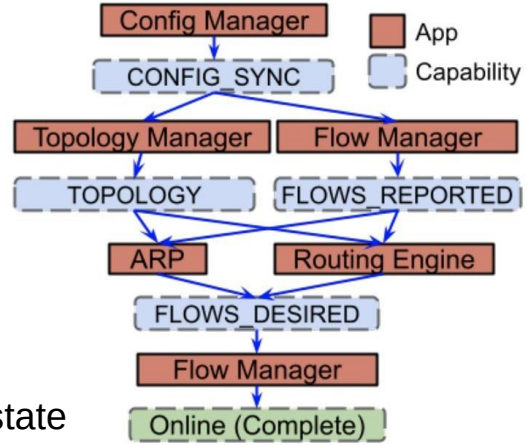
Intent-Based System

- Intent flows down
- Ground truth flows up
- Controller algorithm
  - Continually reconcile intent with ground truth
  - In the least disruptive way possible
- Controllers layer
  - One controller's intent is another controller's ground truth
    - Inter block routing → intra block routing → per-node flow programming
- Intent and ground truth are shared in a pub-sub Network Information Base (NIB)



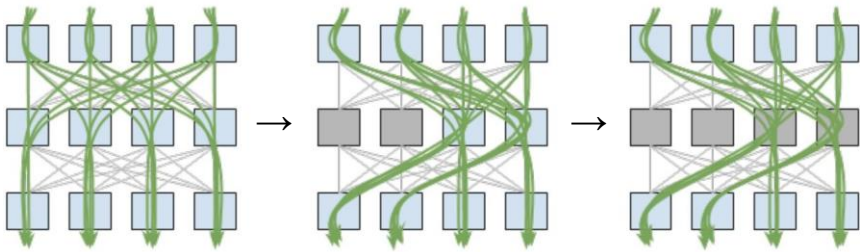
Orion: Continuous Reconciliation

- Ordinary reconciliation is continuous
  - E.g Network failures trigger rerouting
- Typical controller failure: Shared data preserved
  - Rebuild internal controller state as needed and continue
- Special case: Shared data lost
  - Capability Reconciliation: orderly reconciliation of lost data
  - Controllers:
    - Block on abstract capabilities reflecting required input state
    - Explicitly mark provided capabilities ready
  - Avoids need for strong durability of shared data
- Simplify: map all errors to a kind of controller failure
- For all reconciliations, convergence speed matters
  - Implies a loss of control or a period of lost traffic



Handling Controller Disconnects

- Controllers manage switches over an asynchronous network
- Controller observed disconnect ≠ dataplane failure
- Experience: *Controller connection failures* are more common than dataplane network failures.
- Strategy:
  - Aggressively route around small uncorrelated failures (fail-closed)
  - Preserve current state in larger or correlated failures (fail-static)

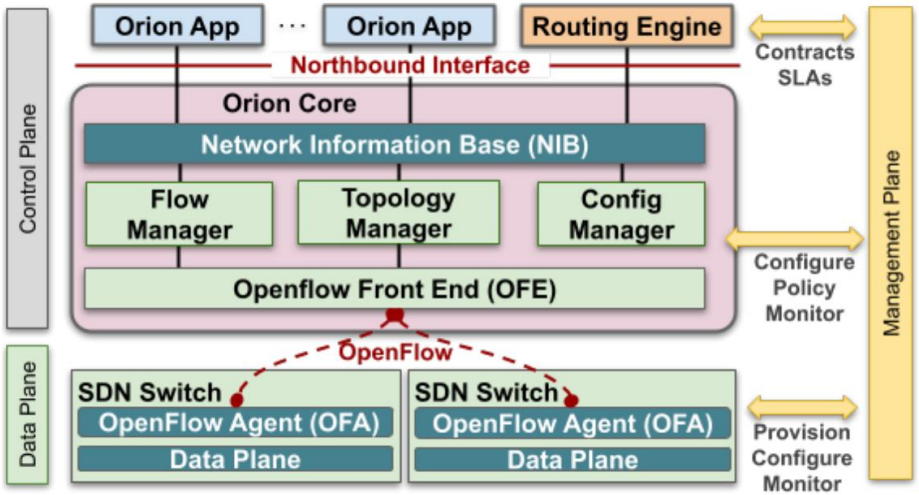


(no-failure) (fail-closed) (fail-static)

Network Information Base (NIB)

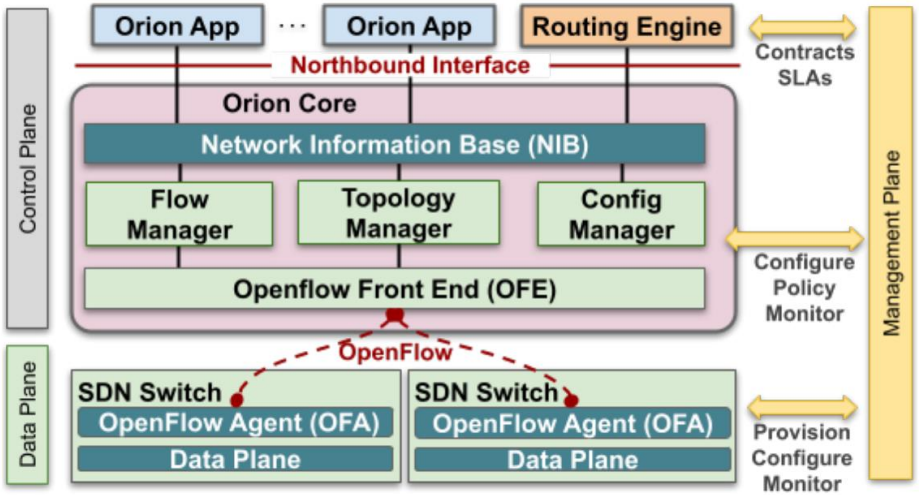
The NIB is the intent store for all Orion applications. It is implemented as a centralized, in-memory datastore with replicas that reconstruct the state from ground-truth on failure. The NIB is coupled with a publish-subscribe mechanism to share state among Orion applications.

- *Configured network topology*
- *Network run-time state*
- *Orion App Configuration*



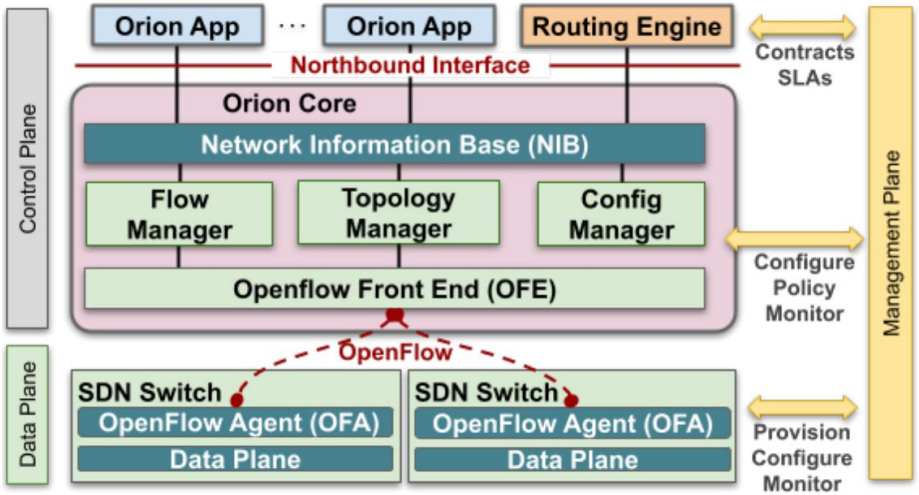
Config Manager

The **Config Manager** provides an external management API to configure all components in an Orion domain. The domain configuration is the set of app configurations running in that domain. For uniformity and ease of sharing, an app config consists of one or more NIB tables.



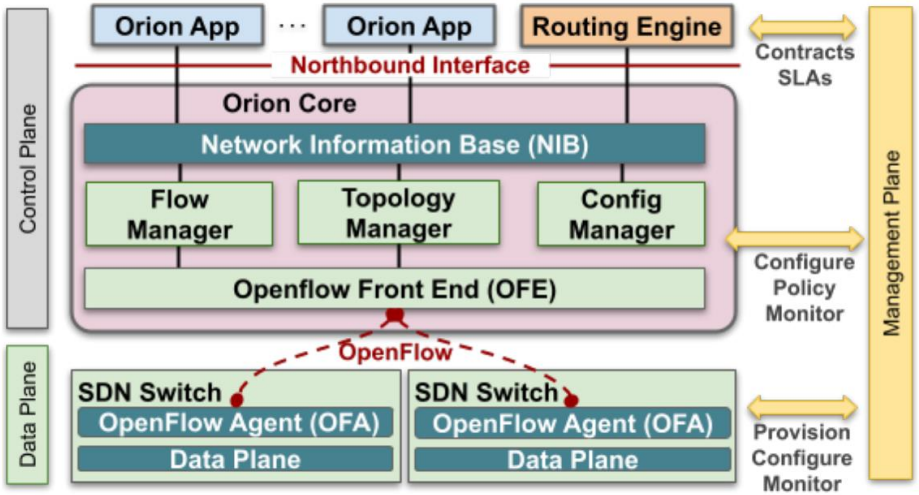
Topology Manager

The **Topology Manager** sets and reports the runtime state of network dataplane topology (node, port, link, interface, etc.). It learns the intended topology from its config in the NIB. By subscribing to events from the switches, it writes the current topology to tables in the NIB. The Topology Manager also periodically queries port statistics from the switches.



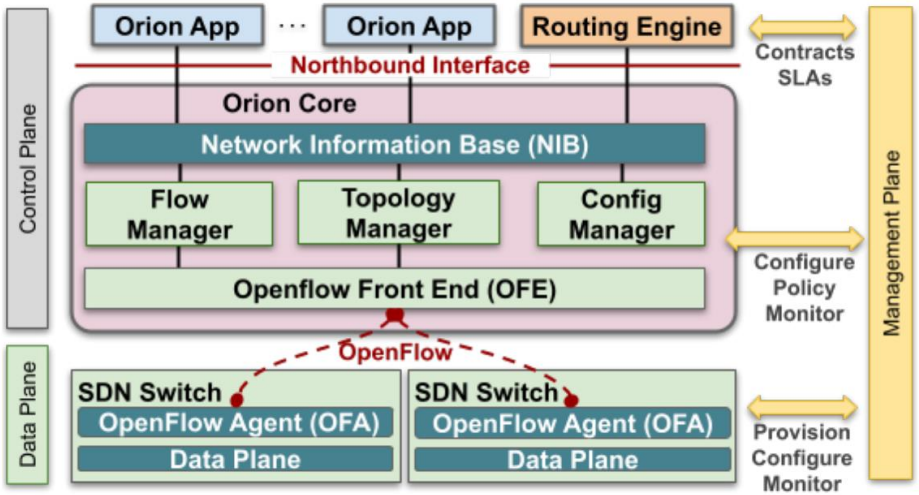
Flow Manager

The **Flow Manager** performs flow state reconciliation, ensuring forwarding state in switches matches intended state computed by Orion apps and reflected in the NIB. Reconciliation occurs when intent changes or every 30 seconds by comparing switch state. The latter primarily provides Orion with switch statistics and corrects out-of-sync state in the rare case that reconciliation on intent change failed.



The OFE (Openflow FrontEnd)

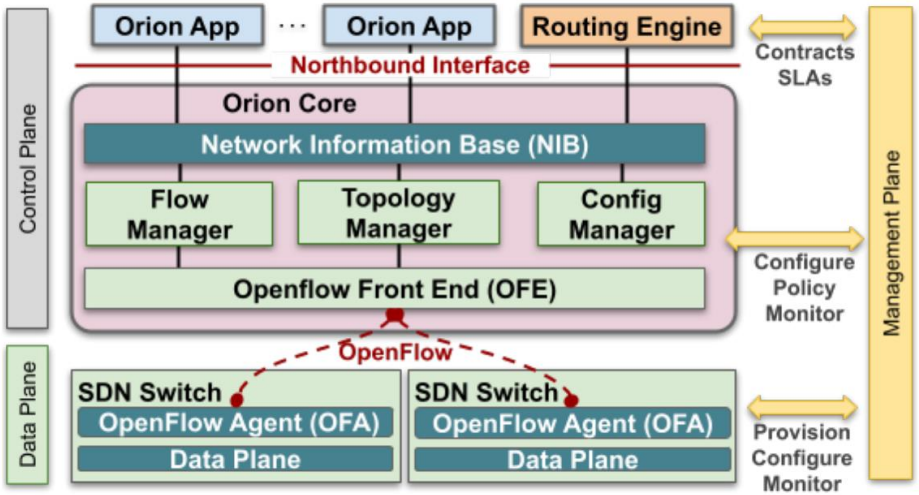
The OFE (Openflow FrontEnd) multiplexes connections to each switch in an Orion domain. The OpenFlow protocol provides programmatic APIs for (i) capabilities advertise-ment, (ii) forwarding operations, (iii) packet IO, (iv) teleme-try/statistics, and (v) dataplane event notifications (e.g. link down). These are exposed to the Topology and Flow Manager components via OFE's northbound RPC interface.



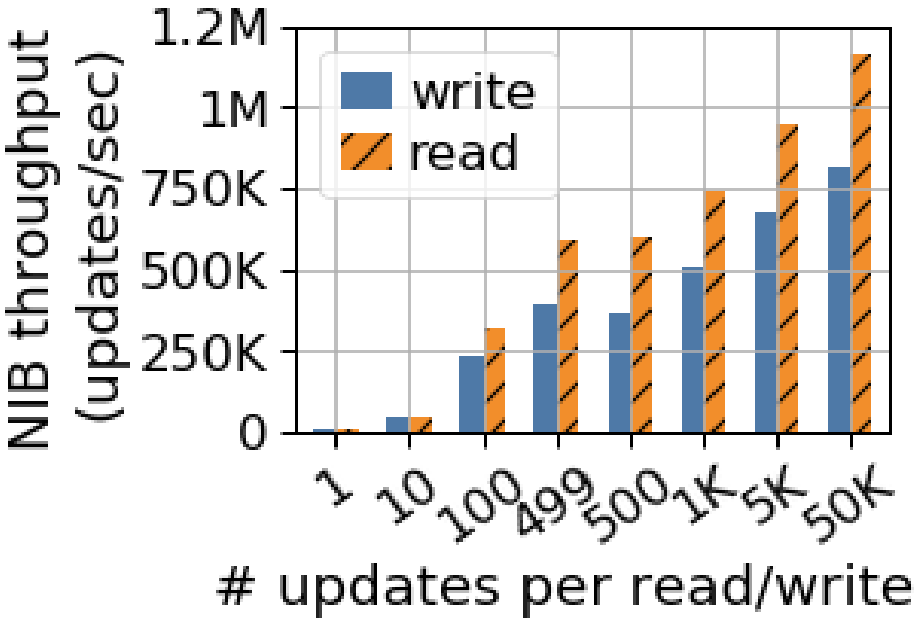
Routing Engine

Routing Engine (RE) is Orion's intra-domain routing controller app, providing common routing mechanisms, such as L3 multi-path forwarding, load balancing, encapsulation, etc.

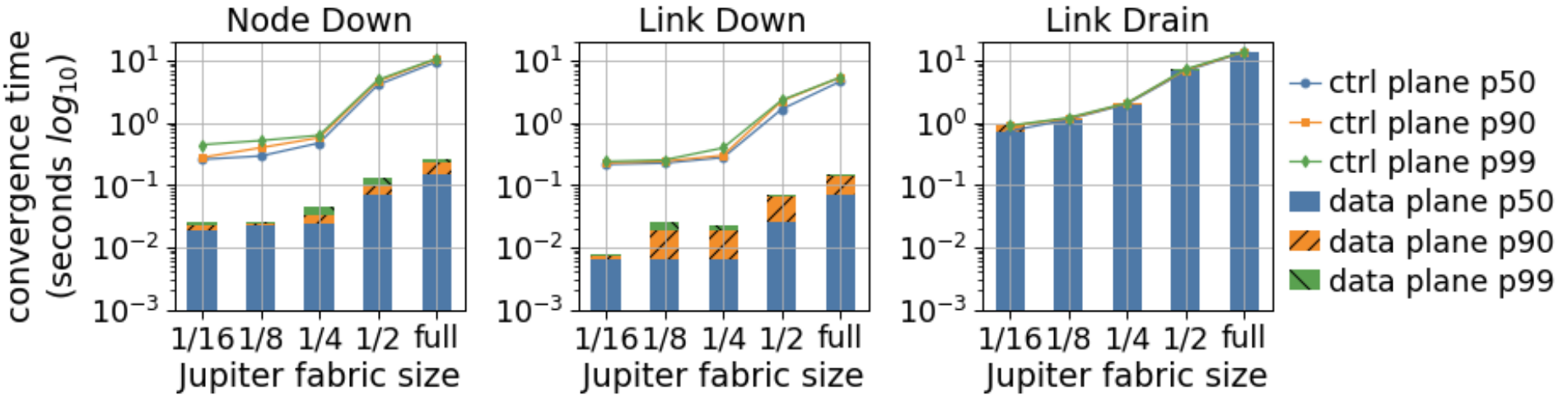
RE provides abstracted topology and reachability information to client routing applications (e.g. an inter-domain routing app or a BGP speaker app).

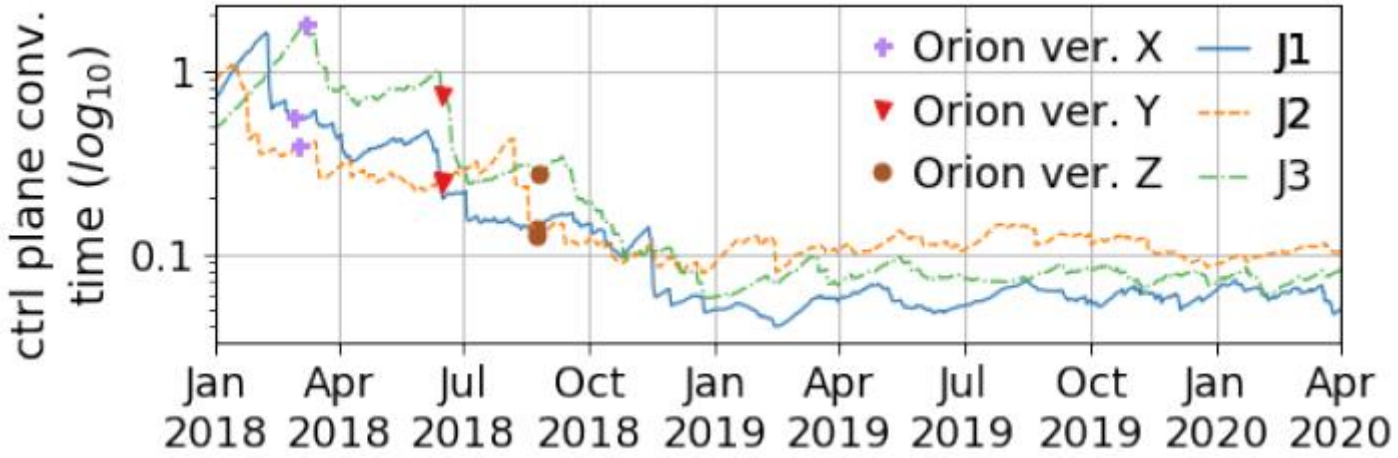


Experiments: NIB performance

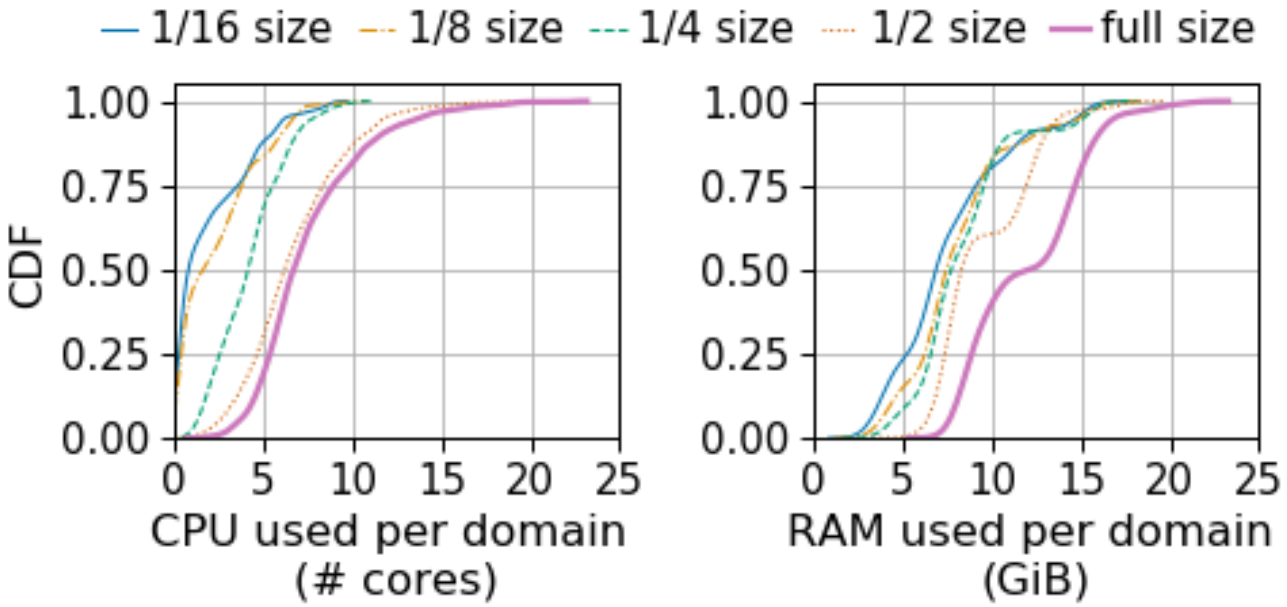


Data and control plane convergence





Controller footprint: CPU and memory



Conclusion

Orion decouples control from individual hardware elements, enabling a transition from pair-wise coordination through slowly-evolving protocols to a high-performance distributed system with a logically centralized view of global fabric state.

We highlight Orion's benefits in availability, feature velocity, and scale while addressing challenges with SDN including aligning failure domains, inter-operating with existing networks, and decoupled failures in the control versus data planes.

Thanks