# Control-Data Plane Separation
## Part II

Lecture 4, Computer Networks (198:552)

Fall 2019

RUTGERS
UNIVERSITY | NEW BRUNSWICK

# Quick review of last lecture

- What a protocol is (rules over messages and actions)
- Edge and Core
- How the Internet core is organized (Ases, intra-AS concerns, inter-AS concerns)
- Separation of decision making (control plane) and decision execution (data plane) for moving packets through the network
- The router data plane:
  - Forwarding table, switching fabric
  - Buffering, Scheduling
- Control plane protocols: OSPF for intra-domain routing

# What differentiates the routing protocols

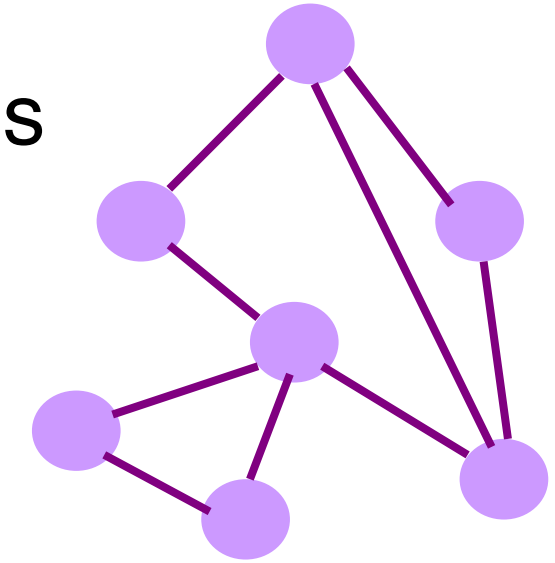
- What outcome it computes: spanning tree, shortest path, local policy, arbitrary end-to-end paths

- What algorithm it runs: Spanning-tree construction, distance vector, link-state routing, path-vector routing, source routing, end-to-end signaling

- How the protocol learns the location of endpoints: flooding, neighbor message exchange, injecting addresses into the protocol, dissemination using a different protocol, directory server

# Different tradeoffs in routing protocols

- State required to represent the paths
- Ability to use all links
- Ability to support multiple paths
- Quality of computed paths
- Complexity of computing the paths
- Costs of propagating endpoint reachability information

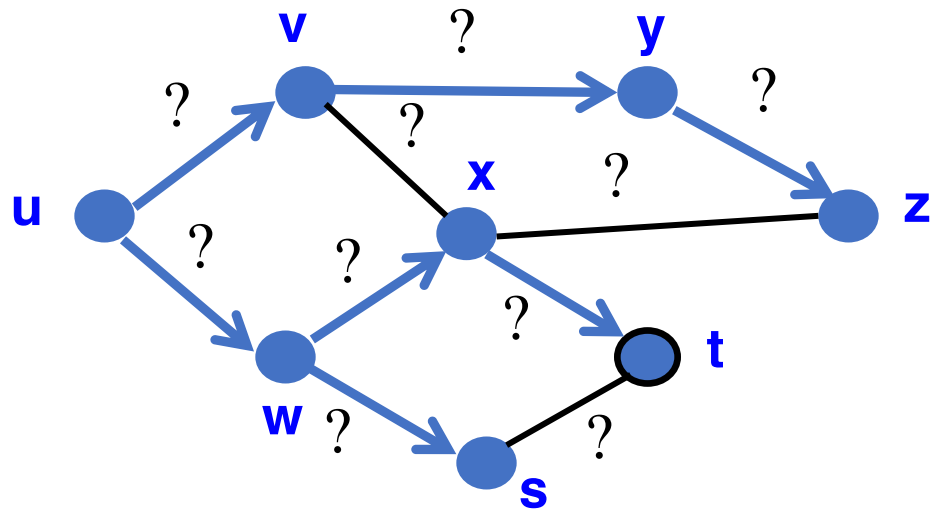


# Implications of traditional routing design

- Vendor dependence
  - Dependent on router software and vendors to get the right hardware behavior
- Opaque network operations
  - Unnecessarily hard for network admins to enforce their intent
  - Ex: compute weights to imply a certain set of “shortest paths”
- Unnecessary protocol complexity
  - a large fraction of OSPF specification is about maintaining consistent knowledge of the network across routers

# OSPF: Shortest-path tree

- Shortest-path tree from u
- Forwarding table at u




| | link |
|---|---|
| v | (u,v) |
| w | (u,w) |
| x | (u,w) |
| y | (u,v) |
| z | (u,v) |
| s | (u,w) |
| t | (u,w) |

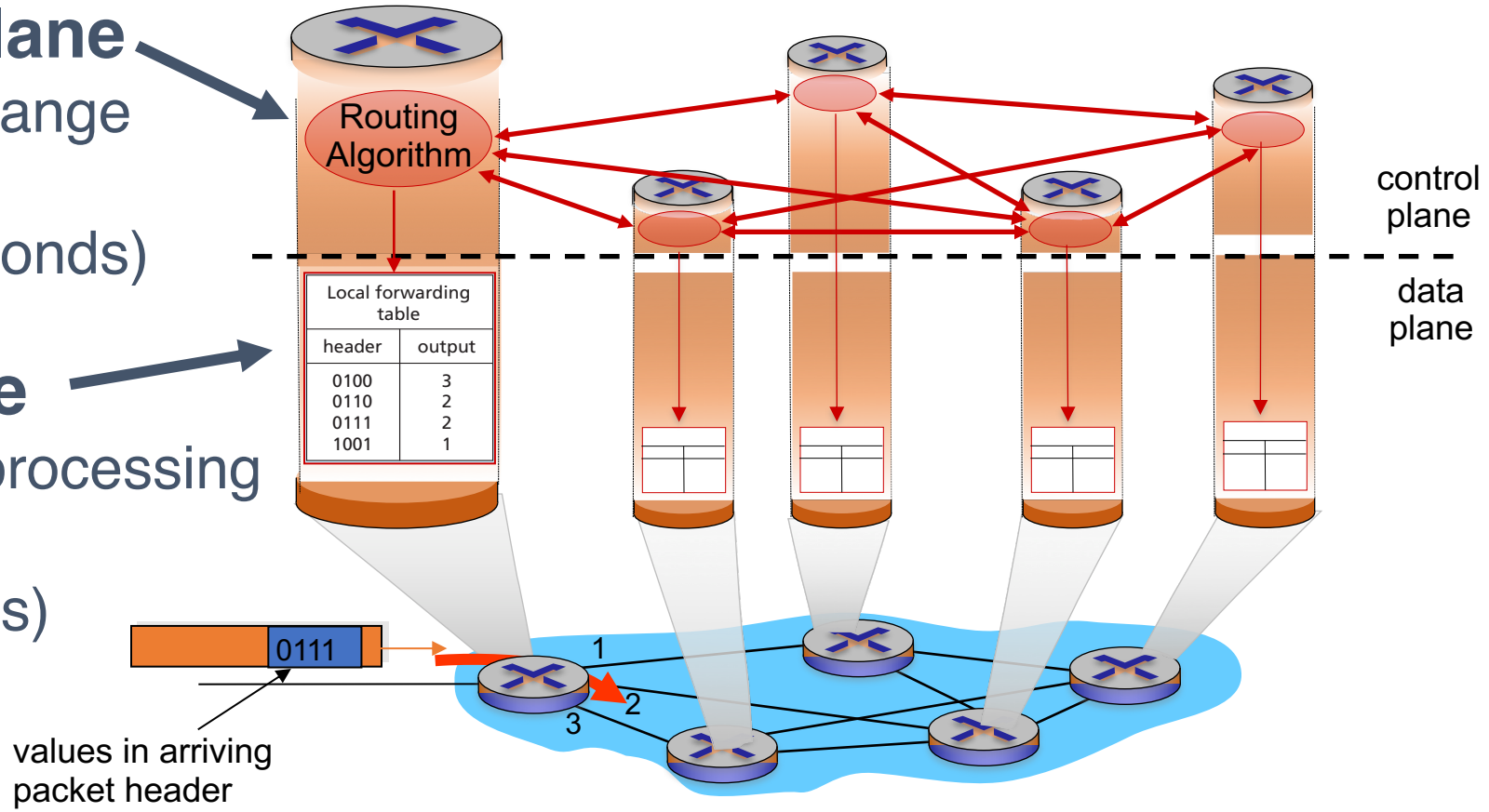
*Counter-intuitive:* Operators may set the link metric to achieve certain shortest-path trees with the protocol

# Software-Defined Networking

# Control & Data Planes inside a router

**Control plane**
per route-change processing
(~ a few seconds)

**Data plane**
per-packet processing
(~ tens of nanoseconds)

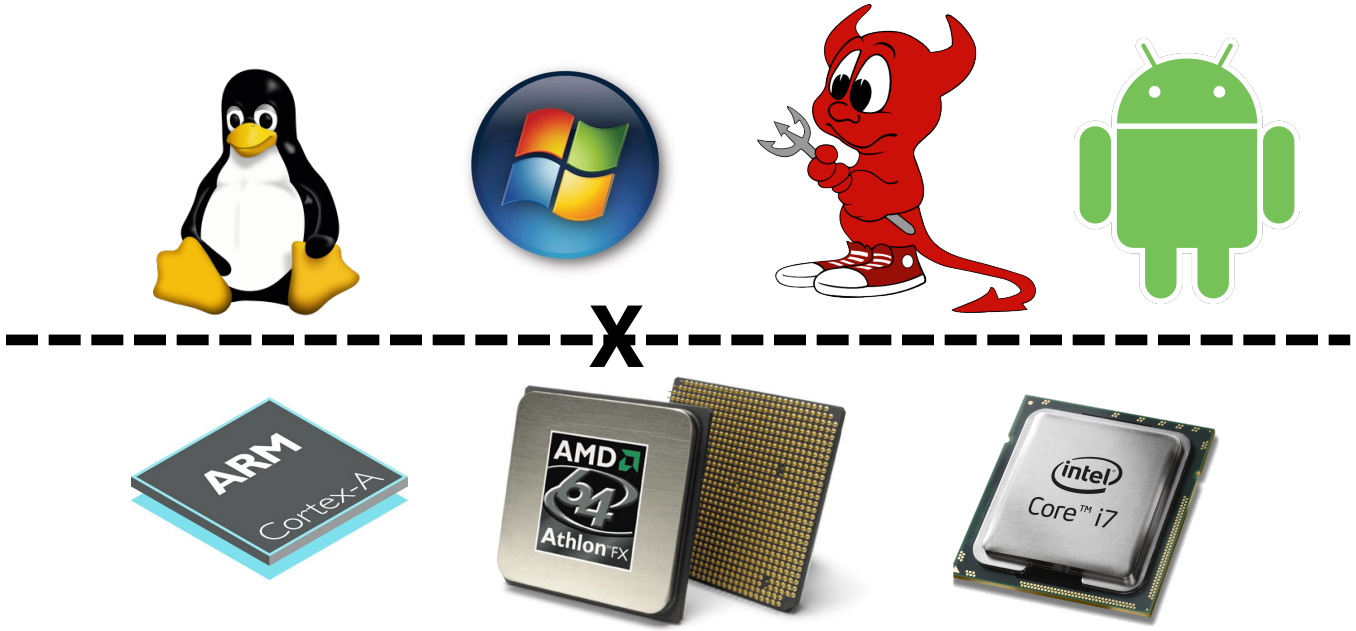



Traditionally:

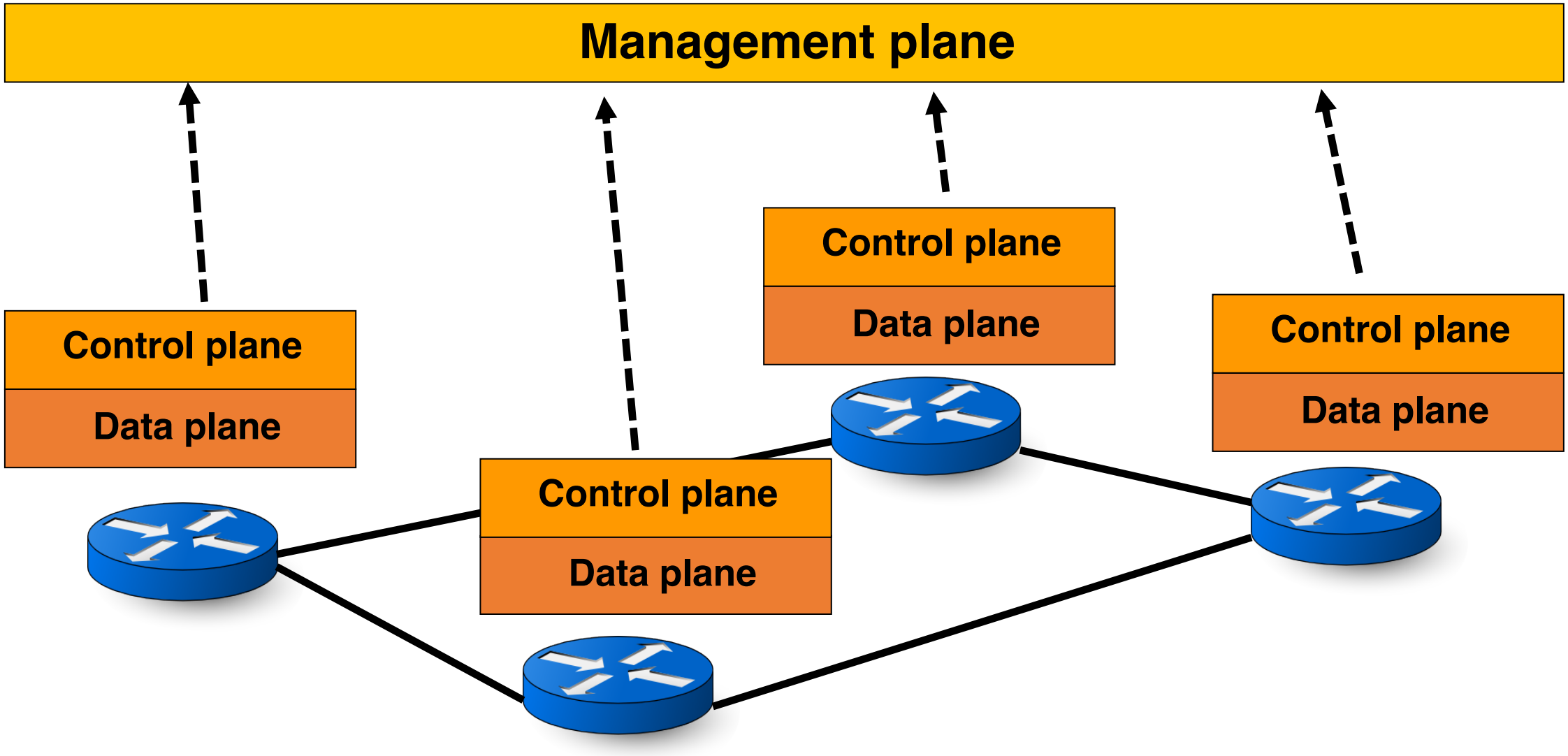
Individual routing algorithm components *in each and every router* interact in the control plane

# Problems with traditional routers

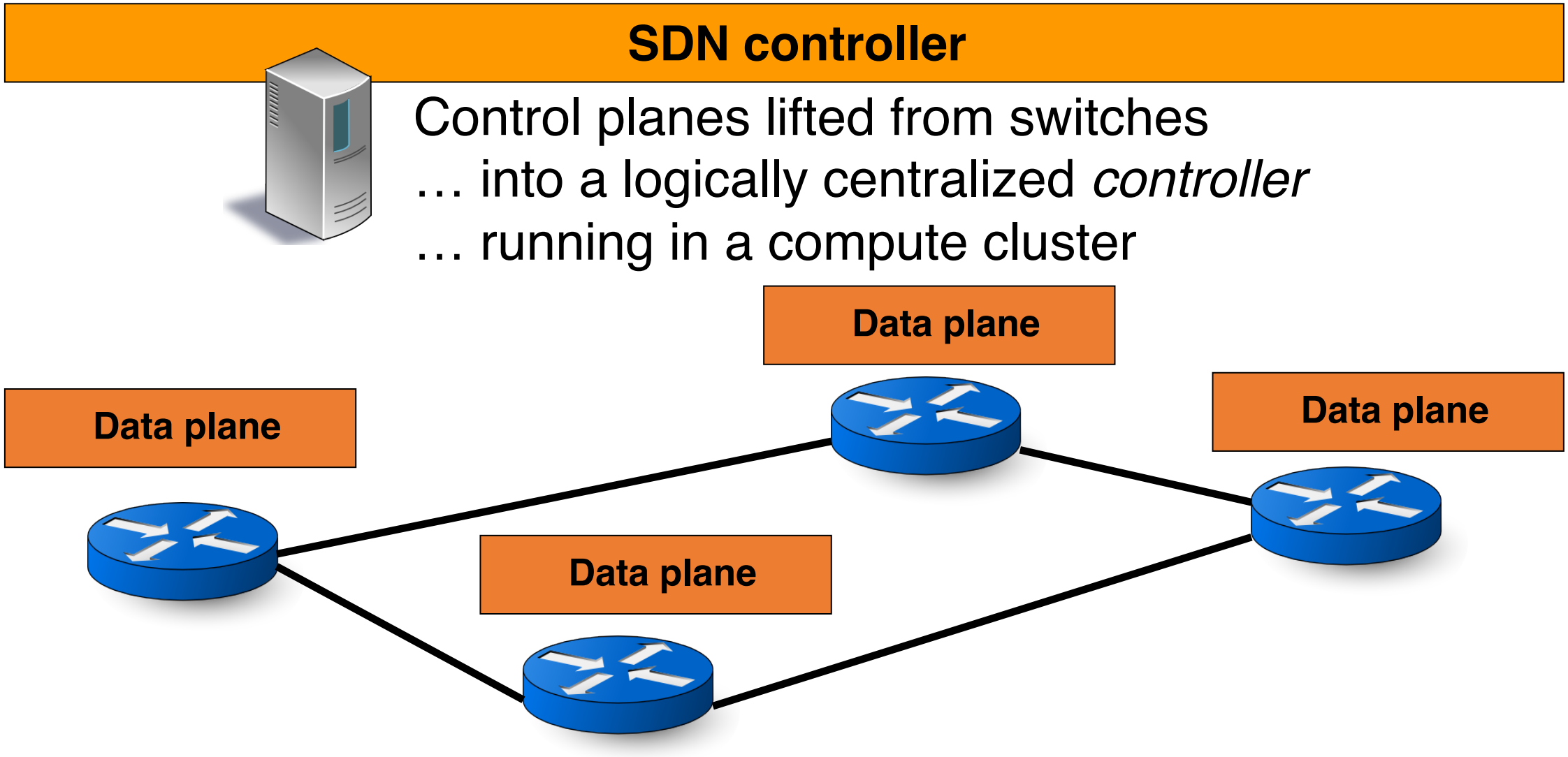
- Management decisions tied to distributed protocols
  - Ex: Set OSPF link weights to force traffic through desired path
  - Ex: Non-deterministic network state after a link failure
- Data and control plane controlled by vendors: proprietary interfaces



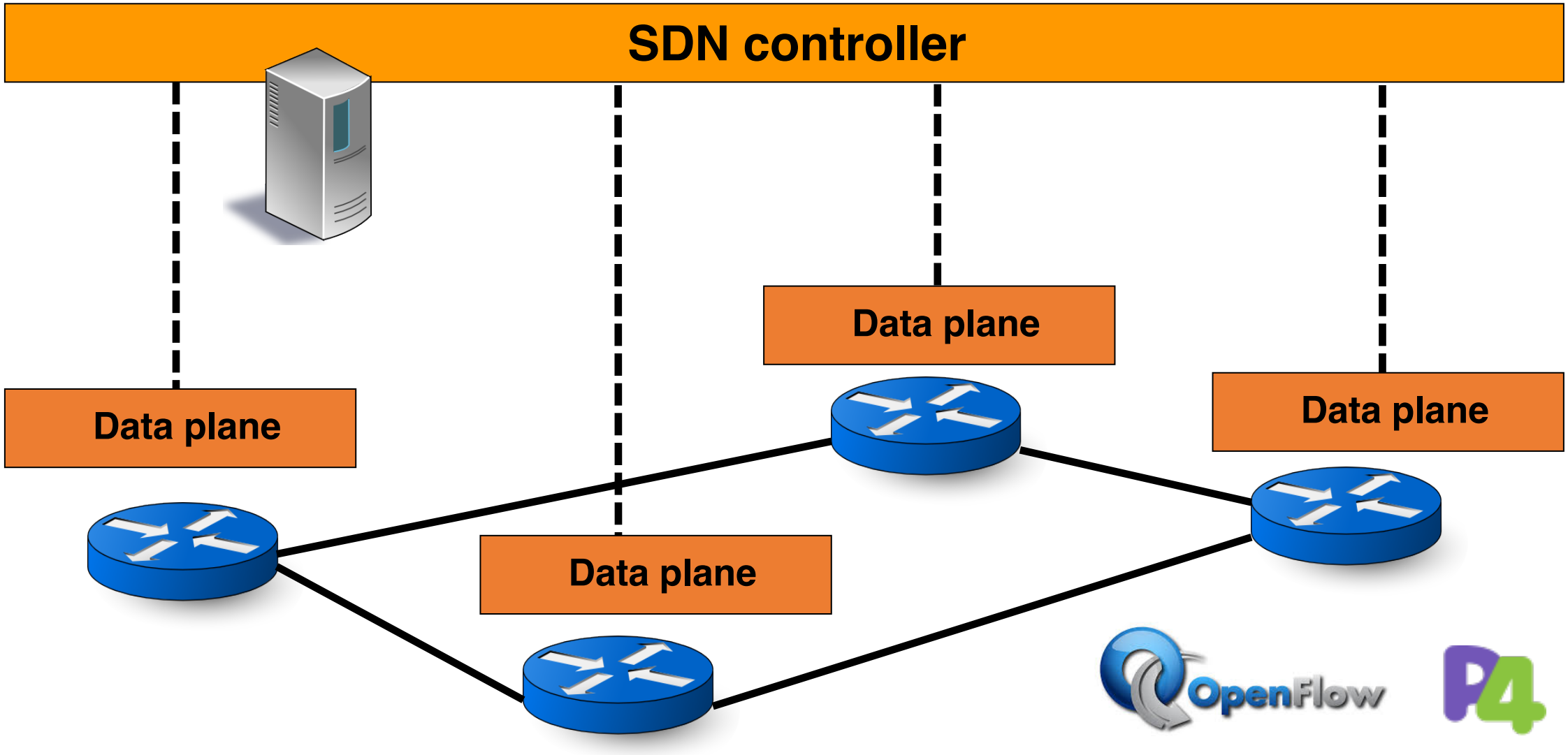
# Traditional IP network



# SDN (1/2): Centralized control plane

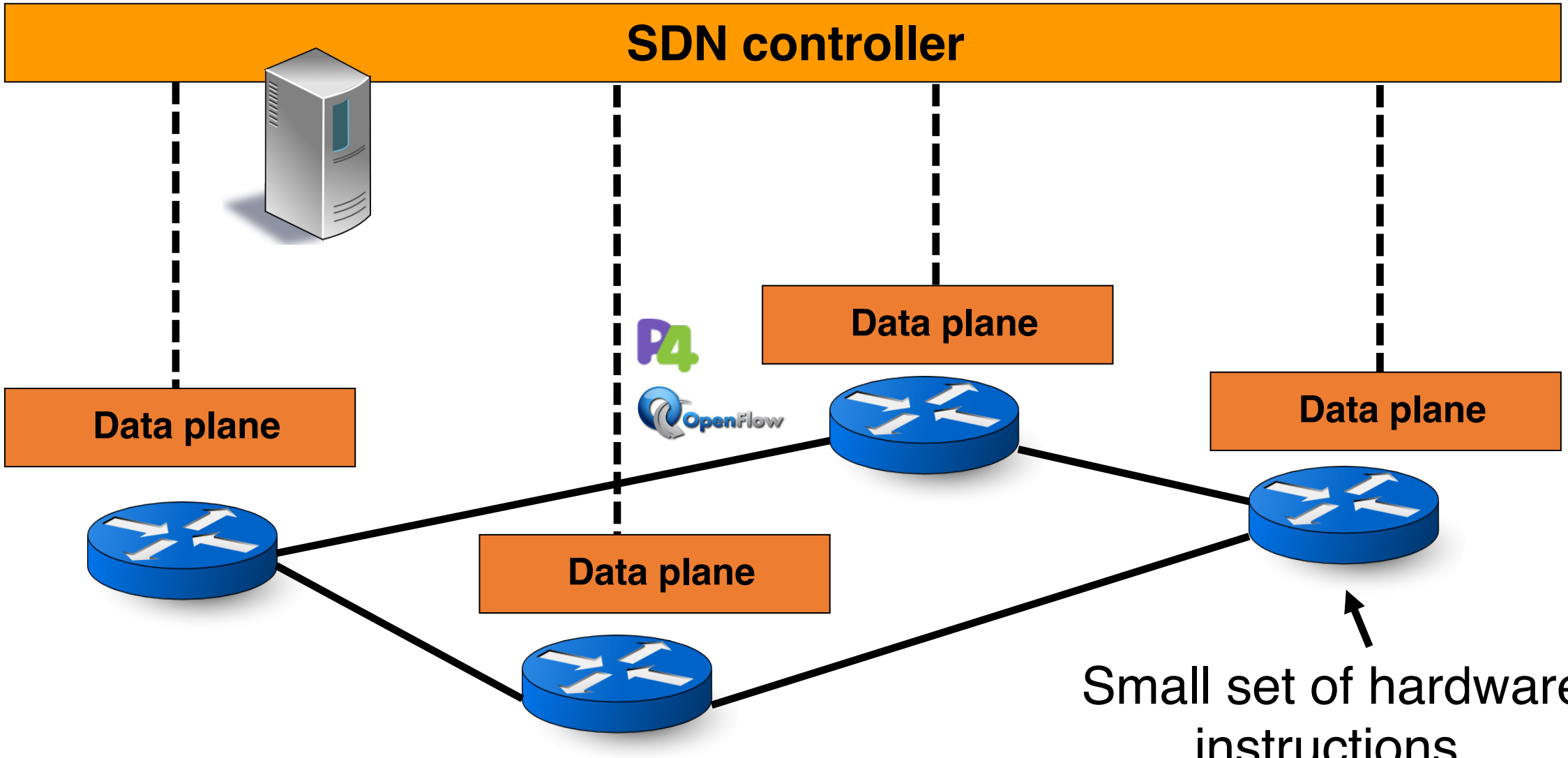


# SDN (2/2): Open interface to data plane



# Some immediate consequences…

# (1) Simpler switches



# Data plane primitive: Match-action rules

- Match arbitrary bits in the packet header

```
Match: 1000x01xx01001x
```

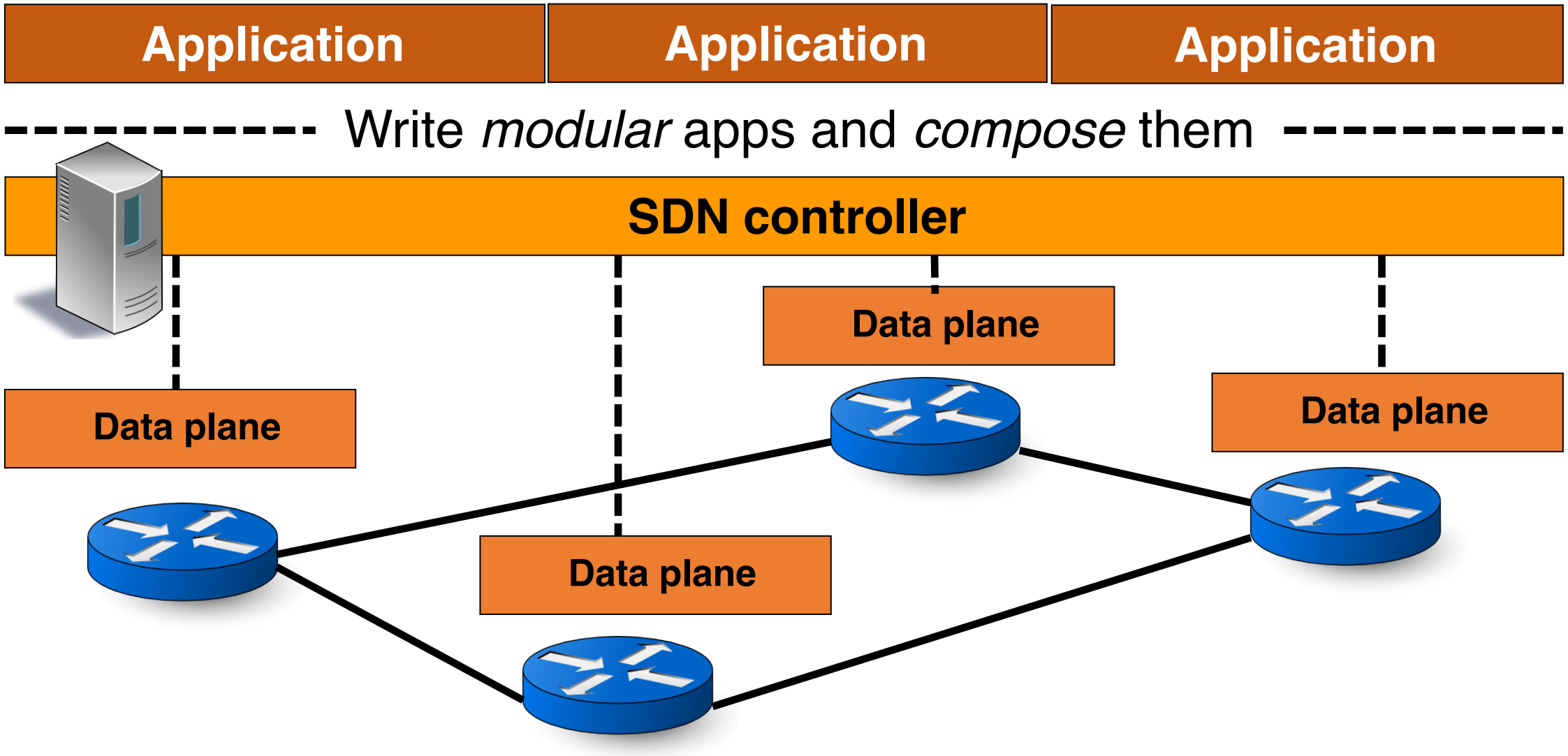
  - Match on any header, or new header
  - Allows any flow granularity
- Actions
  - Forward to port(s), drop, send to controller, count,
  - Overwrite header with mask, push or pop, …
  - Forward at specific bit-rate

```
Action: fwd(port 2)
```

- Prioritized list of rules

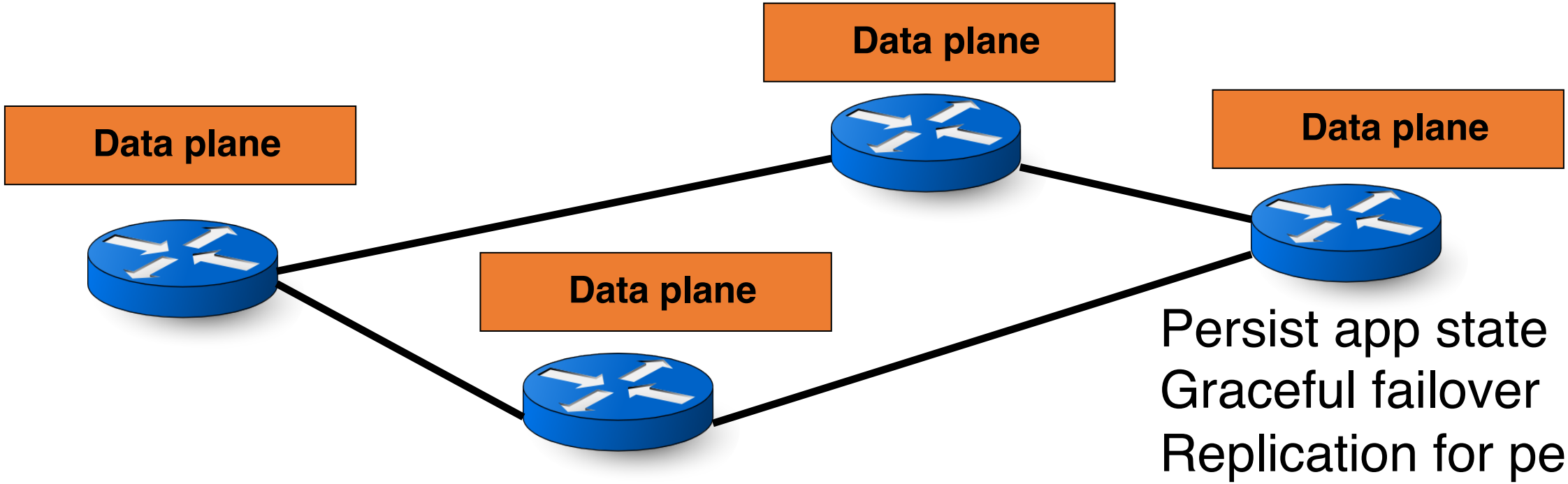
```
Priority: 65500
```

# (2) Network programming abstractions



# (3) Unified network operating system

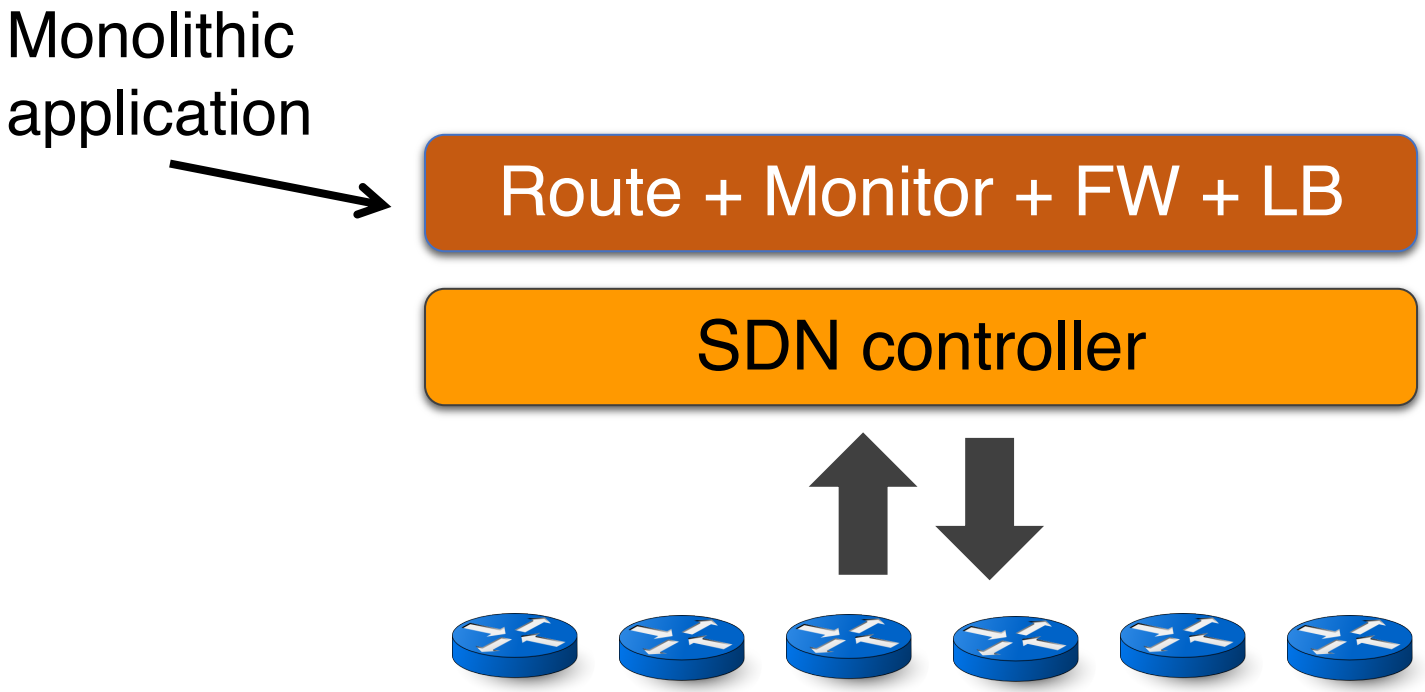
Separate distributed system concerns from administrator intent




Persist app state
Graceful failover
Replication for perf

# Composition of Policies

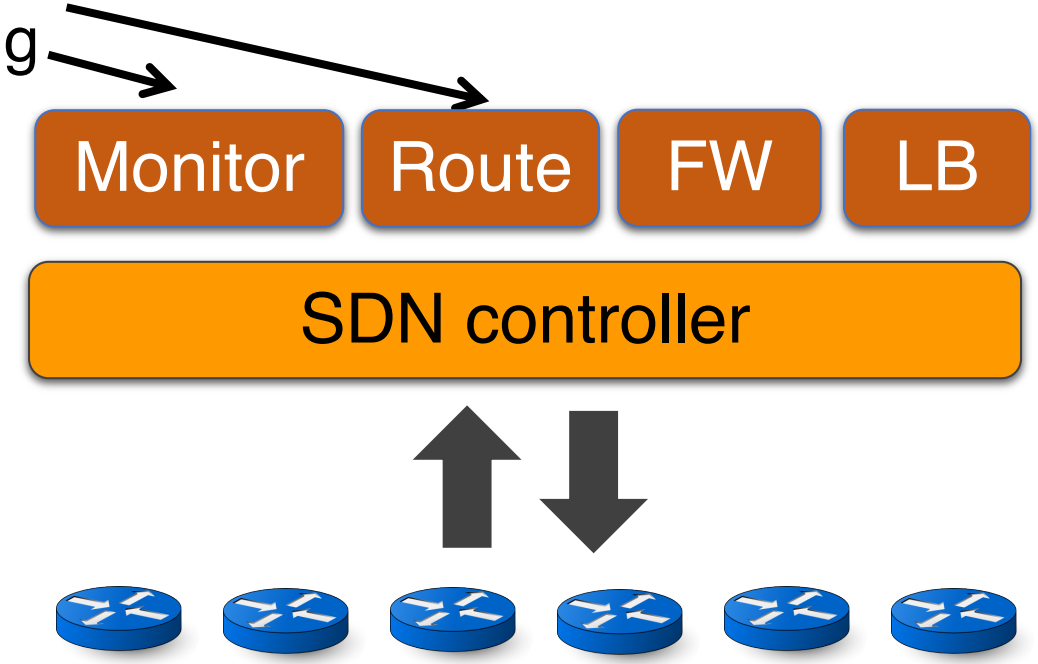
# Combining many networking tasks



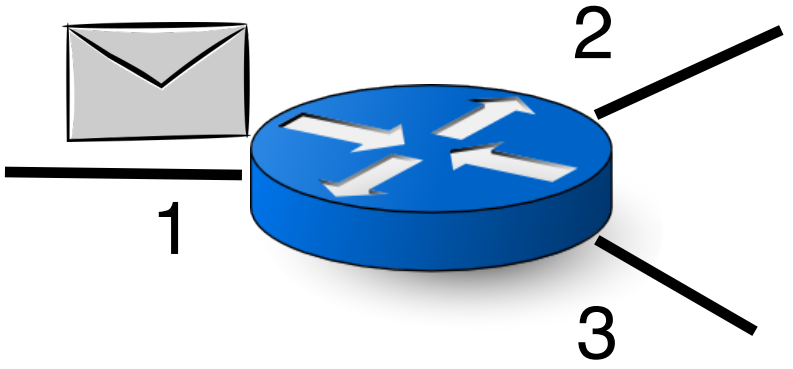

Hard to program, test, debug, reuse, port, …

# Modular controller applications

Each module *partially* specifies the handling of the traffic



# Network policy as a function




- Located packet: headers + switch + port
- Policy: function of a located packet
  - To a *set* of located packets: multicast, drop, forward
- Function can modify packets
  - Headers and location

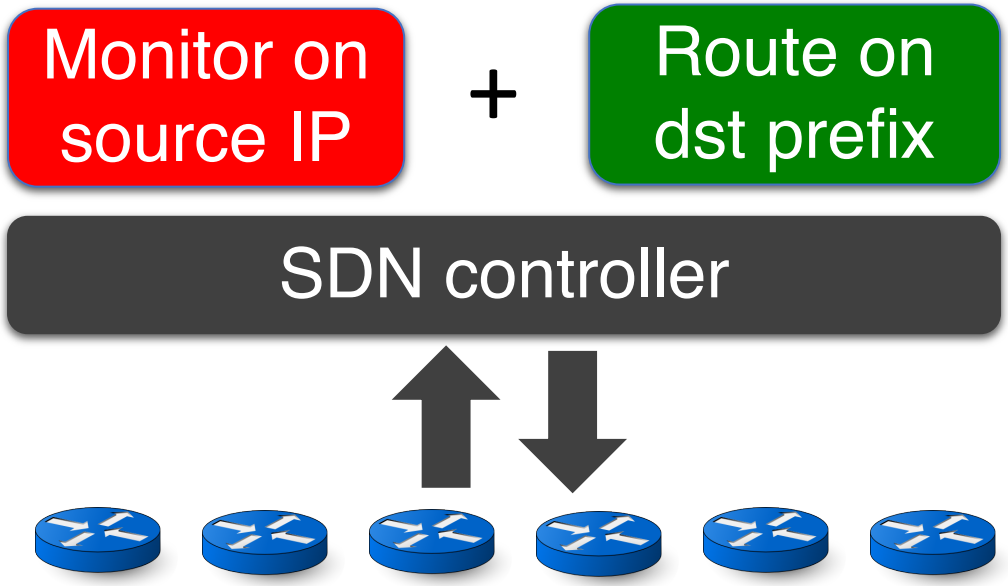
**Match** **Action**

```
dstip == 1.2.3.4 & srcport == 80 → port = 3, dstip = 10.0.0.1
```

# Parallel composition (+)

```
srcip == 5.6.7.8 → count
srcip == 5.6.7.9 → count
```

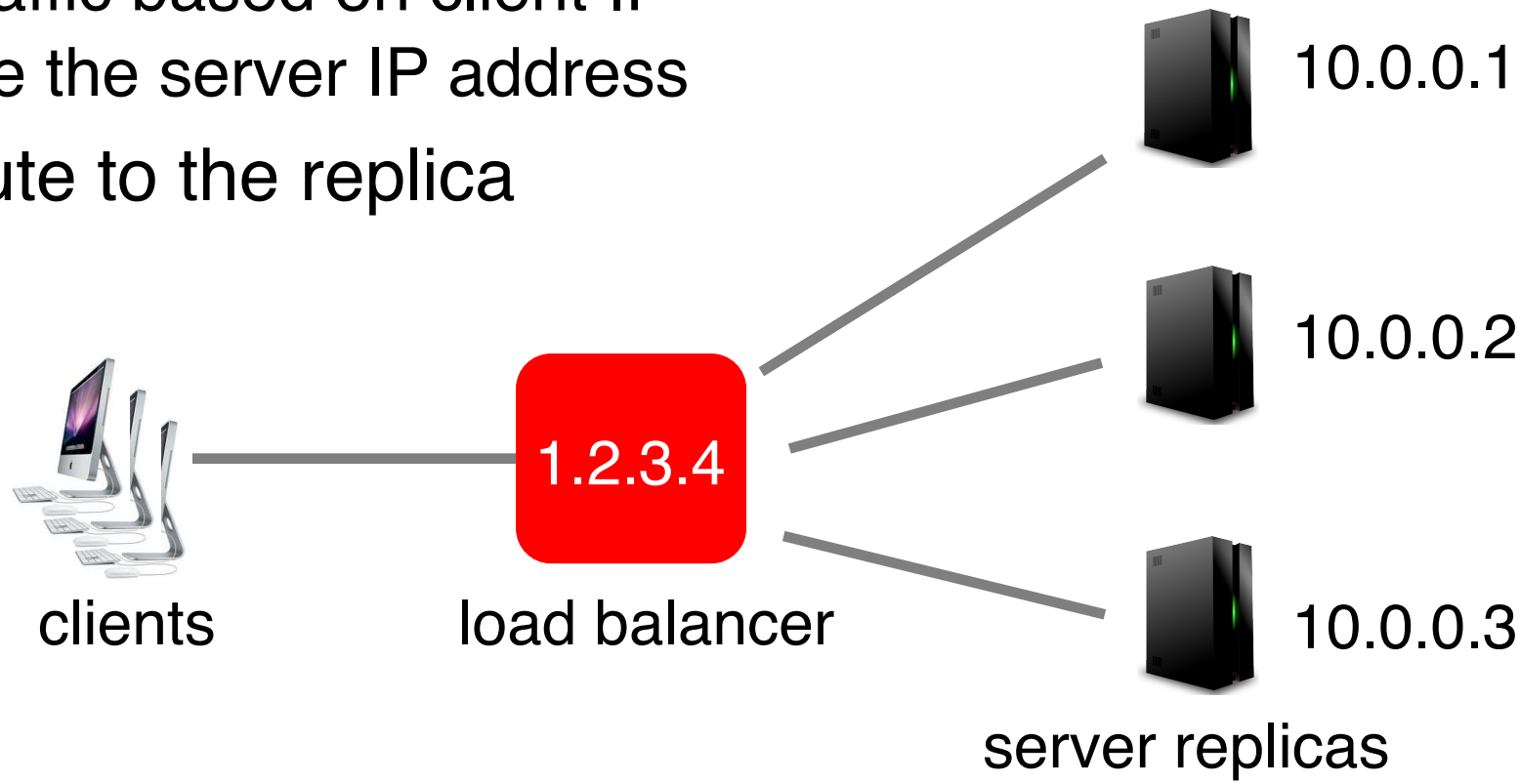
```
dstip == 1.2/16 → fwd(1)
dstip == 3.4.5/24 → fwd(2)
```




```
srcip == 5.6.7.8, dstip == 1.2/16 → fwd(1), count
srcip == 5.6.7.8, dstip == 3.4.5/24 → fwd(2), count
srcip == 5.6.7.9, dstip == 1.2/16 → fwd(1), count
srcip == 5.6.7.9, dstip == 3.4.5/24 → fwd(2), count
```

# Example: Server load balancer

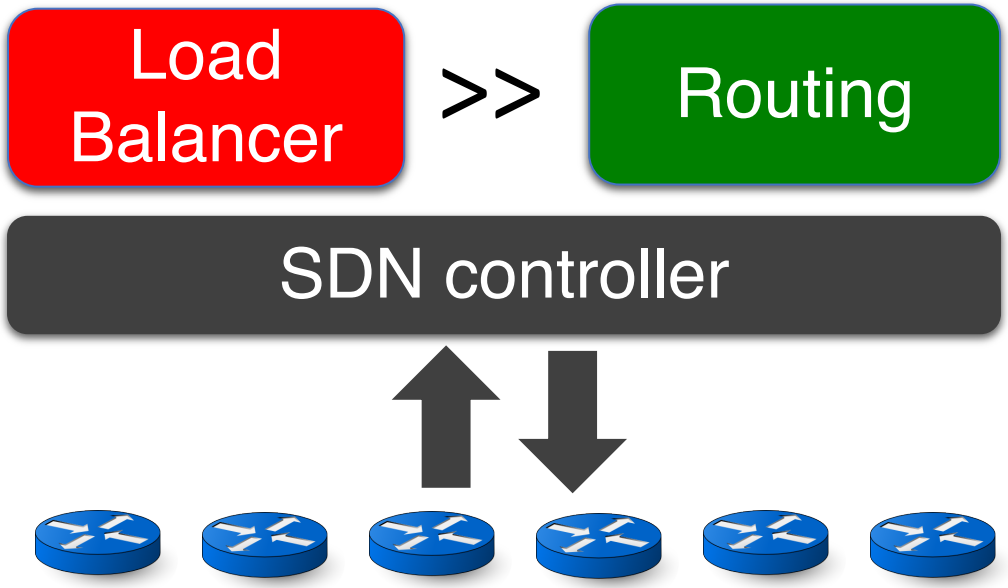
- Spread client traffic over server replicas
  - Public IP address for the service
  - Split traffic based on client IP
  - Rewrite the server IP address
- Then, route to the replica



# Sequential composition (>>)

```
srcip==0*, dstip==1.2.3.4 → dstip=10.0.0.1
srcip==1*, dstip==1.2.3.4 → dstip=10.0.0.2
```

```
dstip==10.0.0.1 → fwd(1)
dstip==10.0.0.2 → fwd(2)
```

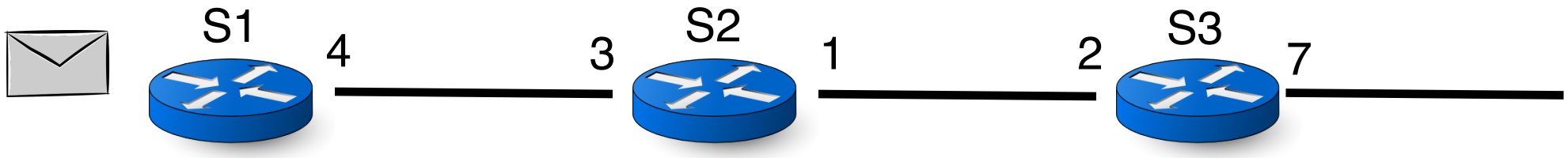



```
srcip==0*, dstip==1.2.3.4 → dstip = 10.0.0.1, fwd(1)
srcip==1*, dstip==1.2.3.4 → dstip = 10.0.0.2, fwd(2)
```

# Implications & Challenges of SDN

# What does SDN make possible/easy?

- Expressing forwarding intent *directly*
  - Example path: `sw==S1→fwd(4) + sw==S2→fwd(1) + sw==S3→fwd(7)`




- Reading state: Measurement through counters
  - Measure exactly the traffic you care about
  - Can modify forwarding to make measurements more accurate!

- *Deterministically* and swiftly handle data plane failures
  - Google's B4: Failover to pre-computed outcomes

# What does SDN make possible/easy?

- Network policy verification
  - Correctness: Reachability, loop-freedom, SLO violations, etc.
  - Performance

- Better router data plane design
  - Decouple evolution of router instruction sets and network policy

- Apply the SDN philosophy to system design as a whole
  - Stateful “network functions” that reside in the core of the network
  - Operating systems
  - End host NICs
  - <insert your idea here!>

# Technical challenges of SDN

- Scalability: controller responsible for many routers
- Response time: Delays between controller and routers
- Reliability: surviving failures of the controller & data plane
- Consistency:
  - Ensuring multiple controllers behave consistently
  - Ensuring controller policy is faithfully implemented
- Security:
  - Entire network may be owned if the controller is vulnerable
- Interoperability: legacy routers and neighboring domains