Administrivia

- Review 1 paper assignments are out
  - See me after lecture if you haven't yet received your assignment
- MUD: Send me your top 1—3 questions on this lecture
- Brainstorm project ideas (teams of 1—3)
  - Discuss with me during office hours or by appointment
- Today: Some more examples of division of labor….

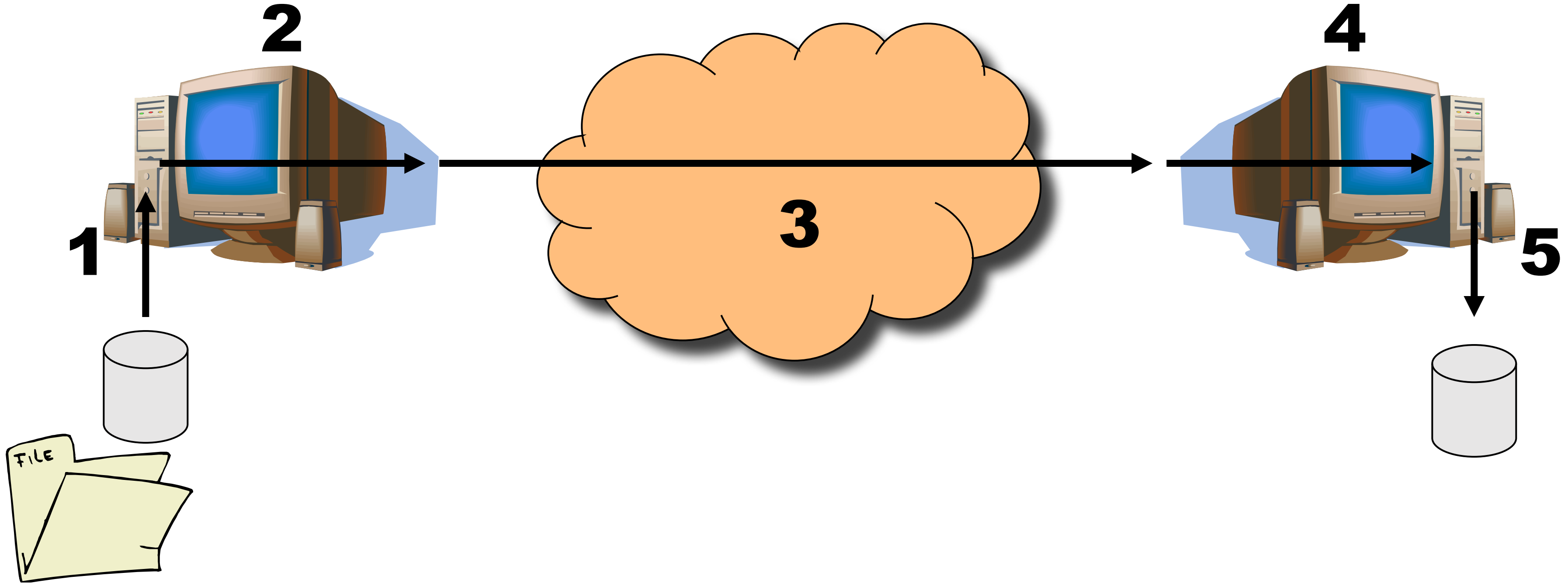
End-to-end arguments in system design

(*ACM Trans. on Computer Systems*, November 1984)

J. Saltzer, D. Reed, and D. Clark

End-to-end argument

- Operations should occur only at the end points
- … unless needed for performance optimization



Many things can go wrong: disk errors, software errors, hardware errors, communication errors, …

Trade-offs

- Put functionality at each hop
  - All applications pay the price
  - End systems *still* need to check for errors

- Place functionality only at the ends
  - Slower error detection
  - End-to-end retransmission wastes bandwidth

- Compromise solution?
  - Reliable end-to-end transport protocol (TCP)
    - Plus file checksums to detect file-system errors
  - “Reasonably reliable” communication network

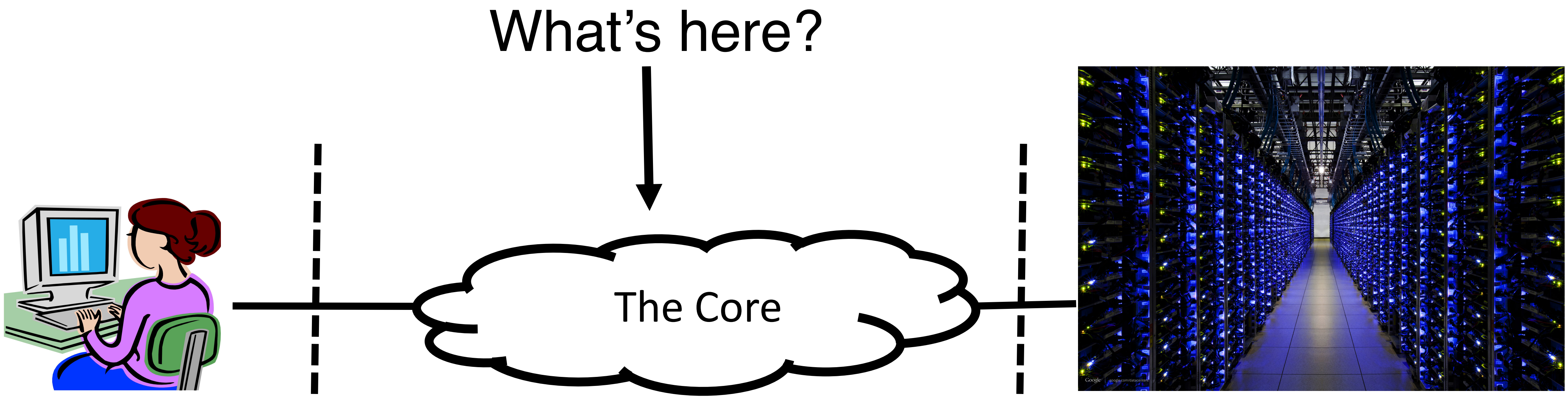
Control-Data Plane Separation

Lecture 3, Computer Networks (198:552)

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Edge vs. Core

- Core: the network consisting of routers
- Edge: the endpoints



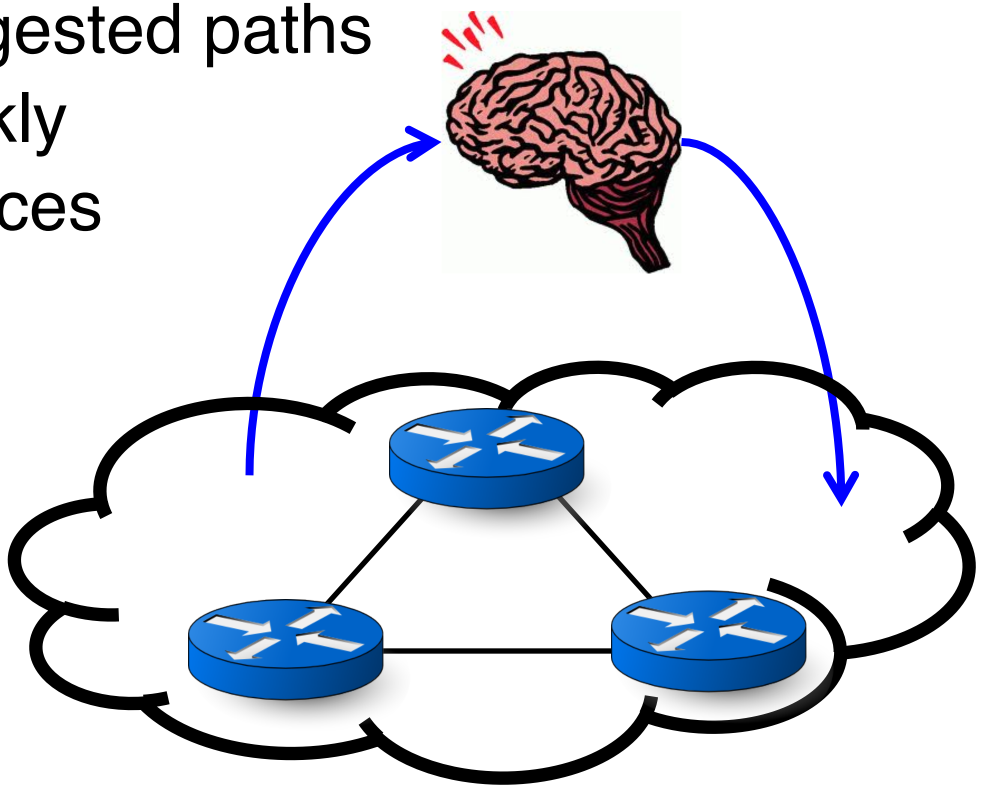
Core: Split into *data* and *control* planes

- Data plane: handle packets
  - Handle individual packets as they arrive
  - Forward, drop, or buffer
  - Mark, schedule, measure, …
- Control plane: handle events
  - Compute paths through the network
  - Track changes in network topology
  - Reserve resources along a path

Motivated by need for high-speed packet forwarding

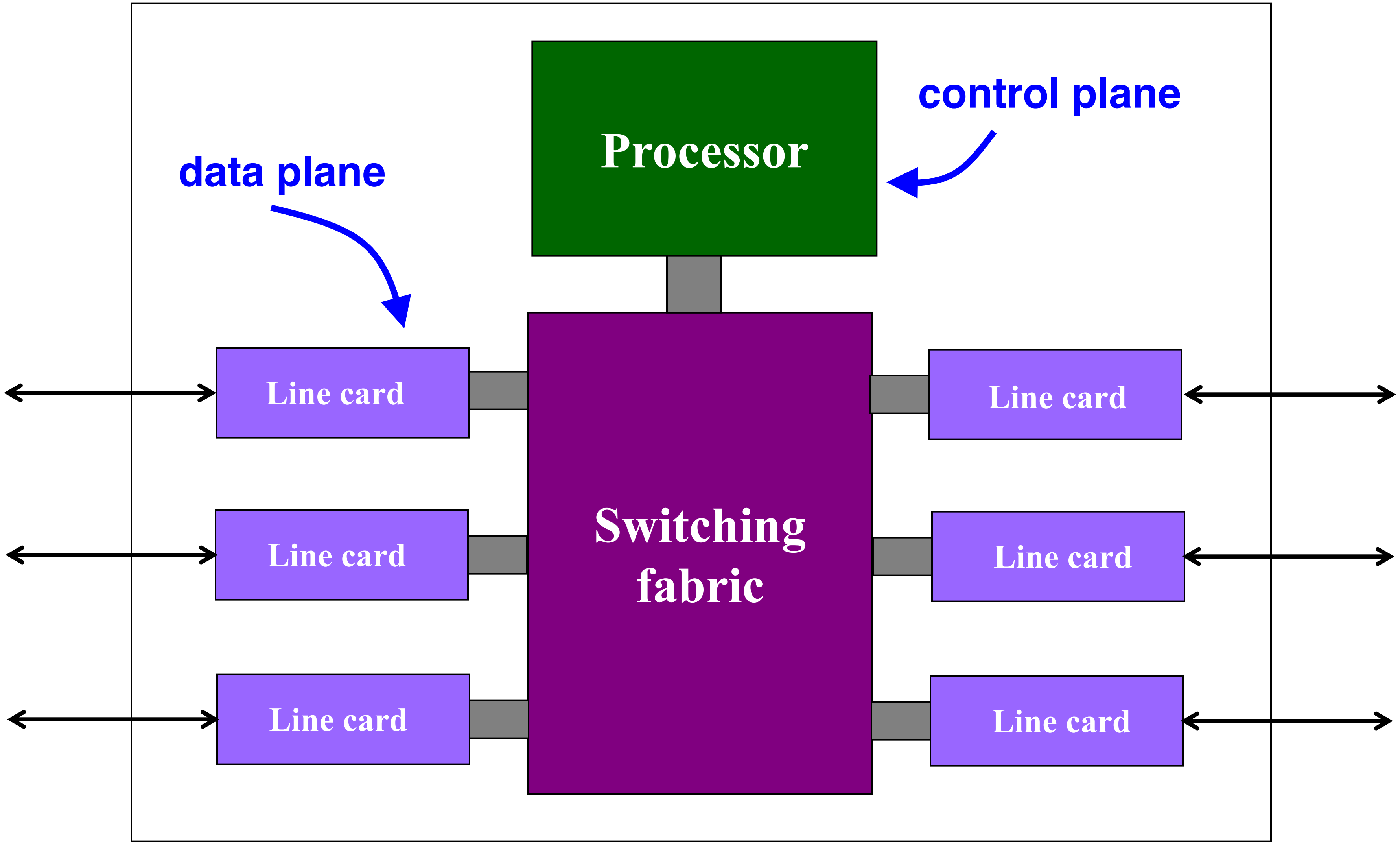
Adding the management plane

- Make networks run *well*
  - The right traffic reaches the right destination
  - Traffic flows over short, uncongested paths
  - Failure recovery happens quickly
  - Routers don't run out of resources



- A control loop with the network
  - Measure (sense): topology, traffic, performance, …
  - Control (actuate): configure control and data planes

Data and Control Planes in a Router



Forwarding vs. *Routing*

- Forwarding: data plane
  - Router directs packets to an output port
  - … by looking up a *forwarding table*

Destination IP	Subnet mask	Output port
10.0.0.0	255.255.255.0	4
8.4.5.3	255.0.0.0	1

- Routing: control plane
  - Routers talk amongst themselves
  - .. to *compute* the forwarding table

Each router creates its own forwarding table

… but the computation itself is distributed.

Routing protocols enable FT computation

- What does the protocol compute?
  - Spanning tree, shortest path, local policy, arbitrary end-to-end paths
- What algorithm does the protocol run?
  - Spanning-tree construction, distance vector, link-state routing, path-vector routing, source routing, end-to-end signaling
- How do routers learn end-host locations?
  - Learning/flooding, injecting into the routing protocol, dissemination using a different protocol, and directory server

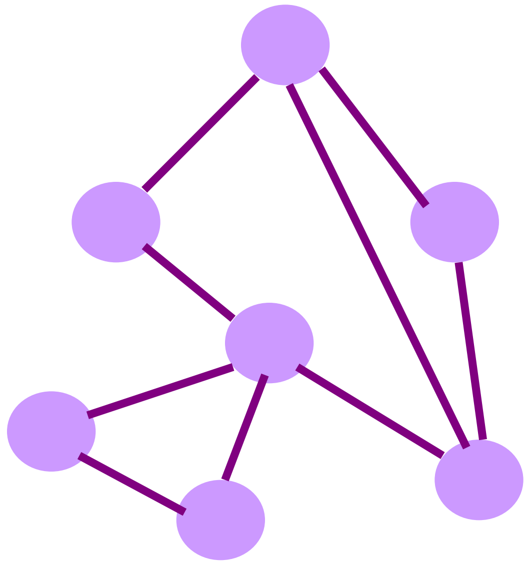
What does the protocol compute?

(the outcome, not the computation)

Different ways to represent paths

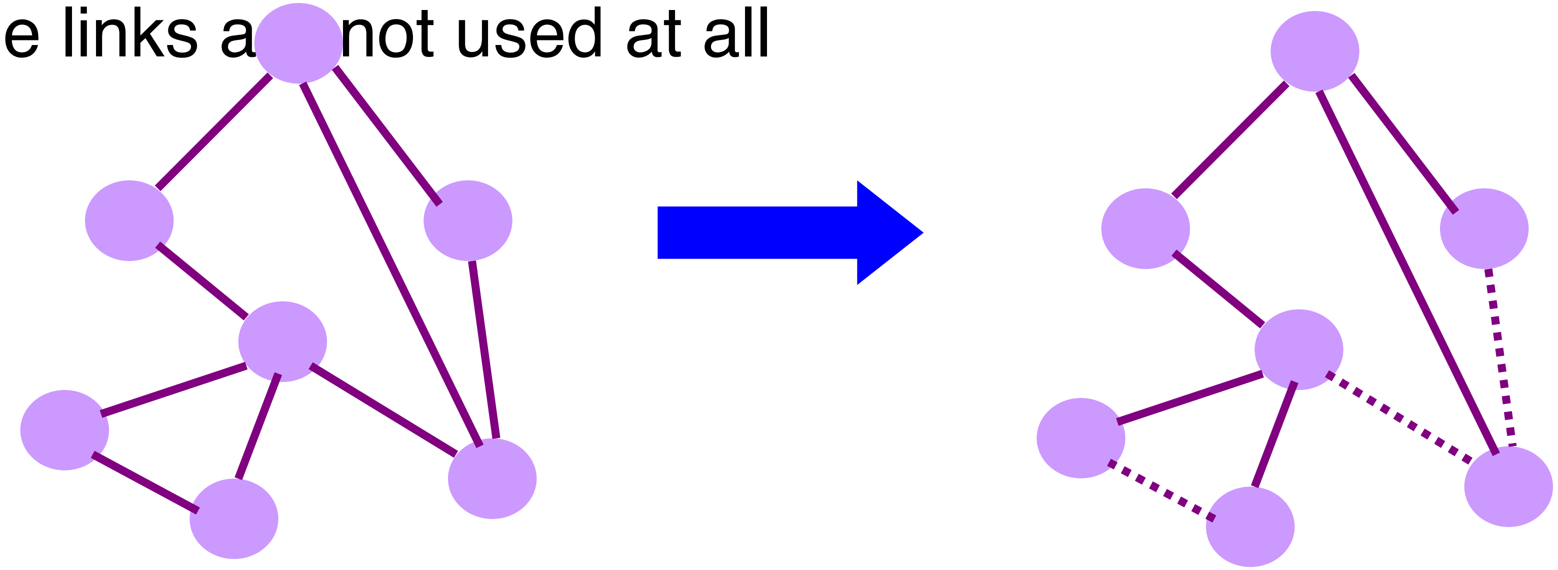
- Trade-offs
  - State required to represent the paths
  - Efficiency of the resulting paths
  - Ability to support multiple paths
  - Complexity of computing the paths
  - Which nodes are in charge

- Applied in different settings
  - LAN, intra-domain, inter-domain



Spanning tree (Ethernet)

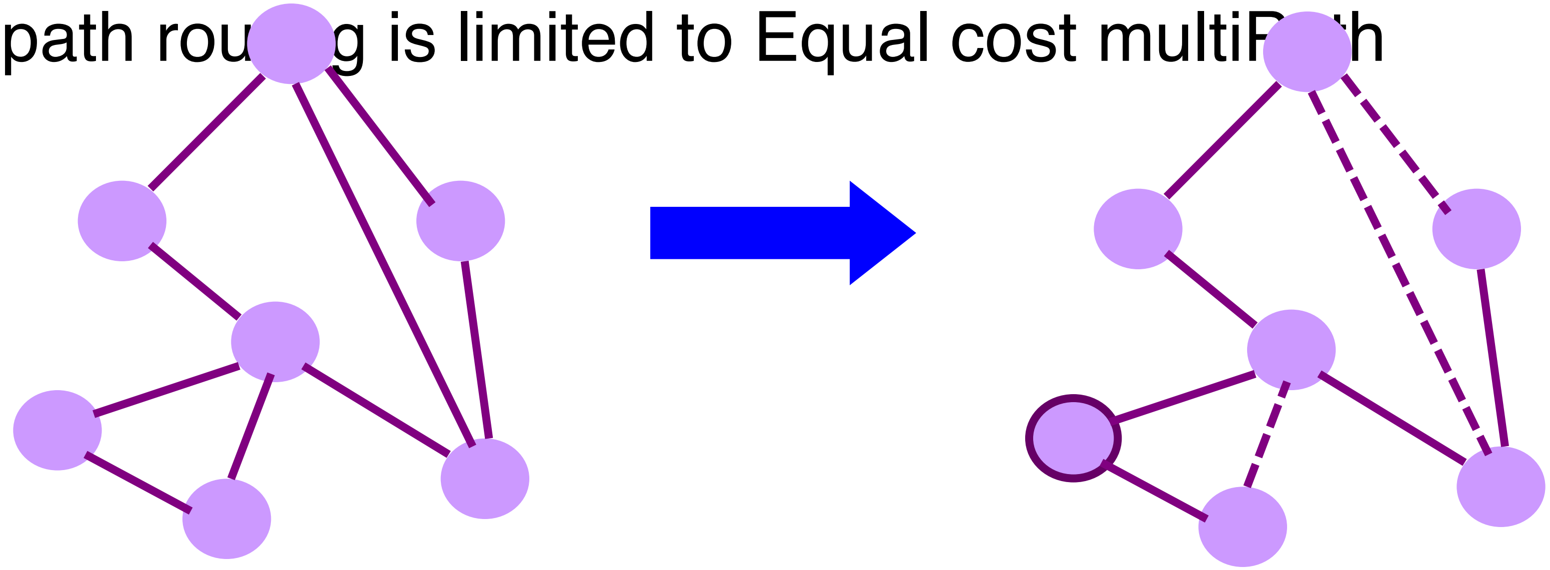
- One tree that reaches every node
  - Single path between each pair of nodes
  - No loops, so can support broadcast easily
- Disadvantages
  - Paths are sometimes long
  - Some links are not used at all



Shortest paths (OSPF/IS-IS)

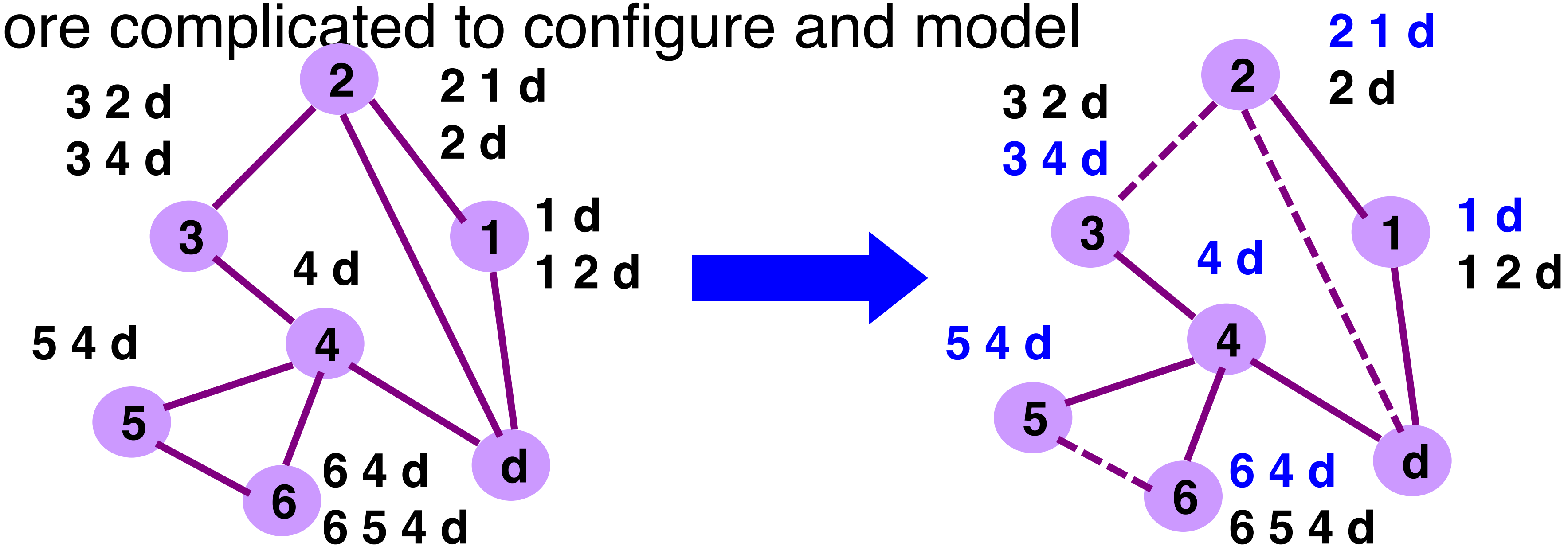
Set by network administrator

- Shortest path(s) between each pair of nodes
  - Separate shortest-path tree rooted at each node
  - Minimum hop count or minimum sum of edge weights
- Disadvantages
  - All nodes need to agree on the link metrics
  - Multipath routing is limited to Equal cost multiPath



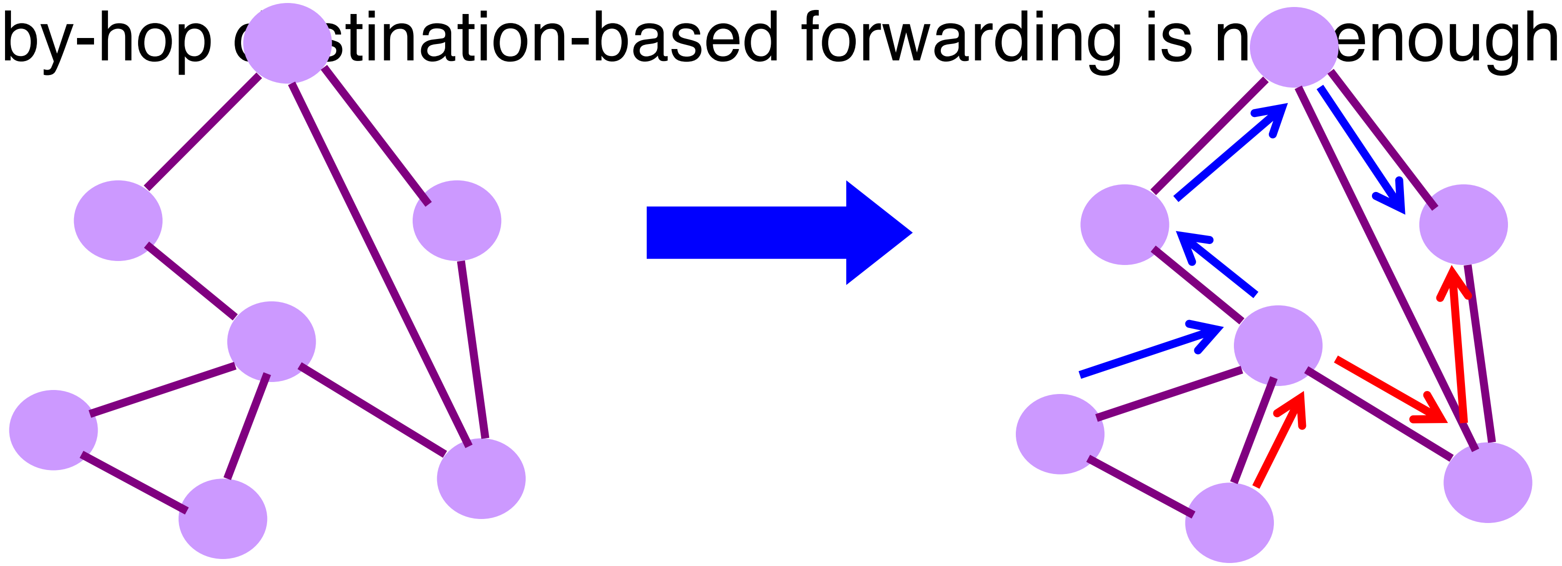
Local policy at each hop (BGP)

- Locally best path
  - Local policy: each node picks the path it likes best
  - … among the paths chosen by its neighbors
- Disadvantages
  - More complicated to configure and model



End-to-end path selection (IP src route)

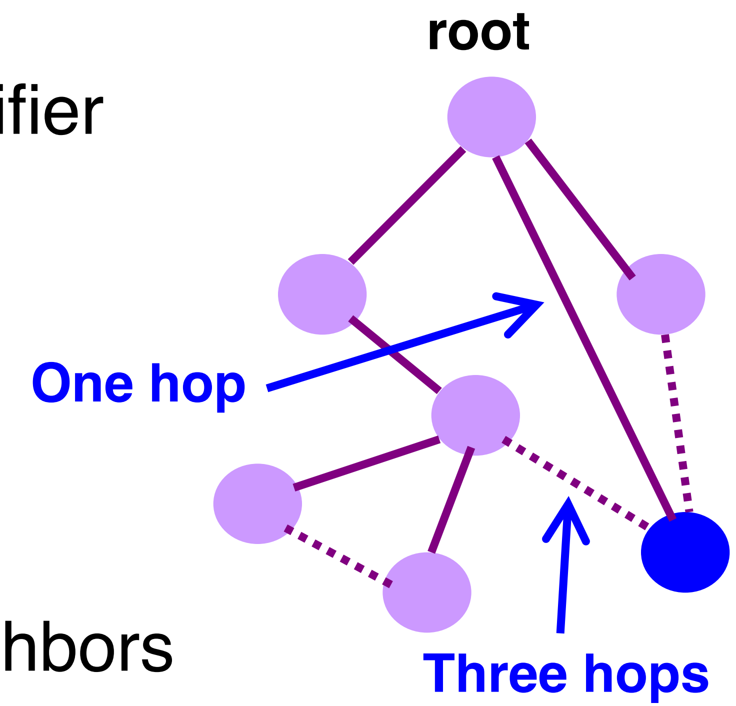
- End-to-end path selection
  - Each node picks its own end to end paths
  - … independent of what other paths other nodes use
- Disadvantages
  - More state and complexity in the nodes
  - Hop-by-hop destination-based forwarding is not enough



How to compute paths?

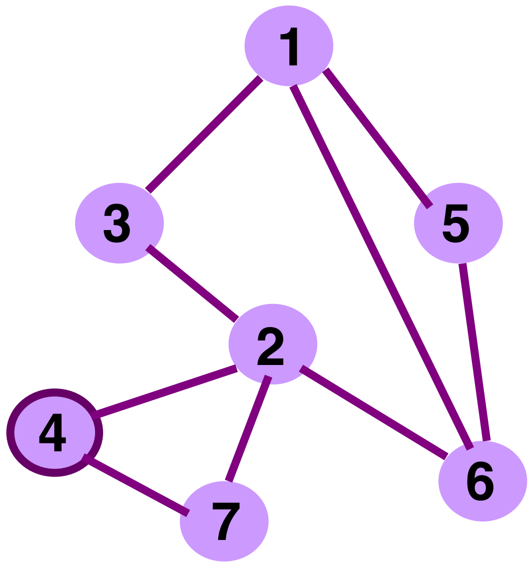
Spanning tree algorithm (Ethernet)

- Elect a root
    - The switch with the smallest identifier
    - And form a tree from there
- Algorithm
    - Repeatedly talk to neighbors
        - “I think node Y is the root”
        - “My distance from Y is d”
    - Update information based on neighbors
        - Smaller id as the root
        - Smaller distance d+1
    - Don’t use interfaces not in the path



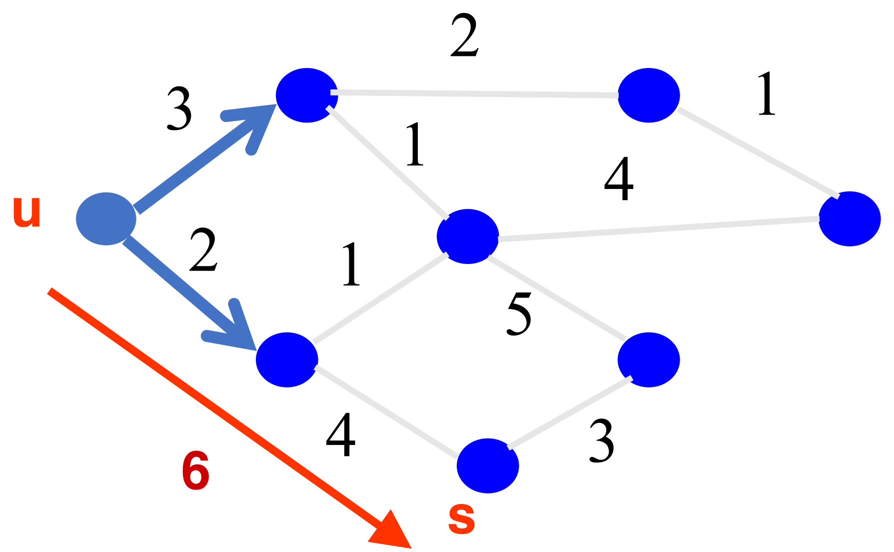
Spanning tree example: switch #4

- Switch #4 thinks it is the root
  - Sends (4, 0) message to 2 and 7
- Switch #4 hears from #2
  - Receives (2, 0) message from 2
  - … and thinks that #2 is the root
  - And realizes it is just one hop away
- Switch #4 hears from #7
  - Receives (2, 1) from 7
  - And realizes this is a longer path
  - So, prefers its own one-hop path
  - And removes 4-7 link from the tree



Shortest-path problem

- Compute: *path costs* to all nodes
  - From a given source u to all other nodes
  - Cost of the path through each outgoing link
  - Next hop along the least-cost path to s



Link-state: Dijkstra's algorithm

- Flood the topology information to all nodes
- Each node computes shortest paths to other nodes

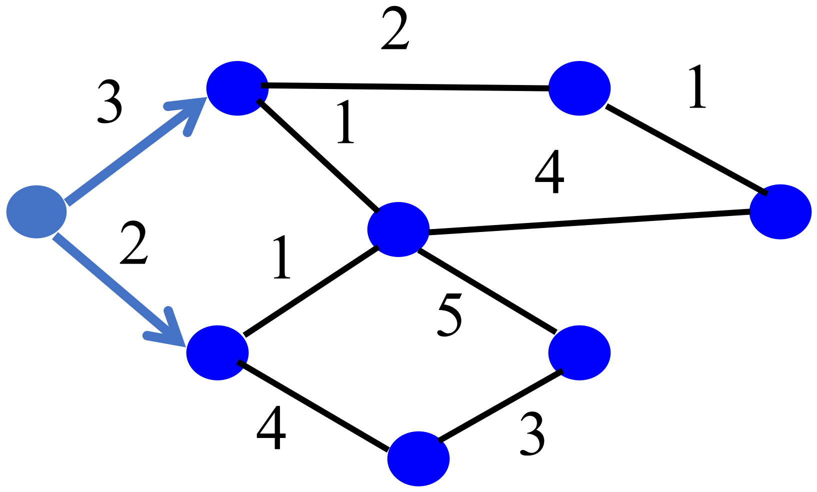
Initialization

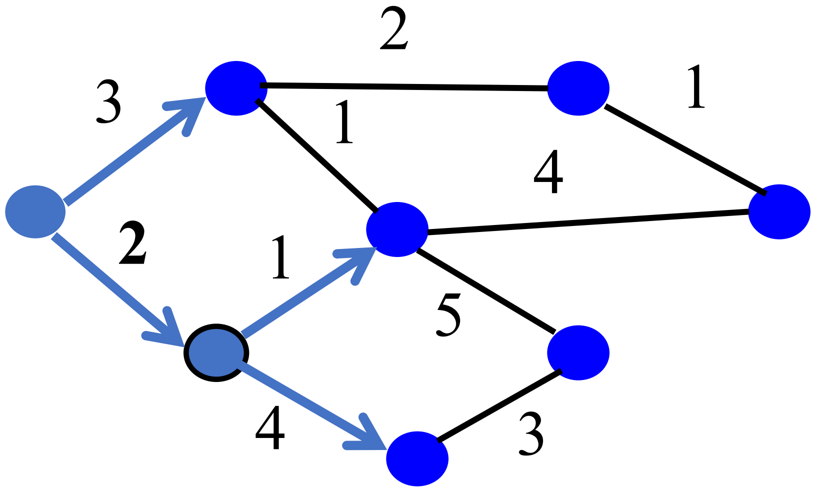
```
S = {u}
for all nodes v
   if (v is adjacent to u)
      D(v) = c(u,v)
   else D(v) = ∞
```

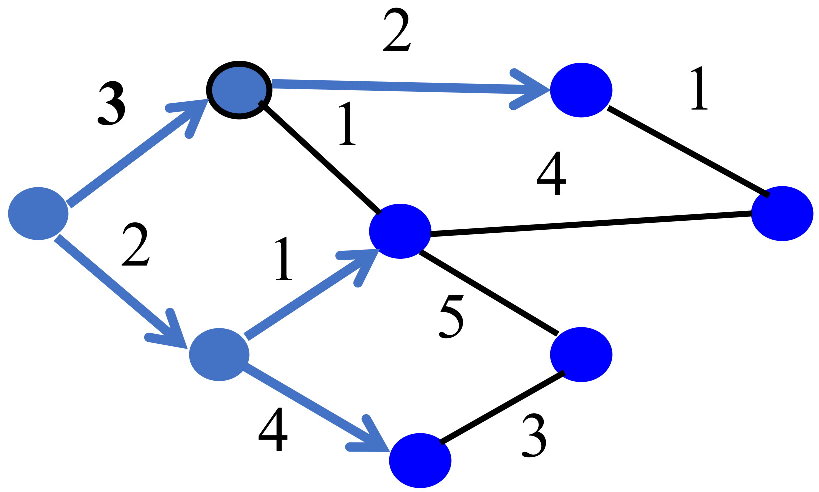
Loop

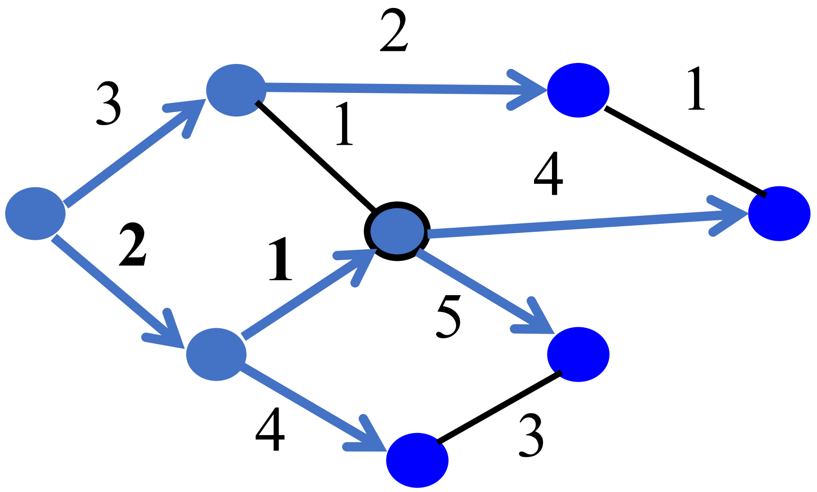
```
add w with smallest D(w) to S
update D(v) for all adjacent v:
   D(v) = min{D(v), D(w) + c(w,v)}
until all nodes are in S
```

Link-state routing example (OSPF/IS-IS)

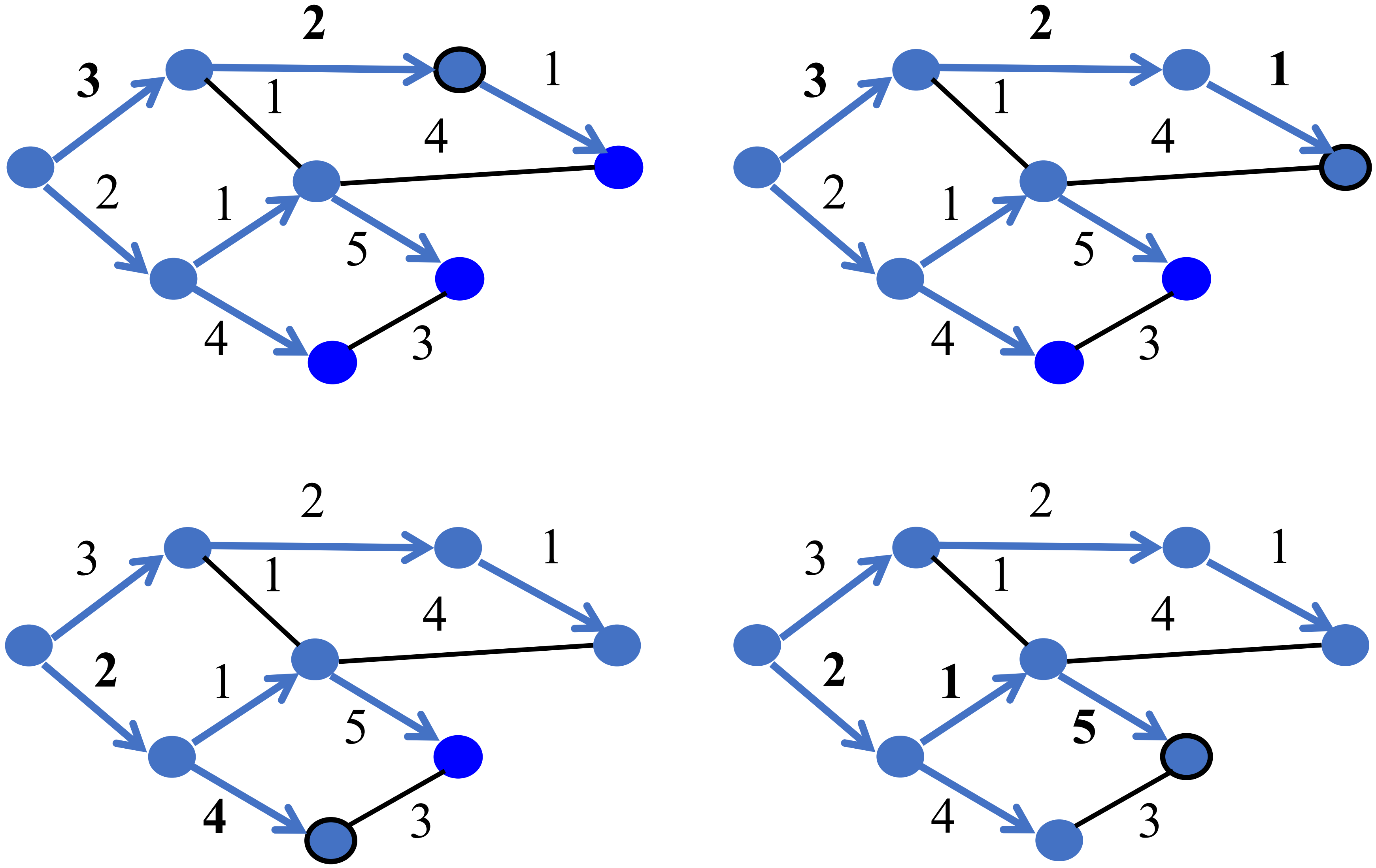






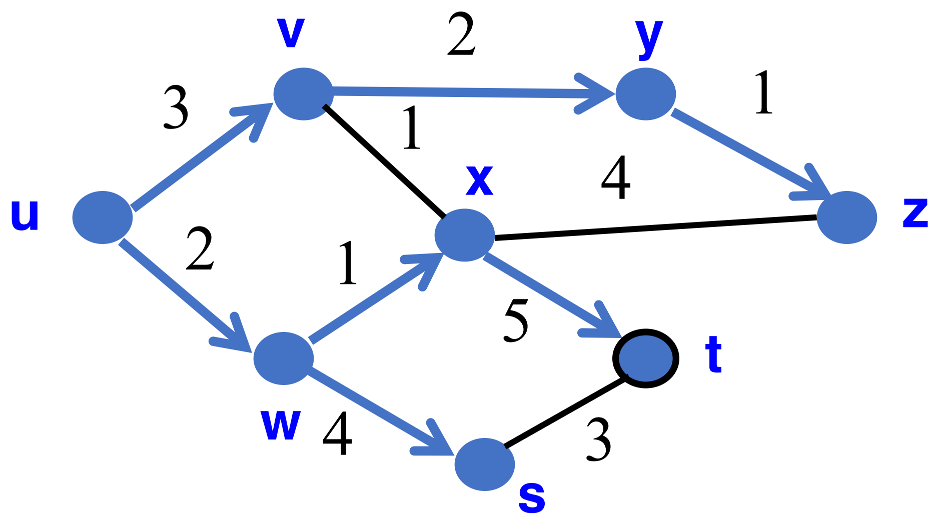


Link-state routing example (cont.)



Link-state: Shortest-path tree

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

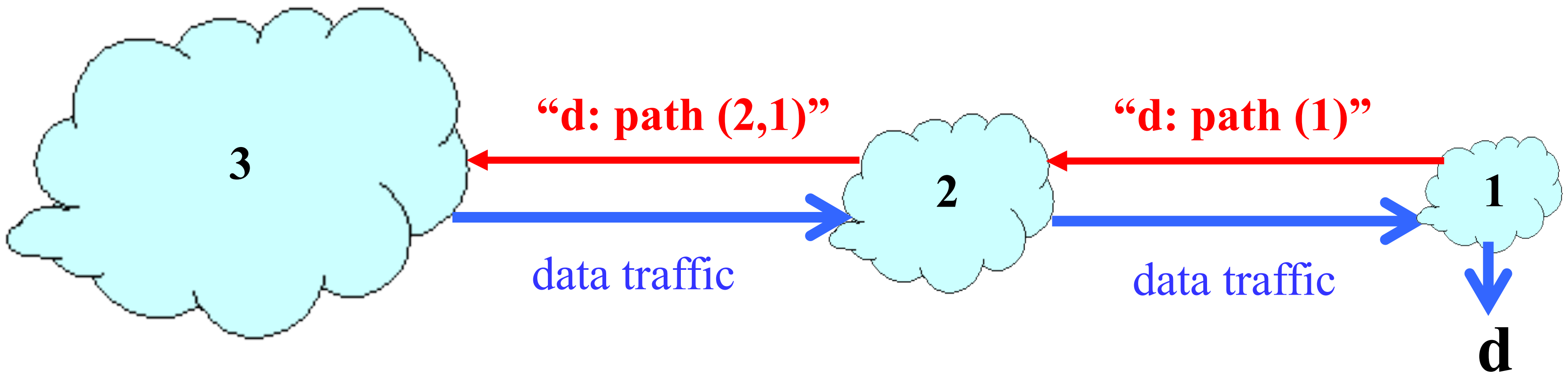


	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

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

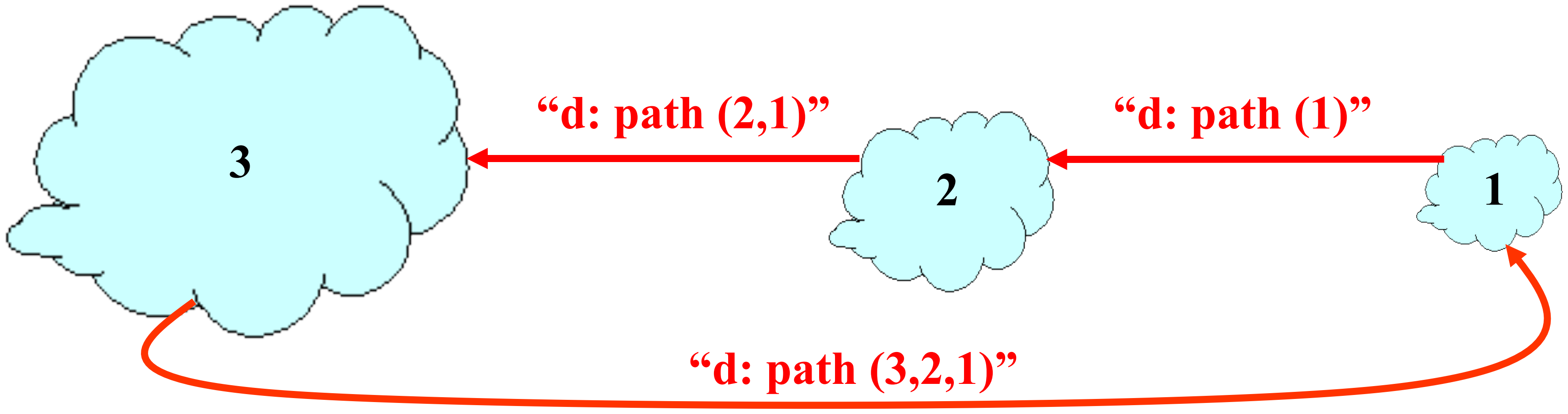
Path-vector routing (BGP)

- Key idea: advertise the entire path
- Distance vector: send *distance metric* per dest d
- Path vector: send the *entire path* for each dest d



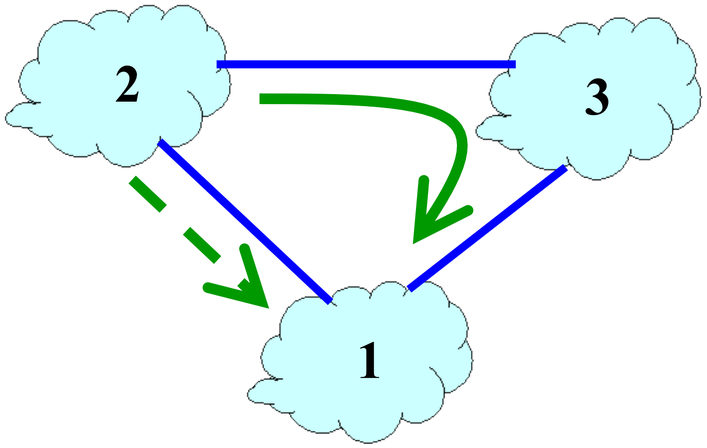
Path-vector: Fast loop detection

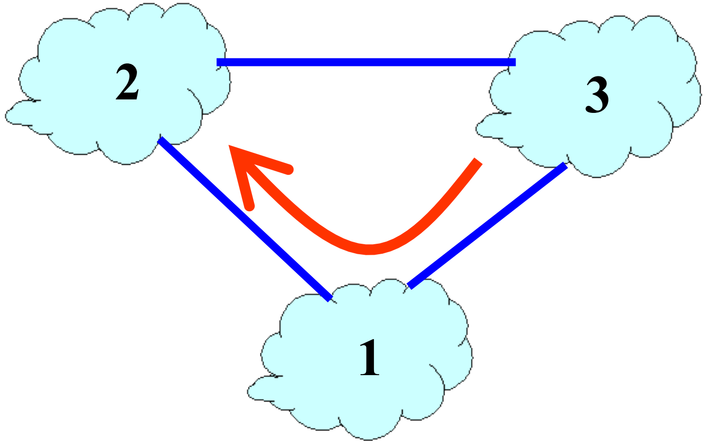
- Node can easily detect a loop
  - Look for its own node identifier in the path
  - E.g., node 1 sees itself in the path "3, 2, 1"
- Node can simply discard paths with loops



Path-vector: Flexible policies

- Each node can apply local policies
  - Path selection: Which path to use?
  - Path export: Which paths to advertise?
- Examples
  - Node 2 may prefer the path “2, 3, 1” over “2, 1”
  - Node 1 may not let node 3 hear the path “1, 2”



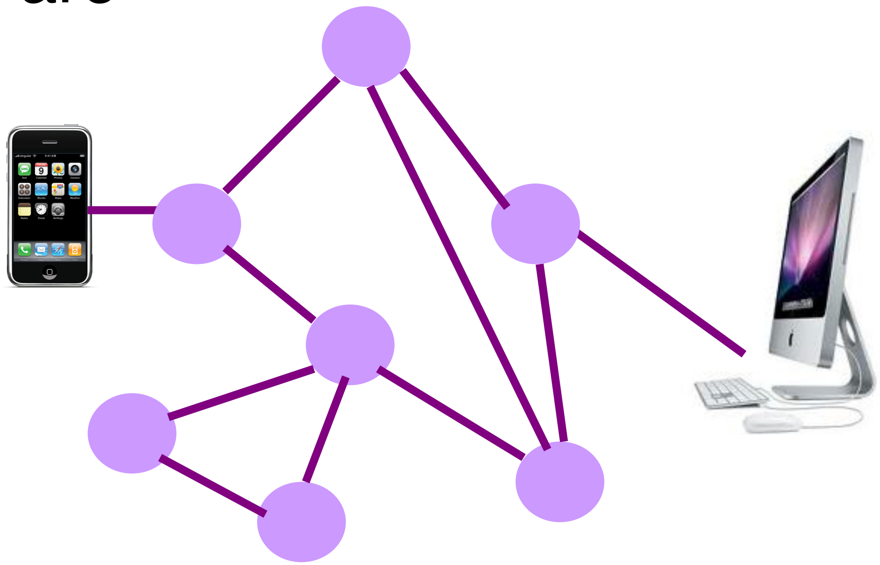


Why are these algorithms distributed?

Learning the location of the endpoints

Finding the endpoints

- Computing the forwarding table
  - Still must figure out where the endpoints are
- How to find the endpoints?
  - Learning/flooding (Ethernet)
  - Injecting into the routing protocol
  - Dissemination using a different protocol
  - Central directory service
- Ways to curb scaling challenges
  - E.g., spanning tree per VLAN for endpoint flooding

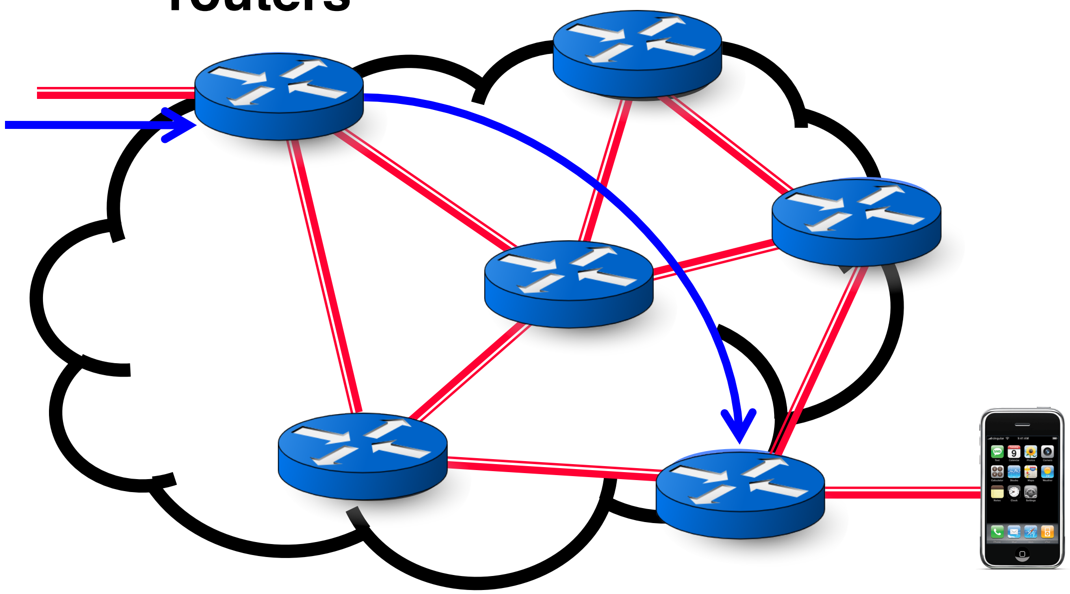


Ex: Disseminate with another protocol

- One router learns the route
- … and shares the information with other routers

disseminate route to other routers

learn a route to d (e.g., via BGP)



Internal BGP (iBGP) used in backbone networks

Conclusion

- Routing is a distributed computation
  - With challenges in scalability and handling dynamics

- Different solutions for different environments
  - Ethernet LAN: spanning tree, MAC learning, flooding
  - Enterprise: link-state routing, injecting subnet addresses
  - Backbone: link-state routing inside, path-vector routing with neighboring domains, and iBGP dissemination
  - Data centers: many different solutions, still in flux
    - An active research area…

Clean-slate 4D architecture

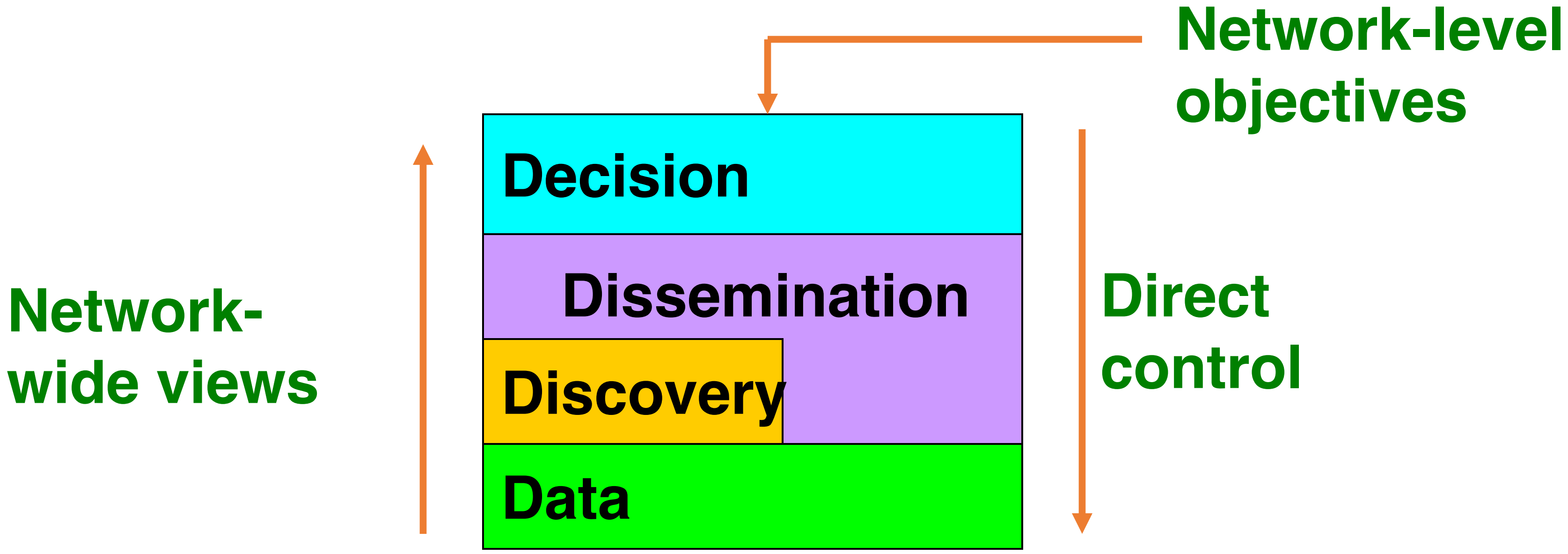
ACM Computer Communications Review (CCR) 2005

Greenberg et al.

Three goals of 4D architecture

- Network-level objectives
  - Configure the network globally; not each router
  - E.g., minimize the maximum link utilization
  - E.g., connectivity under all layer-two failures
- Network-wide views
  - Complete visibility to drive decision-making
  - Traffic matrix, network topology, equipment
- Direct control
  - Direct, *complete* control over data-plane configuration
  - Packet forwarding, filtering, marking, buffering…

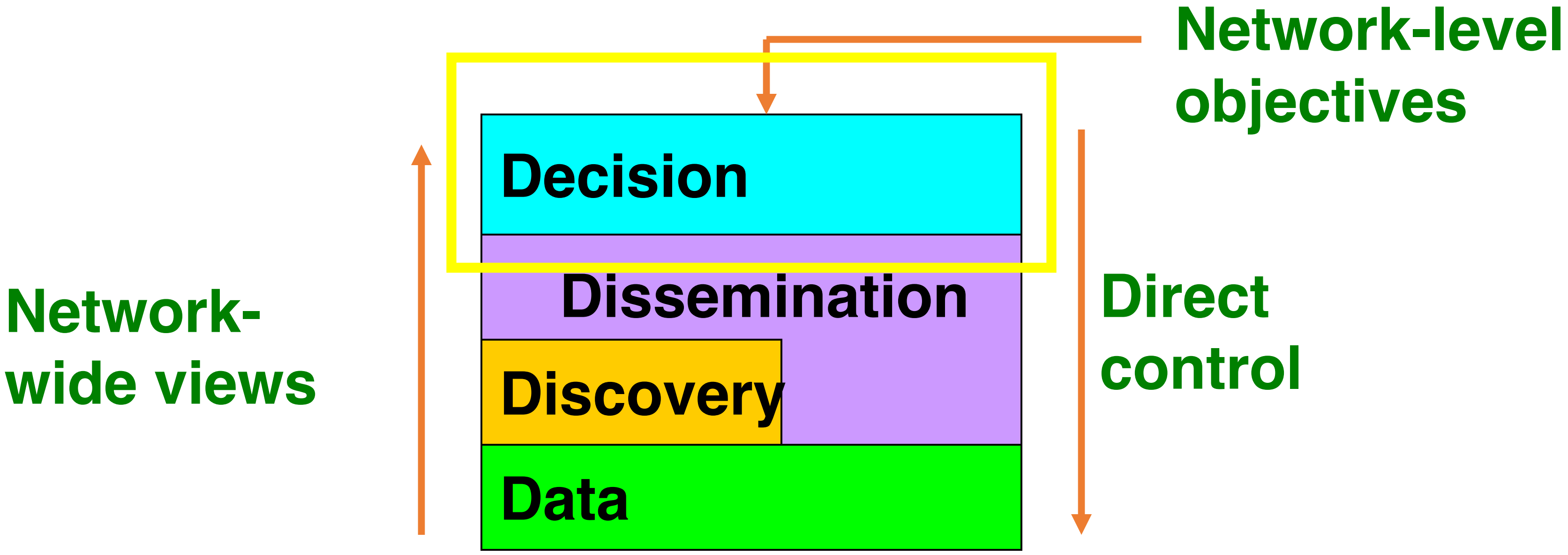
4D: The four planes



- Decision: all management and control logic
- Dissemination: communication to/from the routers
- Discovery: topology and traffic monitoring
- Data: packet handling

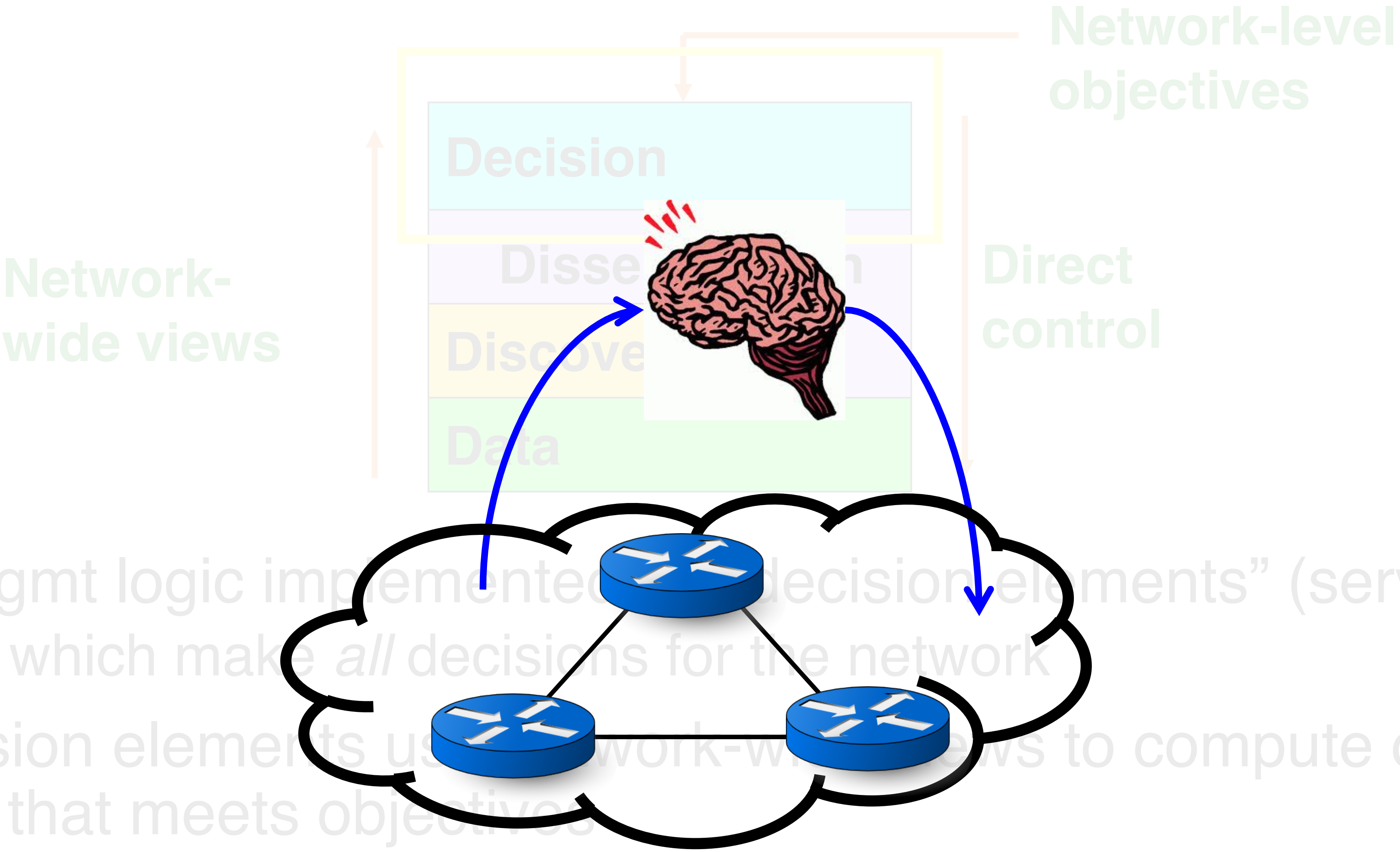
routers

Decision plane



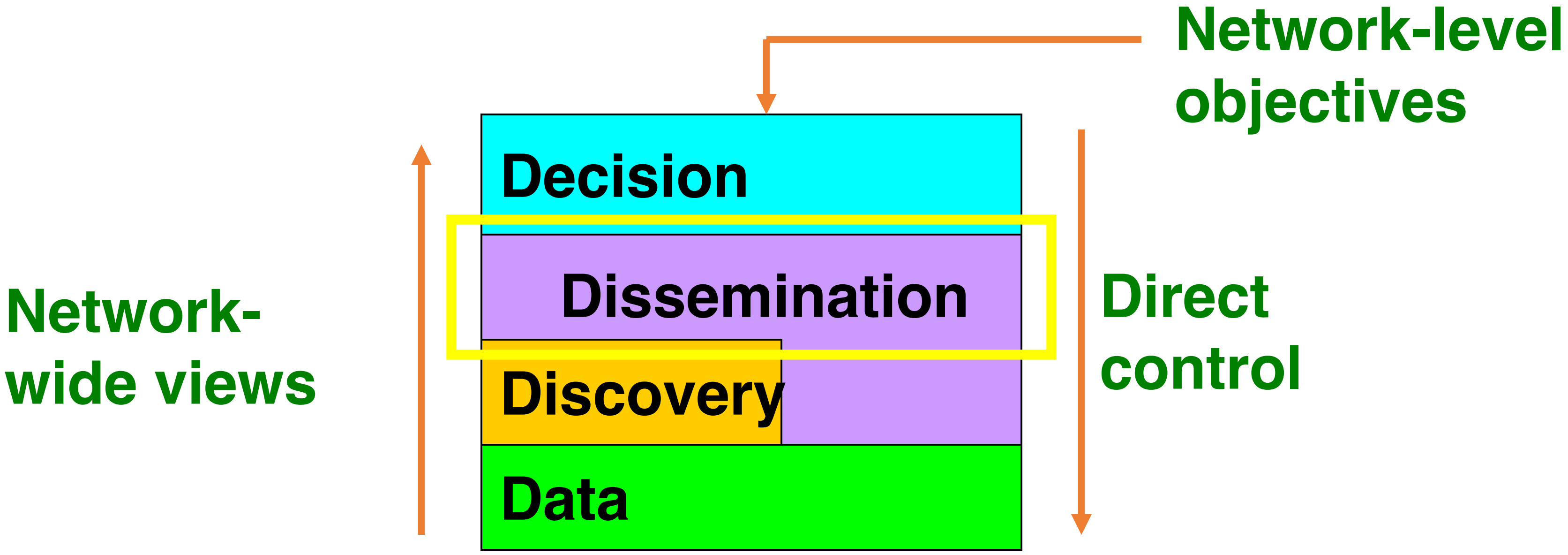
- All mgmt logic implemented on “decision elements” (servers)
  - … which make *all* decisions for the network
- Decision elements use network-wide views to compute data plane state that meets objectives
  - … then *directly* writes this state to routers!

Decision plane



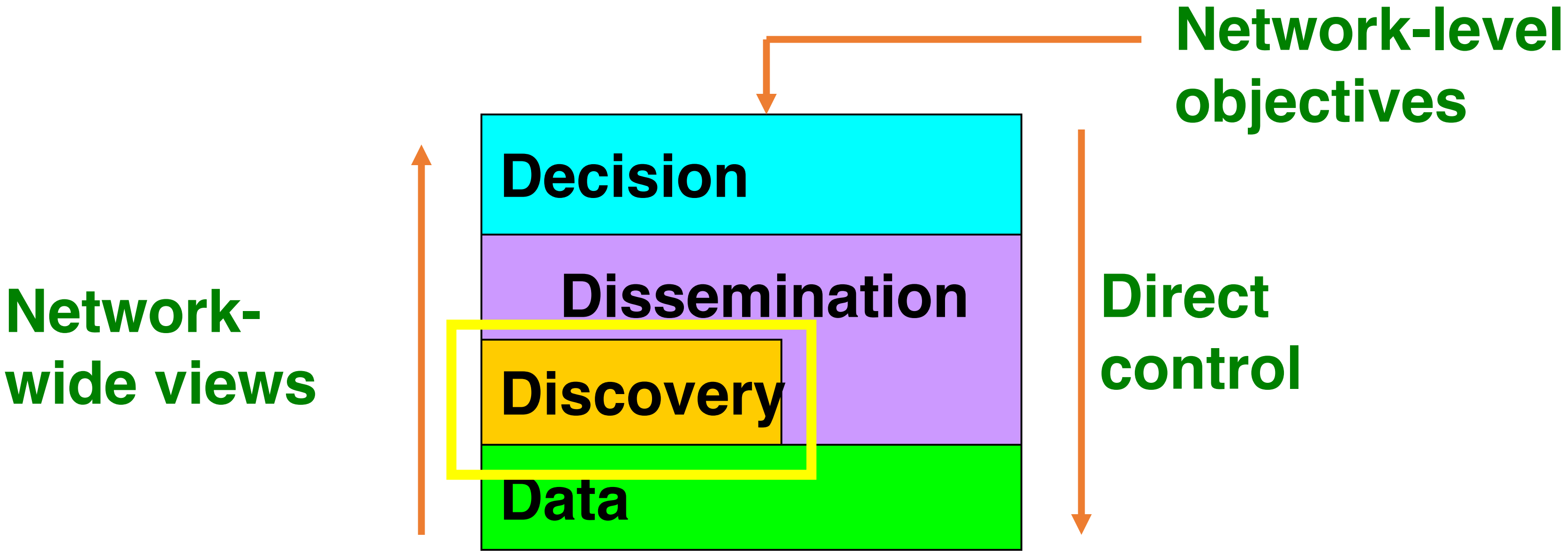
- All mgmt logic implemented on “decision elements” (servers)
  - … which make *all* decisions for the network
- Decision elements use network-wide views to compute data plane state that meets objectives
  - … then *directly* writes this state to routers!

Dissemination plane



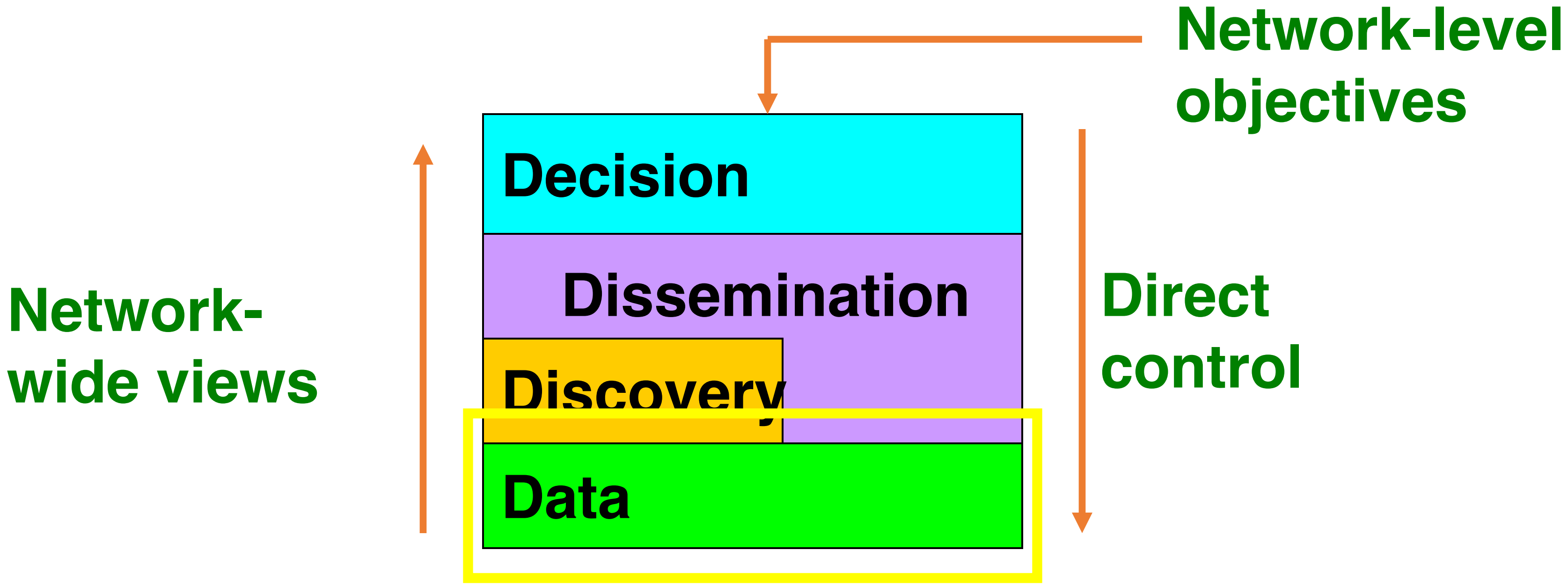
- Provides a robust communication channel to each router
- May run over same links as user data
  - However, logically separate and independently controlled

Discovery Plane



- Each router discovers its own resources and its local environment
- Each router propagates information (e.g., topology, traffic) to the decision elements via dissemination plane

Data Plane



- Spatially distributed routers/switches
- Forward, drop, buffer, shape, mark, rewrite, …
- Can deploy with new or existing technology

Practical technical challenges

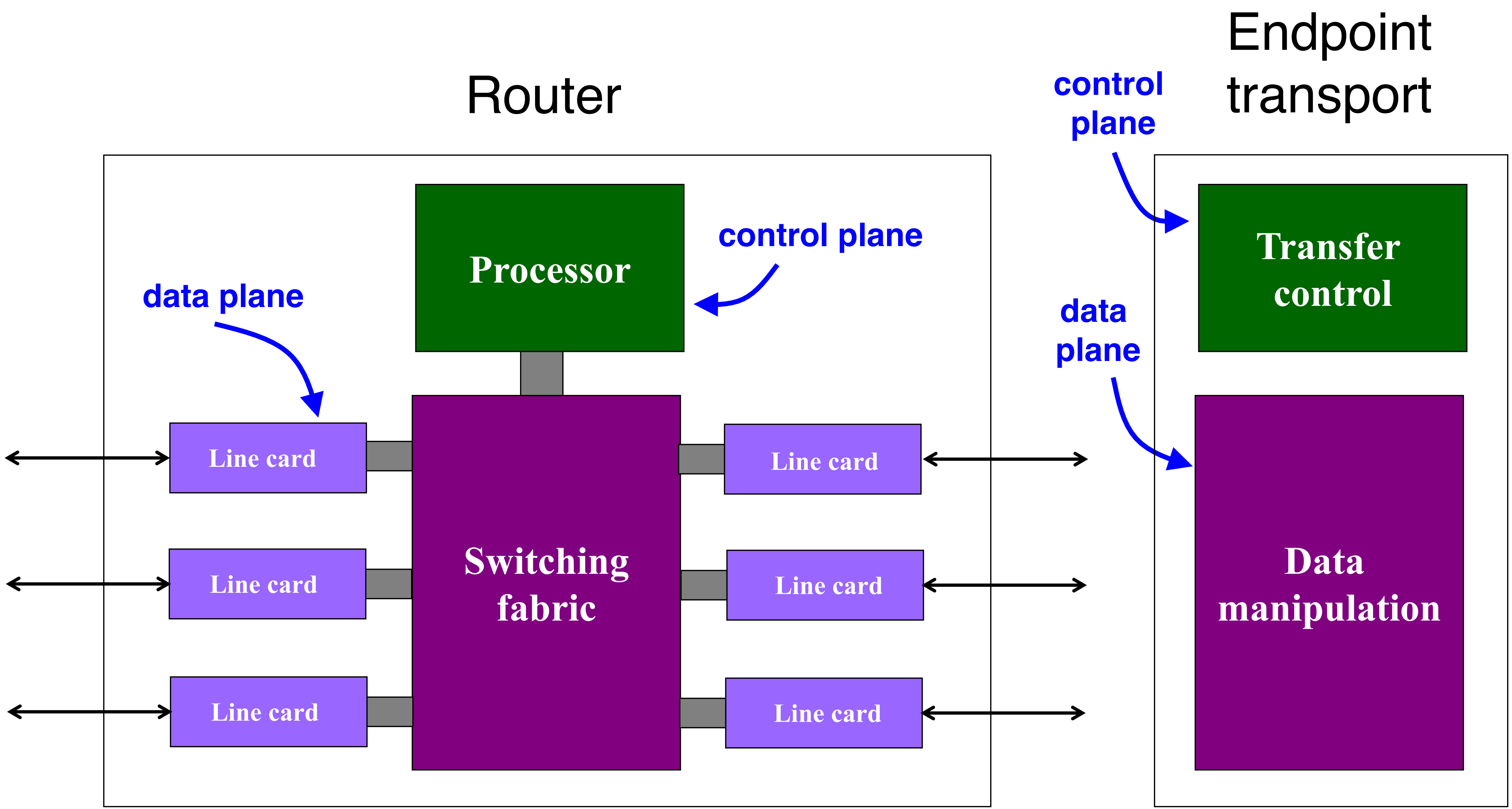
- Scalability: Decision elements responsible for many routers
- Response time: Delays between decision elements and routers
- Reliability: Surviving failures of decision elements, routers, and the dissemination plane
- Consistency: Ensuring multiple decision elements behave consistently
- Security: Network vulnerable to attacks on decision elements
- Interoperability: Legacy routers and neighboring domains

Networking meets Distributed Systems

Architectural considerations for new protocols

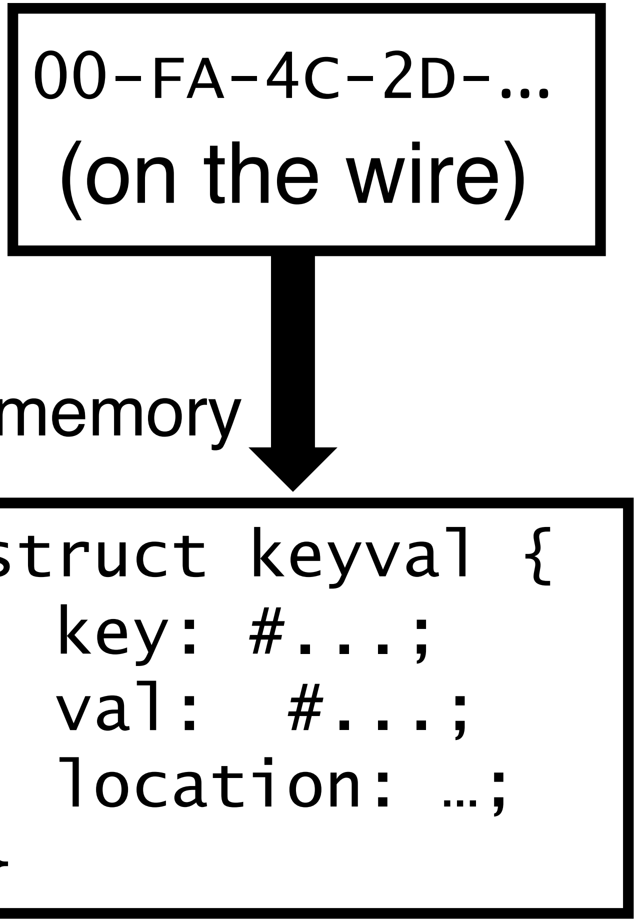
ACM SIGCOMM conference 1990

David Clark and David Tennenhouse


Router
Endpoint transport
control plane
data plane
Processor
control plane
Transfer control
data plane
Line card
Switching fabric
Line card
Line card
Line card
Line card
Line card
Data manipulation

What limits performance of transport?

- Transfer control: do infrequently over connection lifetime
    - Flow and congestion control
    - Loss and reordering, app multiplexing, …
- Data manipulation: on every byte!
    - Moving data from network adapter to endpoint memory
    - Per-packet error correction
    - Presentation conversion



Two key ideas

- App-level framing (ALF)
  - Transport knows little about what data loss means to the app
  - Transport should deliver data to app as *App Data Units (ADUs)*
    - Natural boundary for error correction and application processing

- Integrated layer processing (ILP)
  - Process ADU to completion, including *all* data manipulation
  - Performance improvement:
    - Reduction in number of memory accesses
    - Remove indirect overheads due to cache misses and TLB flushes

Discussion

- Does ALF support any functionality that TCP/UDP don't?
- Does ALF run contrary to layered design?
  - Pros and cons of these two implementations?
- Is presentation conversion still the dominant bottleneck?
  - Why? Why not? If not, what has changed?
- Why should we always run presentation conversion at its full rate?
- What systems design principles did you derive from this paper?

Until next time…

- MUD: Send me your top 1—3 questions on this lecture
- Readings for next class: Jacobson’s TCP, XCP
- Brainstorm project ideas (teams of 1—3)

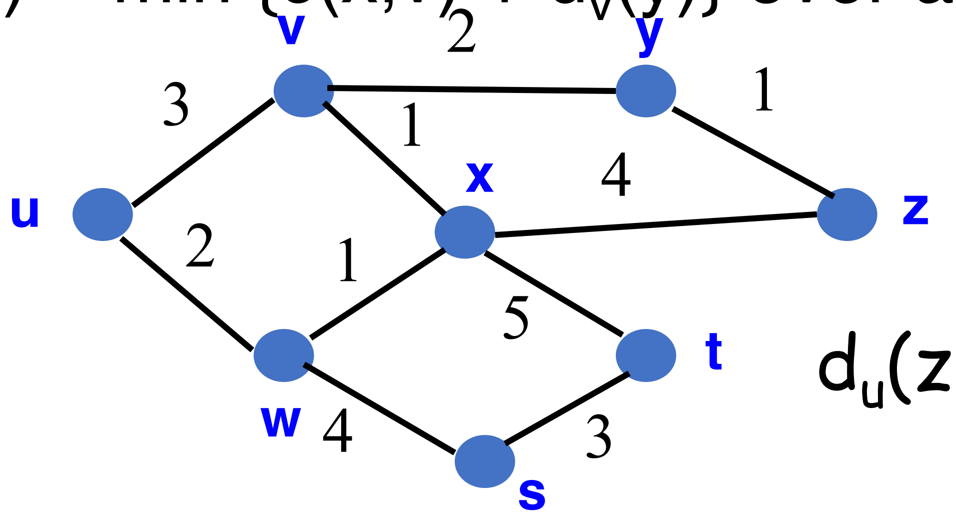
Backup slides

Discussion questions for E2E argument

- When should the network support a function anyway?
  - E.g., link-layer retransmission in wireless networks?
- Whose interests are served by the e2e argument?
- How does a network operator influence the network without violating the e2e argument?
- Does the design of IP and TCP make it *hard* to violate the e2e argument?
  - E.g., middlebox functionality like NATs, firewalls, proxies
- Should the e2e argument apply to routing?

Distance Vector: Bellman-Ford Algo

- Define distances at each node x
  - $d_x(y)$ = cost of least-cost path from x to y
- Update distances based on neighbors
  - $d_x(y) = \min \{c(x,v) + d_v(y)\}$ over all neighbors v



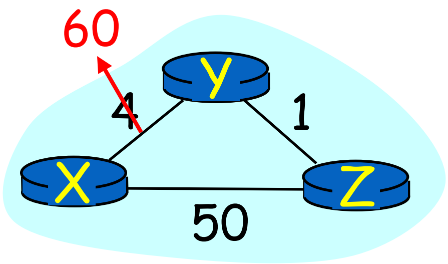
$$d_u(z) = \min\{c(u,v) + d_v(z),\ c(u,w) + d_w(z)\}$$

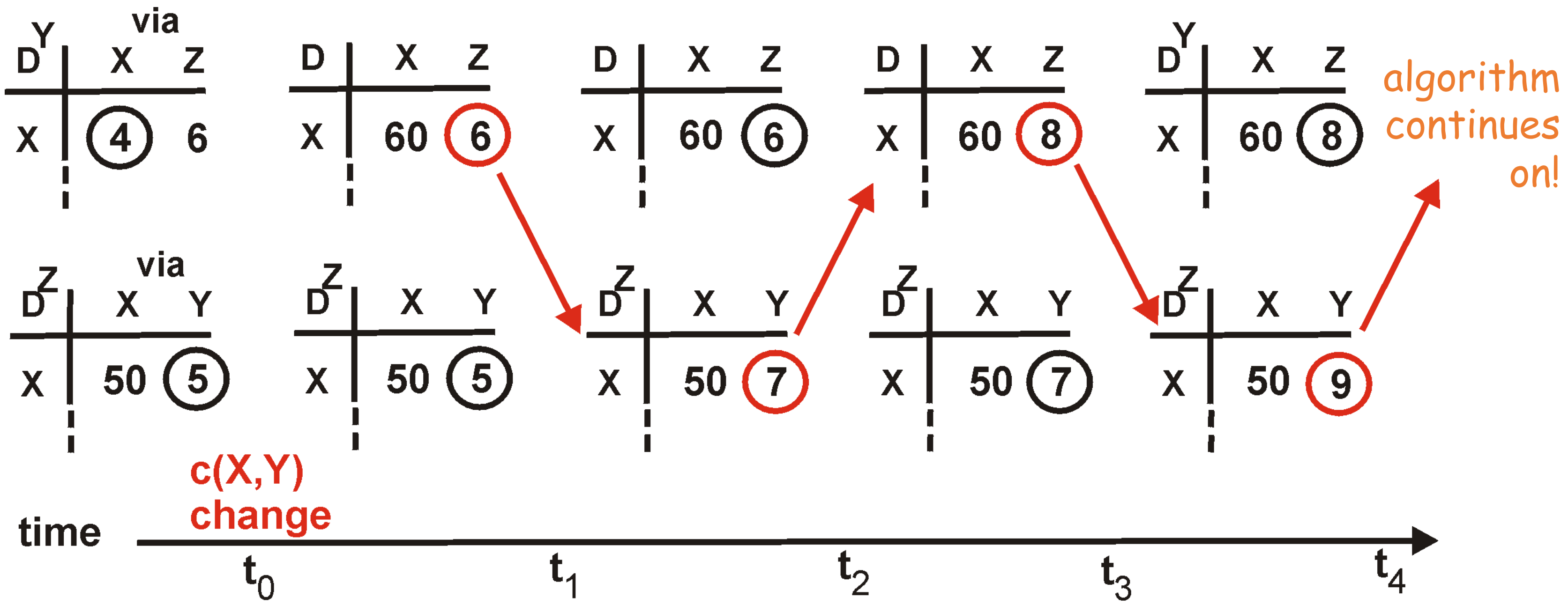
Used in RIP and EIGRP

Distance Vector: Count to Infinity

Link cost changes:

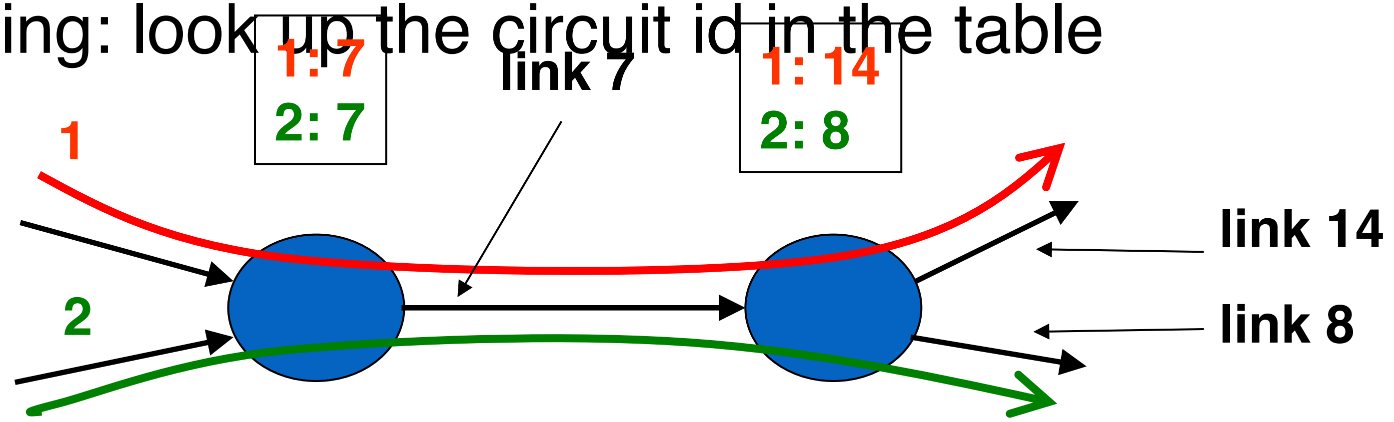
- Good news travels fast
- Bad news travels slow - “count to infinity” problem!





End-to-End Signaling

- Establish end-to-end path in advance
  - Learn the topology (as in link-state routing)
  - End host or router computes and signals a path
- Routers supports virtual circuits
  - Signaling: install entry for each circuit at each hop
  - Forwarding: look up the circuit id in the table



Used in MPLS with RSVP

Source Routing

- Similar to end-to-end signaling
  - But the data packet carries the hops in the path
  - … rather than the routers storing big tables
- End-host control
  - Tell the end host the topology
  - Let the end host select the end-to-end path
- Variations of source routing
  - Strict: specify every hop
  - Loose: specify intermediate points

Used in IP source routing (but almost *always* disabled)