Link-State Routing

Reading: Sections 4.2 and 4.3.4

COS 461: Computer Networks
Spring 2011

Mike Freedman
http://www.cs.princeton.edu/courses/archive/spring11/cos461/

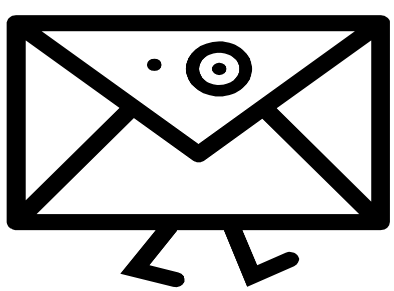
Goals of Today's Lecture

- Inside a router
  - Control plane: routing protocols
  - Data plane: packet forwarding
- Path selection
  - Minimum-hop and shortest-path routing
  - Dijkstra's algorithm
- Topology change
  - Using beacons to detect topology changes
  - Propagating topology information
- Routing protocol: Open Shortest Path First (OSPF)

What is Routing?

- A famous quotation from RFC 791

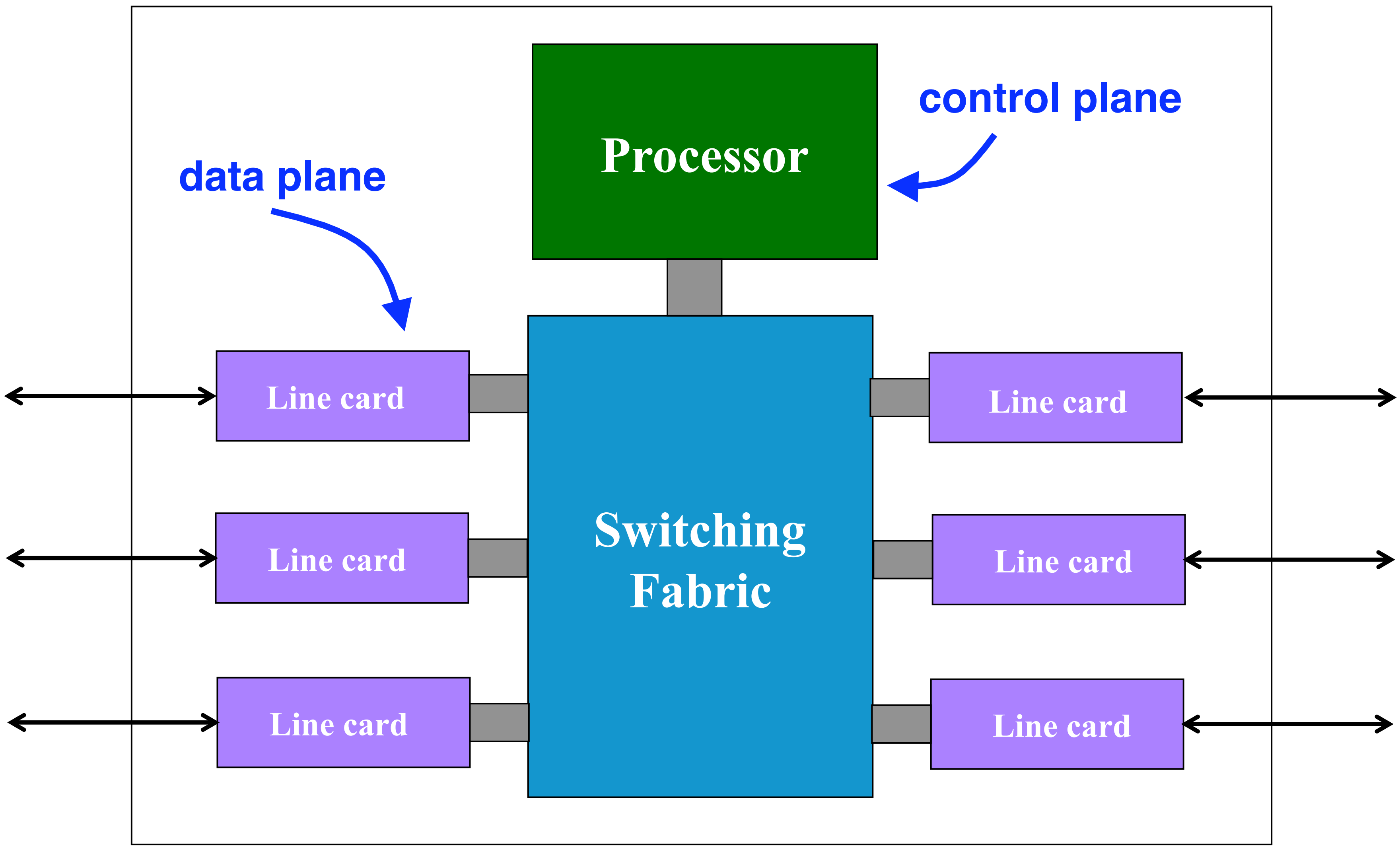
"A *name* indicates what we seek.
An *address* indicates where it is.
A *route* indicates how we get there."
-- Jon Postel



Routing vs. Forwarding

- Routing: control plane
  - Computing paths the packets will follow
  - Routers talking amongst themselves
  - Individual router *creating* a forwarding table
- Forwarding: data plane
  - Directing a data packet to an outgoing link
  - Individual router *using* a forwarding table

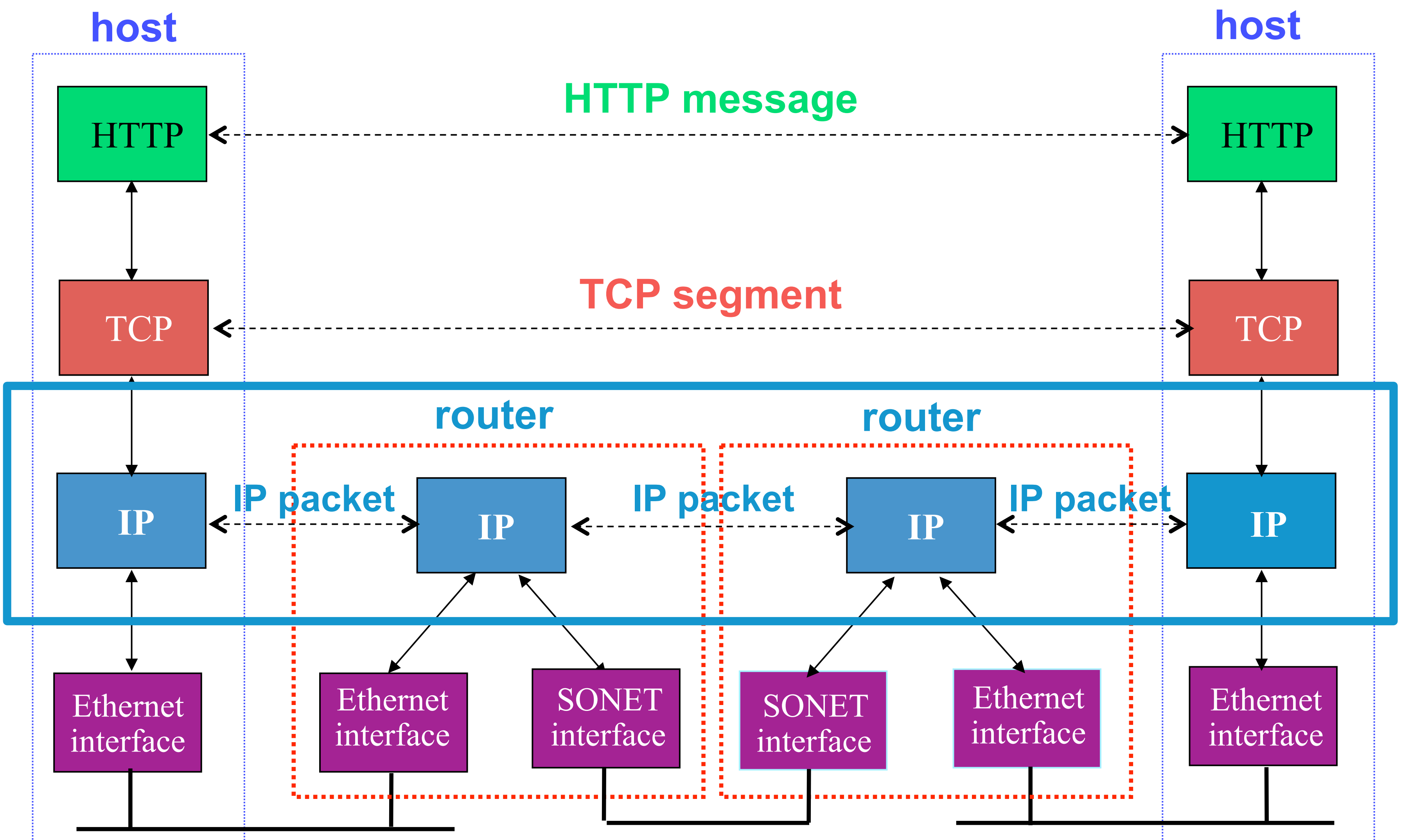
Data and Control Planes



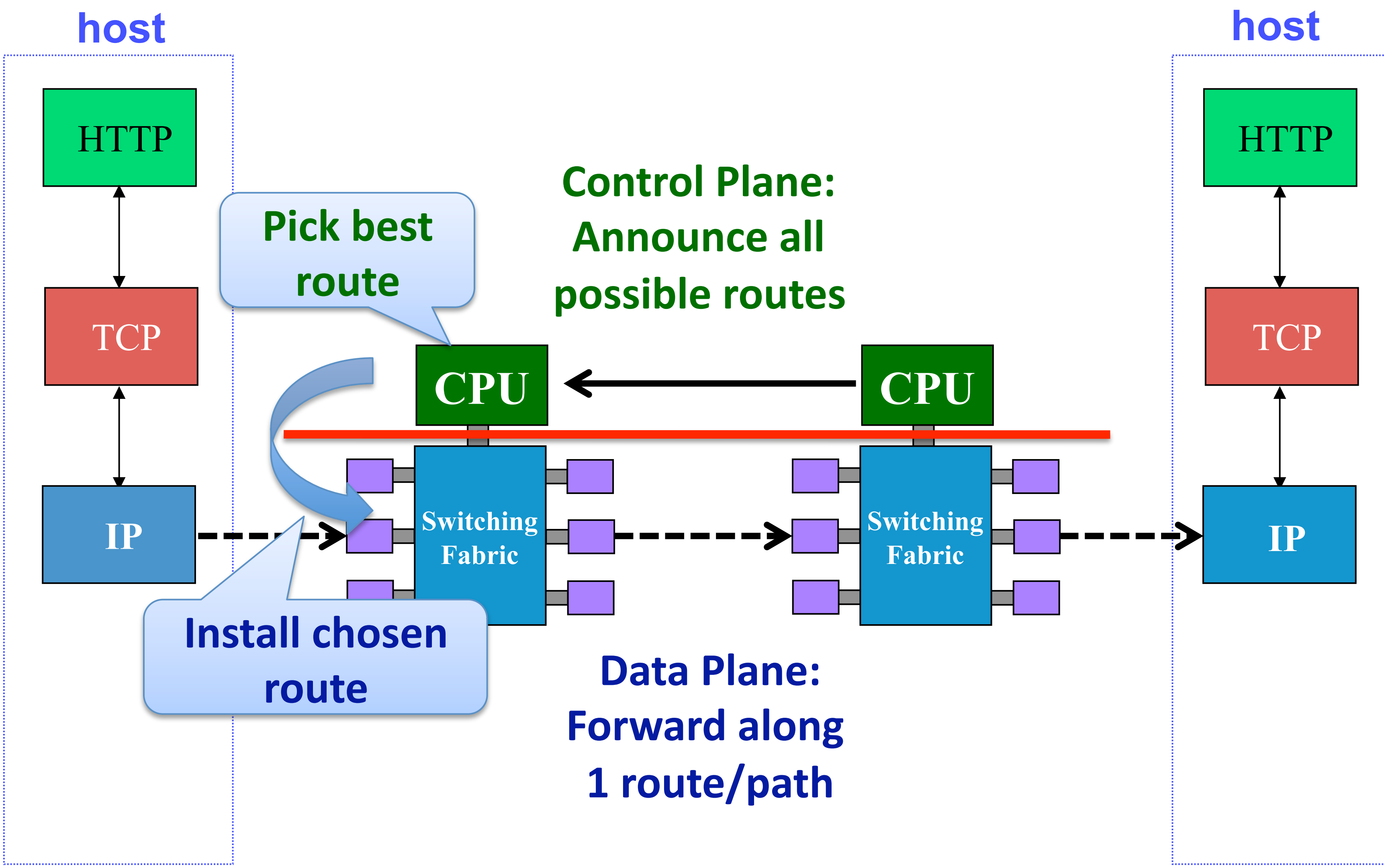
Where do Forwarding Tables Come From?

- Routers have forwarding tables
  - Map IP prefix to outgoing link(s)
- Entries can be statically configured
  - E.g., "map 12.34.158.0/24 to Serial0/0.1"
- But, this doesn't adapt
  - To failures
  - To new equipment
  - To the need to balance load
- That is where routing protocols come in

Recall the Internet layering model

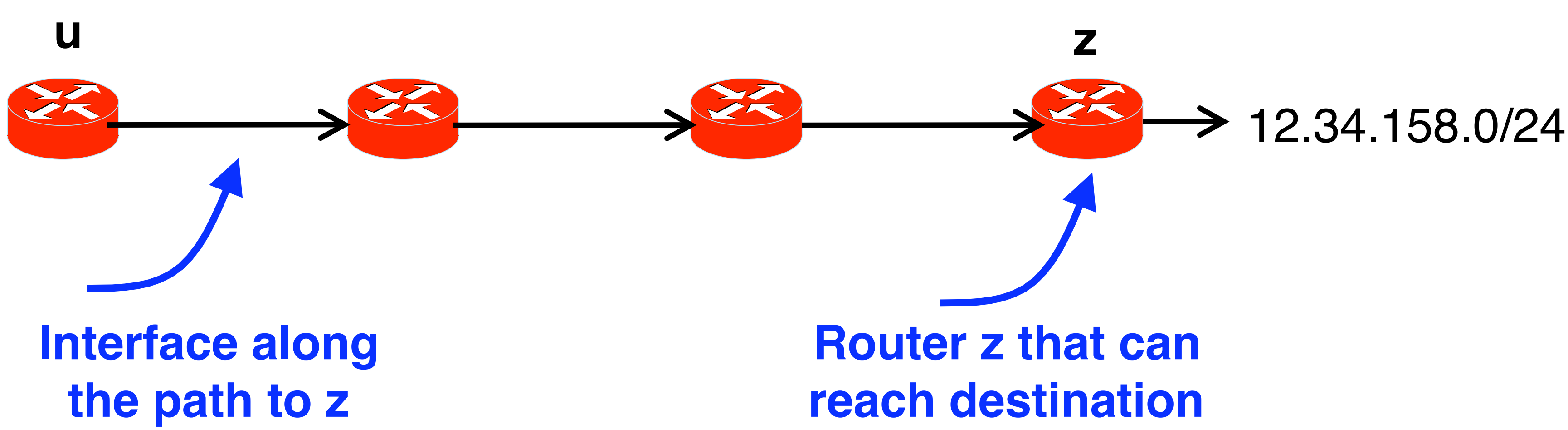


Recall the Internet layering model



Computing Paths Between Routers

- Routers need to know two things
  - Which router to use to reach a destination prefix
  - Which outgoing interface to use to reach that router



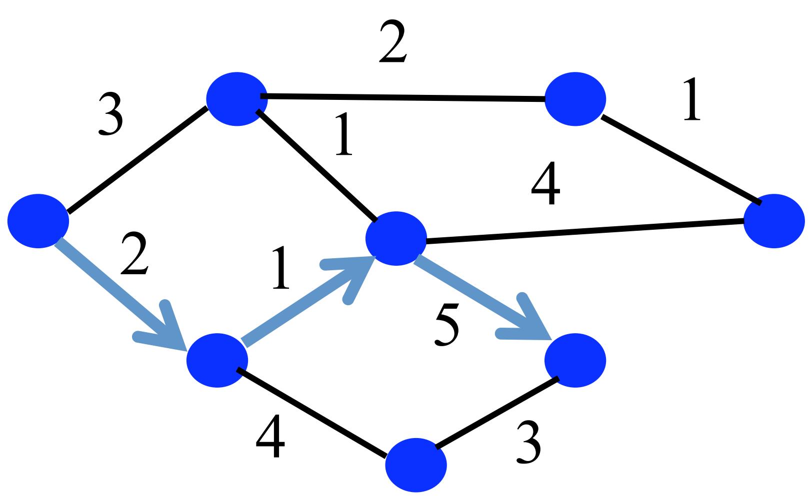
- Today's class: how routers reach each other
  - How *u* knows how to forward packets toward *z*

Computing the Shortest Paths

Assuming you already know
the topology

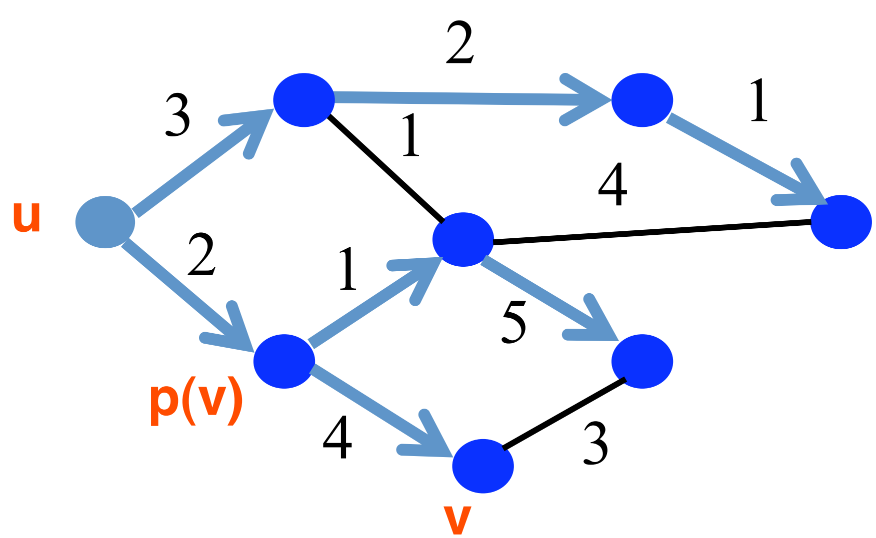
Shortest-Path Routing

- Path-selection model
  - Destination-based
  - Load-insensitive (e.g., static link weights)
  - Minimum hop count or sum of link weights



Shortest-Path Problem

- Given: network topology with link costs
  - **c(x,y)**: link cost from node x to node y
  - Infinity if x and y are not direct neighbors
- Compute: least-cost paths to all nodes
  - From a given source u to all other nodes
  - **p(v)**: predecessor node along path from source to v



Dijkstra's Shortest-Path Algorithm

- Iterative algorithm
  - After k iterations, know least-cost path to k nodes
- **S**: nodes whose least-cost path definitively known
  - Initially, **S = {u}** where u is the source node
  - Add one node to S in each iteration
- **D(v)**: current cost of path from source to node v
  - Initially, **D(v) = c(u,v)** for all nodes v adjacent to u
  - … and **D(v) = ∞** for all other nodes v
  - Continually update D(v) as shorter paths are learned

Dijsktra's Algorithm

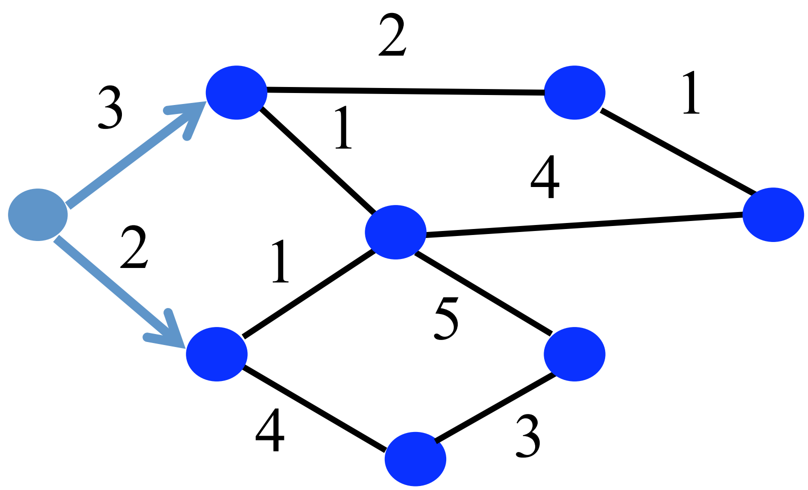
```
Initialization:
  S = {u}
  for all nodes v
    if (v is adjacent to u)
        D(v) = c(u,v)
    else D(v) = ∞

Loop: Do
   find w not in S with the smallest D(w)
   add w to S
   update D(v) for all v adjacent to w and not in S:
       D(v) = min{D(v), D(w) + c(w,v)}
until all nodes in S
```

S: Least cost path known

D(v): Known shortest cost from source to v

C(w,v): Known cost from w to v

Dijkstra's Algorithm Example



Dijkstra's Algorithm Example

Loop: Do

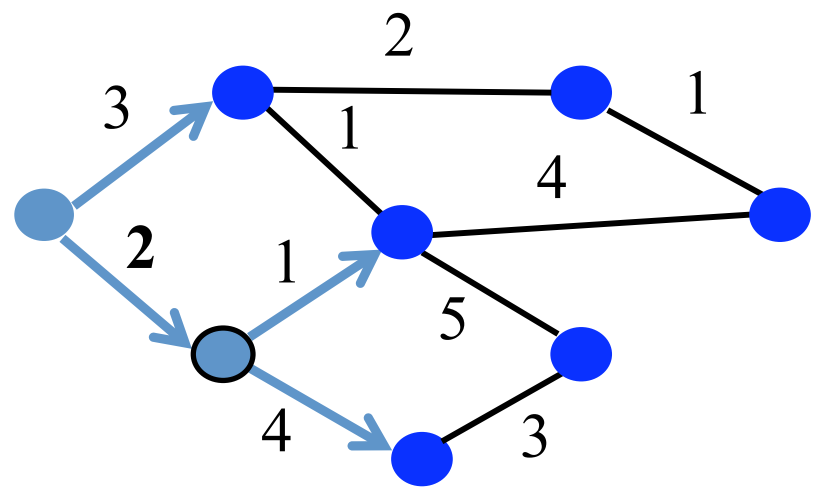
find w not in S with the smallest D(w)

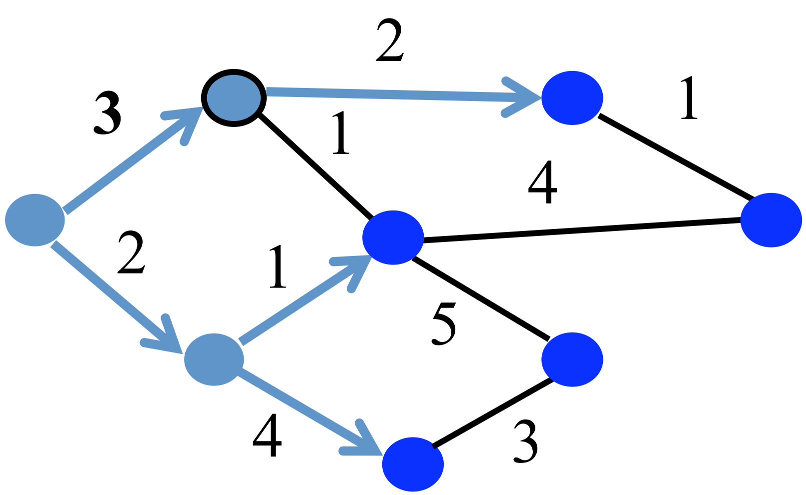
add w to S

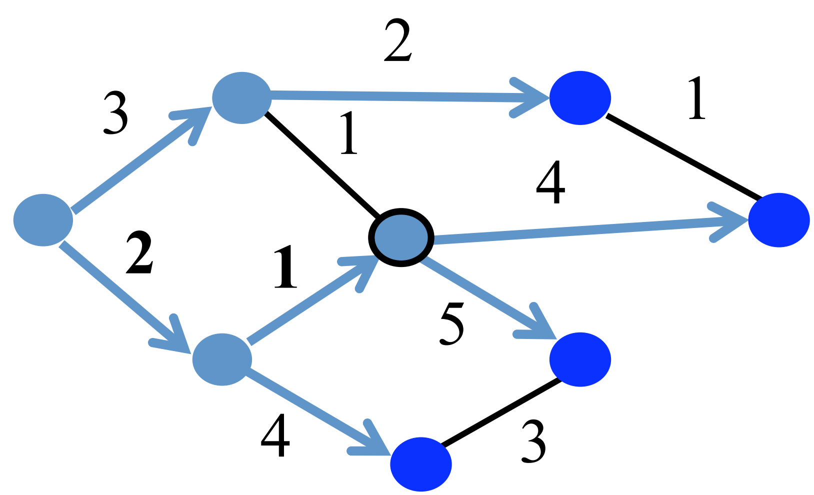
forall v adj to w && not in S:

D(v) = min{ D(v), D(w) + c(w,v) }

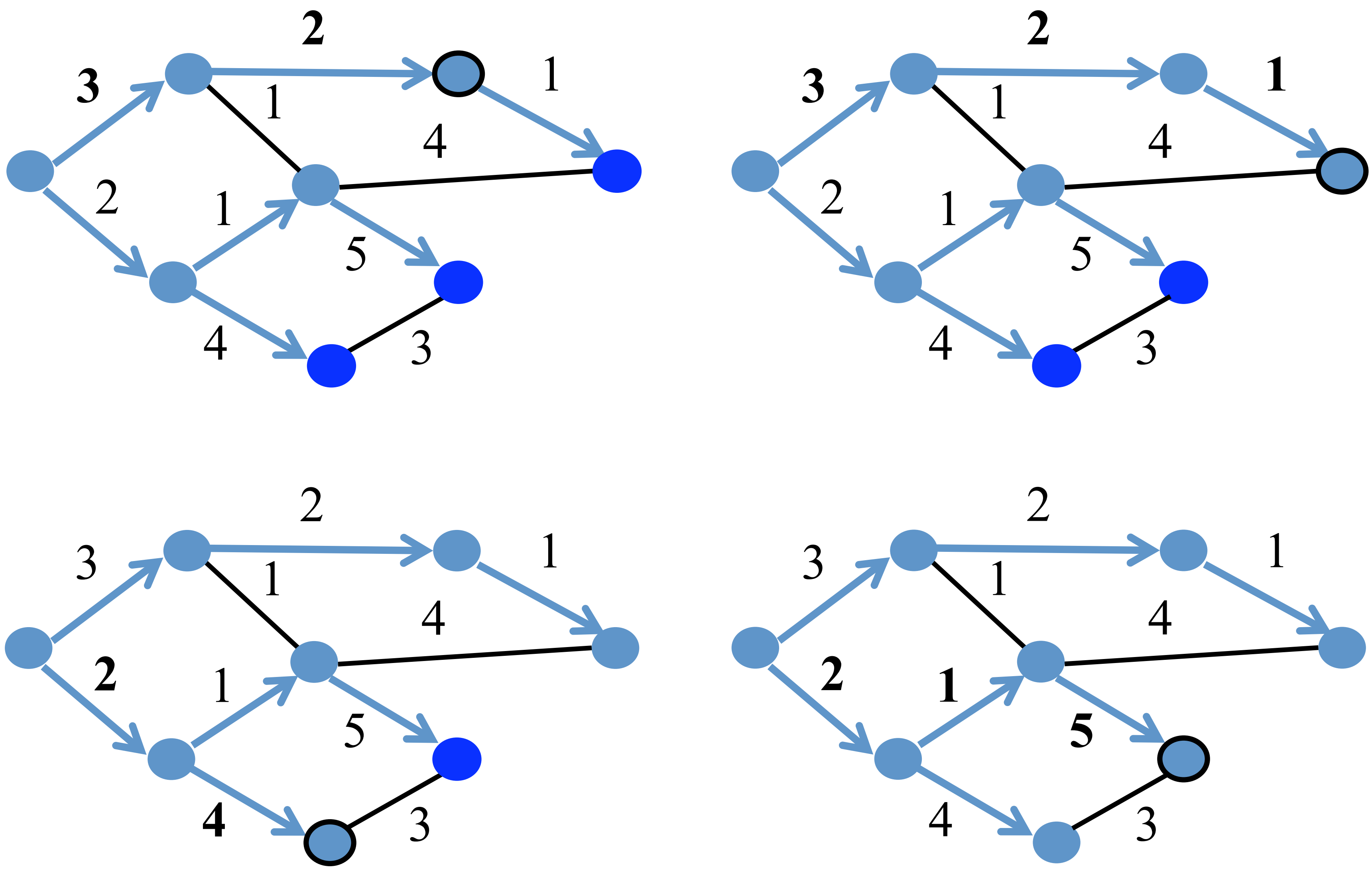
until all nodes in S





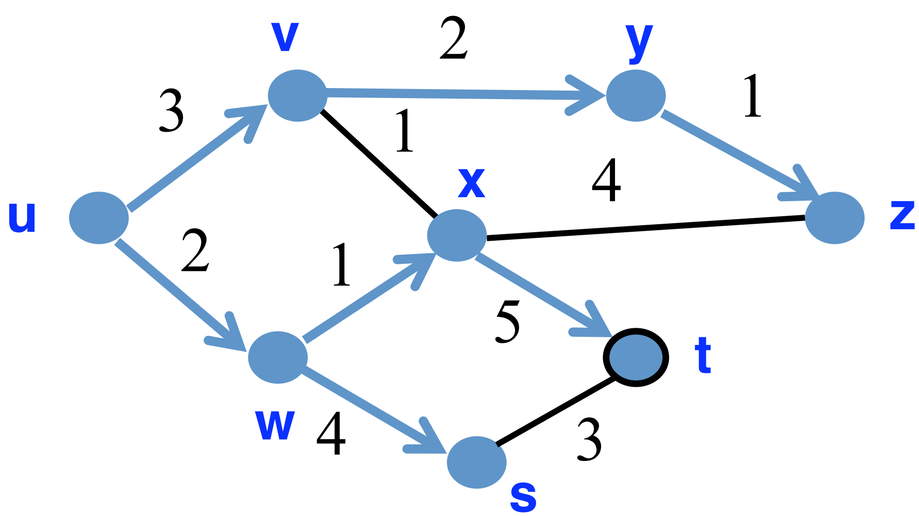


Dijkstra's Algorithm Example



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)

Learning the Topology

By the routers talk amongst themselves

Link-State Routing

- Each router keeps track of its incident links
  - Whether the link is up or down
  - The cost on the link
- Each router broadcasts the link state
  - To give every router a complete view of the graph
- Each router runs Dijkstra's algorithm
  - To compute the shortest paths
  - … and construct the forwarding table
- Example protocols
  - Open Shortest Path First (OSPF)
  - Intermediate System – Intermediate System (IS-IS)

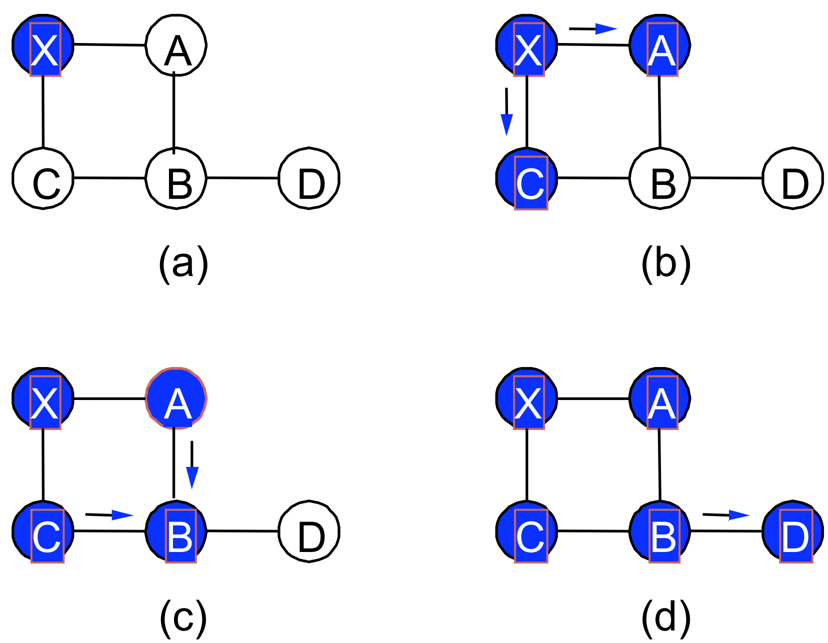
Detecting Topology Changes

- Beaconing
  - Periodic "hello" messages in both directions
  - Detect a failure after a few missed "hellos"

- Performance trade-offs
  - Detection speed
  - Overhead on link bandwidth and CPU
  - Likelihood of false detection

Broadcasting the Link State

- Flooding
  - Node sends link-state information out its links
  - And then the next node sends out all of its links
  - … except the one(s) where the information arrived



Broadcasting the Link State

- Reliable flooding
  - Ensure all nodes receive link-state information
  - … and that they use the latest version
- Challenges
  - Packet loss
  - Out-of-order arrival
- Solutions
  - Acknowledgments and retransmissions
  - Sequence numbers
  - Time-to-live for each packet

When to Initiate Flooding

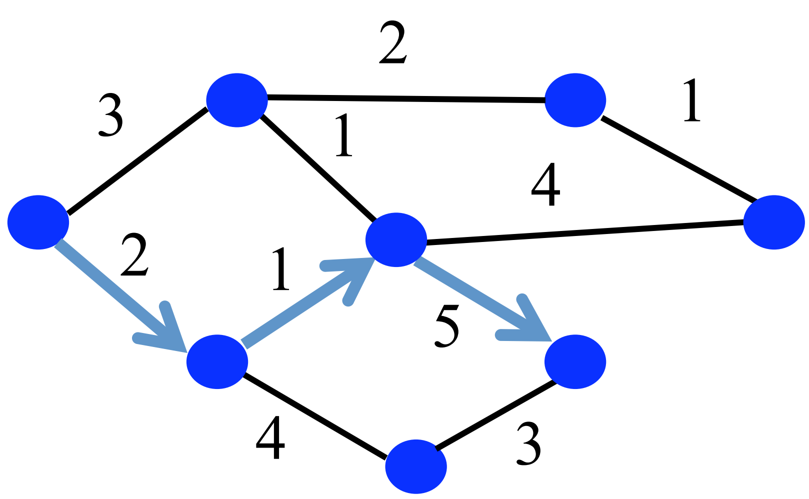
- Topology change
  - Link or node failure
  - Link or node recovery
- Configuration change
  - Link cost change
- Periodically
  - Refresh the link-state information
  - Typically (say) 30 minutes
  - Corrects for possible corruption of the data

When the Routers Disagree

(during transient periods)

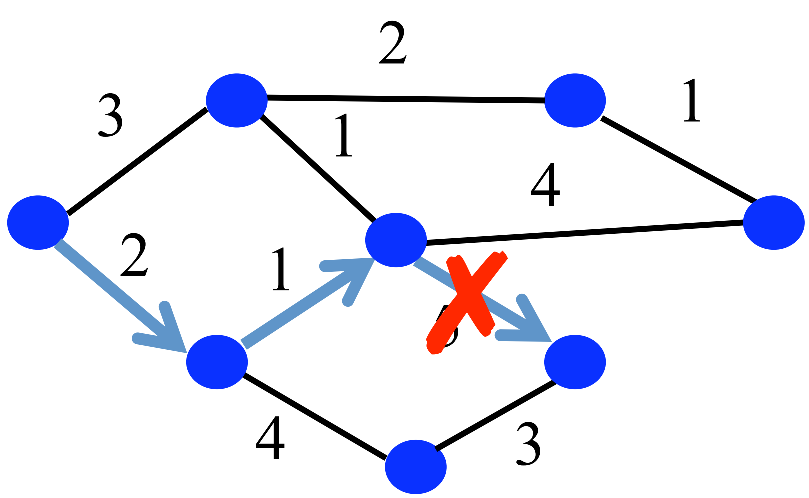
Convergence

- Getting consistent routing information to all nodes
  - E.g., all nodes having the same link-state database
- Consistent forwarding after convergence
  - All nodes have the same link-state database
  - All nodes forward packets on shortest paths
  - The next router on the path forwards to the next hop



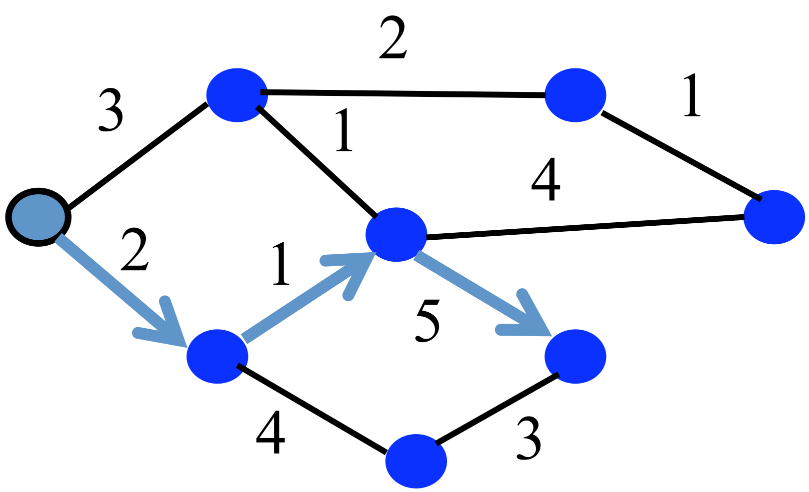
Transient Disruptions

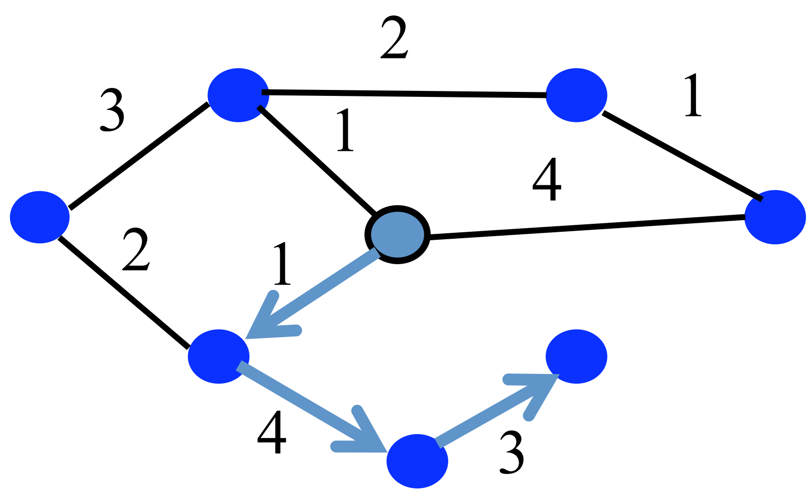
- Detection delay
  - A node does not detect a failed link immediately
  - … and forwards data packets into a “blackhole”
  - Depends on timeout for detecting lost hellos



Transient Disruptions

- Inconsistent link-state database
  - Some routers know about failure before others
  - The shortest paths are no longer consistent
  - Can cause transient forwarding loops





Convergence Delay

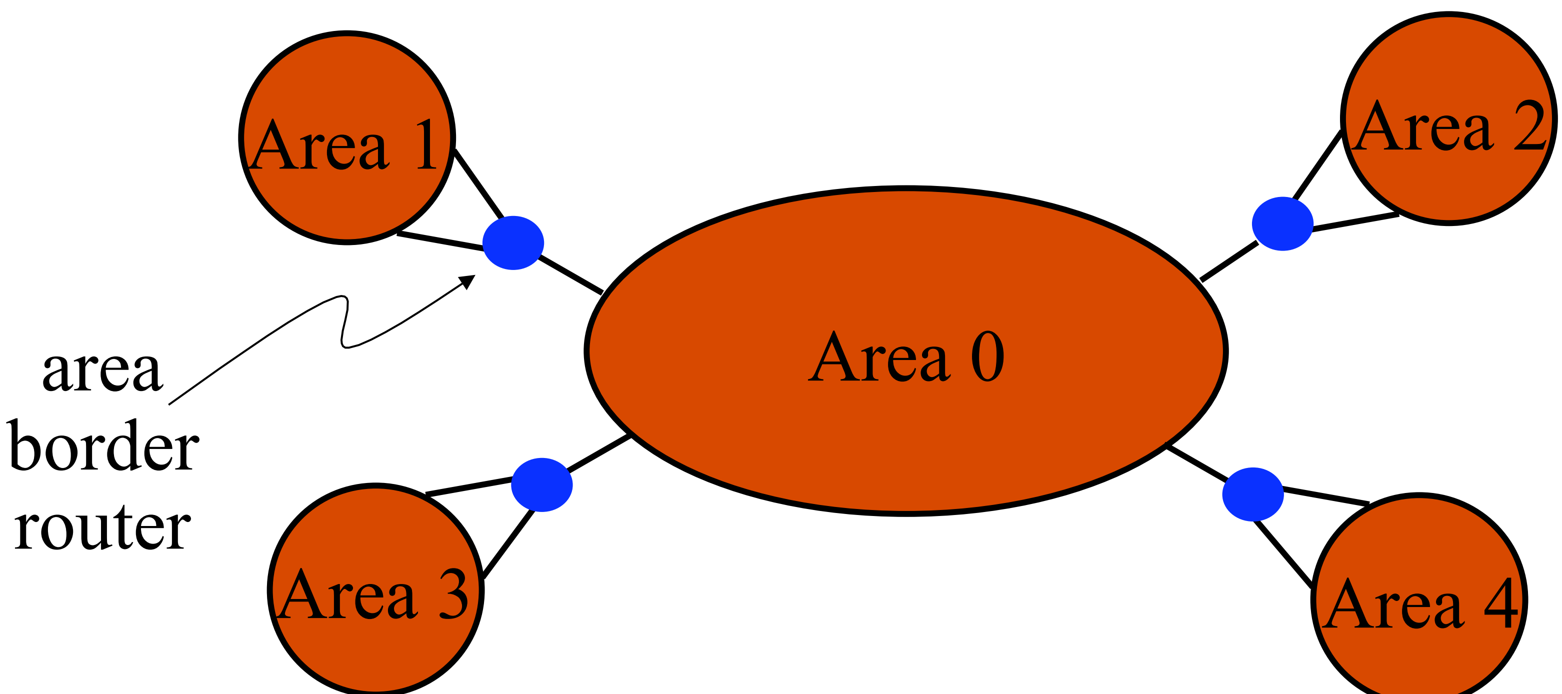
- Sources of convergence delay
  - Detection latency
  - Flooding of link-state information
  - Shortest-path computation
  - Creating the forwarding table
- Performance during convergence period
  - Lost packets due to blackholes and TTL expiry
  - Looping packets consuming resources
  - Out-of-order packets reaching the destination
- Very bad for VoIP, online gaming, and video

Reducing Convergence Delay

- Faster detection
  - Smaller hello timers
  - Link-layer technologies that can detect failures
- Faster flooding
  - Flooding immediately
  - Sending link-state packets with high-priority
- Faster computation
  - Faster processors on the routers
  - Incremental Dijkstra's algorithm
- Faster forwarding-table update
  - Data structures supporting incremental updates

Scaling Link-State Routing

- Overhead of link-state routing
  - Flooding link-state packets throughout the network
  - Running Dijkstra's shortest-path algorithm
- Introducing hierarchy through "areas"



Conclusions

- Routing is a distributed algorithm
  - React to changes in the topology
  - Compute the paths through the network
- Shortest-path link state routing
  - Flood link weights throughout the network
  - Compute shortest paths as a sum of link weights
  - Forward packets on next hop in the shortest path
- Convergence process
  - Changing from one topology to another
  - Transient periods of inconsistency across routers