



Link-State Routing

Reading: Sections 4.2 and 4.3.4

COS 461: Computer Networks
Spring 2010 (MW 3:00-4:20 in COS 105)

Michael Freedman

<http://www.cs.princeton.edu/courses/archive/spring10/cos461/>

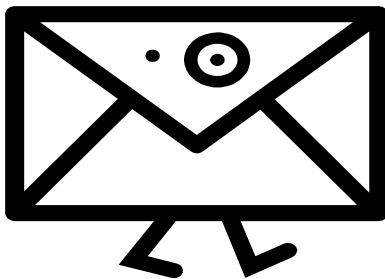
Goals of Today's Lecture

- **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**

What is Routing?

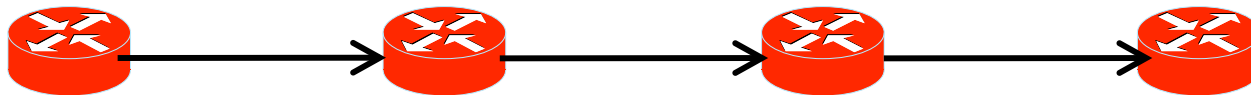
- A famous quotation from RFC 791

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

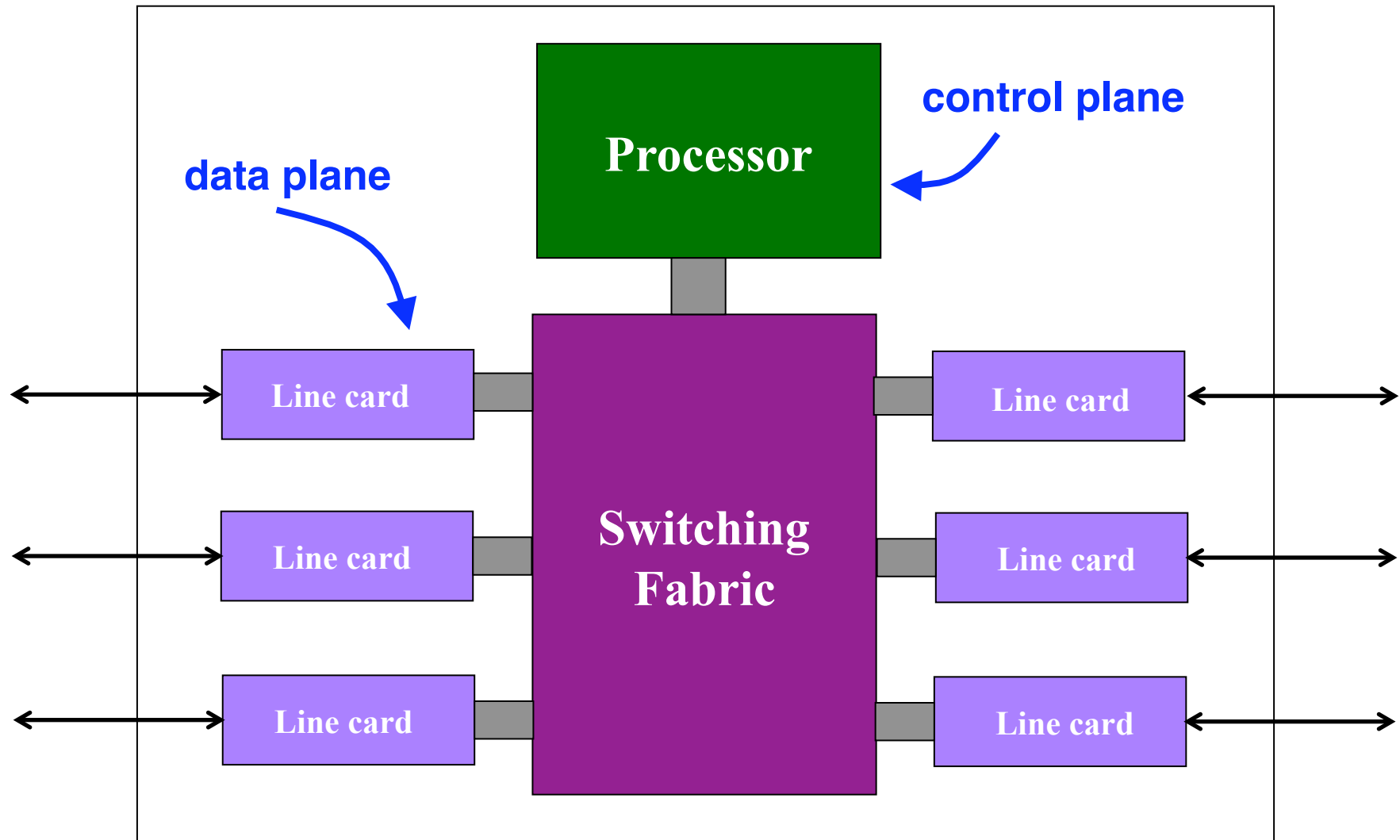


Routing vs. Forwarding

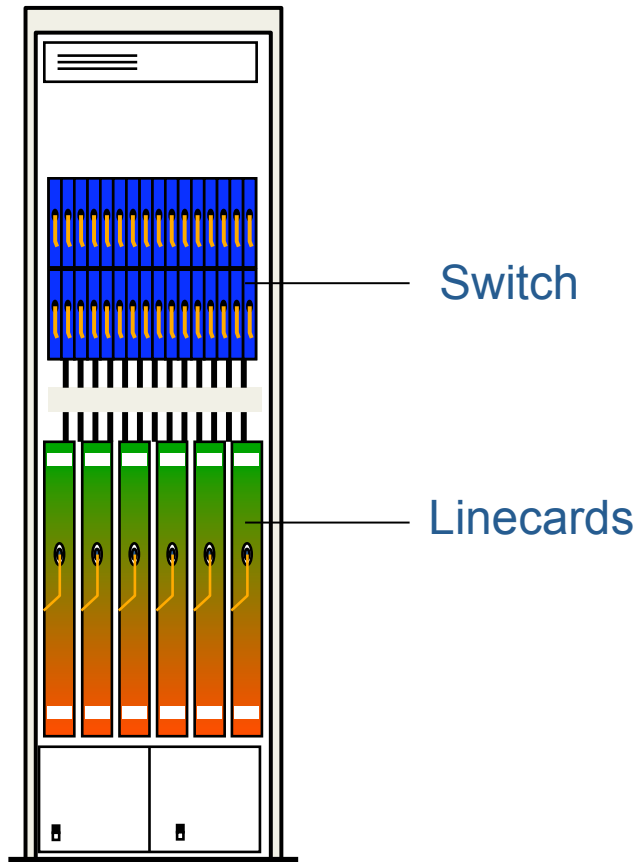
- **Routing:** control plane
 - Computing paths the packets will follow
 - Routers talking amongst themselves
 - Individual router *creating* a forwarding table
- **Forwarding:** data plane
 - Directing a data packet to an outgoing link
 - Individual router *using* a forwarding table



Data and Control Planes



Router Physical Layout



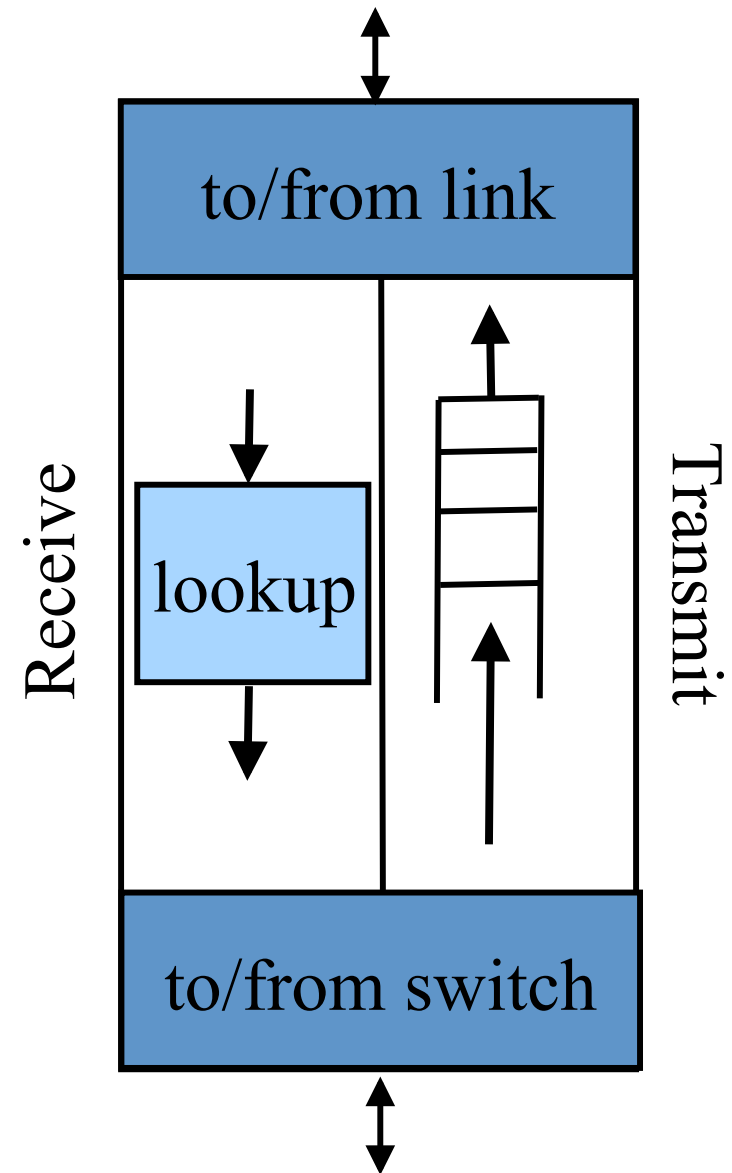
Juniper T series



Cisco 12000

Line Cards (Interface Cards, Adaptors)

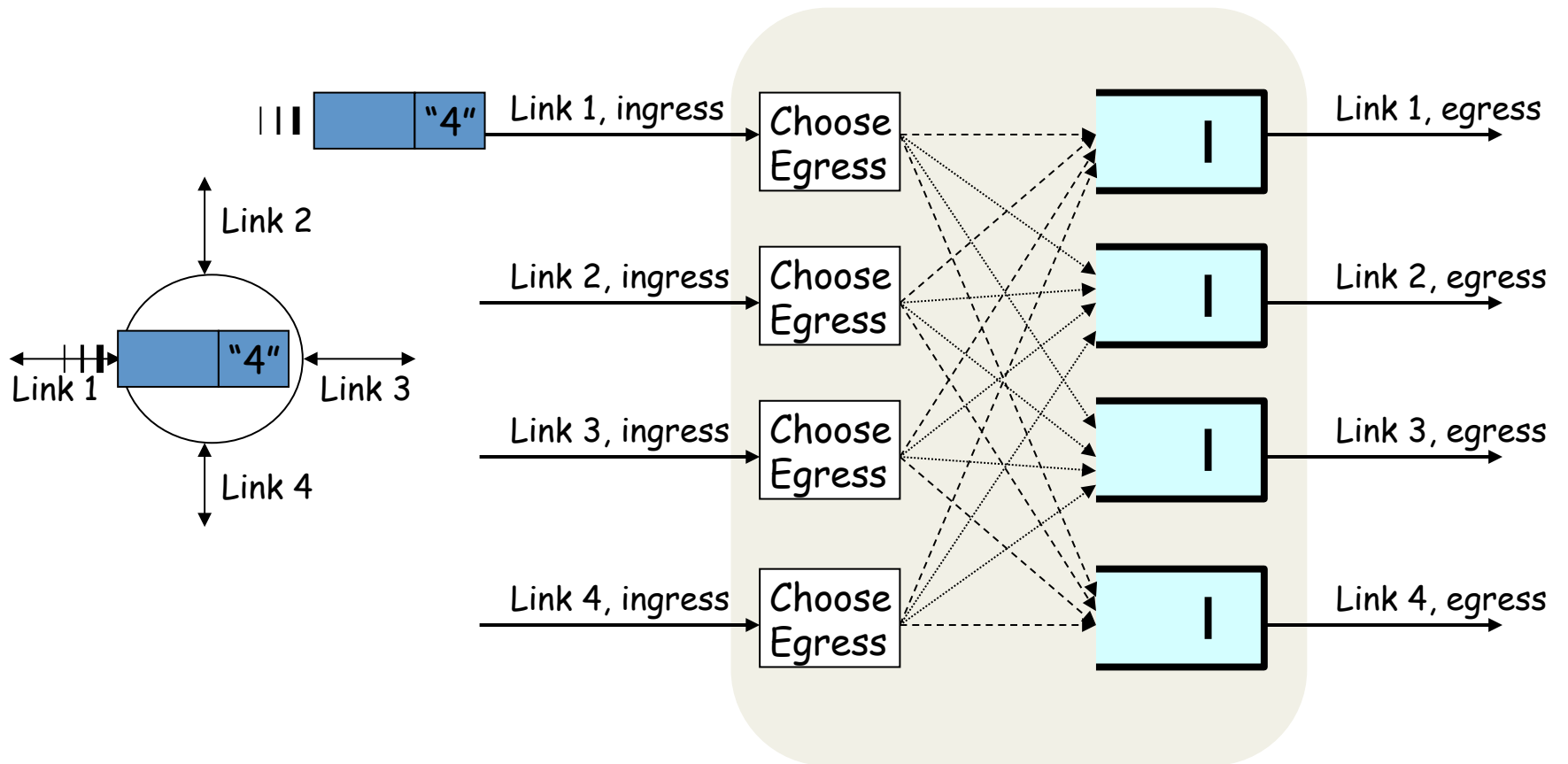
- **Interfacing**
 - Physical link
 - Switching fabric
- **Packet handling**
 - Packet forwarding
 - Decrement time-to-live
 - Buffer management
 - Link scheduling
 - Packet filtering
 - Rate limiting
 - Packet marking
 - Measurement



Switching Fabric

- Deliver packet inside the router
 - From incoming interface to outgoing interface
 - A small network in and of itself
- Must operate very quickly
 - Multiple packets going to same outgoing interface
 - Switch scheduling to match inputs to outputs
- Implementation techniques
 - Bus, crossbar, interconnection network, ...
 - Running at a faster speed (e.g., 2X) than links
 - Dividing variable-length packets into fixed-size cells

Packet Switching



Router Processor

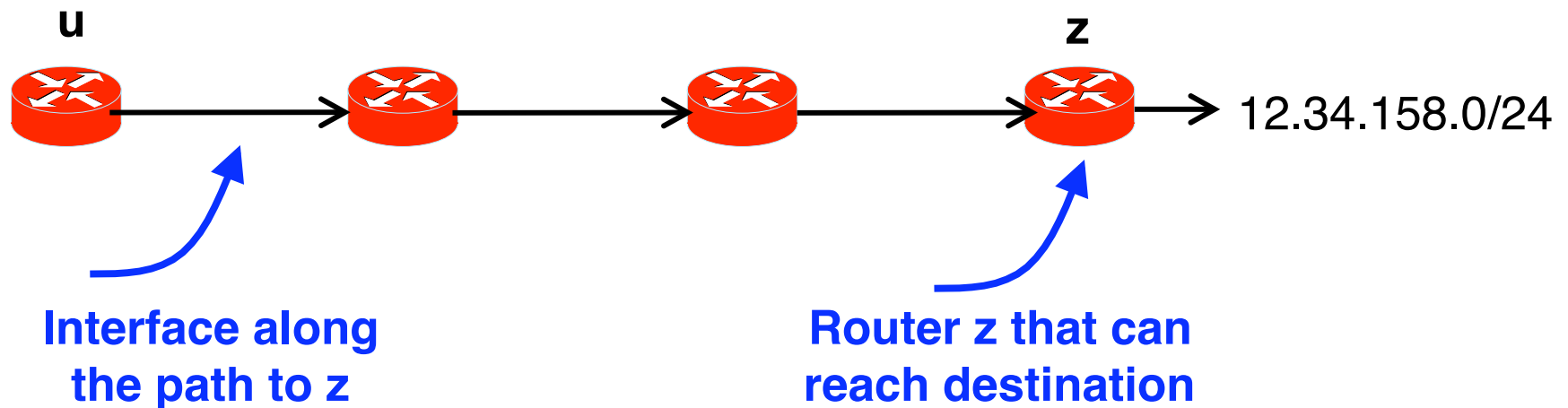
- So-called “Loopback” interface
 - IP address of the CPU on the router
- Interface to network administrators
 - Command-line interface for configuration
 - Transmission of measurement statistics
- Handling of special data packets
 - Packets with IP options enabled
 - Packets with expired Time-To-Live field
- Control-plane software
 - Implementation of the routing protocols
 - Creation of forwarding table for the line cards

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

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



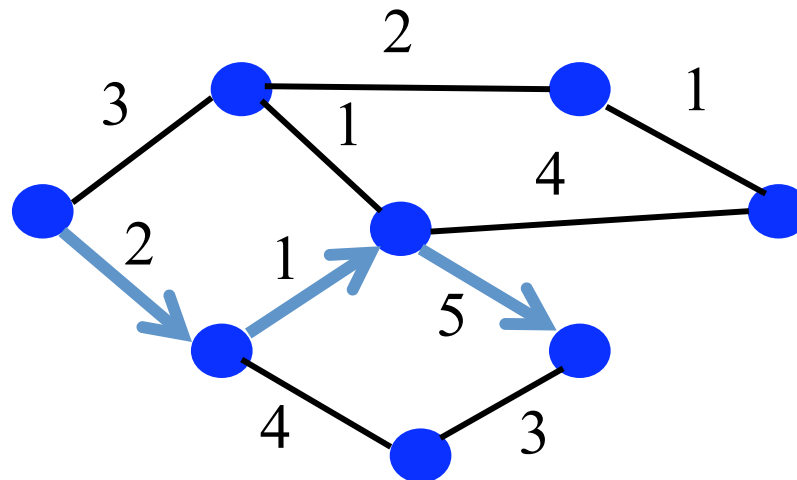
- Today's class: just how routers reach each other
 - How *u* knows how to forward packets toward *z*

Computing the Shortest Paths

assuming you already know
the topology

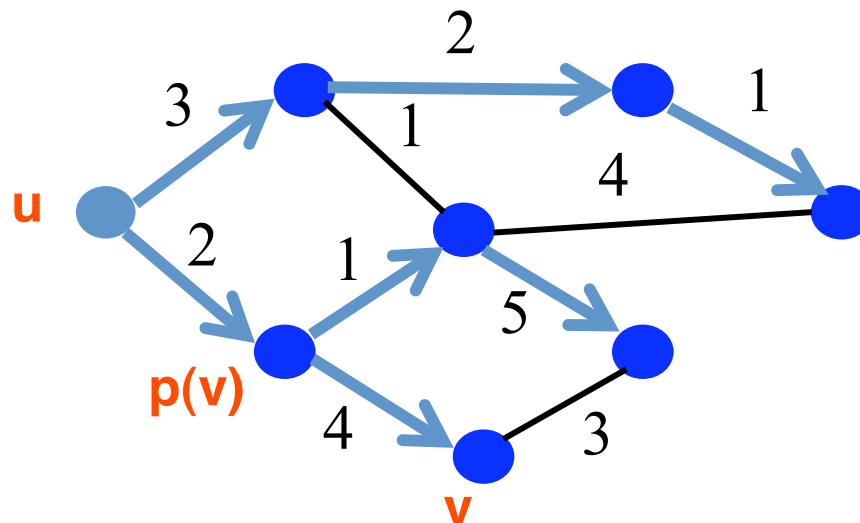
Shortest-Path Routing

- 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, $\mathbf{S} = \{\mathbf{u}\}$ 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, $\mathbf{D(v)} = \mathbf{c(u,v)}$ for all nodes v adjacent to u
 - ... and $\mathbf{D(v)} = \infty$ for all other nodes v
 - Continually update $D(v)$ as shorter paths are learned

Dijkstra's Algorithm

1 **Initialization:**

2 $S = \{u\}$

3 for all nodes v

4 if (v is adjacent to u)

5 $D(v) = c(u, v)$

6 else $D(v) = \infty$

7

8 **Loop**

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

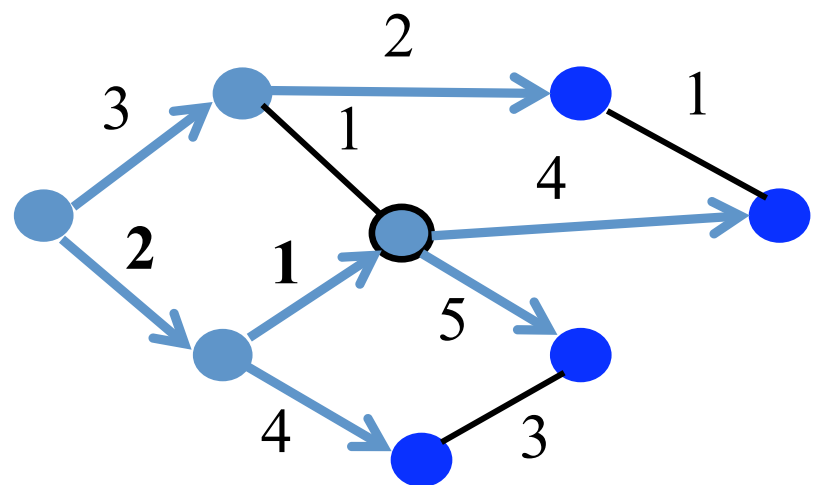
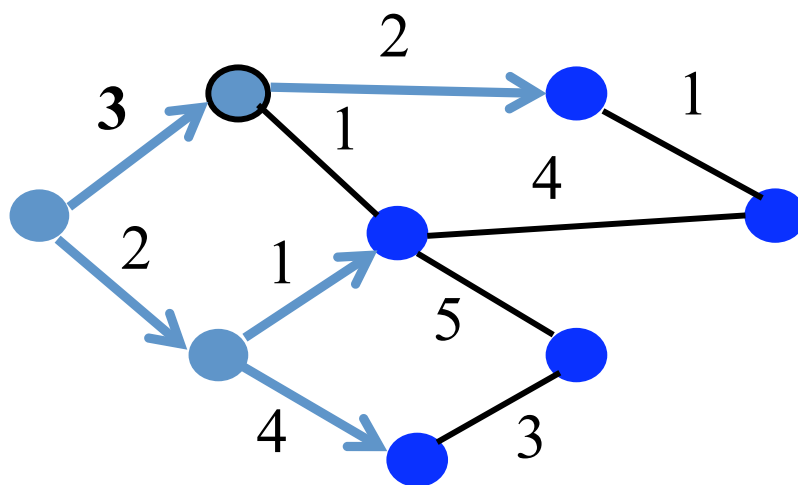
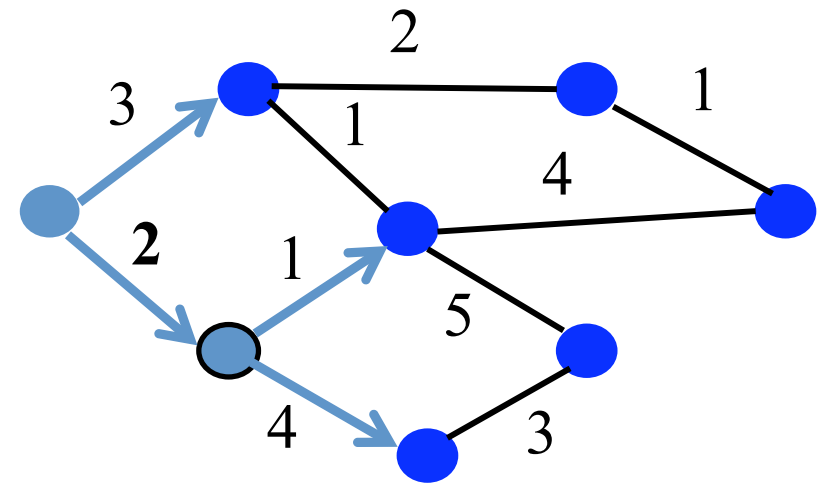
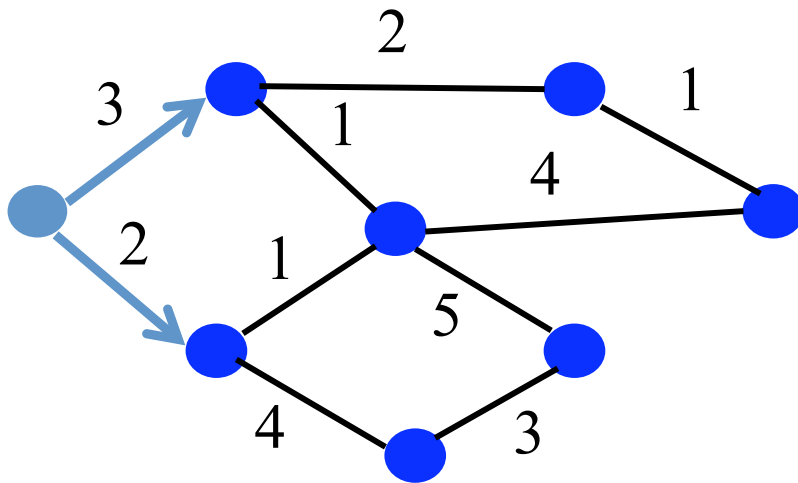
10 add w to S

11 update $D(v)$ for all v adjacent to w and not in S :

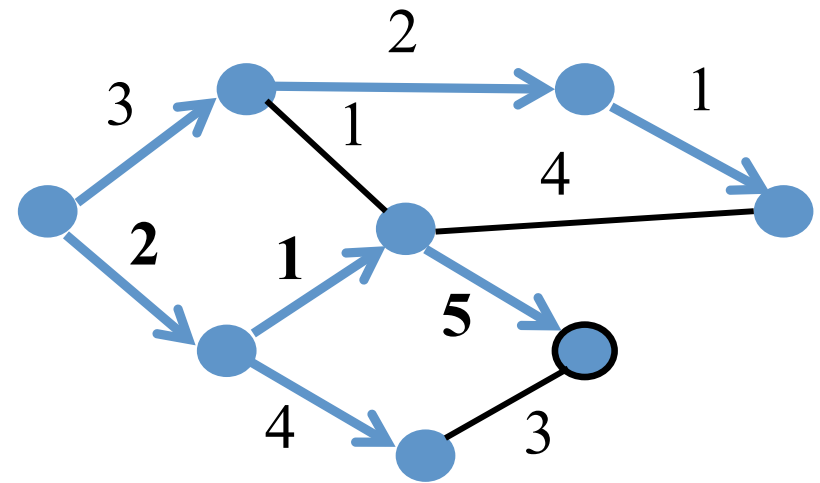
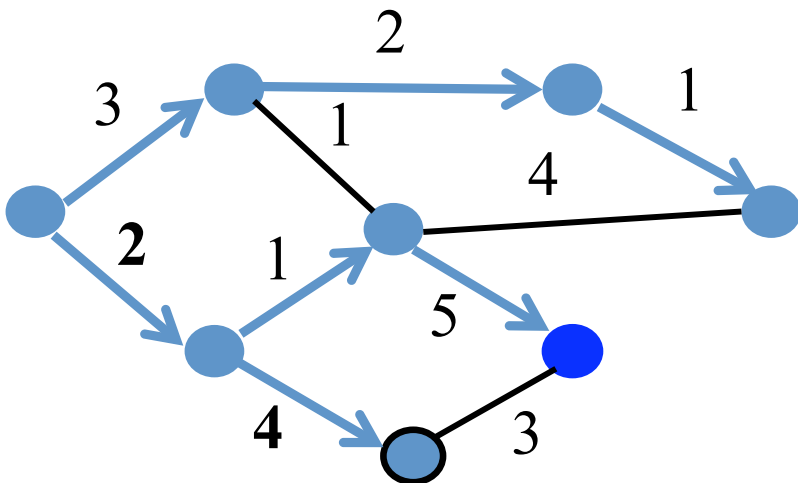
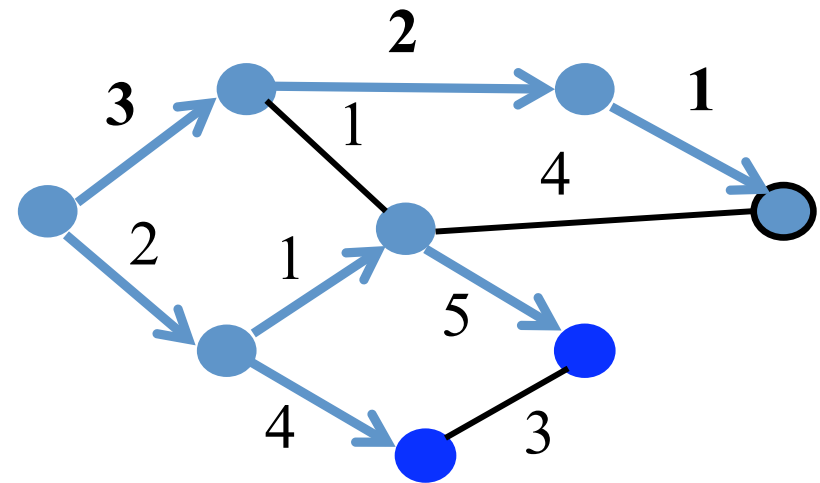
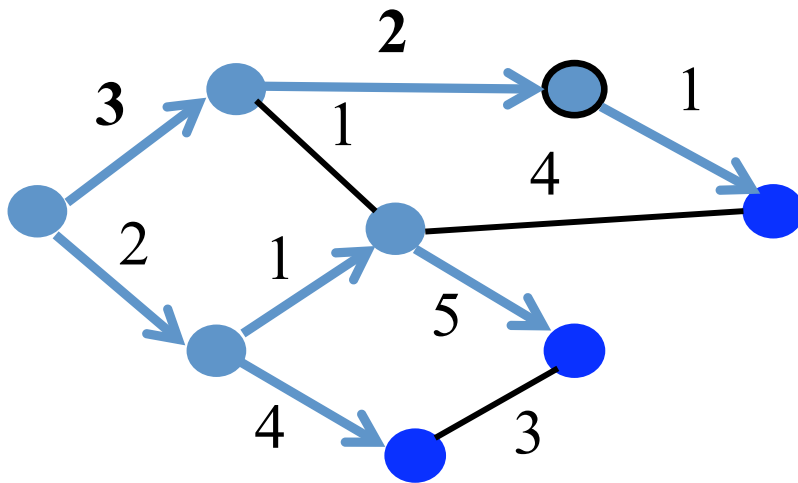
12 $D(v) = \min\{D(v), D(w) + c(w, v)\}$

13 **until all nodes in S**

Dijkstra's Algorithm Example

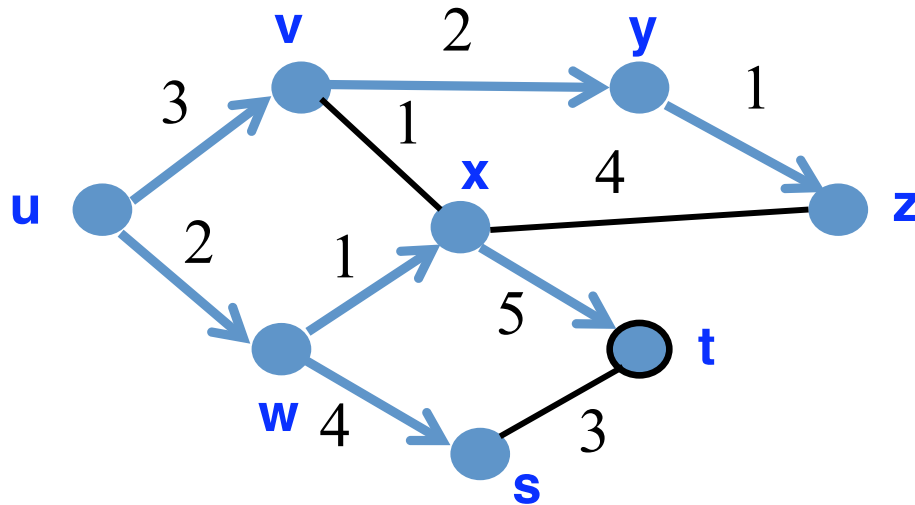


Dijkstra's Algorithm Example



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”

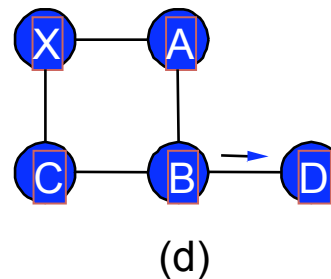
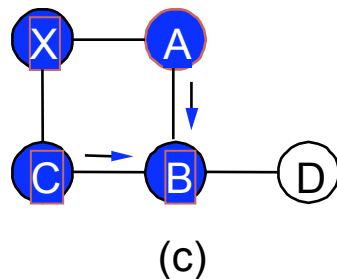
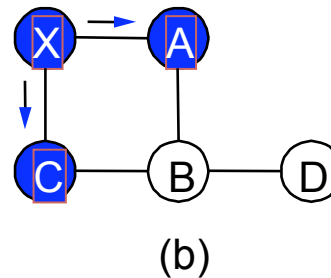
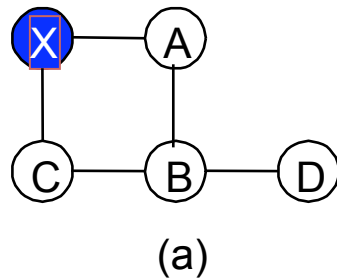


- **Performance trade-offs**

- 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 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

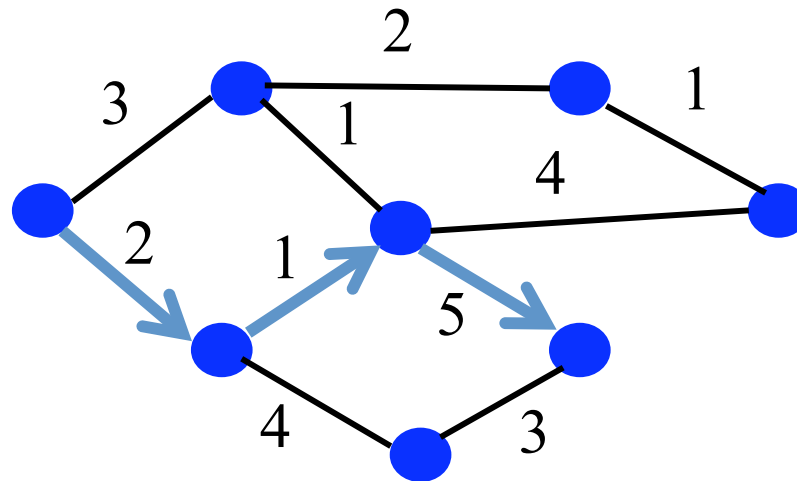
- **Topology change**
 - Link or node failure
 - Link or node recovery
- **Configuration change**
 - Link cost change
- **Periodically**
 - Refresh the link-state information
 - Typically (say) 30 minutes
 - Corrects for possible corruption of the data

When the Routers Disagree

(during transient periods)

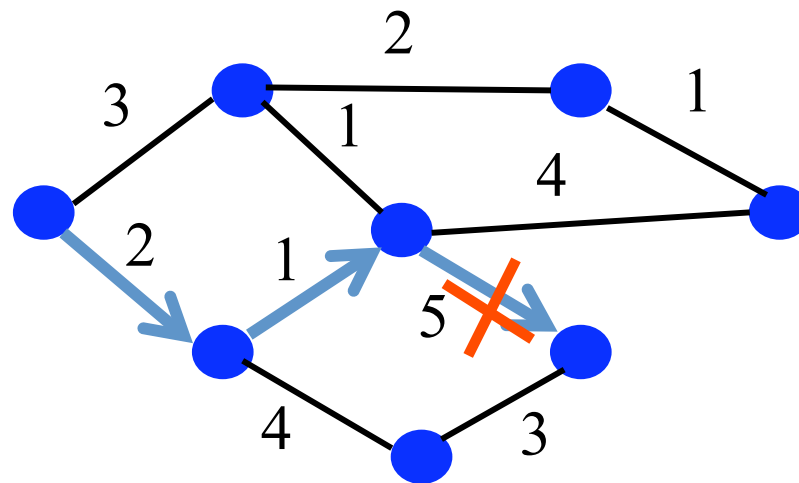
Convergence

- 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



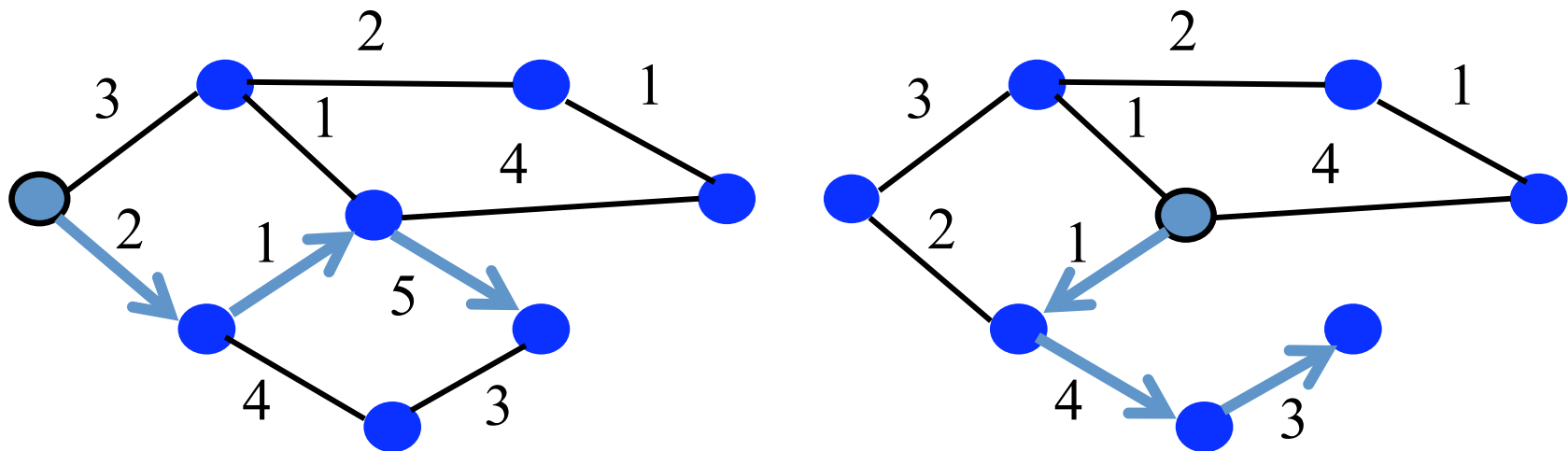
Transient Disruptions

- **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

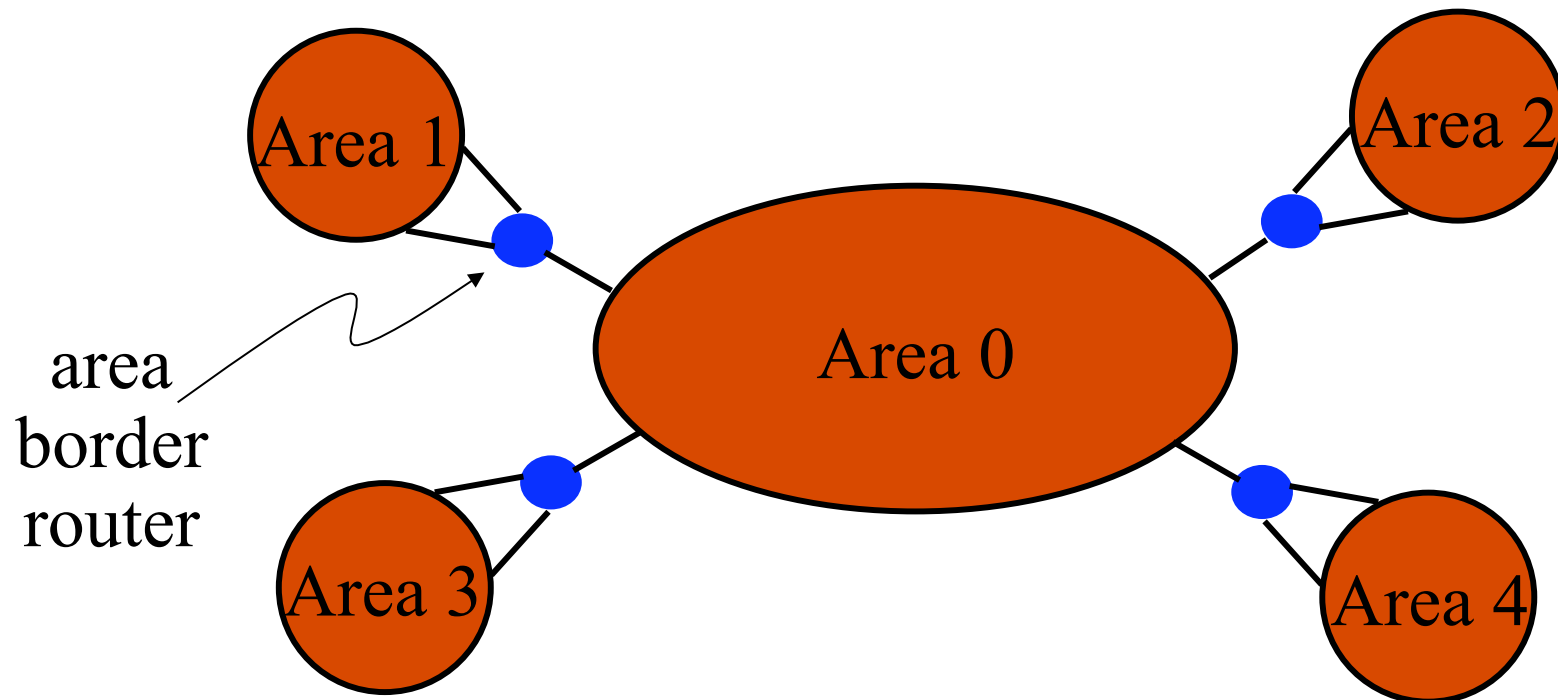
- Sources of convergence delay
 - Detection latency
 - Flooding of link-state information
 - Shortest-path computation
 - Creating the forwarding table
- Performance during convergence period
 - Lost packets due to blackholes and TTL expiry
 - Looping packets consuming resources
 - Out-of-order packets 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