CS5283 – Week 5

Routing

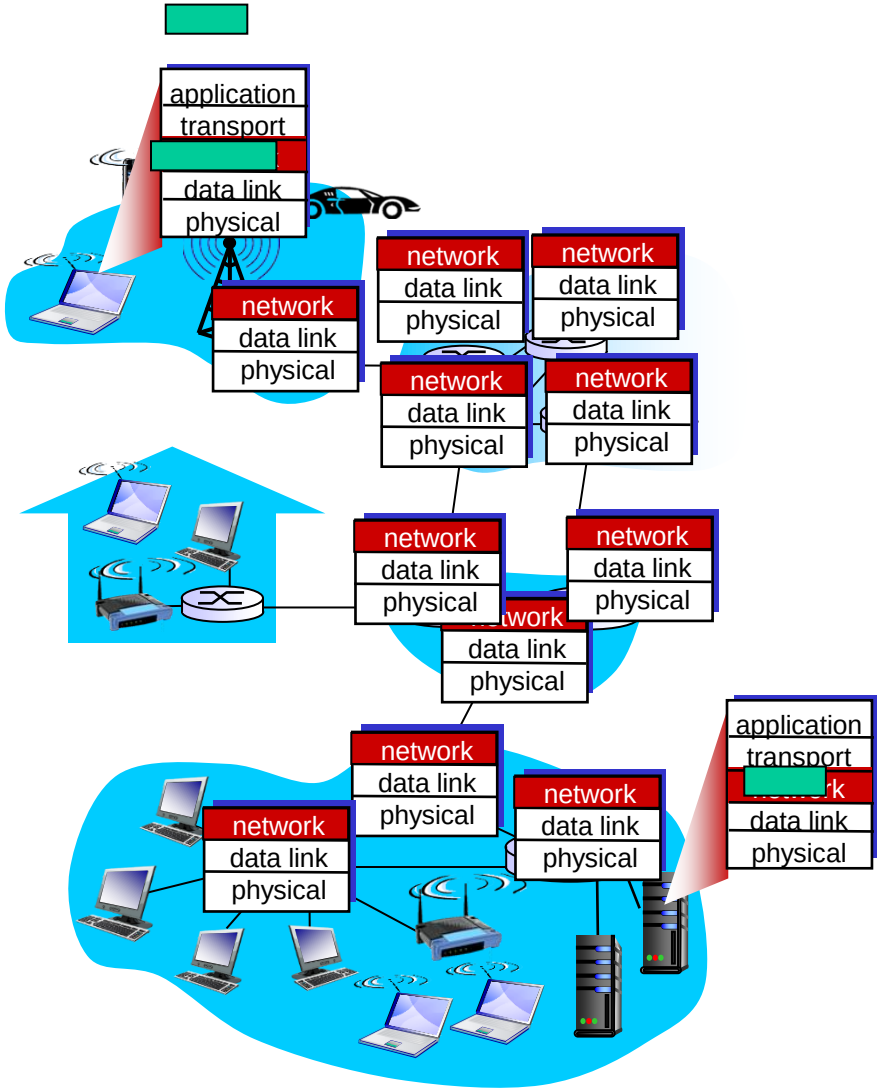
Outline

- Q/A on quiz 1
- Quiz 2 coming, planned to be due 10/7
- Homework 2 reminder, due 9/30
- Q/A on asynchronous content
  - Network Layer Services and Routing: Control Plane vs. Data Plane
  - Routing Protocols: Dijkstra and Bellman-Ford
  - Internet Routing: OSPF, BGP, ASes
- Breakout activity with interactive book content: https://gaia.cs.umass.edu/kurose_ross/interactive/
- Breakout activity with WireShark lab: http://gaia.cs.umass.edu/kurose_ross/wireshark.htm

Network-Layer Services

Network Layer

- Transport segment from sending to receiving host
- On the sending side, encapsulates segments into datagrams
- On the receiving side, delivers segments to the transport layer
- The network-layer protocols in **every** host, router
- The router examines the header fields in all IP datagrams passing through it



Two Key Network-Layer Functions

Network-layer functions:

- ***Forwarding*:** move packets from the router's input to the appropriate router output
- ***Routing*:** determine the route taken by the packets from source to destination
  - *Routing algorithms*

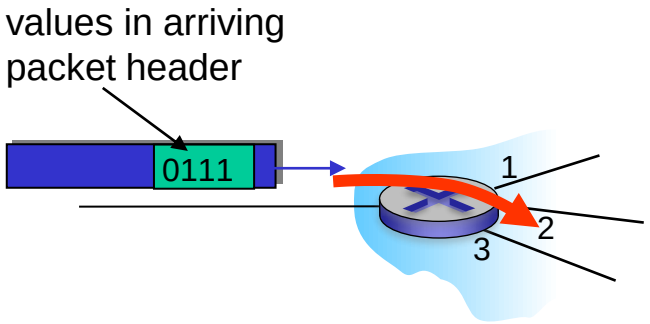
Analogy: taking a trip

- ***Forwarding*:** the process of getting through a single interchange
- ***Routing*:** the process of planning the trip from source to destination

Network Layer: Data Plane, Control Plane

Data plane

- Local, per-router function
- Determines how the datagram arriving on the router input port is forwarded to the router output port
- Forwarding function



Control plane

- Network-wide logic
- Determines how the datagram is routed among routers along end-end path from source host to destination host
- Two control-plane approaches:
  1. ***Traditional routing algorithms:*** implemented in routers
  2. ***Software-defined networking (SDN):*** implemented in (remote) servers

Network-Layer Control Plane

- **Goals:** understand the principles behind the network control plane
  - Traditional routing algorithms
  - SDN controllers
  - Internet Control Message Protocol (ICMP)
  - Network management
- And their instantiation, implementation in the Internet:
  - OSPF, BGP, OpenFlow, ODL and ONOS controllers, ICMP, SNMP

Network-Layer Functions

- **Recall: two network-layer functions:**
  1. ***Forwarding*: move packets from the router's input to the appropriate router output**
  2. ***Routing*: determine the route taken by packets from source to destination**

Data plane

Control plane

- **Two approaches to structuring the network control plane:**
  1. Per-router control (traditional)
  2. Logically centralized control (software defined networking)

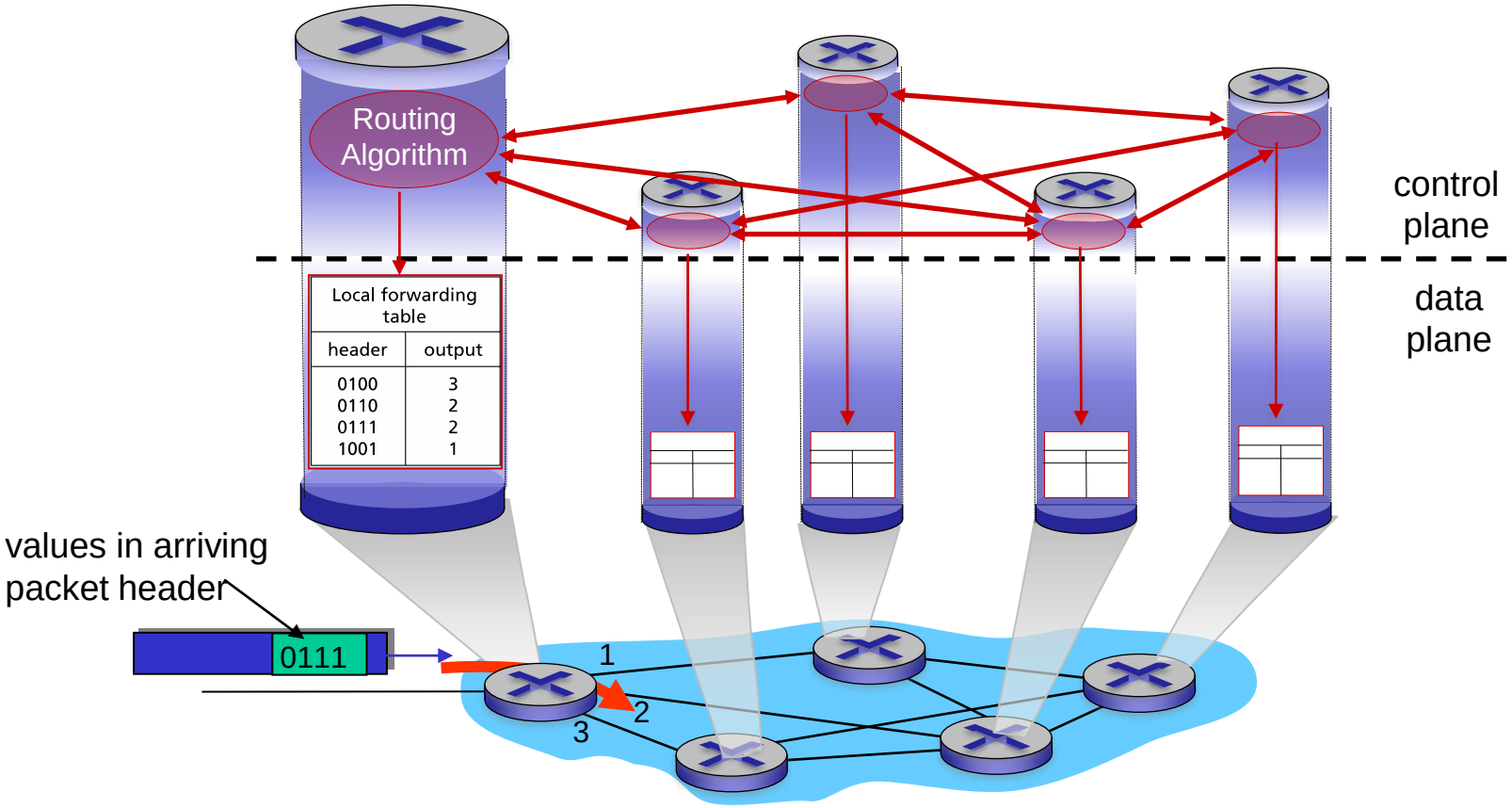
The End

Routing

Control Plane

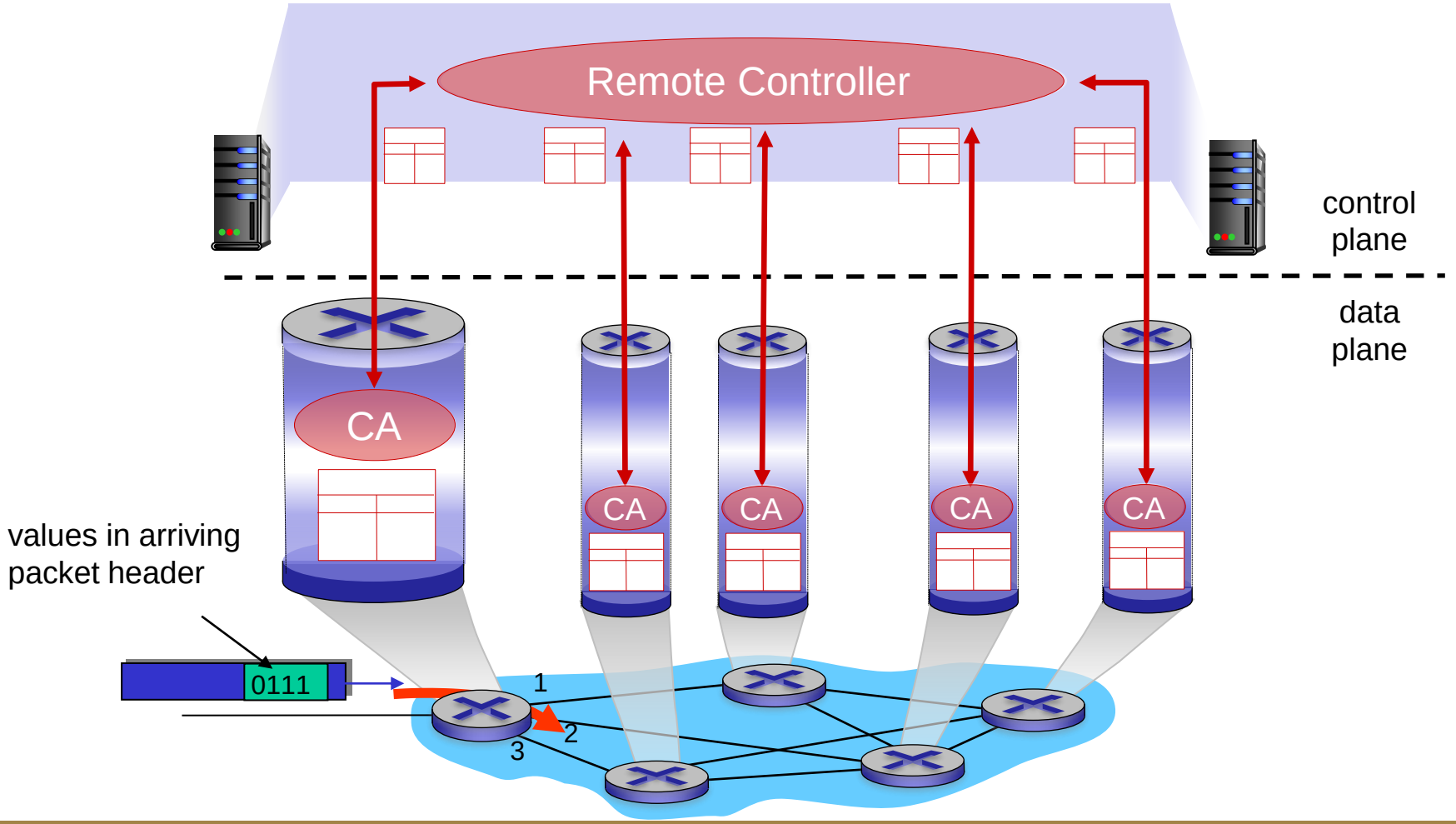
Per-Router Control Plane

The individual routing algorithm components **in each and every router** interact in the control plane.



Logically Centralized Control Plane

A distinct (typically remote) controller interacts with local control agents (CAs).



The End

Forwarding

Data Plane

Network Service Model

Q: What is the ***service model*** for "channel" transporting datagrams from sender to receiver?

Example services for individual datagrams:

- Guaranteed delivery
- Guaranteed delivery with less than 40 msec delay

Example services for a flow of datagrams:

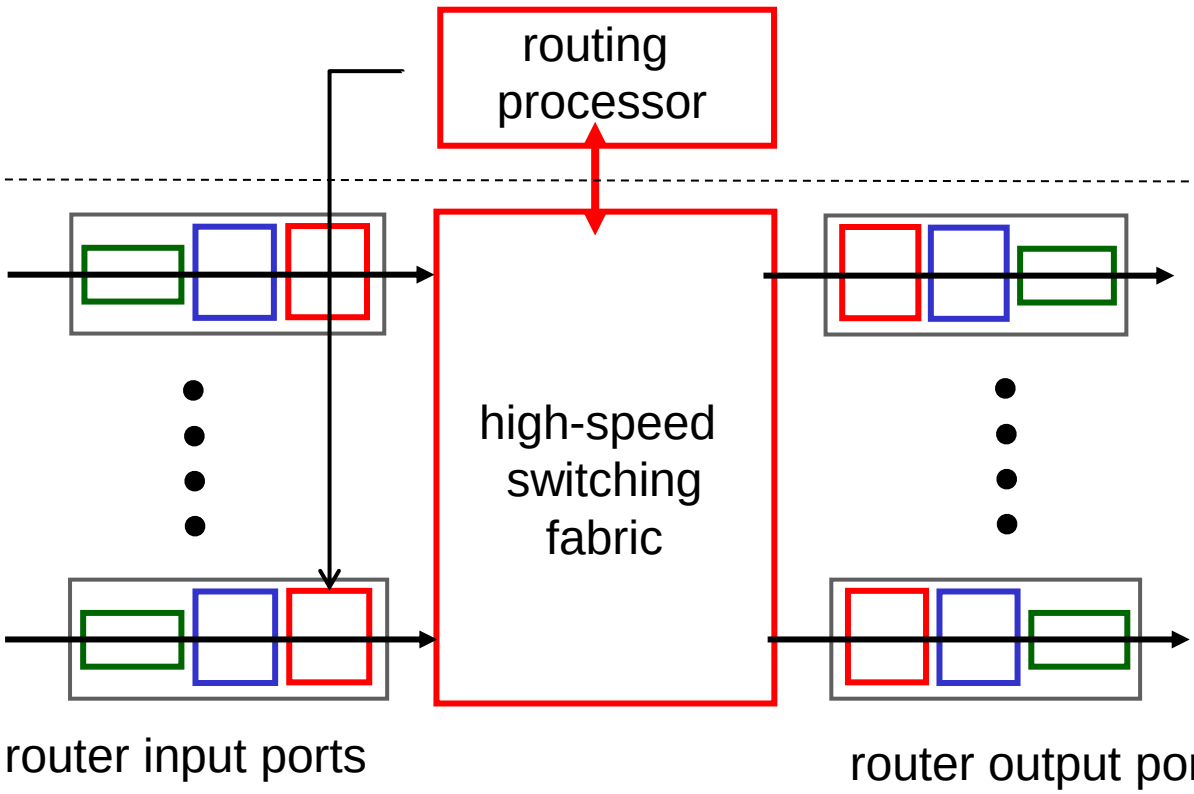
- In-order datagram delivery
- Guaranteed minimum bandwidth to flow
- Restrictions on changes in inter-packet spacing

Network-Layer Service Models

Network architecture	Service model	Guarantees?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Router Architecture Overview

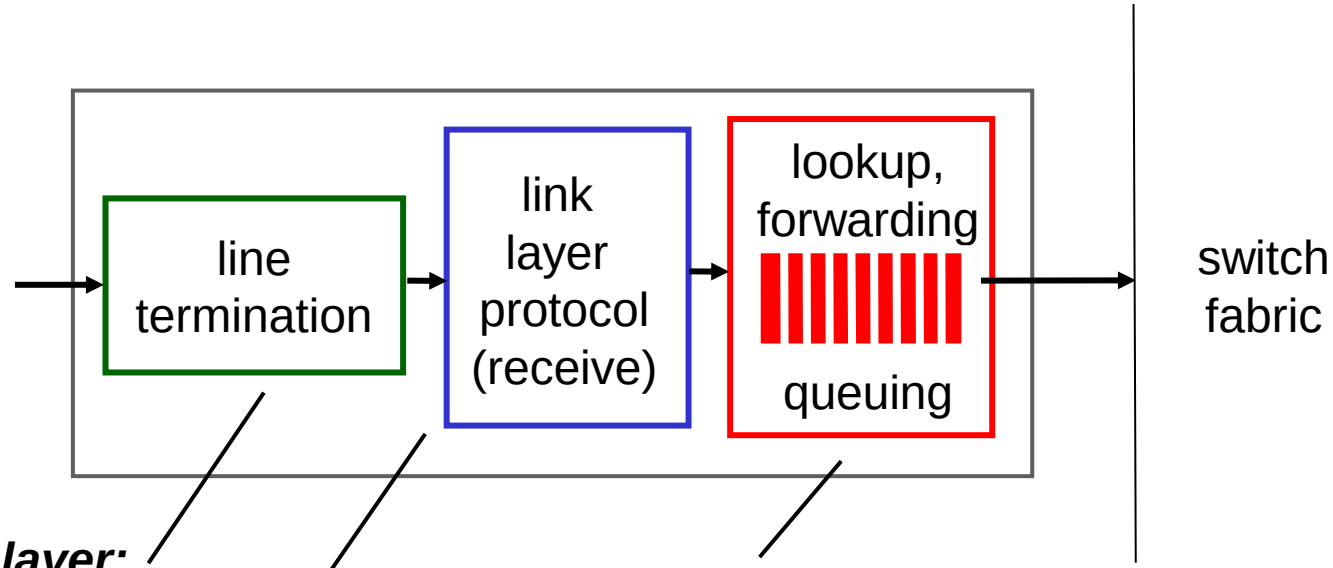
High-level view of generic router architecture:



Routing, management control plane (software) operates in millisecond time frame

Forwarding data plane (hardware) operates in nanosecond time frame

Input Port Functions



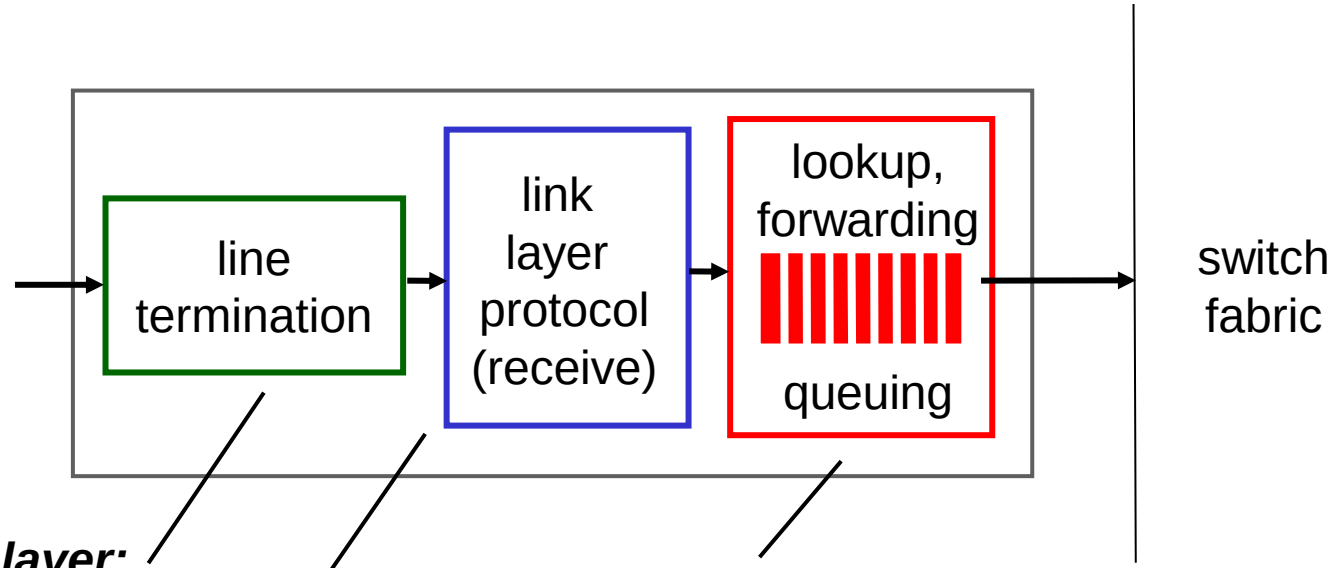
Physical layer:
bit-level reception

Data link layer:
e.g., Ethernet
see Chapter 5

Decentralized switching:

- Using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- Goal: complete input port processing at 'line speed'
- Queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input Port Functions



Physical layer:
bit-level reception

Data link layer:
e.g., Ethernet
see Chapter 5

Decentralized switching:

- Using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- **Destination-based forwarding:** forward based only on destination IP address (traditional)
- **Generalized forwarding:** forward based on any set of header field values

Destination-Based Forwarding

Forwarding table

Destination address range	Link interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Q: But what happens if ranges don't divide up so nicely?

Longest Prefix Matching

- When looking for the forwarding table entry for a given destination address, use the ***longest*** address prefix that matches the destination address.

Destination address range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

Examples:

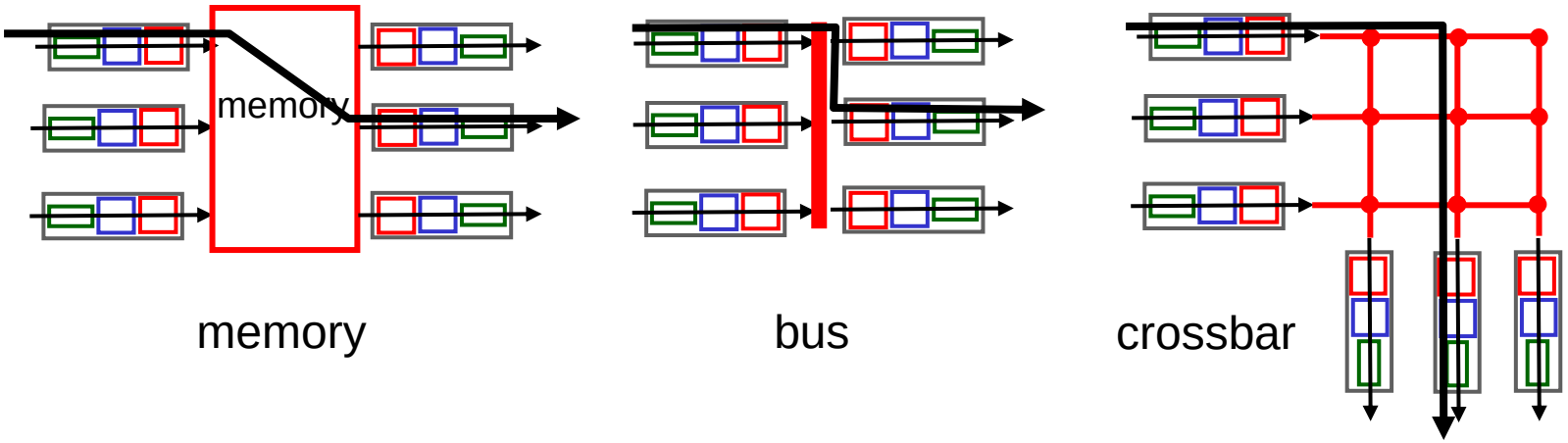
- DA: 11001000 00010111 00010110 10100001 **Which interface?**
- DA: 11001000 00010111 00011000 10101010 **Which interface?**

Longest Prefix Matching

- We'll see ***why*** the longest prefix matching is used shortly, when we study addressing
- The longest prefix matching: often performed using ternary content addressable memories (TCAMs)
  - **Content addressable:** present address to TCAM; retrieve the address in one clock cycle, regardless of table size
  - Cisco catalyst: can fit up to about one million routing table entries in TCAM

Switching Fabrics

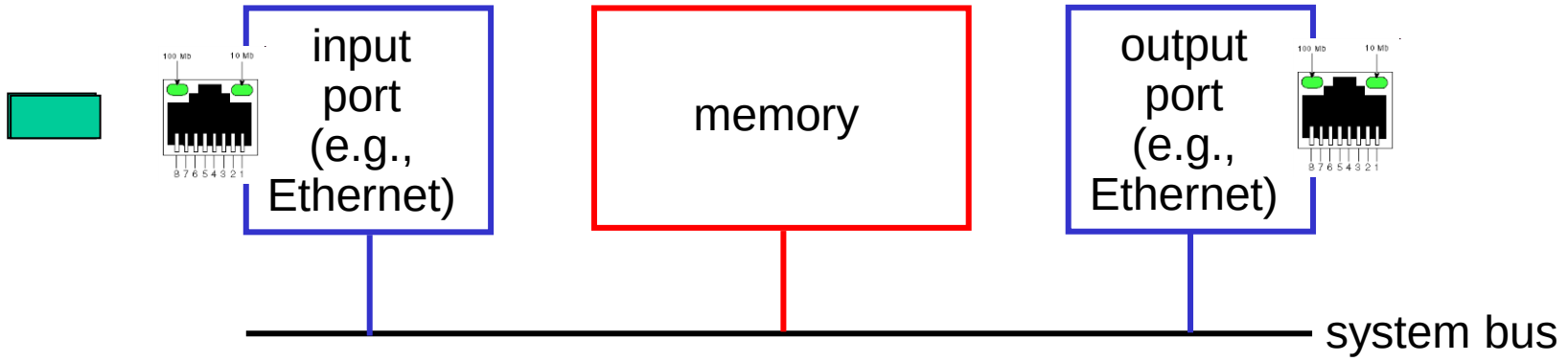
- Transfer the packet from the input buffer to the appropriate output buffer
- Switching rate: the rate at which packets can be transferred from input to output
  - Often measured as a multiple of the input/output line rate
  - N inputs: switching rate N times line rate desirable
- Three types of switching fabrics



Switching via Memory

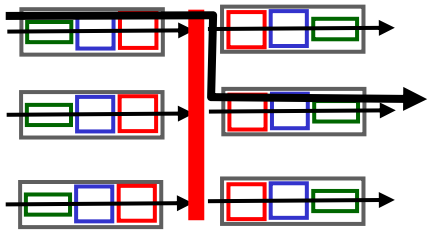
First generation routers:

- Traditional computers with switching under direct control of the CPU
- Packet copied to system's memory
- Speed limited by memory bandwidth (two bus crossings per datagram)



Switching via a Bus

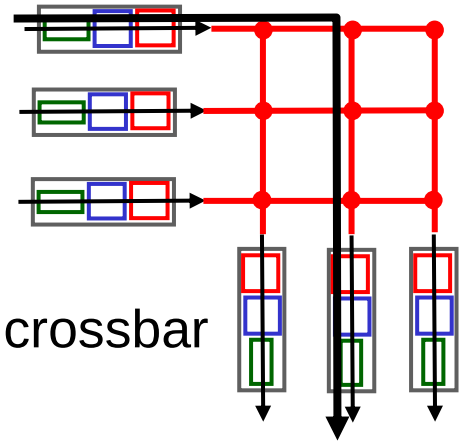
- Datagram from the input port memory to the output port memory via a shared bus
- **Bus contention:** switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers



bus

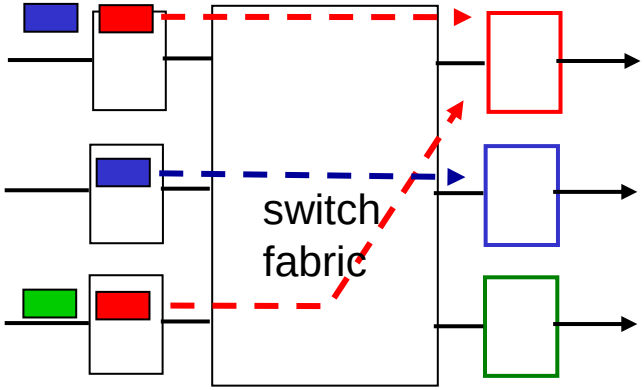
Switching via Interconnection Network

- Overcome bus bandwidth limitations
- Banyan networks, crossbar, other interconnection nets initially developed to connect processors in the multiprocessor
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric
- Cisco 12000: switches 60 Gbps through the interconnection network

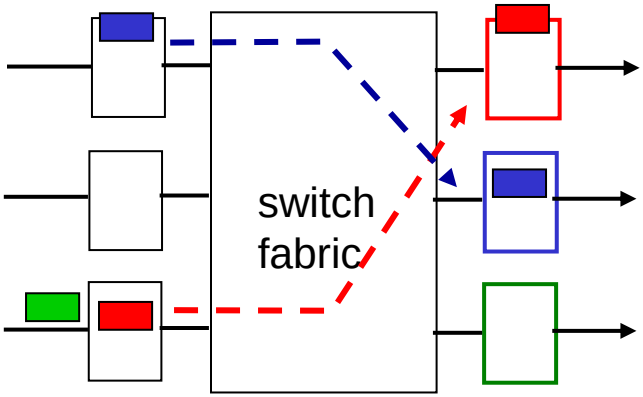


Input Port Queuing

- Fabric slower than input ports combined → queuing may occur at input queues
  - **Queuing delay and loss due to input buffer overflow!**
- **Head-of-the-line (HOL) blocking:** the queued datagram at the front of the queue prevents others in the queue from moving forward

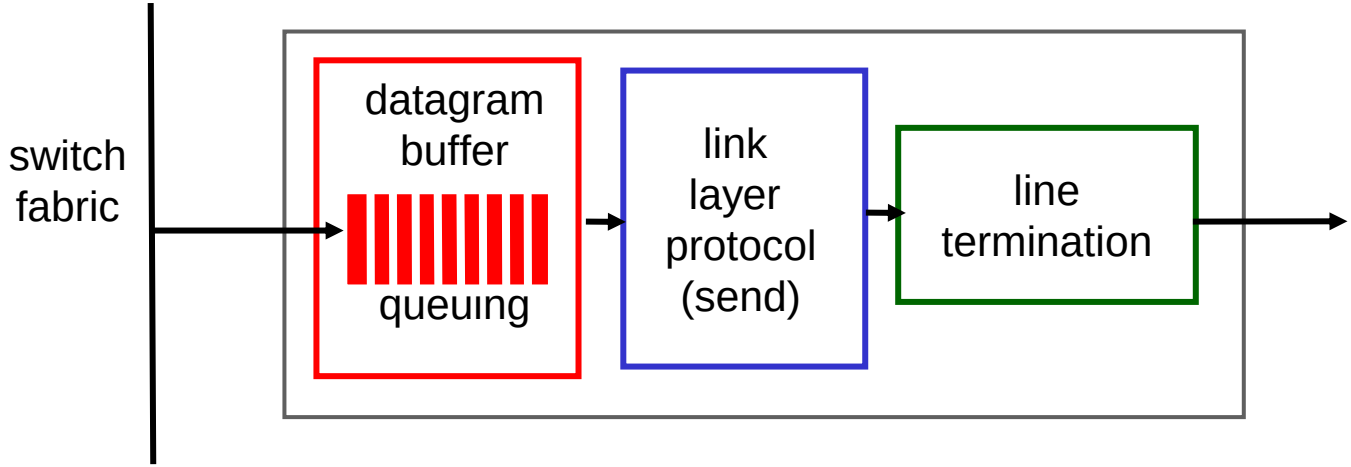


Output port contention:
only one red datagram can be transferred;
lower red packet is blocked



One packet time later: the green packet experiences HOL blocking

Output Ports

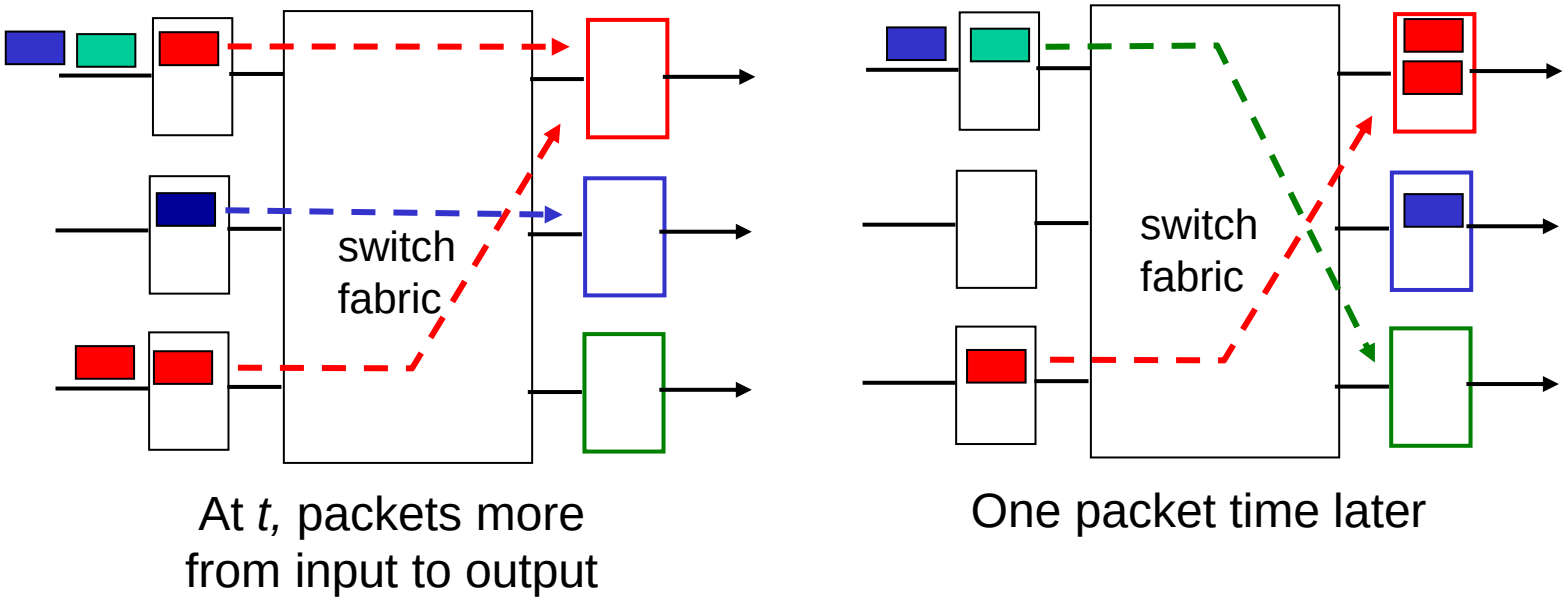


- **Buffering** required
  the fabric faster tha
- **Scheduling**
  datagrams fo

Datagram (packets) can be lost due to congestion, lack of buffers

Priority scheduling—who gets the best performance, network neutrality

Output Port Queuing



At *t*, packets more from input to output

One packet time later

- Buffering when arrival rate via switch exceeds output line speed
- **Queuing (delay) and loss due to output port buffer overflow!**

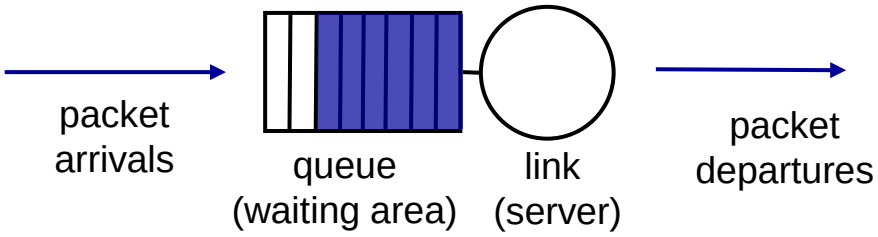
How Much Buffering?

- RFC 3439 rule of thumb: average buffering is equal to “typical” RTT (say 250 msec) times link capacity C
  - E.g., C = 10 Gbps link: 2.5 Gbit buffer
- Recent recommendation: with *N* flows, buffering equal to:

$$\frac{RTT \cdot C}{\sqrt{N}}$$

Scheduling Mechanisms

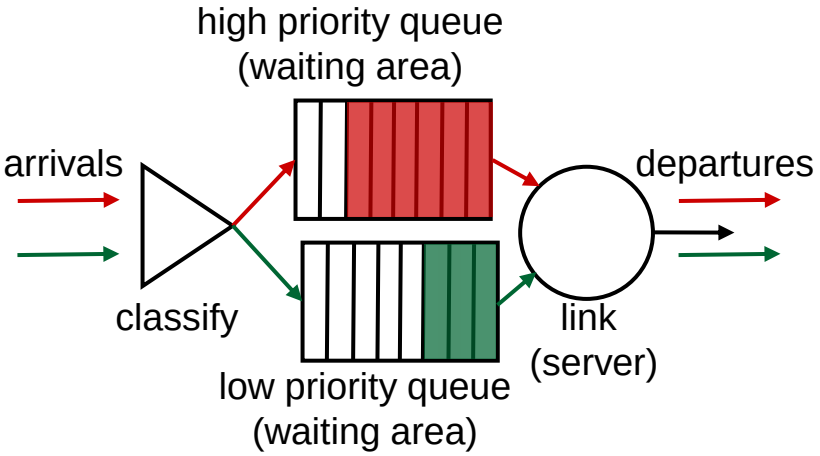
- **Scheduling:** choose the next packet to send on link
- **FIFO (first in first out) scheduling:** send in order of arrival to queue
  - Real-world example?
  - ***Discard policy:*** if packet arrives to full queue: who to discard?
    - ***Tail drop:*** drop arriving packet
    - ***Priority:*** drop/remove on priority basis
    - ***Random:*** drop/remove randomly

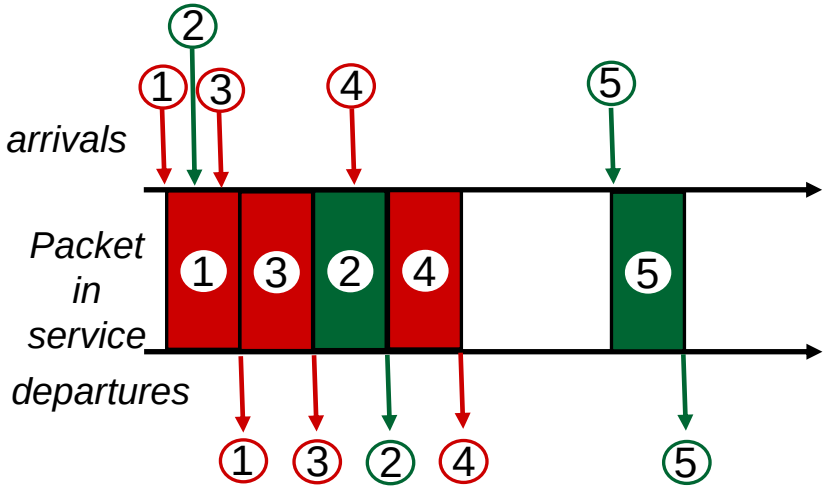


Scheduling Policies: Priority

Priority scheduling: send the highest priority queued packet

- Multiple *classes*, with different priorities
  - The class may depend on the marking or other header information, e.g., IP source/ destination, port numbers, etc.
  - Real-world example?

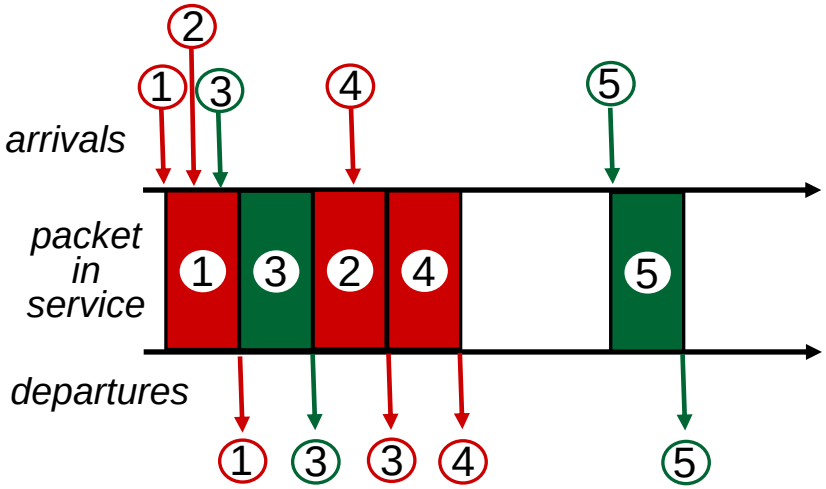




Scheduling Policies: Still More

Round Robin (RR) scheduling:

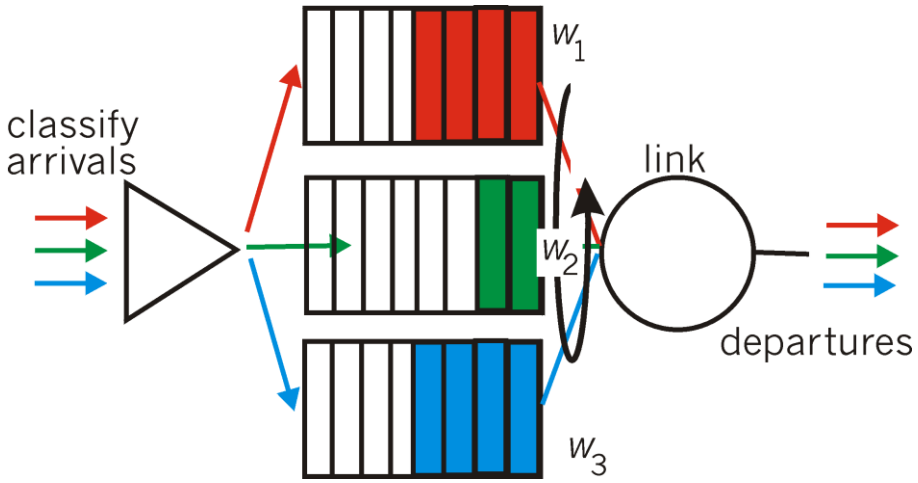
- Multiple classes
- Cyclically scan class queues, sending one complete packet from each class (if available)
- Real-world example?



Scheduling Policies: Still More

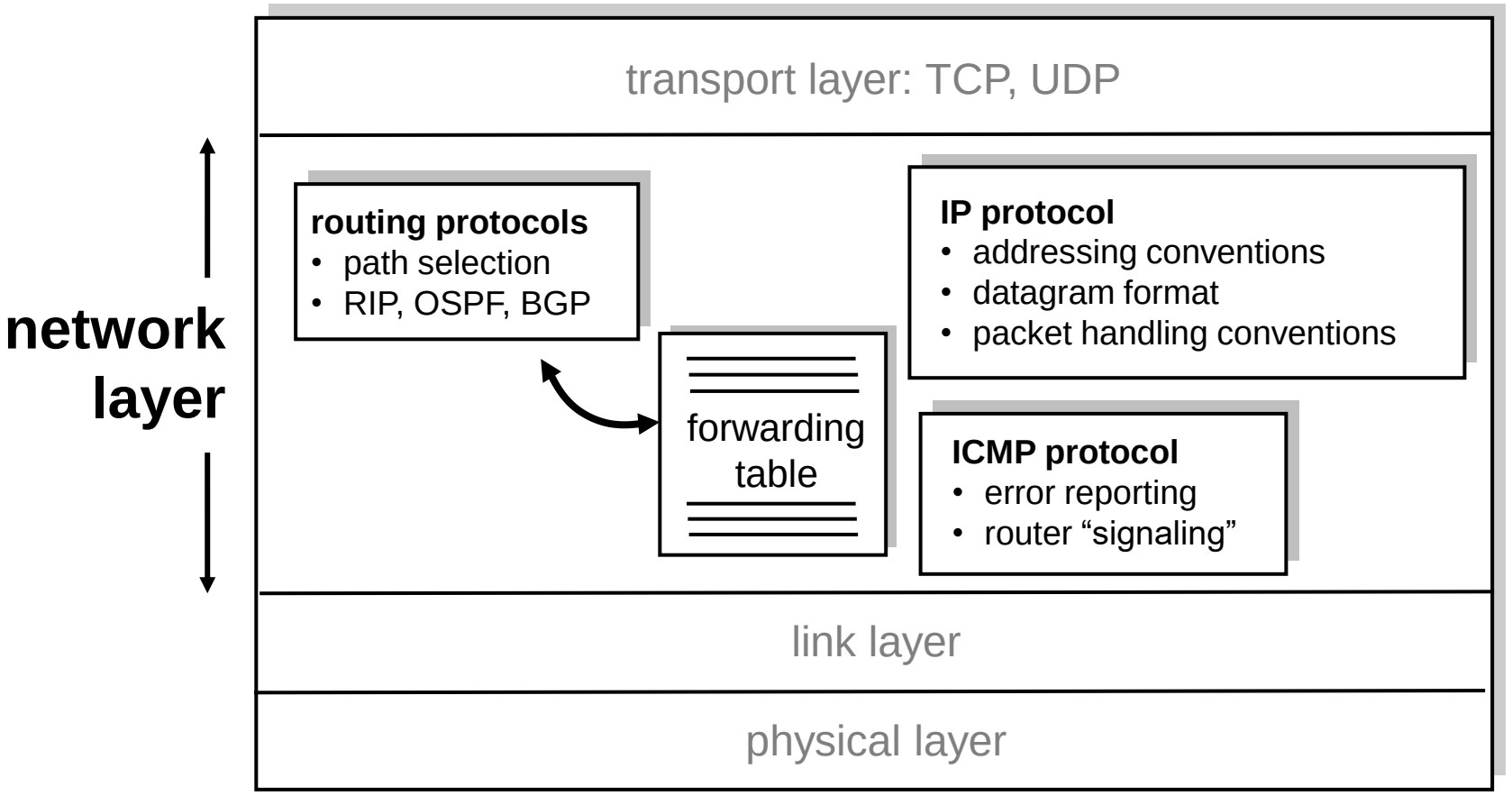
Weighted fair queuing (WFQ):

- Generalized Round Robin
- Each class gets a weighted amount of service in each cycle
- Real-world example?



The Internet Network Layer

Host, router network-layer functions:

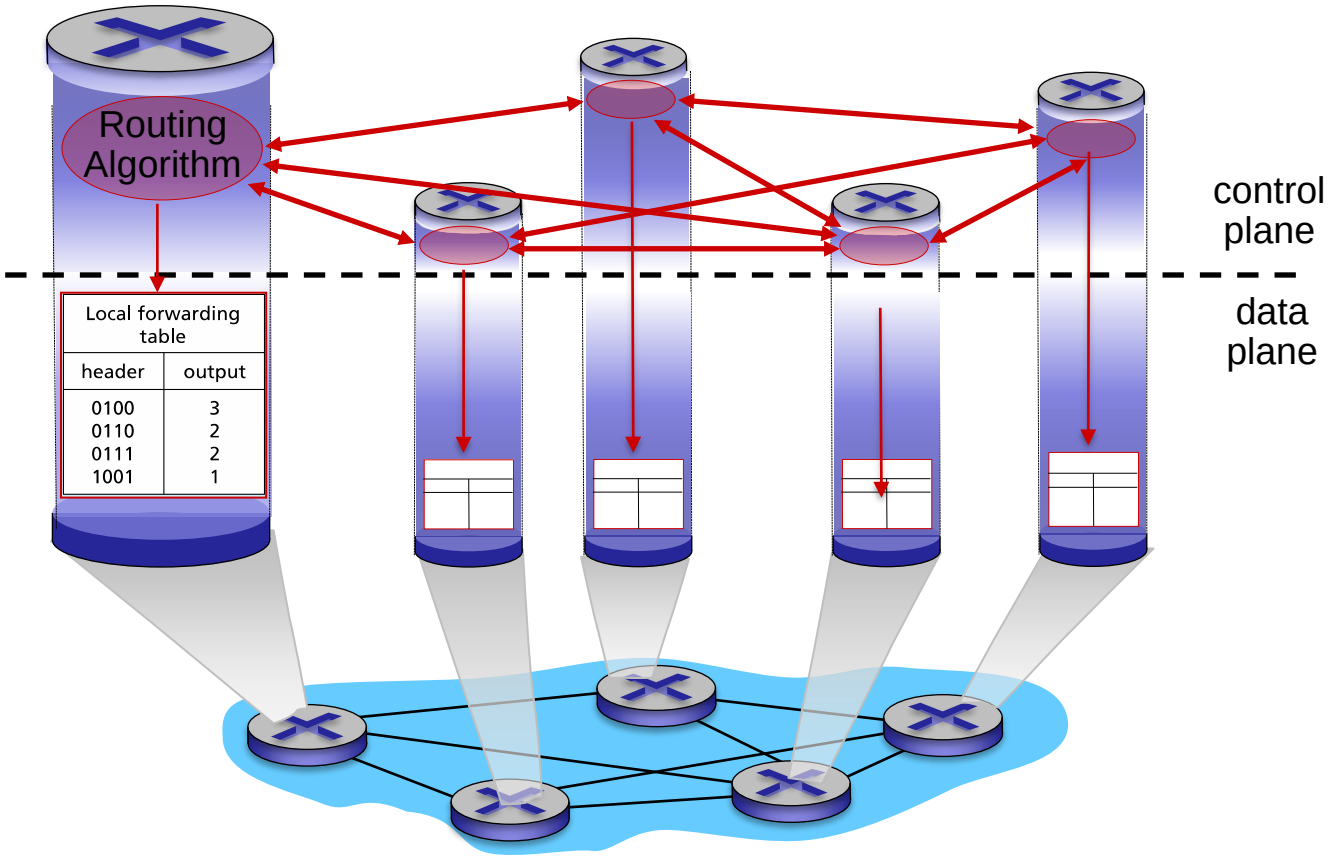


The End

Routing Protocols

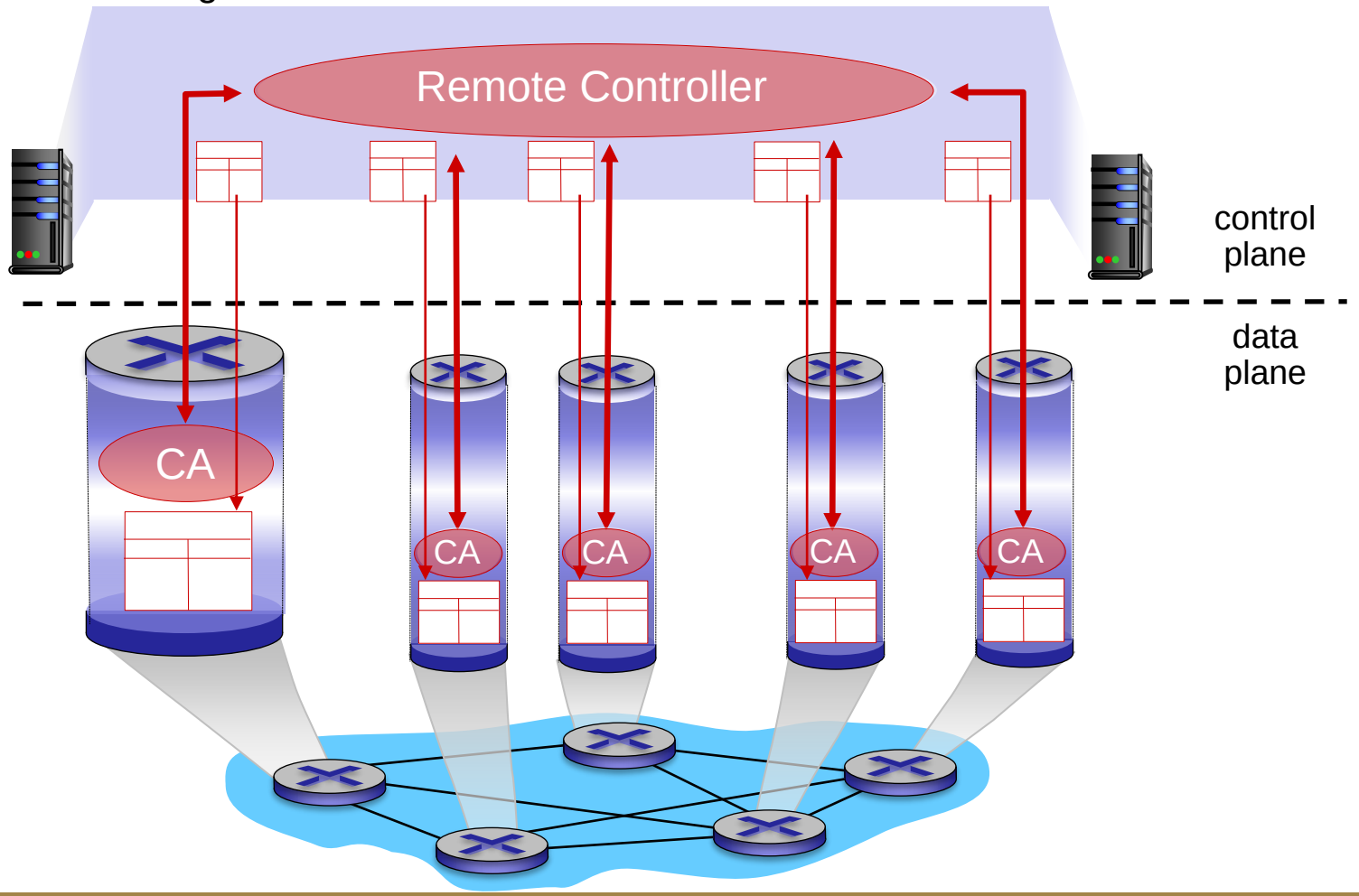
Per-Router Control Plane

Individual routing algorithm components **in each and every router** interact with each other in the control plane to compute forwarding tables.



Logically Centralized Control Plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables.

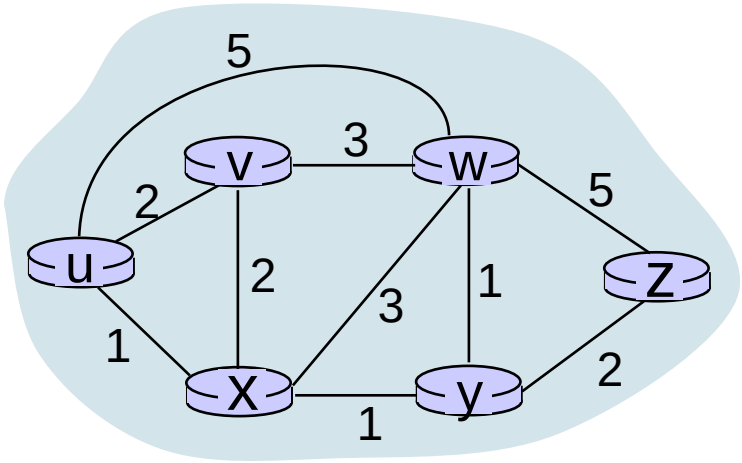


Routing Protocols

Routing protocol goal: determine the “good” paths (equivalently routes) from the sending hosts to the receiving host, through a network of routers

- Path: a sequence of routers the packets will traverse in going from the given initial source host to the given final destination host
- “Good”: least “cost,” “fastest,” “least congested”
- Routing: a “top-10” networking challenge!

Graph Abstraction of the Network



Graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links = { (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

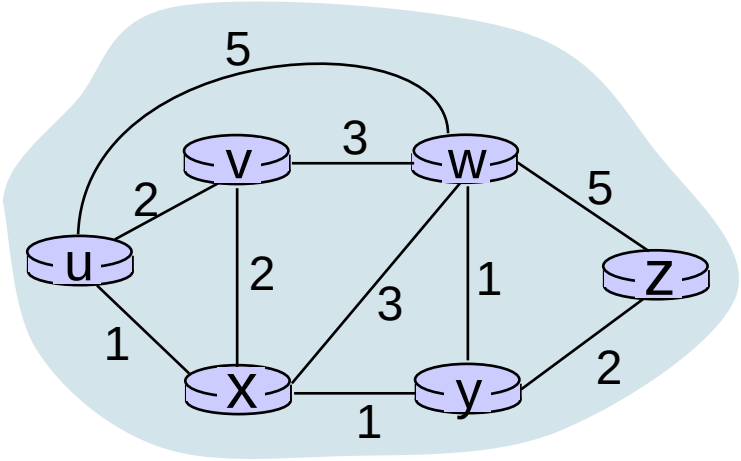
Aside: graph abstraction is useful in other network contexts, e.g.,

P2P, where *N* is a set of peers and *E* is a set of TCP connections

Graph Abstraction: Costs

c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

Cost could always be 1, or inversely related to bandwidth, or inversely related to congestion



Cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Key question: What is the least-cost path between u and z?

Routing algorithm: algorithm that finds that least cost path

The End

Dijkstra’s Algorithm

Routing Algorithm Classification

- **Q: Global or decentralized information?**
- **Global:**
  - All routers have complete topology, link cost information
  - ***"Link-state" algorithms***
- **Decentralized:**
  - The router knows physically-connected neighbors, link costs to neighbors
  - Iterative process of computation, exchange of information with neighbors
  - ***"Distance vector" algorithms***

- **Q: Static or dynamic?**
- **Static:**
  - Routes change slowly over time
- **Dynamic:**
  - Routes change more quickly
    - Periodic update
    - In response to link cost changes

A Link-State Routing Algorithm

Dijkstra's algorithm

- Net topology, link costs known to all nodes
  - Accomplished via "link-state broadcast"
  - All nodes have the same information
- Computes least cost paths from one node ("source") to all other nodes
  - Gives ***forwarding table*** for that node
- Iterative: after k iterations, know the least cost path to k destination 's

Notation

- ***c(x,y):*** link cost from node x to y; = ∞ if not direct neighbors
- ***D(v):*** current value of cost of path from source to destination v
- ***p(v):*** predecessor node along path from source to v
- ***N':*** set of nodes whose least cost path is definitively known

Dijkstra's Algorithm

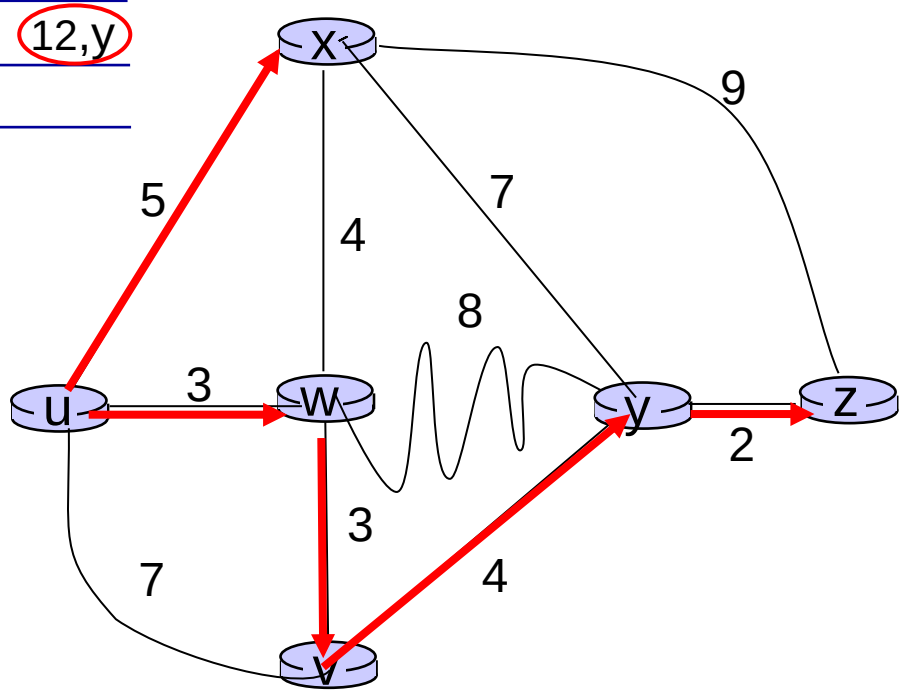
1 ***Initialization:***
2 N' = {u}
3 for all nodes v
4 if v adjacent to u
5 then D(v) = c(u,v)
6 else D(v) = ∞
7
8 ***Loop***
9 find w not in N' such that D(w) is a minimum
10 add w to N'
11 update D(v) for all v adjacent to w and not in N' :
12 **D(v) = min(D(v), D(w) + c(w,v))**
13 /* new cost to v is either old cost to v or known
14 shortest path cost to w plus cost from w to v */
15 ***until all nodes in N'***

Dijkstra's Algorithm: Example

Step	N'	D(**v**) p(v)	D(**w**) p(w)	D(**x**) p(x)	D(**y**) p(y)	D(**z**) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

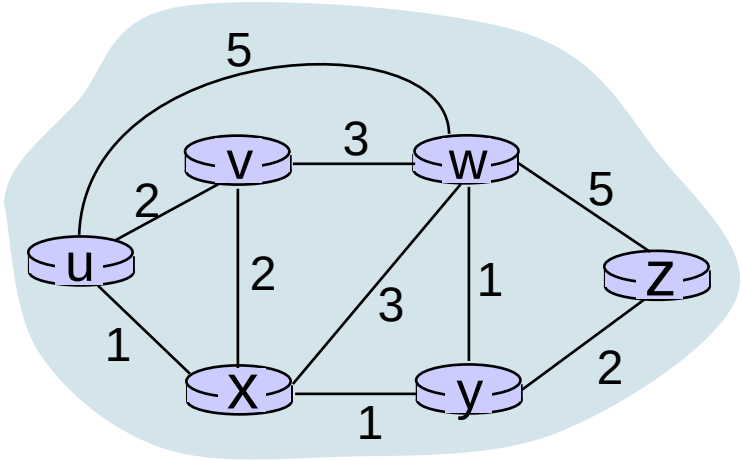
Notes:

- Construct the shortest path tree by tracing the predecessor nodes
- Ties can exist (can be broken arbitrarily)



Dijkstra's Algorithm: Another Example

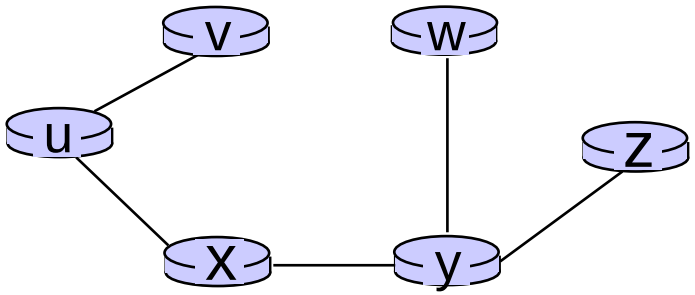
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



*Check out the online interactive exercises for more examples:
http://gaia.cs.umass.edu/kurose_ross/interactive/

Dijkstra's Algorithm: Example (2)

Resulting shortest path tree from u:

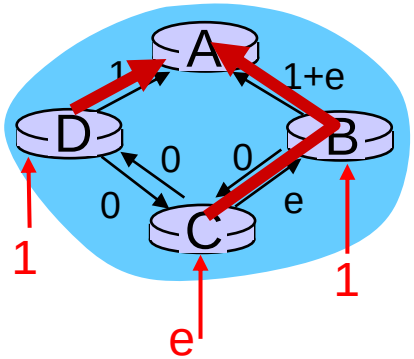


Resulting forwarding table in u:

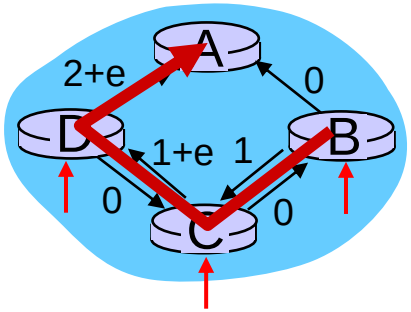
destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Dijkstra's Algorithm, Discussion

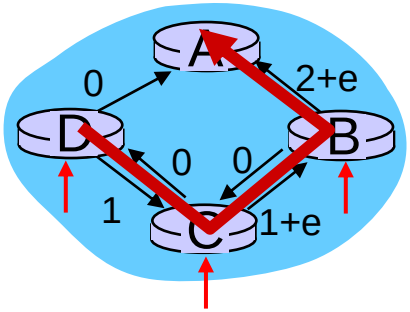
- **Algorithm complexity:** n nodes
  - Each iteration: need to check all nodes, w, not in N
  - n(n+1)/2 comparisons: $O(n^2)$
  - More efficient implementations possible: O(nlogn)
- **Oscillations possible:**
  - E.g., the support link cost equals the amount of carried traffic:



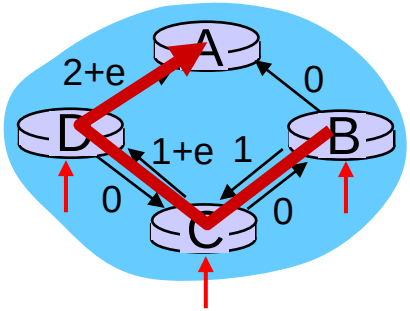
Initially



Given these costs, find new routing… resulting in new costs



Given these costs, find new routing… resulting in new costs



Given these costs, find new routing… resulting in new costs

The End

Distance Vector Algorithm and Bellman-Ford

Distance Vector Algorithm

The Bellman-Ford equation (dynamic programming)

Let

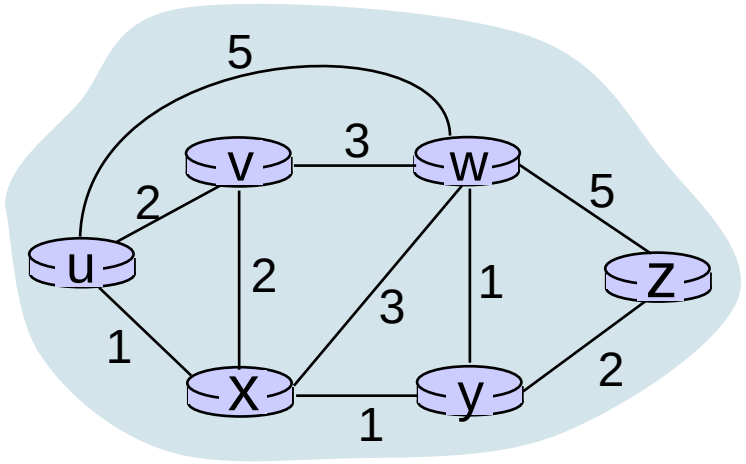
$d_x(y)$:= cost of the least cost path from x to y

Then

$$\mathbf{dx(y) = \min_v \{c(x,v) + d_v(y)\}}$$

- $\min_v$: *min* taken over all neighbors v of x
- $c(x,v)$: cost to neighbor v
- $d_v(y)$: cost from neighbor v to destination y

Bellman-Ford Example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

The node achieving minimum is the next hop in the shortest path, used in forwarding table.

Distance Vector Algorithm

- **$D_x(y)$** = estimate of least cost from x to y
  - x maintains distance vector
    $D_x = [D_x(y): y \epsilon N]$
- Node x:
  - Knows the cost to each neighbor v: **c(x,v)**
  - Maintains its neighbors' distance vectors; for each neighbor v, x maintains
    $D_v = [D_v(y): y \epsilon N]$

Distance Vector Algorithm

Key idea:

- From time to time, each node sends its own distance vector estimate to its neighbors
- When x receives a new DV estimate from a neighbor, it updates its own DV using the B-F equation:

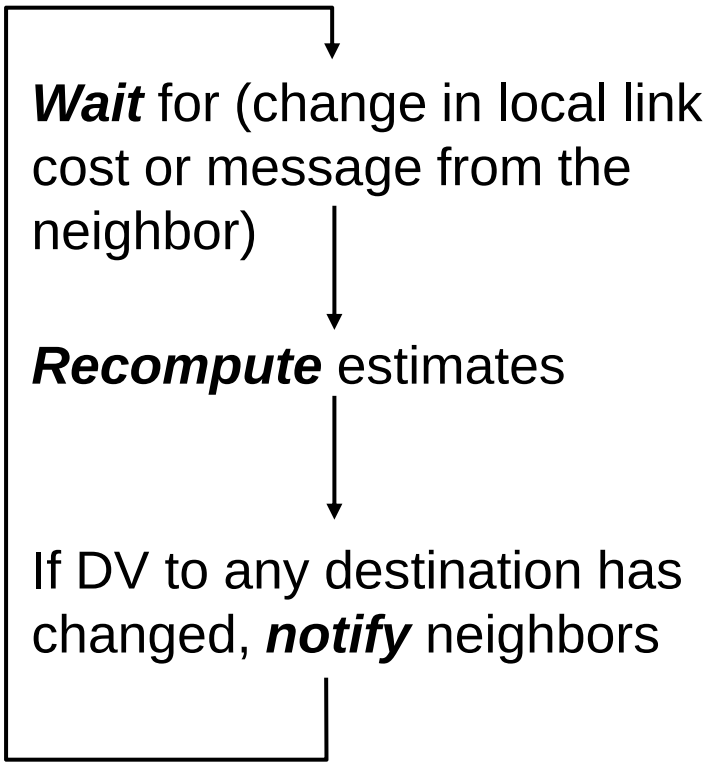
  $$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \text{ for each node } y \in N$$

- Under minor, natural conditions the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$

Distance Vector Algorithm

- **Iterative, asynchronous:** each local iteration caused by:
  - Local link cost change
  - DV update message from its neighbor
- **Distributed:**
  - Each node notifies its neighbors *only* when its DV changes
    - The neighbors then notify their neighbors, if necessary

Each node:



$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node x table

from	cost to x	cost to y	cost to z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

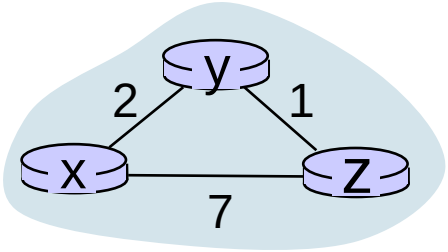
from	cost to x	cost to y	cost to z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

from	cost to x	cost to y	cost to z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from	cost to x	cost to y	cost to z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



time

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node x table

from	cost to x	cost to y	cost to z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from	cost to x	cost to y	cost to z
x	0	2	3
y	2	0	1
z	7	1	0

from	cost to x	cost to y	cost to z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from	cost to x	cost to y	cost to z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from	cost to x	cost to y	cost to z
x	0	2	7
y	2	0	1
z	7	1	0

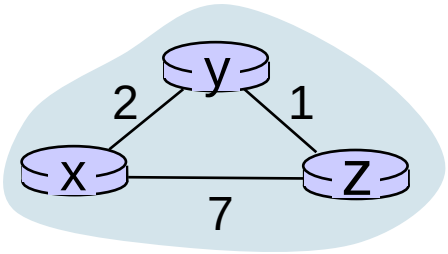
from	cost to x	cost to y	cost to z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from	cost to x	cost to y	cost to z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from	cost to x	cost to y	cost to z
x	0	2	7
y	2	0	1
z	3	1	0

from	cost to x	cost to y	cost to z
x	0	2	3
y	2	0	1
z	3	1	0

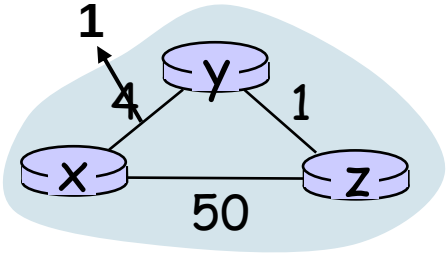


time

Distance Vector: Link Cost Changes

Link cost changes:

- Node detects local link cost change
- Updates routing information, recalculates the distance vector
- If DV changes, notify neighbors



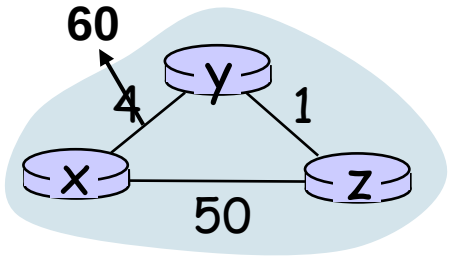
"Good news travels fast"

- t_0 : *y* detects link cost change, updates its DV, informs its neighbors
- t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x*, sends its neighbors its DV
- t_2 : *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Distance Vector: Link Cost Changes

- **Link cost changes:**
  - The node detects local link cost change
  - **Bad news travels slow**—"count-to-infinity" problem!
  - 44 iterations before algorithm stabilizes: see text
- **Poisoned reverse:**
  - If Z routes through Y to get to X:
    - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
  - Will this completely solve the count-to-infinity problem?



Comparison of LS and DV Algorithms

- **Message complexity**
  - **LS:** with n nodes, E links, O(nE) messages sent
  - **DV:** exchange between neighbors only
    - Convergence time varies
- **Speed of convergence**
  - **LS:** $O(n^2)$ algorithm requires O(nE) messages
    - May have oscillations
  - **DV:** convergence time varies
    - May be routing loops
    - Count-to-infinity problem
- **Robustness:** what happens if the router malfunctions?
- **LS:**
  - The node can advertise incorrect ***link*** cost
  - Each node computes only its own table
- **DV:**
  - DV node can advertise incorrect ***path*** cost
  - Each node's table used by others
    - Error propagate through the network

The End

Scaling Routing Protocols

Making Routing Scalable

- Our routing study thus far—idealized
  - All routers identical
  - Network “flat”
- … *not* true in practice

- **Scale:** with billions of destinations
  - Can’t store all destinations in routing tables!
  - The routing table exchange would swamp links!
- **Administrative autonomy**
  - Internet is a network of networks
  - Each network admin may want to control routing in its own network

Internet Approach to Scalable Routing

Aggregate routers into regions known as **“autonomous systems” (AS) (a.k.a. “domains”)**

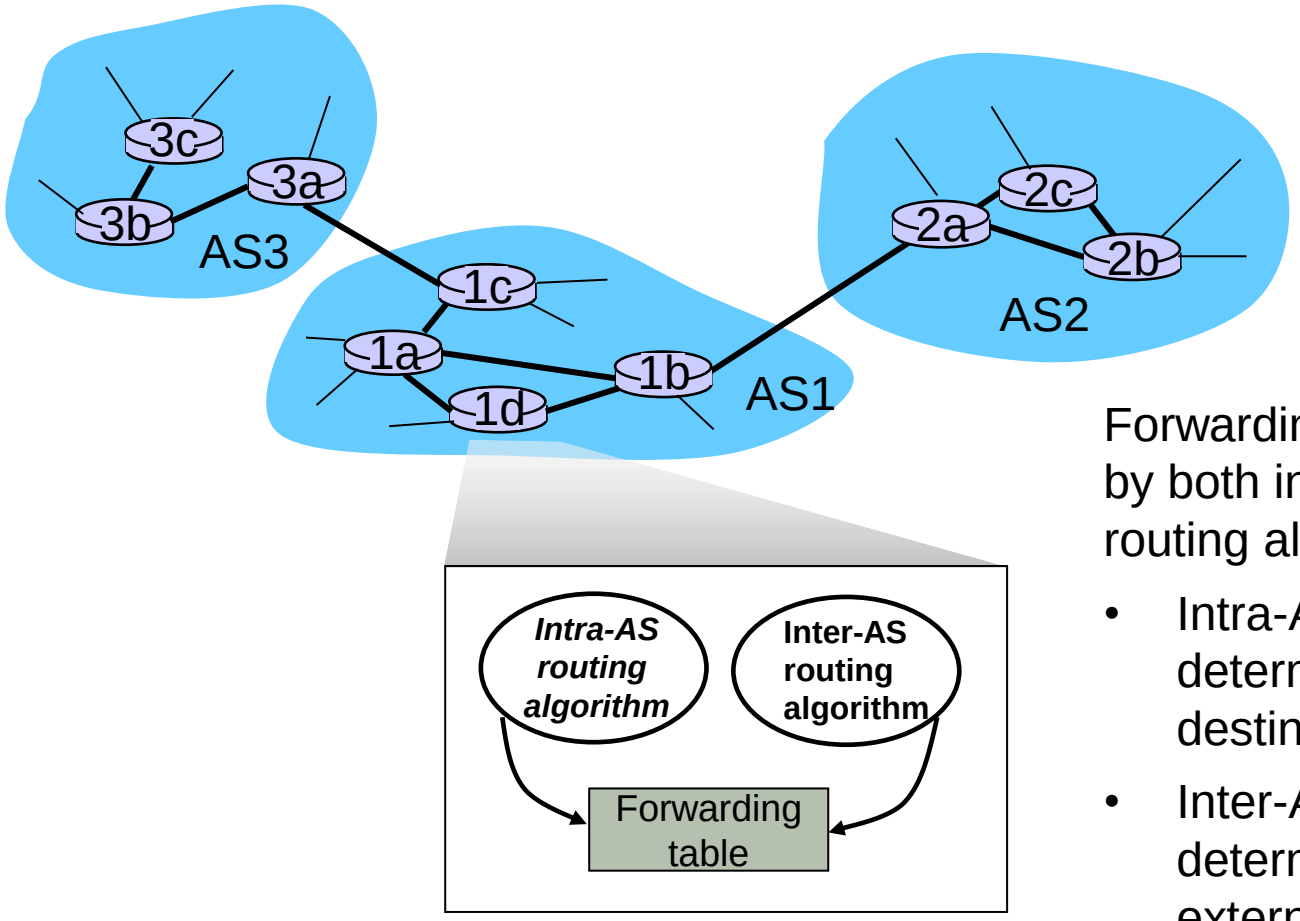
Intra-AS routing

- Routing among hosts, routers in the same AS (“network”)
- All routers in the AS must run the ***same*** intra-domain protocol
- Routers in *different* AS can run *different* intra-domain routing protocol
- Gateway router: at the “edge” of its own AS, has link(s) to router(s) in other ASes

Inter-AS routing

- Routing among ASes
- Gateways perform inter-domain routing (as well as intra-domain routing)

Interconnected ASes



Forwarding table configured by both intra- and inter-AS routing algorithms

- Intra-AS routing determines entries for destinations within the AS
- Inter-AS and intra-AS determine entries for external destinations

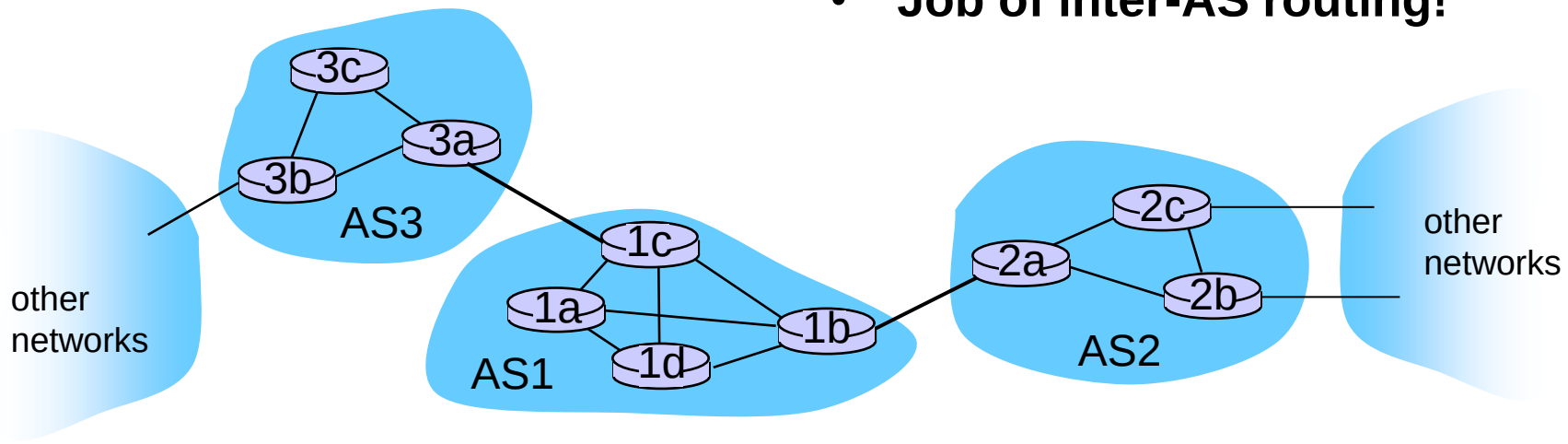
Inter-AS Tasks

- Suppose the router in AS1 receives a datagram destined outside of AS1:
  - The router should forward the packet to the gateway router, but which one?

- **AS1 must:**

1. Learn which destinations are reachable through AS2 and which through AS3
2. Propagate this reachability information to all routers in AS1

- **Job of inter-AS routing!**



Intra-AS Routing

- Also known as **interior gateway protocols (IGP)**
- Most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First (the Intermediate System-Intermediate System, or IS-IS protocol is essentially the same as OSPF)
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary for decades, until 2016)

The End

Open Shortest Path First (OSPF)

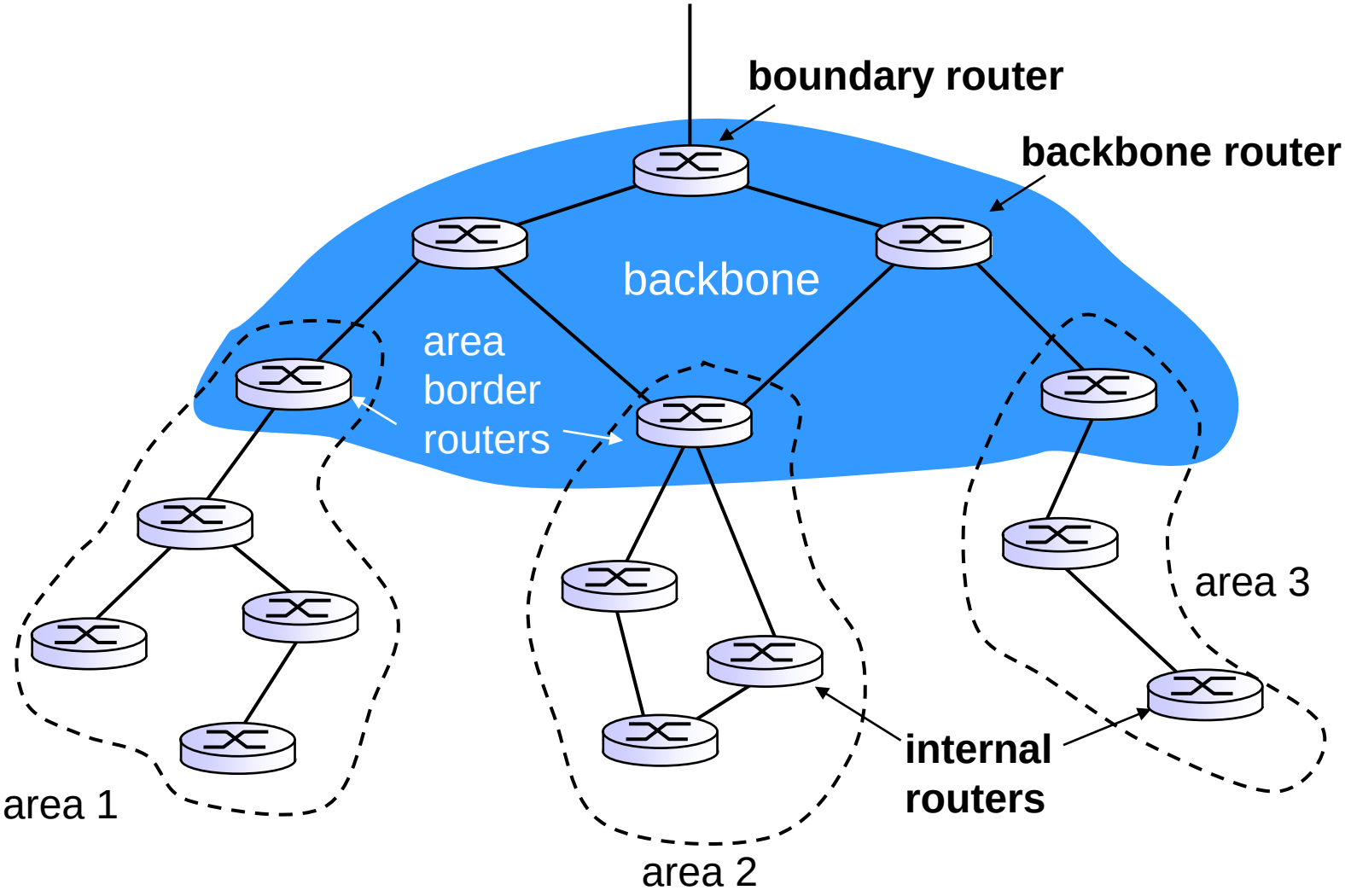
OSPF (Open Shortest Path First)

- “Open”: publicly available
- Uses link-state algorithm
  - Link-state packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra’s algorithm
- The router floods OSPF link-state advertisements to all other routers in the **entire** AS
  - Carried in OSPF messages directly over the IP (rather than TCP or UDP)
  - Link-state: for each attached link
- **IS-IS routing** protocol: nearly identical to OSPF

OSPF “Advanced” Features

- **Security:** all OSPF messages authenticated (to prevent malicious intrusion)
- **Multiple** same-cost **paths** allowed (only one path in RIP)
- For each link, multiple cost metrics for different **TOS** (e.g., satellite link cost set low for best effort TOS; high for real-time TOS)
- Integrated uni- and **multicast** support:
  - Multicast OSPF (MOSPF) uses the same topology data base as OSPF
- **Hierarchical** OSPF in large domains

Hierarchical OSPF



Hierarchical OSPF

- **Two-level hierarchy:** local area, backbone
  1. Link-state advertisements only in area
  2. Each node has detailed area topology; only know direction (shortest path) to nets in other areas
- **Area border routers (ABRs):** “summarize” distances to nets in own area, advertise to other area border routers
- **Backbone routers:** run OSPF routing limited to the backbone
- **Boundary routers:** connect to other ASes

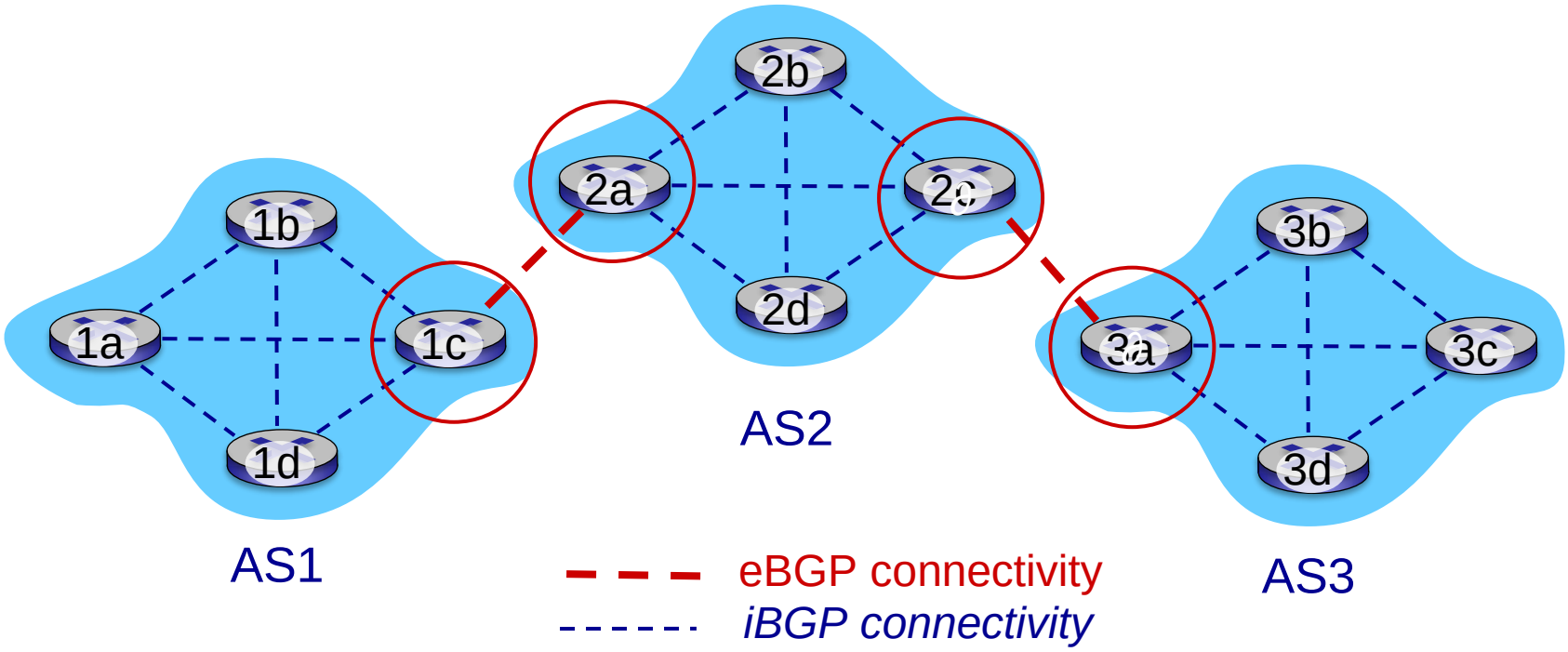
The End

BGP

Internet Inter-AS Routing: BGP

- **BGP (Border Gateway Protocol):** *the* de facto inter-domain routing protocol
  - "Glue that holds the Internet together"
- BGP provides each AS a means to:
  - **eBGP:** obtain subnet reachability information from neighboring ASes
  - **iBGP:** propagate reachability information to all AS-internal routers
  - Determine "good" routes to other networks based on reachability information and ***policy***
- Allows the subnet to advertise its existence to the rest of the Internet: ***"I am here"***

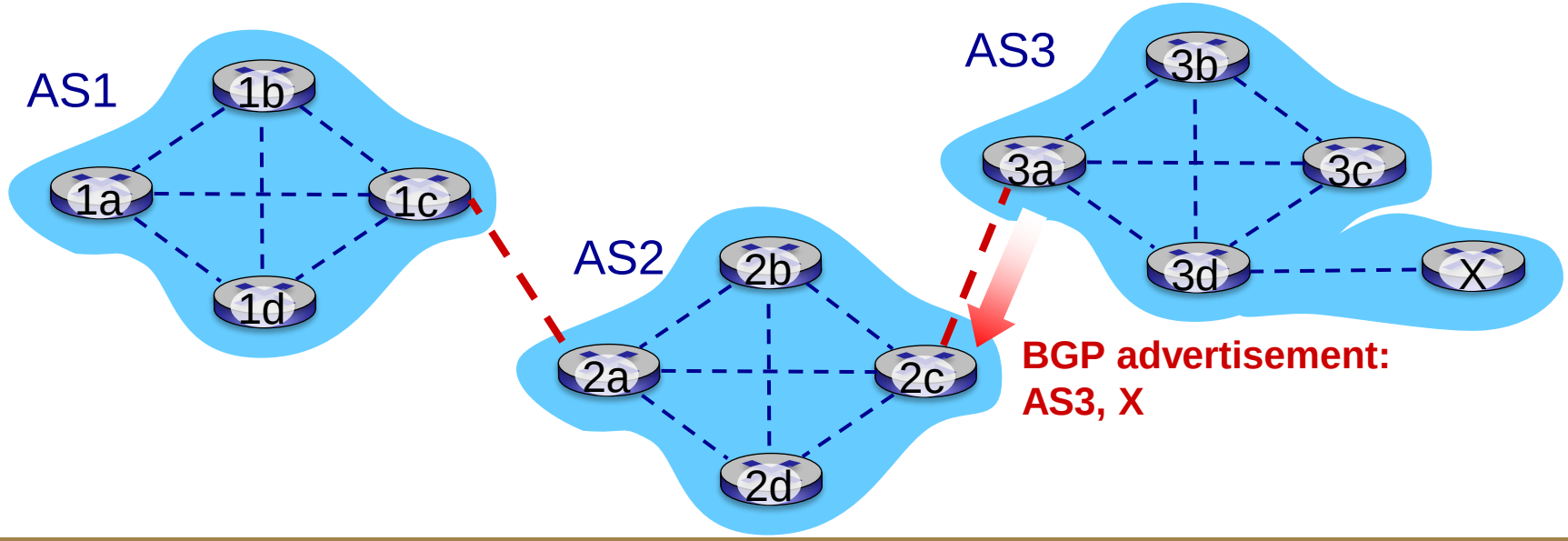
eBGP, iBGP Connections



Gateway routers run both eBGP and iBGP protocols.

BGP Basics

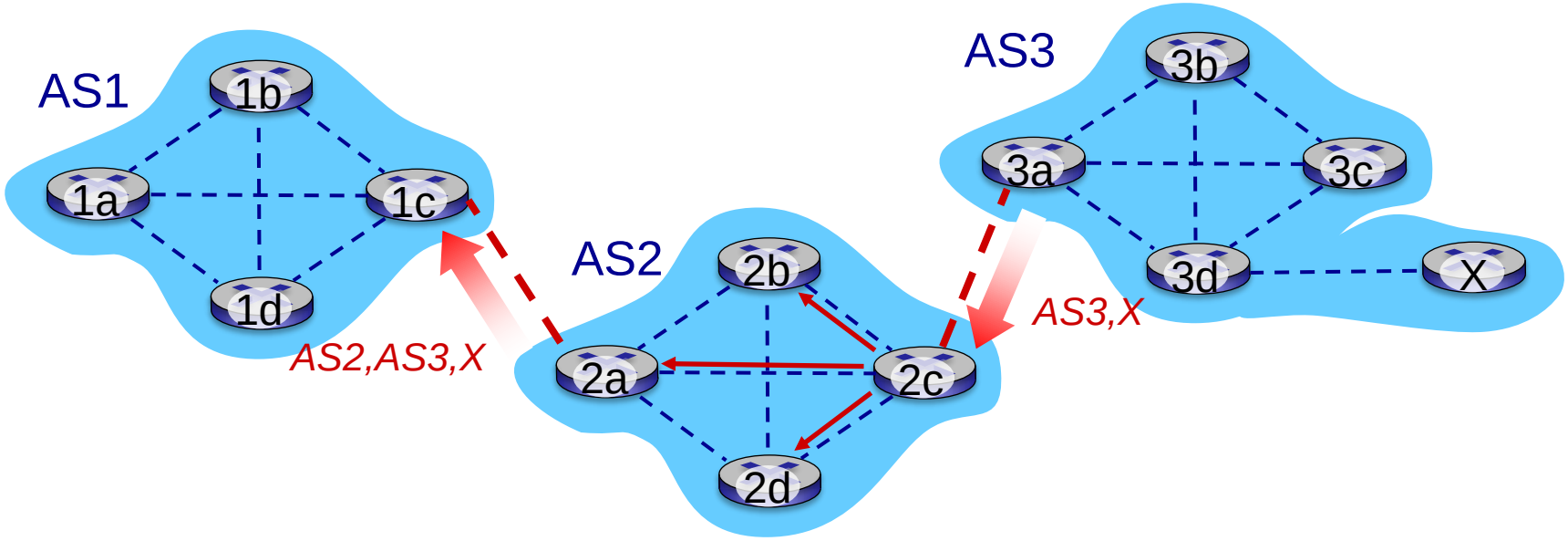
- **BGP session:** two BGP routers ("peers") exchange BGP messages over a semi-permanent TCP connection:
  - Advertising **paths** to different destination network prefixes (BGP is a "path vector" protocol)
- When the AS3 gateway router 3a advertises path **AS3,X** to the AS2 gateway router 2c:
  - AS3 **promises** to AS2 it will forward datagrams towards X



Path Attributes and BGP Routes

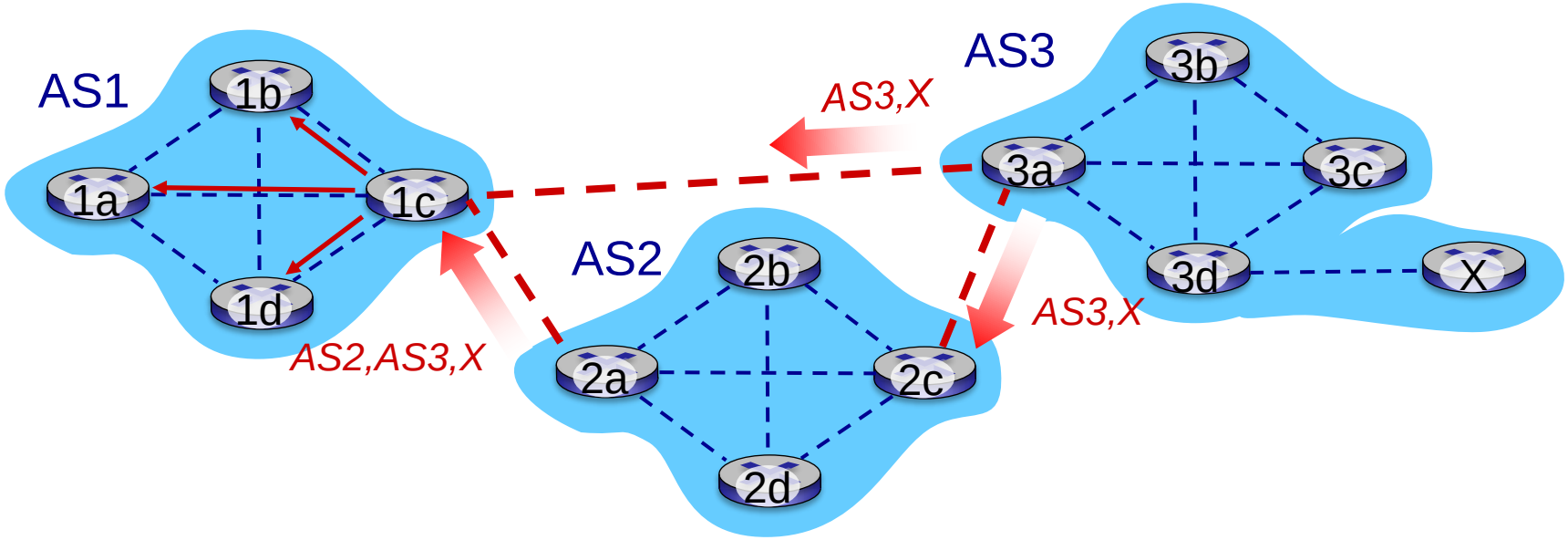
- The advertised prefix includes BGP attributes
  - Prefix + attributes = “route”
- Two important attributes:
  1. ***AS-PATH:*** list of ASes through which the prefix advertisement has passed
  2. ***NEXT-HOP:*** indicates the specific internal-AS router to next-hop AS
- **Policy-based routing:**
  - The gateway receiving the route advertisement uses **import policy** to accept/decline the path (e.g., never route through AS Y).
  - The AS policy also determines whether to **advertise** the path to other neighboring ASes

BGP Path Advertisement



- The AS2 router 2c receives the path advertisement **AS3,X** (via eBGP) from the AS3 router 3a
- Based on AS2 policy, the AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, the AS2 router 2a advertises (via eBGP) path **AS2, AS3, X** to the AS1 router 1c

BGP Path Advertisement



The gateway router may learn about ***multiple*** paths to destination:

- The AS1 gateway router 1c learns path **AS2,AS3,X** from 2a.
- The AS1 gateway router 1c learns path **AS3,X** from 3a.
- Based on policy, the AS1 gateway router 1c chooses path **AS3,X, and advertises the path within AS1 via iBGP.**

BGP Route Selection

The router may learn about more than one route to destination AS, select a route based on:

1. Local preference value attribute: policy decision
2. The shortest AS-PATH
3. The closest NEXT-HOP router: hot potato routing
4. Additional criteria

Why Different Intra-, Inter-AS Routing?

- **Policy:**
  - Inter-AS: the admin wants control over how its traffic is routed, who routes through its net
  - Intra-AS: single admin, so no policy decisions needed
- **Scale:**
  - Hierarchical routing saves table size, reduced update traffic
- **Performance:**
  - Intra-AS: can focus on performance
  - Inter-AS: policy may dominate over-performance

The End