COMP2001J Computer Networks

Lecture 8 – Network Layer
(routing)

Dr. Shen WANG (王燊)

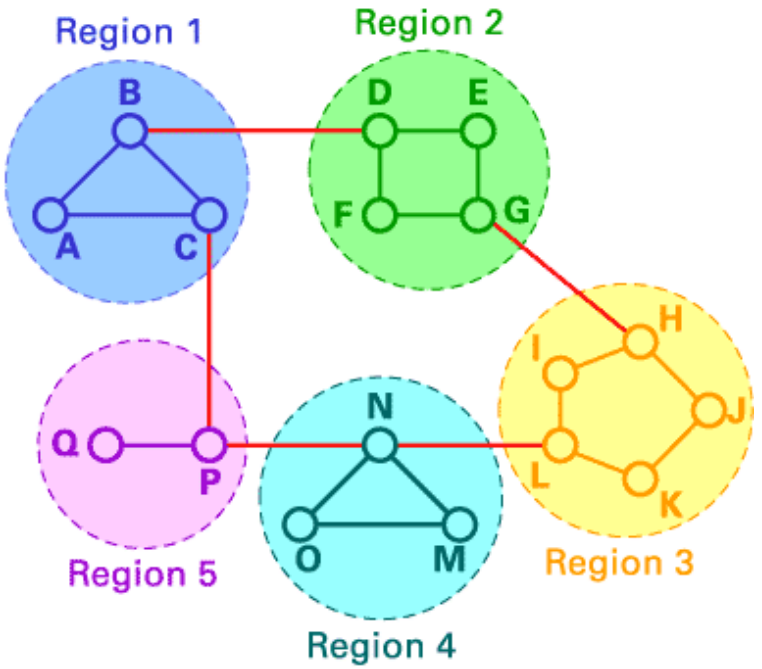
shen.wang@ucd.ie

Lab Exam

- 16th April (Tuesday), 9:55am, 2nd floor, room 5, Zhixing Building
- Submit a *.pkt file and a *.pdf file
  - "*" = "name_UCDnumber"
- You **have to** attend!
- You **can** check slides and lab notes.
- You **can NOT** talk and check the Internet.
- For preparation, to save time during exam:
  - Review all slides so far, including today's. You need to answer questions in some steps of the Lab Exam in the *.pdf file, which requires the correct understanding of concepts.
  - Re-do all 4 Packet Tracer Labs. To confirm you understand why it works.

Outline

- Introduction
- Algorithms
  - Distance vector
  - Link state
- Hierarchical routing
- Routing protocols
  - RIP
  - OSPF
  - BGP



Introduction

- How does a router know which line it should forward to, given the destination network of a received packet?
  - Forwarding table. But how is this table generated?
- Network layer is responsible for finding an optimal path for delivering each packet.
- You might use shortest path finding algorithms learned before, but routers in the network is a very distributed environment. They need to work together to:
  - Know the network, which is changing dynamically
  - To find out optimal paths

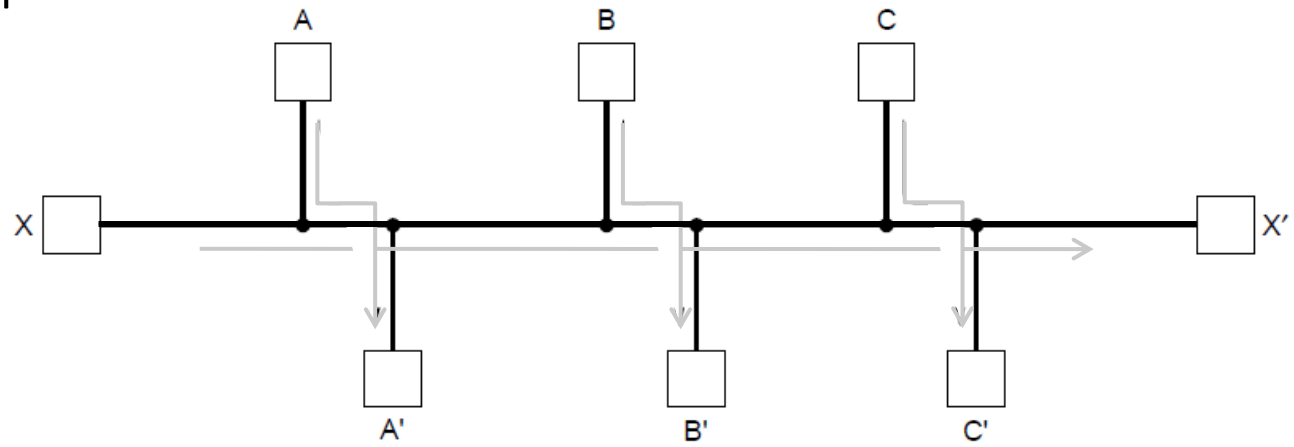
Routing and Forwarding

- Routing is the process of discovering network paths
  - Model the network as a graph of nodes and links
  - Decide what to optimize (e.g., fairness vs efficiency)
  - Update routes for changes in topology (e.g., failures)
    - We focus on **adaptive** routing (or dynamic routing, rather that static routing that does not response to changes or failures) schemes that update routes in response to failures.

- Forwarding is the sending of packets along a path

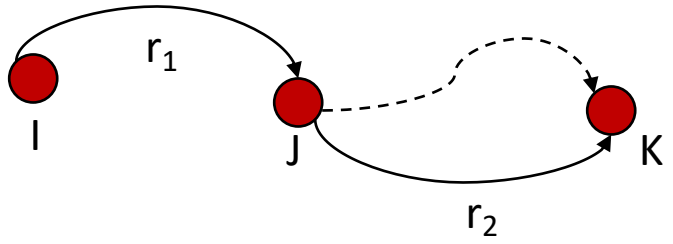
Fairness v.s. Efficiency

- The traffic demands are A->A', B->B', C->C' and X->X'.
- What would be fair?
  - For each flow they get the same amount of bandwidth. If all network links have unit capacity then we would give each flow ½ a unit of capacity. The total network traffic is then 2 units.
- What would be efficient?
  - If we gave the X->X' flow no bandwidth then we could give each of the other three flows 1 unit. The total network traffic is then 3 units. So it is more efficient, but it is not fair. So we will have to decide what we want to optimize.



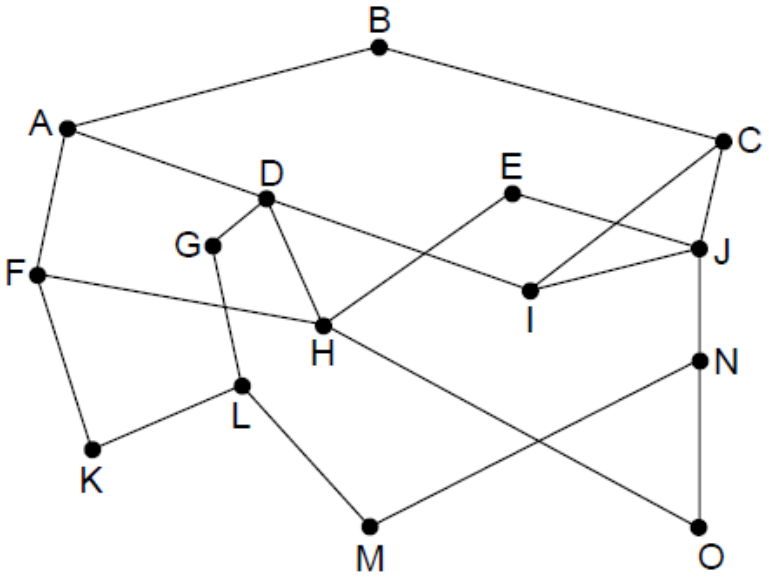
The Optimality Principle

- If router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route.
- Proof by contradiction:
  - call the part of the route from I to J: r_1 and the rest of the route r_2 .
  - If a route better than r_2 existed from J to K, it could be concatenated with r_1 to improve the route from I to K, contradicting our statement that $r_1 + r_2$ is optimal.

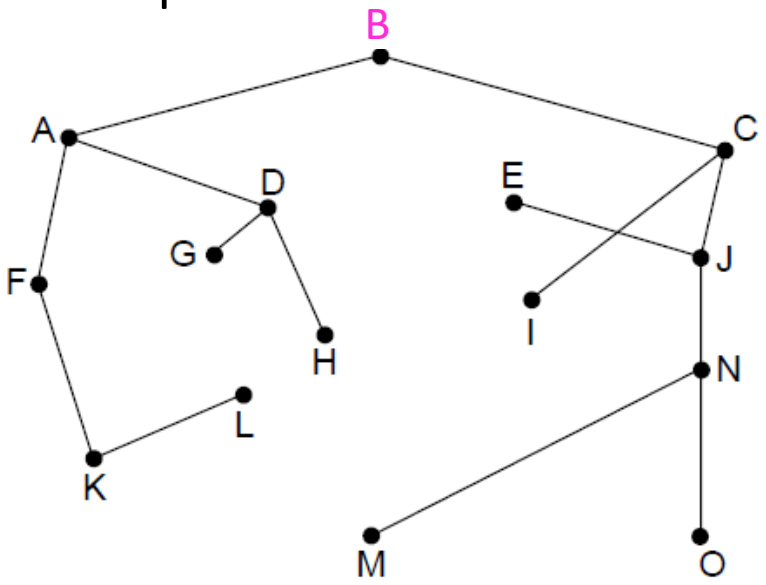


Sink Tree

- Each portion of a best path is also a best path; the union of them to a router is a tree called the sink tree
  - It is different from “the minimum spanning tree”.
  - Best means fewest hops in the example



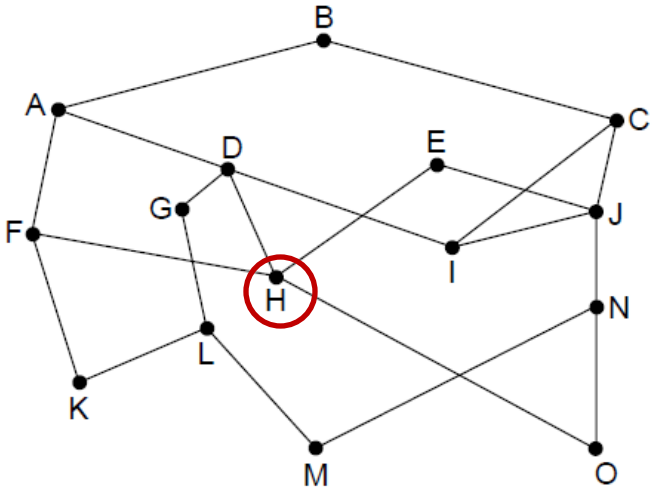
Network



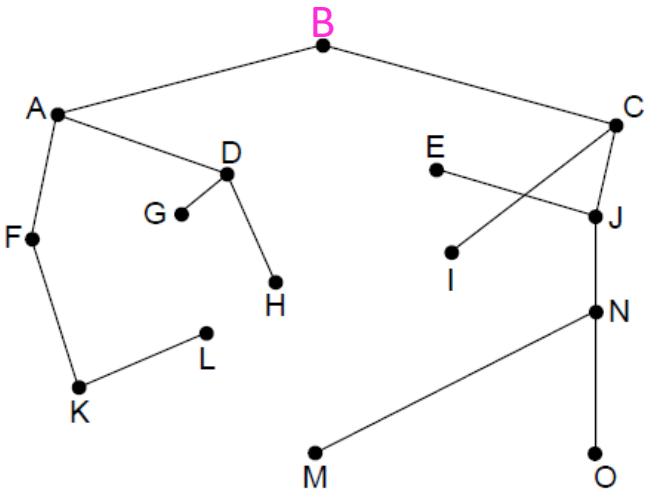
Sink tree of best paths to router B

Sink Tree -> DAG

- If there are multiple paths that are equally good, then one best path from one node to another is chosen at random. This is simple and useful as there is a single route from each router to each destination.
- For example, H can be reached in 3 hops via H-D-A-B as shown, or by H-F-A-B (not shown).
- If all equally best paths are kept then their union is a DAG (directed acyclic graph).



Network



Sink tree of best paths to router B

Shortest Path Algorithm

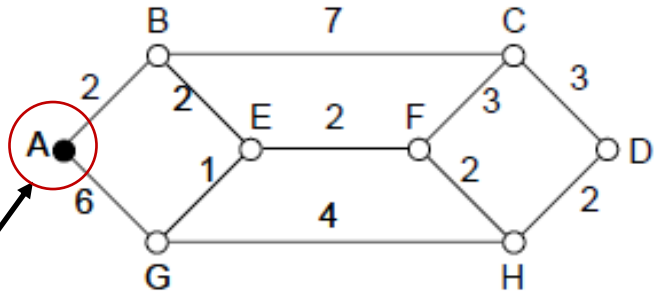
- Dijkstra's algorithm computes a sink tree on the graph:
  - Each link is assigned a non-negative weight/distance
  - Shortest path is the one with lowest total weight
  - Using weights of 1 gives paths with fewest hops

- Algorithm:
  - Start with sink (destination), set distance at other nodes to infinity
  - Relax distance to all other "tentative" nodes
  - Pick the lowest distance node (then becomes "permanent" node), add it to "sink tree"
  - Repeat until all nodes are in the sink tree

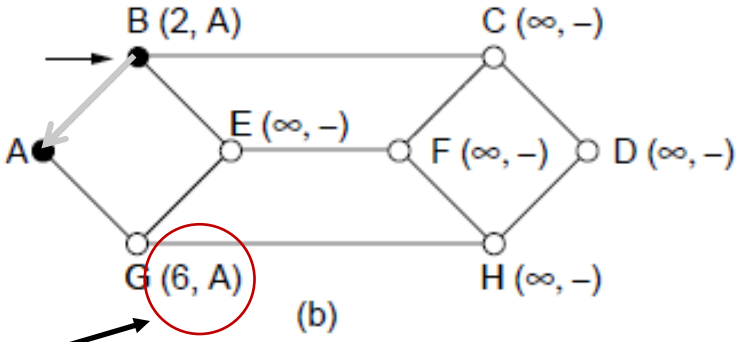
Weights

- The notion of weight generalizes distance to other cost metrics.
  - Setting weights to be 1 gives paths with fewest hops.
  - Setting weight to be distance gives paths that are shortest or lowest delay.
  - Setting weight to be lower for higher capacity links favours higher capacity paths.

A network and first five steps in computing the shortest paths **from A to D**. Arrows show the sink tree so far.

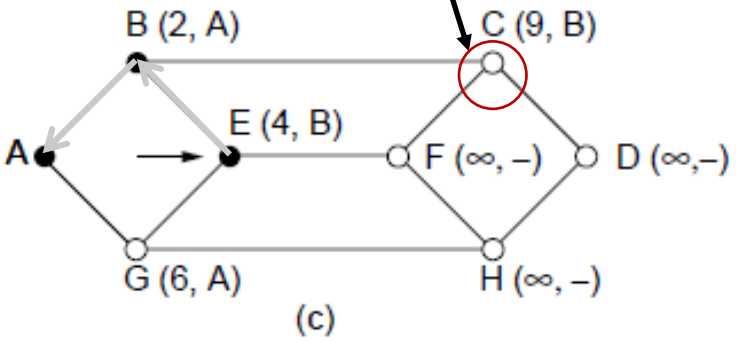


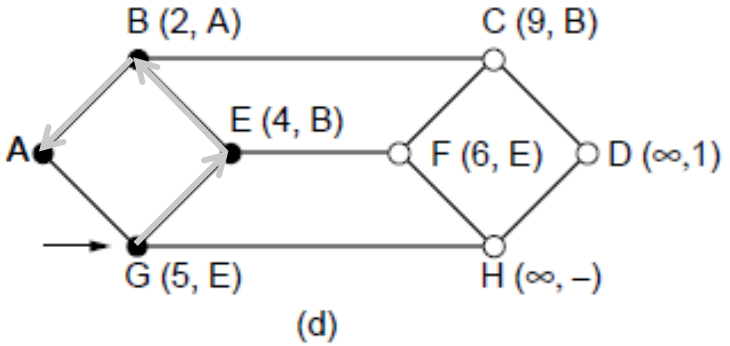
Permanent node

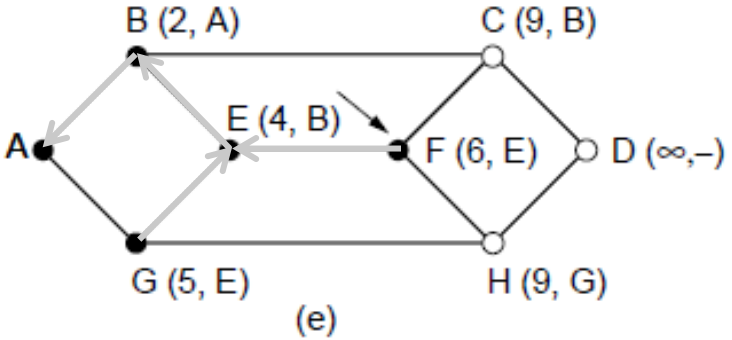


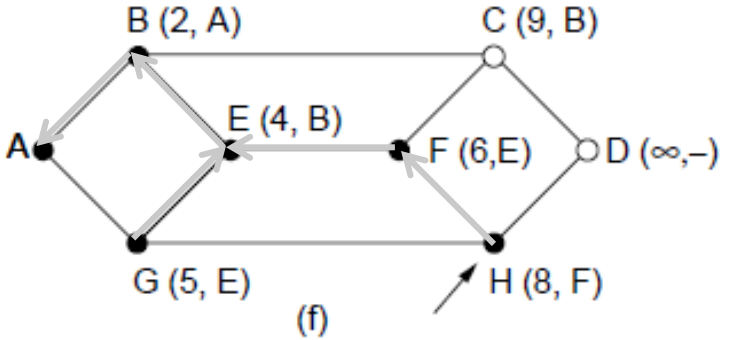
Tentative node

(distance to starting node, predecessor)









Shortest Path Algorithm (s->t)

. . .

Searching in a reversed way from t->s

```
for (p = &state[0]; p < &state[n]; p++) {
      p->predecessor = –1;
      p->length = INFINITY;
      p->label = tentative;
}
state[t].length = 0;  state[t].label = permanent;
k = t;
do {
      for (i = 0; i < n; i++)
            if (dist[k][i] != 0 && state[i].label == tentative) {
                  if (state[k].length + dist[k][i] < state[i].length) {
                        state[i].predecessor = k;
                        state[i].length = state[k].length + dist[k][i];
                  }
            }
```

. . .

Start with the sink (k: destination node), all other nodes are unreachable

Relaxation step. Lower distance to nodes linked to newest member of the sink tree

Shortest Path Algorithm

```
    k = 0; min = INFINITY;
    for (i = 0; i < n; i++)
        if (state[i].label == tentative && state[i].length < min) {
            min = state[i].length;
            k = i;
        }
    state[k].label = permanent;
} while (k != s);
```

Find the lowest distance, add it to the sink tree, and repeat until done

- This code keeps going until it reaches the source node s; if it continued until there were no tentative nodes then it would have found the entire sink tree (to all other nodes).
- The final output, the predecessor links, don't need to be reversed, if we search from t -> s instead of from s -> t. As the link has the same cost in each direction so paths are symmetrical (or the same in each direction).

Distance Vector Routing

- Distance vector is a distributed routing algorithm
  - Shortest path computation is split across nodes
  - Based on the **Bellman–Ford algorithm**

- Algorithm:
  - Each node knows distance (weight value) of links to its neighbors
  - Each node advertises distance vector to all neighbors. This distance vector shows the lowest known distances to all other routers (full routing table! Costly!)
  - Each node uses received vectors to update its own
  - Repeat periodically

Distance Vector Routing Table

- In distance vector routing, each router maintains a routing table indexed by, and containing one entry for each router in the network.

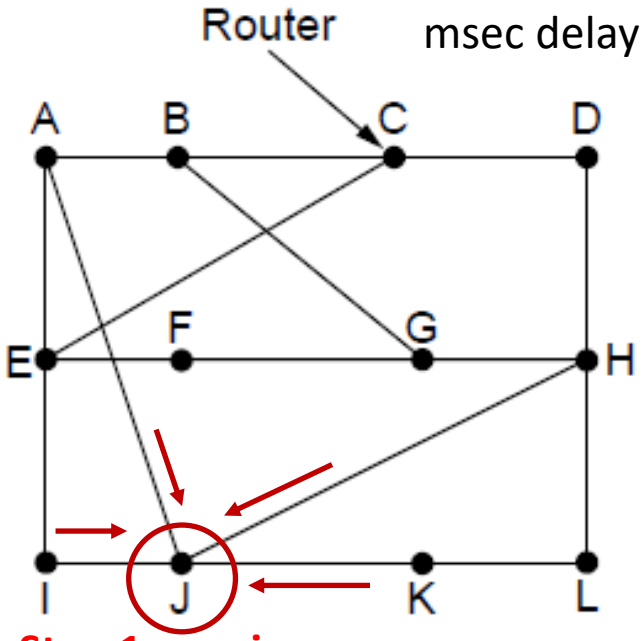
- This entry has two parts:
  - the preferred outgoing line to use for that destination
  - an estimate of the distance to that destination.
    - The distance might be measured as the number of hops or using another metric, as we discussed for computing shortest paths.

Convergence

- The settling of routes to best paths across the network is called **convergence**.

- How long will it take to converge? The diameter of the network.
  - Diameter: the longest shortest paths of all pairs in a network.

- Let's see an example, how a router J:
  - Receives distance vectors from all its neighbors
  - Update its own distance vector accordingly using its latest estimation of all links to its neighbors
  - Compute the best forwarding line and distance for packets targeting on G



A claims to have a 12-msec delay to B, a 25-msec delay to C, a 40-msec delay to D, etc.

New estimated delay from J

To	A	I	H	K	Line
A	0	24	20	21	8 A
B	12	36	31	28	20 A
C	25	18	19	36	28 I
D	40	27	8	24	20 H
E	14	7	30	22	17 I
F	23	20	19	40	30 I
G	18	31	6	31	18 H
H	17	20	0	19	12 H
I	21	0	14	22	10 I
J	9	11	7	10	0 –
K	24	22	22	0	6 K
L	29	33	9	9	15 K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	

Step 1, receives distance vectors Network

G: (A: 26, I:41, **H:18**, K:37)

Step 3, computes best outgoing line and distance given target G

New vector for J

Step 2, updates own distance vector

J has measured or estimated its delay to its neighbors, A, I, H, and K, as 8, 10, 12, and 6 msec, respectively

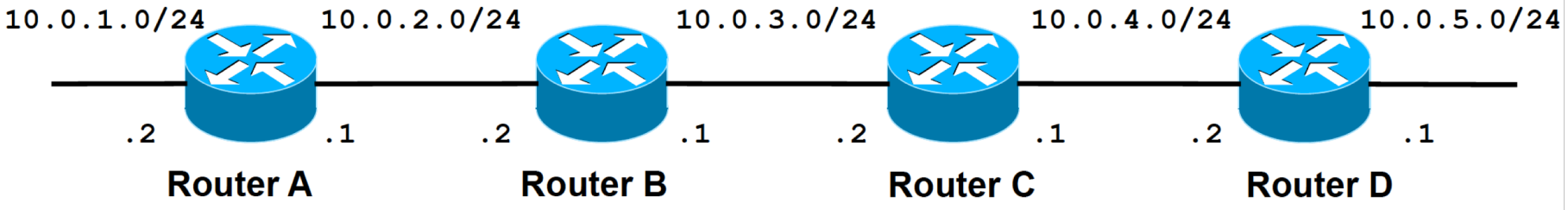
Delay vectors received at J from Neighbors A, I, H and K

Distance Vector Routing

- How J computes its new route to router G?
- It knows that J->A: 8 msec, and furthermore A claims that A->G: 18 msec, so J knows that J->A->G: 26 msec Similarly, it computes the delay to G via
  - I, 41 (31+10) msec
  - **H**, **18** (6 + 12) msec
  - K, 37 (31 + 6) msec
  - The best of these values is 18, so it makes an entry in its routing table that the delay to G is 18 msec and that the route to use is via H.
- The same calculation is performed for all the other destinations.

Example

Assume: - link cost is 1, i.e., c(v,w) = 1
- all updates, updates occur simultaneously
- Initially, each router only knows the cost of connected interfaces



Router A

Net	via	cost
t=0:		
10.0.1.0	-	0
10.0.2.0	-	0
t=1:		
10.0.1.0	-	0
10.0.2.0	-	0
10.0.3.0	10.0.2.2	1
t=2:		
10.0.1.0	-	0
10.0.2.0	-	0
10.0.3.0	10.0.2.2	1
10.0.4.0	10.0.2.2	2

Router B

Net	via	cost
t=0:		
10.0.2.0	-	0
10.0.3.0	-	0
t=1:		
10.0.1.0	10.0.2.1	1
10.0.2.0	-	0
10.0.3.0	-	0
10.0.4.0	10.0.3.2	1
t=2:		
10.0.1.0	10.0.2.1	1
10.0.2.0	-	0
10.0.3.0	-	0
10.0.4.0	10.0.3.2	1
10.0.5.0	10.0.3.2	2

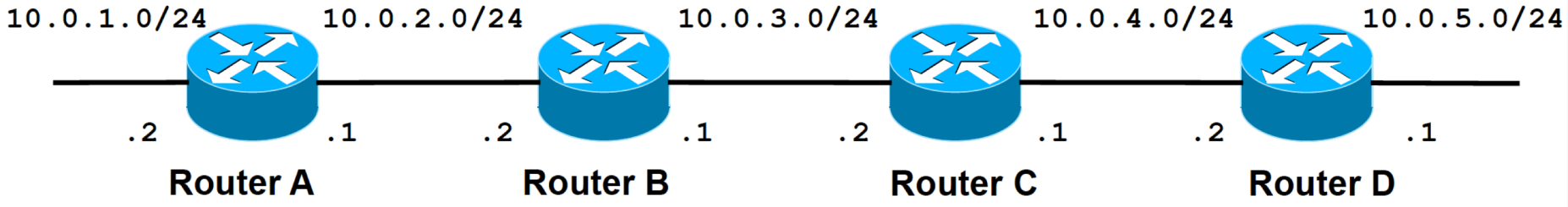
Router C

Net	via	cost
t=0:		
10.0.3.0	-	0
10.0.4.0	-	0
t=1:		
10.0.2.0	10.0.3.1	1
10.0.3.0	-	0
10.0.4.0	-	0
10.0.5.0	10.0.4.2	1
t=2:		
10.0.1.0	10.0.3.1	2
10.0.2.0	10.0.3.1	1
10.0.3.0	-	0
10.0.4.0	-	0
10.0.5.0	10.0.4.2	1

Router D

Net	via	cost
t=0:		
10.0.4.0	-	0
10.0.5.0	-	0
t=1:		
10.0.3.0	10.0.4.1	1
10.0.4.0	-	0
10.0.5.0	-	0
t=2:		
10.0.2.0	10.0.4.1	2
10.0.3.0	10.0.4.1	1
10.0.4.0	-	0
10.0.5.0	-	0

Example



Net	via	cost
t=2:		
10.0.1.0	*–*	*0*
10.0.2.0	*–*	*0*
10.0.3.0	*10.0.2.2*	*1*
10.0.4.0	*10.0.2.2*	*2*
t=3:		
10.0.1.0	–	0
10.0.2.0	–	0
10.0.3.0	10.0.2.2	1
10.0.4.0	10.0.2.2	2
10.0.5.0	10.0.2.2	3

Net	via	cost
t=2:		
10.0.1.0	*10.0.2.1*	*1*
10.0.2.0	*–*	*0*
10.0.3.0	*–*	*0*
10.0.4.0	*10.0.3.2*	*1*
10.0.5.0	*10.0.3.2*	*2*
t=3:		
10.0.1.0	10.0.2.1	1
10.0.2.0	–	0
10.0.3.0	–	0
10.0.4.0	10.0.3.2	1
10.0.5.0	10.0.3.2	2

Net	via	cost
t=2:		
10.0.1.0	*10.0.3.1*	*2*
10.0.2.0	*10.0.3.1*	*1*
10.0.3.0	*–*	*0*
10.0.4.0	*–*	*0*
10.0.5.0	*10.0.4.2*	*1*
t=3:		
10.0.1.0	10.0.3.1	2
10.0.2.0	10.0.3.1	1
10.0.3.0	–	0
10.0.4.0	–	0
10.0.5.0	10.0.4.2	1

Net	via	cost
t=2:		
10.0.2.0	*10.0.4.1*	*2*
10.0.3.0	*10.0.4.1*	*1*
10.0.4.0	*–*	*0*
10.0.5.0	*–*	*0*
t=3:		
10.0.1.0	10.0.4.1	3
10.0.2.0	10.0.4.1	2
10.0.3.0	10.0.4.1	1
10.0.4.0	–	0
10.0.5.0	–	0

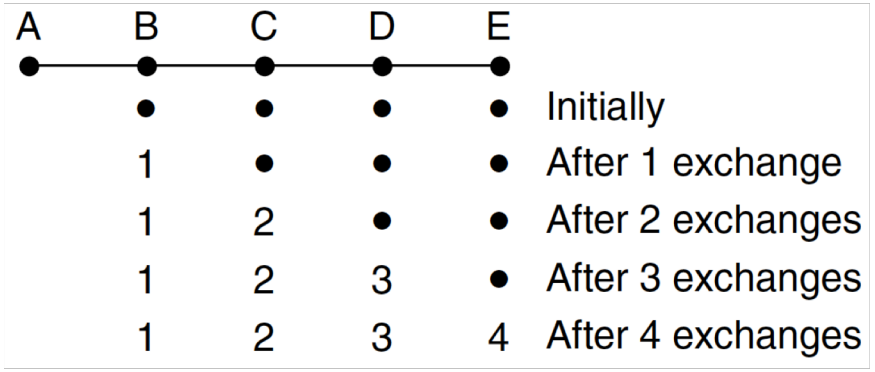
Now, routing tables have converged!

Characteristics of Distance Vector Routing

- Periodic Updates
  - Updates to the routing tables are sent at the end of a certain time period. A typical value is 90 seconds.
- Triggered Updates
  - If a metric changes on a link, a router immediately sends out an update without waiting for the end of the update period.
- Full Routing Table Update
  - Most distance vector routing protocol send their neighbours the entire routing table (not only entries which change).
- Route invalidation timers
  - Routing table entries are invalid if they are not refreshed. A typical value is to invalidate an entry if no update is received after 3-6 update periods.

“Fast Good News”

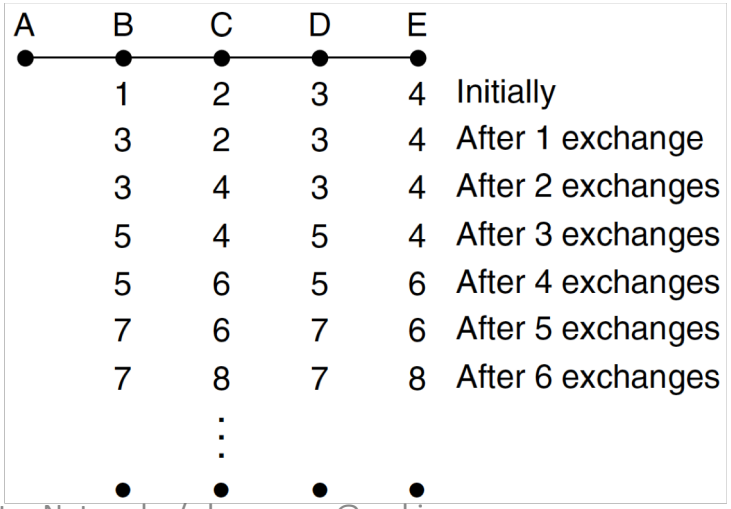
- Consider a five-node (linear) network, where the distance is the number of hops. Suppose A is down initially and all the other routers know this. When A comes up, the other routers learn about it via the vector exchanges.
  - At the first exchange, B learns that its left-hand neighbour has zero delay to A. B now makes an entry in its routing table indicating that A is one hop away to the left. All the other routers still think that A is down.
  - On the next exchange, C learns that B has a path of length 1 to A, so it updates its routing table to indicate a path of length 2, but D and E do not hear the good news until later.
- In a network whose longest path is of length N hops, within N exchanges everyone will know about newly revived links and routers.



“Slow Bad News”

- Consider that all the links and routers are initially up. Suddenly, A goes down.
  - At the first vector exchange, B does not hear anything from A. But, C says ‘‘Do not worry; I have a path to A of length 2.’’ B thinks it can reach A via C, with a path length of 3. D and E do not update their entries for A.
  - On the second exchange, C notices that each of its neighbours claims to have a path to A of length 3. It picks one of them at random and makes its new distance to A 4.
- All routers gradually work their way up to infinity, as no router ever has a value more than one higher than the minimum of all its neighbours.
- The number of exchanges required depends on the numerical value used for infinity (usually set “infinity” to the longest path plus 1).

This problem is known as the **count-to-infinity problem.**

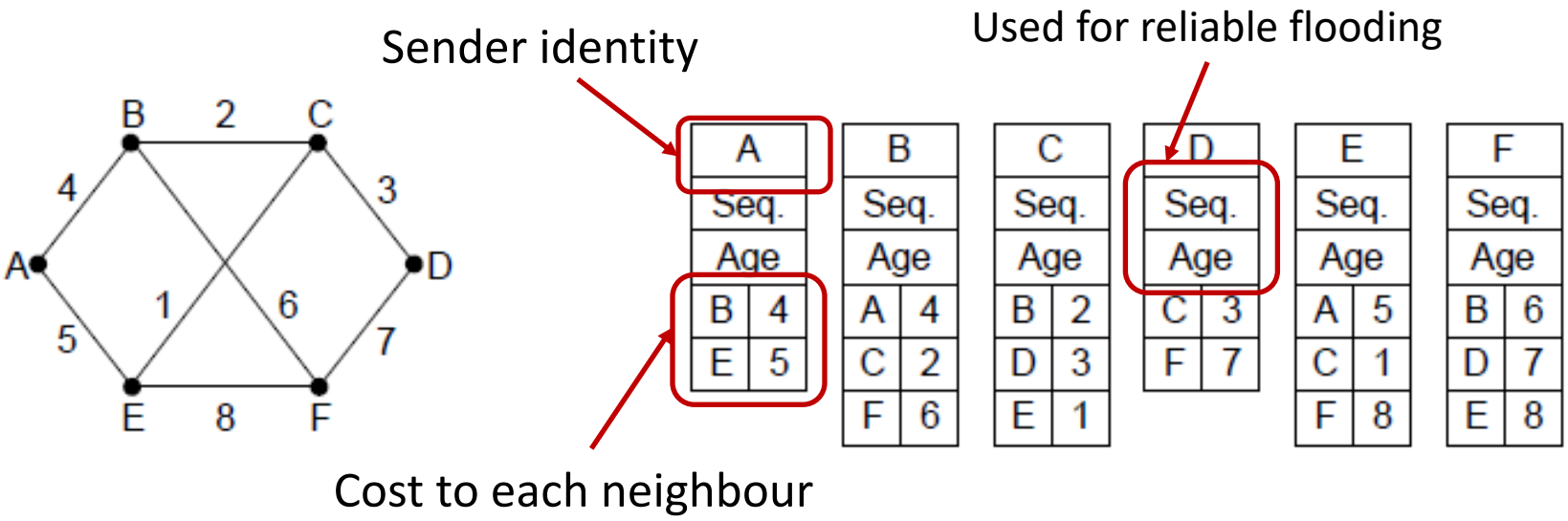


Link State Routing

- Link state is an alternative to distance vector
  - More computation but simpler dynamics
  - Widely used in the Internet (OSPF, ISIS)
- Algorithm:
  - Each node **floods** information about its neighbors in LSPs (Link State Packets) -> all nodes learn the full network graph
  - Each node runs Dijkstra's algorithm to compute the path to take for each destination

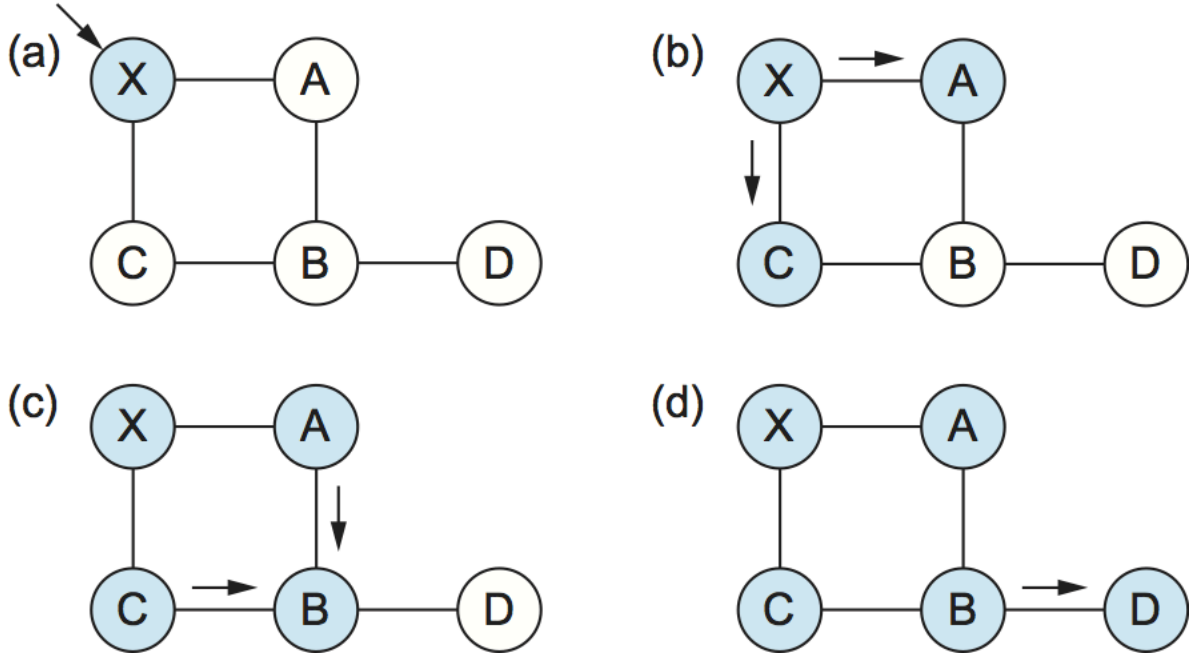
LSPs

- LSP (Link State Packet) for a node lists neighbors and weights of links to reach them



Flooding

- A simple method to send a packet to all network nodes. Each node floods a new packet received on an incoming link by sending it out all of the other links. (recall the learning process for Bridge)



Flooding

- Flooding generates infinite numbers of duplicate packets!
- Nodes need to keep track of flooded LSPs to avoid sending them out a second time, and only keep the latest LSPs.
  - Not a good idea to use a hop limit as it can still blow up exponentially
- Note that flooding doesn't actually find any routes that can be reused.

Flooding – Advantages

- Flooding ensures that a packet is delivered to every node in the network.
- Flooding is tremendously robust. Even if large numbers of routers are down, flooding will find a path if one exists, to get a packet to its destination
- Flooding also requires little setup. The routers only need to know their neighbours.

Reliable Flooding

- Seq. number and age are used for reliable flooding
- To guard against errors on the links, new LSPs are acknowledged on the lines they are received and sent on all other lines
- Example shows the LSP database at router B. B has links to A, C and F in the network.
  - B received LSPs from A, C, and F directly and so acknowledged A, C, and F respectively and sent that LSP on both other links.
  - But B received E and D on two links, so it acknowledged both and sent only on the third link.

			Send flags			ACK flags			
Source	Seq.	Age	A	C	F	A	C	F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Why Hierarchical Routing?

- As networks grow in size, the router routing tables grow proportionally.
- Not only is router memory consumed by ever-increasing tables, but more CPU time is needed to scan them and more bandwidth is needed to send status reports about them.
- At a certain point, the network may grow to the point where it is no longer feasible for every router to have an entry for every other router, so the routing will have to be done hierarchically.

Hierarchical Routing

- Hierarchical routing resembles to reach a given telephone
  - first head towards the right country, then the right city in the country, then the phone in the city.

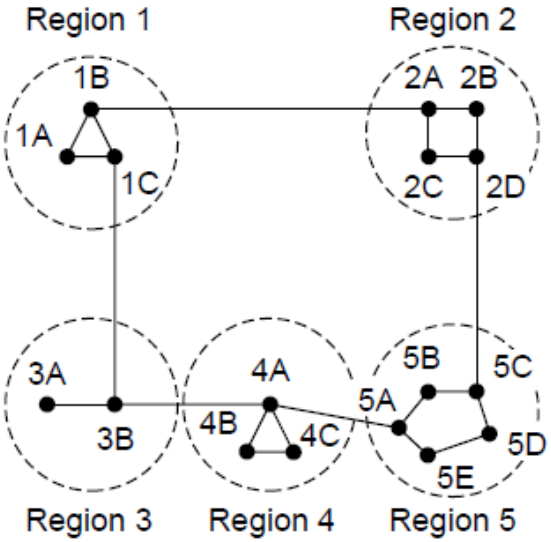
- Each node keeps only one entry per region for other regions, plus an entry for all nodes in the local region.

- The advantages are smaller routing tables, smaller routing computations to run at nodes, and fewer/smaller messages to send to describe the network.

Hierarchical Routing

Routing in a two-level hierarchy with five regions

- reduces 1A's routing table from 17 to 7
- but could slightly increase some paths



Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Best choice to reach nodes in 5 except for 5C

the best route from 1A to 5C is via region 2, but with hierarchical routing all traffic to region 5 goes via region 3, because that is better for most destinations in region 5

RIP(Routing Information Protocol)

- RIP is one of the oldest distance-vector routing protocols which employ the hop count as a routing metric.
- RIP prevents routing loops by implementing a limit on the number of hops (15) allowed in a path from source to destination. This also limits the size of networks that RIP can support.
- In most networking environments, RIP is NOT the preferred choice for routing as its time to converge and scalability are poor. However, it is easy to configure, because RIP does not require any parameters unlike other protocols.
- RIP uses the User Datagram Protocol (UDP) as its transport protocol, and is assigned the reserved port number 520.

OSPF (Open Shortest Path First)

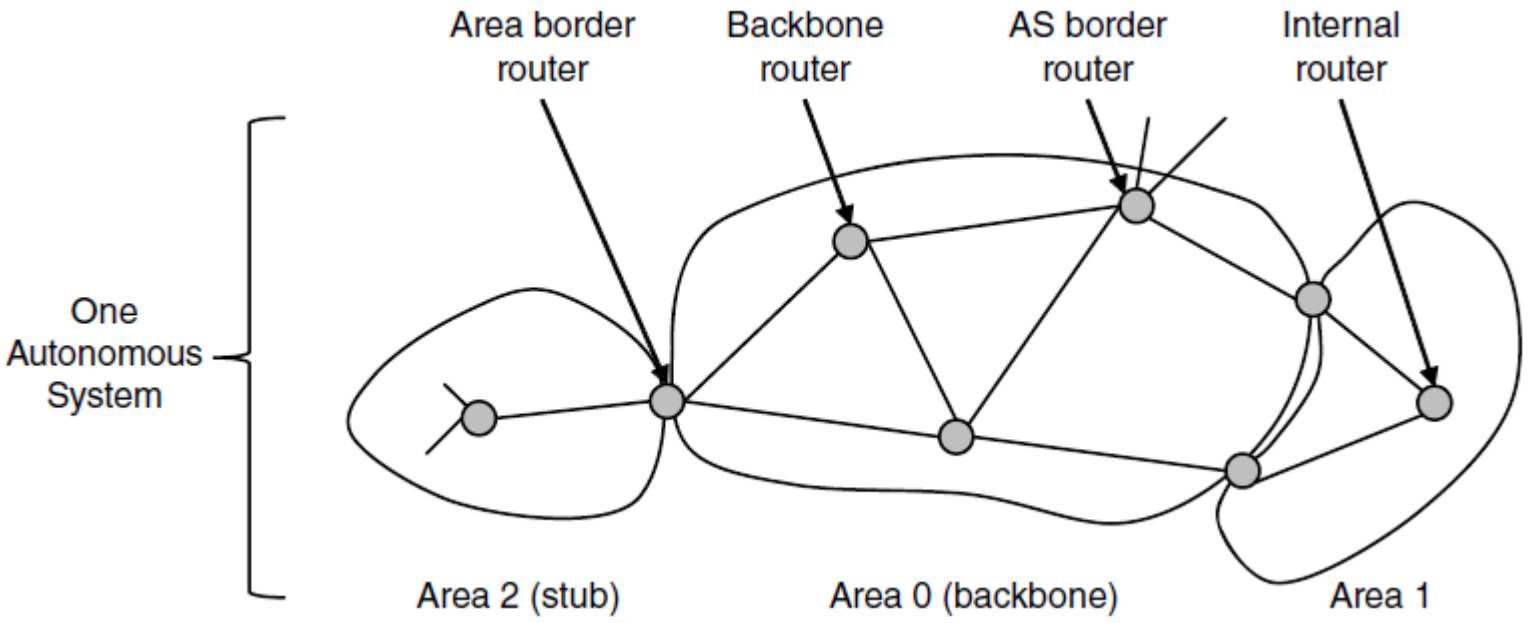
- The Internet is made up of a large number of independent networks or ASes (**Autonomous Systems**) that are operated by different organizations, usually a company, university, or ISP.

- Inside of its own network, an organization can use its own algorithm for internal routing, or **intradomain routing** (its protocol also called **interior gateway protocol**, e.g. OSPF, RIP)

- Later, we will study the problem of routing between independently operated networks, or **interdomain routing** (its protocol also called **exterior gateway protocol,** e.g. BGP, Border Gateway Protocol).

OSPF - Interior Routing Protocol

- OSPF divides one large network (Autonomous System) into areas connected to a backbone area
  - Helps to scale; summaries go over area borders



OSPF-Interior Routing Protocol

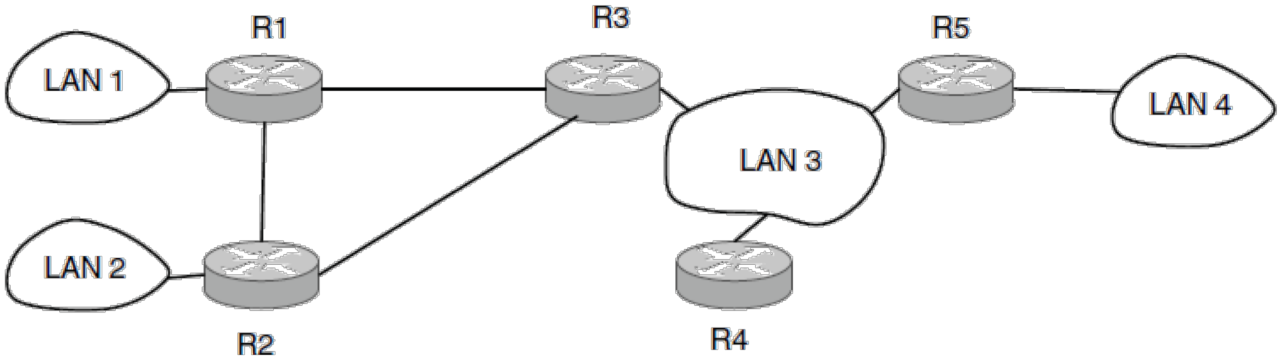
- OSPF uses link-state routing:
  - Uses messages below to reliably flood topology
  - Then runs Dijkstra to compute routes

Message type	Description
Hello	Used to discover who the neighbors are
Link state update	Provides the sender's costs to its neighbors
Link state ack	Acknowledges link state update
Database description	Announces which updates the sender has
Link state request	Requests information from the partner

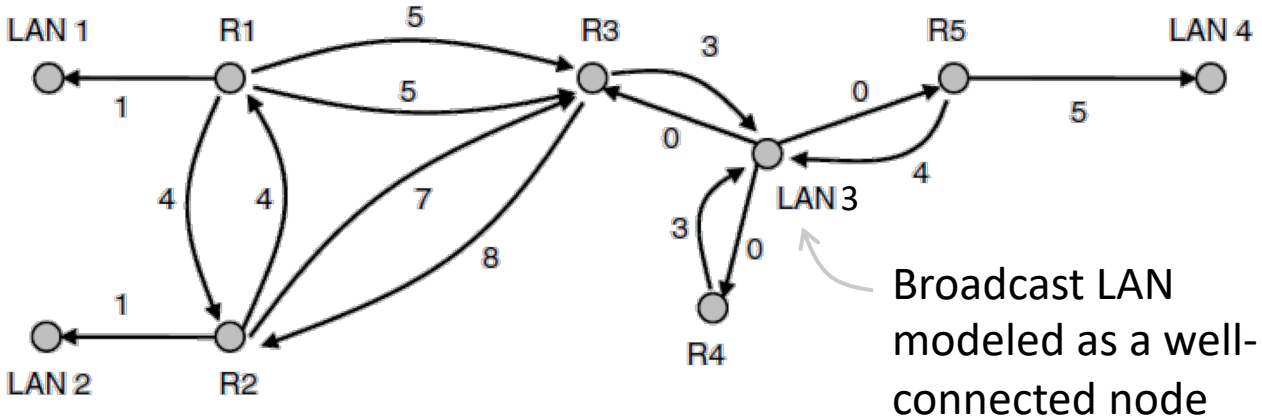
OSPF- Interior Routing Protocol

- OSPF computes routes for a single network (e.g., ISP)
  - Models network as a graph of weighted edges

Network:



Graph:



Broadcast LAN modeled as a well-connected node

BGP— Exterior Routing Protocol

- BGP (Border Gateway Protocol) computes routes across interconnected, autonomous networks, using path vector protocol

- Key role is to respect networks' policy constraints, not efficiency (which is the target of OSPF and RIP)

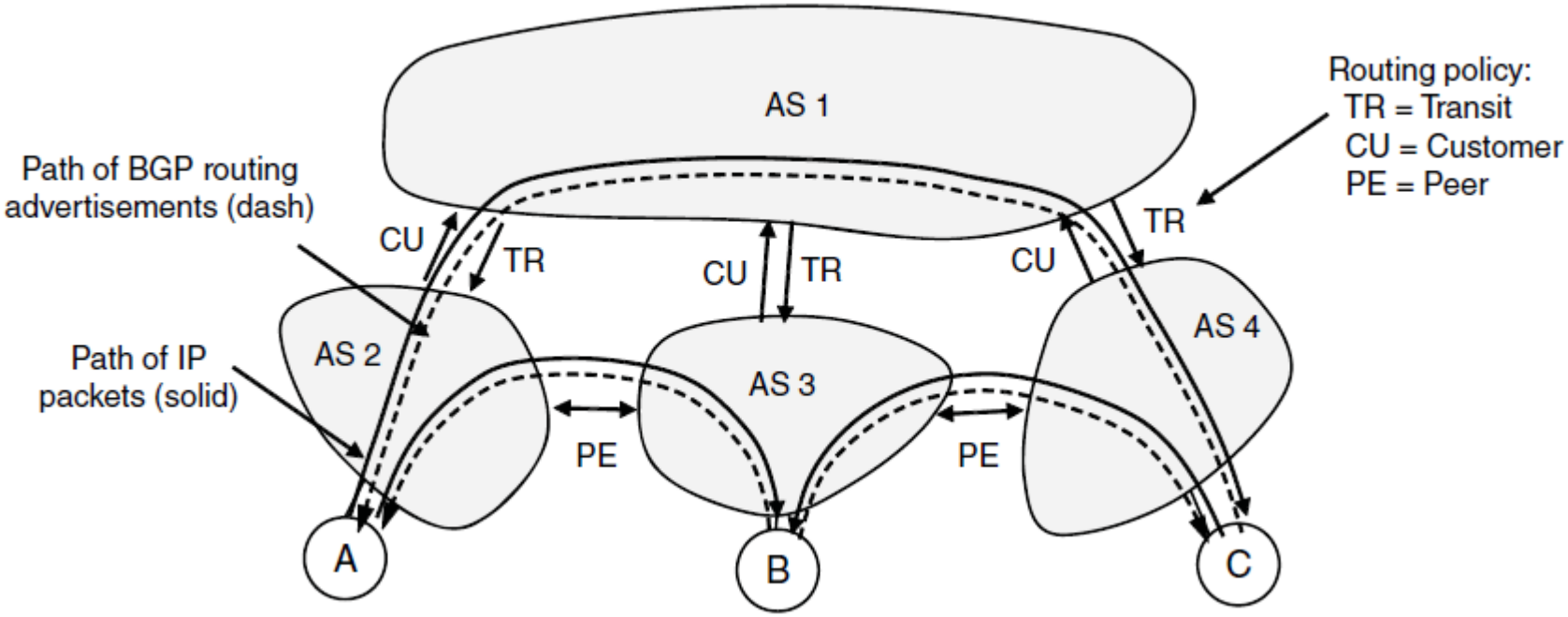
- Path vector protocol (compared to distance vector DV)
  - Pick a route to use (DV: has to have the minimum distance)
  - Keeps track of the path used (DV: maintain the route cost)
  - Uses TCP for exchanging patch vector (DV: UDP)

BGP— Exterior Routing Protocol

- Example policy constraints:
  - No commercial traffic for educational network
  - Never put Iraq on route starting at Pentagon
  - Choose cheaper network
  - Choose better performing network
  - Don’t go from Apple to Google to Apple
- Since different networks have different practices and goals we can’t reduce the preferred routes to a single weight number attached to links.

BGP— Exterior Routing Protocol

- Common policy distinction is transit vs. peering:
  - Transit carries traffic for pay; peers for mutual benefit
  - AS1 carries AS2↔AS4 (Transit) but not AS3 (Peer)



BGP— Exterior Routing Protocol

- BGP propagates messages along policy-compliant routes
  - Message has prefix, AS path (to detect loops) and next-hop IP (to send over the local network)