Network Layer

COMP90007 Internet Technologies

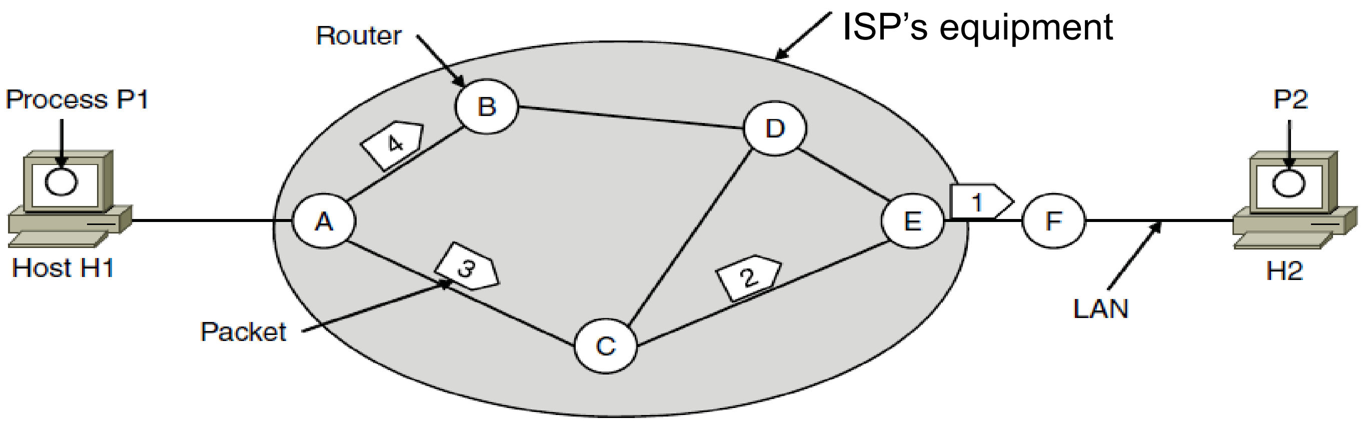
Lecturer: Ling Luo

Semester 2, 2020

Routing

Consider the network as a **graph of nodes** and **links**:

- Routing is the process of discovering network paths
- Decide what to optimize: hops, delay, etc.
- Update routes for changes in topology (e.g., failures)



A's table (initially)

Dest.	Line
A	⊠
B	B
C	C
D	B
E	C
F	C

A's table (later)

A	⊠
B	B
C	C
D	B
E	B
F	B

C's Table

A	A
B	A
C	⊠
D	E
E	E
F	E

E's Table

A	C
B	D
C	C
D	D
E	⊠
F	F

Routing Algorithms (1)

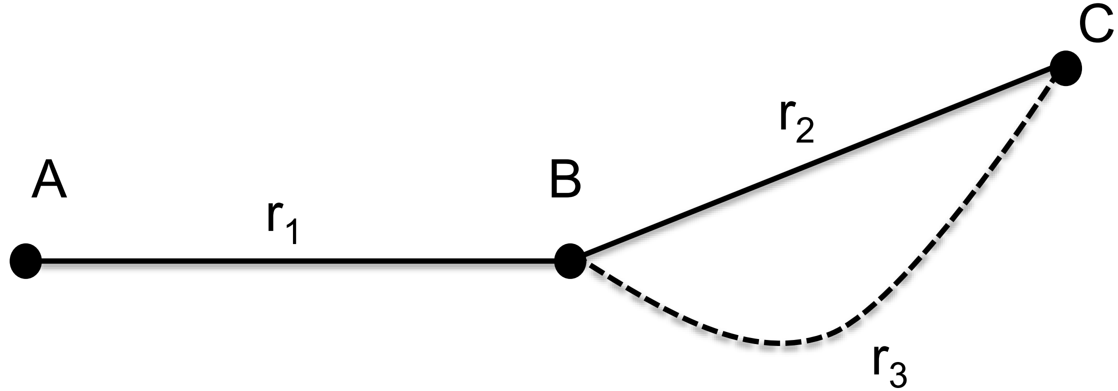
- The **routing algorithm** is responsible for deciding on **which output line an incoming packet should be transmitted**
- **Non-Adaptive Algorithms**
  - Static decision-making process (static routing)
- **Adaptive Algorithms**
  - Dynamic decision-making process (dynamic routing)
  - Changes in network topology, traffic, etc.

Routing Algorithms (2)

- Non-adaptive
  - Shortest path routing
  - Flooding
- Adaptive
  - Distance vector routing
  - Link state routing
- Hierarchical routing
- Broadcasting routing
- Multicasting routing

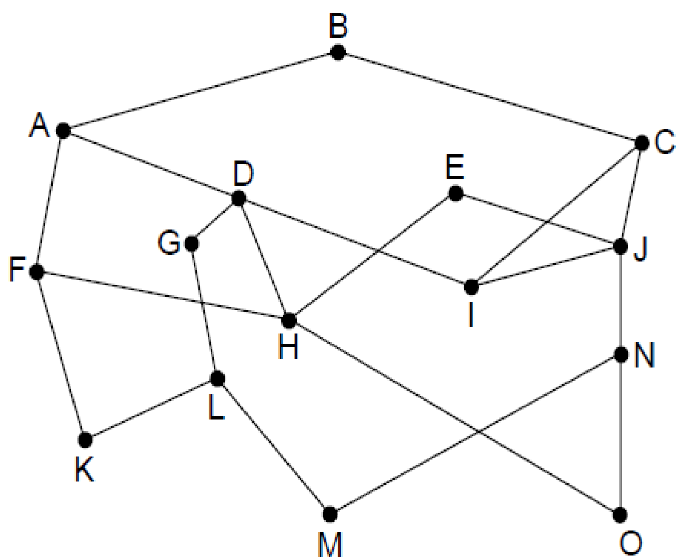
Optimality Principle

- If router B is on the optimal path from router A to router C, then the optimal path from B to C also falls along the same route.

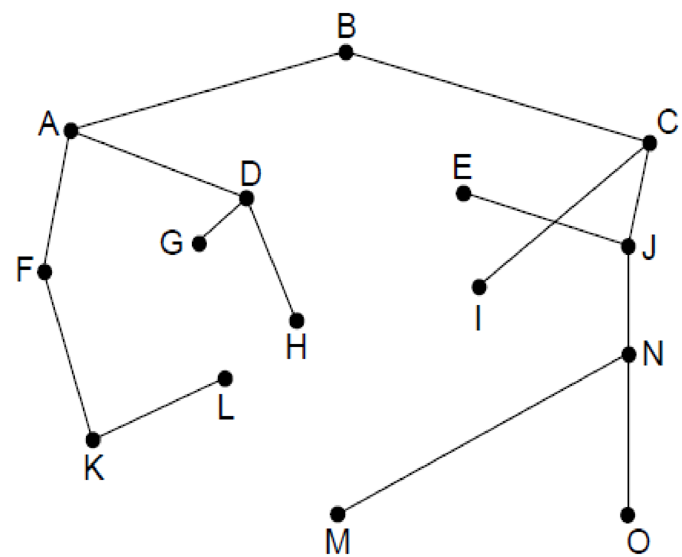


Sink Tree

- **Sink Tree**: the set of optimal routes from all sources to a given destination forms a tree rooted at the destination
- Goal of a routing algorithm: discover and utilise the sink trees for all routers



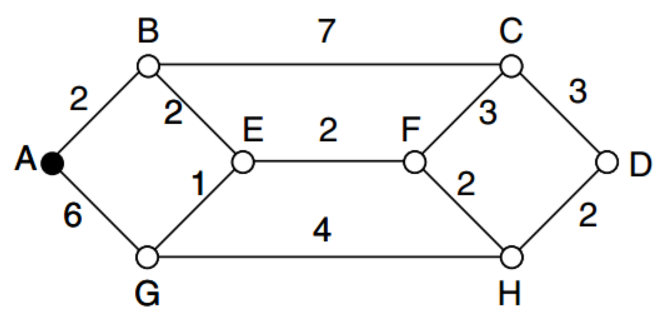
Network



Sink tree of best paths to router B

Shortest Path Routing

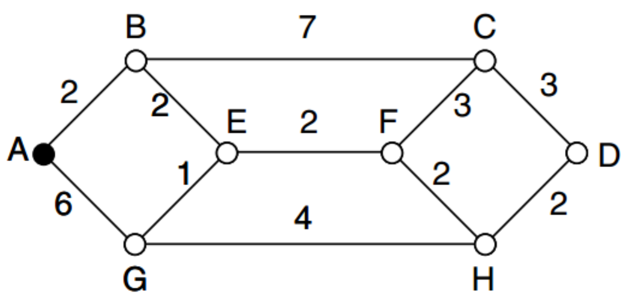
- A non-adaptive algorithm
- Shortest path can be determined by building a graph with each node representing a router, and each arc representing a communication link
- Metrics: number of hops, distance, delay etc.
- To choose a path between 2 routers, the algorithm finds the shortest path between them on the graph



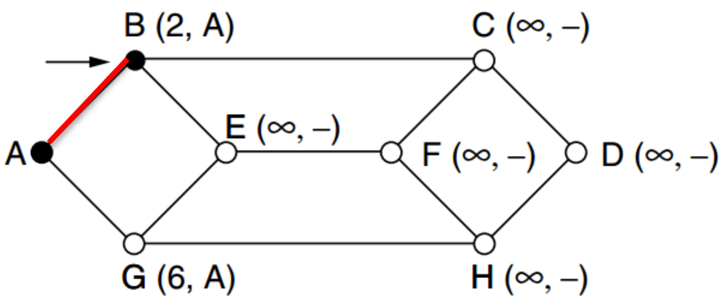
Shortest Path: Dijkstra's Algorithm (1)

- 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:
  1) Create **a set *P, tracking* the nodes added in the tree**. Initialize it as empty.
  2) For each node, assign a **distance value *d* from the node to sink**. Initialize the distance for all nodes as infinity.
  3) Start from the sink node, assign distance as 0*.*
  4) **Repeat** when *P* doesn't include all nodes:
     i. For all the nodes not in *P, c*ompare distance *d*
     ii. Pick a node *v* with min distance and add it to *P*
     iii. Update *d* for all the adjacent nodes of *v* (newly added node)

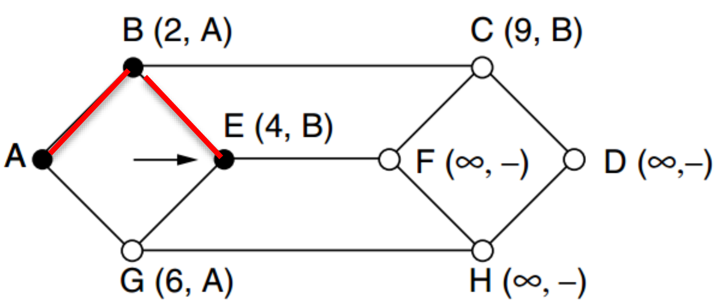
Shortest Path: Dijkstra's Algorithm (2)



(a)



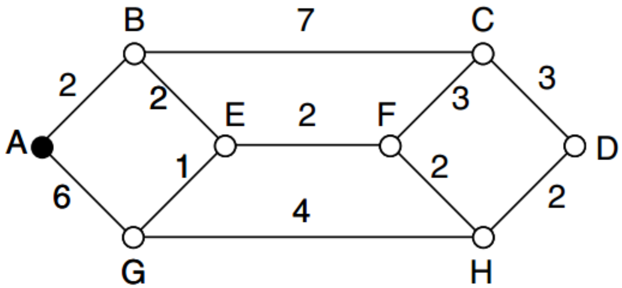
(b)

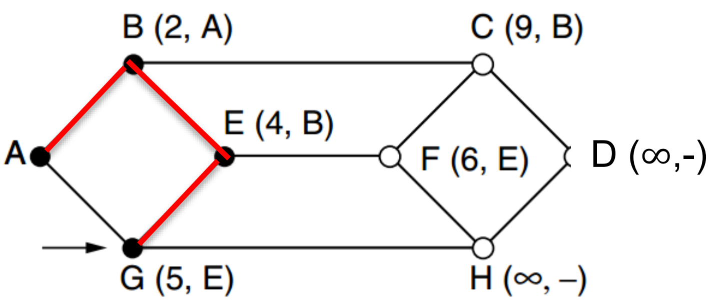


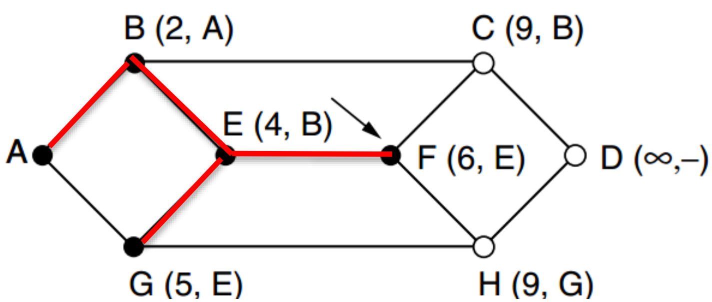
(c)

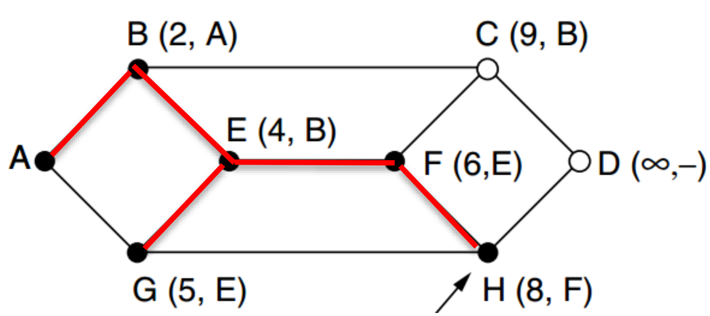
Distance to A **Set P**

n	A	B	C	D	E	F	G	H	Set P
1	0	∞	∞	∞	∞	∞	∞	∞	{A}
2	--	**2**	∞	∞	∞	∞	6	∞	{A, B}
3	--	--	9	∞	**4**	∞	6	∞	{A, B, E}









. . .

Distance to A **Set P**

n	A	B	C	D	E	F	G	H	Set P
1	0	∞	∞	∞	∞	∞	∞	∞	{A}
2	--	2	∞	∞	∞	∞	6	∞	{A, B}
3	--	--	9	∞	4	∞	6	∞	{A, B, E}
4	**--**	**--**	9	∞	**--**	6	**5**	∞	{A, B, E, G}
5	**--**	**--**	9	∞	**--**	**6**	**--**	9	{A, B, E, G, F}
6	**--**	**--**	9	∞	**--**	**--**	**--**	**8**	{A, B, E, G, F, H}
7	**--**	**--**	**9**	10	**--**	**--**	**--**	**--**	{A, B, E, G, F, H, C}
8	**--**	**--**	**--**	**10**	**--**	**--**	**--**	**--**	{A, B, E, G, F, H, C, D}

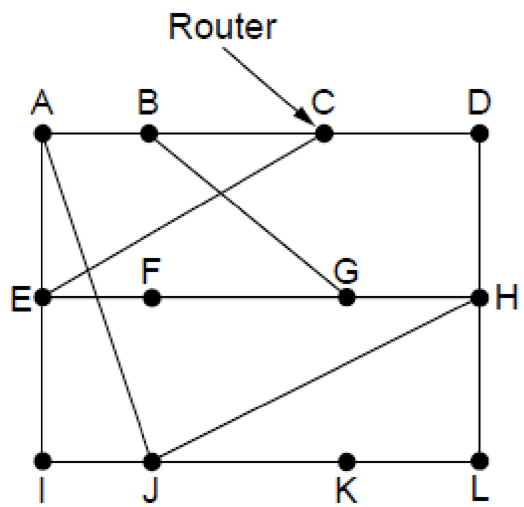
Flooding

- A non-adaptive algorithm
- Every incoming packet is sent out on **every outgoing line except the one on which it arrived**
- Inefficient: generates a large number of duplicate packets
- Selective flooding is an improved variation
  - Routers send packets only on links which are approximately in the right direction

Distance Vector Routing (1)

- A dynamic algorithm
  - Each router maintains a table which includes the best known distance to each destination and which line to use to get there
  - Tables are updated by exchanging information with neighboring routers
  - Global information shared locally
- Algorithm:
  1) Each node knows distance of links to its neighbors
  2) Each node **advertises** vector of lowest known distances to **all neighbors**
  3) Each node uses received vectors to **update** its own
  4) Repeat periodically

Distance Vector Routing (2)



Network

To	A	I	H	K	New estimated delay from J	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		

New vector for J

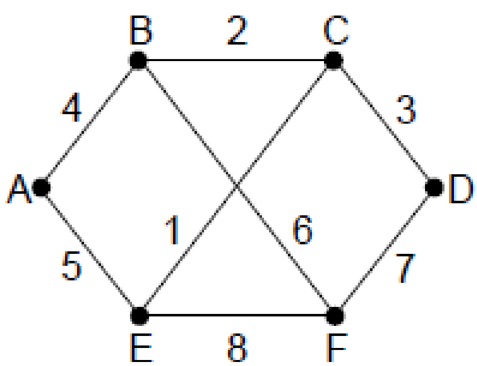
Vectors received at J from neighbors A, I, H and K

Link State Routing

- A dynamic algorithm
  - An alternative to distance vector: **too long to converge** after the network topology changed
  - Widely used in the Internet, e.g. Open Shortest Path First (OSPF)
  - More computation but simpler dynamics
  - Local information shared globally, using flooding
- Algorithm: each router has to
  1) Discover neighbors and learn network addresses
  2) Measure delay or cost to each neighbor
  3) **Build link state packet**
  4) Send this packet to **all other routers**
  5) **Compute the shortest path** to every other router, e.g. using Dijkstra's algorithm

Building Link State Packets

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



Network

Link State Packets

A	
Seq.	
Age	
B	4
E	5

B	
Seq.	
Age	
A	4
C	2
F	6

C	
Seq.	
Age	
B	2
D	3
E	1

D	
Seq.	
Age	
C	3
F	7

E	
Seq.	
Age	
A	5
C	1
F	8

F	
Seq.	
Age	
B	6
D	7
E	8

LSP for all nodes

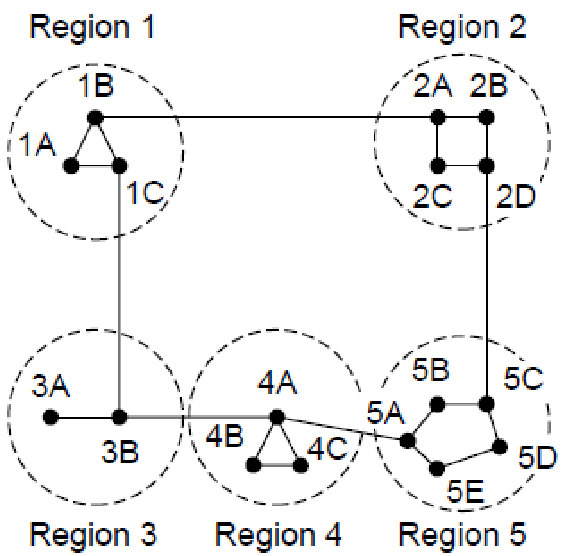
- When to build LSP?
  - Periodically at regular intervals
  - Build them when some significant event occurs, e.g. a line or neighbor going down or coming back up again, or changing its properties considerably

Hierarchical Routing (1)

- As networks grow in size, routing tables expand and this impacts CPU and memory requirements
- Dividing all routers into regions increases efficiencies
  - Each router knows everything about other routers in its region but **nothing about routers in other regions**
  - Routers which connect to two regions act as exchange points for routing decisions

Hierarchical Routing (2)

- Hierarchical routing reduces the work of computation but may result in slightly longer paths than flat routing



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

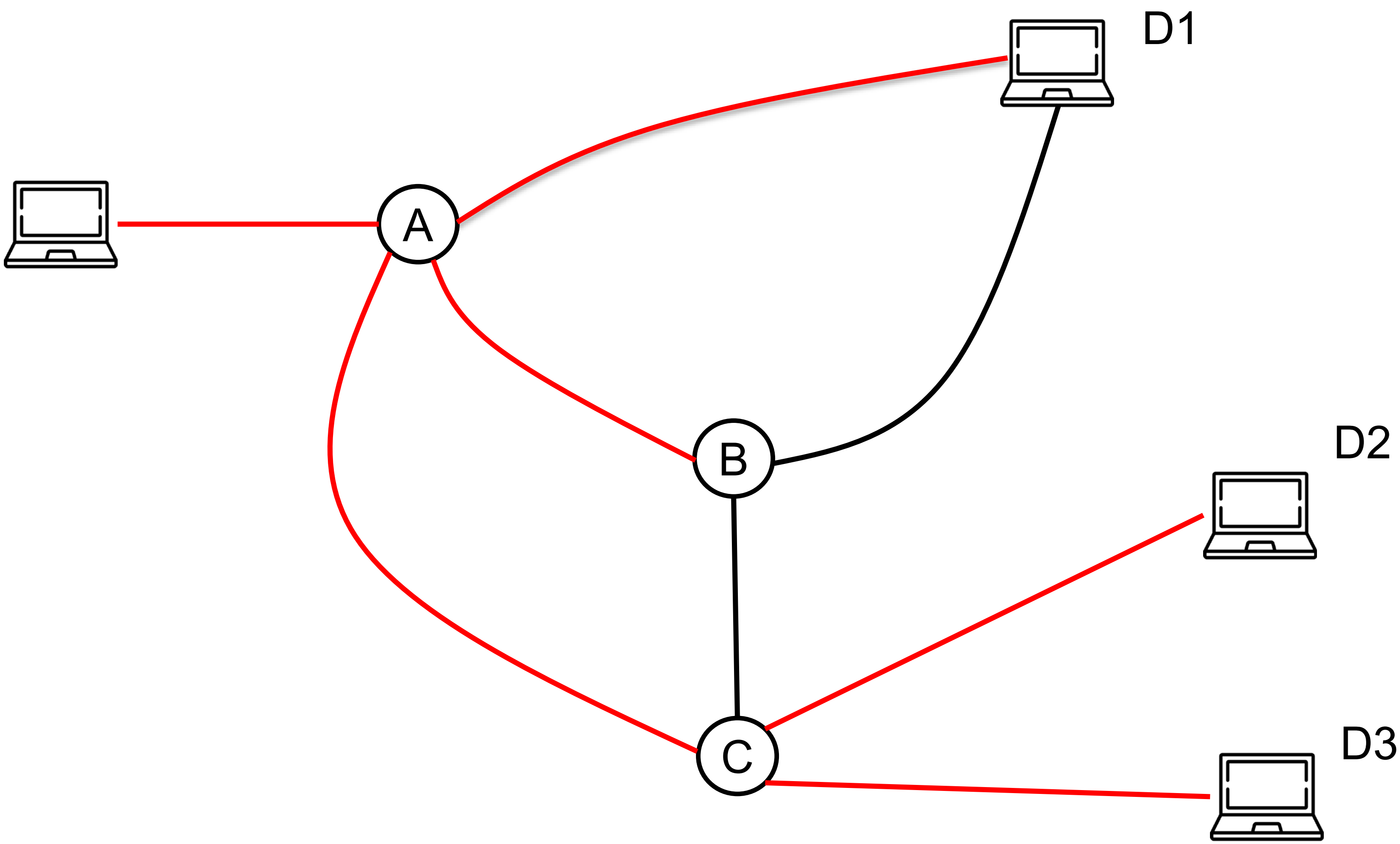
Hierarchical table for 1A

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

Broadcast Routing (1)

- Broadcast routing allows hosts to send messages to all other hosts.
  - Single distinct packet to each destination: inefficient, and source needs all destination addresses
  - Multi-destination routing: a router copies the packet for each outgoing line. Use bandwidth more efficiently, but source needs to know all the destination addresses
  - Flooding
  - Reverse path forwarding

Broadcast Routing (2)



Broadcast Routing (3)

- Reverse path forwarding

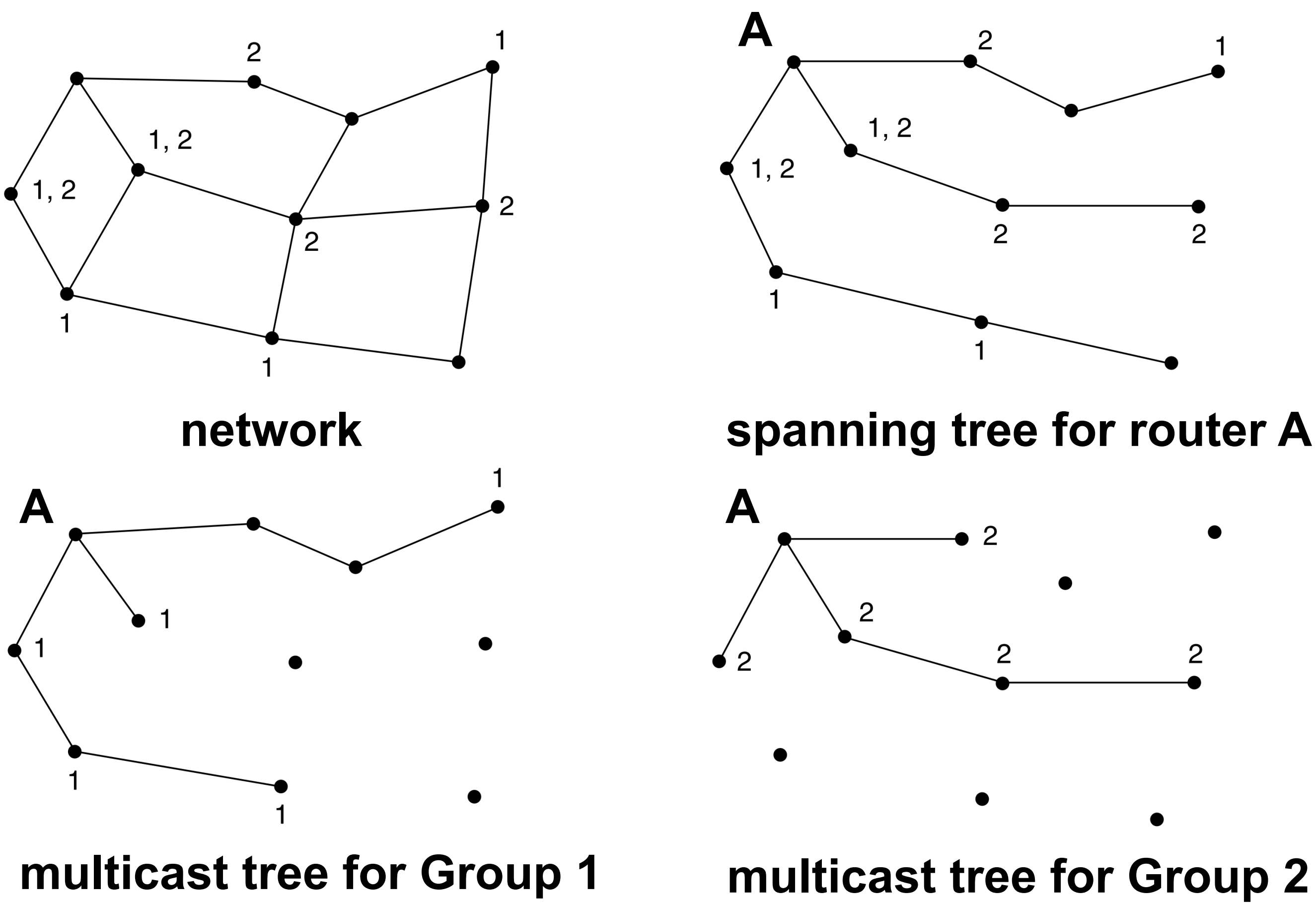
  The router checks if the broadcast packet arrived on the line normally used for sending packets to the source of the broadcast:

  - **Yes**: there is a **high probability** that the route used to transmit this packet is **the best,** and this packet is the **first copy**. The router then copies the packet and forwards them onto all other lines.
  - **No:** the packet is **discarded as** a likely **duplicate**.

Multicast Routing (1)

- Multicast routing allows hosts to send a message to a well-defined group within the whole network
- Each router computes a spanning tree covering all other routers
  - Spanning tree: subset of the graph that includes all nodes, but no loops.
  - Prunes the spanning tree to eliminate all lines which do not lead to members of the group

Multicast Routing (2)



network

spanning tree for router A

multicast tree for Group 1

multicast tree for Group 2