Distance Vector Routing

Overview

- Simple, early routing approach
  - —Used in ARPANET, and RIP
- One of the two main approaches to routing
  - —Distributed version of Bellman Ford
  - —Works, but very slow convergence after some failures
- Link-state algorithms are now typically used in practice
  - —More involved, better behaviour

Distance Vector Setting

- Each node computes its forwarding table in a distributed setting:

1. Nodes know only the cost to their neighbors; not the topology
2. Nodes can talk only to neighbors using messages
3. All nodes run the same algorithm concurrently
4. Nodes and links may fail, messages may be lost

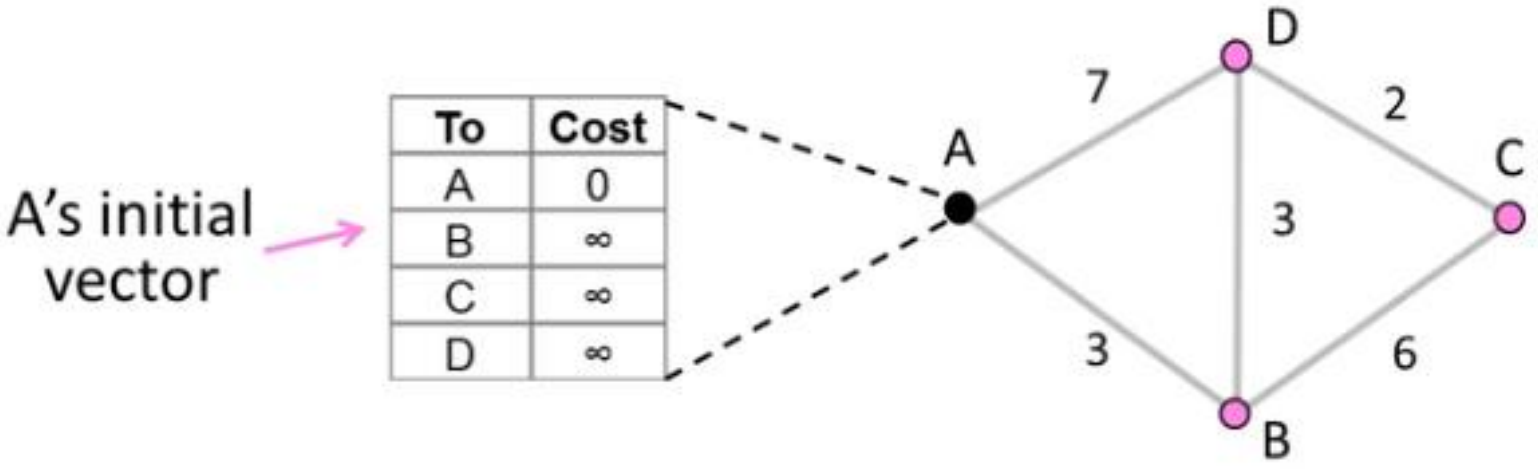
Distance Vector Algorithm

- Each node maintains a vector of distances (and next hops) to all destinations

1. Initialize vector with 0 (zero) cost to self, ∞ (infinity) to other destinations
2. Periodically send vector to neighbors
3. Update vector for each destination by selecting the shortest distance heard, after adding cost to neighbor link
   - —Use the best neighbor for forwarding

Distance vector Example

- Consider a simple network. Each node runs on its own
- E.g., node A can talk to nodes B and D



DV Example (2)

- First exchange, A hears from B, D finds 1-hop routes
  - A always learn min (B+3, D+7)

To	B says	D says
A	∞	∞
B	0	∞
C	∞	∞
D	∞	0

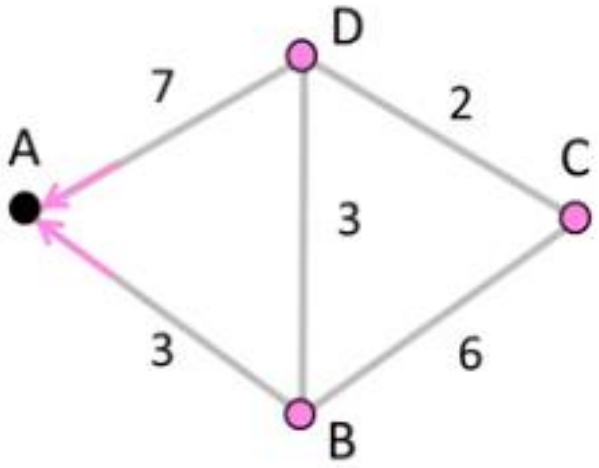
→

B +3	D +7
∞	∞
3	∞
∞	∞
∞	7

→

A learns Cost	Next
0	--
3	B
∞	--
7	D

= learned better route



DV Example (3)

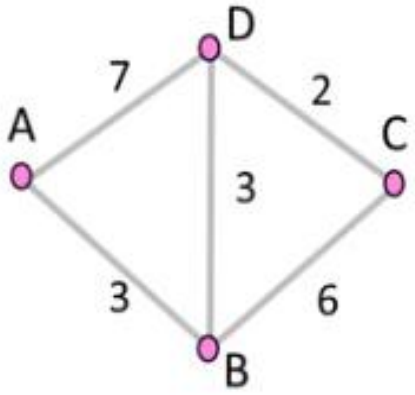
- First exchange for all nodes to find best 1-hop routes
  - E.g., B learns min(A+3, C+6, D+3)

To	A says	B says	C says	D says
A	0	∞	∞	∞
B	∞	0	∞	∞
C	∞	∞	0	∞
D	∞	∞	∞	0

→

A learns		B learns		C learns		D learns	
Cost	Next	Cost	Next	Cost	Next	Cost	Next
0	--	3	A	∞	--	7	A
3	B	0	--	6	B	3	B
∞	--	6	C	0	--	2	C
7	D	3	D	2	D	0	--

= learned better route



DV Example (4)

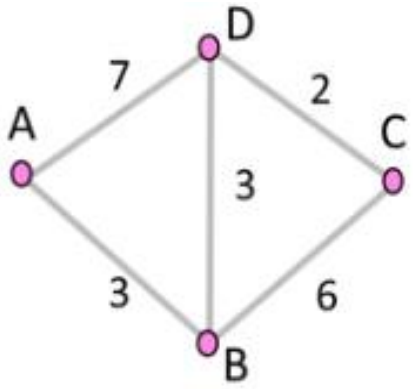
- Second exchange for all nodes to find best 2-hop routes

To	A says	B says	C says	D says
A	0	3	∞	7
B	3	0	6	3
C	∞	6	0	2
D	7	3	2	0

→

A learns		B learns		C learns		D learns	
Cost	Next	Cost	Next	Cost	Next	Cost	Next
0	--	3	A	9	B	6	B
3	B	0	--	5	D	3	B
9	D	5	D	0	--	2	C
6	B	3	D	2	D	0	--

= learned better route



DV Example (5)

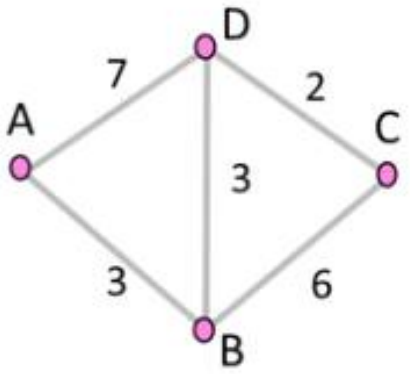
- Third exchange for all nodes to find best 3- hop routes

To	A says	B says	C says	D says
A	0	3	9	6
B	3	0	5	3
C	9	5	0	2
D	6	3	2	0

→

A learns		B learns		C learns		D learns	
Cost	Next	Cost	Next	Cost	Next	Cost	Next
0	--	3	A	8	D	6	B
3	B	0	--	5	D	3	B
8	B	5	D	0	--	2	C
6	B	3	D	2	D	0	--

= learned better route



DV Example (5)

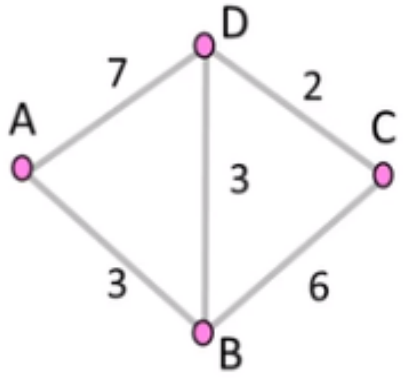
- Fourth and subsequent exchanges; converged

To	A says	B says	C says	D says
A	0	3	8	6
B	3	0	5	3
C	8	5	0	2
D	6	3	2	0

→

A learns		B learns		C learns		D learns	
Cost	Next	Cost	Next	Cost	Next	Cost	Next
0	--	3	A	8	D	6	B
3	B	0	--	5	D	3	B
8	B	5	D	0	--	2	C
6	B	3	D	2	D	0	--

= learned better route

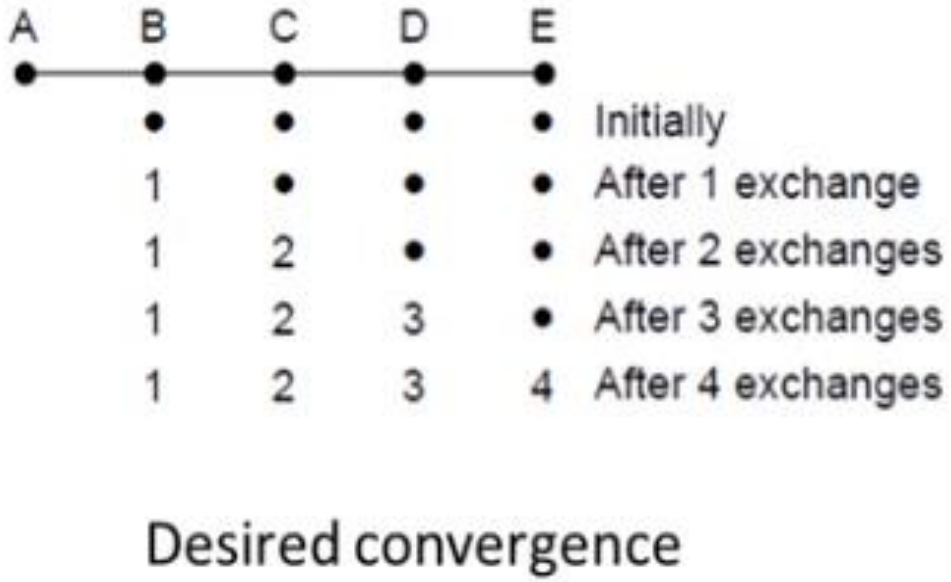


Distance Vector Dynamics

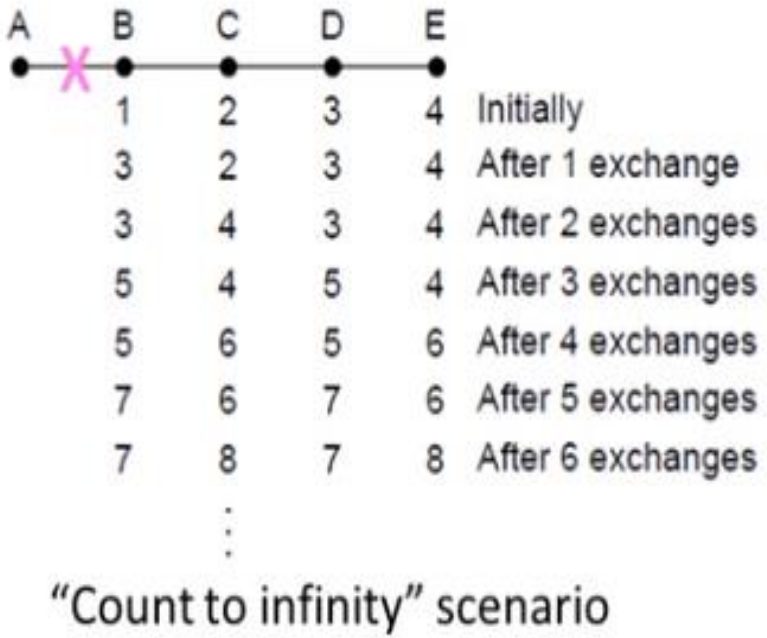
- Adding routes:
  - —News travel one hop per exchange
- Removing Routes
  - —When a node fails, no more exchanges, other node forget
- But partitions (unreachable nodes in divided network) are a problem
  - —"Count to infinity" scenario

DV Dynamics (2)

- Good news travel quickly, bad news slowly (inferred)



Desired convergence



"Count to infinity" scenario

DV Dynamics (3)

- Various heuristics to address
  - —E.g.- “spilt horizon, poison reverse” (Don’t send route back to where you learned it from.)
- But none are very effective
  - —Link state now favoured in practice
  - —Except when very resource-limited

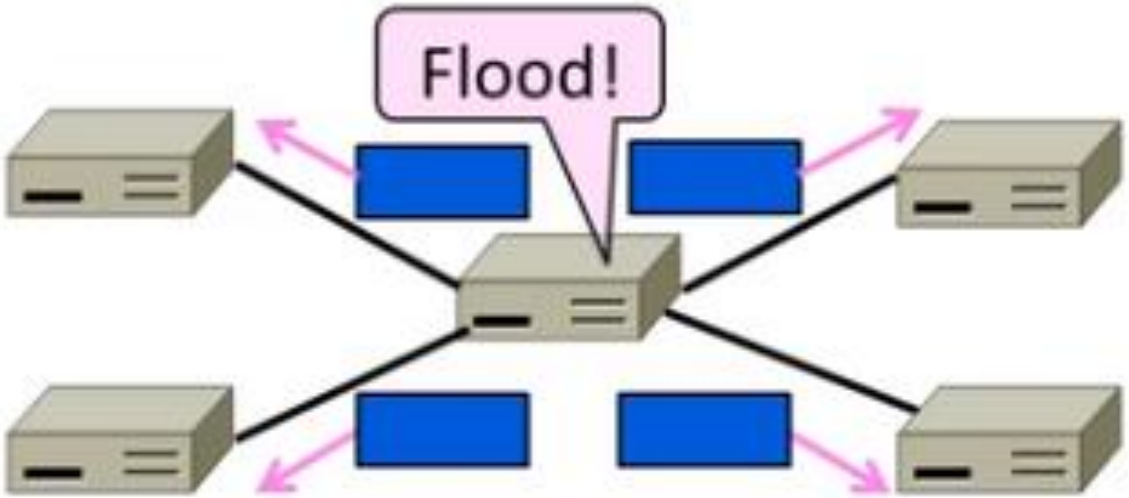
RIP (Routing Information Protocol)

- DV protocol with hop count as metric
  - —Infinity is 16 hops; limits network size
  - —Includes spilt horizon, poison reverse
- Router send vectors every 30 secs
  - —Runs on top of UDP
  - —Timeout in 180 secs to detect failures
- RIPv1 is specified in RFC1058 (1988)

Flooding

Overview

- How to broadcast a message to all nodes in the network with flooding
- Simple mechanism, but inefficient



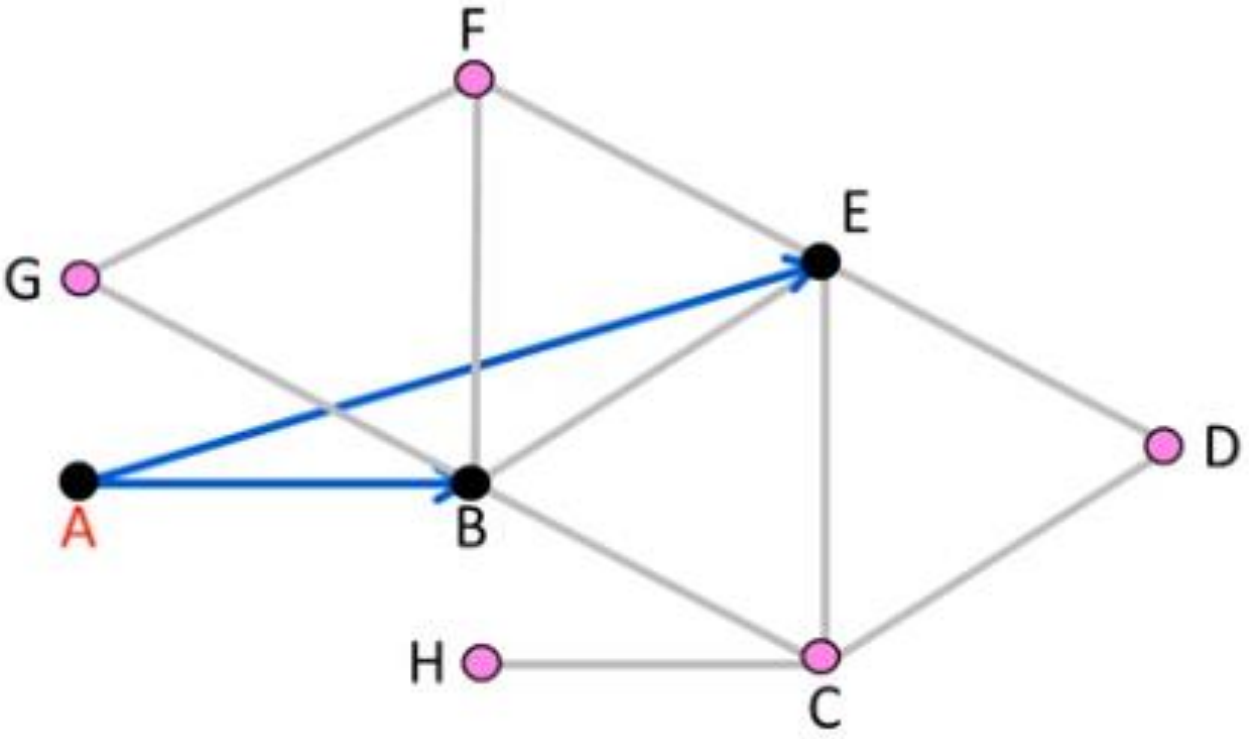
Flooding

- Rule used at each node:
  - —Sends an incoming message on to all other neighbors
  - —Remember the message so that it is only flood once

- Inefficient because one node may receive multiple copies of message

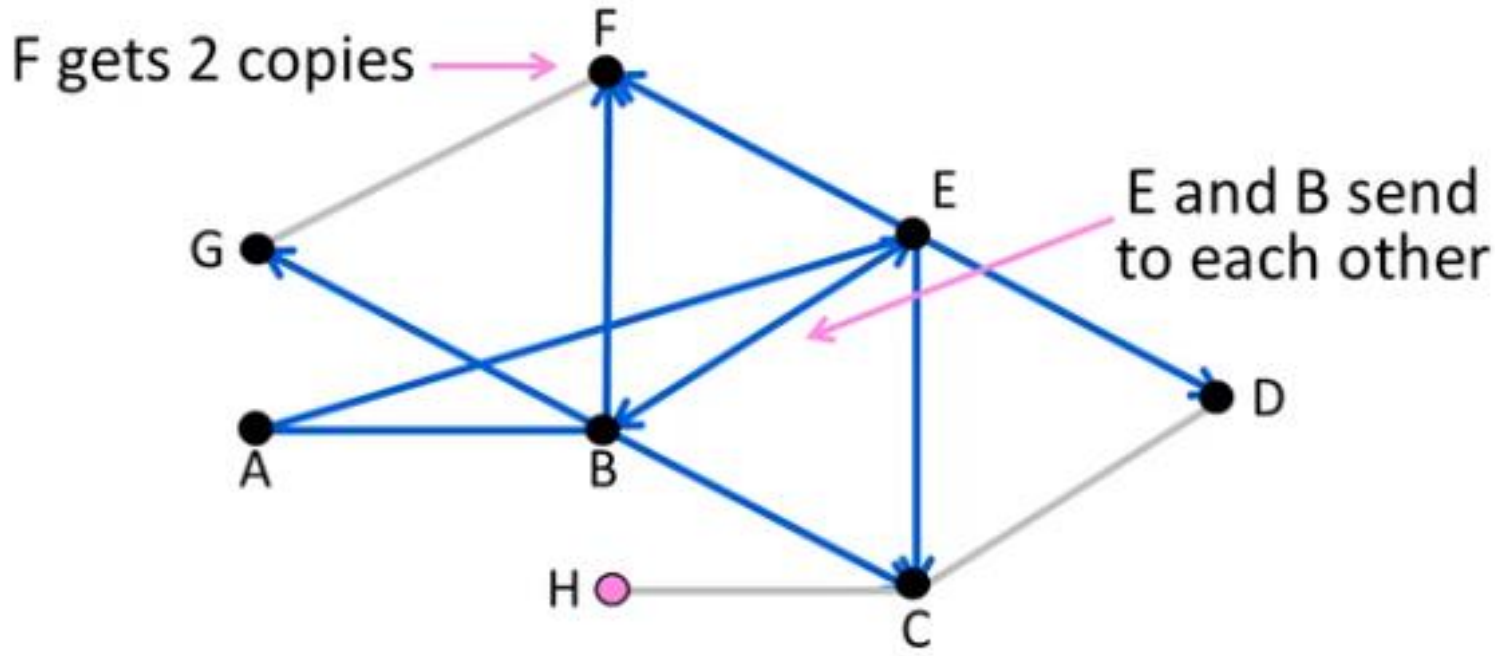
Flooding (2)

- Consider a flood from A; first reaches B via AB, E via AE



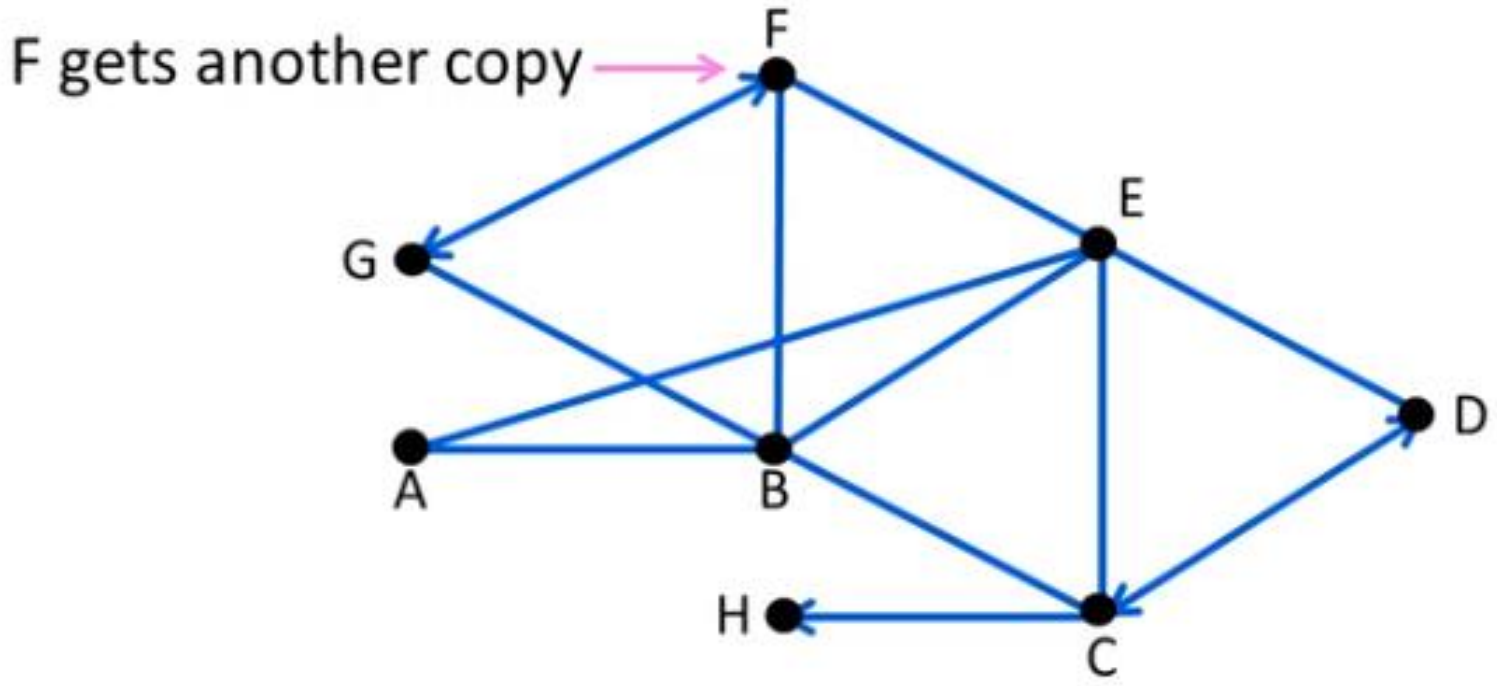
Flooding (3)

- Next B floods BC, BE, BF, BG, and E floods EB, EC, ED, EF



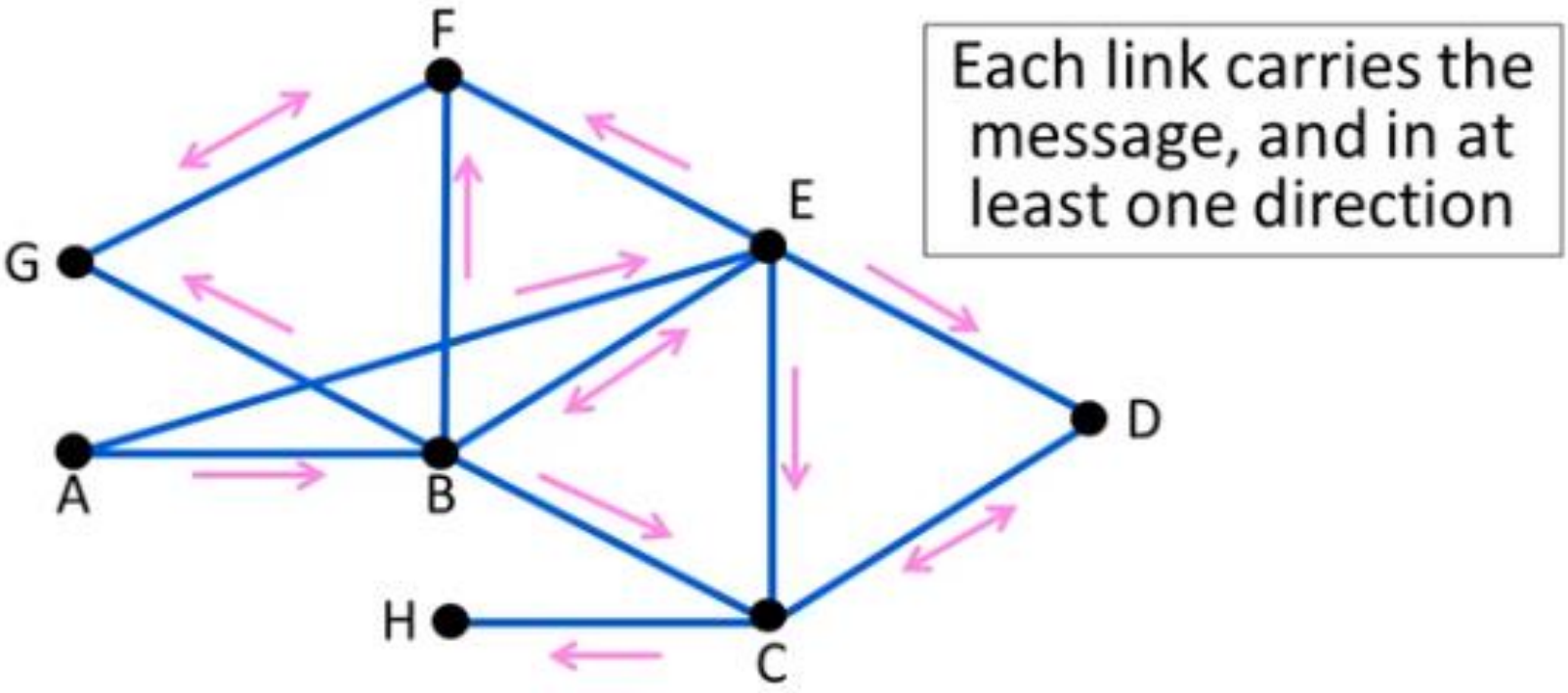
Flooding (4)

- C floods CD,CH; D floods DC; F floods FG; G floods GF



Flooding (5)

- H has no-one to flood … and we're done



Flooding Details

- Remember message (to stop flood) using source and sequence number
  - So next message (with higher sequence number) will go through
- To make flooding reliable, use ARQ
  - So receiver acknowledges, and sender resends if needed