University of Glasgow

School of Computing Science

60 YEARS OF COMPUTING AT GLASGOW

Intra-domain Routing (1)

Networked Systems (H)

Lecture 9

Lecture Outline

- Routing concepts
- Intra-domain unicast routing
    - Distance vector protocols
    - …

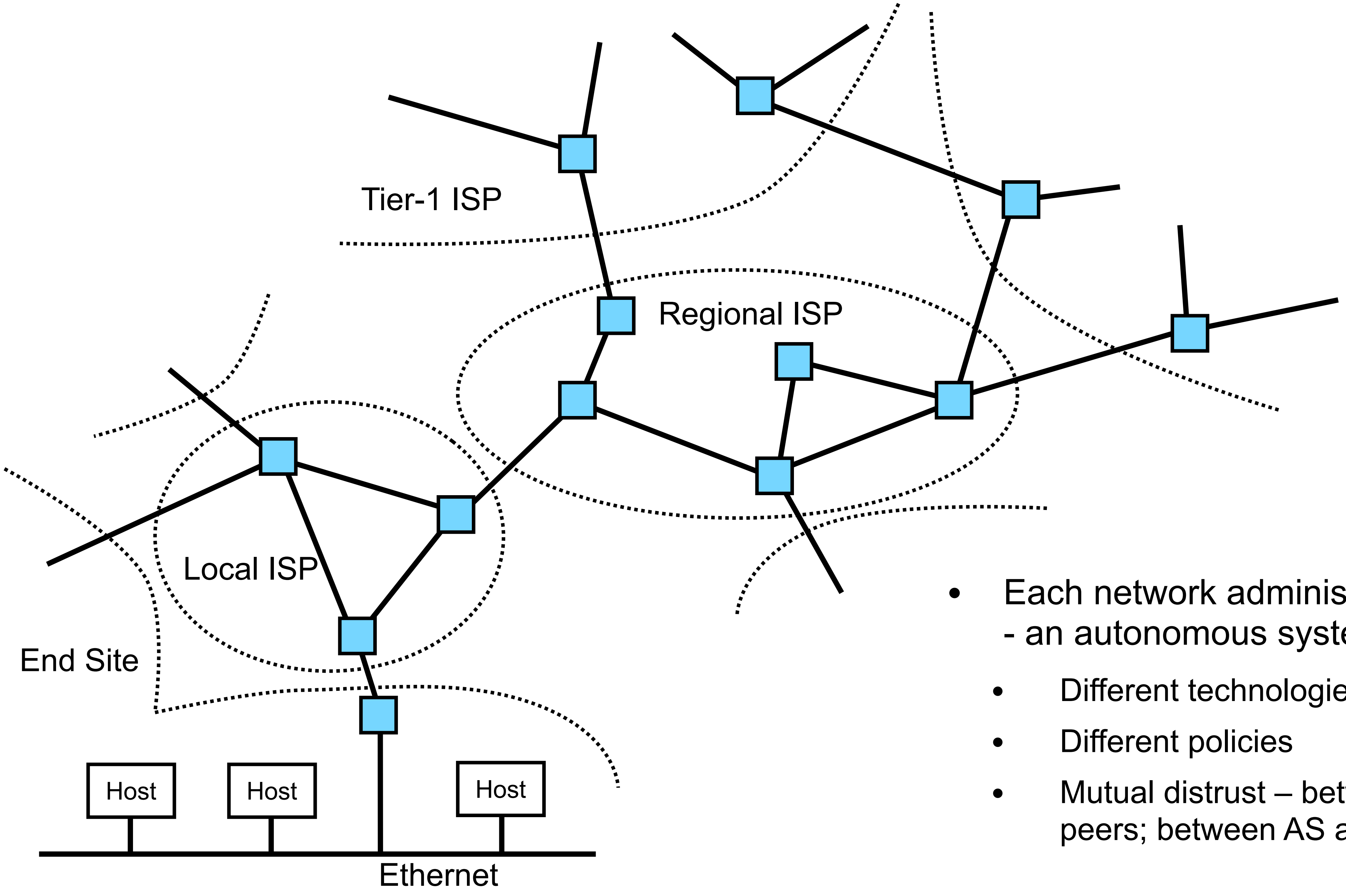
Routing

- Network layer responsible for routing data from source to destination across multiple hops
    - Nodes learn (a subset of) the network topology and run a routing algorithm to decide where to forward packets destined for other hosts
        - End hosts usually have a simple view of the topology (“my local network” and “everything else”) and a simple routing algorithm (“if it’s not on my local network, send it to the default gateway”)
        - Gateway devices (“routers”) exchange topology information, decide best route to destination based on knowledge of the entire network topology

Unicast Routing

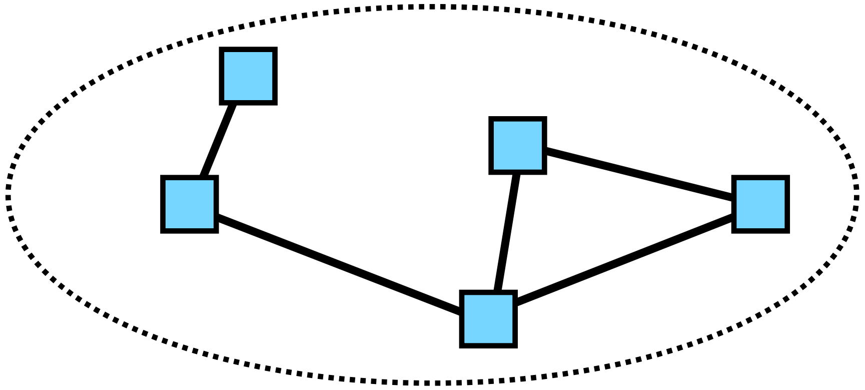
- Routing algorithms to deliver packets from a source to a single destination
- Choice of algorithm affected by usage scenario
    - Intra-domain routing
    - Inter-domain routing
    - Politics and economics

Routing in the Internet



- Each network administered separately - an autonomous system (AS)
  - Different technologies
  - Different policies
  - Mutual distrust – between AS and its peers; between AS and its customers

Intra-domain Unicast Routing



- Each network administered separately - an autonomous system (AS)
    - Different technologies
    - Different policies
    - Mutual distrust – between AS and its peers; between AS and its customers

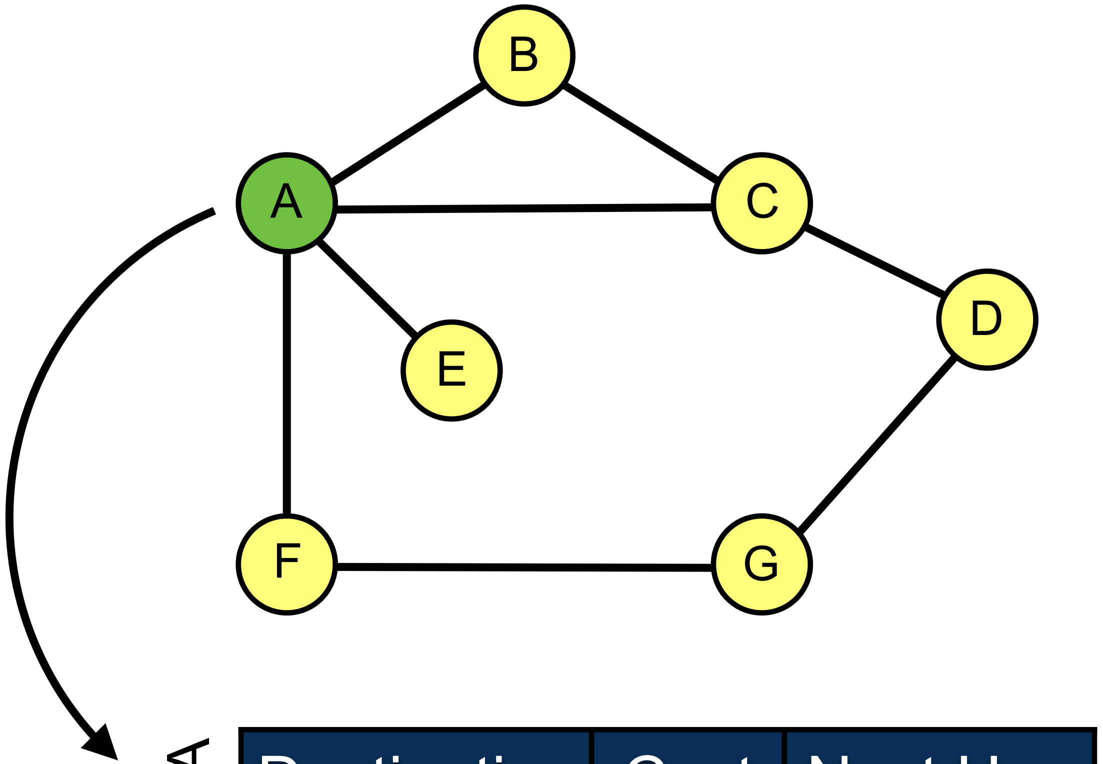
Intra-domain Unicast Routing

- Routing within an AS
  - Single trust domain
    - No policy restrictions on who can determine network topology
    - No policy restrictions on which links can be used
  - Desire efficient routing → shortest path
    - Make best use of the network you have available
  - Two approaches
    - Distance vector – the Routing Information Protocol (RIP)
    - Link state – Open Shortest Path First routing (OSPF)

Distance Vector Routing

- Each node maintains a vector containing the distance to every other node in the network
  - Periodically exchanged with neighbours, so eventually each node knows the distance to all other nodes
    - The routing table “converges” on a steady state
  - Links which are down or unknown have distance = ∞
- Forward packets along route with least distance to destination

Distance Vector: Example



Routing Table at Node A

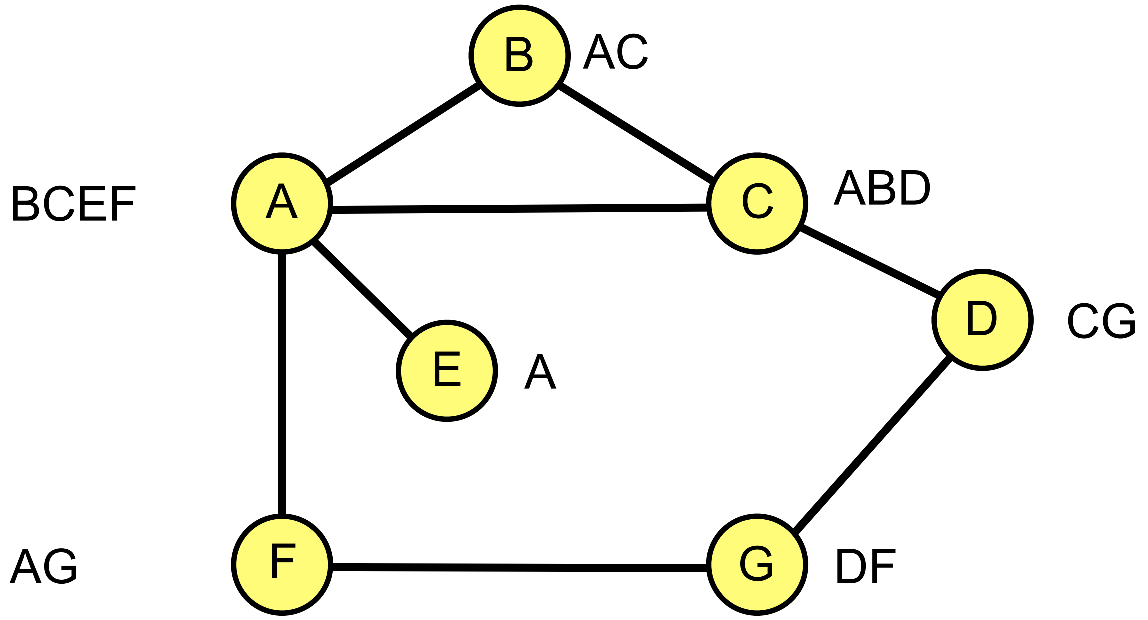
Destination	Cost	Next Hop
B	∞	-
C	∞	-
D	∞	-
E	∞	-
F	∞	-
G	∞	-

Information stored at node A, to allow routing to the other nodes

- This example uses names (A, B, C, …) to keep the diagram readable
- Real implementations identify nodes by their IP address, or by IP prefixes if routing to networks
- Initially table is empty – know of no other nodes

Corresponding tables at every other node

Distance Vector: Example



Time: 0

Nodes initialised; only know their immediate neighbours

Routing Table at Node A

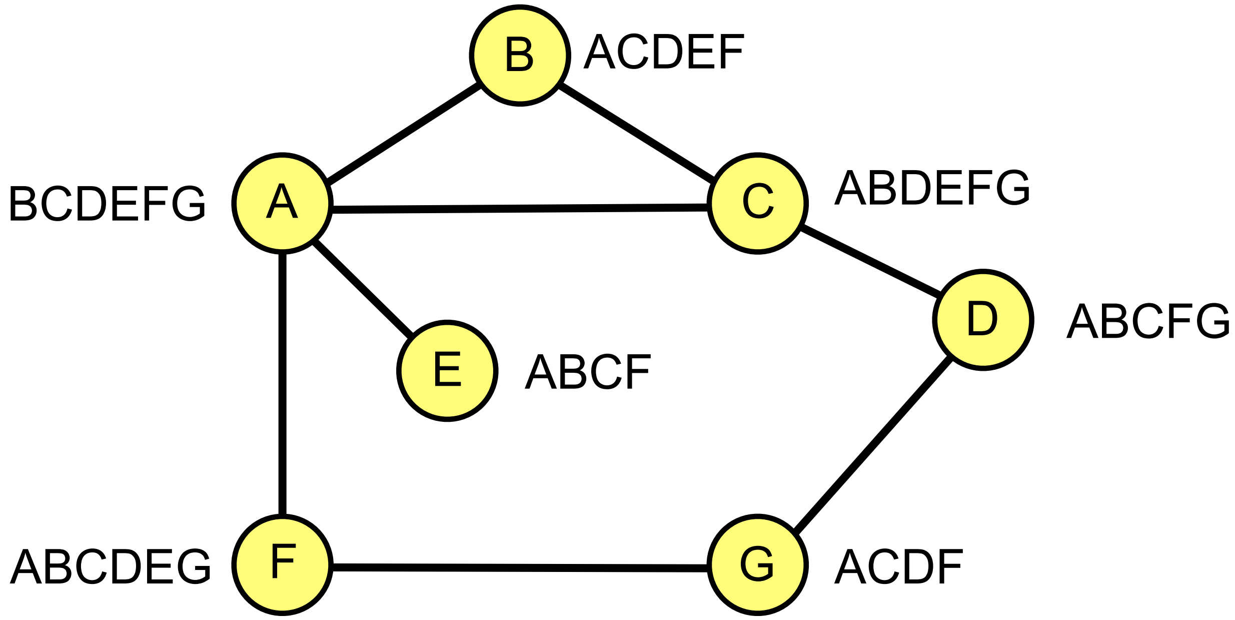
Destination	Cost	Next Hop
B	1	B
C	1	C
D	∞	-
E	1	E
F	1	F
G	∞	-

Distance to Reach Node

Information Stored at Node

	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Distance Vector: Example



Time: 1

Nodes also know neighbours of their neighbours – routing data has spread one hop

Routing Table at Node A

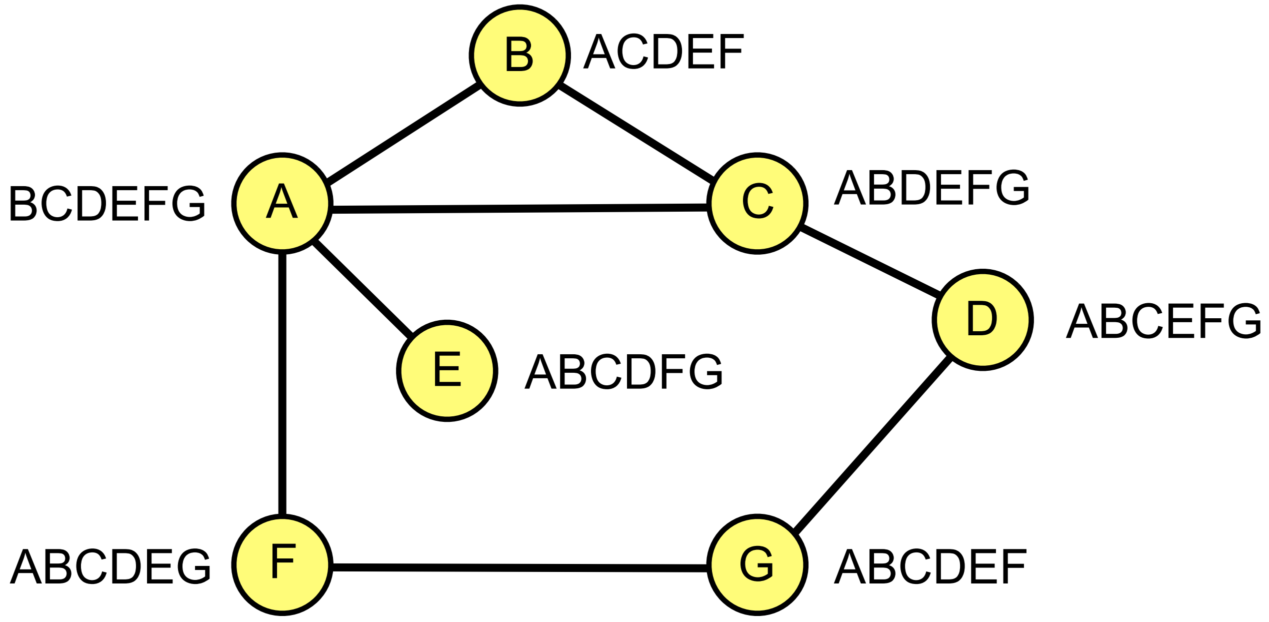
Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Distance to Reach Node

Information Stored at Node

	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	∞
C	1	1	0	1	2	2	2
D	2	2	1	0	∞	2	1
E	1	2	2	∞	0	2	∞
F	1	2	2	2	2	0	1
G	2	∞	2	1	∞	1	0

Distance Vector: Example



Time: 2

Routing data has spread two hops – table complete

Routing Table at Node A

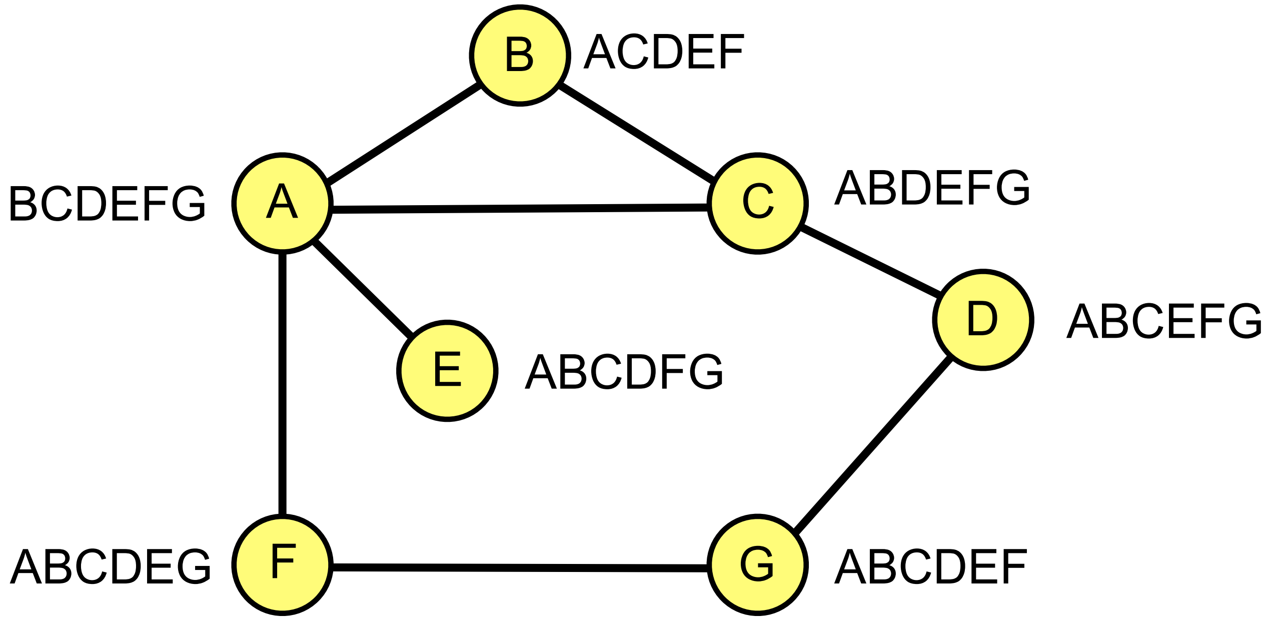
Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Distance to Reach Node

Information Stored at Node

	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

Distance Vector: Example



Time: 3

Nodes continue to exchange distance metrics in case the topology changes

Routing Table at Node A

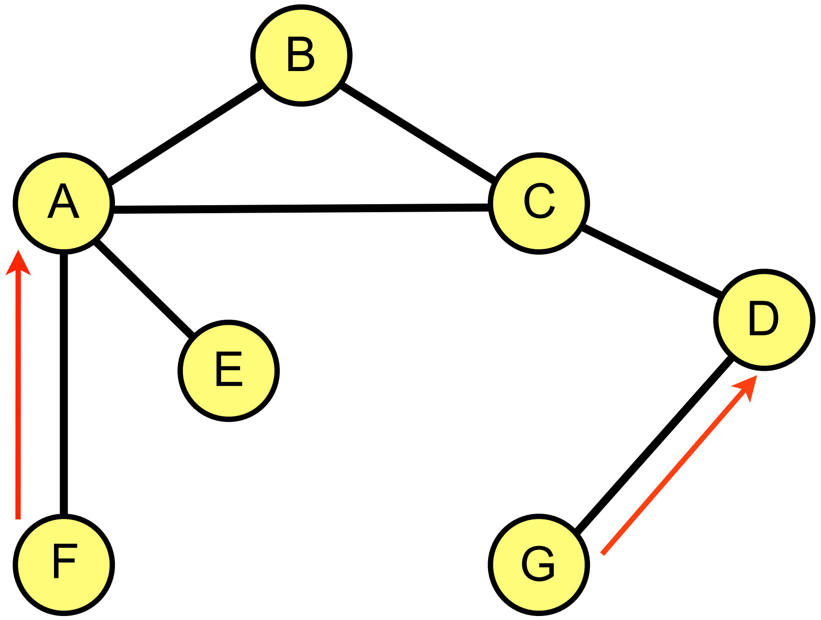
Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Distance to Reach Node

Information Stored at Node

	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

Distance Vector: Example



Time: 4

Link between F and G fails
F and G notice, set the link distance to ∞, and pass an update to A and D

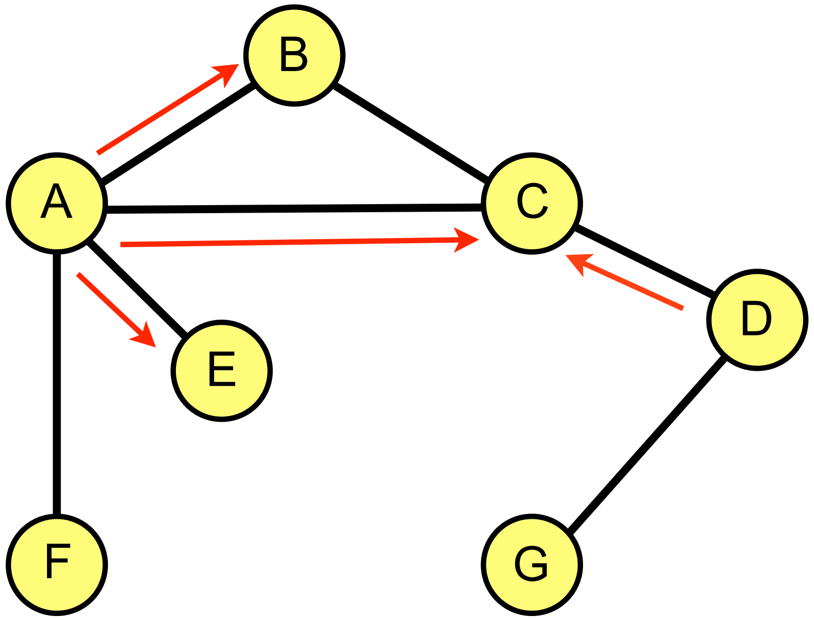
Routing Table at Node A

Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Information Stored at Node / Distance to Reach Node

	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	∞
G	2	3	2	1	3	∞	0

Distance Vector: Example



Time: 5

A sets its distance to G to ∞
D sets its distance to F to ∞
Both pass on news of the link failure

Routing Table at Node A

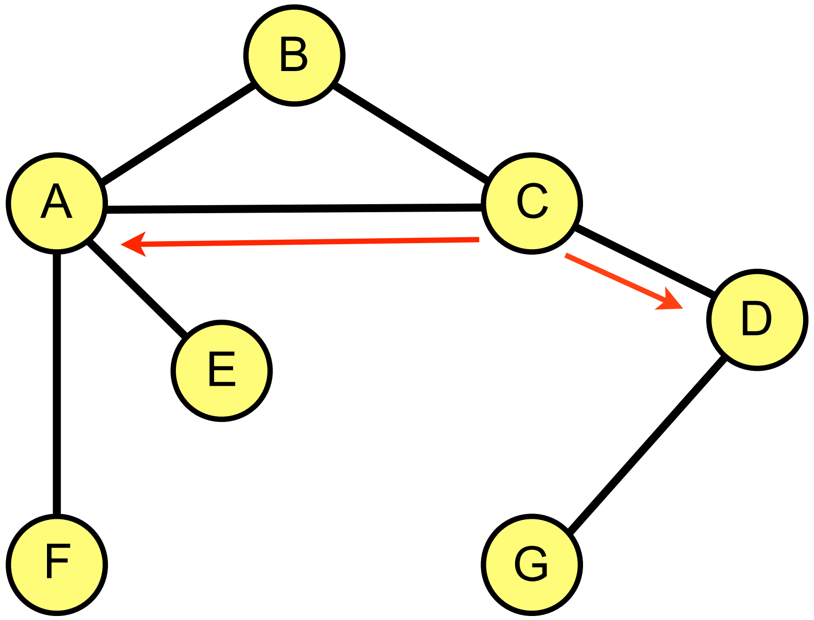
Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	∞	-

Distance to Reach Node

Information Stored at Node

	A	B	C	D	E	F	G
A	0	1	1	2	1	1	∞
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	∞	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	∞
G	2	3	2	1	3	∞	0

Distance Vector: Example



Time: 6

C knows it can reach F and G in 2 hops via alternate paths, so advertises shorter routes; network begins to converge

Routing Table at Node A

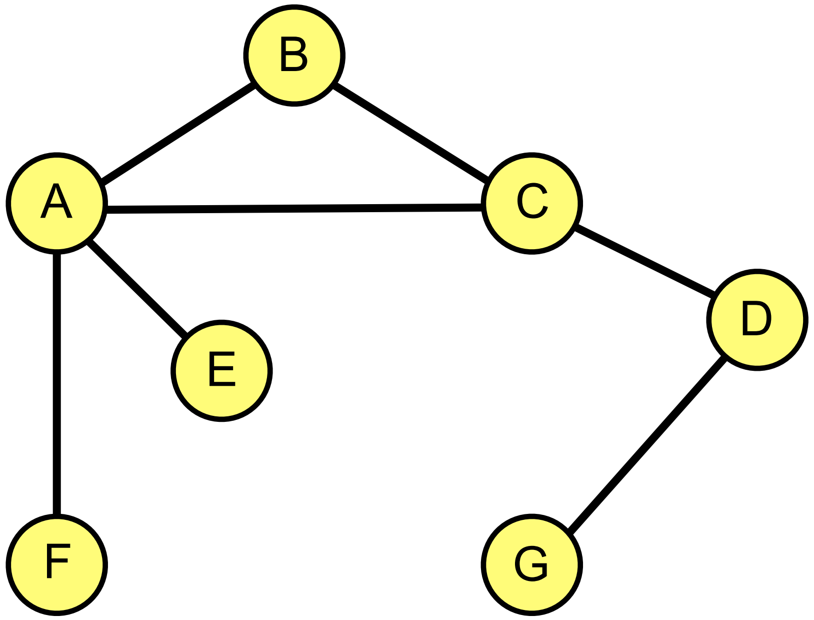
Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	3	C

Distance to Reach Node

Information Stored at Node

	A	B	C	D	E	F	G
A	0	1	1	2	1	1	3
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	3	1
E	1	2	2	3	0	2	∞
F	1	2	2	2	2	0	∞
G	2	3	2	1	3	∞	0

Distance Vector: Example



Time: 7

Eventually, the network is stable in a new topology

Routing Table at Node A

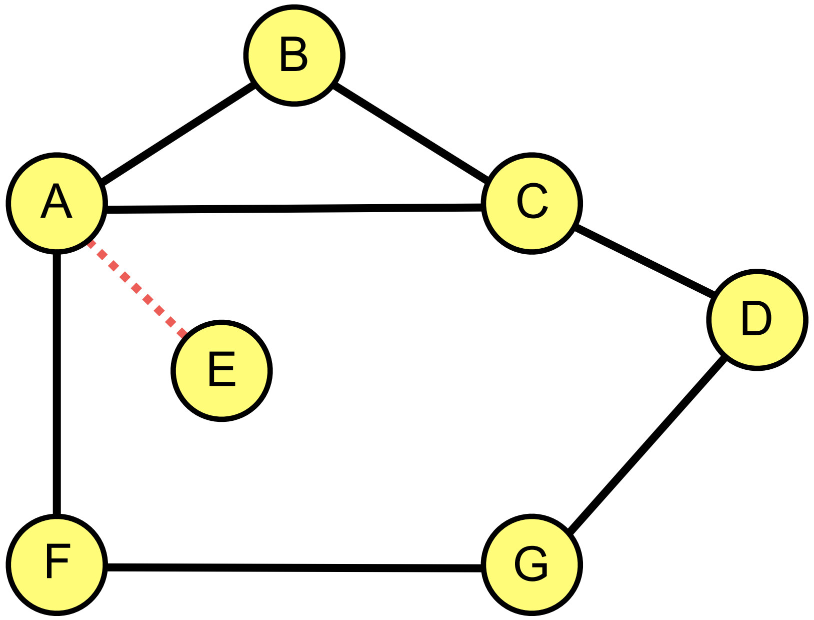
Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	3	C

Distance to Reach Node

Information Stored at Node

	A	B	C	D	E	F	G
A	0	1	1	2	1	1	3
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	3	1
E	1	2	2	3	0	2	4
F	1	2	2	2	2	0	4
G	2	3	2	1	3	4	0

Count to Infinity Problem



What if A-E link fails?

A advertises distance ∞ to E at the same time as C advertises a distance 2 to E (the old route via A).

B receives both, concludes that E can be reached in 3 hops via C, and advertises this to A. C sets its distance to E to ∞ and advertises this.

A receives the advertisement from B, decides it can reach E in 4 hops via B, and advertises this to C.

C receives the advertisement from A, decides it can reach E in 5 hops via A…

Loops, eventually counting up to infinity...

Solution 1: How big is infinity?

- Simple solution: `#define ∞ 16`
- Bounds time it takes to count to infinity, and hence duration of the disruption
- Provided the network is never more than 16 hops across!

Solution 2: Split Horizon

- When sending a routing update, do not send route learned from a neighbour back to that neighbour
  - Prevents loops involved two nodes, doesn’t prevent three node loops (like the previous example)
  - No general solution exists – distance vector routing always suffers slow convergence due to the count to infinity problem

Distance vector routing

- Count-to-infinity problem not solvable in general – implies distance vector algorithm only suitable for small networks

- Next lecture: link-state routing