University of Glasgow

School of Computing Science

Intra-domain Routing

Networked Systems (H)

Lecture 5

Lecture Outline

- Routing concepts
- Intra-domain unicast routing
  - Distance vector protocols
  - Link state protocols
  - Intra-domain routing in practice

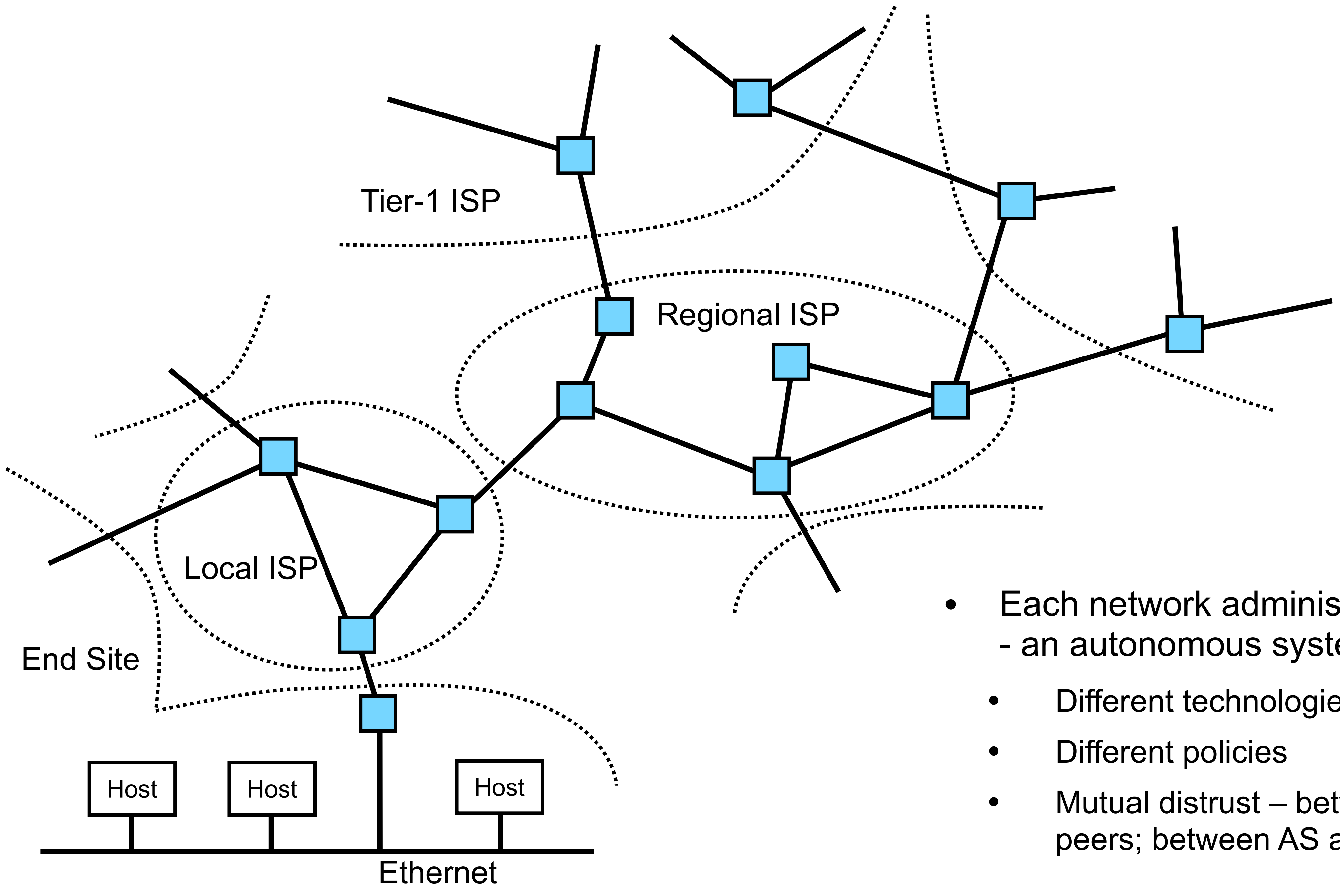
Routing

- 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

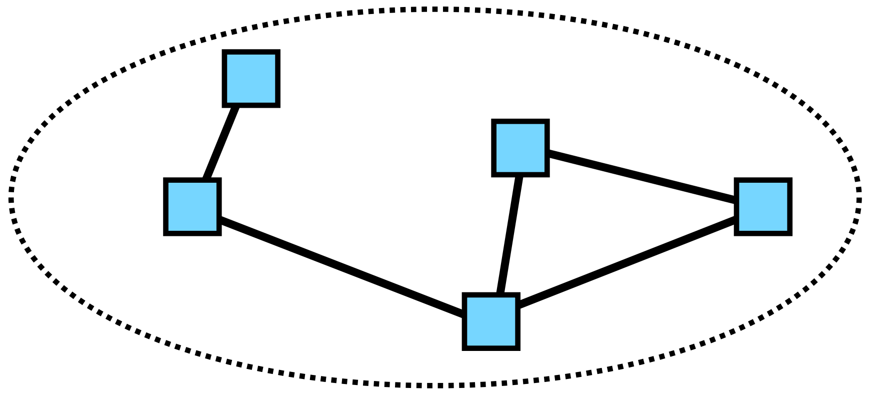
- Routing algorithms to deliver packets from a source to a single destination
- Choice of algorithm affected by usage scenario
    - Intra-domain routing
    - Inter-domain routing
    - Politics and economics

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

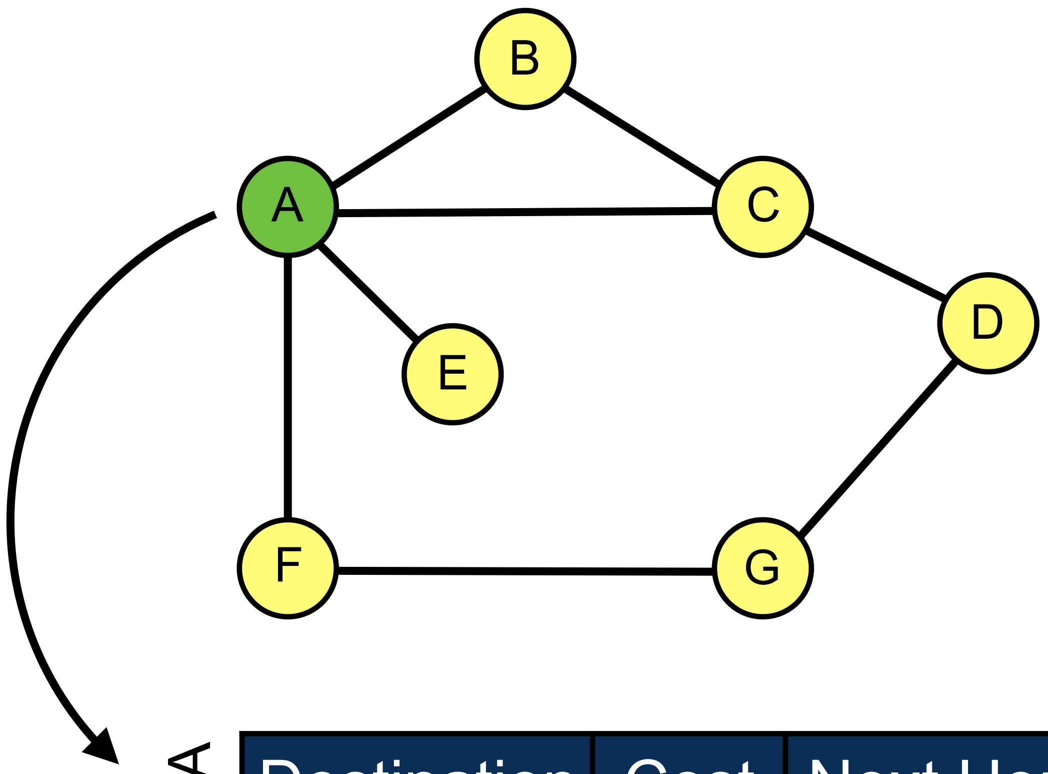
- Routing within an AS
  - Single trust domain
    - No policy restrictions on who can determine network topology
    - No policy restrictions on which links can be used
  - Desire efficient routing → shortest path
    - Make best use of the network you have available
  - Two approaches
    - Distance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

Intra-domain Routing: Distance Vector Protocols

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

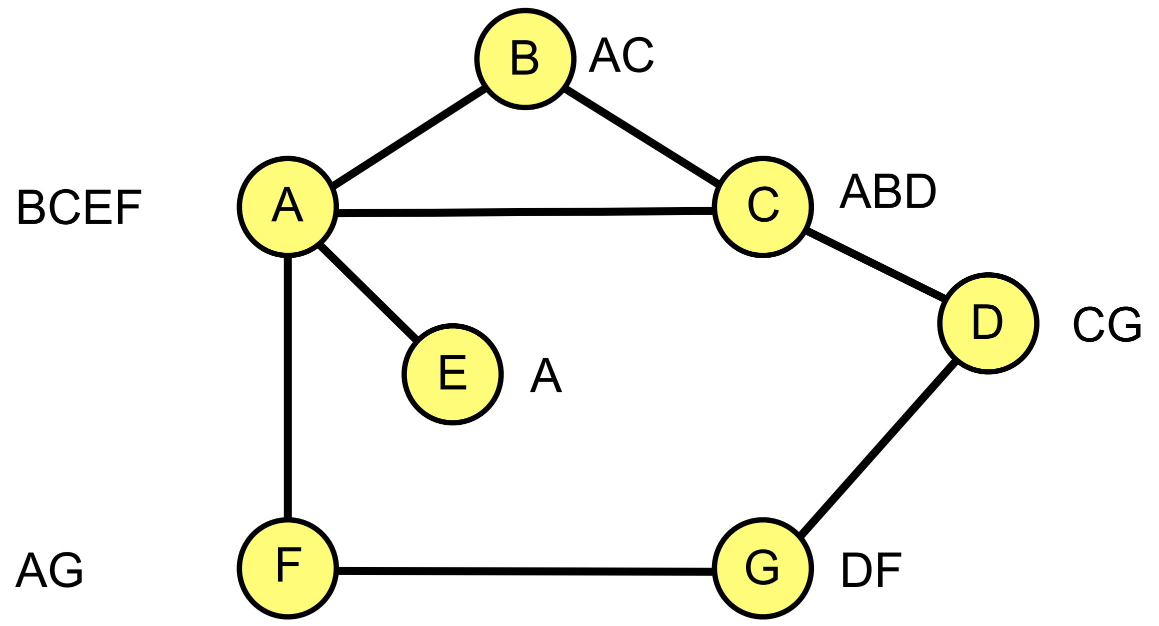
Destination	Cost	Next Hop
B	∞	-
C	∞	-
D	∞	-
E	∞	-
F	∞	-
G	∞	-

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

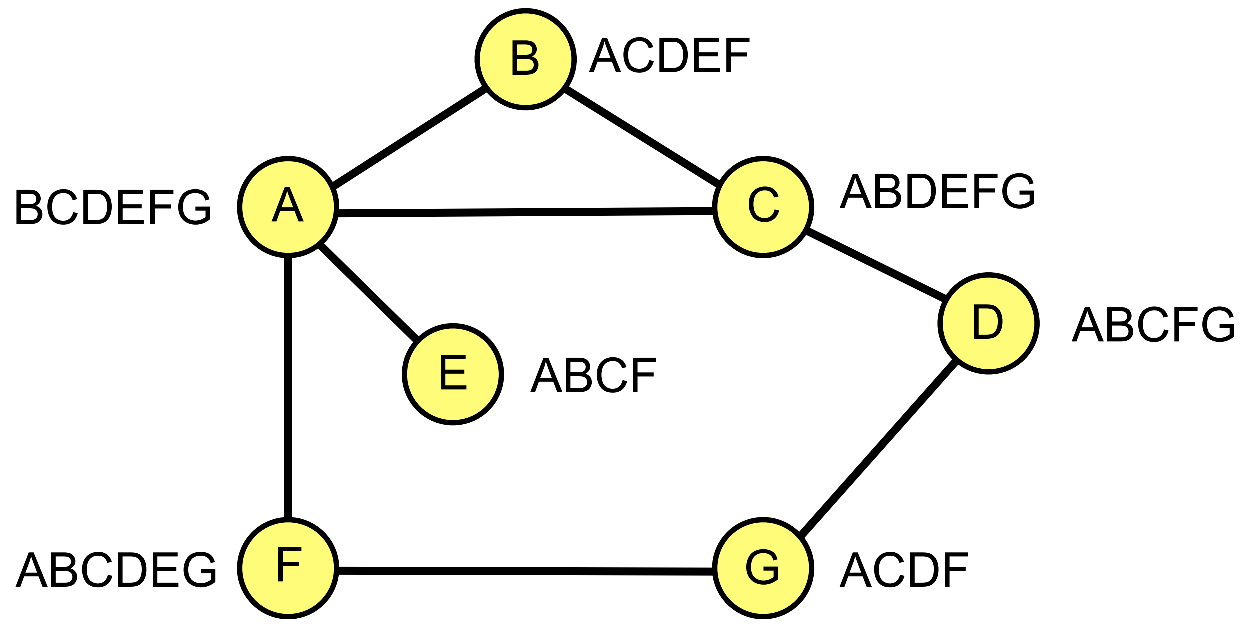
Destination	Cost	Next Hop
B	1	B
C	1	C
D	∞	-
E	1	E
F	1	F
G	∞	-

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

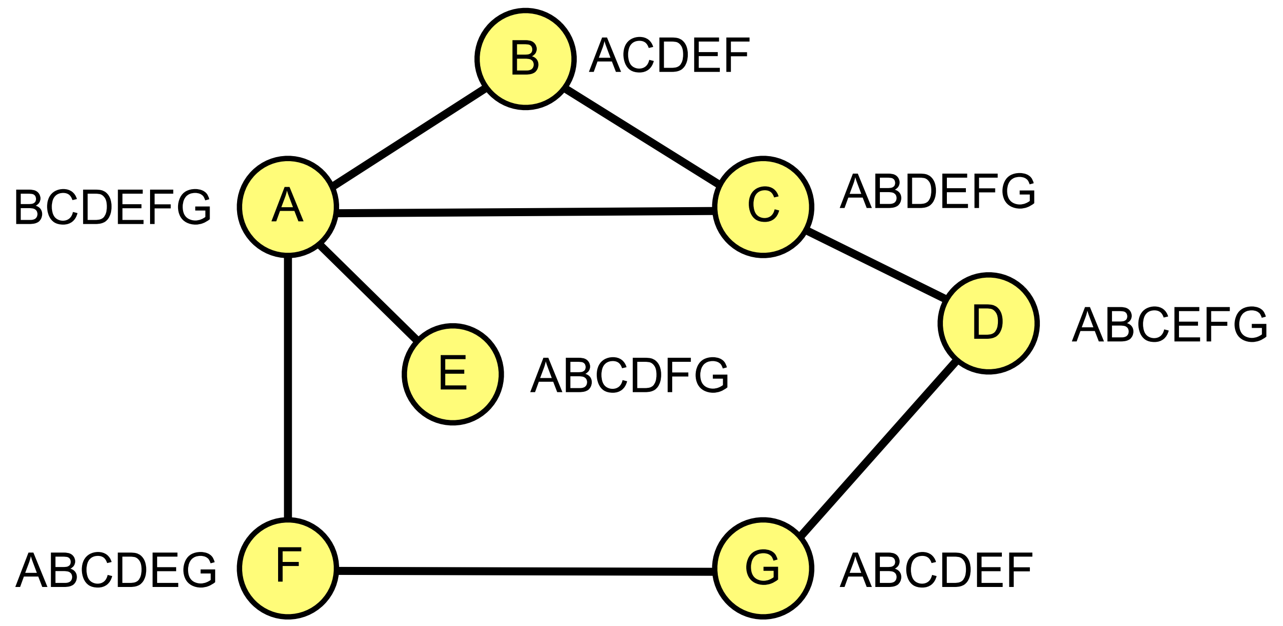
Routing Table at Node A

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

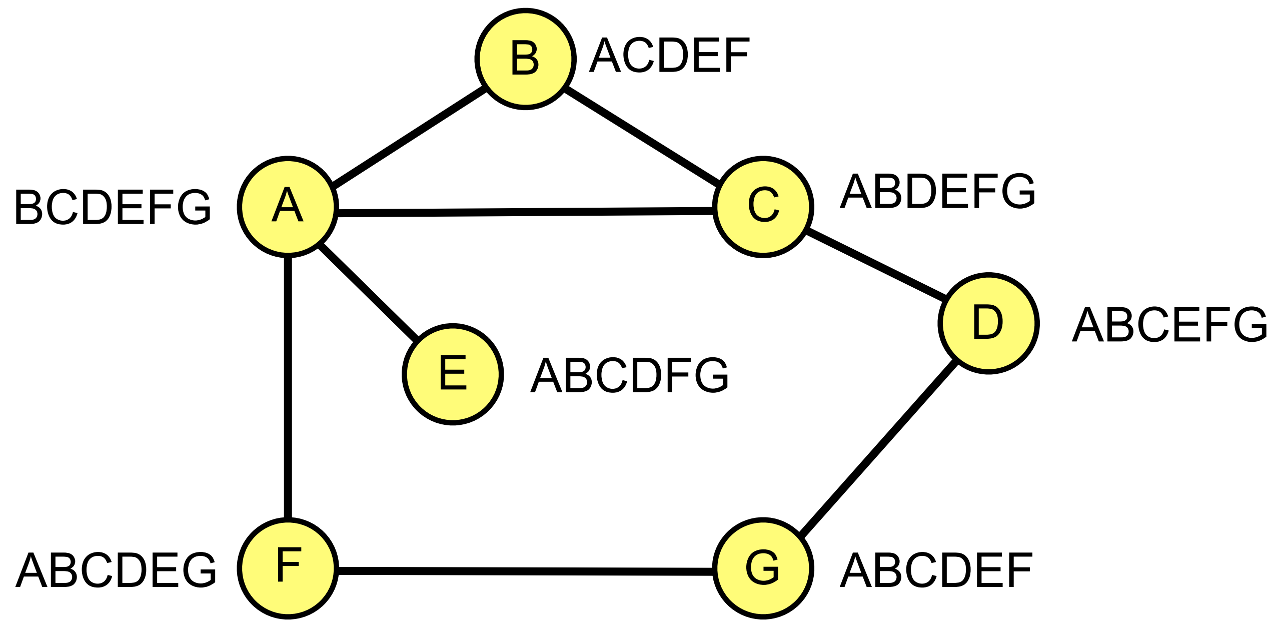
Routing Table at Node A

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 (rows: 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

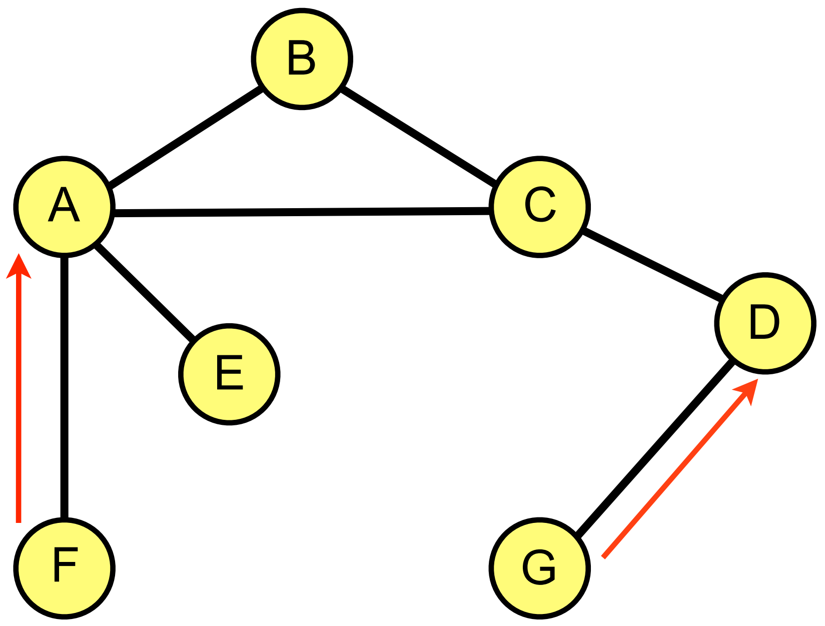
Routing Table at Node A

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

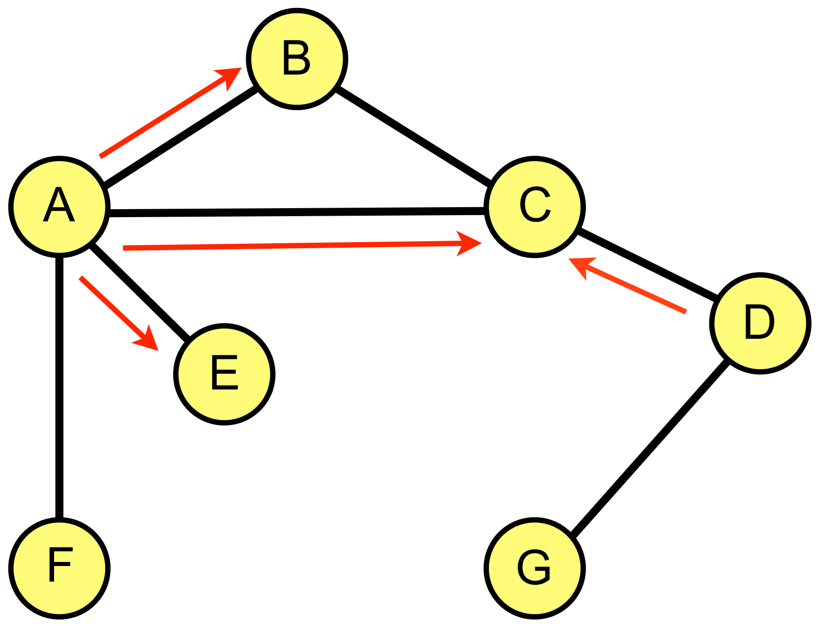
Routing Table at Node A

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

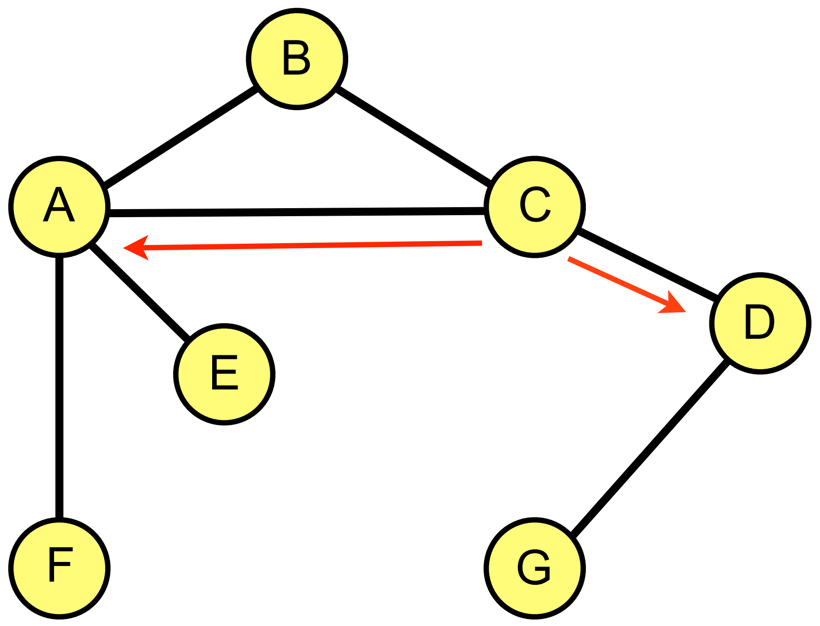
Routing Table at Node A

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

Information Stored at Node / Distance to Reach 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

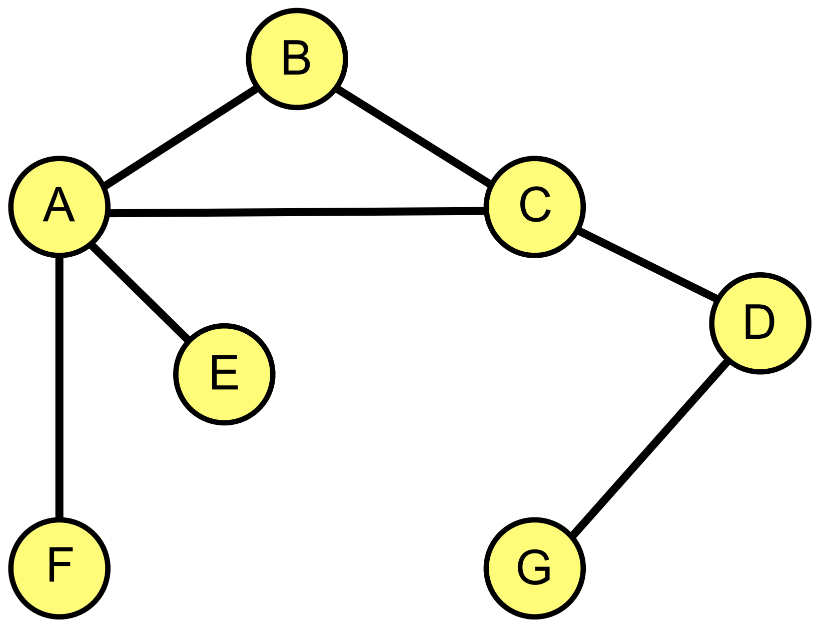
Routing Table at Node A

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

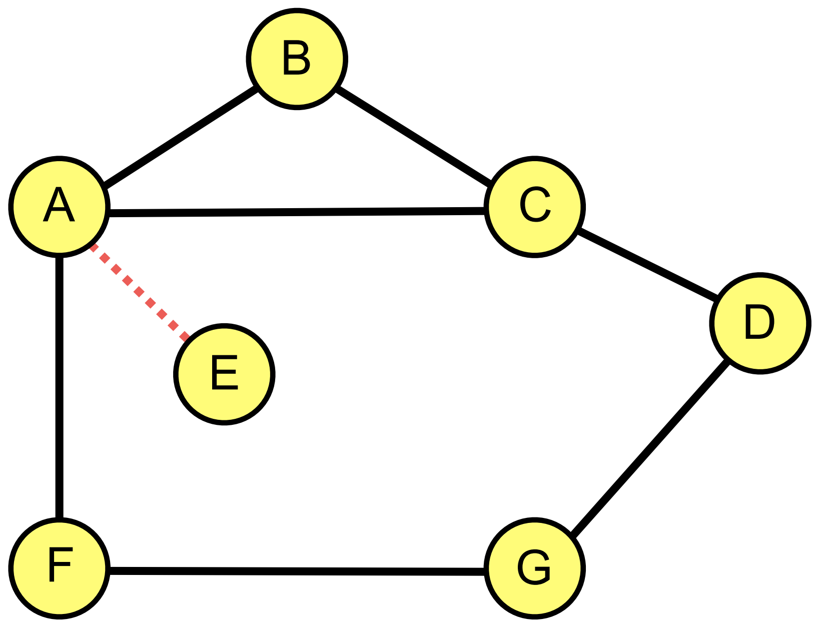
Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	3	C

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

Limitations of Distance vector routing

- Distance vector routing tries to minimise state at nodes – as a consequence, is slow to converge
- Count-to-infinity problem not solvable in general – implies distance vector algorithm only suitable for small networks

- An alternative is *link state* routing

Intra-domain Routing: Link State Protocols

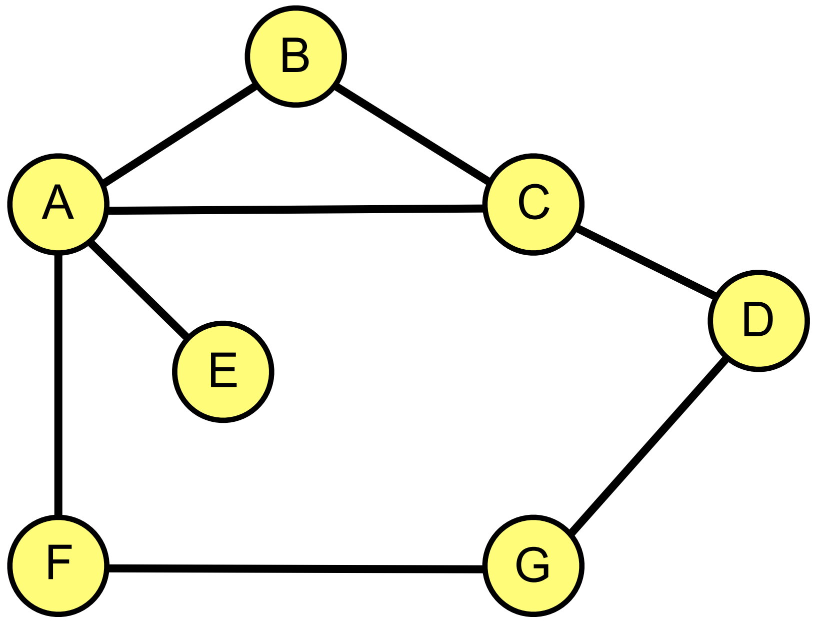
Link State Routing

- Nodes know the links to their neighbours, and the cost of using those links – the *link state* information
- Reliably flood this information, giving all nodes complete map of the network
- Each node then directly calculates shortest path to every other node, uses this as routing table

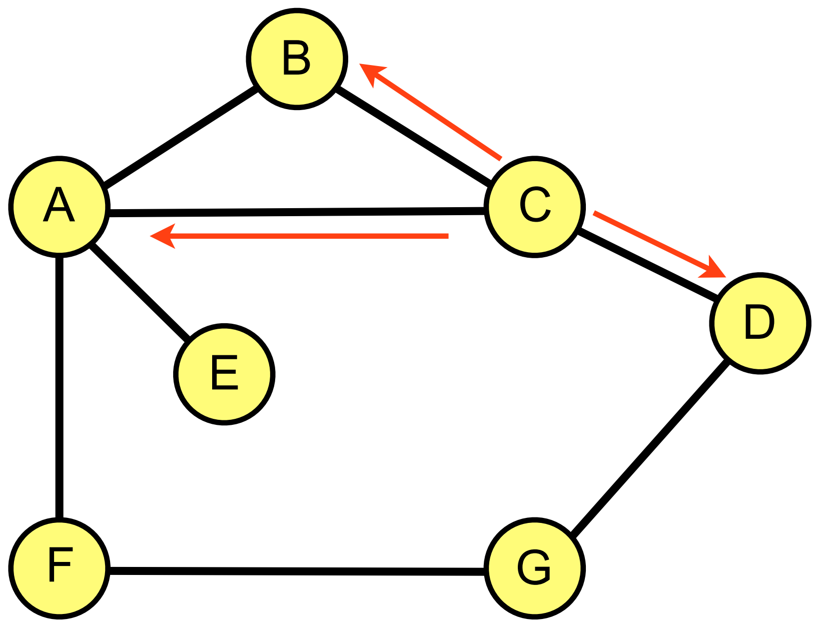
Link State Information

- Link state information updates are flooded on start-up, and when the topology changes
- Each update contains:
  - The address of node that sent the update
  - List of directly connected neighbours of that node
    - With the cost of the link to each neighbour
    - With the range of addresses assigned to each link, if it connects to more than one host
  - A sequence number

Flooding Link State Updates

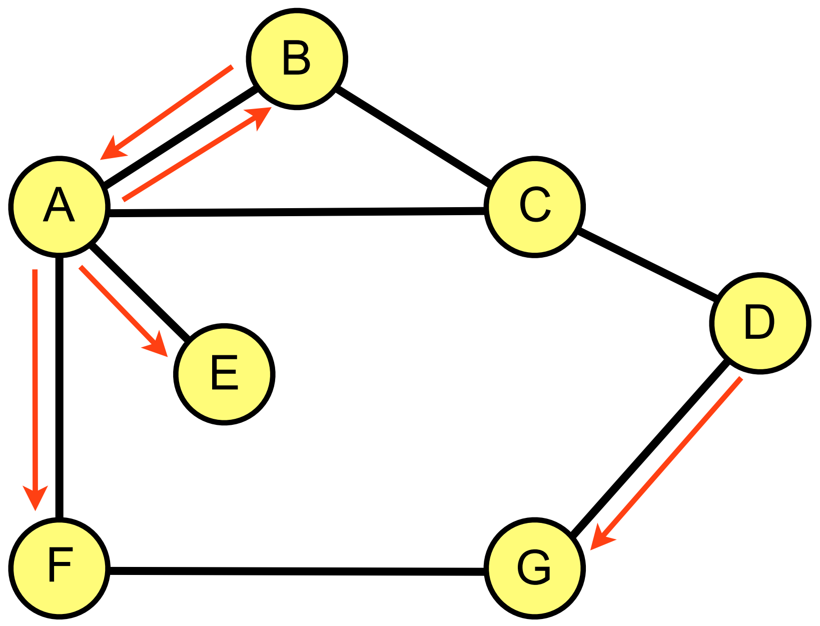


Flooding Link State Updates



Node C sends an update to each of its neighbours

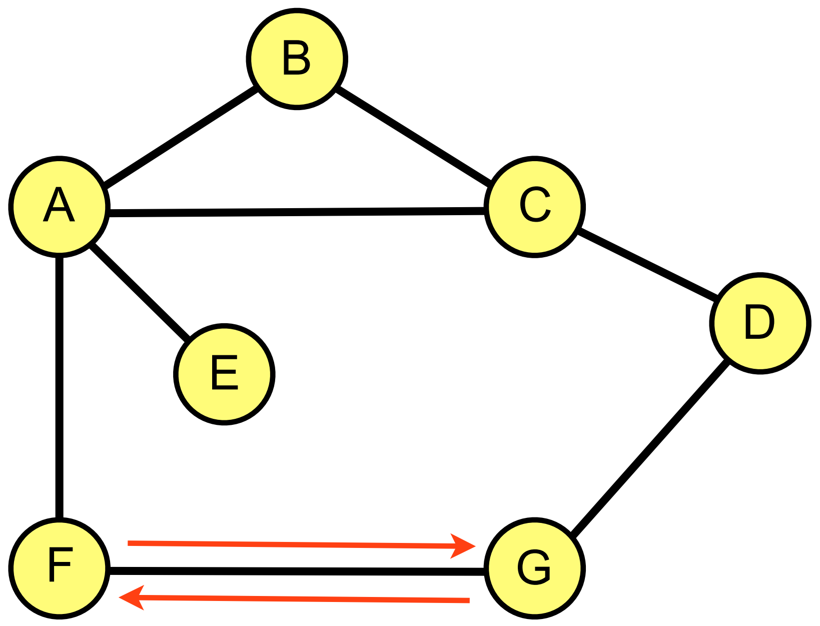
Flooding Link State Updates



Node C sends an update to each of its neighbours

Each receiver compares the sequence number with that of the last update from C, if greater it forwards the update on all links except the link on which it was received.

Flooding Link State Updates

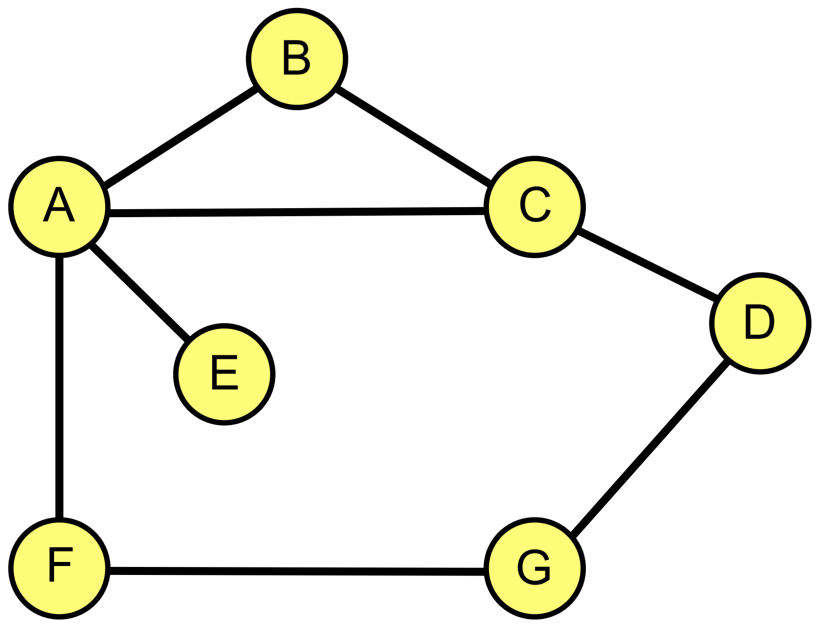


Node C sends an update to each of its neighbours

Each receiver compares the sequence number with that of the last update from C, if greater it forwards the update on all links except the link on which it was received.

Each receiver compares the sequence number with that of the last update from C, if greater it forwards the update on all links except the link on which it was received.

Flooding Link State Updates



Node C sends an update to each of its neighbours

Each receiver compares the sequence number with that of the last update from C, if greater it forwards the update on all links except the link on which it was received.

Each receiver compares the sequence number with that of the last update from C, if greater it forwards the update on all links except the link on which it was received.

Eventually, the entire network has received the update

Calculate Shortest Paths

- Flooding link state data from all nodes ensures all nodes know the entire topology
- Each node uses Dijkstra’s shortest-path algorithm to calculate optimal route to every other node
    - Optimal is assumed to be the shortest path, by weight

Shortest Path Algorithm

Definitions:

N	set of all nodes in the graph
$l(i, j)$	weight of link from *i* to *j* (∞ if no link, 0 if *i* = *j*)
s	source node from which we're calculating shortest paths

Dijkstra's Algorithm for an undirected connected graph:

```
M = {s}                                The set of nodes that have been checked
foreach n in N - {s}:                  The distance to directly connected neighbouring nodes
    C(n) = l(s, n)

while (N ≠ M):
    c = ∞
    foreach n in (N-M)
        if C(w) < c then w = n         Find node w such that C(w) is the minimum for all nodes in (N-M)

    M += {w}                           Add one node at a time, starting with the closest
    foreach n in (N-M):
        if C(n) > C(w) + l(w, n) then C(n) = C(w) + l(w, n)     Best route to n is via w
```

Result:

$C(x)$ cost of the shortest path from *s* to *x*

Forwarding and Route Updates

- Forward packets based on calculated shortest path
  - Static forwarding decision based on weights distributed by the routing protocol
  - Does not take into account network congestion

- Recalculate shortest paths on every routing update
  - Updates occur if a link fails, or a new link is added

Distance Vector *vs* Link State

Distance vector routing:

- Simple to implement
- Routers only store the distance to each other node: O(n)
- Suffers from slow convergence

Link State routing:

- More complex
- Requires each router to store complete network map: O(n^2)
- Much faster convergence

Slow convergence times make distance vector routing unsuitable for large networks

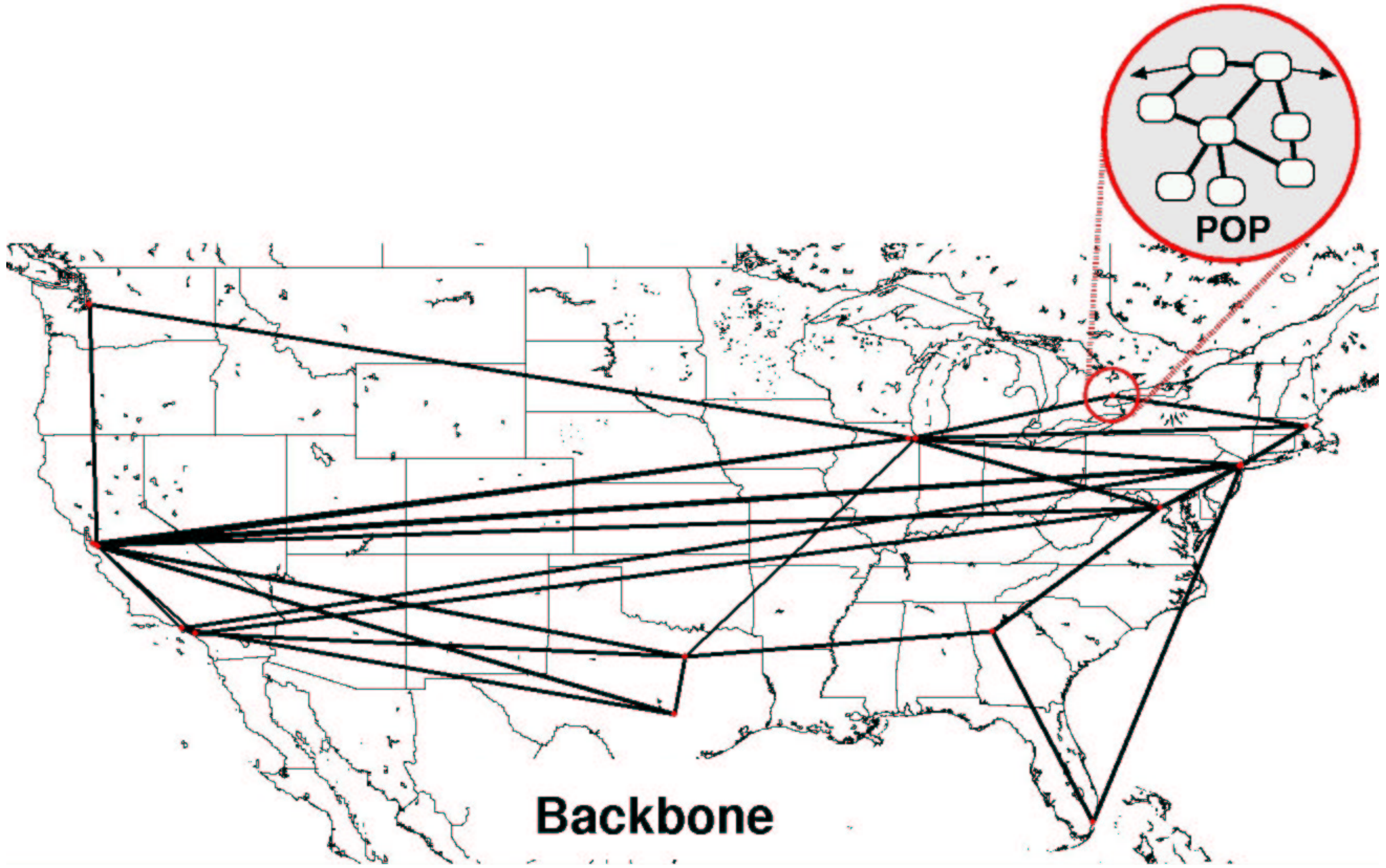
Intra-domain Routing in Practice

Where is intra-domain routing used?

- Intra-domain routing operates within a network
  - Any network operated as a single entity – autonomous system – could be local area, nationwide, or even worldwide
  - Operates a single routing protocol – typically the OSPF link-state protocol exchanging routes to IP address prefixes
  - Running on IP routers within an autonomous system, typically with fibre connections wide area and ethernet local area
  - Exchange routes to IP prefixes, representing regions in the network topology

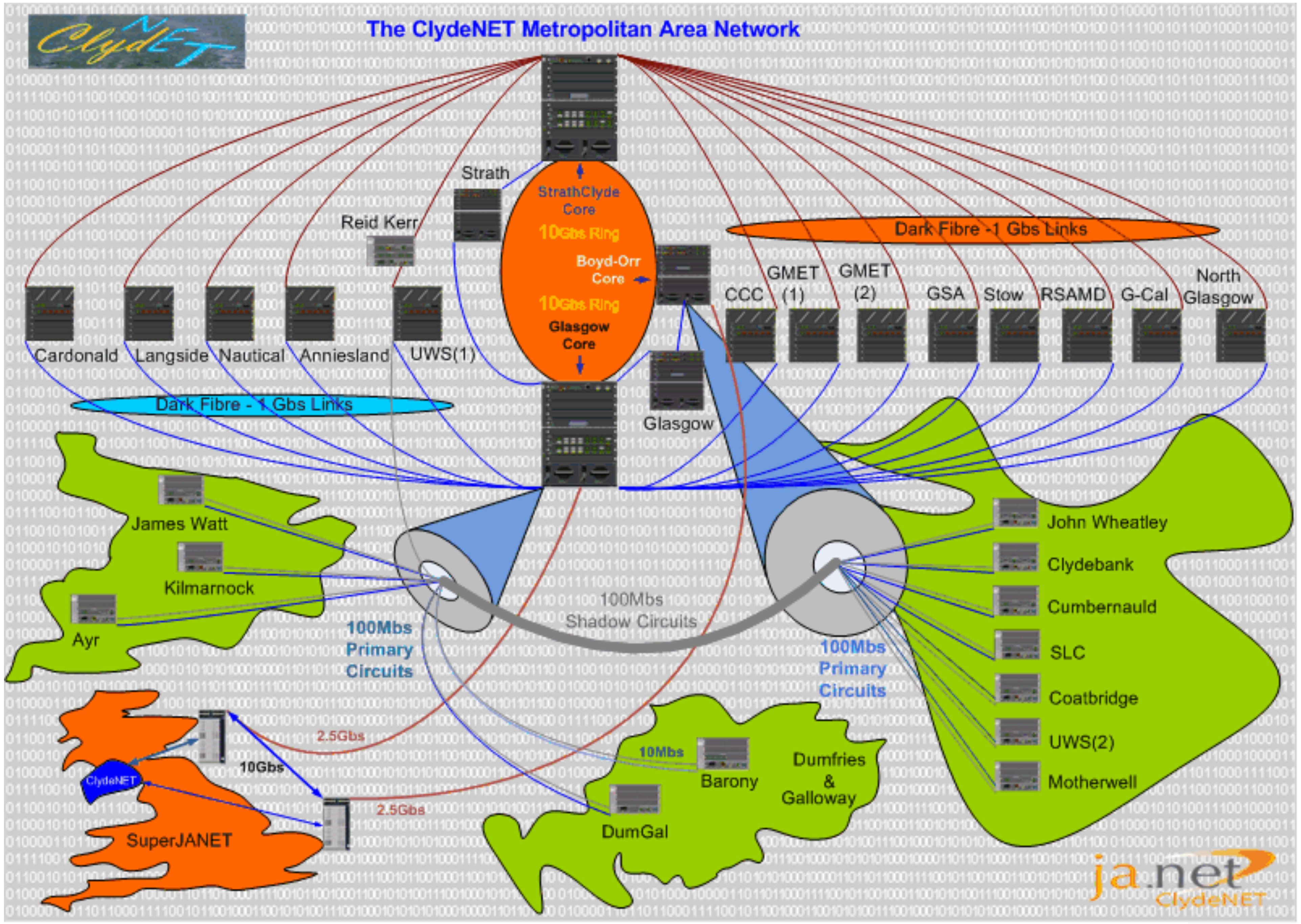
What is the Topology of Real Networks?

•



N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Network Topologies with Rocketfuel", Presentation at ACM SIGCOMM conference, 2002

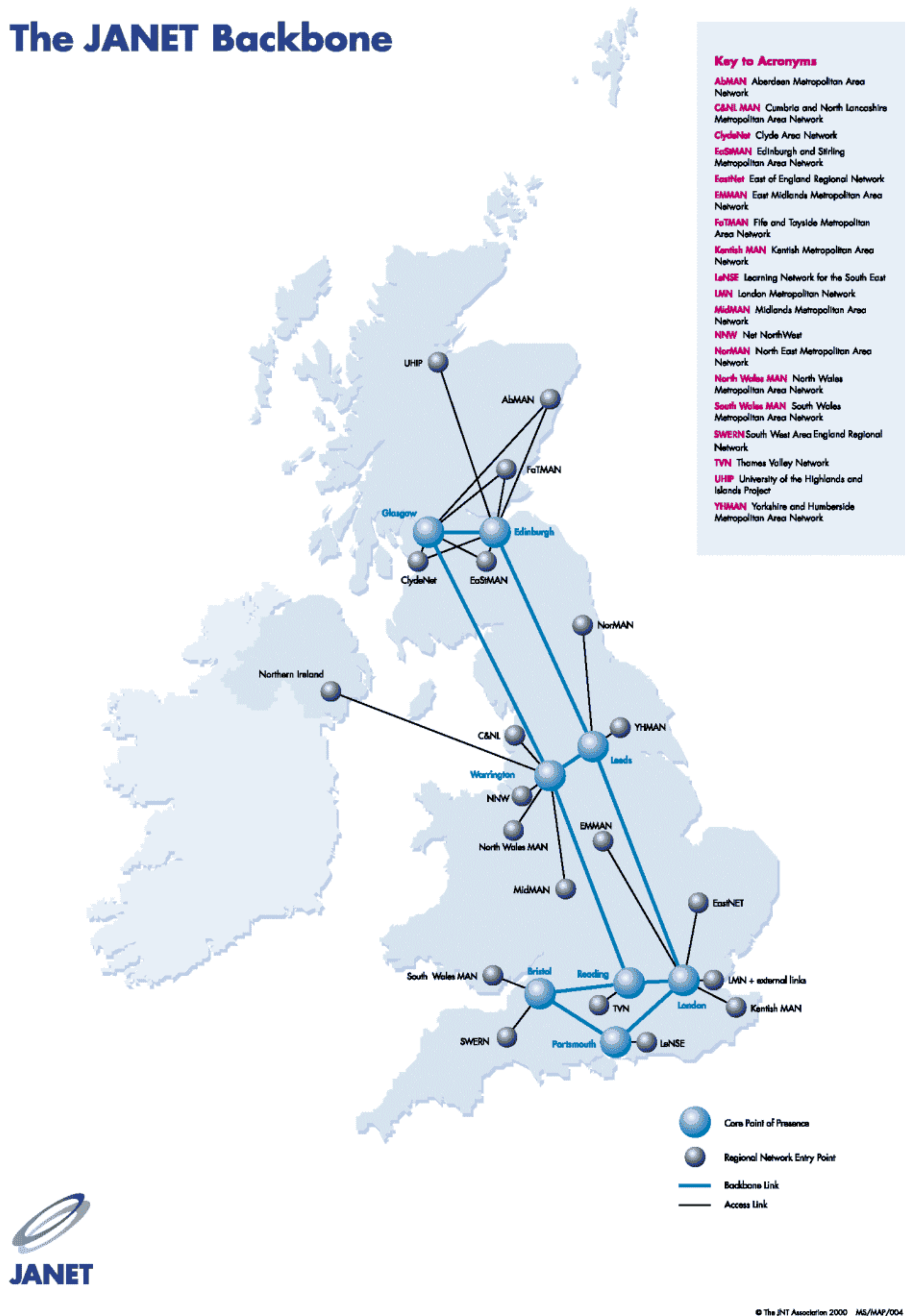
Example: ClydeNet



ClydeNet is the metropolitan area network for the Glasgow region – the regional JANET PoP

Example: The JANET Backbone (2001)

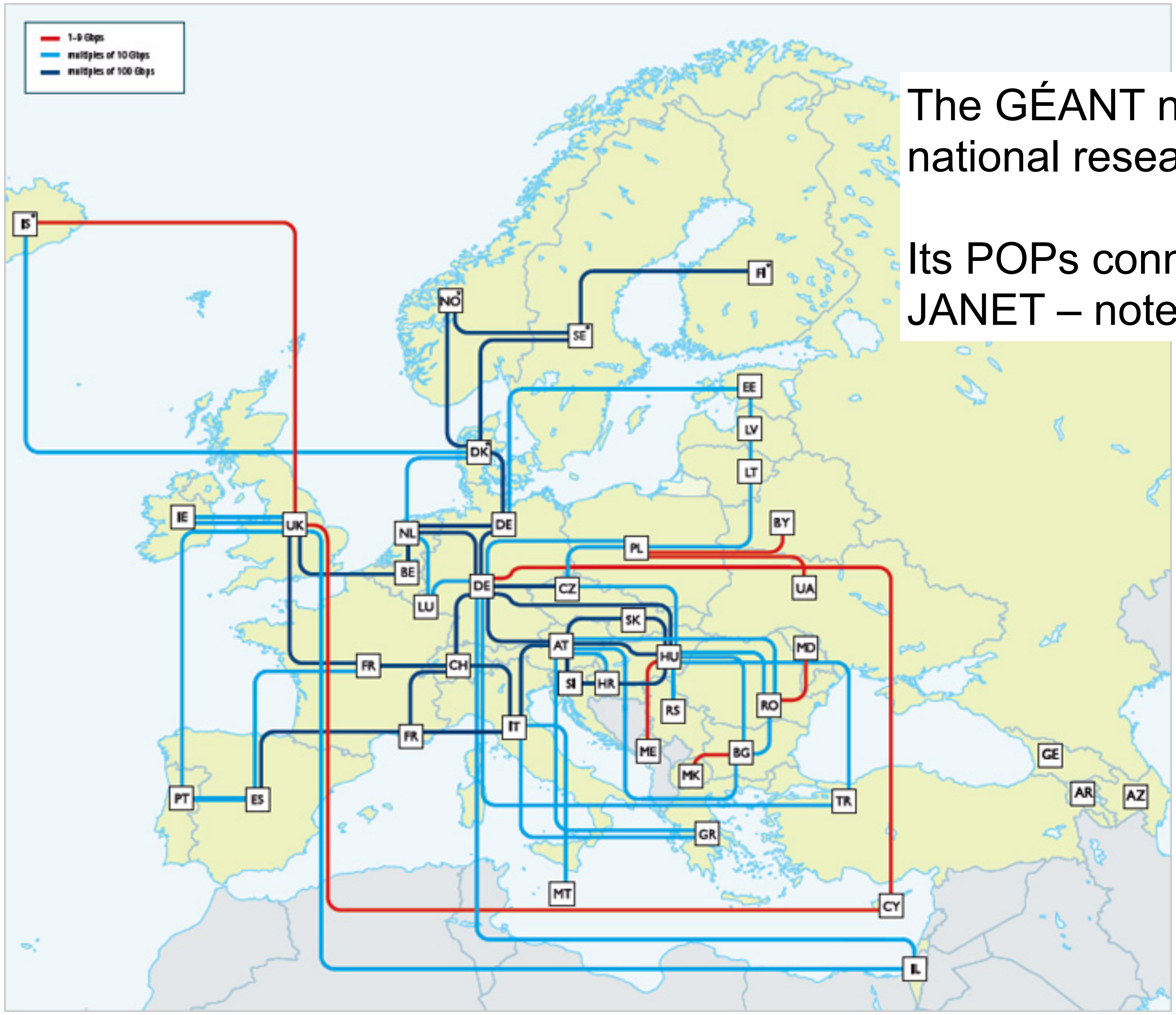
- JANET is the UK national research network
- It interconnects major universitie and metropolitan area networks - its customers are a mix of end sites and other networks



Key to Acronyms

AbMAN	Aberdeen Metropolitan Area Network
C&NL MAN	Cumbria and North Lancashire Metropolitan Area Network
ClydeNet	Clyde Area Network
EaStMAN	Edinburgh and Stirling Metropolitan Area Network
EastNet	East of England Regional Network
EMMAN	East Midlands Metropolitan Area Network
FaTMAN	Fife and Tayside Metropolitan Area Network
Kentish MAN	Kentish Metropolitan Area Network
LeNSE	Learning Network for the South East
LMN	London Metropolitan Network
MidMAN	Midlands Metropolitan Area Network
NNW	Net NorthWest
NorMAN	North East Metropolitan Area Network
North Wales MAN	North Wales Metropolitan Area Network
South Wales MAN	South Wales Metropolitan Area Network
SWERN	South West Area England Regional Network
TVN	Thames Valley Network
UHIP	University of the Highlands and Islands Project
YHMAN	Yorkshire and Humberside Metropolitan Area Network

Example: GÉANT



The GÉANT network inter-connects the national research networks in Europe

Its POPs connect to networks such as JANET – note almost fractal structure

Summary

- Distance vector *vs* link state routing
- Local, metropolitan, or wide area – key is routing within a single autonomous system