COMP3310/6331 – #18

Routing

Dr Markus Buchhorn: *markus.buchhorn@anu.edu.au*

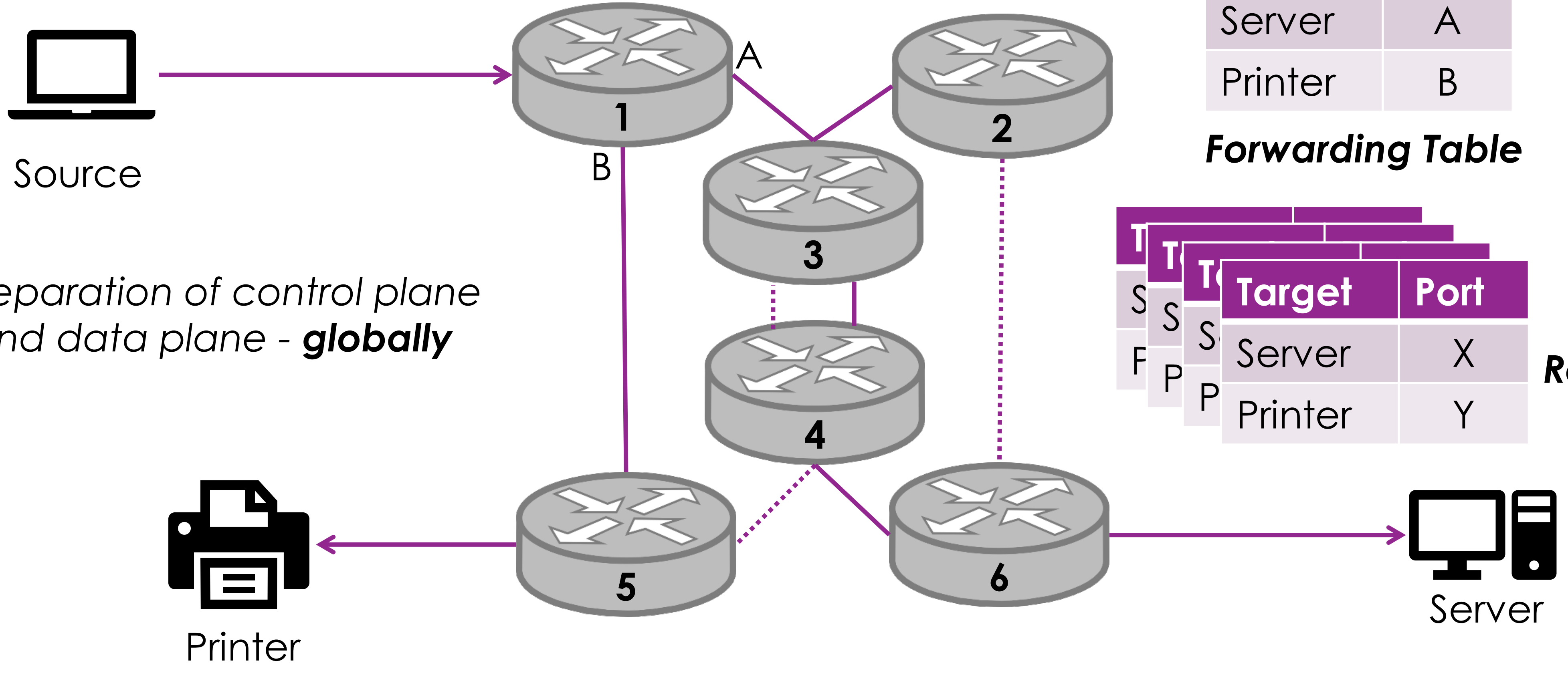
The biggest application of all?

- Routing
  - How do packets get across the Internet?
  - Without it there is no Internet

- Most complex, longest-running application ever?
  - Millions of devices
  - Running 24/7
  - Shifting Pb/s of traffic
  - Dealing with multiple topology changes every second

- And crosses from technology to humanity
  - ‘Optimal’ for what or who

Packet Forwarding and Routing



Router 1: Next Hops

Target	Port
Server	A
Printer	B

Forwarding Table

Target	Port
Server	X
Printer	Y

Routing

*Separation of control plane and data plane - **globally***

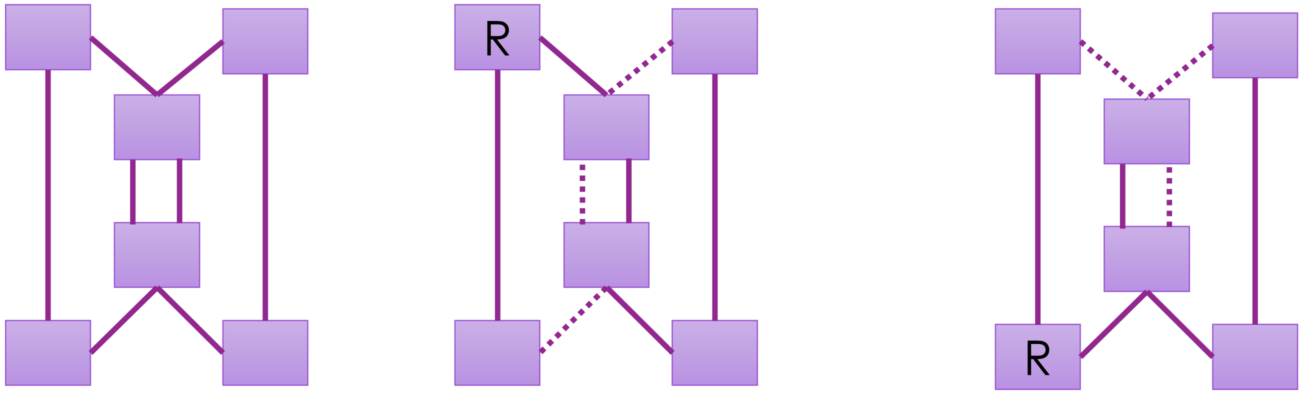
Back into the network layer

- Distinction between forwarding and routing
  - Local decisions vs global decisions
  - Given a network with multiple paths/interfaces, which one do you send to?

- Focus on **unicast** routing
  - Opposed to broadcast, multicast, anycast routing

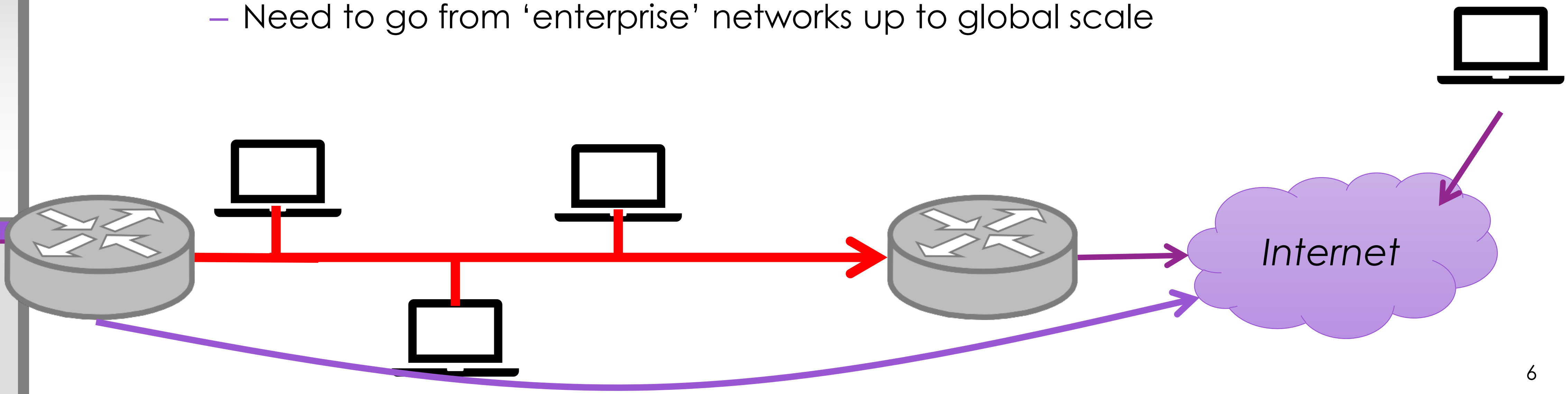
Spanning Tree = routing?

- An wide-area view that spans a network
  – Removes loops, establishes reliable paths
  – But only runs at layer-2 (single-technology), and doesn't scale
  – Wastes paths
    - No measure of 'quality' of a path
    - Doesn't use redundant paths when beneficial!



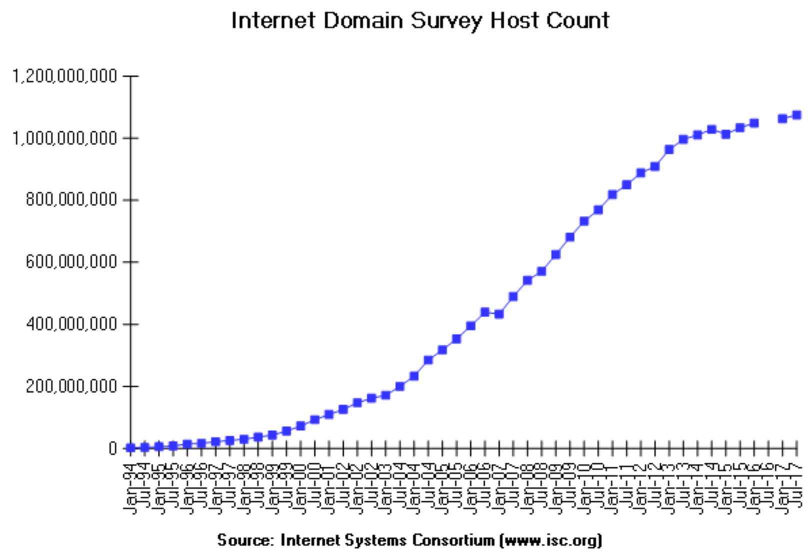
From local to global

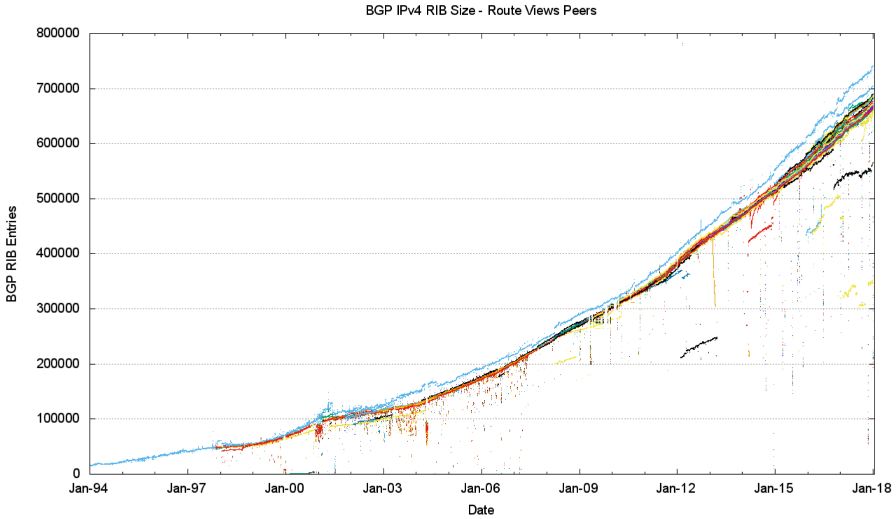
- Locally find devices via ARP, but that doesn't scale.
- But then what on the WAN?
  - LAN to WAN, a single default route?
    - Can have backup paths
  - Routers advertise a prefix (subnet) – aggregation!
  - Need to go from 'enterprise' networks up to global scale



Global routing is hard

- "Routing table" sizes – <u>growing</u> (1M+)
- Updates – <u>growing</u> (170k/day)
- Computing forwarding tables – <u>growing</u>
- Routers – <u>used to be simple</u> computers
  - 100Gbps = one small IP packet every 5 nanoseconds.
  - *"Performing a lookup into a data structure of around one million entries for an imprecise match of a 32-bit value within 5 nanoseconds represents an **extremely challenging** silicon design problem."*





Carrier-grade routers

“Routing” at different timescales

- Routing = sending some traffic over some path at some time
  - It’s allocating bandwidth across the network
  - While adapting to requirements and changing conditions

What you’re doing	How quickly	Because
Forwarding/ “load-sensitive” routing	Seconds	Bursts, Congestion
Routing	Minutes	Changes, failures
Traffic Engineering	Hours	Long-term load
Provisioning	Months	Customer demand

Expectations

- Routing has to work "right", all the time

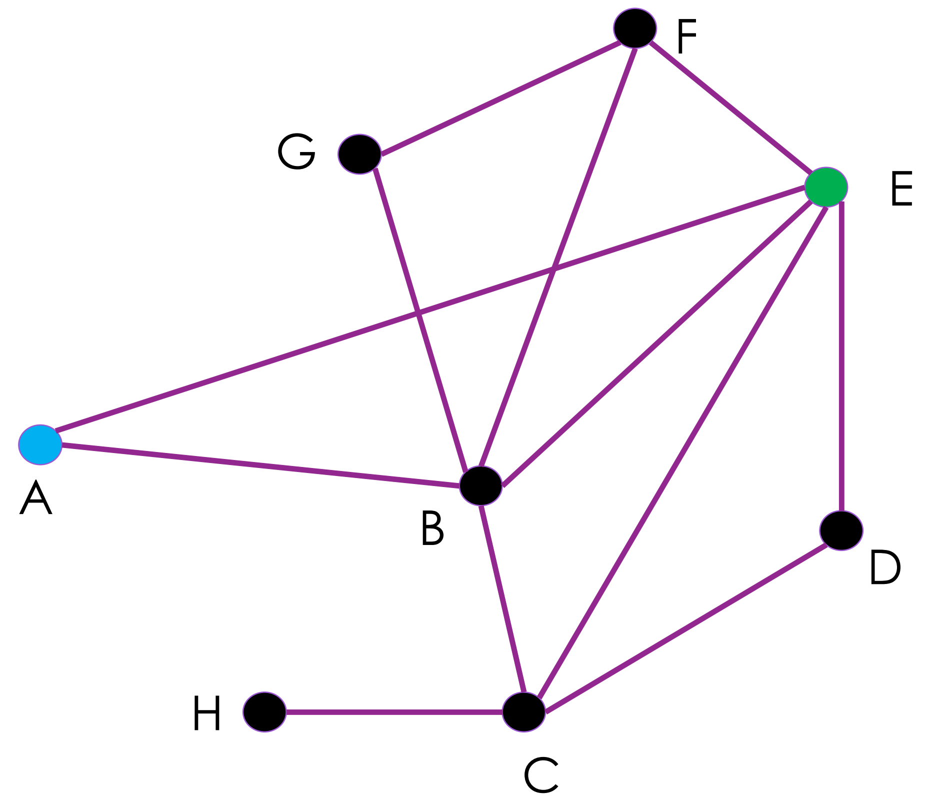
Expect	Because
Correctness	It has to get packets from A to B
Efficiency	Use available bandwidth well
Fairness	Don't ignore capable network elements
Convergence	Recover quickly from any disturbances
Scalability	Copes with increasingly large and complex networks

Routing context – ideally:

- Decentralised, no controller, no hub
  - Up to a point…
- All nodes (routers) are alike
  - Speak the same language, run the same algorithm, at the same time
- Learn through message exchanges with "neighbours"
- Need to deal with router, link and message failures

What is the ‘best’ route?

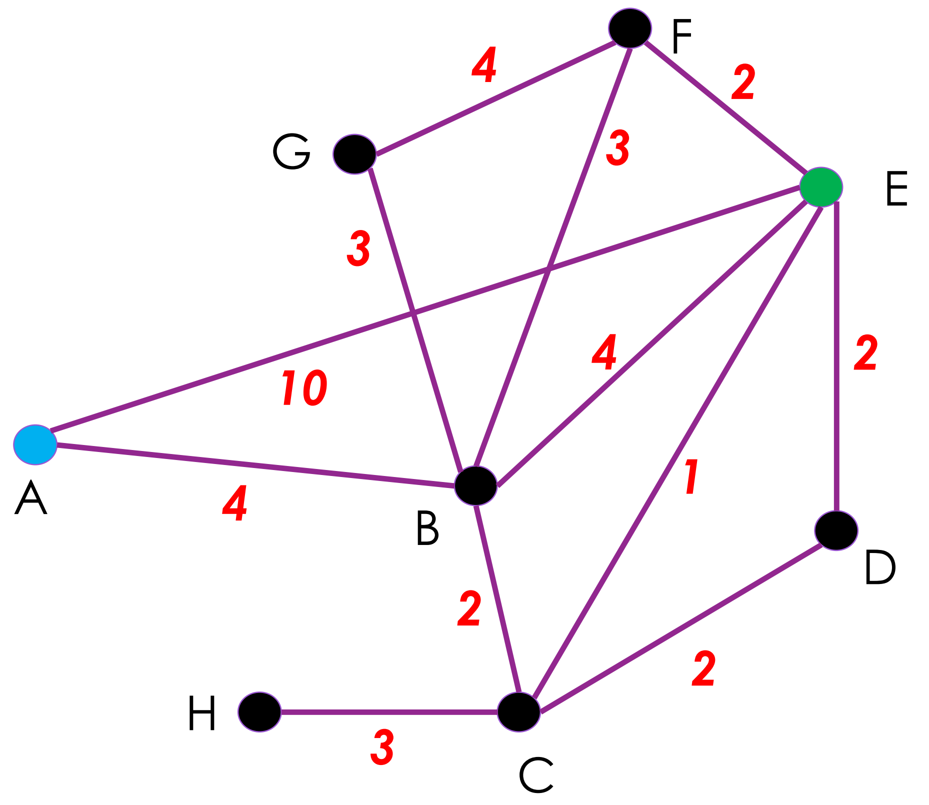
- Various measures of ‘best’
  - Latency (delay = distance)
  - Bandwidth (slow)
  - Cost (money)
  - Hops (forwarding delays)



- For a <u>fixed</u> topology
  - Ignores link congestion
  - Ignores router load

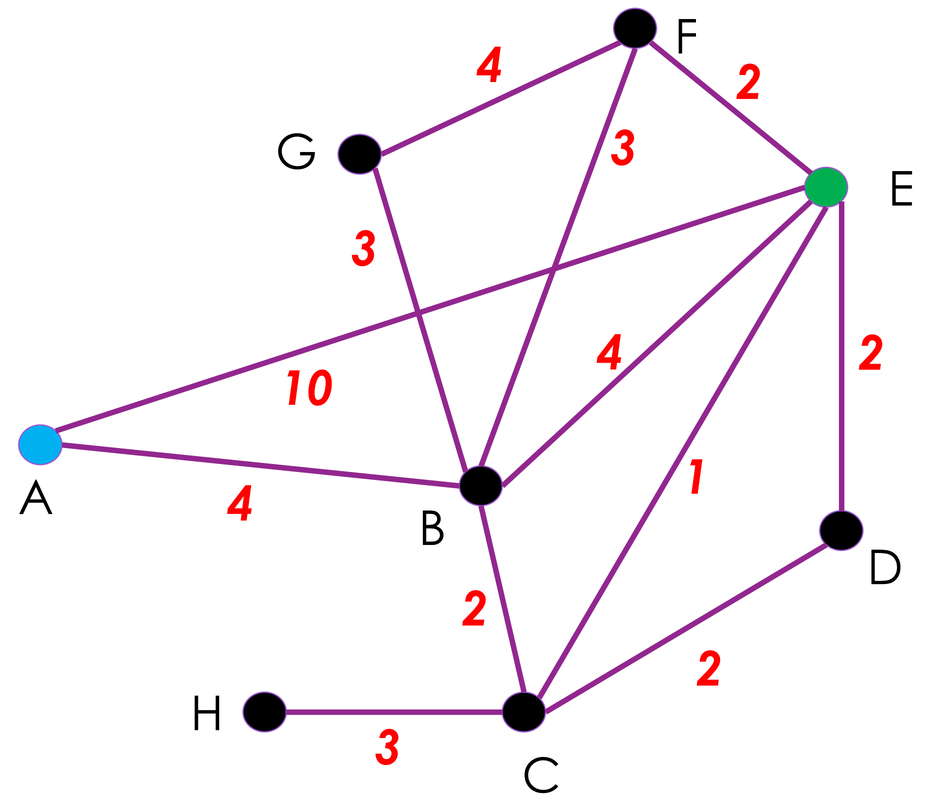
Shortest-path routing

- AKA “lowest-cost” path routing
- Associate some **cost** with each link
  - *You choose: $$, ms, hops, bps, …*
  - *In each direction – can be asymmetric*
- Add up the total end-to-end
- And try to minimise the total
  - If tied, pick one



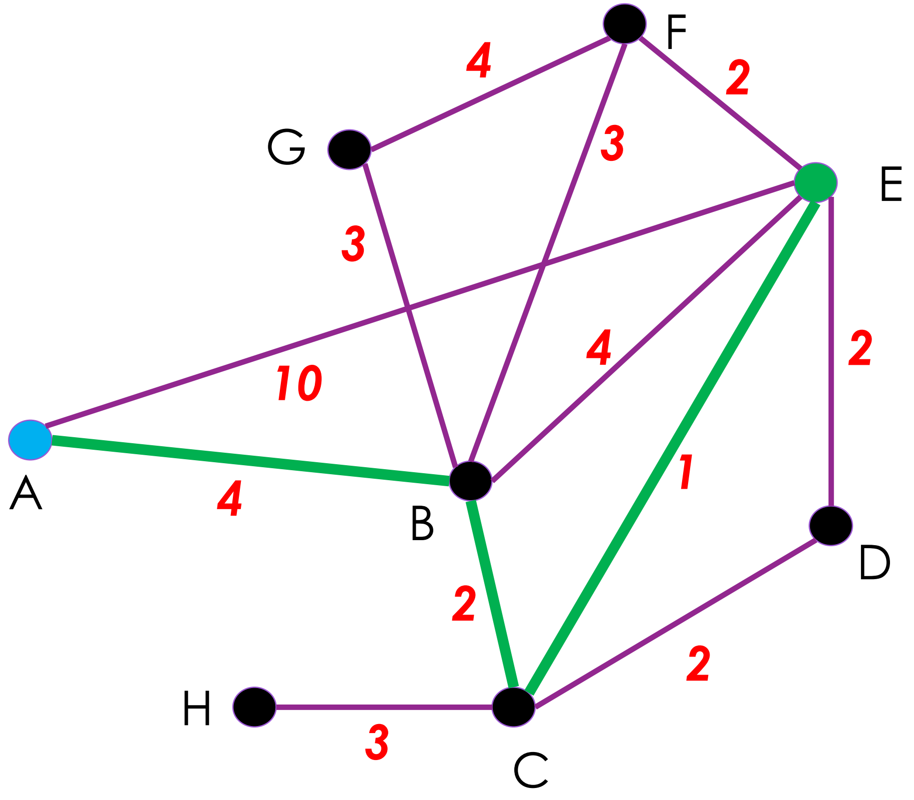
Eyeball analysis

- Shortest (lowest cost) path A to E?
- AE = 10
- ABFE = 9
- ABE = 8
- **ABCE = 7**



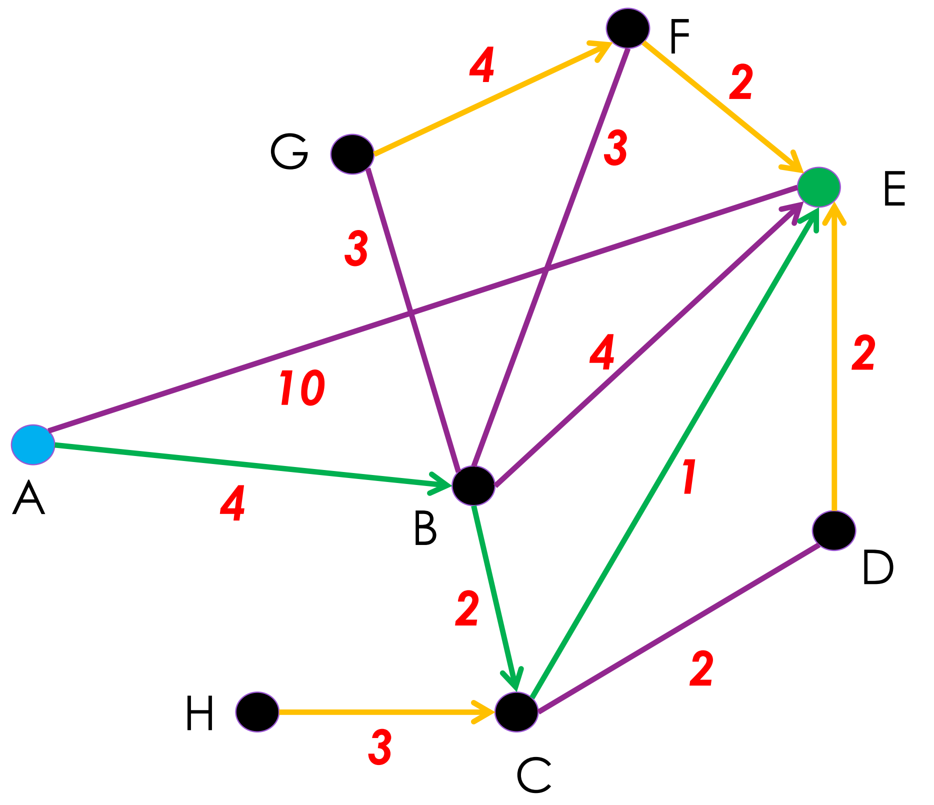
Optimality property

- *Sub-paths of the shortest path are themselves shortest paths*
- **ABCE = 7 = shortest path**
- So are
  - AB, ABC
  - BC, BCE
  - CE
- If there's a shorter sub-path, it'd be on the shortest total path



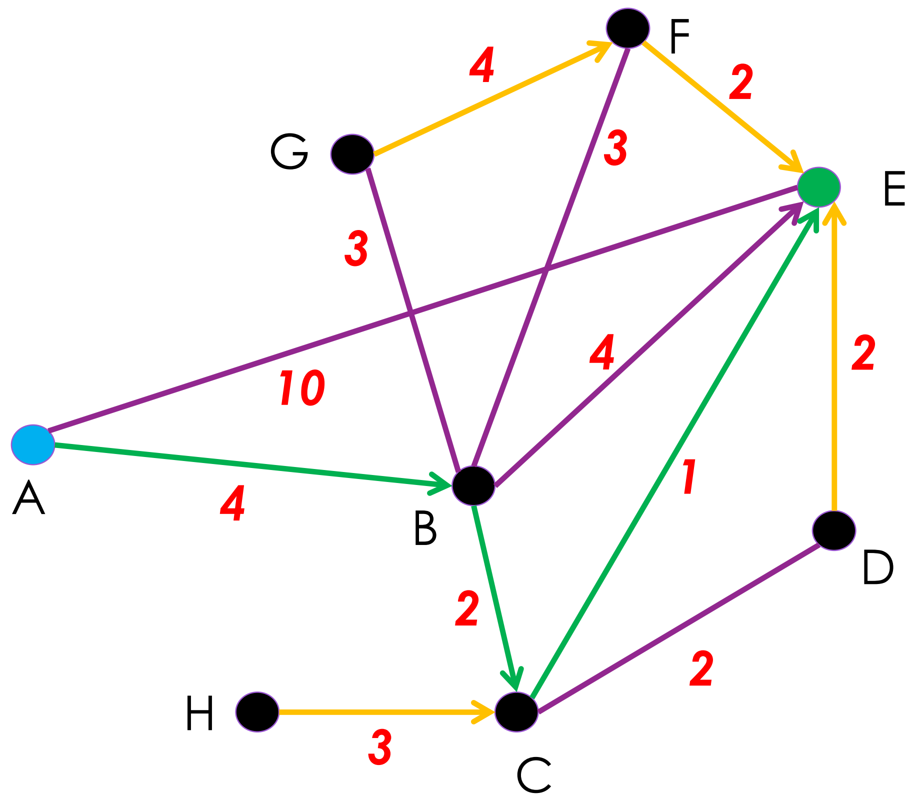
Sink (and source) trees

- *Union of all shortest paths towards a node from each source*
- Consider E's sink tree.
  - ABCE = shortest for A, B, C
  - Work out the rest
- Source tree = sink tree
  - Usually, but doesn't have to.
    - Asymmetric costs => different trees
    - What happens if BC=2, CB=5?



So what?

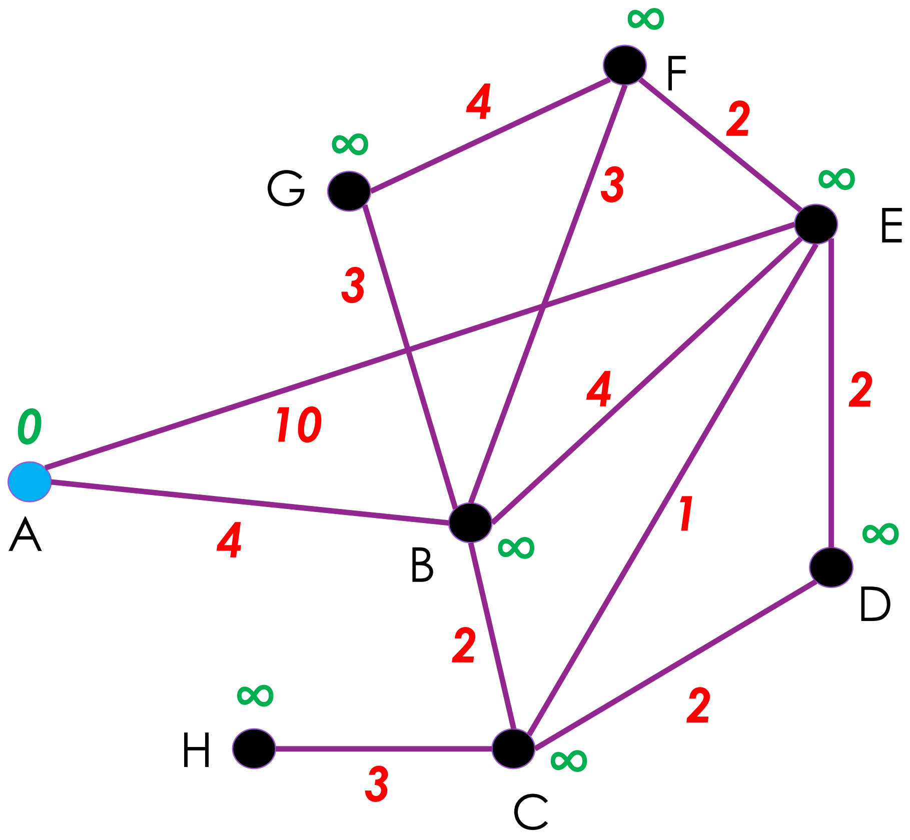
- Regardless of where you start, routing decisions only considers **destination**
  - Source is irrelevant (*)
    - A, B, H get to C – and then to E
- Each node only needs to know **next hop on the optimal path** => forwarding table
- Forwarding table =
  - Next hop for every destination



Computing shortest paths

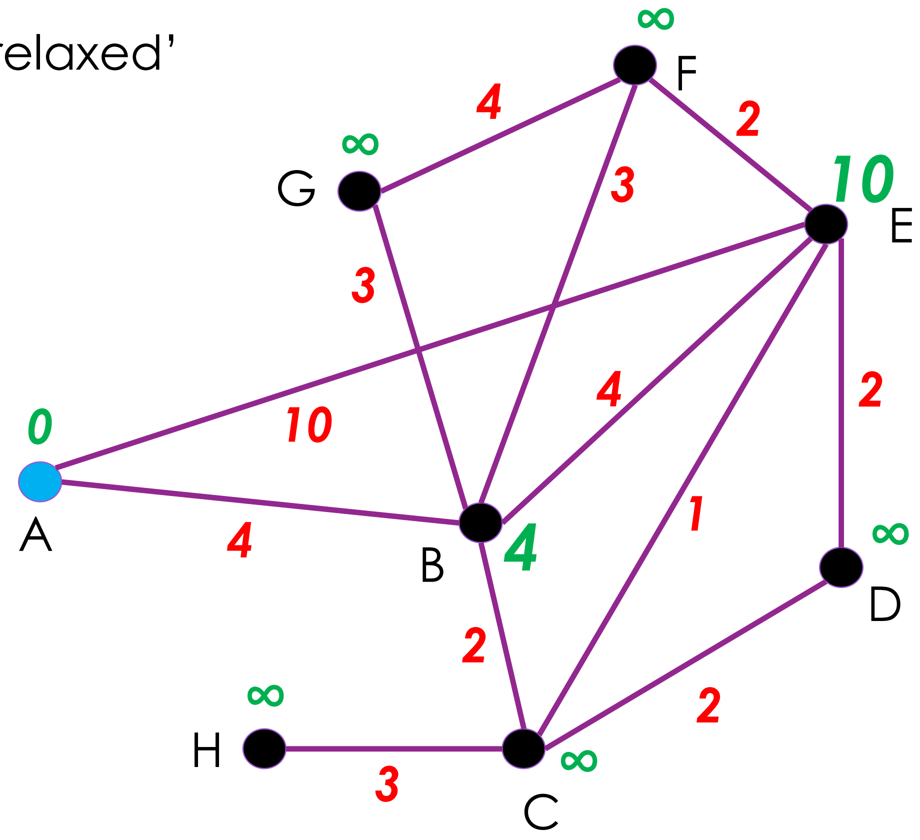
- Eyeballs are good, computers are better…
- (Edsger W.) Dijkstra's algorithm, 1959
  - Identifies source tree for a single source, when **given** the topology and costs
  - Uses Optimality Property
- Algorithm outline:
  - Start at source node, mark all others as 'tentative'
  - Give source "0" node-cost, everyone else is "infinite" cost
  - Loop: while (tentative nodes)
    - Identify lowest-cost node, **confirm** it
    - Add link to source tree
    - Modify ('relax') other costs by distances you now know

Walkthrough – tree from A (calculated by A)



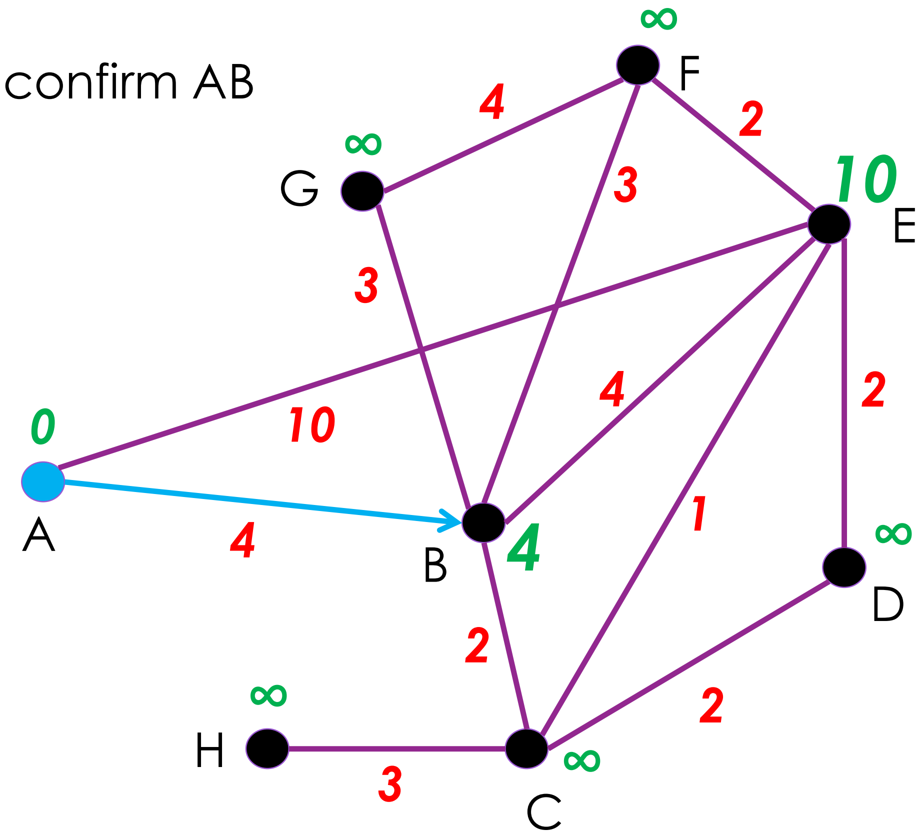
Walkthrough – tree from A

Neighbours of A get ‘relaxed’
(costs are reduced)



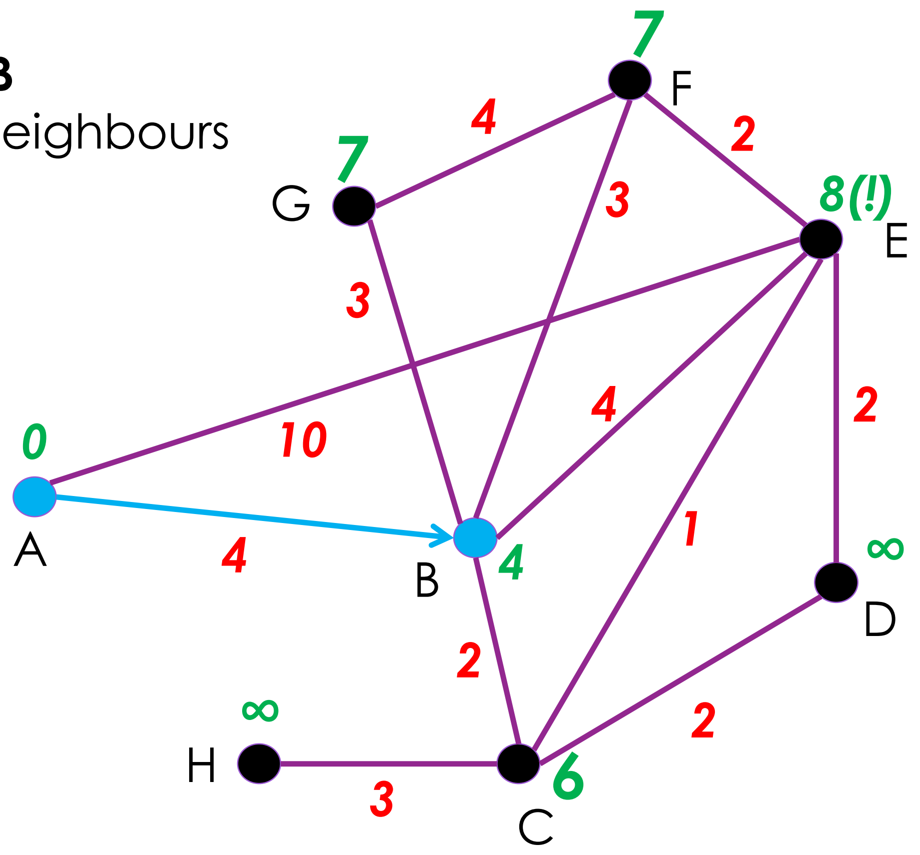
Walkthrough – tree from A

B is lowest node-cost, confirm AB



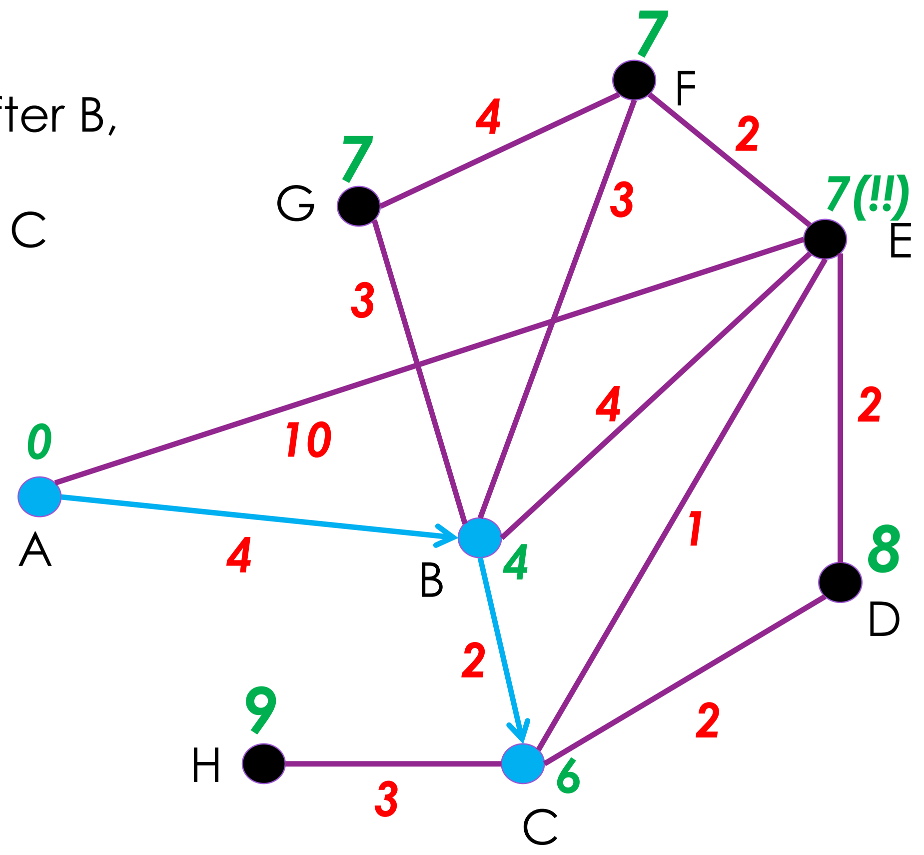
Walkthrough – tree from A

Now do neighbours of **B**
Add your own cost to neighbours



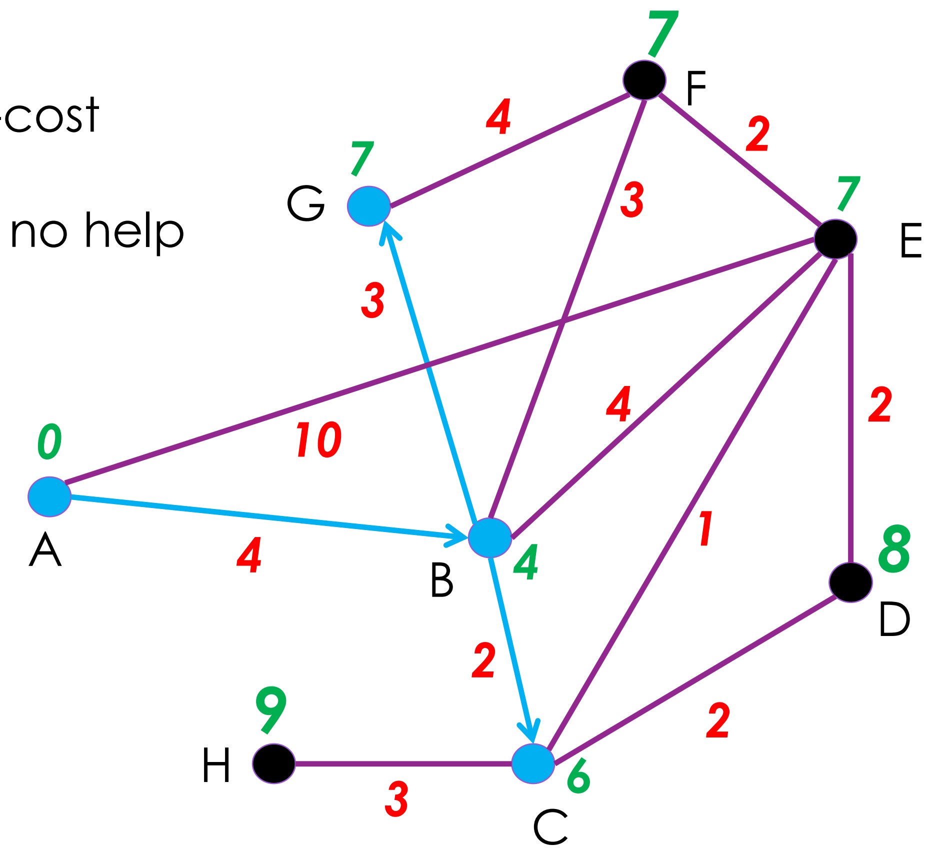
Walkthrough – tree from A

C is next lowest cost after B,
So confirm (B)C
Then do neighbours of C



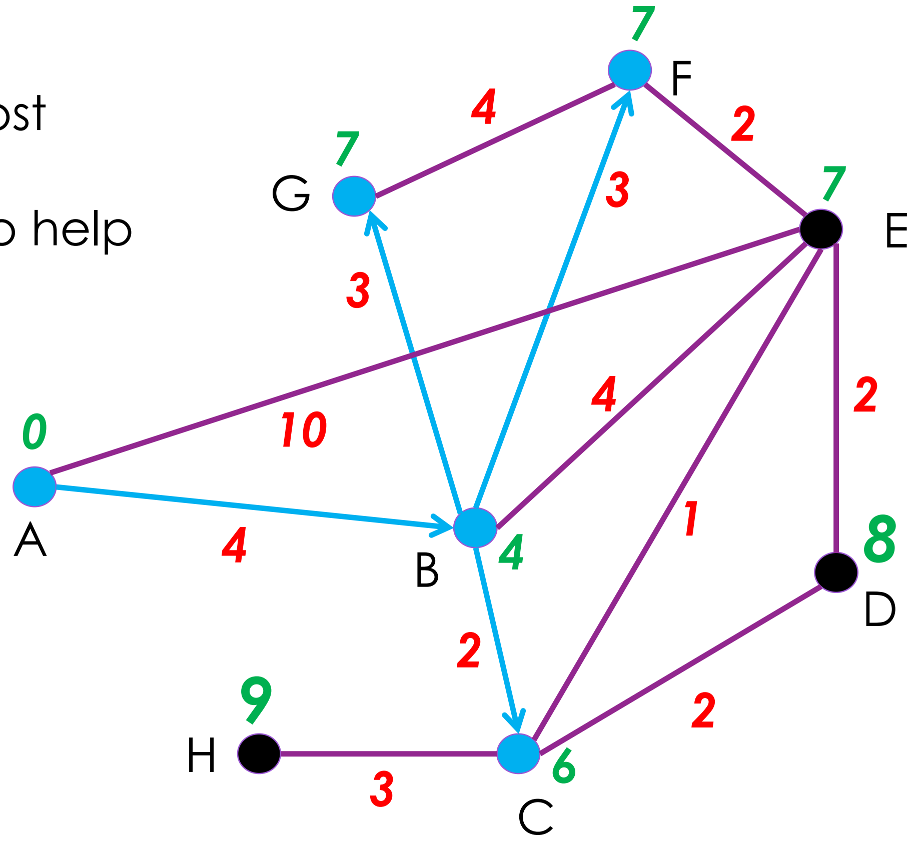
Walkthrough – tree from A

E,F,G are equal lowest-cost
Pick one: confirm (B)G
Do neighbour(s) of G – no help



Walkthrough – tree from A

E,F are equal lowest-cost
Pick one: confirm F
Do neighbours of F – no help

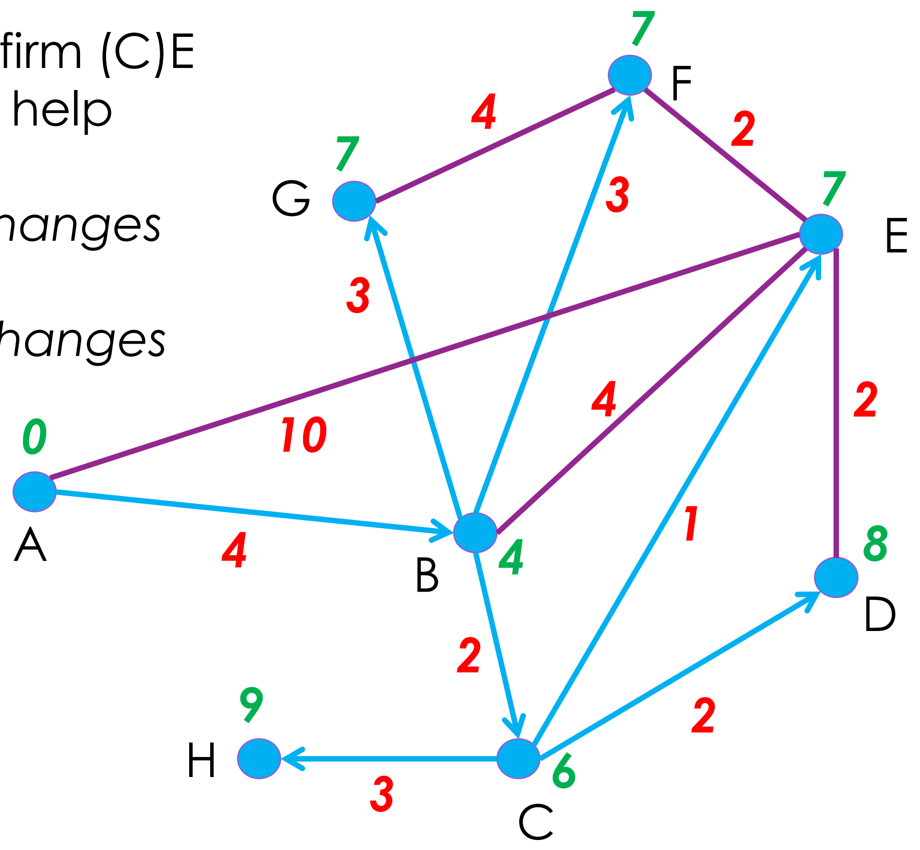


Walkthrough – tree from A

E is the lowest-cost, confirm (C)E
Do neighbours of E – no help

And repeat for D - no changes

And repeat for H – no changes



Dijkstra

- Works out from the source
  - And you need to repeat for every source
- Leverages optimality property
  - Use sub-paths to build longer shortest-paths
- Has some scaling issues in complex networks
  - Imagine 1000+ nodes, 1000’s links
  - Imagine changes at any single point
  - Lots of research…
- Needs complete topology
  - At each node/source

Bring on Distance Vector routing!

- When you **don't** know the topology…
  - Calculate Source tree in a distributed fashion
  - Looks a bit like Spanning Tree
- Nodes only know costs to neighbours
- Nodes only talk to neighbours
- Nodes all run the same algorithm
- Nodes/links may fail or lose messages

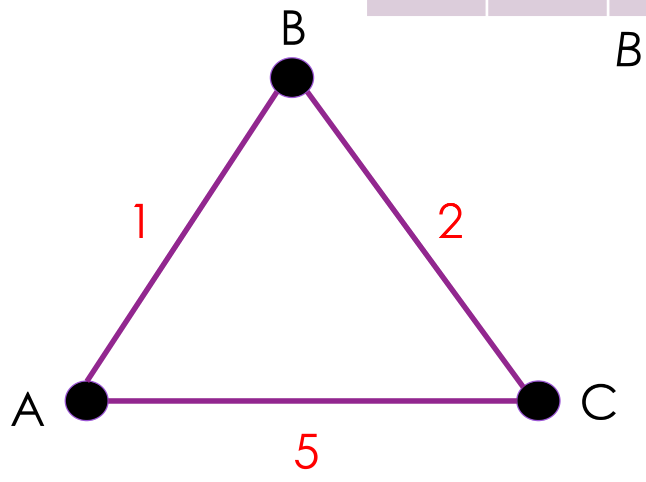
The algorithm

- Distributed Bellman-Ford
- One of two major approaches for routing protocols
  - Link-state is the other one
- Early routing approach (Routing Information Protocol (RIP), 1988)
  - Simple to use, but slow to converge, and somewhat fragile – still improving
- Each node stores a vector of distances, and next hops, to all destinations
  - Initially vector has 0 cost to self, infinity to all others
  - Send vector to neighbours
  - Update for each destination with lowest cost heard, adding cost of link
  - Repeat

Simplest (non-trivial) example

	A	B	C
A			
B	1	0	2
C			

B's table



	A	B	C
A	0	1	5
B			
C			

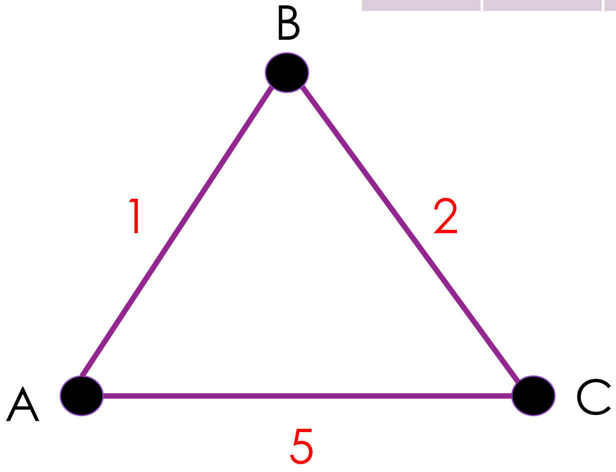
A's table

	A	B	C
A			
B			
C	5	2	0

C's table

B, **C** vectors shared with **A**

	A	B	C
A			
B	1	0	2
C			

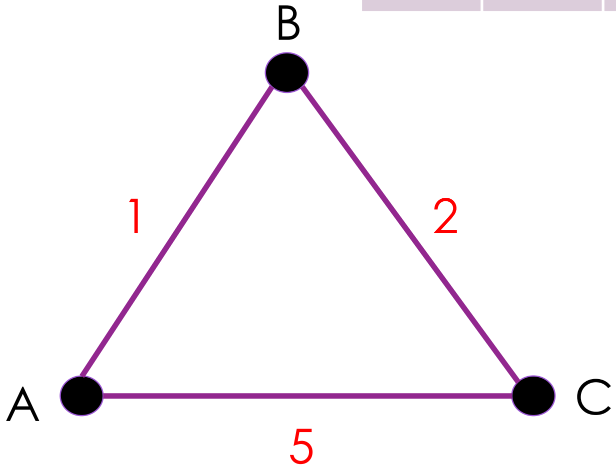


	A	B	C
A	0	1	5
B	1	0	2
C	5	2	0

	A	B	C
A			
B			
C	5	2	0

A updates its vectors…

	A	B	C
A			
B	1	0	2
C			

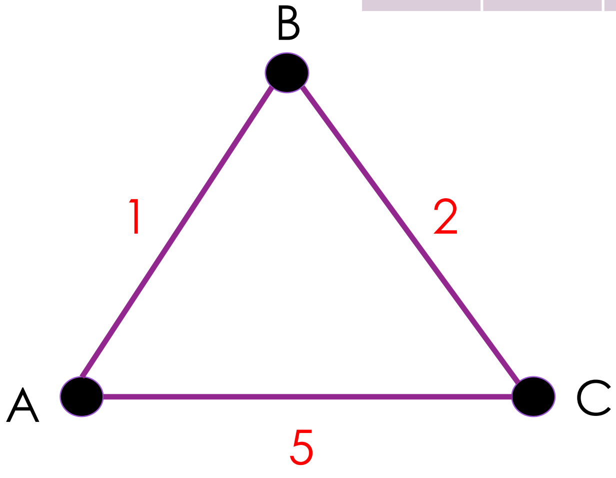


	A	B	C
A	0	1	**3(B)**
B	1	0	2
C	5	2	0

	A	B	C
A			
B			
C	5	2	0

Other routers do the same…

	A	B	C
A	0	1	5
B	1	0	2
C	3	2	0

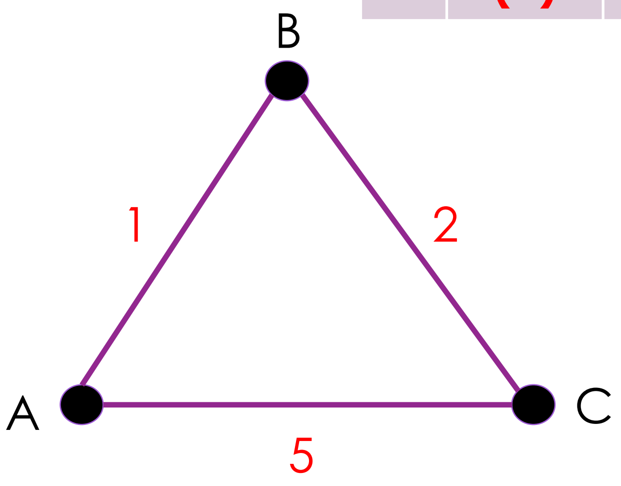


	A	B	C
A	0	1	**3(B)**
B	1	0	2
C	5	2	0

	A	B	C
A	0	1	5
B	1	0	2
C	**3(B)**	2	0

Everyone updates… and converges

	A	B	C
A	0	1	**3(B)**
B	1	0	2
C	**3(B)**	2	0



	A	B	C
A	0	1	**3(B)**
B	1	0	2
C	**3(B)**	2	0

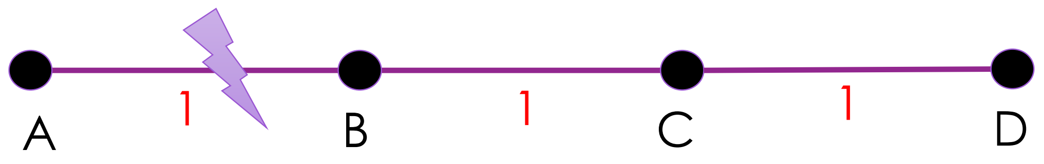
	A	B	C
A	0	1	**3(B)**
B	1	0	2
C	**3(B)**	2	0

Distance-vector routing

- Adding routes:
  - One hop wider awareness for every message exchange
- Deleting routes:
  - Deliberately or due to failures
  - Drop out of vectors, other nodes delete
- One small problem
  - Count to infinity…
  - When a particular piece of the network falls off.

Count to infinity

- Good news travels quickly, Bad news travels slowly



- Normally everyone else (C, D) sees a path to A **through B**
  - When **A** falls off, **B** sees a path to **A** <u>through **C**</u>, which sees a path through **B**…

- Deal with problem via "poison reverse", "split horizon"
  - Don't advertise route to the node you learnt it from
  - These don't scale well in certain circumstances

Link state routing

- The other, more common routing algorithm
- More computation, but better behaviours
  - Scales well to enterprise networks, though not globally
- Used in
  - *Open Shortest Path First* (OSPF),
  - *Intermediate System to Intermediate System* (IS-IS)
- Same baseline rules for a federated environment
  - Only talk to neighbours
  - Only know their costs
  - don’t know the topology (to start with)
  - Deal with node/link/message failures

Link state routing

- Simple Algorithm
- 3 parts:
  - **Flood** the network
  - **Learn** the topology
  - **Compute** tables with Dijkstra
  - And repeat every time there's a change.
- That's it!

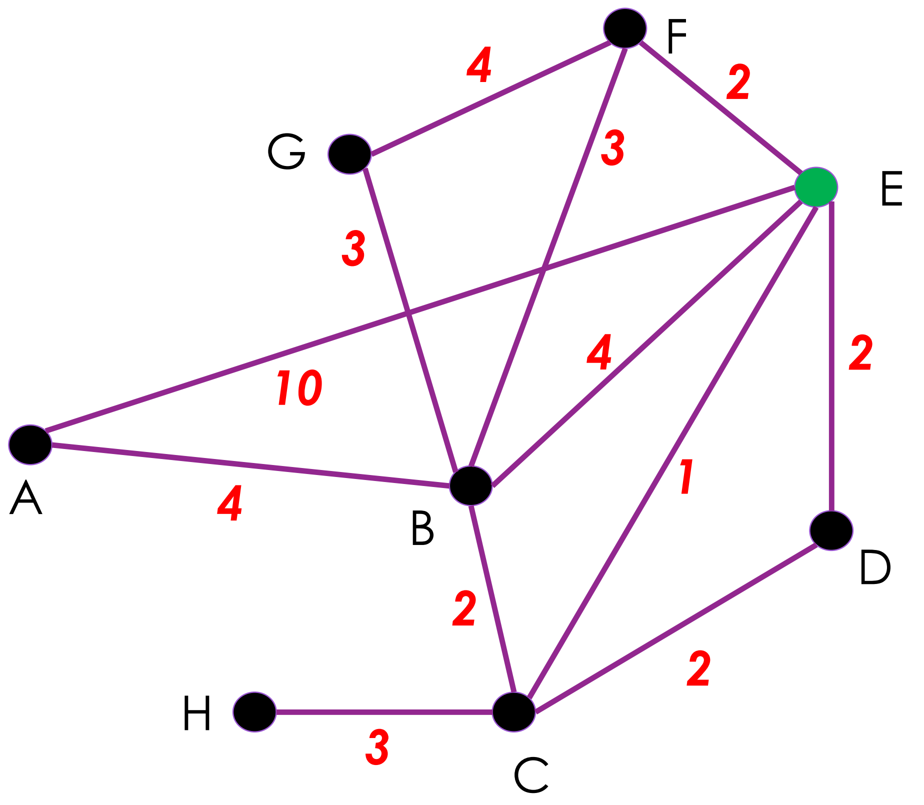
Flooding?

- Broadcast incoming (broadcast) message to all (other) outbound interfaces
  - Yes, it’s bad.
  - Unless it isn’t.
- Just keep track so you don’t repeat yourself, or cause storms
  - Use incrementing sequence numbers
  - May get multiple copies, deal with it
- Ironically, if you miss a message, use ARQ

Link State flooding

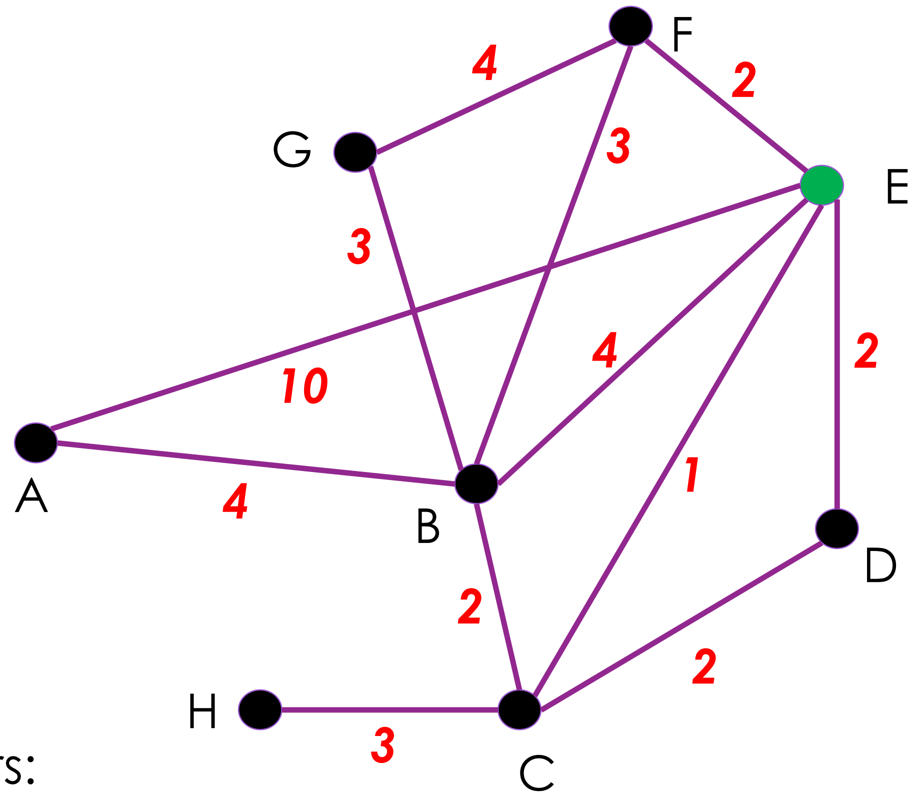
- Each node floods **link-state-packet (LSP)**
  - to the entire reachable network

Node E LSP	(+ some Seq #)
A	10
B	4
C	1
D	2
F	2



Link-state topology analysis

- Listen for LSPs, learn the topology
- Then run Dijkstra locally
  - On each node!
  - Wasteful
    - replicated communication/computation,
    - Lots of CPU grunt needed
  - But it's effective.
- On changes (node/link failures) the neighbours:
  - Detect a link down, or lack of heartbeat packets
  - flood updated LSP
  - and everybody recomputes
- Various (rare) failure modes (flooding fails, node flaps, seq# errors, races, …)
  - Manage by ageing LSPs and timeouts



Remember our routing expectations?

Expectation	Distance Vector	Link State
Correctness	Distributed Bellman-Ford	Replicated Dijkstra
Efficiency	Reasonable – shortest path	Reasonable – shortest path
Fairness	Reasonable – shortest path	Reasonable – shortest path
Convergence	**Slow… many exchanges**	**Fast – flood and compute**
Scalability	**Excellent – storage/compute**	**Ok – storage/compute**

Equal-cost multipath routing (ECMP)

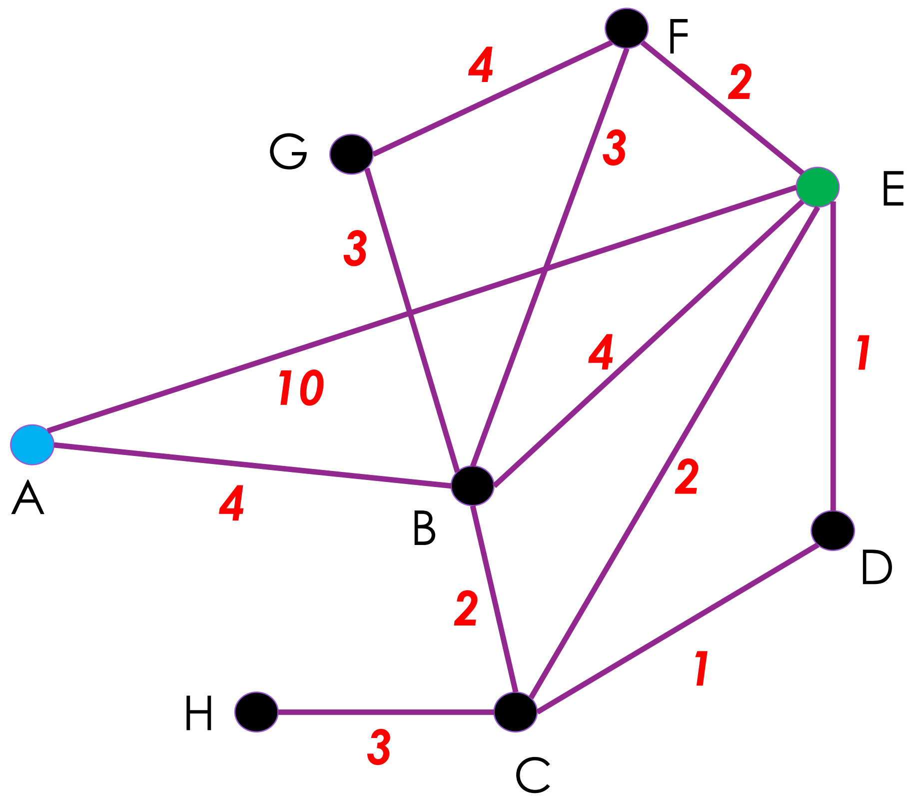
- Not a protocol/algorithm, but an extension for flexibility

- Allow for multiple paths for packets between source and destination
  - Greater redundancy
  - Improve performance
    - Capacity increase
    - Load balance

- Need to
  - Detect them, and
  - Forward traffic along them

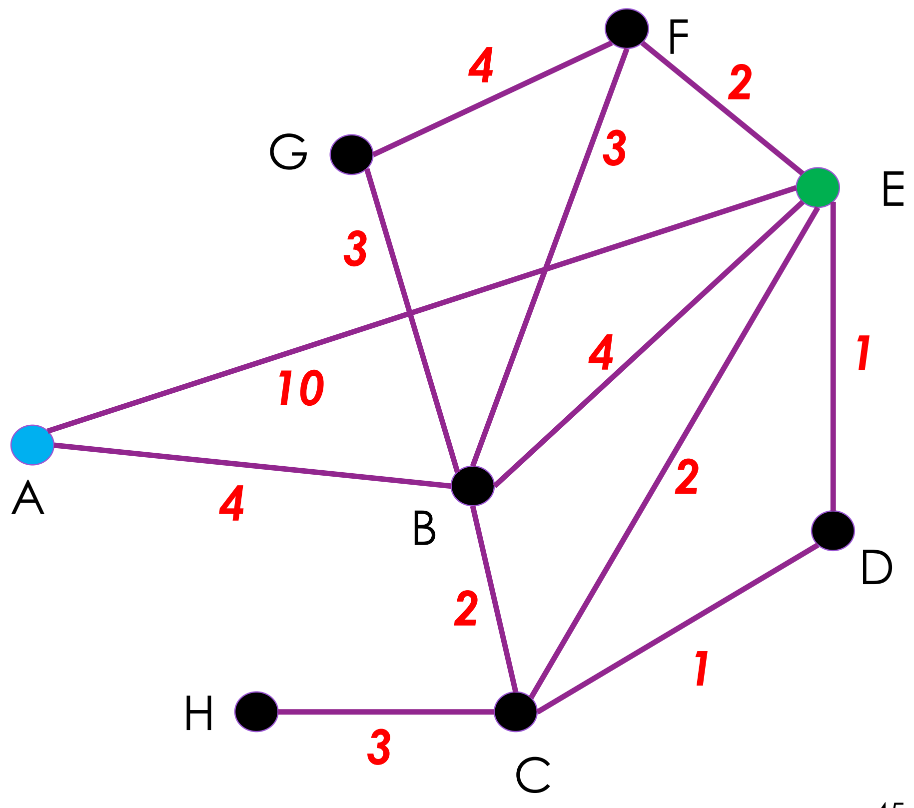
ECMP detection

- One approach: <u>don't tiebreak</u>
- Allow shortest path to be a **set**
  - Rather than a single choice
- A to E
  - With modified costs
  - ABE = 8
  - ABCE = 8
  - ABCDE = 8
- Not a tree now but a *directed acyclic graph*



ECMP forwarding

- Forwarding tables now have a set of interfaces for each destination
- Allocate each packet <u>randomly</u>?
  - Good for load balancing,
  - Bad for jitter (packet delay variation)
- Allocate by '<u>relationship</u>'
  - Use the destination **and** source IP#
    - *E.g.* ***E*** *chooses:* ***F-H*** *goes* ***EC****,* ***E-H*** *goes* ***EDC***
    - *Equal cost, and consistent performance*
- Allocate by '<u>flow</u>'
  - Using flow identifiers (IPv6)
- Less balanced, but more predictable



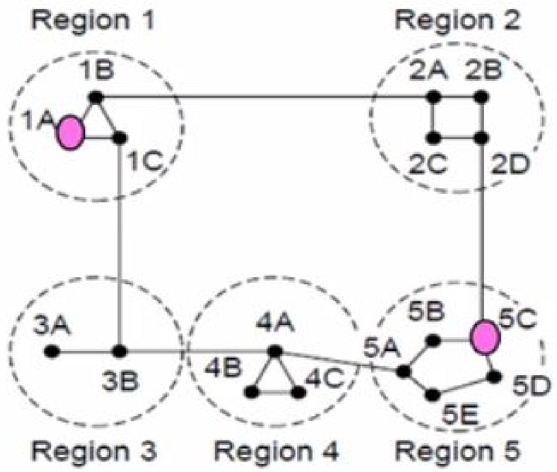
Hierarchical routing

- Scaling problems as identified earlier
  - Routing tables growing
  - Routing computing growing
  - Forwarding tables growing
- Network aggregation
  - Already have LAN prefixes aggregating a whole subnet
    - Don't need to advertise every single host on your LAN
  - Can treat a **group** of subnets as a larger subnet
    - E.g. adjacent /24s within a /16 (150.203.aaa.bbb)
    - But not all subnets now are 'adjacent'
  - What about geographical aggregation?

Routing to a region

- Aggregate nodes/subnets
  - Hide internal complexity
  - Shorter tables

- Downside:
  - Less optimal paths
  - **Full** 1A to 5C [1B] = 5 hops
  - **Hier.** 1A to 5C [1C] = **6** hops



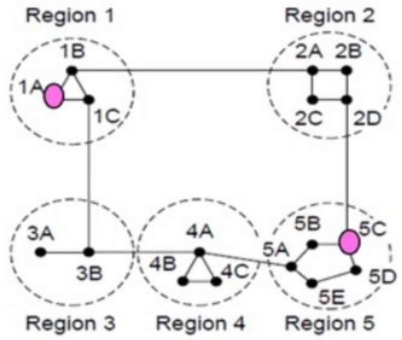
Full table for 1A

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

Routing to a region



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

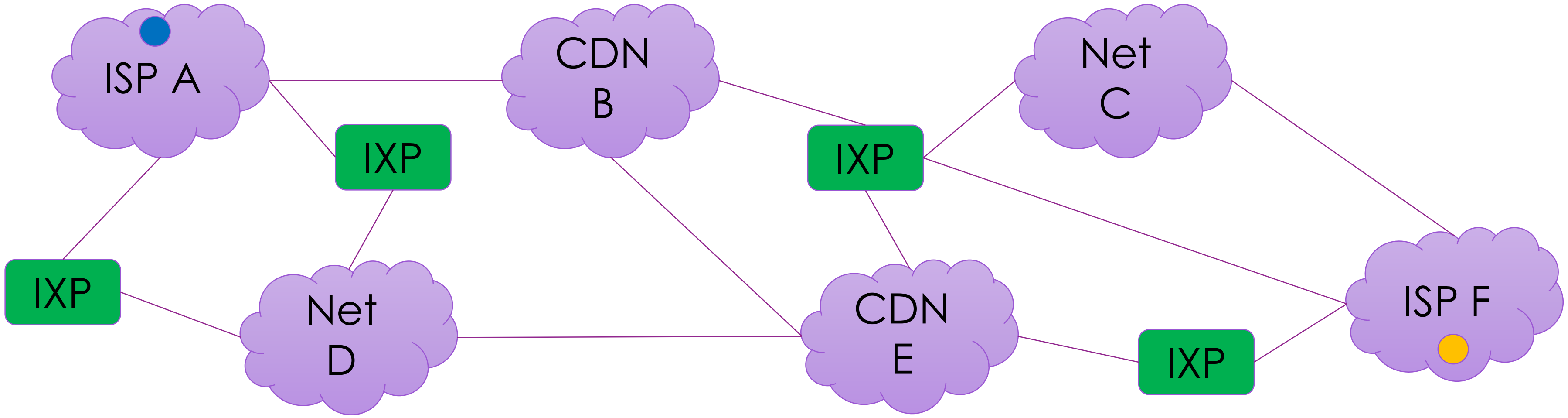
Hierarchical table for 1A

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

- Outside of a region, routers get told **one** route to get to that region
  - All hosts are aggregated into a smaller table,
  - Reducing communication and computation
- There can still be more than one route into or out of a region
  - It's still a local router decision how to get in/out for a particular region
  - A region sets a context, and designates some **border routers**
  - Within a region, we make our own (excellent) arrangements

Policy-based routing – and routing policies

- At the heart of the Internet
  - Multiple ISPs, interconnecting via *Internet Exchange Points* (IXP)
  - All running a business. Or a country.



Policy routing

- Already have dynamic, complex, large routing databases and algorithms

- Now Introduce <u>human needs</u>: **Layers 8+**
  - Money
  - Politics
  - Security
  - Religion

- i.e. we have POLICIES to add to our protocols
  - *E.g. National Research and Education Networks have an R&E traffic policy*
    - Wholesale purchase **AND** don't compete with commercial providers **AND** social good

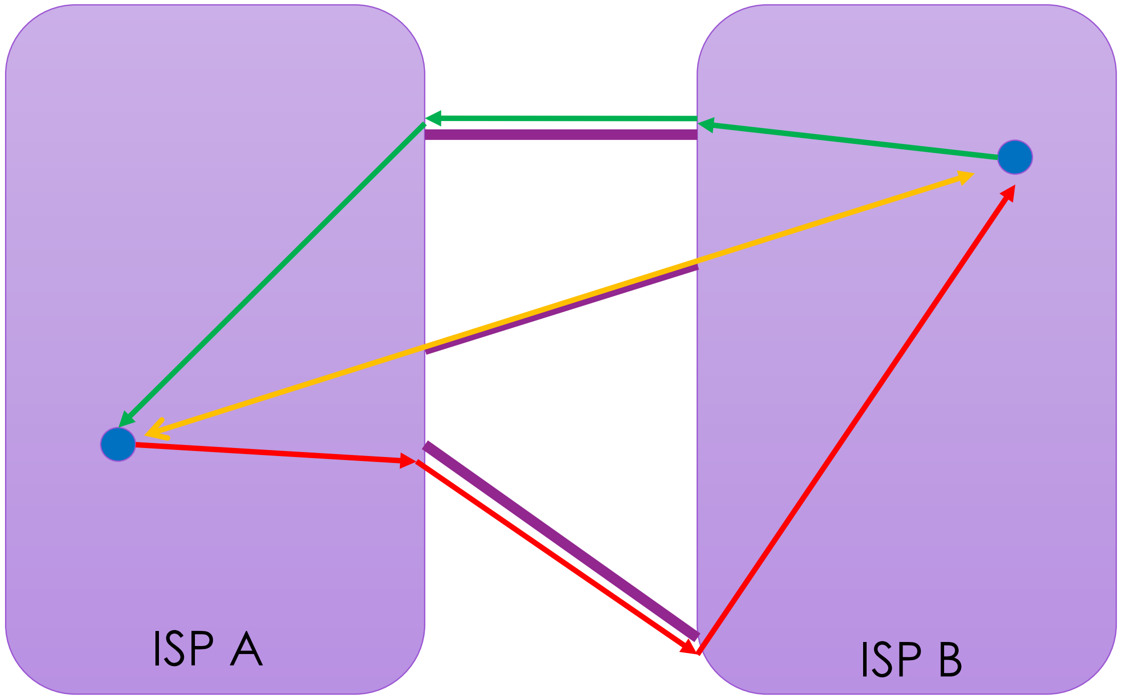
Shortest path is a local priority…

- E.g. each ISP policy: offload as quickly as possible

- Technical Term:
  - **Hot Potato Routing**

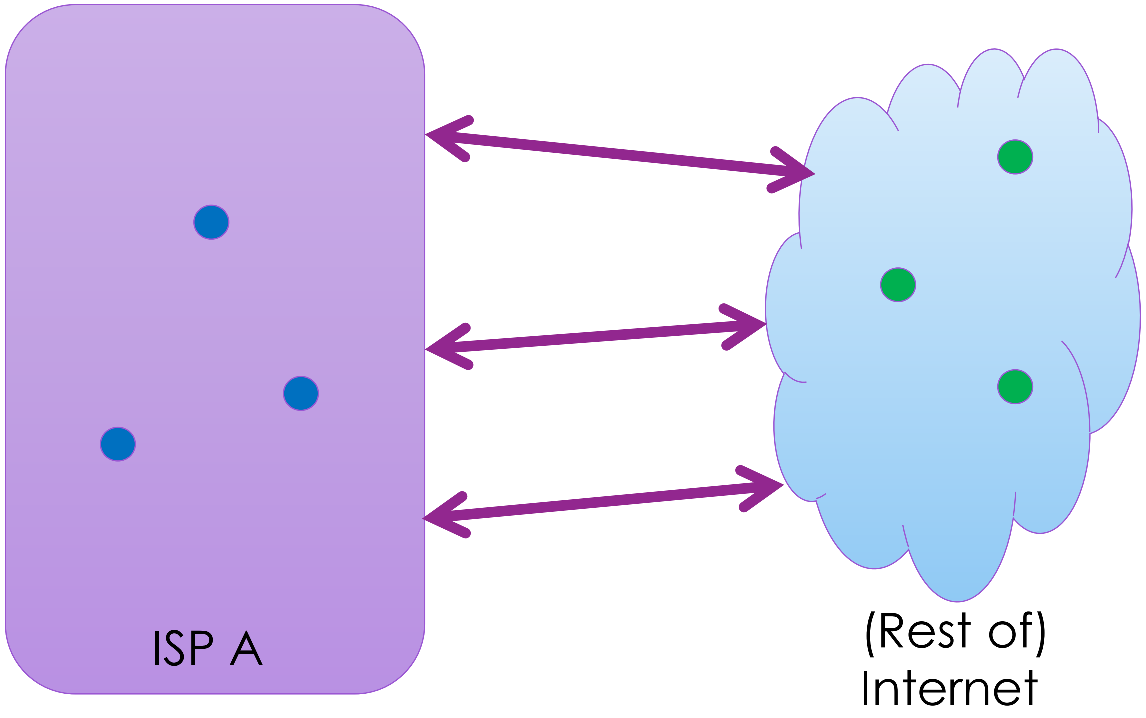
- Sub-optimal shortest path
- Asymmetric paths!

- Hierarchy is (consciously) broken, for good business reasons



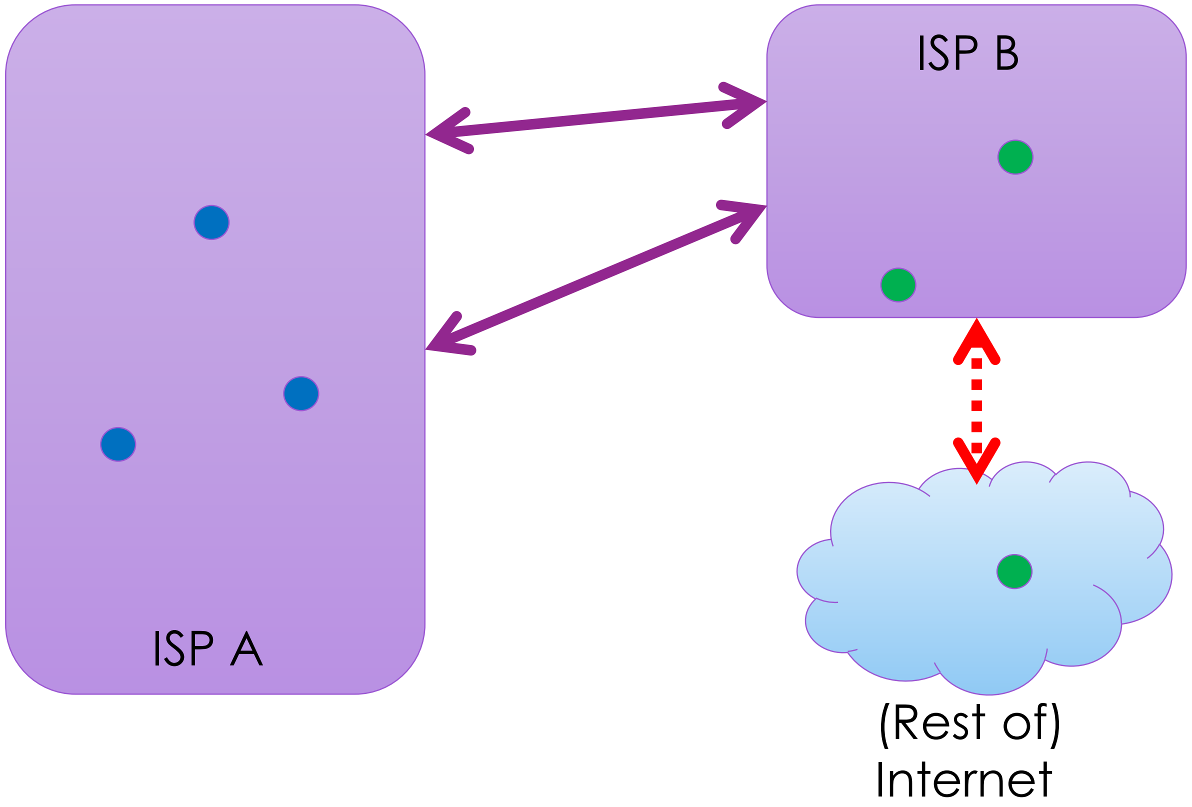
Most common policies: Transiting and Peering

- ISPs
  - Take your traffic and pass it through their network to the Internet
  - They take the Internet traffic and pass it through to you
  - And you pay them.



Common policies: Transiting and Peering

- ISPs
  - Take your traffic and pass it through to the other network
  - They take the other networks traffic and pass it through to you
  - You cannot reach the Internet through them.
  - Mutual benefit
    - No money exchanged
    - A CDN, or a cloud provider, or an NREN, or…

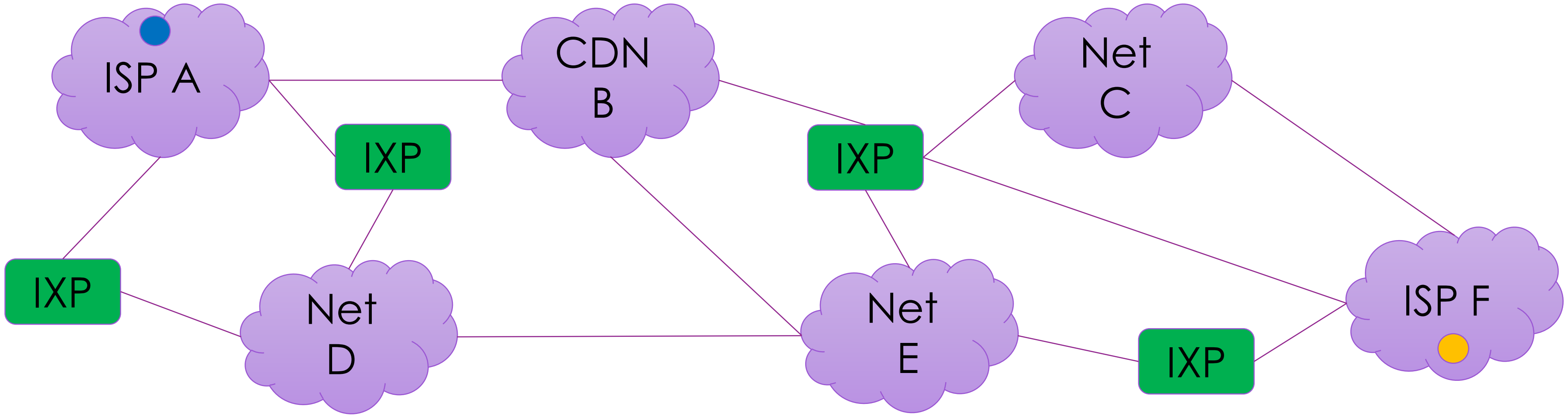


Border Gateway Protocol (BGP)

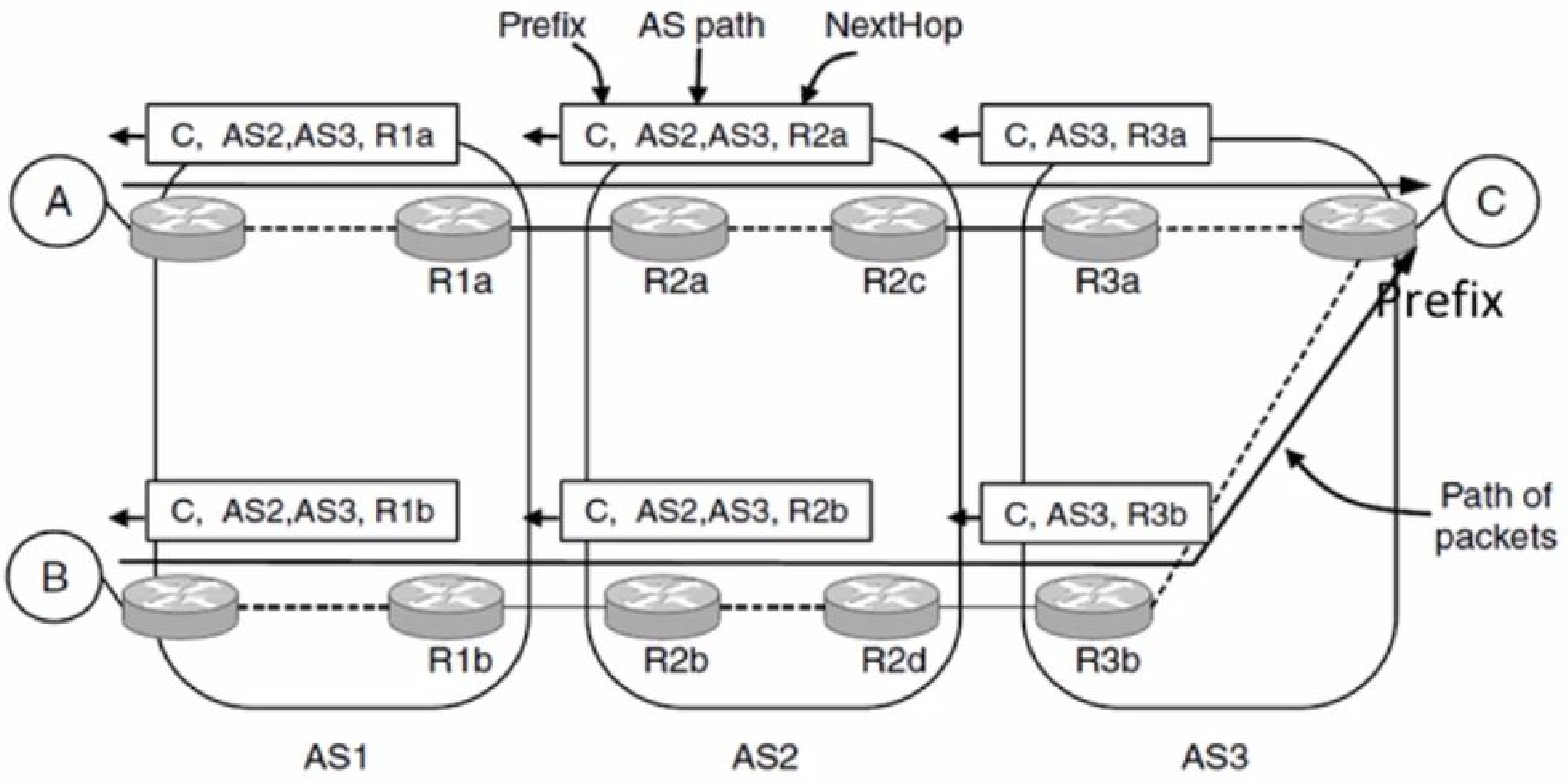
- The main Internet routing protocol today
- Key concepts:
- Separation of <u>interior</u> routing protocols and <u>exterior</u> routing protocols
  - Intradomain vs Interdomain
    - Enterprise vs International
- Identifies <u>Border Routers</u> (or Gateways) which run BGP
  - Creates an edge between interior and exterior routing
- Aggregates nodes within an '<u>Autonomous System</u>' (AS)
  - Think a region, a business, an ISP

BGP is more DV than LS

- Instead of Distance Vector it is a **Path Vector**
- Announcements:
  - IP Prefix, Next Hop
  - And the Path: list of AS's to transit
    - This allows loops to be detected and removed
    - *No distance indications*



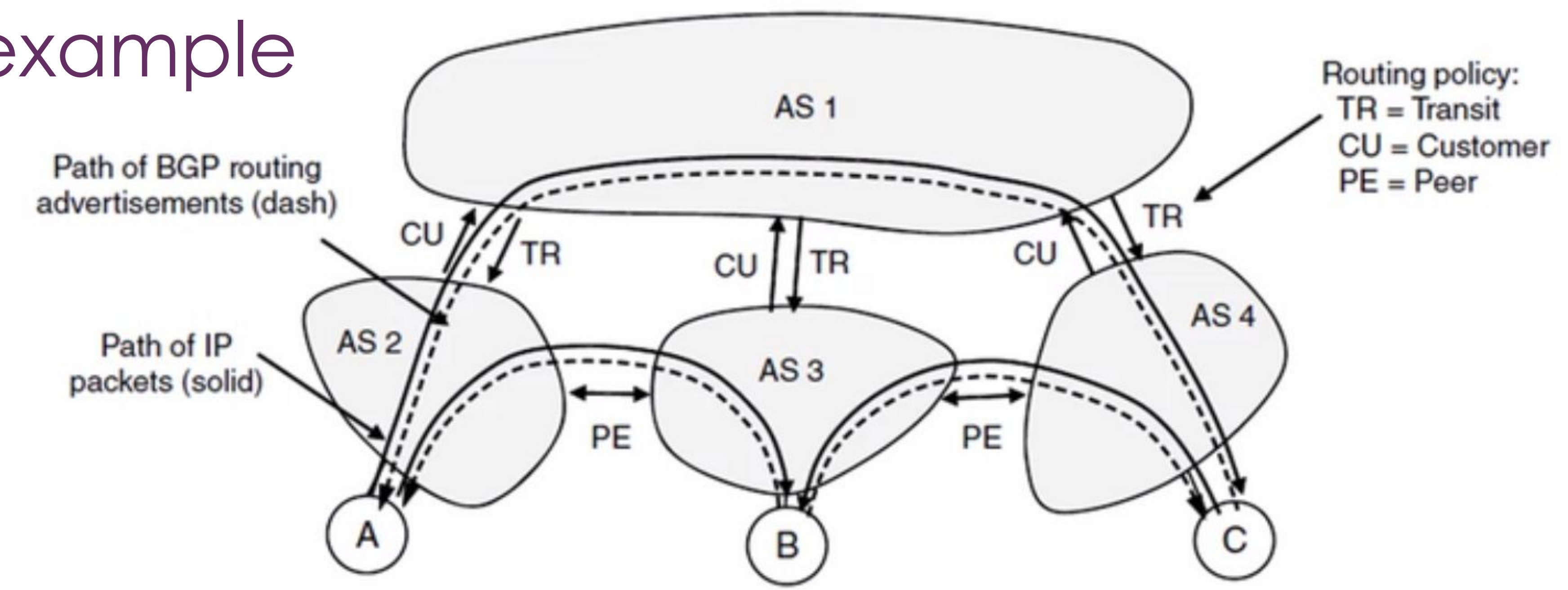
BGP route advertisments



Policy implementation

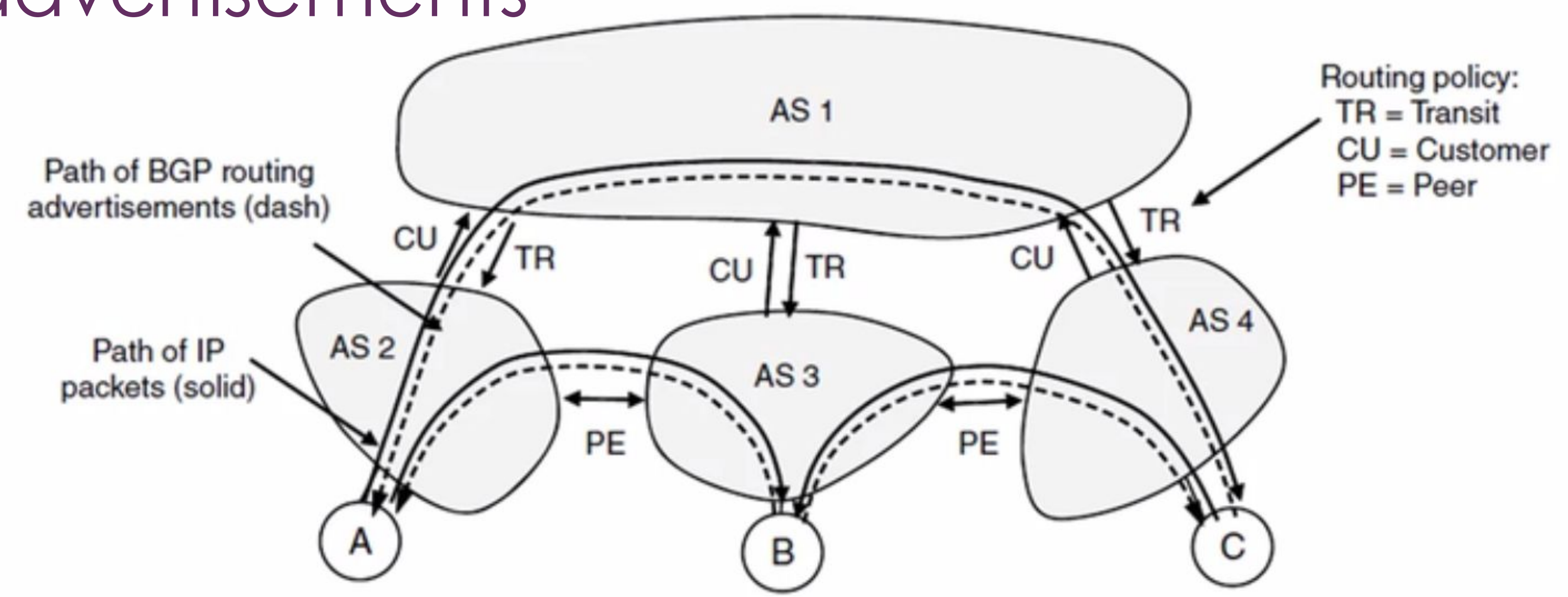
- BGP allows you to configure your route “advertisements”
- Border routers advertise available paths – with policy constraints
  - Only to those AS’s that may use them
  - And filter out those they cannot use
  - E.g. offer transit to some, peering to others
  - E.g. offer a faster path to some, slower to others ($$$)
- Border routers listen for available paths
  - And (given a choice) pick the one that suits them – for any reason!
    - Shortest, cheapest, friendliest, safest, politically/contractually-suitable, …
    - Human rather than ‘technical’ optimisation

BGP example



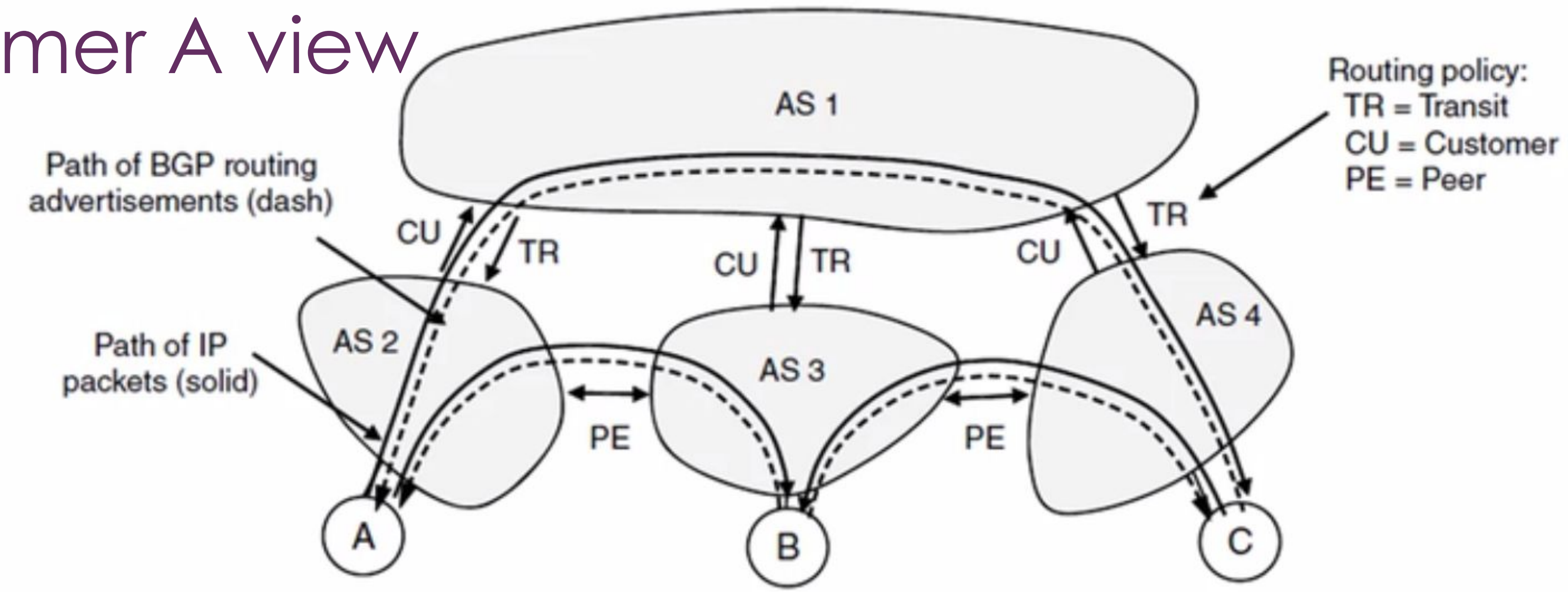
- Various businesses here:
  - AS1 is selling transit to AS2,3,4
  - AS2 and AS3 are peering (ditto AS3 and AS4)
  - AS2 is selling transit to customer A

BGP advertisements



- Various advertisements here:
  - Customer: **[A, (AS2), router2U]** is sent by AS2 to AS1
  - Transit: **[B,(AS1,AS3), router1L]**and **[C,(AS1,AS4), router1L]** is sent by AS1 to AS2
  - Peer: **[B,(AS3), router3L]** is sent by AS3 to AS2, **[A,(AS2), router2R]** is sent by AS2 to AS3

Customer A view



- So AS2 (and hence customer A)
  - Hears one option for reaching **C:** (AS1, AS4)
  - Hears two options for reaching **B**: *Transit*(AS1,AS3) and *Peer*(AS3)
    - And peering traffic is usually free…

In closing

- Routing is complicated and hard
  - this has been a very high-level view!
  - DV, LS and BGP are very important
- Internet is large and complex
- Policies are an important factor
  - the internet is also a business
- **Connecting** interior and exterior routing/gateway protocols
  - Literally an edge case. Haven't even discussed it
- Performance is challenging
  - Scalability, convergence, reliability, trustworthiness, optimisation, …
  - All in a (globally) distributed system
- Anybody looking for a PhD topic?