changelog

8 Oct 2024 (after lecture): destination unreachable: fix too-big font size for ping

8 Oct 2024 (after lecture): ‘flooding’: correct extra gateway for 2::3/4::1 router’s table entry for 4::/ route

8 Oct 2024 (after lecture): ‘flooding’: correct first message sent to specify 2001:db8:4::/40

8 Oct 2024: spanning tree example: fix missing hilite of G-E edge

22 Oct 2024 (after lecture): Pakistan hijack: correct addresses Youtube advertized so they match Pakistan Telecom’s

routing tables

IP addresses	gateway	iface
`2001:0db8:40:f000::/44`	`---`	`int1`
`2001:0db8:40:e000::/44`	`2001:0db8:40:f000::2`	`int1`
`2001:0db8:40:d000::/44`	`---`	`int3`
`3fff:1000:19::/48`	`---`	`ext1`
…	…	…
default	`fe80::17`	`ext2`

IP addresses	gateway	iface
`192.0.2.0/25`	`---`	`int1`
`192.0.2.128/26`	`192.0.2.1`	`int1`
`192.0.2.192/26`	`192.0.2.2`	`int1`
`198.51.100.0/25`	`192.0.2.1`	`int1`
`198.51.100.128/25`	`---`	`int2`
…	…	…
default	`203.0.113.1`	`ext`

filling routing tables

easy part: what networks are you directly connected to

- that range of IP addresses, that interface

harder part: other routers on connected router

need to learn:

- addresses of other router
- which networks can be reached through them directly or indirectly

need to choose between multiple ways of reaching networks

problems when forwarding

no entry in routing table

no entry in neighbor table

(after attempting ARP, or neighbor discovery)

packet too big for next network

there's an infinite loop in the route

problems when forwarding

no entry in routing table

no entry in neighbor table

(after attempting ARP, or neighbor discovery)

packet too big for next network

there's an infinite loop in the route

destination host unreachable

```
$ ping 128.143.67.254
PING 128.143.67.254 (128.143.67.254) 56(84) bytes of data.
From 128.143.63.1 icmp_seq=1 Destination Host Unreachable
From 128.143.63.1 icmp_seq=6 Destination Host Unreachable
^C
--- 128.143.67.254 ping statistics ---
10 packets transmitted, 0 received, +2 errors, 100% packet los
pipe 4
```

—

```
$ ping6 2606:8e80:7007:ef1a::1
PING 2606:8e80:7007:ef1a::1(2606:8e80:7007:ef1a::1) 56 data by
From 2606:8e80:7007:ef1a:cf1f:3948:b5c1:a522 icmp_seq=1
        Destination unreachable: Address unreachable
....
```

ICMPv6 destination unreachable messages

IPv6 header with ICMP as next protocol

1 byte type = 1 (destination unreachable)

1 byte code =

    examples: address unreachable, administritatively prohibited

most of contents of message causing problem

    only most to avoid exceeding max packet size
    should let OS figure out which socket to send error to

generating destination unreachable

by routers: reached correct network, machine not there

by routers: no route to network at all

by routers: administrator rule prohibits forwarding

by destination host: no program listening to that ‘port’

…

different code values for all cases

machine can also choose to send nothing back

ICMPv4 destination unrachable

basically same format as ICMPv6, but...

different type/code integer values

only IPv4 header + 64 bytes of original packet included

problems when forwarding

no entry in routing table

no entry in neighbor table

(after attempting ARP, or neighbor discovery)

packet too big for next network

there's an infinite loop in the route

fragmentation

max frame data size on my local network = 1500 bytes, but...

```
$ ping6 fe80::da07:b6ff:fed9:ae50 -s 4000
PING fe80::da07:b6ff:fed9:ae50 (fe80::da07:b6ff:fed9:ae50) 40
4008 bytes from fe80::da07:b6ff:fed9:ae50%eno1: icmp_seq=1 tt
4008 bytes from fe80::da07:b6ff:fed9:ae50%eno1: icmp_seq=2 tt
4008 bytes from fe80::da07:b6ff:fed9:ae50%eno1: icmp_seq=3 tt
...
$ ping -s 4000 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 4000(4028) bytes of data.
4008 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.891 ms
4008 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.806 ms
4008 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.748 ms
```

fragmentation

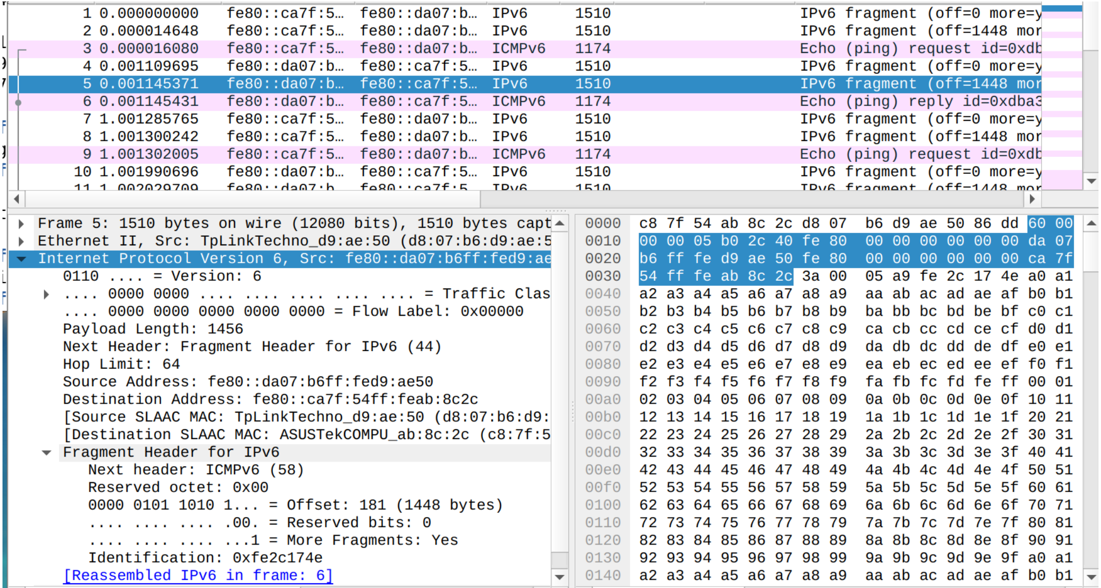
original sender or router splits packet into multiple

each part called a *fragment*

stored temporarily and “reassembled” at receiver

- Linux defaults:
- max 64 packet gap between fragments per source IP
- 30 second time limit before discaded
- 3-4MB buffer of packets

IPv6 fragments



IPv4 fragments

No.	Time	Source	Destination	Protocol	Length	Seq#	Ack#	Info
1	0.000000000	192.168.1.232	192.168.1.1	IPv4	1514			Fragmented IP protocol (pro
2	0.000011181	192.168.1.232	192.168.1.1	IPv4	1514			Fragmented IP protocol (pro
3	0.000012915	192.168.1.232	192.168.1.1	ICMP	1082			Echo (ping) request id=0xc
4	0.000776503	192.168.1.1	192.168.1.232	IPv4	1514			Fragmented IP protocol (pro
5	0.000812871	192.168.1.1	192.168.1.232	IPv4	1514			Fragmented IP protocol (pro
6	0.000812931	192.168.1.1	192.168.1.232	ICMP	1082			Echo (ping) reply id=0xc
7	1.023997142	192.168.1.232	192.168.1.1	IPv4	1514			Fragmented IP protocol (pro
8	1.024012951	192.168.1.232	192.168.1.1	IPv4	1514			Fragmented IP protocol (pro
9	1.024014785	192.168.1.232	192.168.1.1	ICMP	1082			Echo (ping) request id=0xc
10	1.024692402	192.168.1.1	192.168.1.232	IPv4	1514			Fragmented IP protocol (pro
11	1.024730263	192.168.1.1	192.168.1.232	IPv4	1514			Fragmented IP protocol (pro

```
Frame 5: 1514 bytes on wire (12112 bits), 1514 bytes captur
Ethernet II, Src: TpLinkTechno_d9:ae:50 (d8:07:b6:d9:ae:50)
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not
    Total Length: 1500
    Identification: 0xaa6b (43627)
    001. .... = Flags: 0x1, More fragments
    ...0 0000 1011 1001 = Fragment Offset: 1480
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x25c3 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.1
    Destination Address: 192.168.1.232
    [Reassembled IPv4 in frame: 6]
Data (1480 bytes)
[Community ID: 1:Ac2BWwekehTAnPxIuiD9sNt+ItE=]
```

```
0000  c8 7f 54 ab 8c 2c d8 07  b6 d9 ae 50 08 00 45 00
0010  05 dc aa 6b 20 b9 40 01  25 c3 c0 a8 01 01 c0 a8
0020  01 e8 c0 c1 c2 c3 c4 c5  c6 c7 c8 c9 ca cb cc cd
0030  ce cf d0 d1 d2 d3 d4 d5  d6 d7 d8 d9 da db dc dd
0040  de df e0 e1 e2 e3 e4 e5  e6 e7 e8 e9 ea eb ec ed
0050  ee ef f0 f1 f2 f3 f4 f5  f6 f7 f8 f9 fa fb fc fd
0060  fe ff 00 01 02 03 04 05  06 07 08 09 0a 0b 0c 0d
0070  0e 0f 10 11 12 13 14 15  16 17 18 19 1a 1b 1c 1d
0080  1e 1f 20 21 22 23 24 25  26 27 28 29 2a 2b 2c 2d
0090  2e 2f 30 31 32 33 34 35  36 37 38 39 3a 3b 3c 3d
00a0  3e 3f 40 41 42 43 44 45  46 47 48 49 4a 4b 4c 4d
00b0  4e 4f 50 51 52 53 54 55  56 57 58 59 5a 5b 5c 5d
00c0  5e 5f 60 61 62 63 64 65  66 67 68 69 6a 6b 6c 6d
00d0  6e 6f 70 71 72 73 74 75  76 77 78 79 7a 7b 7c 7d
00e0  7e 7f 80 81 82 83 84 85  86 87 88 89 8a 8b 8c 8d
00f0  8e 8f 90 91 92 93 94 95  96 97 98 99 9a 9b 9c 9d
0100  9e 9f a0 a1 a2 a3 a4 a5  a6 a7 a8 a9 aa ab ac ad
0110  ae af b0 b1 b2 b3 b4 b5  b6 b7 b8 b9 ba bb bc bd
0120  be bf c0 c1 c2 c3 c4 c5  c6 c7 c8 c9 ca cb cc cd
```

varying frame size support

also called *maximum transmission unit* (MTU)

typical Ethernet, Wifi — 1500 bytes

Ethernet with "jumbo frames" – 65535 bytes

IPsec ESP VPN over 1500-byte MTU network – $\sim$1400–1440 bytes

VPN — simulated network link over other network links

routers making fragments

option in IPv4 to handle frame size mismatch, but not great:

extra data sent over network (especially if just over max size)

- extra copies of main headers on each fragment

extra work at receiver to reconstruct fragments

lose whole packet if one fragment is lost

- but other routers likely to still waste time forwarding all other fragments

avoiding fragmentation

IPv4 — DF (don't fragment) flag in packets

- if set, routers not allowed to fragment packet

IPv6 — routers never fragment packets

- any fragments made at source machine only

avoiding fragmentation

IPv4 — DF (don't fragment) flag in packets

- if set, routers not allowed to fragment packet

IPv6 — routers never fragment packets

- any fragments made at source machine only

when set — ICMP error

- ICMPv6: Packet Too Big
- ICMPv4: destination unreachable + reason code of fragmentation needed
- (hopefully, bad networks might drop packet instead)

ICMPv6 error tells you maximum supported size

- (by first link that got packet rejected — might be more constraining link later)

exercise: fragmentation perf

assume:

- Ethernet header/trailer: 26 bytes
- IPv4 header: 20 bytes + 0 bytes of options
- TCP header: 20 bytes + 16 bytes of options

suppose local network supports 65535-byte ethernet payloads

and remote network suports 1500-byte ethernet payloads

and fragmentation happens

exericses:

- lowest overhead TCP segment size?
- overhead for 64000-byte TCP segments?
- highest overhead TCP segment size?

problems when forwarding

no entry in routing table

no entry in neighbor table

(after attempting ARP, or neighbor discovery)

packet too big for next network

there's an infinite loop in the route

time-to-live (v4) / hop limit (v6)

stored in IP header

when forwarding packet, router will:

subtract one from TTL / hop limit

and recompute checksum accordingly

if TTL/hop limit $= 0$, drop packet

usually send back ICMP "Time Exceeded" error

traceroute

ICMP Time Exceeded messages come from router

$\rightarrow$ tells you which routers are involved

traceroute

ICMP Time Exceeded messages come from router

$\rightarrow$ tells you which routers are involved

`traceroute` command: deliberately packets with low TTL/hop limit

print out what time exceeded messages we get back

typically sent with TTL/hop limit $= 255$ so it doesn't get lost

('backwards' path might be longer than forwards one)

traceroute example

```
traceroute to ripe.net (193.0.11.51), 30 hops max, 60 byte packets
 1  128.143.63.1 (128.143.63.1)  6.367 ms  8.562 ms  8.577 ms
 2  cr01-gil-ae15-00.net.virginia.edu (128.143.221.17)  0.370 ms  0.334 ms  0.349 ms
 3  * * *
 4  br01-udc-et-1-2-0.net.virginia.edu (128.143.236.5)  0.502 ms  0.468 ms  0.488 ms
 5  i2-vt.net.virginia.edu (192.35.48.34)  3.374 ms  3.448 ms  3.413 ms
 6  192.122.175.15 (192.122.175.15)  5.715 ms  5.628 ms  5.590 ms
 7  fourhundredge-0-0-0-17.4079.core1.ashb.net.internet2.edu (163.253.1.8)  29.163 ms
    fourhundredge-0-0-0-16.4079.core1.ashb.net.internet2.edu (163.253.1.2)  28.880 ms
    fourhundredge-0-0-0-17.4079.core1.ashb.net.internet2.edu (163.253.1.8)  28.876 ms
 8  fourhundredge-0-0-0-1.4079.core1.clev.net.internet2.edu (163.253.1.123)  29.568 ms
    28.667 ms  28.666 ms
 9  fourhundredge-0-0-0-0.4079.core2.newy32aoa.net.internet2.edu (163.253.1.239)  29.608 ms
    29.476 ms  29.400 ms
10  fourhundredge-0-0-0-19.4079.core1.newy32aoa.net.internet2.edu (163.253.1.40)  28.958 ms
    28.999 ms
    fourhundredge-0-0-0-21.4079.core1.newy32aoa.net.internet2.edu (163.253.1.44)  29.280 ms
11  e1-3-2-502.asd001b-jnx-06.surf.net (145.145.166.18)  115.822 ms  115.823 ms  115.744 ms
12  lo0-2.asd001b-jnx-01-surfinternet.surf.net (145.145.128.4)  115.988 ms  115.932 ms  115.
13  gw.amsix.telrtr.ripe.net (80.249.208.71)  121.956 ms  121.968 ms  121.844 ms
14  * * *
15  * * *
```

traceroute sent

No.	Time	Source	Destination	TTL	Protocol	Length	Info
1	0.000000000	128.143.71.27	193.0.11.51	1	UDP	74	54510 → 33434 Len=32
2	0.000042000	128.143.71.27	193.0.11.51	1	UDP	74	56464 → 33435 Len=32
3	0.000078065	128.143.71.27	193.0.11.51	1	UDP	74	36104 → 33436 Len=32
4	0.000115226	128.143.71.27	193.0.11.51	2	UDP	74	34004 → 33437 Len=32
5	0.000151405	128.143.71.27	193.0.11.51	2	UDP	74	57973 → 33438 Len=32
6	0.000186502	128.143.71.27	193.0.11.51	2	UDP	74	50866 → 33439 Len=32
7	0.000222625	128.143.71.27	193.0.11.51	3	UDP	74	48263 → 33440 Len=32
8	0.000257734	128.143.71.27	193.0.11.51	3	UDP	74	60098 → 33441 Len=32
9	0.000292642	128.143.71.27	193.0.11.51	3	UDP	74	58655 → 33442 Len=32
10	0.000327537	128.143.71.27	193.0.11.51	4	UDP	74	40741 → 33443 Len=32
11	0.000362342	128.143.71.27	193.0.11.51	4	UDP	74	48193 → 33444 Len=32
12	0.000397460	128.143.71.27	193.0.11.51	4	UDP	74	60985 → 33445 Len=32
13	0.000433117	128.143.71.27	193.0.11.51	5	UDP	74	38126 → 33446 Len=32
14	0.000468223	128.143.71.27	193.0.11.51	5	UDP	74	38788 → 33447 Len=32

traceroute received

30	0.003908850	192.35.48.34	128.143.71.27	251,1	ICMP	110	Time-to-live exceeded
31	0.003908954	192.35.48.34	128.143.71.27	251,1	ICMP	110	Time-to-live exceeded
35	0.006246570	192.122.175.15	128.143.71.27	250,1	ICMP	110	Time-to-live exceeded
36	0.006246708	192.122.175.15	128.143.71.27	250,1	ICMP	110	Time-to-live exceeded
37	0.006246784	192.122.175.15	128.143.71.27	250,1	ICMP	110	Time-to-live exceeded
38	0.006346847	128.143.63.1	128.143.71.27	64,1	ICMP	70	Time-to-live exceeded
43	0.008594648	128.143.63.1	128.143.71.27	64,1	ICMP	70	Time-to-live exceeded
44	0.008647695	128.143.63.1	128.143.71.27	64,1	ICMP	70	Time-to-live exceeded
50	0.029854913	163.253.1.8	128.143.71.27	243,1	ICMP	186	Time-to-live exceeded
51	0.029855048	163.253.1.2	128.143.71.27	243,1	ICMP	186	Time-to-live exceeded
52	0.029904130	163.253.1.8	128.143.71.27	243,1	ICMP	186	Time-to-live exceeded
53	0.030634797	163.253.1.123	128.143.71.27	244,2	ICMP	186	Time-to-live exceeded
54	0.031475863	128.143.236.90	128.143.71.27	253,1	ICMP	70	Time-to-live exceeded
55	0.031476027	128.143.236.90	128.143.71.27	253,1	ICMP	70	Time-to-live exceeded

aside: multiple paths

- only showing *forward* path
  - routing in reverse direction is often different
- sometimes multiple forward paths
  - way we've shown routing table so far does not allow this

constructing routing/neighbor tables

interesting task: how to fill tables

two general strategies:

routers/switches learn from neighbors

"distributed"

information gathered on single controller machine
which configures routers/switches

"centralized"

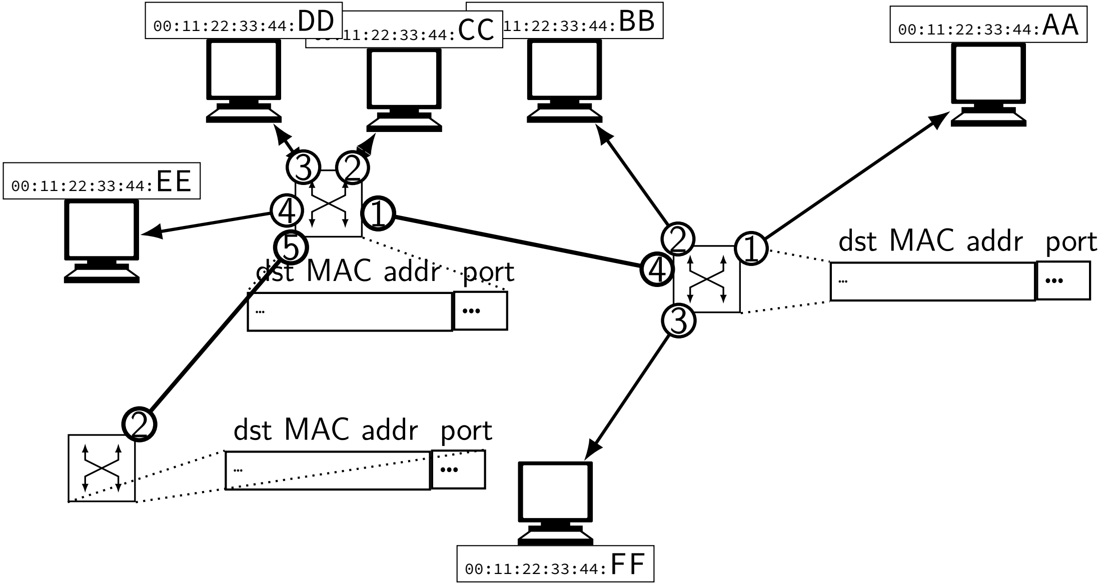
basic flooding

idea: broadcast message to whole network

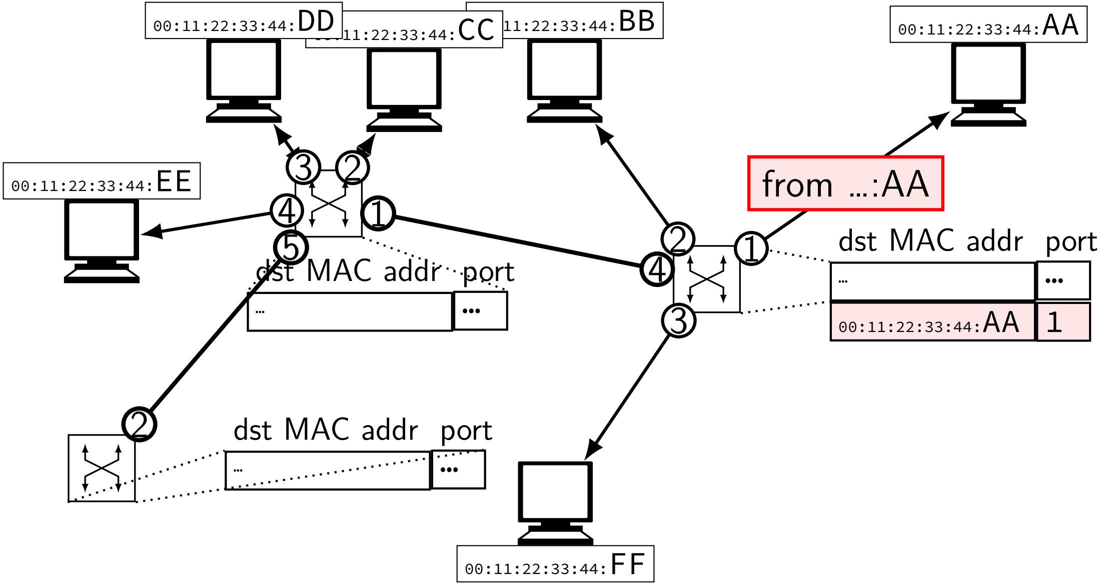
where message comes from = way to send back

used this idea in MAC learning

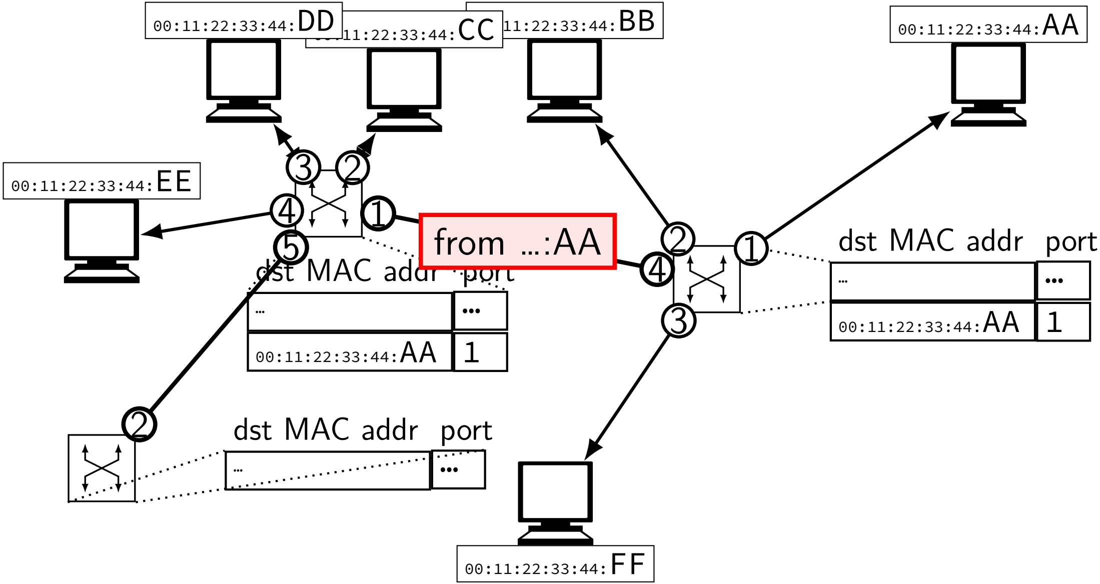
flooding one entry



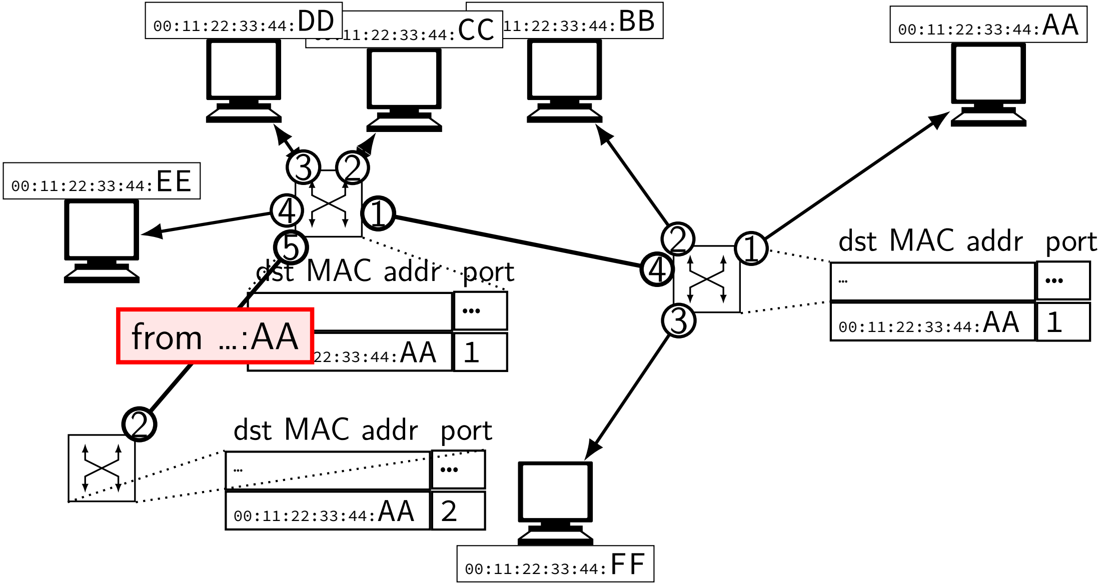
flooding one entry



flooding one entry

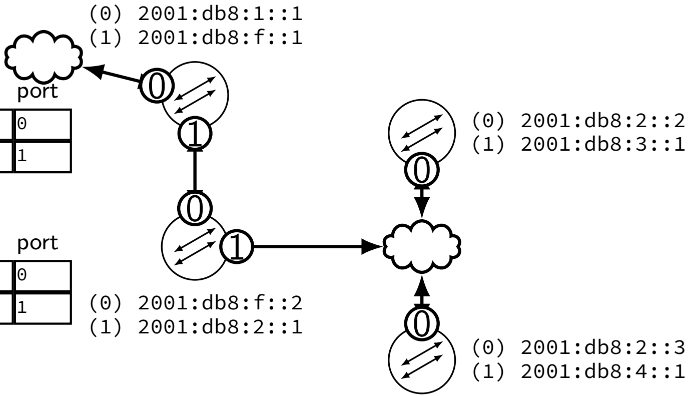


flooding one entry



‘flooding’

addresses	gateway	port
2001:db8:2::/40	---	0
2001:db8:3::/40	---	1



addresses	gateway	port
2001:db8:1::/40	---	0
2001:db8:f::/40	---	1

addresses	gateway	port
2001:db8:f::/40	---	0
2001:db8:2::/40	---	1

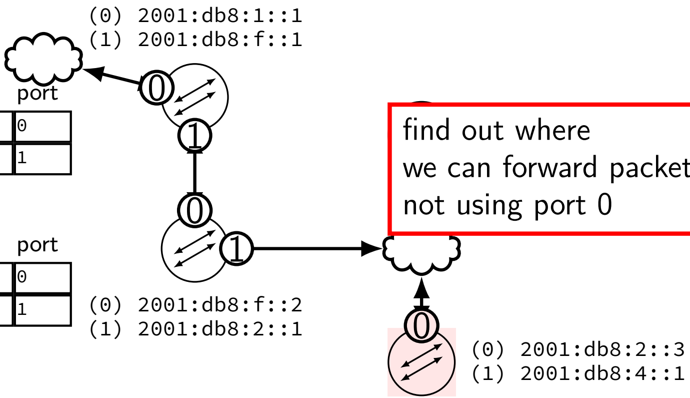
addresses	gateway	port
2001:db8:2::/40	---	0

‘flooding’

addresses	gateway	port
2001:db8:2::/40	---	0
2001:db8:3::/40	---	1

addresses	gateway	port
2001:db8:1::/40	---	0
2001:db8:f::/40	---	1

addresses	gateway	port
2001:db8:f::/40	---	0
2001:db8:2::/40	---	1



find out where
we can forward packets
not using port 0

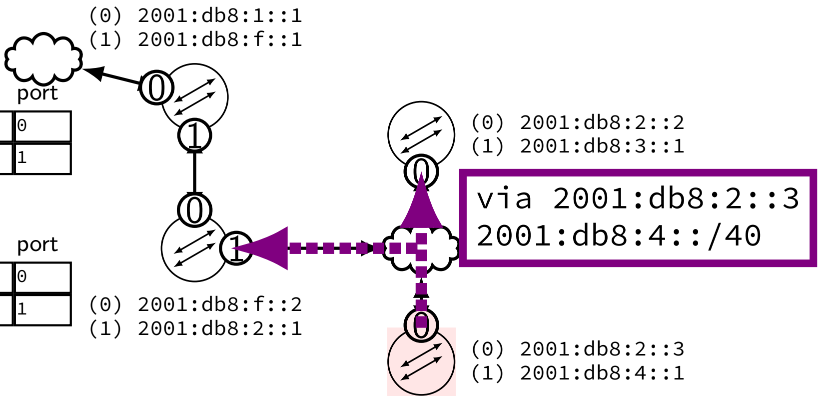
addresses	gateway	port
2001:db8:2::/40	---	0
2001:db8:4::/40	---	1

‘flooding’

addresses	gateway	port
2001:db8:2::/40	---	0
2001:db8:3::/40	---	1

addresses	gateway	port
2001:db8:1::/40	---	0
2001:db8:f::/40	---	1

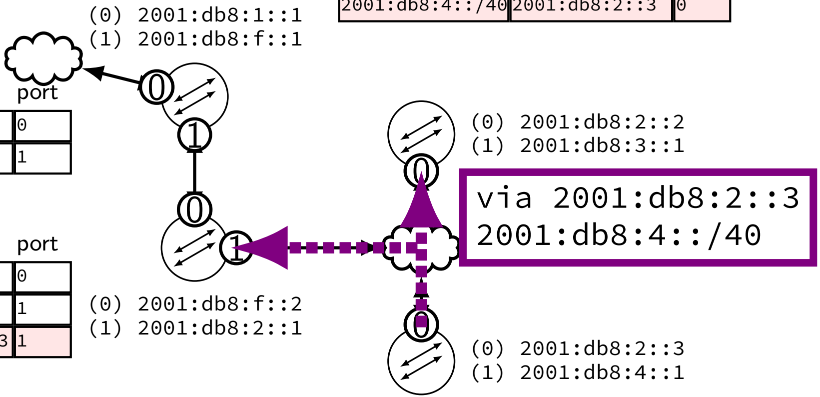
addresses	gateway	port
2001:db8:f::/40	---	0
2001:db8:2::/40	---	1



addresses	gateway	port
2001:db8:2::/40	---	0
2001:db8:4::/40	---	1

‘flooding’

addresses	gateway	port
2001:db8:2::/40	---	0
2001:db8:3::/40	---	1
2001:db8:4::/40	2001:db8:2::3	0



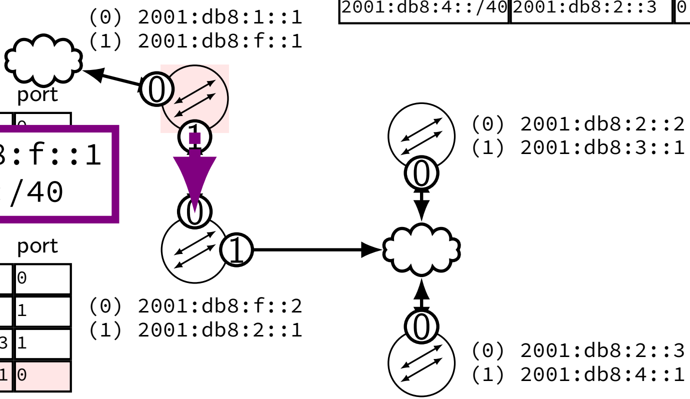
addresses	gateway	port
2001:db8:1::/40	---	0
2001:db8:f::/40	---	1

addresses	gateway	port
2001:db8:f::/40	---	0
2001:db8:2::/40	---	1
2001:db8:4::/40	2001::db8:2::3	1

addresses	gateway	port
2001:db8:2::/40	---	0
2001:db8:4::/40	---	1

‘flooding’

addresses	gateway	port
2001:db8:2::/40	---	0
2001:db8:3::/40	---	1
2001:db8:4::/40	2001:db8:2::3	0



addresses	gateway	port
2001:db8:1::/40		0
2001:db		

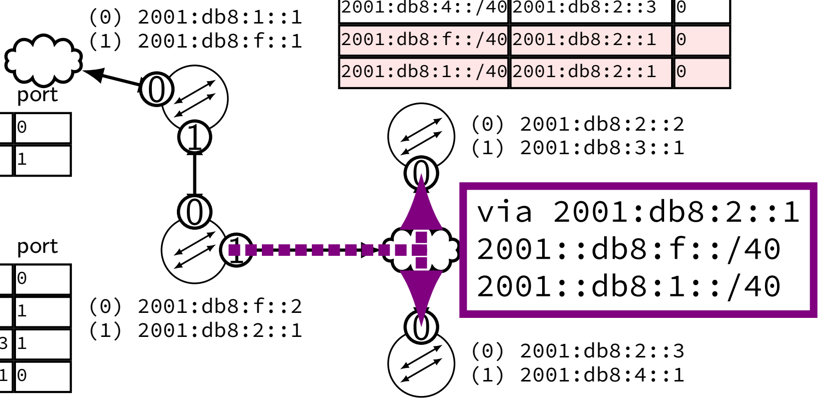
via 2001:db8:f::1
2001:db8:1::/40

addresses	gateway	port
2001:db8:f::/40	---	0
2001:db8:2::/40	---	1
2001:db8:4::/40	2001::db8:2::3	1
2001:db8:1::/40	2001::db8:f::1	0

addresses	gateway	port
2001:db8:2::/40	---	0
2001:db8:4::/40	---	1

‘flooding’

addresses	gateway	port
2001:db8:2::/40	---	0
2001:db8:3::/40	---	1
2001:db8:4::/40	2001:db8:2::3	0
2001:db8:f::/40	2001:db8:2::1	0
2001:db8:1::/40	2001:db8:2::1	0



addresses	gateway	port
2001:db8:1::/40	---	0
2001:db8:f::/40	---	1

addresses	gateway	port
2001:db8:f::/40	---	0
2001:db8:2::/40	---	1
2001:db8:4::/40	2001::db8:2::3	1
2001:db8:1::/40	2001::db8:f::1	0

addresses	gateway	port
2001:db8:2::/40	---	0
2001:db8:4::/40	---	1
2001:db8:f::/40	2001:db8:2::1	0
2001:db8:1::/40	2001:db8:2::1	0

eventual convergence

‘flooding’ algorithm:

periodically send on each network:

> list of routes you have that don’t double-back to same network

when receiving routes sent on network:

> add routing table entry for each route

eventual convergence

'flooding' algorithm:

periodically send on each network:

list of routes you have that don't double-back to same network

when receiving routes sent on network:

add routing table entry for each route

not handled: multiple paths?

only one path?

only one path on network means:

if a link fails, bad news

network forms a tree

routing like this?

for IP routing, generally want to have multiple paths

...but this is basically how MAC learning works

but it requires a network that is a tree

what if we don't start with one?

spanning tree

given a general network, only activate subset of links

...such that network is tree

that is only one path between each node

allows us to do flooding strategy

makes simple MAC learning/broadcast just work

centralized spanning tree?

one algorithm you might learn in DSA2:

mark one node called *the root* as 'in the tree'

repeatedly:

add the 'first' link that goes to a node not in the tree
mark newly connected node as 'in the tree'

result $=$ spanning tree

centralized spanning tree?

one algorithm you might learn in DSA2:

mark one node called *the root* as 'in the tree'

repeatedly:

add the 'first' link that goes to a node not in the tree
mark newly connected node as 'in the tree'

result = spanning tree

a careful ordering

algorithm works with any idea of which link/node is first

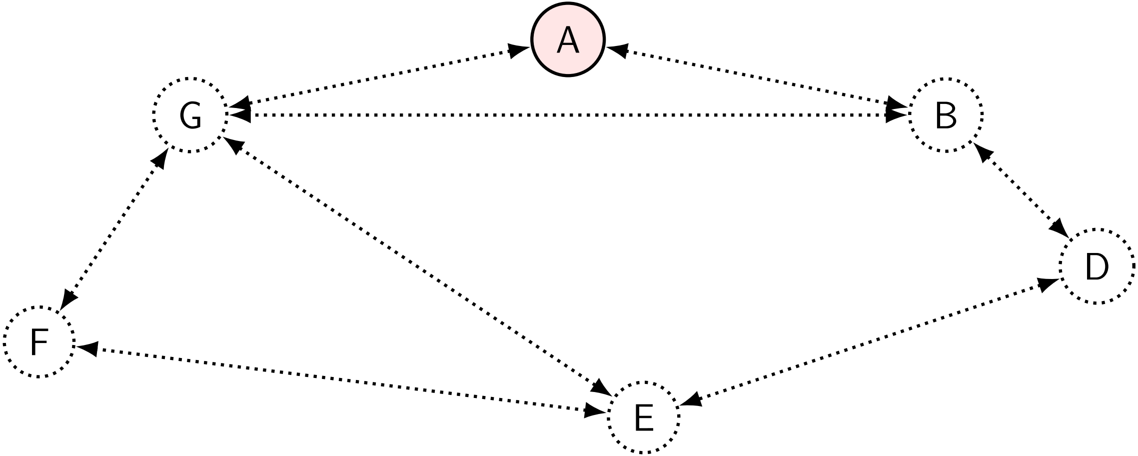
we'll choose a particular ordering (for reasons you'll see later)

root (first node) is one with earliest 'name'

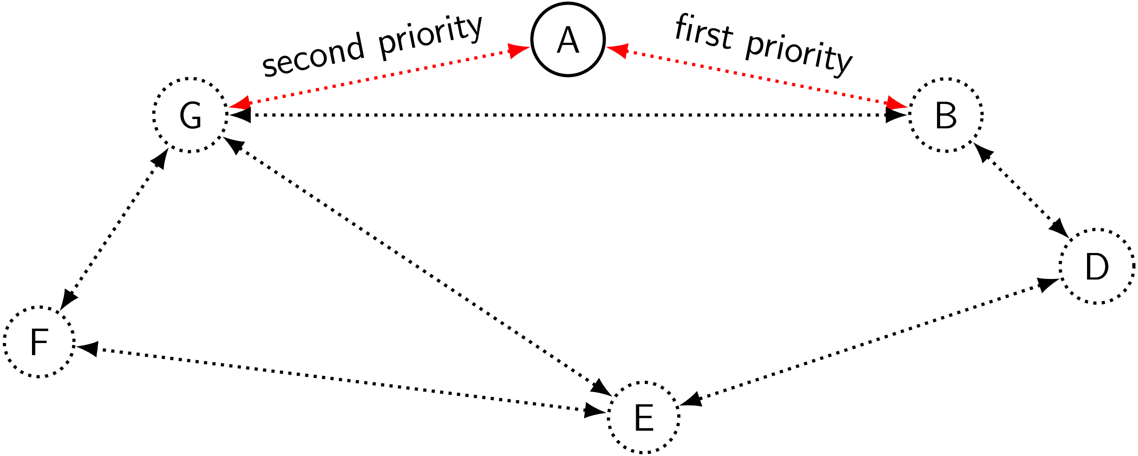
links closer to the root before further links

links from nodes with earlier names before later ones

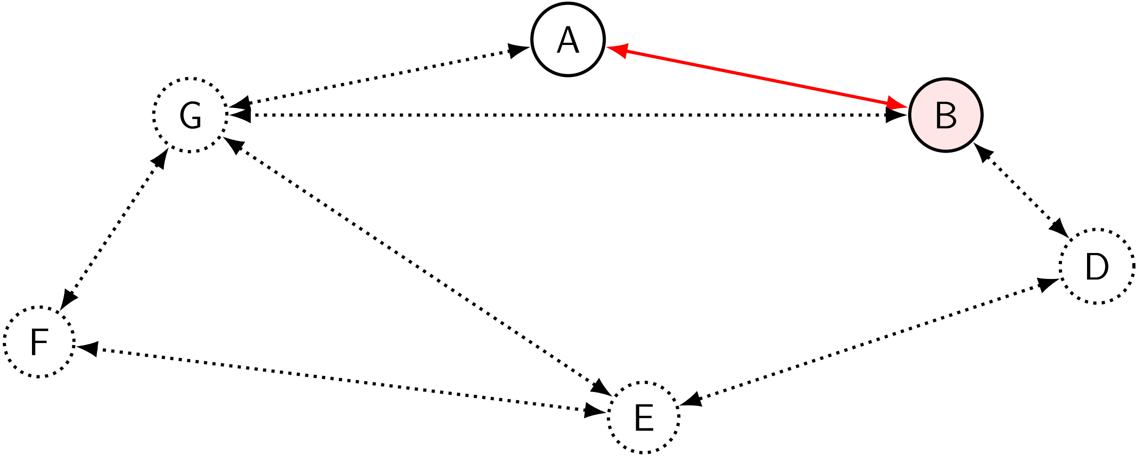
spanning tree example



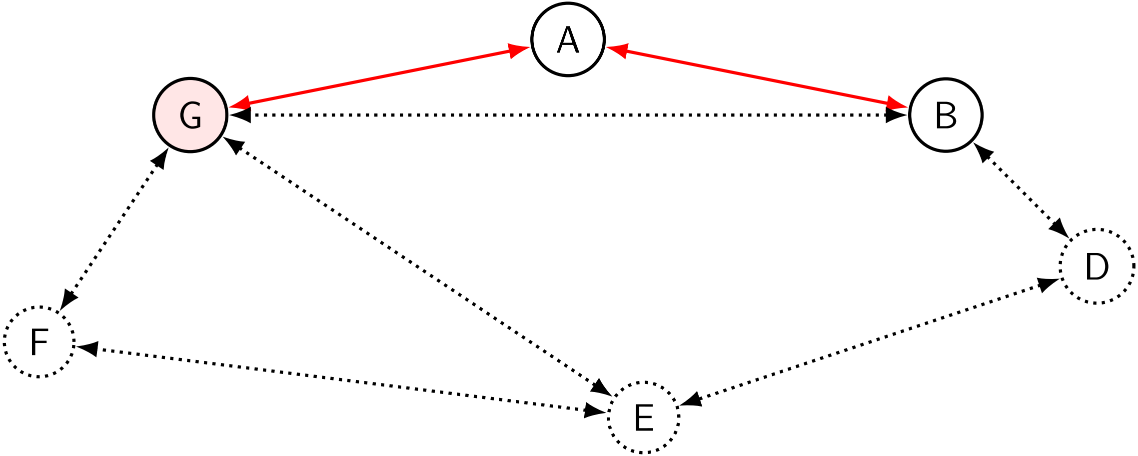
spanning tree example



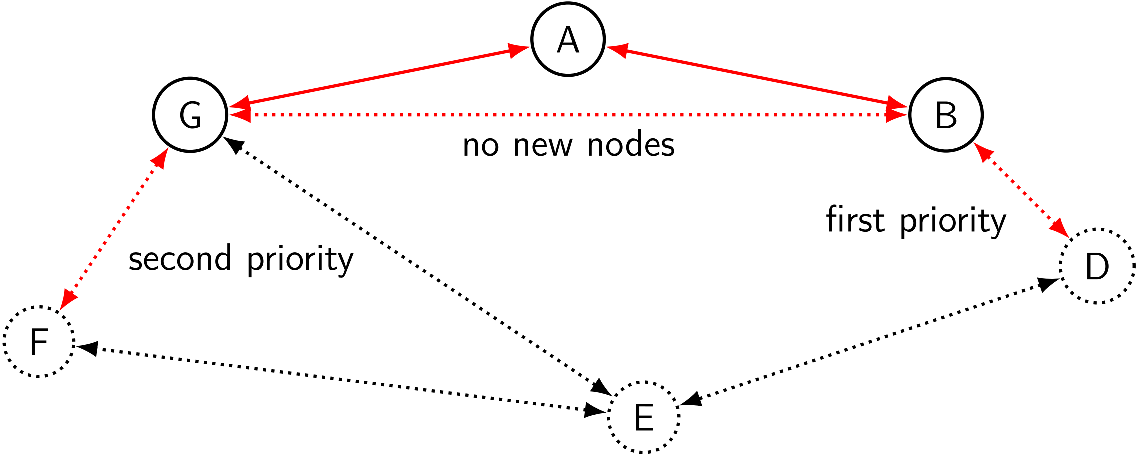
spanning tree example



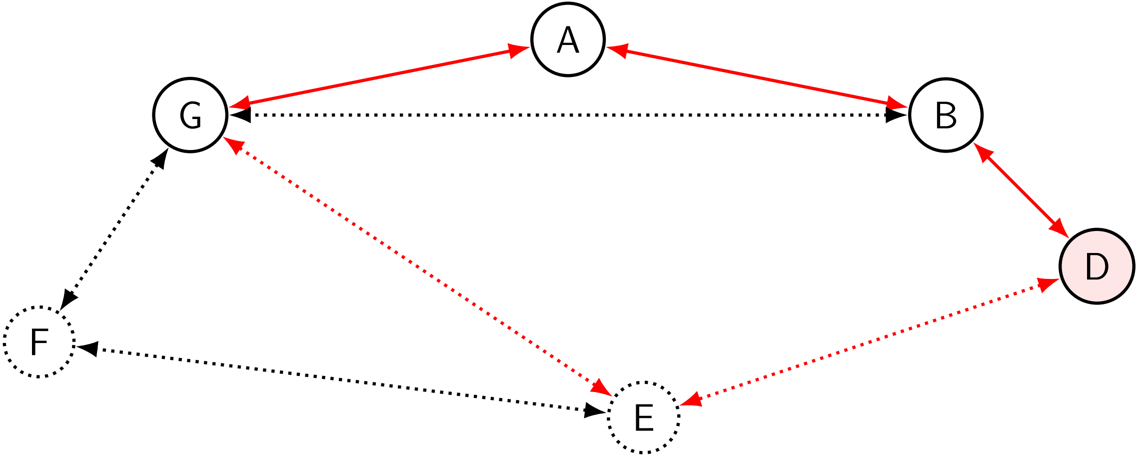
spanning tree example



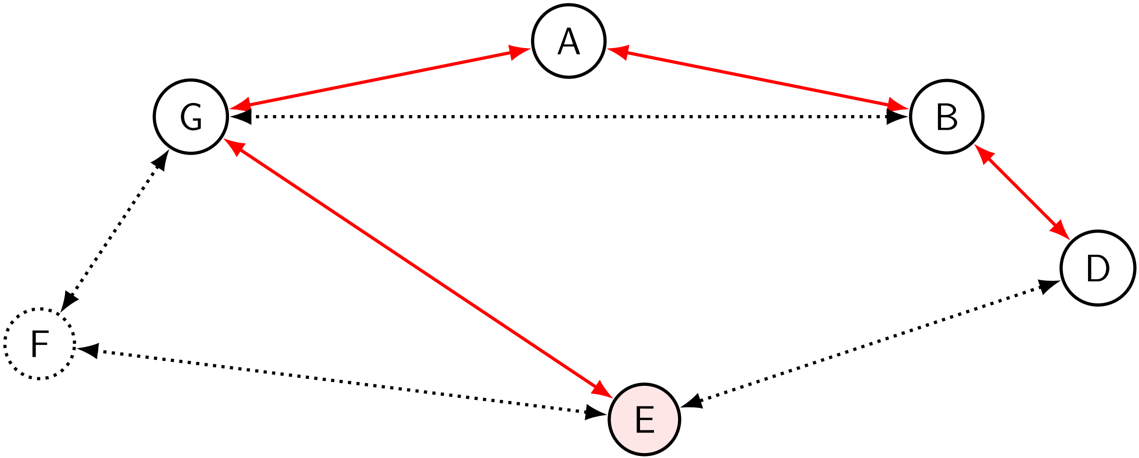
spanning tree example



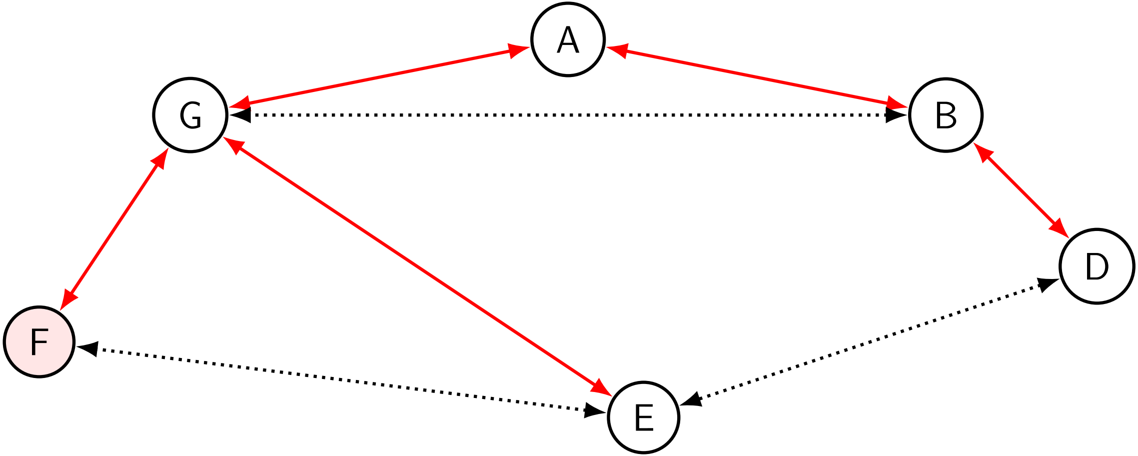
spanning tree example



spanning tree example



spanning tree example



detecting ‘mistakes’

this method: consistent results every time

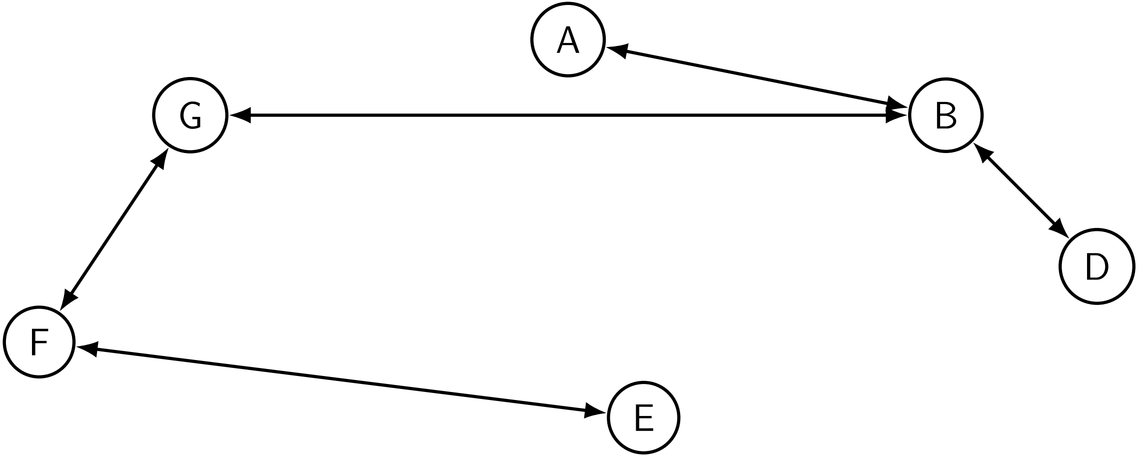
but assumes we start from scratch

we’re going to want a way of doing this dynamically

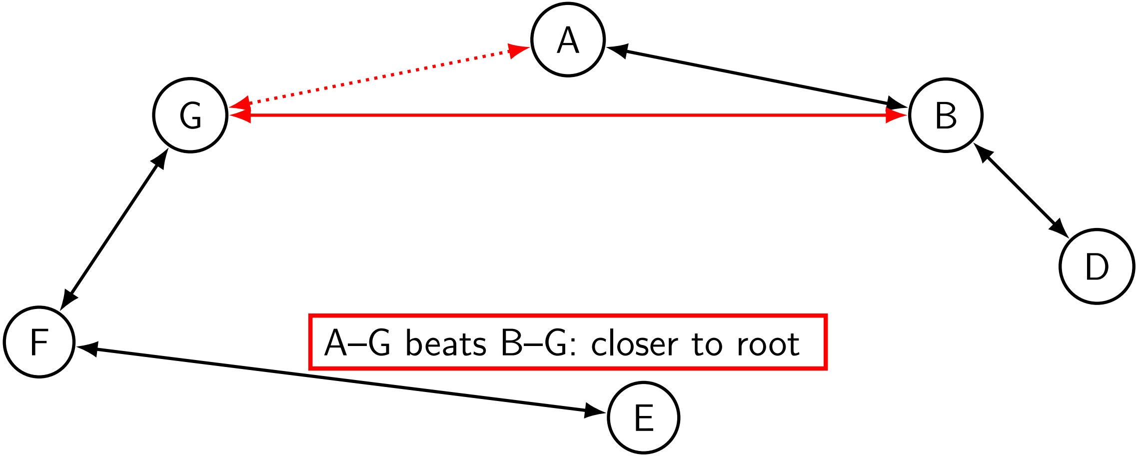
let’s say we find a wrong configuration —

can we fix it?

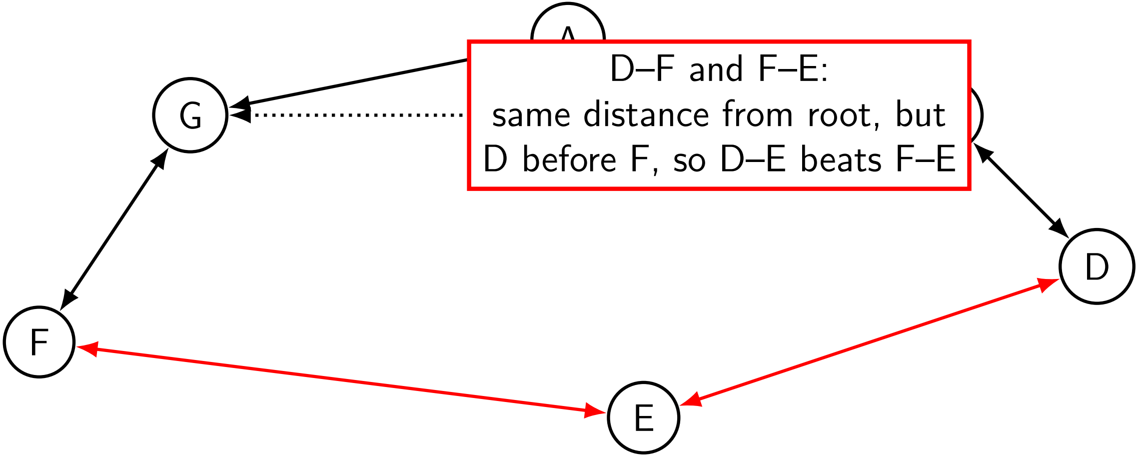
fixing wrong links



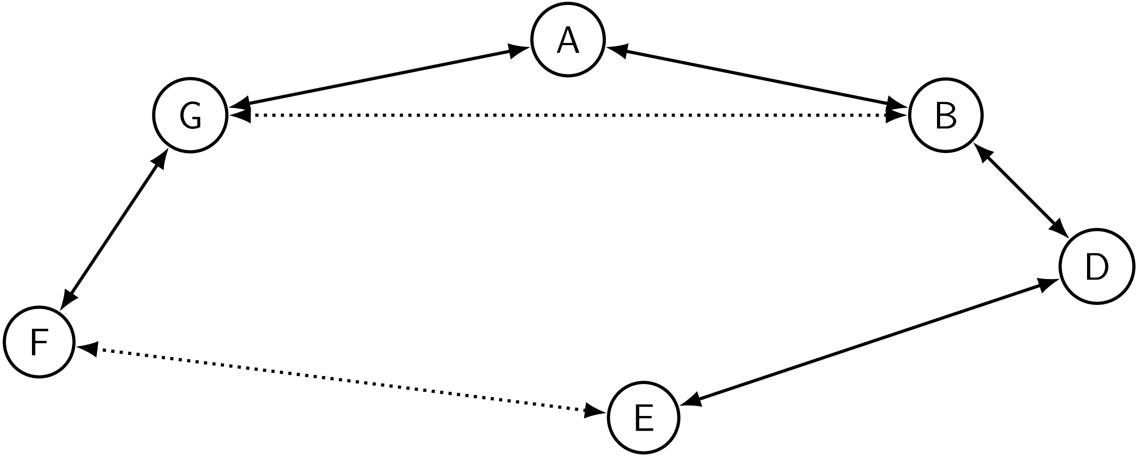
fixing wrong links



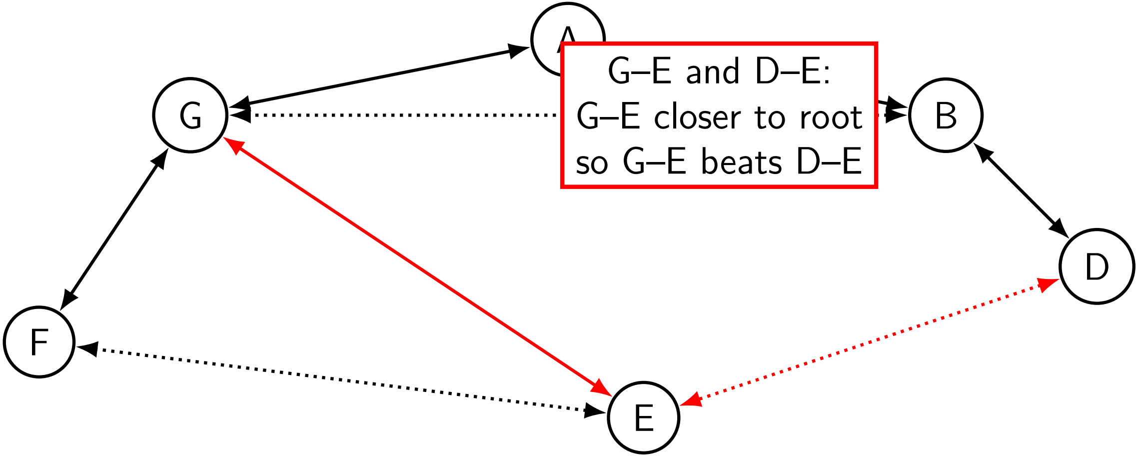
fixing wrong links



fixing wrong links



fixing wrong links



spanning tree protocol

each node tracks:

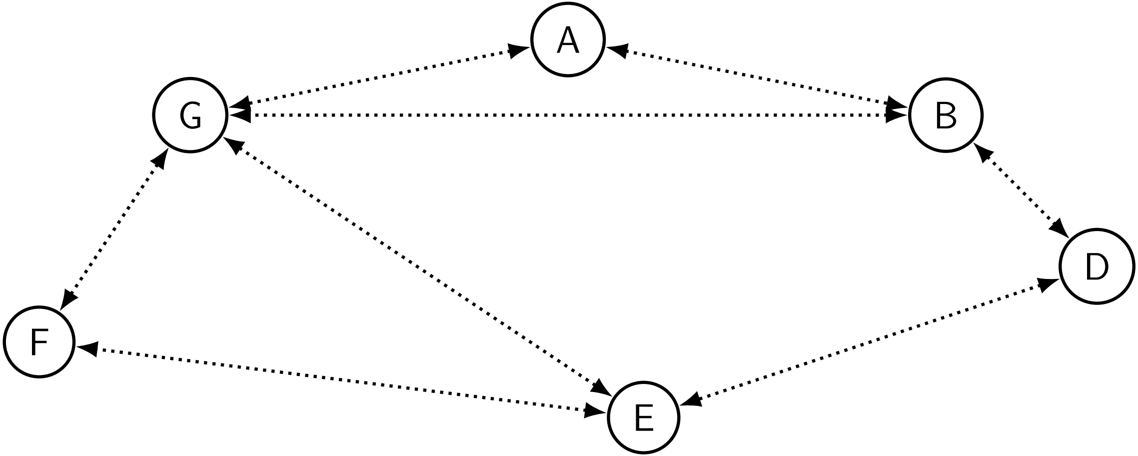
- what it believes is root of tree
- its link toward root of tree
- its distance to root of tree
- which other nodes think it's closer to root of tree

periodically sends information to neighbors

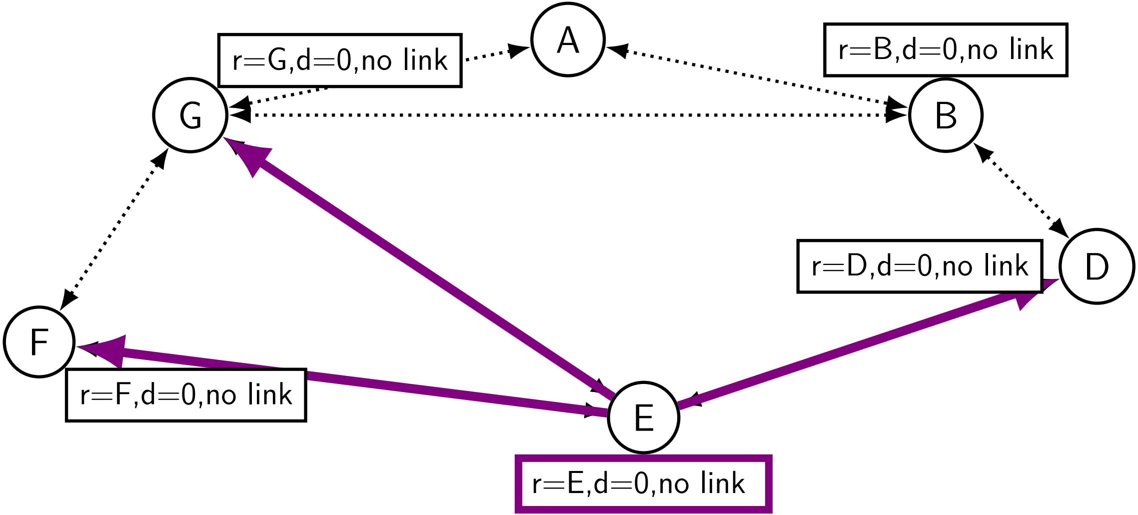
when receiving information, update:

- root to lower ID number (if possible)
- link to lower-distance link (if possible)
- link to lower-ID, same-distance link (if possible)
- which other nodes think it is closer

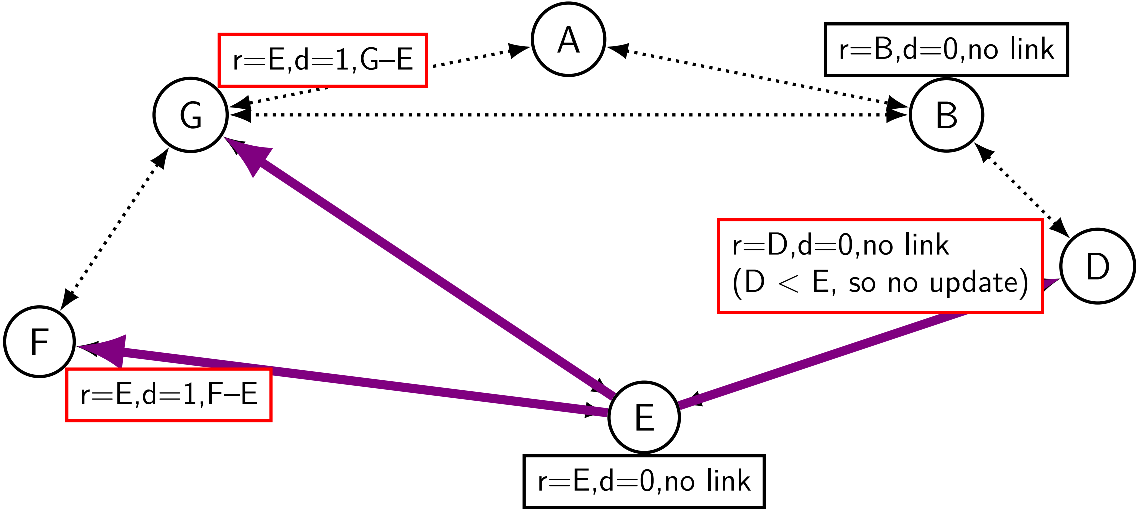
some example updates



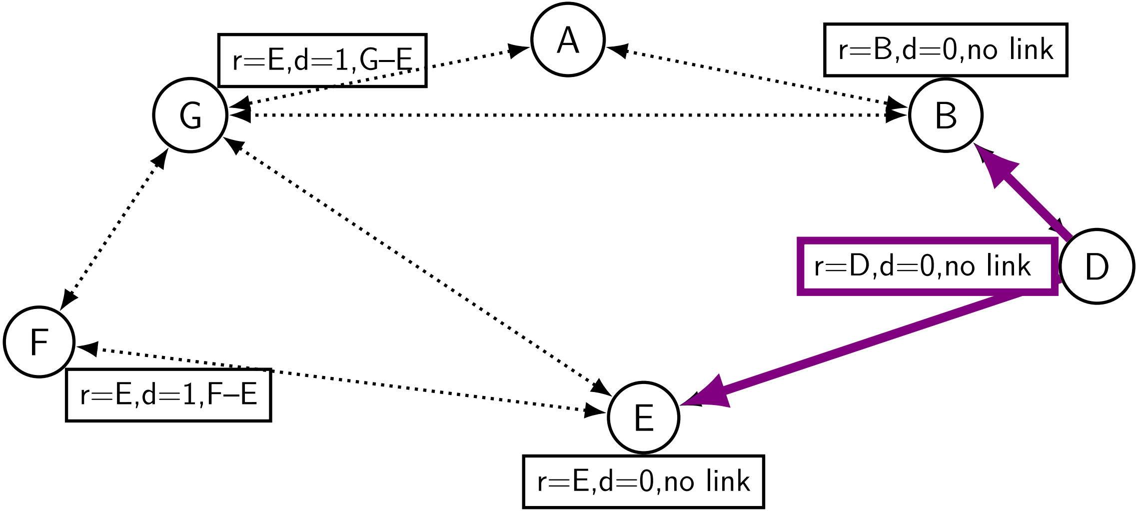
some example updates



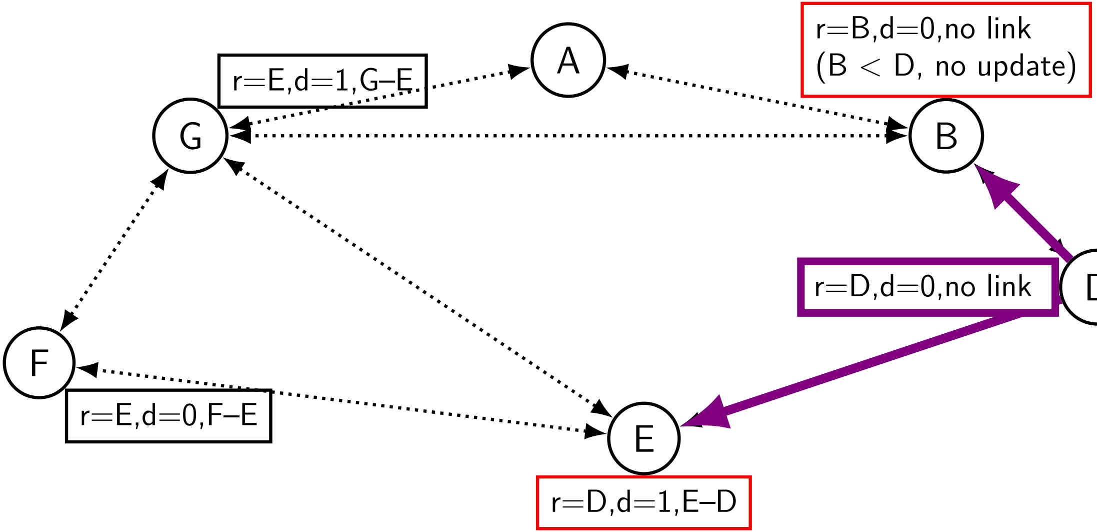
some example updates



some example updates



some example updates



spanning trees in practice (1)

commonly used on Ethernet for switches

links not in spanning tree are 'blocked'

- not used for normal traffic
- assumption: would cause loop $\rightarrow$ infinite packets

delay before activating port

- avoid temporary routing loops while figuring out tree

periodically send updates to all neighbors

- order of seconds

spanning tree in practice (2)

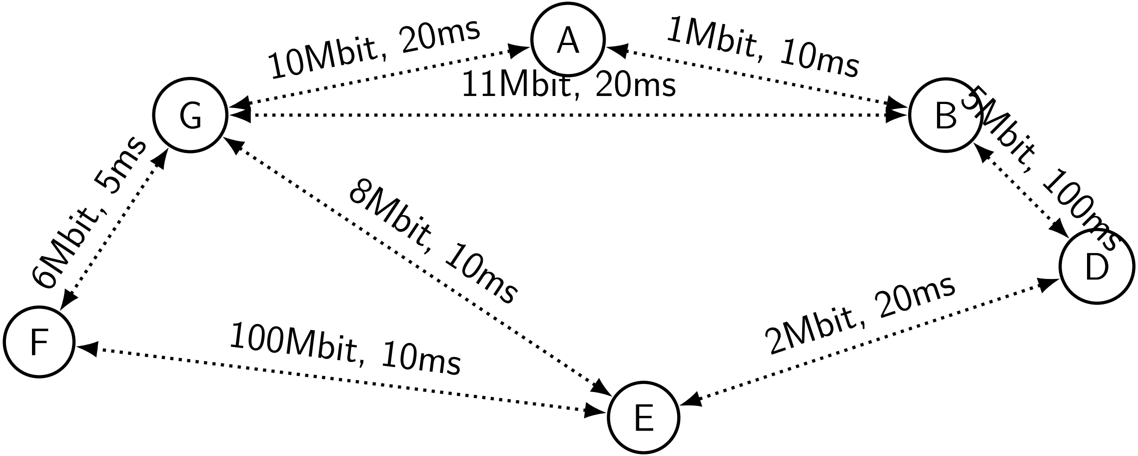
real protocol supports variable ‘cost’ for links

so ‘distance to root’ might be lower for faster links

modern variant (Rapid Spanning Tree Protocol) selects “backup” port to root

goal: faster switchover on failure

exercise: best routes?



A to B? B to E? F to G?

routing metrics

want some way of saying how ‘good’ link is

typically “cost”/“distance” value (so lower is better)

in practice, most commonly

$$\frac{\text{constant}}{\text{bandwidth}}$$

could also try to:

- take financial costs into account
- take lantency into account
- take reliability into account
- spread flows out among more links

all-pairs Bellman-Ford

one algorithm to find all shortest paths in graph (network)

$d(A, B)$ = best distance from A to B

$p(A, B)$ = next node on path from A to B

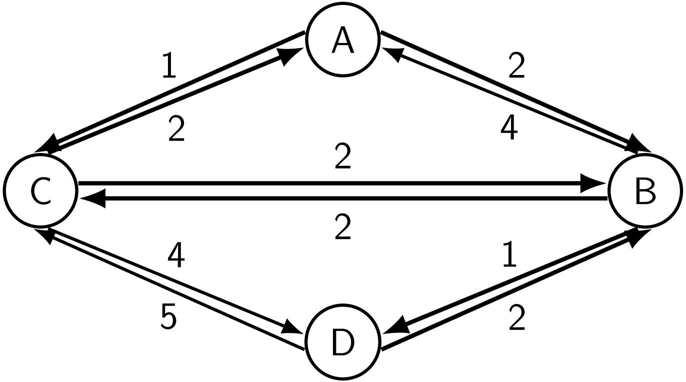
initially $d(X, X) = \infty$ for all nodes X

repeatedly* do the following:

for each link from A to B, distance c:

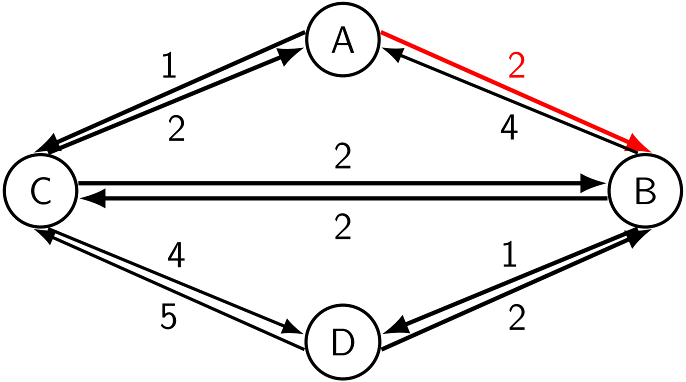
for each node X:
if $c + d(B, X) < d(A, X)$,
then $d(A, X) \leftarrow c + d(B, X)$, $p(A, X) = B$

running Bellman-Ford



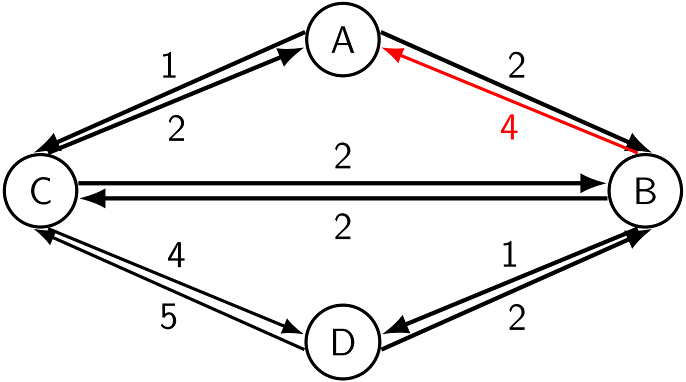
	A	B	C	D
A	0/A	∞/—	∞/—	∞/—
B	∞/—	0/B	∞/—	∞/—
C	∞/—	∞/—	0/C	∞/—
D	∞/—	∞/—	∞/—	0/D

running Bellman-Ford



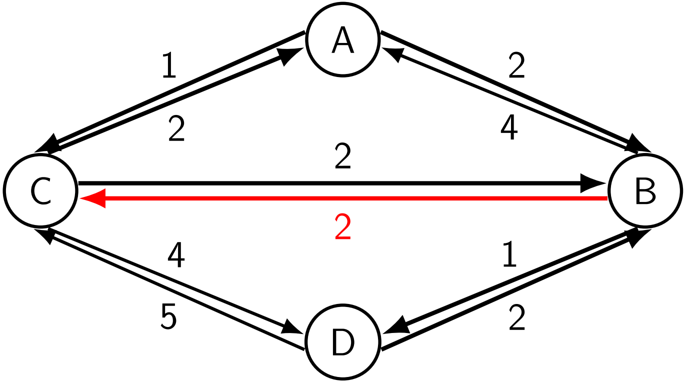
	A	B	C	D
A	0/A	2/B	∞/—	∞/—
B	∞/—	0/B	∞/—	∞/—
C	∞/—	∞/—	0/C	∞/—
D	∞/—	∞/—	∞/—	0/D

running Bellman-Ford



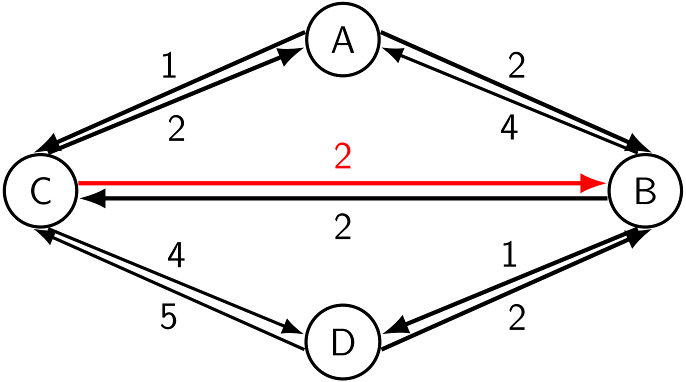
	A	B	C	D
A	0/A	2/B	∞/—	∞/—
B	4/A	0/B	∞/—	∞/—
C	∞/—	∞/—	0/C	∞/—
D	∞/—	∞/—	∞/—	0/D

running Bellman-Ford



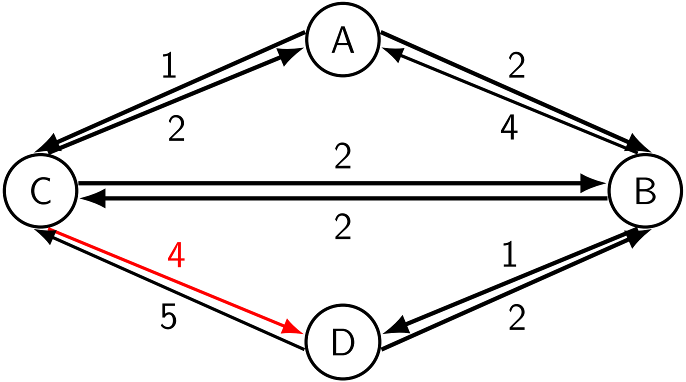
	A	B	C	D
A	0/A	2/B	∞/—	∞/—
B	4/A	0/B	2/C	∞/—
C	∞/—	∞/—	0/C	∞/—
D	∞/—	∞/—	∞/—	0/D

running Bellman-Ford



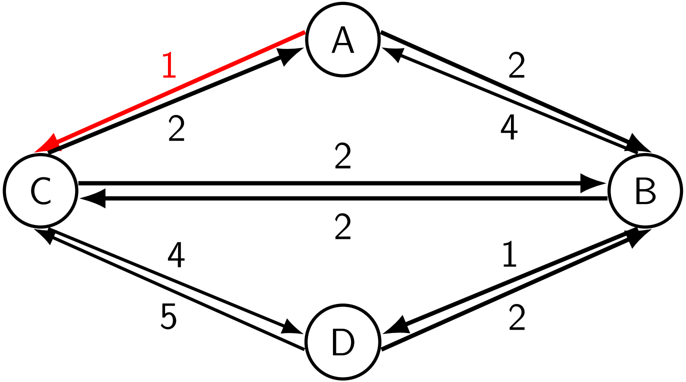
	A	B	C	D
A	0/A	2/B	∞/—	∞/—
B	4/A	0/B	2/C	∞/—
C	6/B	2/B	0/C	∞/—
D	∞/—	∞/—	∞/—	0/D

running Bellman-Ford



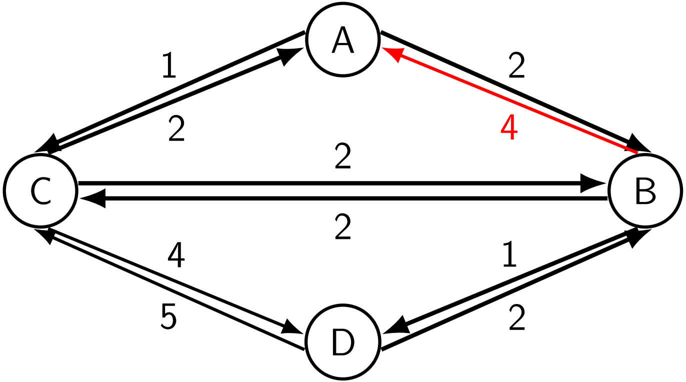
	A	B	C	D
A	0/A	2/B	∞/—	∞/—
B	4/A	0/B	2/C	∞/—
C	6/B	2/B	0/C	4/D
D	∞/—	∞/—	∞/—	0/D

running Bellman-Ford



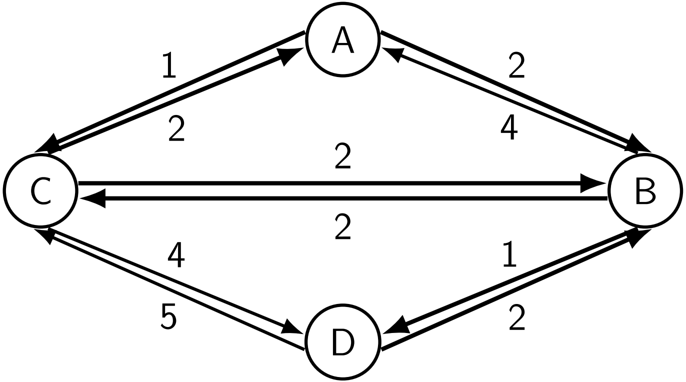
	A	B	C	D
A	0/A	2/B	1/C	5/D
B	4/A	0/B	2/C	∞/—
C	6/B	2/B	0/C	4/D
D	∞/—	∞/—	∞/—	0/D

running Bellman-Ford



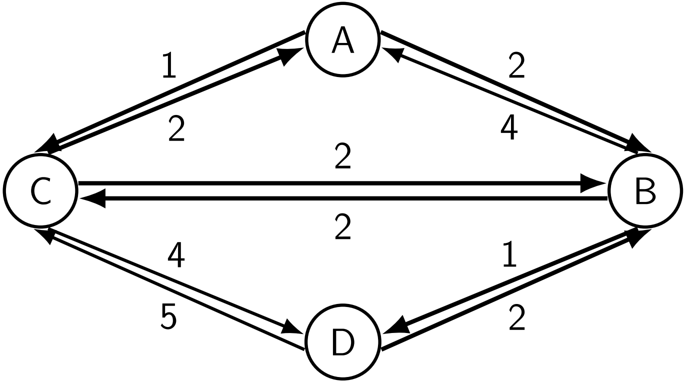
	A	B	C	D
A	0/A	2/B	1/C	5/D
B	4/A	0/B	2/C	9/A
C	6/B	2/B	0/C	4/D
D	∞/—	∞/—	∞/—	0/D

running Bellman-Ford



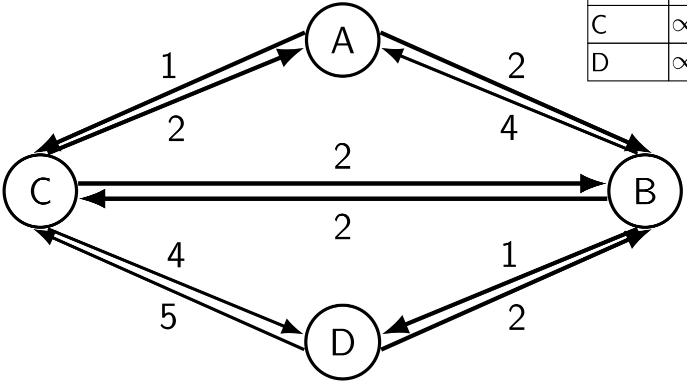
	A	B	C	D
A	0/A	2/B	1/C	5/D
B	4/A	0/B	2/C	9/A
C	6/B	2/B	0/C	4/D
D	∞/—	∞/—	∞/—	0/D

running Bellman-Ford



	A	B	C	D
A	0/A	2/B	1/C	3/B
B	4/A	0/B	2/C	1/D
C	2/A	2/B	0/C	4/D
D	6/B	2/B	4/B	0/D

distributing Bellman-Ford



	A	B	C	D
A	0/A	∞/—	∞/—	∞/—
B	∞/—	0/B	∞/—	∞/—
C	∞/—	∞/—	0/C	∞/—
D	∞/—	∞/—	∞/—	0/D

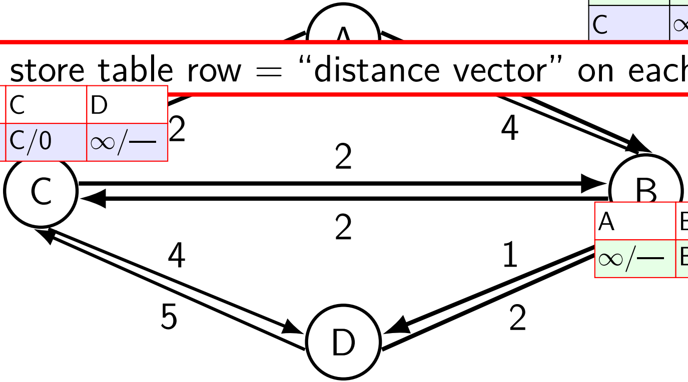
distributing Bellman-Ford

	A	B	C	D
A	0/A	∞/—	∞/—	∞/—
B	∞/—	0/B	∞/—	∞/—
C	∞/—	∞/—	0/C	∞/—
		—	∞/—	0/D

store table row = “distance vector” on each node

A	B	C	D
A/0	∞/—	∞/—	∞/—

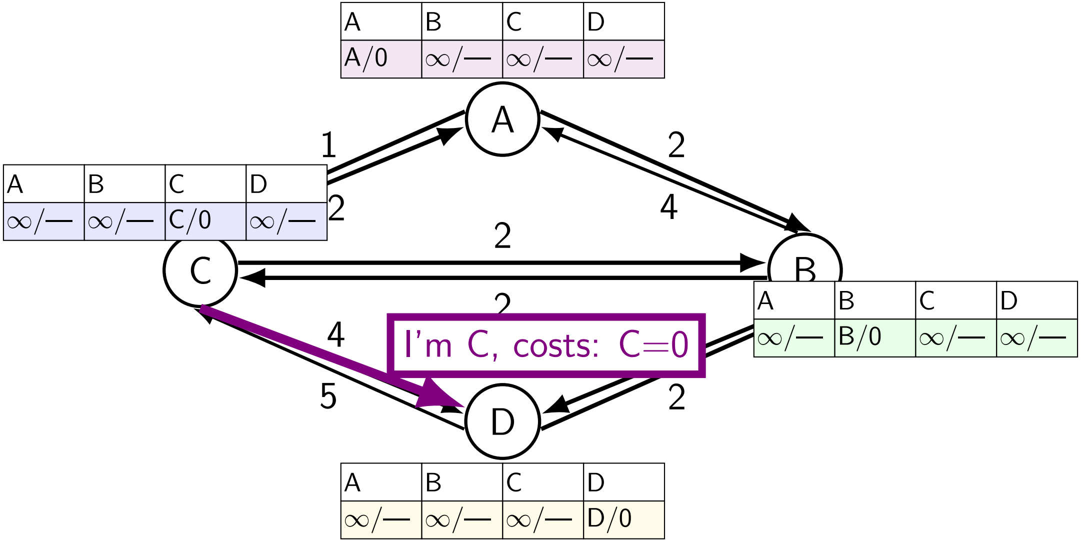
A	B	C	D
∞/—	∞/—	C/0	∞/—



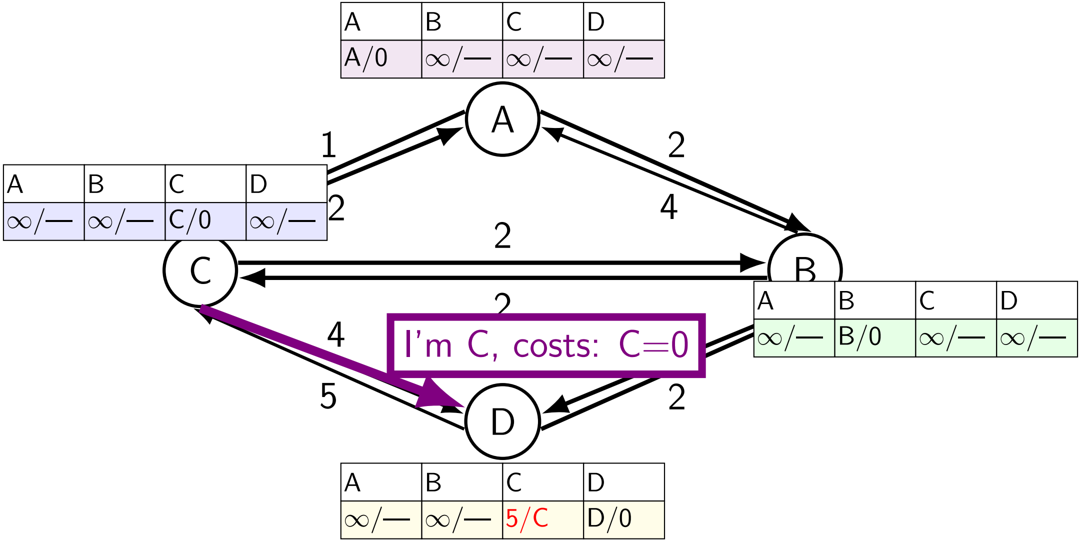
A	B	C	D
∞/—	B/0	∞/—	∞/—

A	B	C	D
∞/—	∞/—	∞/—	D/0

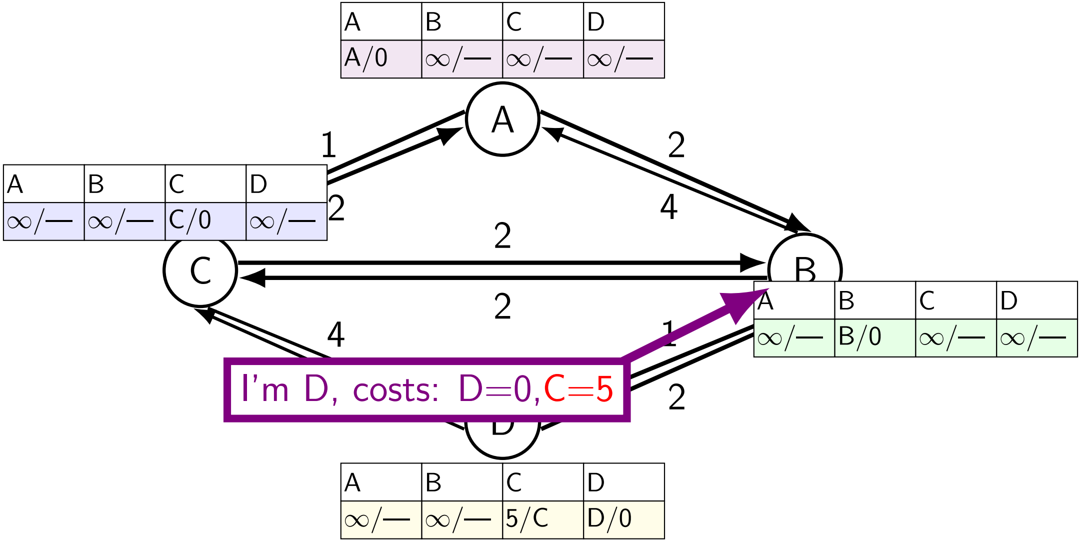
distributing Bellman-Ford



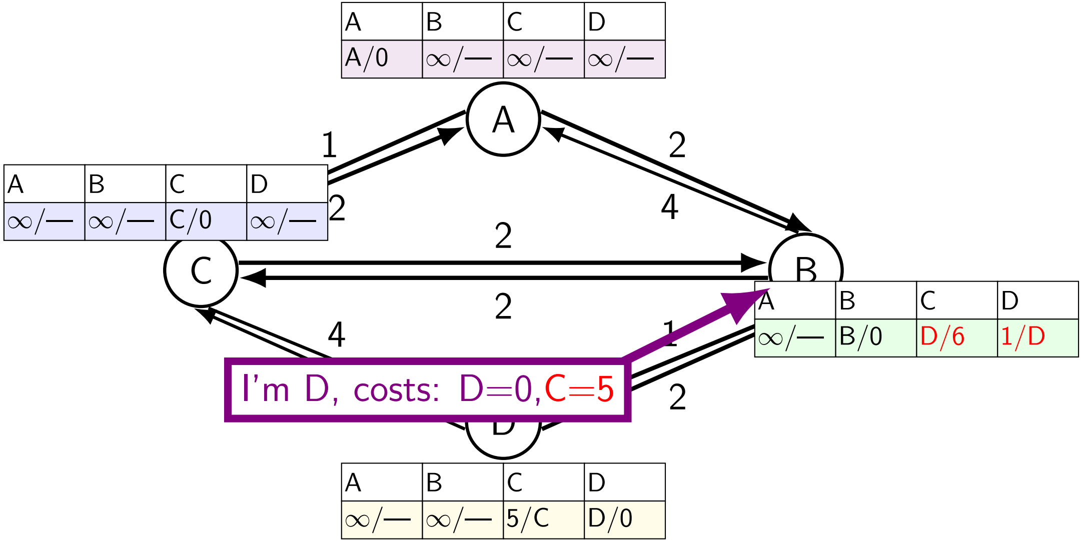
distributing Bellman-Ford



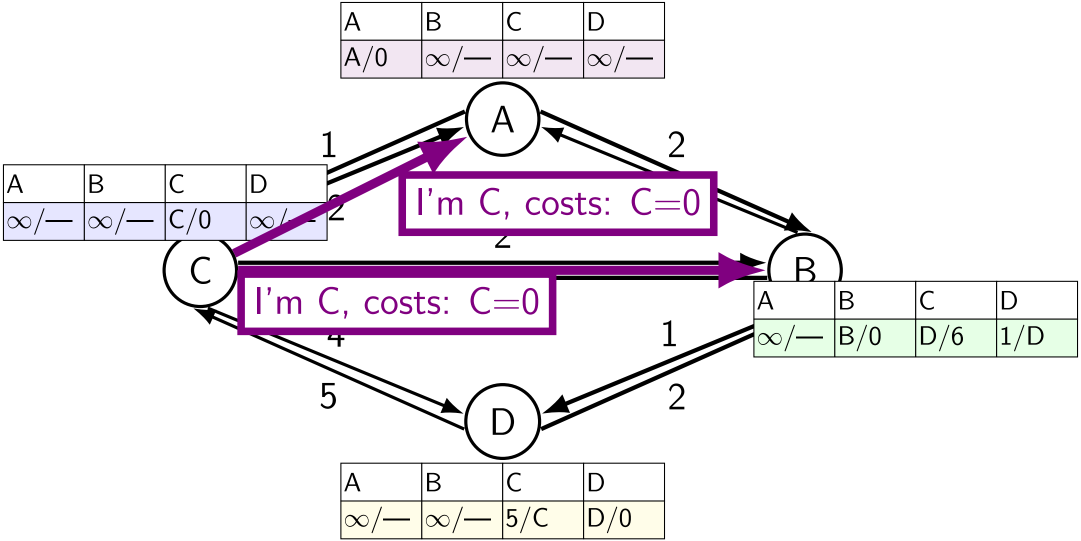
distributing Bellman-Ford



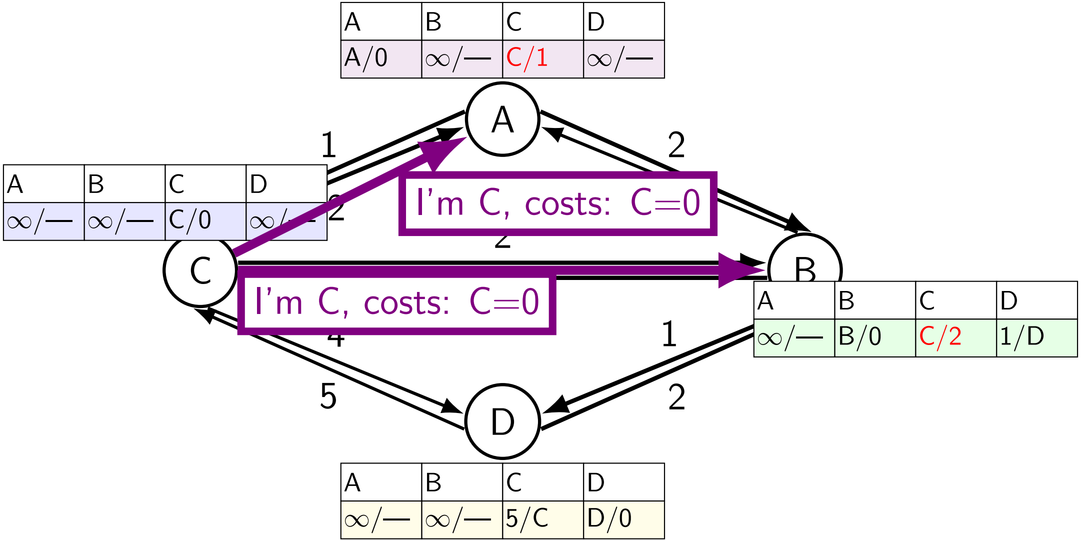
distributing Bellman-Ford



distributing Bellman-Ford

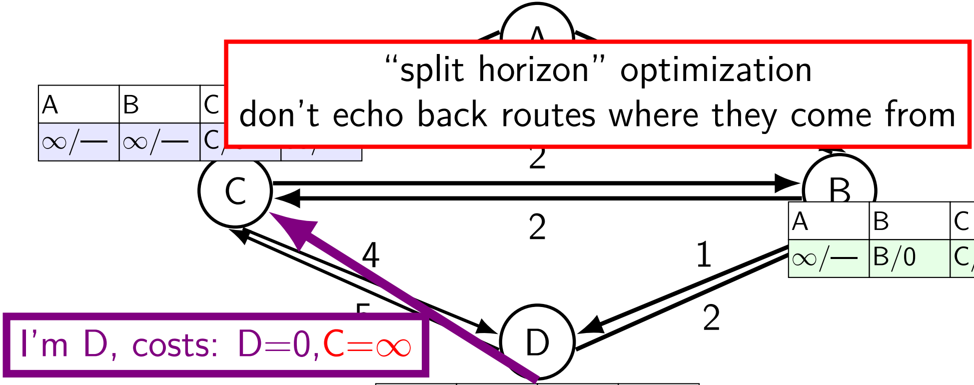


distributing Bellman-Ford



distributing Bellman-Ford

A	B	C	D
A/0	∞/—	C/1	∞/—

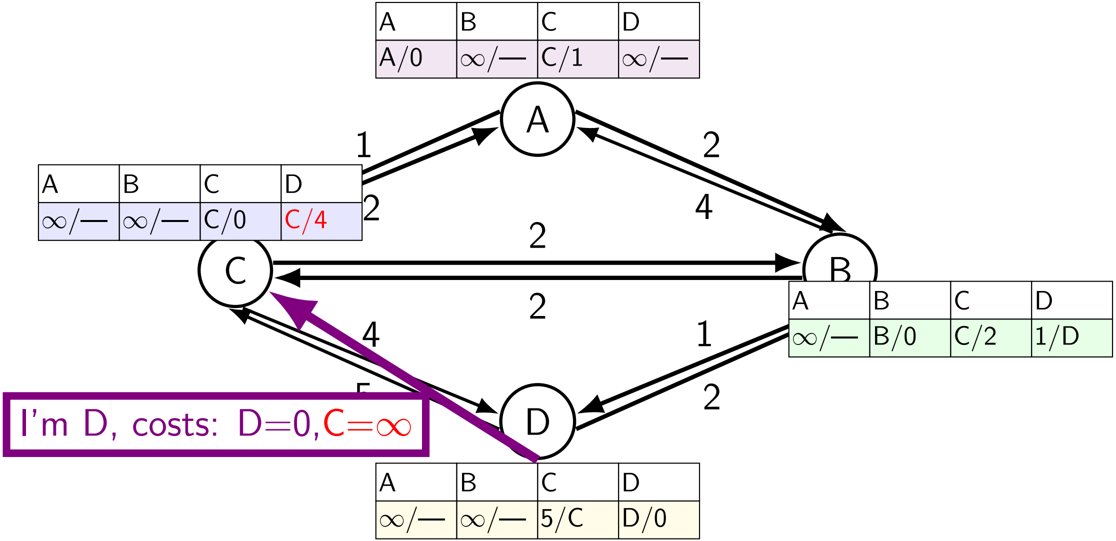


A	B	C	[illegible]
∞/—	∞/—	C/[illegible]	[illegible]

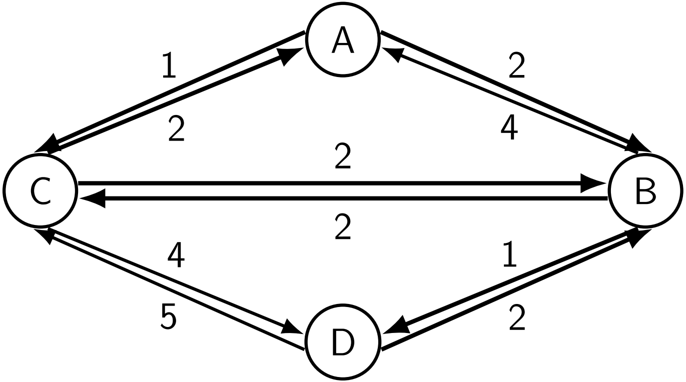
A	B	C	D
∞/—	B/0	C/2	1/D

A	B	C	D
∞/—	∞/—	5/C	D/0

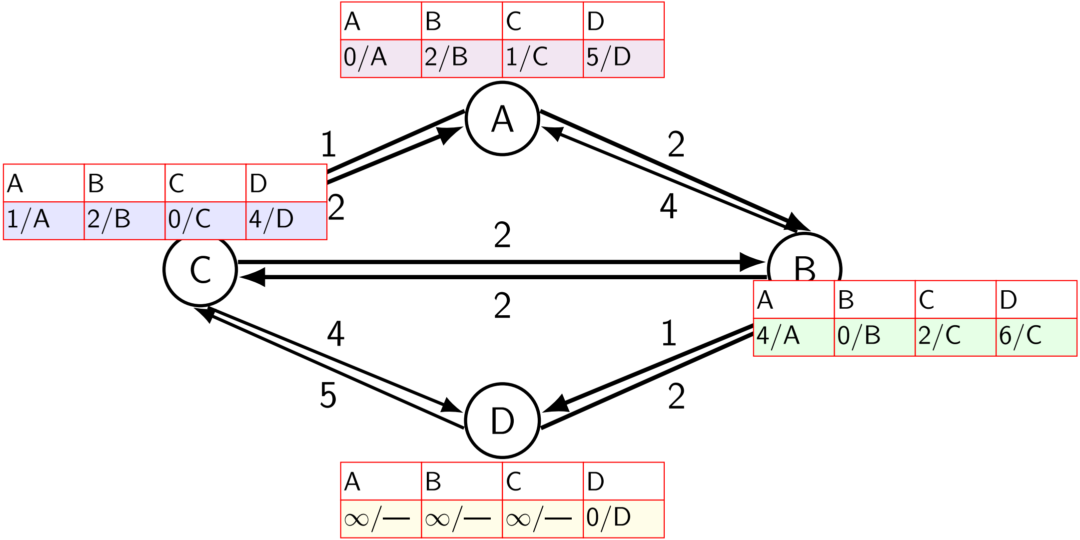
distributing Bellman-Ford



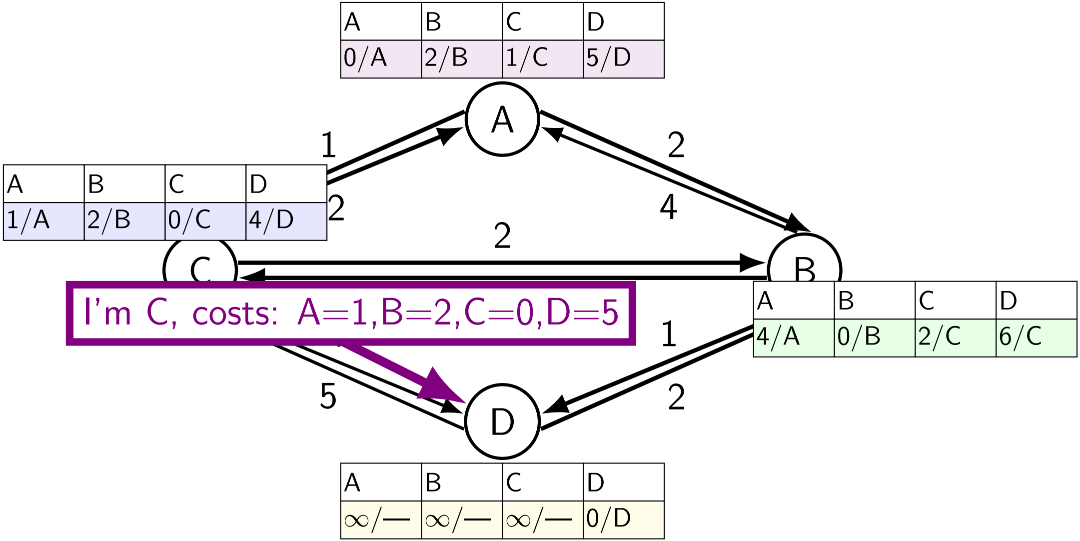
distributing Bellman-Ford (2)



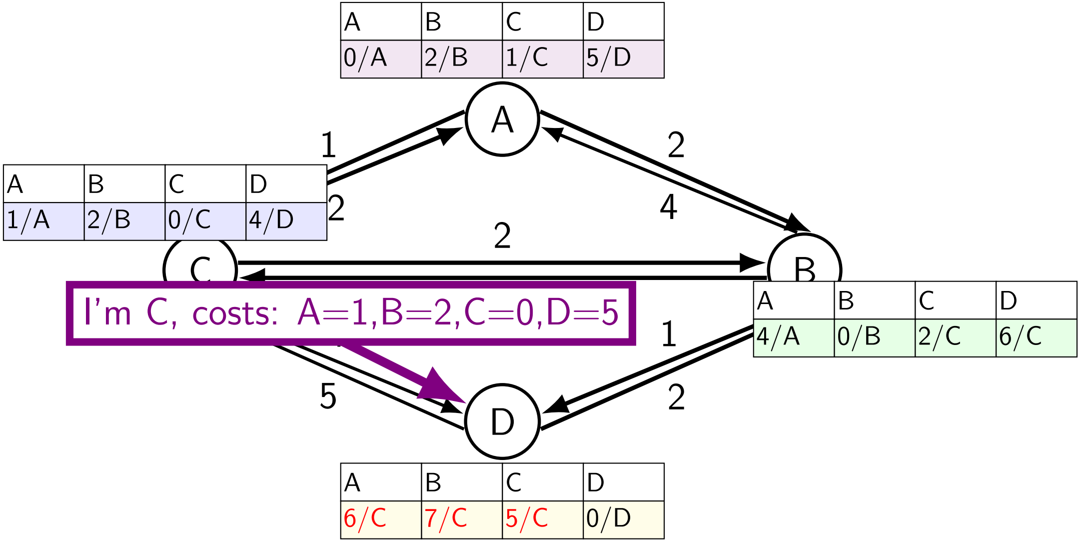
distributing Bellman-Ford (2)



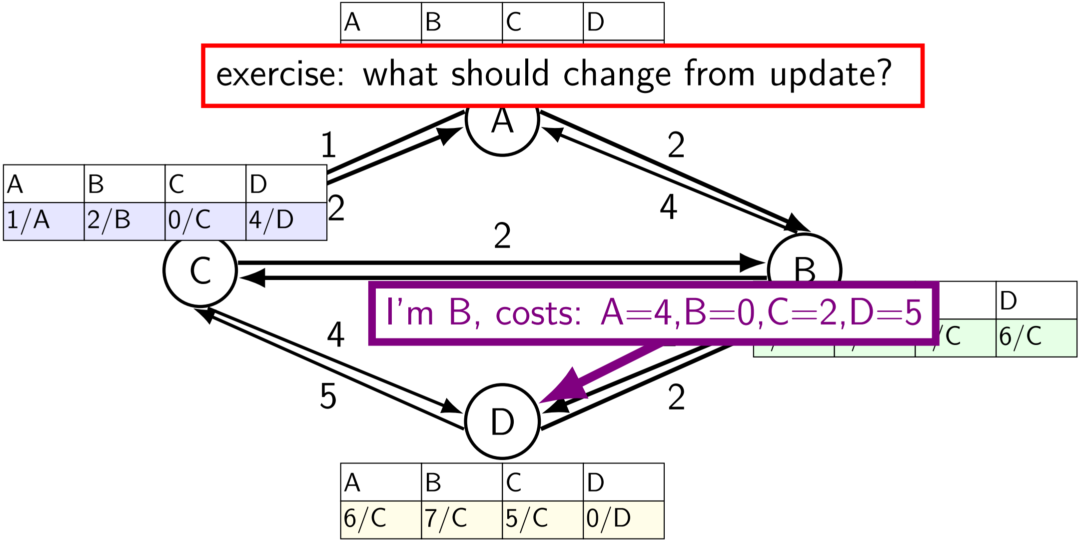
distributing Bellman-Ford (2)



distributing Bellman-Ford (2)

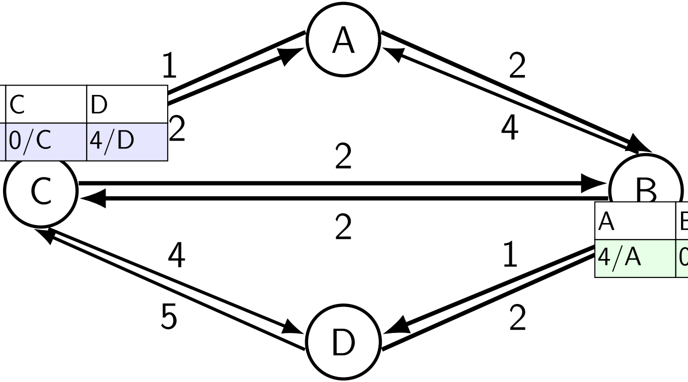


distributing Bellman-Ford (2)



distributing Bellman-Ford (2)

A	B	C	D
0/A	2/B	1/C	5/D



A	B	C	D
1/A	2/B	0/C	4/D

A	B	C	D
4/A	0/B	2/C	6/C

A	B	C	D
6/C	2/B	5/C	0/D

networks v routers (DV)

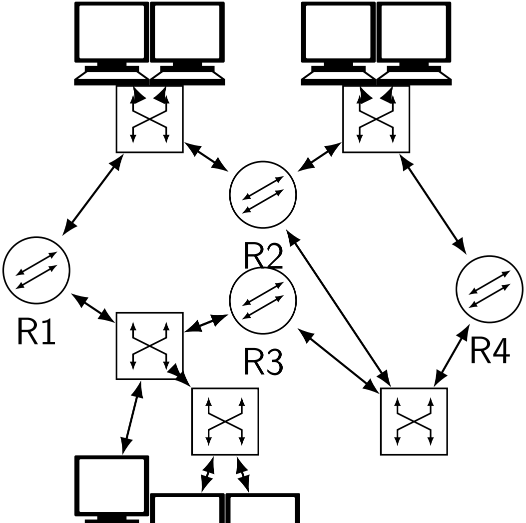
imprecision on graphs — acting as if we want distance to routers

but really want distance to networks

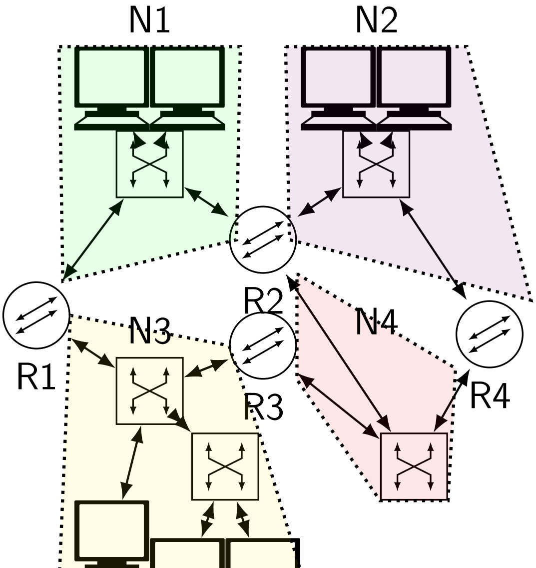
distance vectors will track distance to *networks*

but next hops will be routers

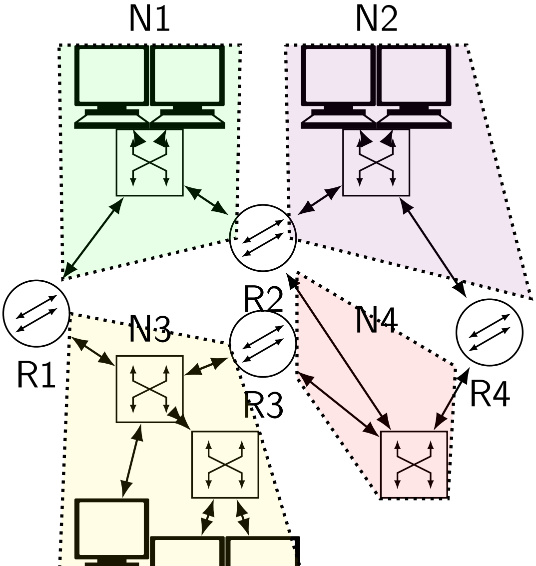
networks v routers (DV)

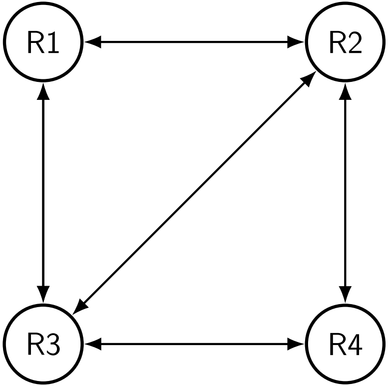


networks v routers (DV)

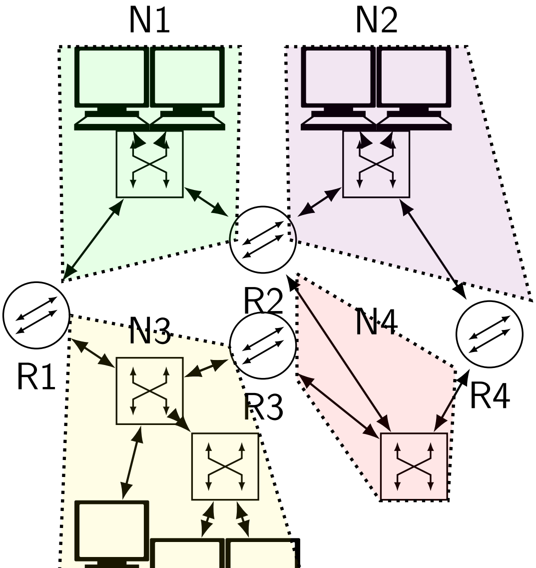


networks v routers (DV)



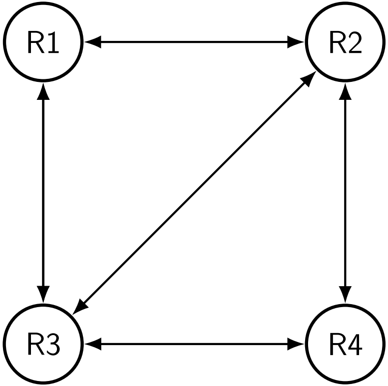


networks v routers (DV)



N1	N2	N3	N4
1/R1	∞/—	1/R1	∞/—

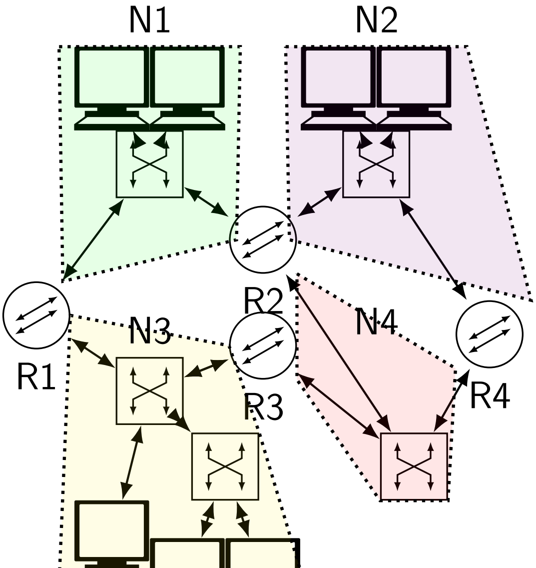
N1	N2	N3	N4
1/R2	1/R2	∞/—	1/R2



N1	N2	N3	N4
∞/—	∞/—	1/R3	1/R3

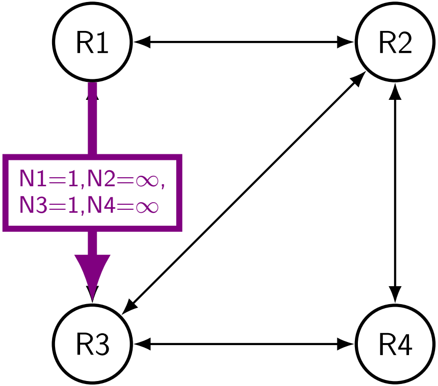
N1	N2	N3	N4
∞/—	1/R4	∞/—	1/R4

networks v routers (DV)



N1	N2	N3	N4
1/R1	∞/—	1/R1	∞/—

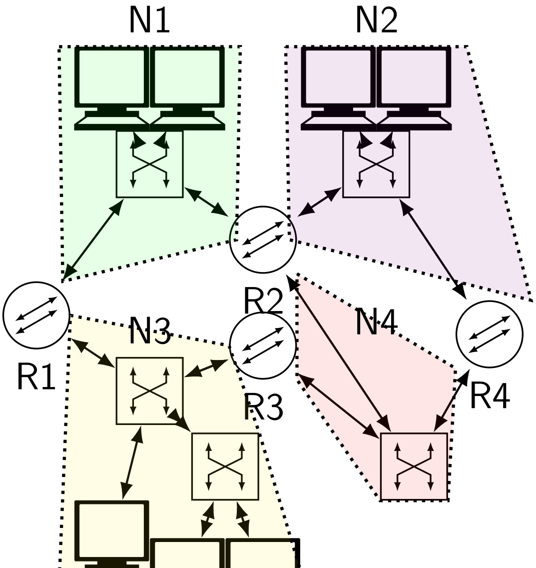
N1	N2	N3	N4
1/R2	1/R2	∞/—	1/R2



N1	N2	N3	N4
2/R1	∞/—	1/R3	1/R3

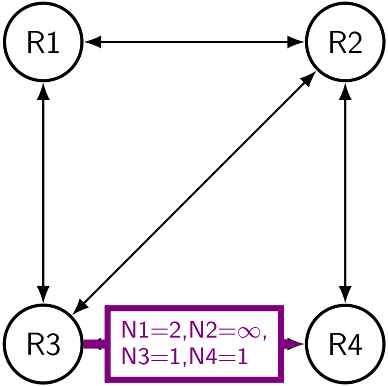
N1	N2	N3	N4
∞/—	1/R4	∞/—	1/R4

networks v routers (DV)



N1	N2	N3	N4
1/R1	∞/—	1/R1	∞/—

N1	N2	N3	N4
1/R2	1/R2	∞/—	1/R2



N1	N2	N3	N4
2/R1	∞/—	1/R3	1/R3

N1	N2	N3	N4
∞/—	1/R4	∞/—	1/R4

distance vector routing

each node keeps *distance vector*

- distance to each other node (network)
- also which neighbor to go through to get that distance

periodically send distance vector to all neighbors

when receiving distance vector from X, check

"would going through X give me a better distance?"

- if so, update distance + which neighbor

Routing Information Protocol

router broadcast on networks it's connected to packet containing list of:

- networks it can reach (example: 1.2.3.0/24)
- its next hop to that network
- its metric (distance) to reach that network

each router on that network processes that packet

on receiving distances, routers see if they can update their routes

- routes will be to networks (1.2.3.0/24, etc.), not routers

local information

routers need to track themselves:

which networks they can reach directly

(which networks is it connected to)

the ‘distance’ it needs to reach those networks

(probably based on its bandwidth to that network?)

RIP — when to update

policy: every approx. 30 seconds always AND

immediately on changes ("triggered")

means that connecting new router should better routes quickly

links going down

problem with our update rule:

assumes routes only get better

reality: sometimes links go down

need to find different route

updating for removal (1)

let's say I'm A and my distance vector is:

A=0 via A, B=4 via B, C=5 via D, D=4 via D

if my link to D goes down, new distance vector should be?

updating for removal (1)

let's say I'm A and my distance vector is:

A=0 via A, B=4 via B, C=5 via D, D=4 via D

if my link to D goes down, new distance vector should be?

A=0 via A, B=4 via B, C=∞ via no one, D=∞ via no one

later updates might fix ∞s

updating for removal (2)

let's say I'm A and my distance vector is:

B=4 via B, C=5 via D, D=4 via D

and D tells me its distance vector is

B=8 via A, C=∞ via no one, D=0 via D

then my (A)'s new distance vector should be?

updating for removal (2)

let's say I'm A and my distance vector is:

B=4 via B, C=5 via D, D=4 via D

and D tells me its distance vector is

B=8 via A, C=∞ via no one, D=0 via D

then my (A)'s new distance vector should be?

B=4 via B, C=∞ via no one, D=4 via D

updating for removal (3)

let's say I'm A and my distance vector is:

B=4 via B, C=5 via D, D=4 via D

and D tells me its distance vector is

B=8 via A, C=5 via A, D=0 via D

then my (A)'s new distance vector should be?

updating for removal (3)

let's say I'm A and my distance vector is:

B=4 via B, C=5 via D, D=4 via D

and D tells me its distance vector is

B=8 via A, C=5 via A, D=0 via D

then my (A)'s new distance vector should be?

B=4 via B, C=∞ via no one, D=4 via D

updating for removal (4)

let's say I'm A and my distance vector is:

B=4 via B, C=5 via D, D=4 via D

and D tells me its distance vector is

B=3 via B, C=8 via B, D=0 via D

then my (A)'s new distance vector should be?

updating for removal (4)

let's say I'm A and my distance vector is:

B=4 via B, C=5 via D, D=4 via D

and D tells me its distance vector is

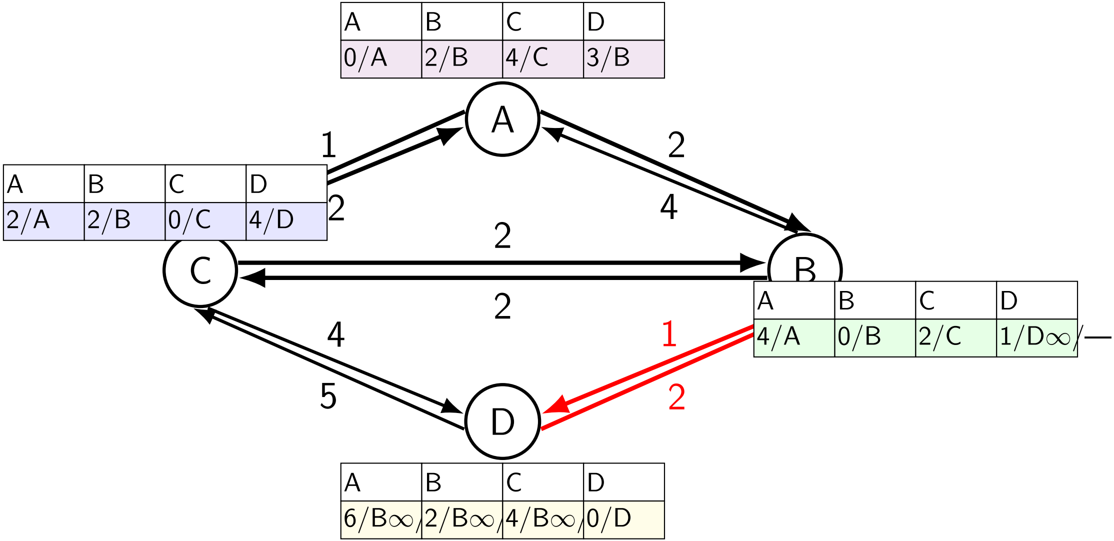
B=3 via B, C=8 via B, D=0 via D

then my (A)'s new distance vector should be?

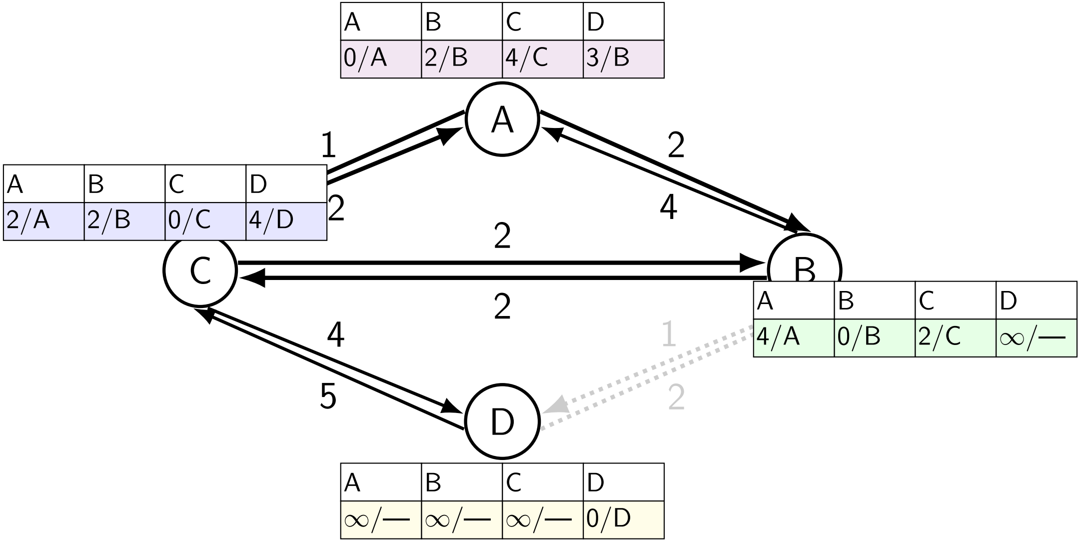
B=4 via B, C=12 via D, D=4 via D

probably later update from B will overwrite route to C

removal?


A B C D
0/A 2/B 4/C 3/B
A B C D
2/A 2/B 0/C 4/D
A B C D
4/A 0/B 2/C 1/D∞/—
A B C D
6/B∞/ 2/B∞/ 4/B∞/ 0/D
A
B
C
D
1
2
2
4
2
2
4
5
1
2

removal?


A
B
C
D
A B C D
0/A 2/B 4/C 3/B
A B C D
2/A 2/B 0/C 4/D
A B C D
4/A 0/B 2/C ∞/—
A B C D
∞/— ∞/— ∞/— 0/D
1
2
2
4
2
2
4
5
1
2

split horizon with poison reverse

when sending distance vectors, ‘posion’ routes to same node

- make sure other node won’t go back to us…
- only to have us go back to them

example, if I’m A and routes are:

- A: 0 via A; B: 2 via B; C: 4 via C; D: 6 via B

when sending to B send:

- A: 0; B: 2; C: 4; D: ∞

without split horizon?

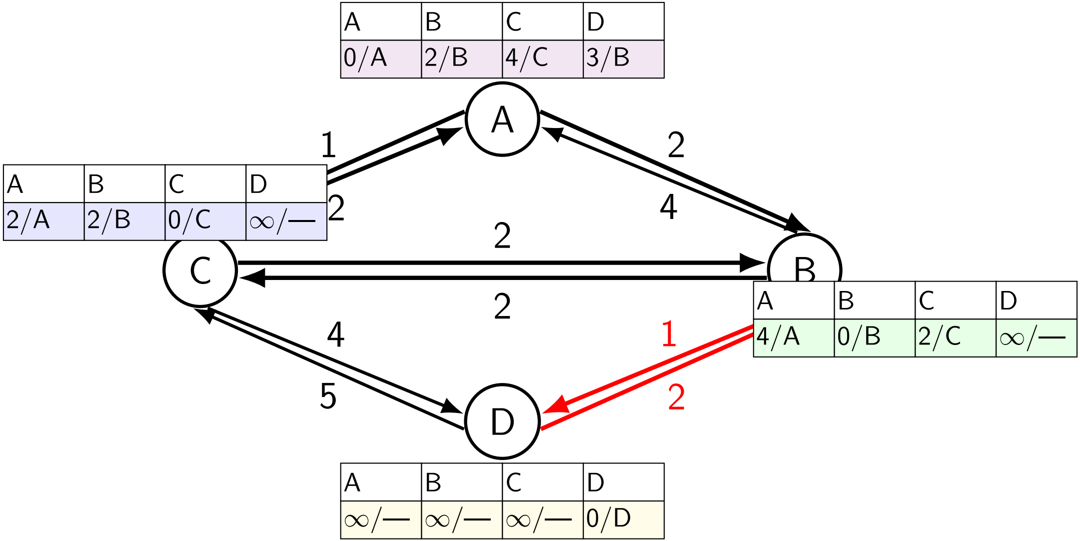
can create routing loop

example: if D unreachable from B, then B goes to A and A goes to B

called "count-to-infinity" problem

- because A, B will keep updating distance higher and higher

avoided count-to-infinity



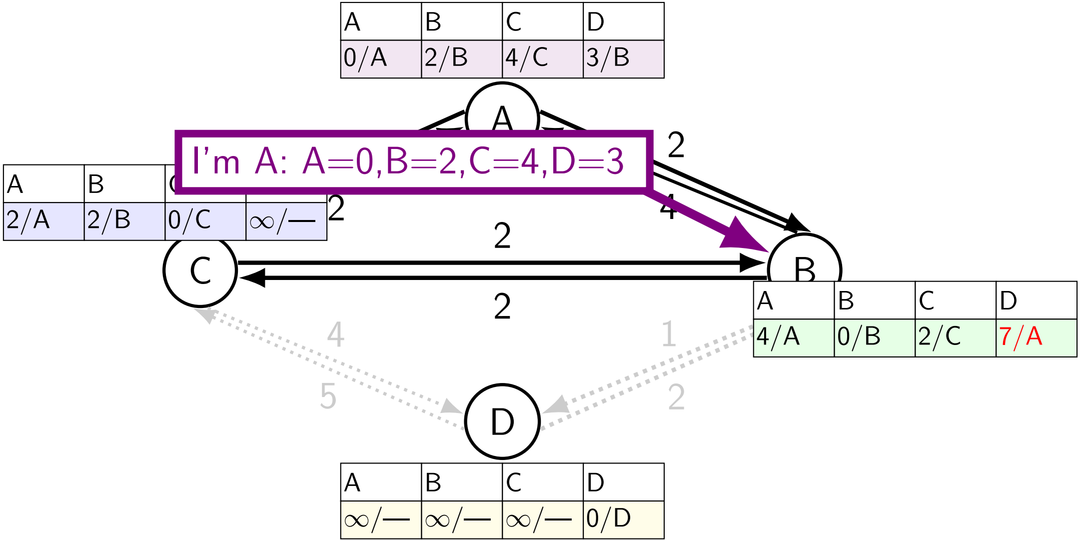
A	B	C	D
0/A	2/B	4/C	3/B

A	B	C	D
2/A	2/B	0/C	∞/—

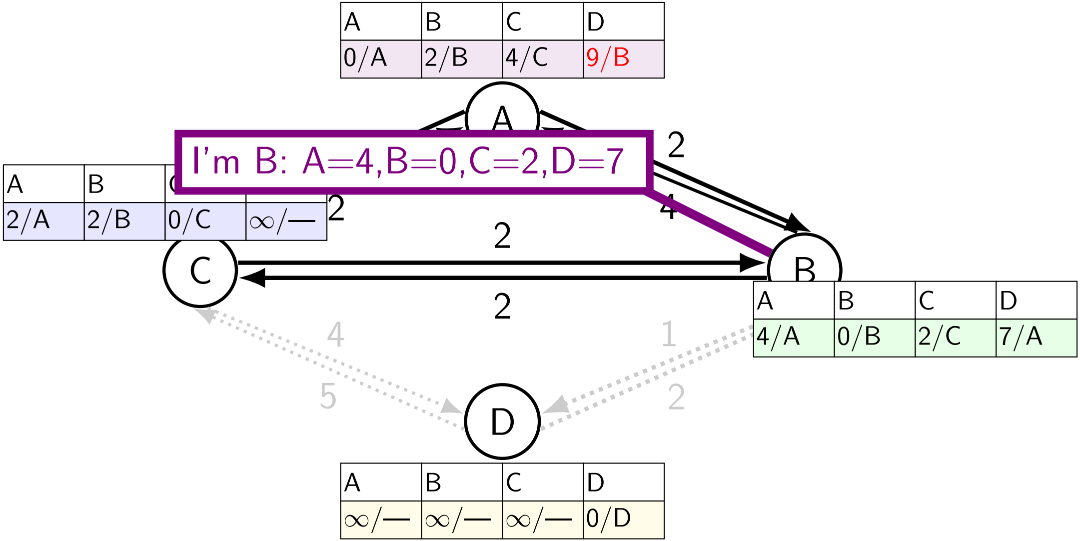
A	B	C	D
4/A	0/B	2/C	∞/—

A	B	C	D
∞/—	∞/—	∞/—	0/D

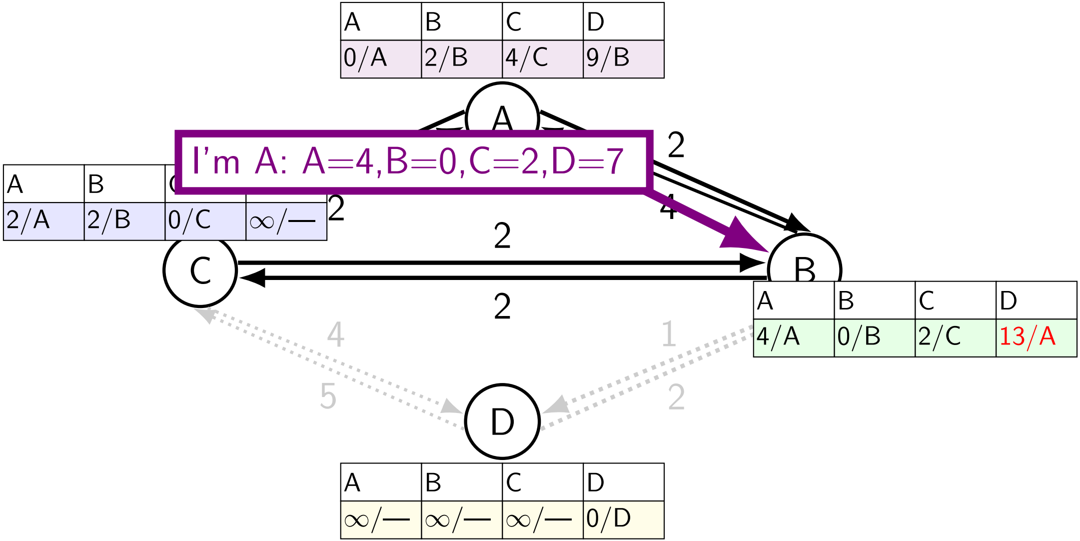
avoided count-to-infinity



avoided count-to-infinity



avoided count-to-infinity



trivial loop

oops: A to B to A to B to A to B to …

this case: relatively easy to avoid

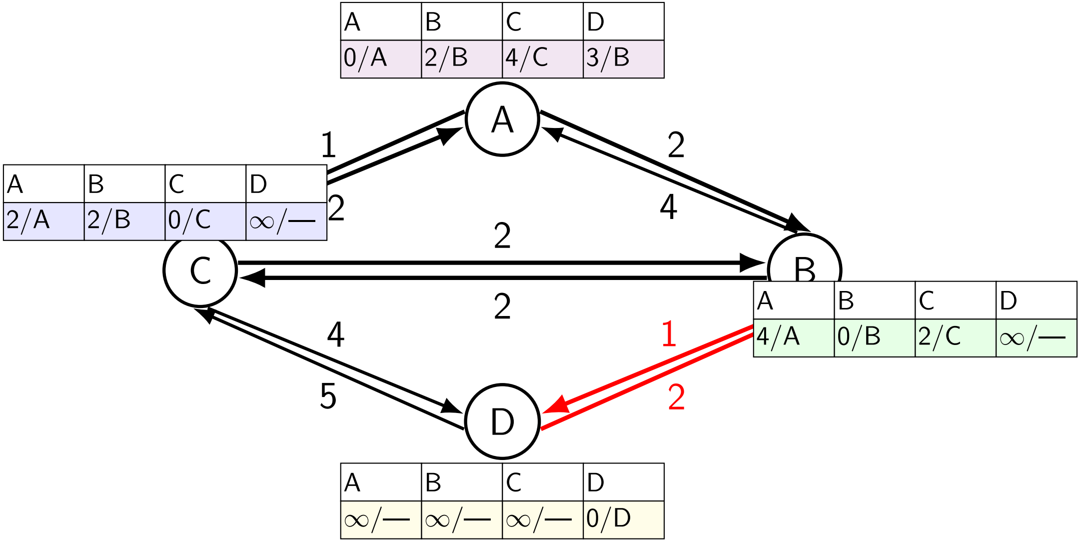
split horizon incomplete solution

split horizon prevents trivial loops, but…

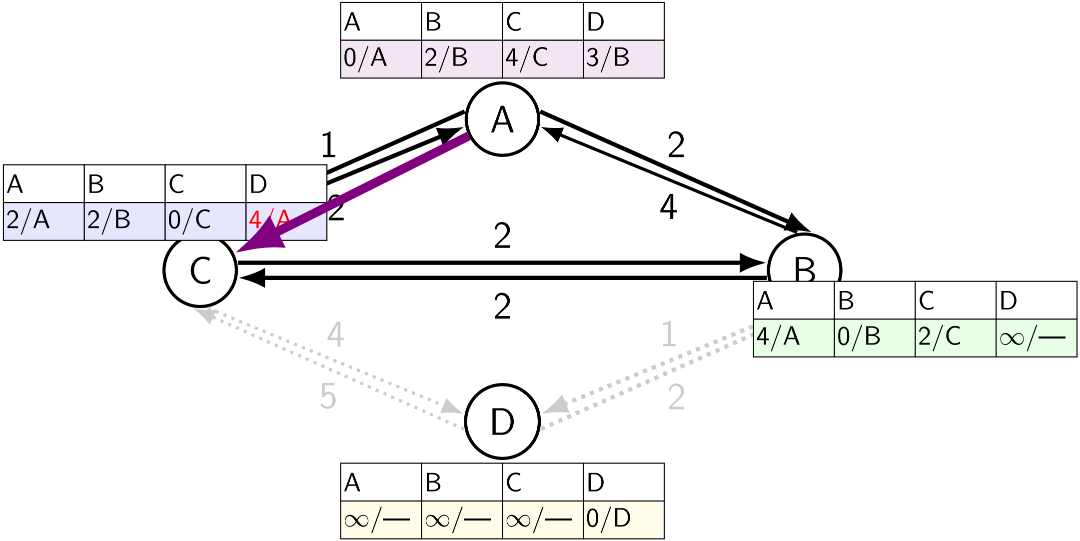
doesn't actually solve the count-to-infinty problem

in well-connected network, there will be longer loops

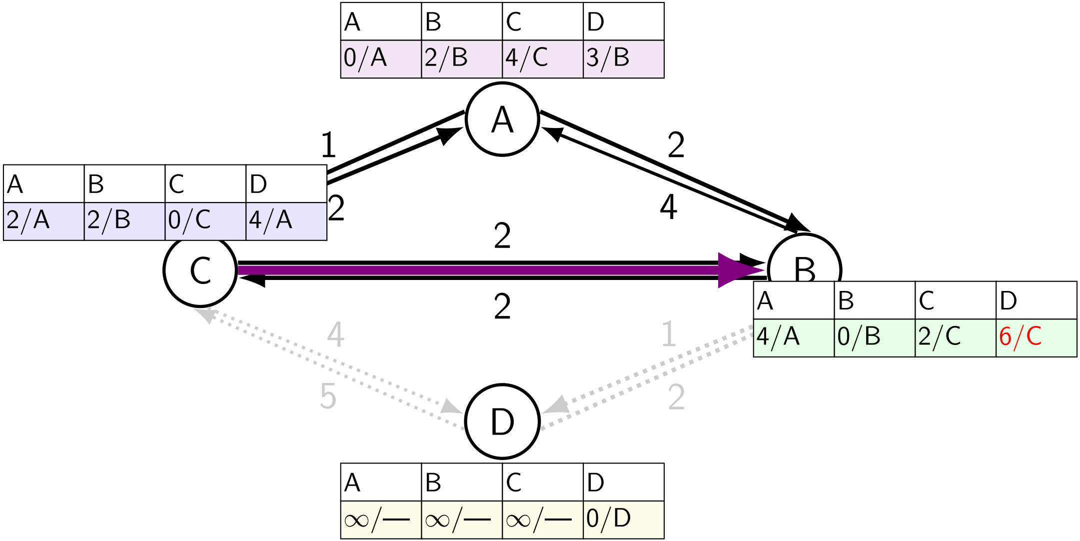
count-to-infinity (v2)



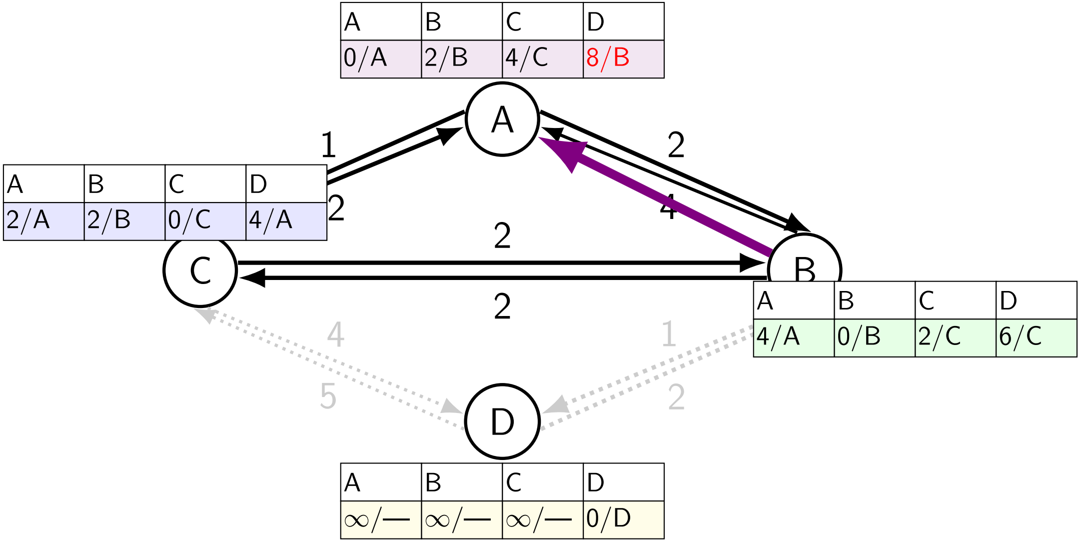
count-to-infinity (v2)



count-to-infinity (v2)



count-to-infinity (v2)



count-to-infinity

when node becomes unreachable, can have ‘phantom’ routes

keep propogating in loop, incrementing metric forever

RIP solution: maximum metric is 15 (hops)

better count-to-infinity solutions?

can share information about more than just neighbors

we'll see two examples:

link-state routing protocols (example: OSPF)

- every router learns full map of network

border gateway protocol (BGP)

- (basically) track *list of hops* alongside distances
- eliminate potential routes that would create duplicate hops (loops)

link-state routing

will keep idea of sharing state with neighbors...

but weren't sharing enough state!

other routing idea:

routers collect *complete map* of network

example protocol for this: OSPF

Open Shortest Path First

OSPF link-state advertisements (router)

<table>
<tr><td colspan="2">age</td><td>options</td><td>type</td></tr>
<tr><td colspan="4">ID</td></tr>
<tr><td colspan="4">advertising router</td></tr>
<tr><td colspan="4">sequence number</td></tr>
<tr><td colspan="2">checksum</td><td colspan="2">length</td></tr>
<tr><td colspan="4">depends on LSA type</td></tr>
</table>

OSPF LSA sequences/ages

sequence number for getting correction version of LSAs

- some tricky rules to handle routers restarting (losing track of sequence number) and sequence number wraparound

maximum 'age' for link-state advertisements

- typically minutes
- too-old LSAs not used for routing
- deliberately setting age = MaxAge used to invalidate LSAs

OSPF LSA types

‘router’:

list of links for router
links = connect to other router or network
links refer to ID numbers of network/router LSAs
metrics for each link

‘network’:

list of routers for network
different version of external and internal networks

(later) ‘summary’:

part of support for *areas*
used when sysadmin doesn’t want all routers processing whole network map

link-state database

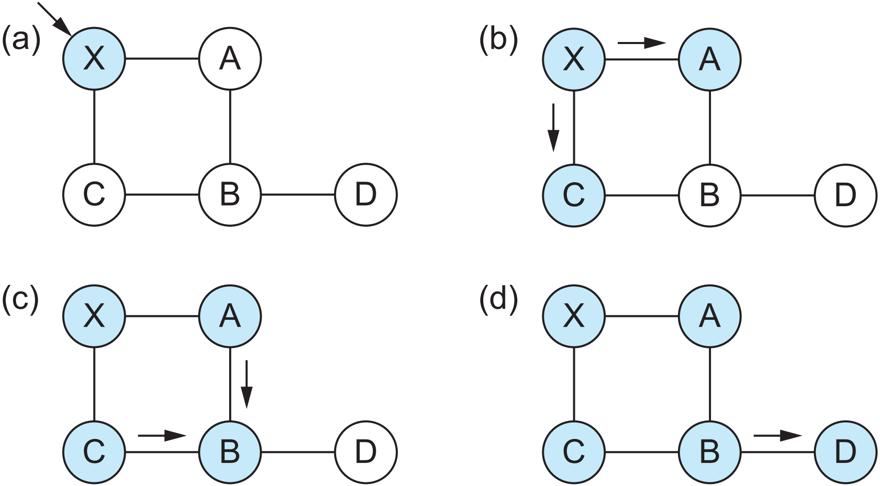
whole collection of advertisements

every router, network, link between router+network

all the metrics for those

reliable flooding (picture)

Peterson and Davie, *Computer Networks: A Systems Approach*, Figure 88



missing from the picture

in picture: seems like each router directly connected to each other

often we have multiple routers connected to local network

can/will share link state packets by broadcasting on local network

reliable flooding in OSPF — setup

for each subnetwork:

> choose a designated and backup router
> make sure backup becomes designated on failure

designated router will take care of propogating updates to everyone on network

…including waiting for acknowledgments, etc.

reliable flooding in OSPF

then, when receiving/generating link state packet:

send to every designed+backup router of subnetwork that

- you are connected to, and
- you are not designated/backup router for, and
- you did not receive the packet from

send to every router on every subnetwork that

- you are designated router for

send = send + resend if no ACK

finding shortest paths

given full picture of network

want to find all shortest paths from self

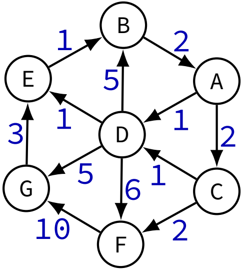
shortest 'distance' = lowest sum of metric

only need next hop, but will compute whole path to find that

assumption: everyone using shortest path

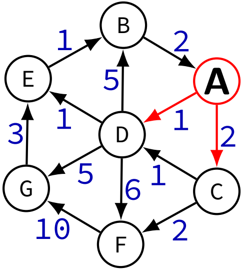
usual solution: Dijkstra's algorithm

Dijkstra's algorithm example 1



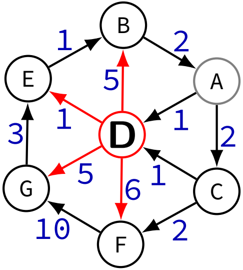
	dist	prev	path
A	0	—	A
B	∞	—	—
C	∞	—	—
D	∞	—	—
E	∞	—	—
F	∞	—	—
G	∞	—	—

Dijkstra's algorithm example 1



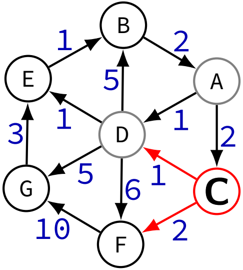
	dist	prev	path
A	0	—	A
B	∞	—	—
C	2	A	A→C
D	1	A	A→D
E	∞	—	—
F	∞	—	—
G	∞	—	—

Dijkstra's algorithm example 1



	dist	prev	path
A	0	—	A
B	6	D	A→D→B
C	2	A	A→C
D	1	A	A→D
E	2	D	A→D→E
F	7	D	A→D→F
G	6	D	A→D→G

Dijkstra's algorithm example 1

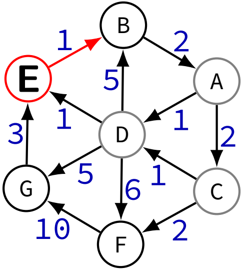


D is adjacent —
but not a shorter path

	dist	prev	path
A	0	—	A
B	6	D	A→D→B
C	2	A	A→C
D	1	A	A→D
E	2	D	A→D→E
F	4	C	A→C→F
G	6	D	A→D→G

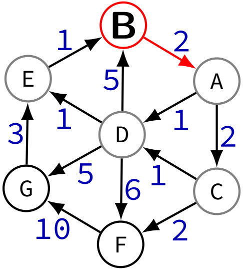
F updated from distance 7 (via D)

Dijkstra's algorithm example 1



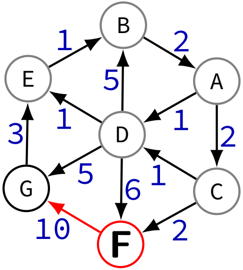
	dist	prev	path
A	0	—	A
B	3	E	A→D→E→B
C	2	A	A→C
D	1	A	A→D
E	2	D	A→D→E
F	4	C	A→C→F
G	6	D	A→D→G

Dijkstra's algorithm example 1



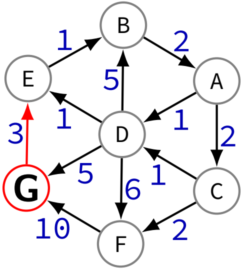
	dist	prev	path
A	0	—	A
B	3	E	A→D→E→B
C	2	A	A→C
D	1	A	A→D
E	2	D	A→D→E
F	4	C	A→C→F
G	6	D	A→D→G

Dijkstra's algorithm example 1



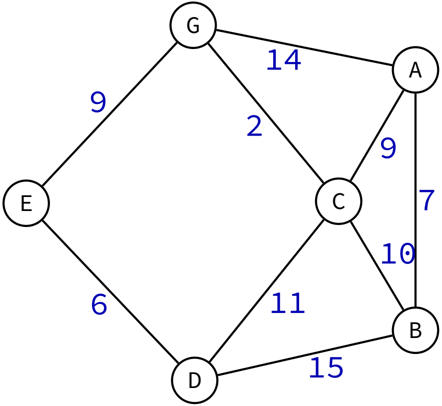
	dist	prev	path
A	0	—	A
B	3	E	A→D→E→B
C	2	A	A→C
D	1	A	A→D
E	2	D	A→D→E
F	4	C	A→C→F
G	6	D	A→D→G

Dijkstra's algorithm example 1



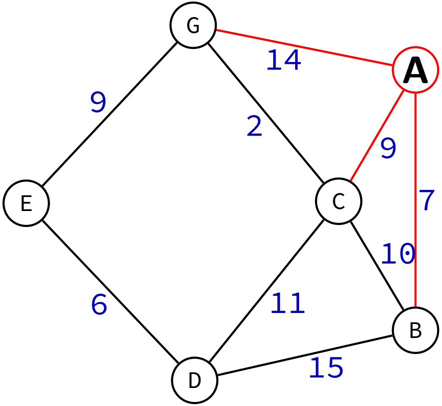
	dist	prev	path
A	0	—	A
B	3	E	A→D→E→B
C	2	A	A→C
D	1	A	A→D
E	2	D	A→D→E
F	4	C	A→C→F
G	6	D	A→D→G

Dijkstra's algorithm example 2



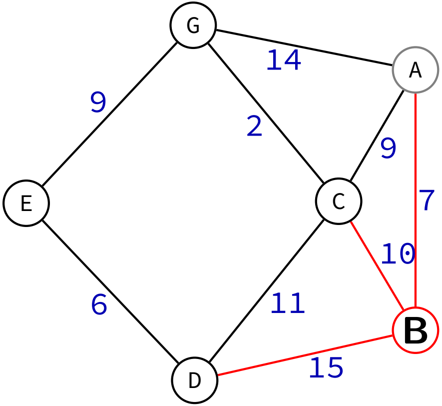
	dist	prev	path
A	0	—	A
B	∞	—	—
C	∞	—	—
D	∞	—	—
E	∞	—	—
G	∞	—	—

Dijkstra's algorithm example 2



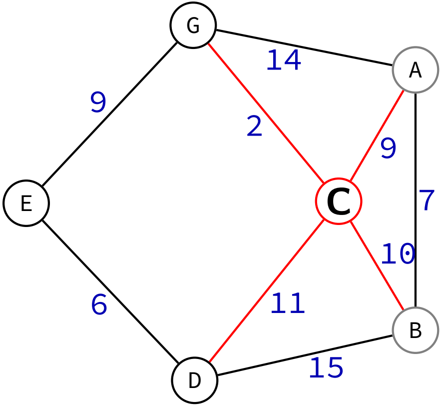
	dist	prev	path
A	0	—	A
B	7	A	A→B
C	9	A	A→C
D	∞	—	—
E	∞	—	—
G	14	A	A→G

Dijkstra's algorithm example 2



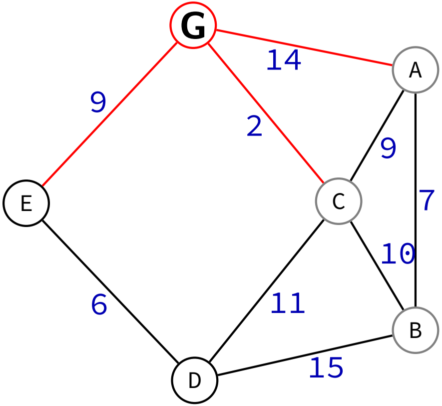
	dist	prev	path
A	0	—	A
B	7	A	A→B
C	9	A	A→C
D	22	B	A→B→D
E	∞	—	—
G	14	A	A→G

Dijkstra's algorithm example 2



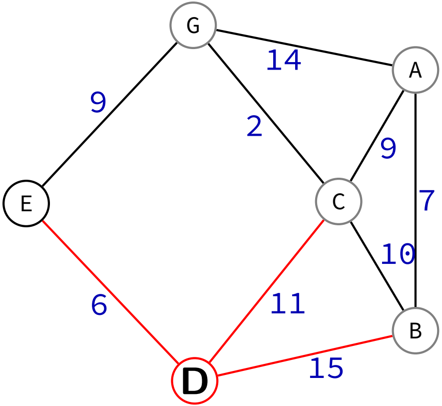
	dist	prev	path
A	0	—	A
B	7	A	A→B
C	9	A	A→C
D	20	C	A→C→D
E	∞	—	—
G	11	C	A→C→G

Dijkstra's algorithm example 2



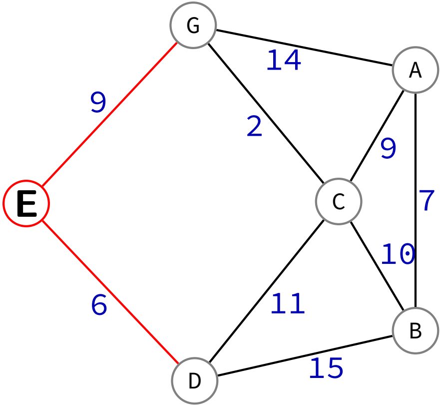
	dist	prev	path
A	0	—	A
B	7	A	A→B
C	9	A	A→C
D	20	C	A→C→D
E	20	G	A→C→G→E
G	11	C	A→C→G

Dijkstra's algorithm example 2



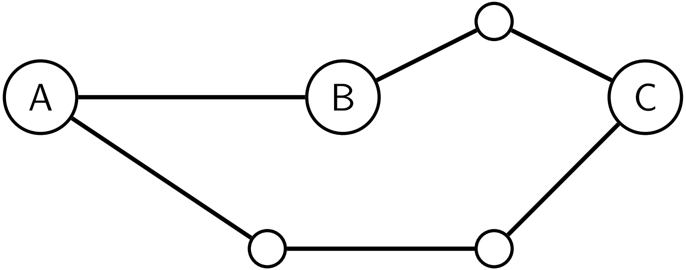
	dist	prev	path
A	0	—	A
B	7	A	A→B
C	9	A	A→C
D	20	C	A→C→D
E	20	G	A→C→G→E
G	11	C	A→C→G

Dijkstra's algorithm example 2



	dist	prev	path
A	0	—	A
B	7	A	A→B
C	9	A	A→C
D	20	C	A→C→D
E	20	G	A→C→G→E
G	11	C	A→C→G

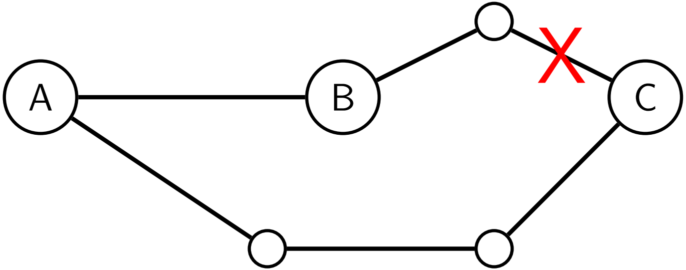
consistency



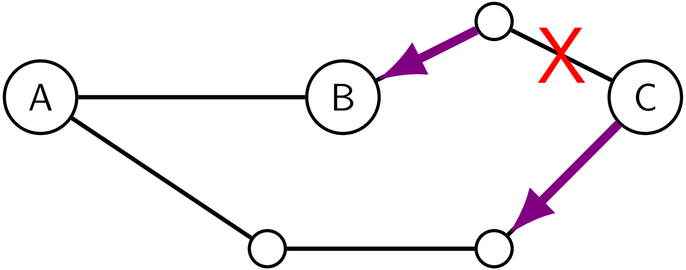
if A sends packet for C to B, how does A know B won't send it back?

hope: if A thought shortest path to C was through B, then B should agree

inconsistency



inconsistency



B thinks best route to C: through A
A thinks best route to C: through B

temporary bad routes

while waiting for link state updates to propogate

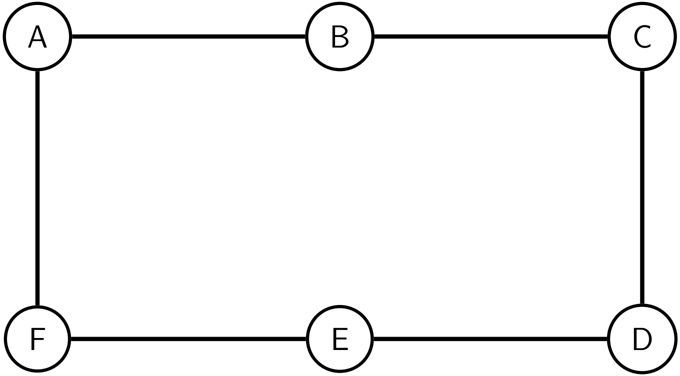
can have too-slow routes

can have routing loops

hope: this is only a few seconds at most

 and routing loop doesn't cause huge explosion of traffic

convergence time

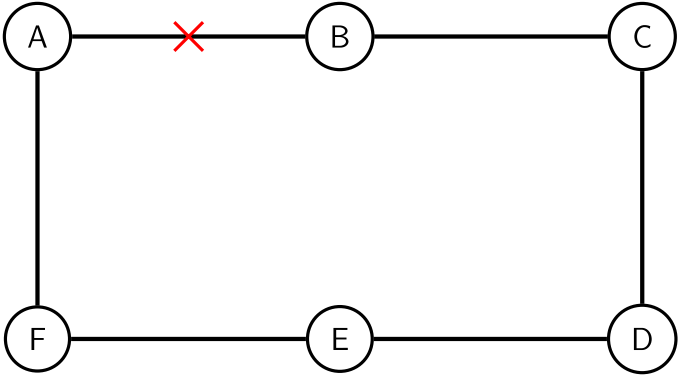


exercise: how many steps to fix A's next hop for B, C, D, etc....

with distance vector?

with link state?

convergence time

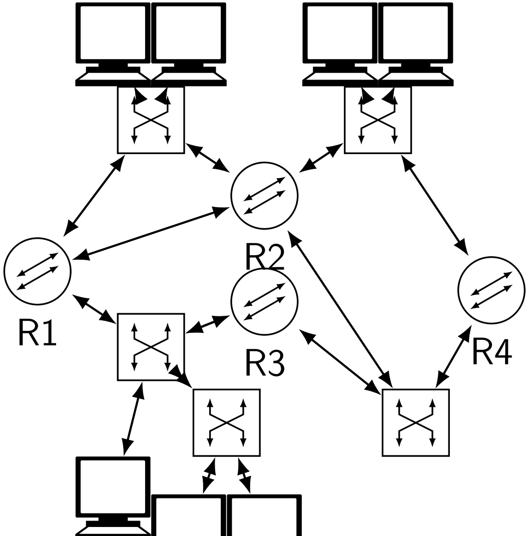


exercise: how many steps to fix A's next hop for B, C, D, etc....

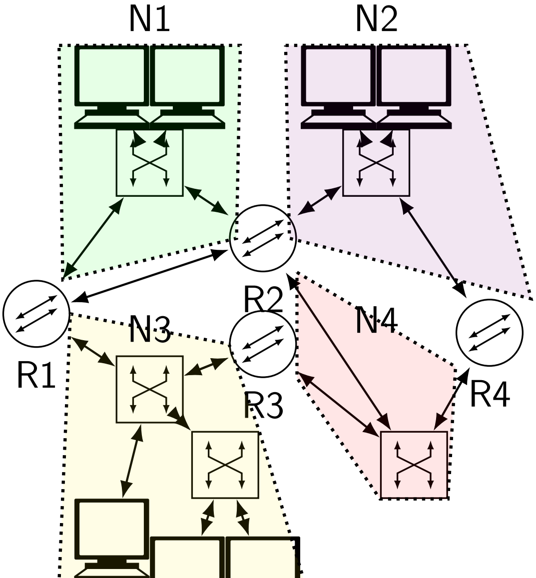
with distance vector?

with link state?

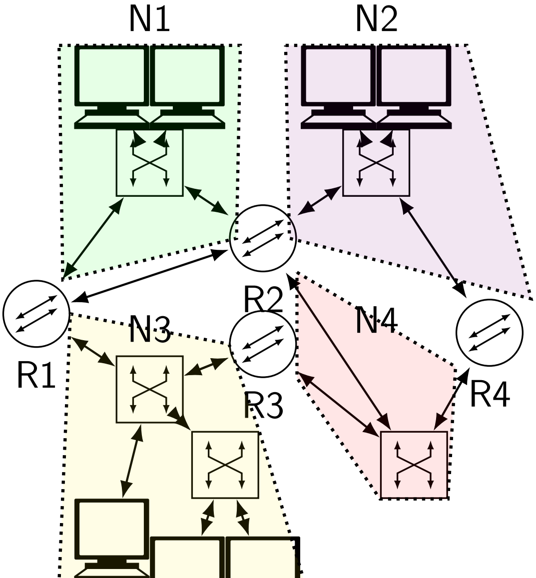
networks v routers (LS)

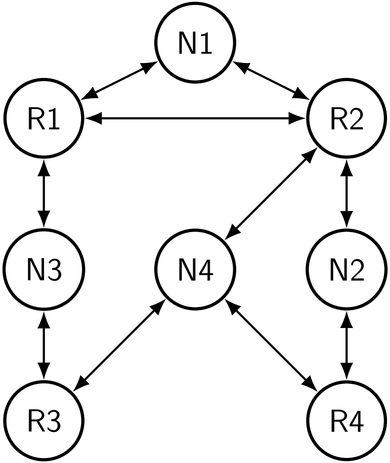


networks v routers (LS)



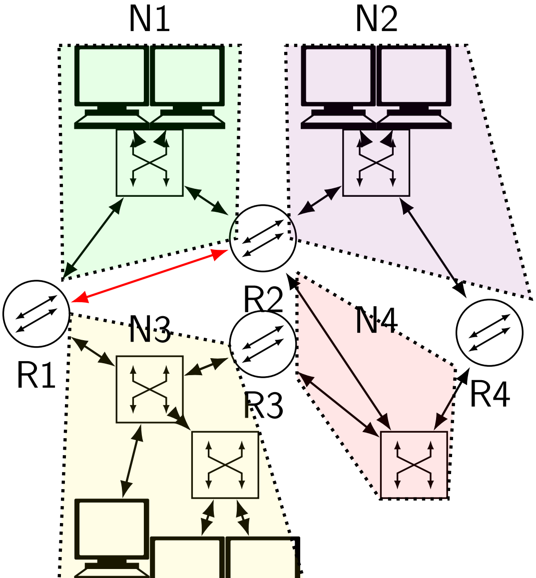
networks v routers (LS)

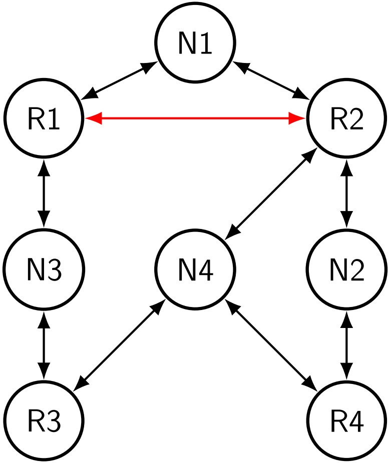




graph has nodes for routers+networks

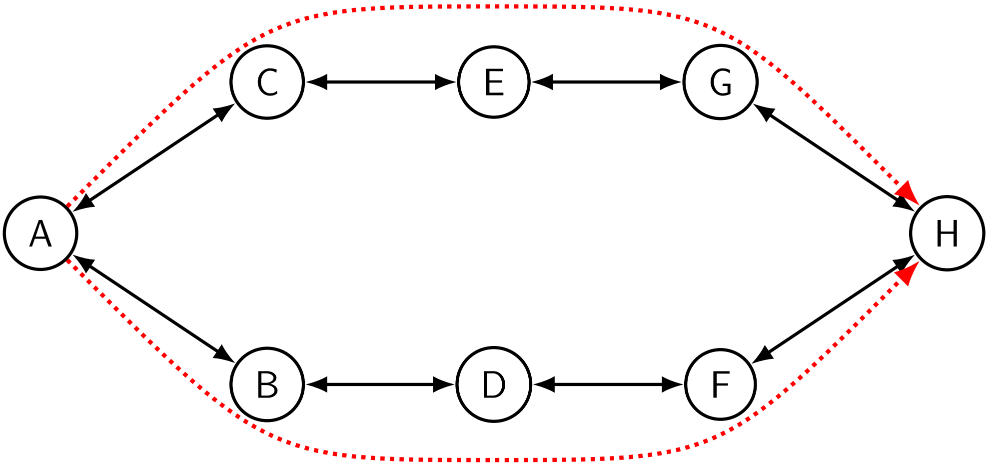
networks v routers (LS)





can have direct router-router link

two good choices?



splitting packets

naïve idea: send every other packet on bottom link

problem: bottom and top link will have different latencies

(even if only temporarily from queuing)

$\implies$ packets will be reordered a lot

this is pretty bad for TCP

(and many other things)

equal cost multipath (ECMP)

split packets by flow

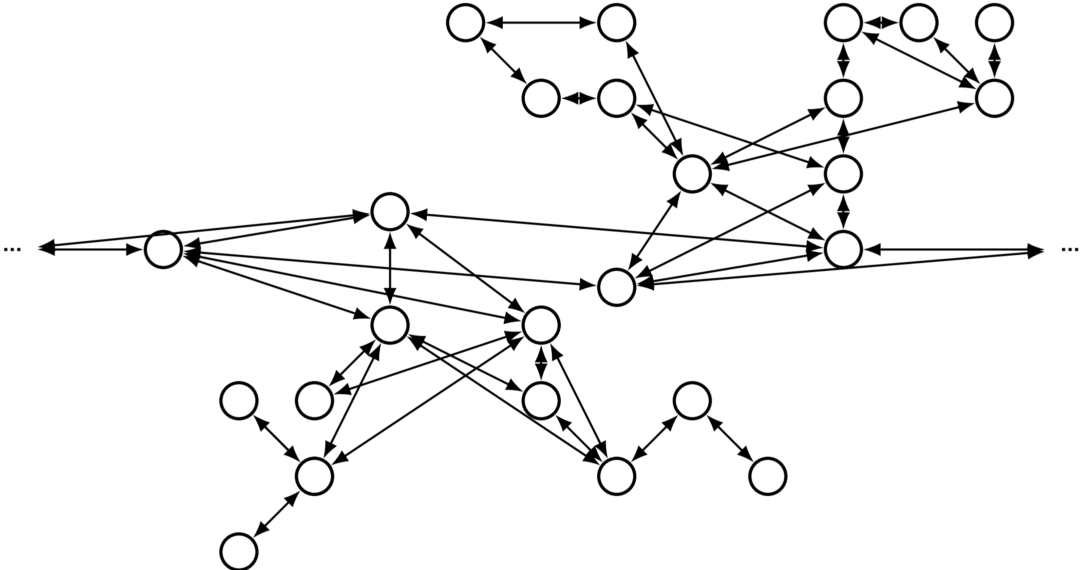
goal:

each TCP connection chooses one of the N links
…but don't want to track list of TCP connections

solution:

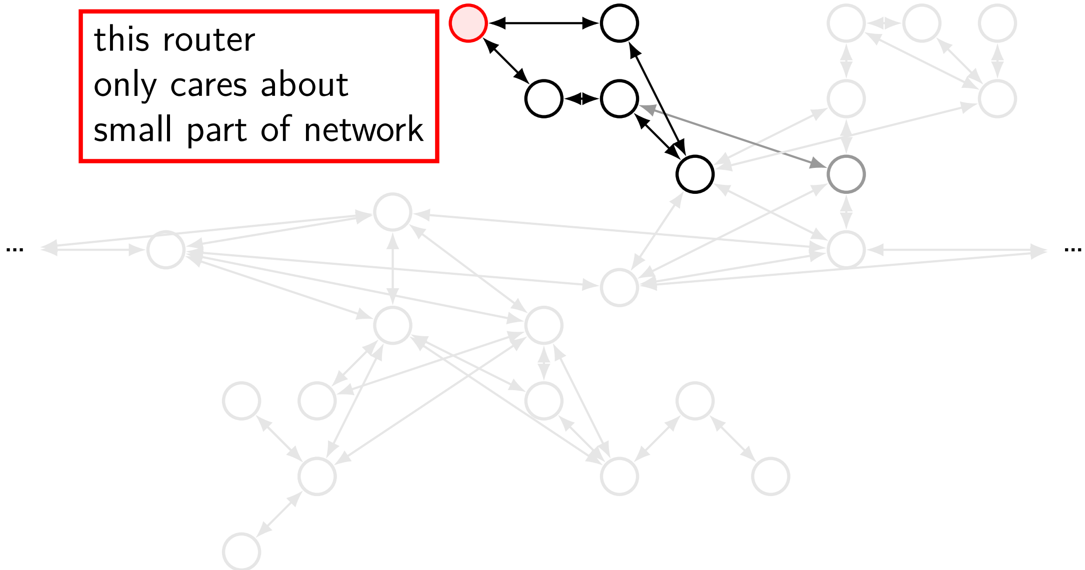
take a hash of the connection info in header
use link index $\left\lfloor \dfrac{\text{hash value} \times N}{\text{max hash value}} \right\rfloor$

a big network



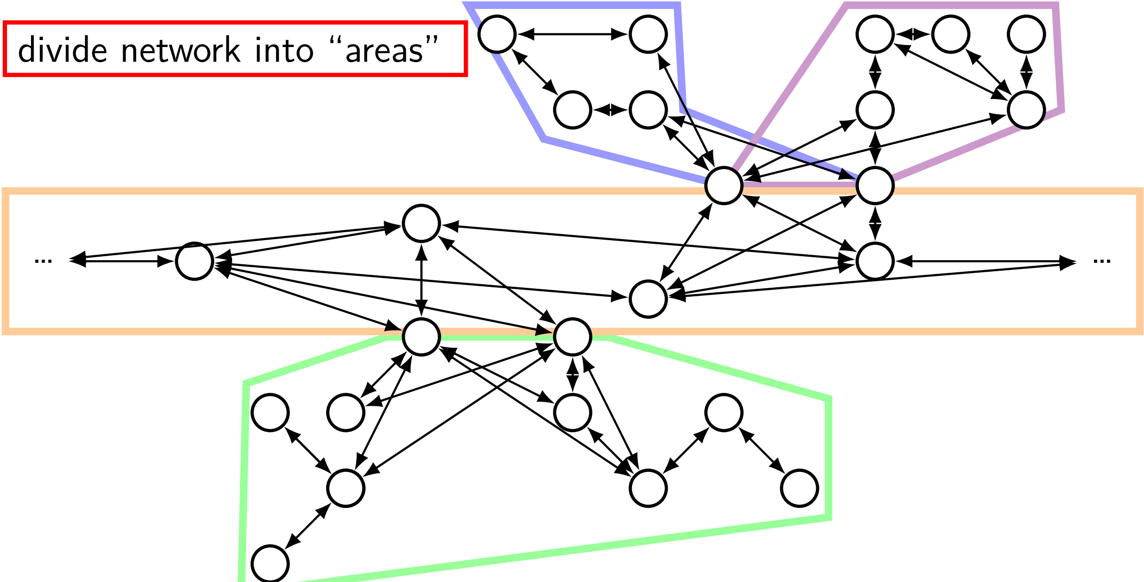
a big network

this router
only cares about
small part of network



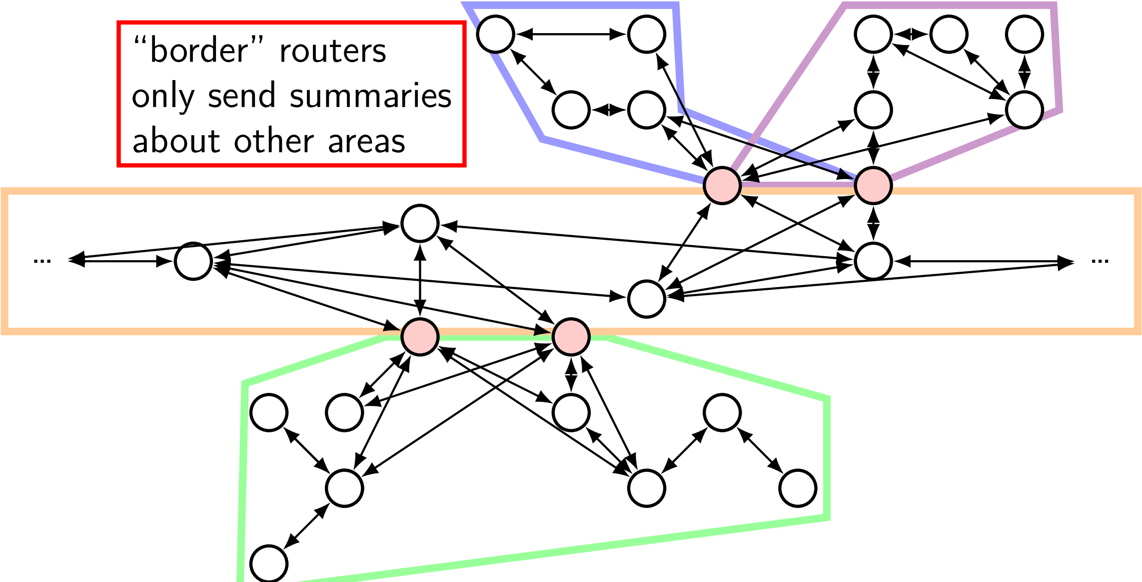
a big network

divide network into “areas”



a big network

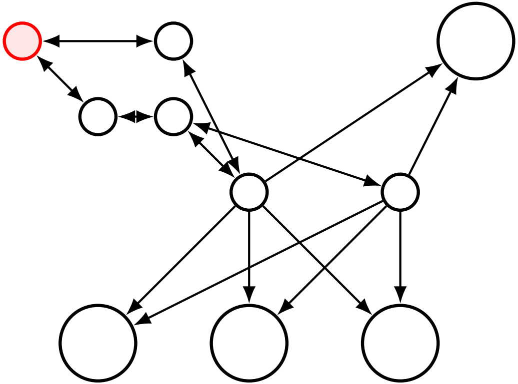
“border” routers
only send summaries
about other areas



a big network

router sees simpler summary of network
→ hopefully faster routing

...



...

distance vector in link state?

summaries are distance vectors!

area border routers just saying which networks + metric

idea: mix simpler distance vectors with more flexible link-state

but distance vector problems?

recall: count-to-infinity

let's say areas A, B, C, D all connected to each other...

...and area D goes offline:

could packet for D loop area A to B to C to A to B to C to ...

but distance vector problems?

recall: count-to-infinity

let's say areas A, B, C, D all connected to each other...

...and area D goes offline:

could packet for D loop area A to B to C to A to B to C to ...

OSPF solution: disallow this network configuration

backbone

OSPF area 0 is called “backbone”

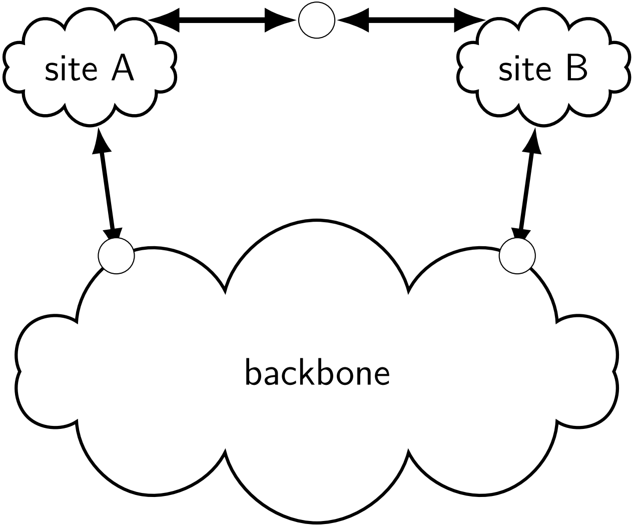
border routers only summarize routes sent to backbone *or* not obtained from other area border routers

means routing between areas must either:

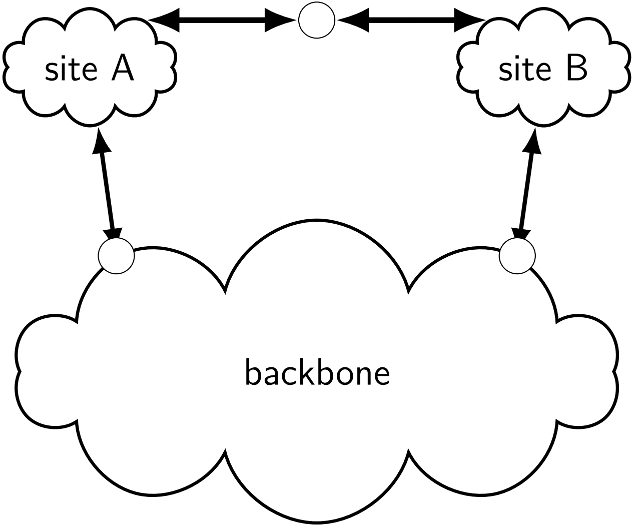
go through the backbone, or
only go through one border router

makes loops not possible

backbone limits

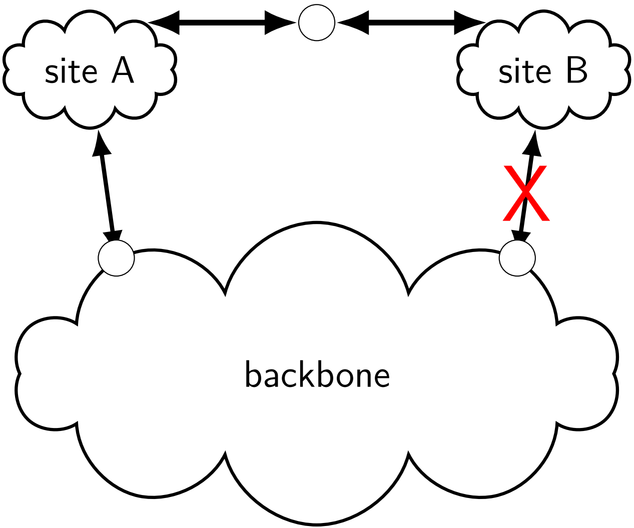


backbone limits



org with two sites —
can configure as areas
three border routers

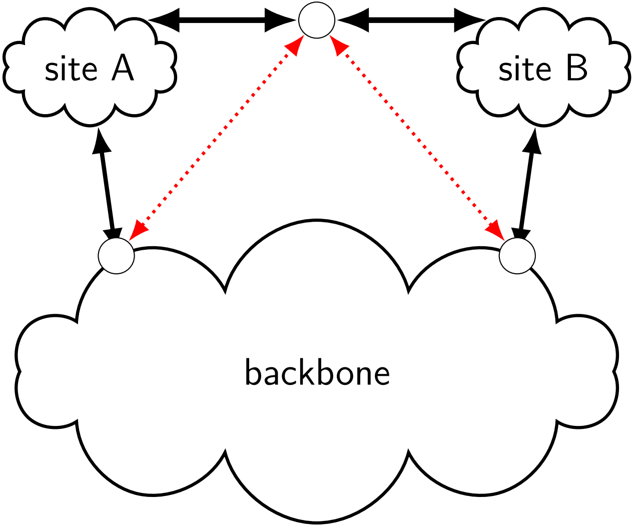
backbone limits



if B's link to backbone fails, B should use building A's

but disallowed by anti-loop rule

backbone limits



could fix this
by connecting A/B
border router
to backbone...
solution:
"virtual links"

OSPF virtual links

“tunnel” backbone through another area

- route as if ‘direct’ connection between two border routers
- but connection implemented by going through area
- both ends considered part of backbone

metric for virtaul link = metric of route through area

configured explicitly by administrator

interdomain routing

so far: routing within one organization

lots of trust/sharing:

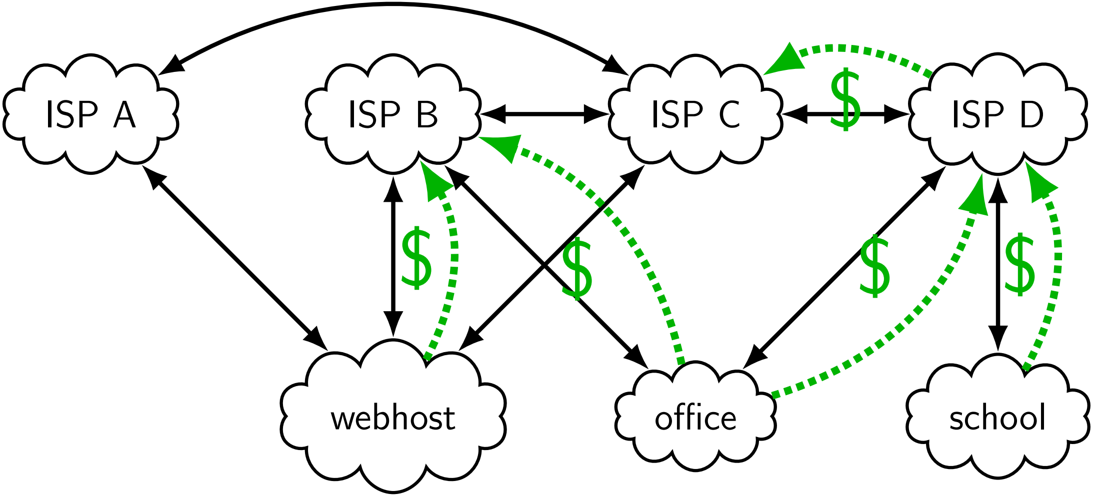
okay to send packets through (essentially) every router

okay for any router to ‘announce’ any address

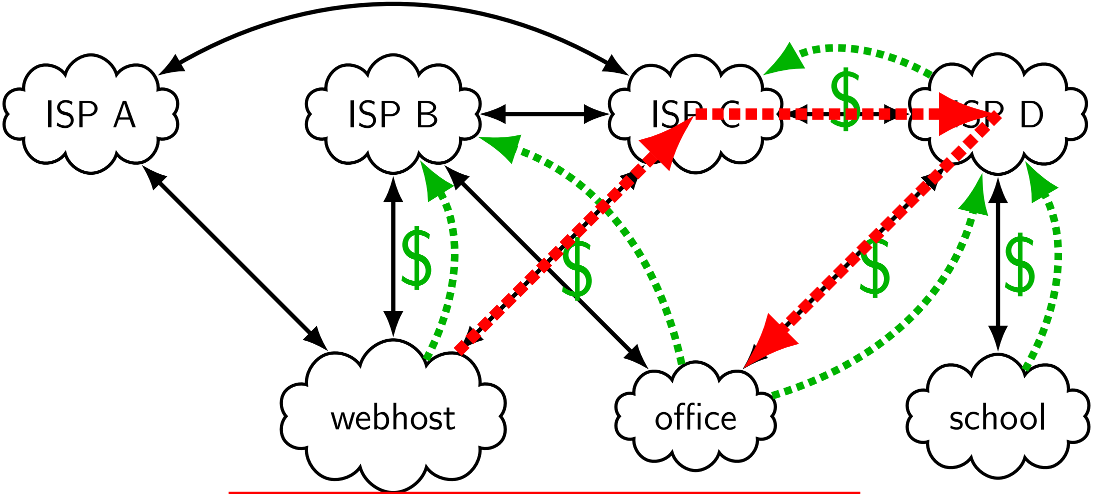
okay to share (almost) full map of network

not what we want for interdomain routing

some business considerations

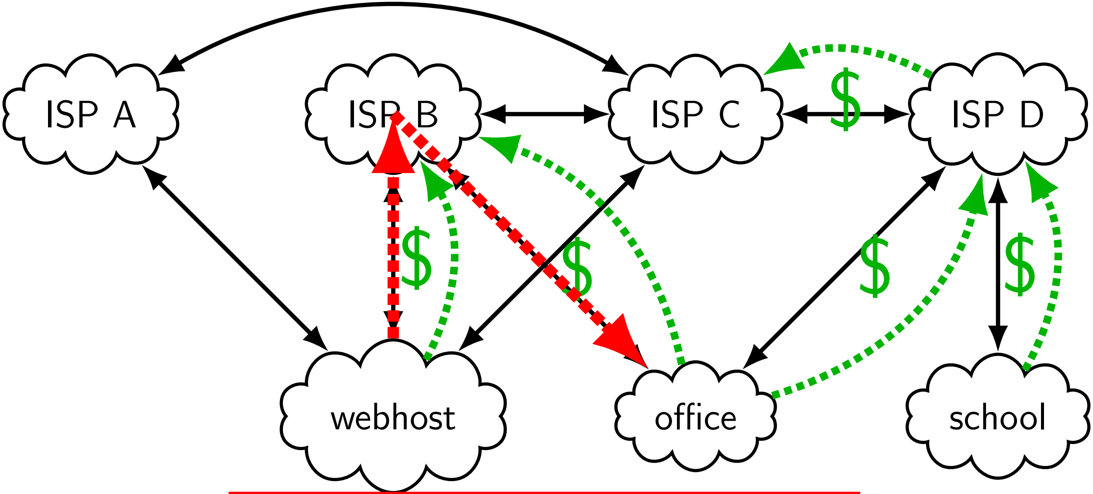


some business considerations



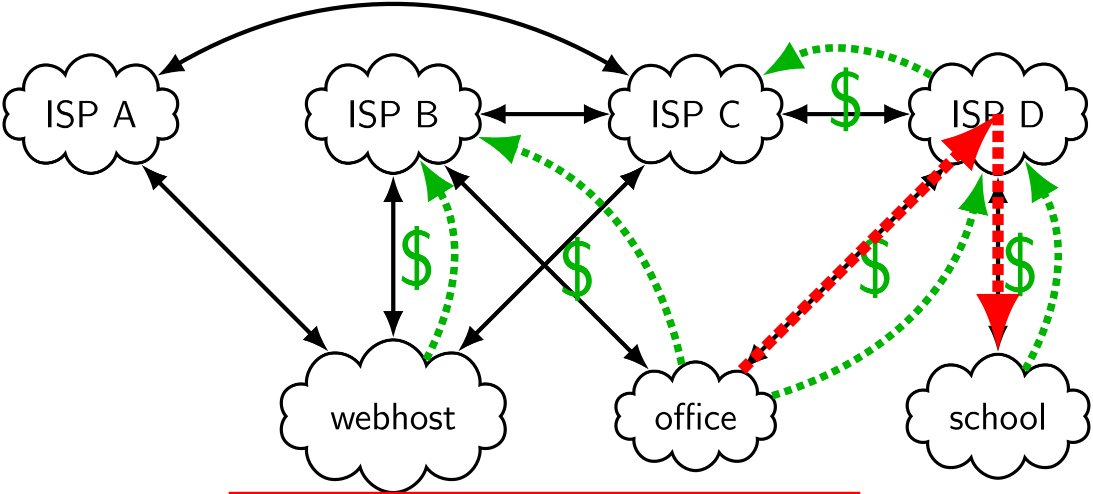
exercise: does this route make sense?

some business considerations



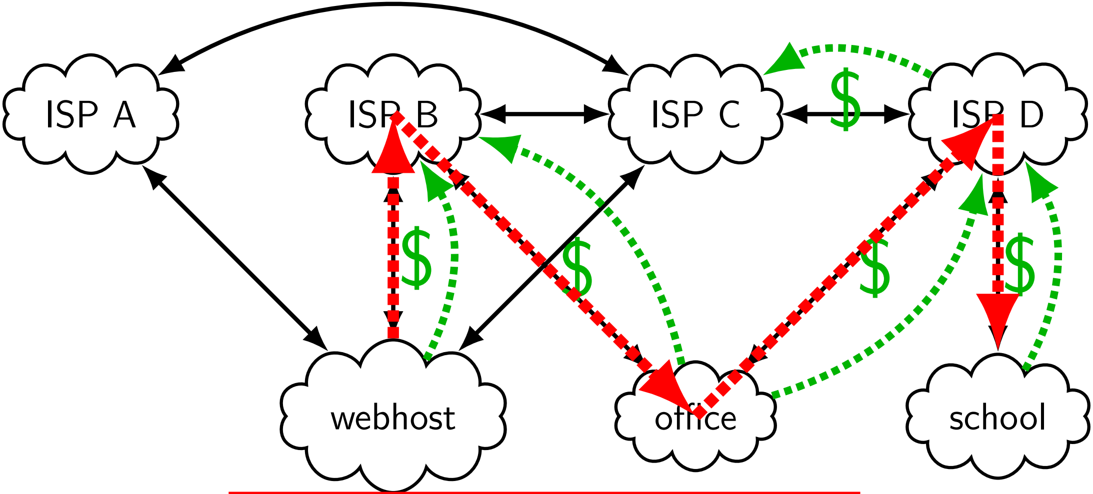
exercise: does this route make sense?

some business considerations



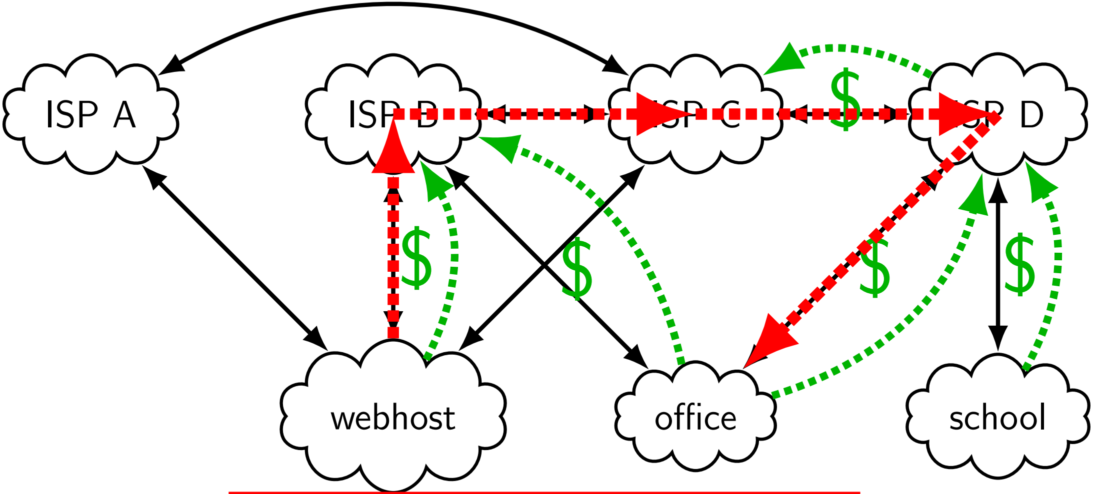
exercise: does this route make sense?

some business considerations



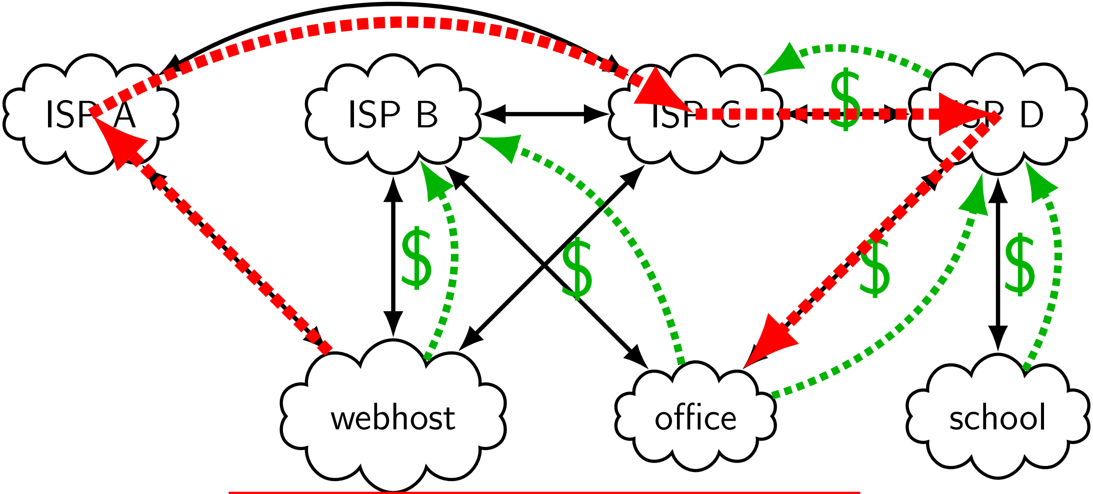
exercise: does this route make sense?

some business considerations



exercise: does this route make sense?

some business considerations



exercise: does this route make sense?

autonomous system

autonomous system (AS) — one "routing domain"

- typically = set of networks administrated by one organization
- decides what routing to use internally
- should be fully connected internally

scope of OSPF instance = one AS

each AS can connect to other ASes

- well-defined protocol for sending routes to other ASes

AS numbers

for Internet routing, ASes are assigned numbers

assigned by IANA and RIRs (similar to IP addresses)

originally 16-bit, now extended to 32-bit

some private use / special AS numbers

relationship types

provider/customer

- typically: customer pays provider
- provider connects customers everywhere it can (customer paid for it)
- customer does **not** provide paths through its network

peer/peer

- often: no payment ('settlement-free')
- if A peers with B...
- A gets connected to B's customers (customers paid B for this)
- A does not get connected to B's other peers (no one paid B for this)
- A does not get connected to B's providers (no one paid B for this)

connecting big networks?

some options:

(which are basically the same as connecting parts of big network)

run a fiber between two buildings

permitting and construction needed

pay for direct access to fiber someone else ran ("dark fiber")

burying one fiber costs similar to burying bundle, so spares

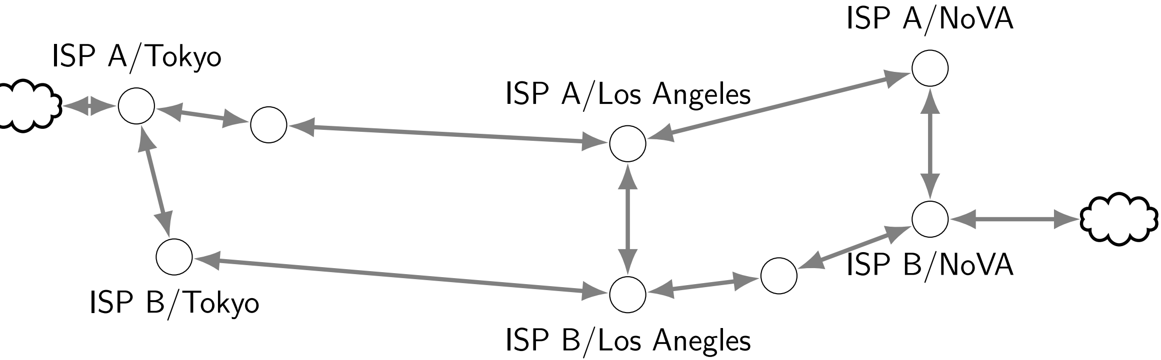
pay a telecom for a site-to-site connection

"gaurenteed" bandwidth+latency between two sites

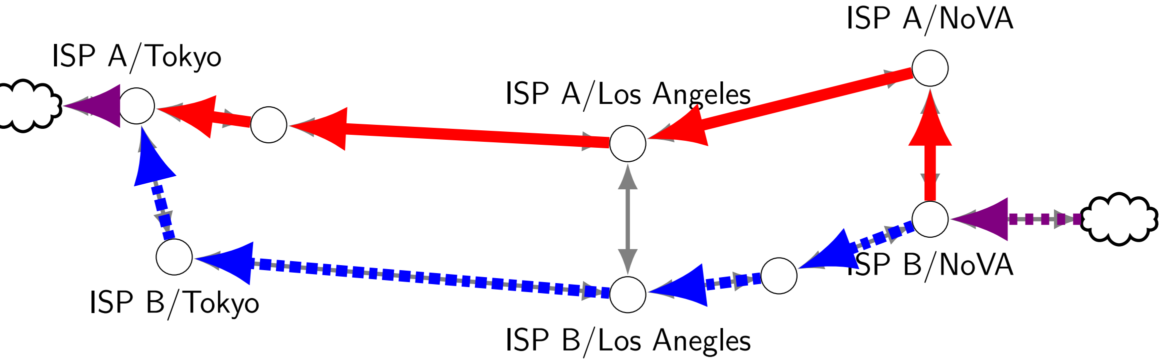
may or may not use series of dedicated fibers

get space in common datacenter, pay datacenter operator for connection

going the distance



going the distance



does ISP A or ISP B help packets cross the Pacific?

distance preferences

ISP B→ISP A across the Pacific:

for ISP B:

- cheaper to hand-off packet to ISP A as soon as possible
- more control over performance if handing off as late as possible

for ISP A:

- cheaper to require ISP B to hand-off packet as late as possible
- more control over performance if B sends as soon as possible

maybe part of ISP A and ISP B peering agreement

Border Gateway Protocol

protocol for sending routes between networks

used whereever routers from different ASes connect

“EBGP”

used within AS to share routes out of AS internally

“IBGP”

each router constructs list of routes to offer

each router receives list of routes, exports to OSPF/etc.

BGP connections

BGP (TCP) connections made between routers

each router keeps track of set routes advertised by other

command sent to add or withdraw specific routes

not like distance vector where we kept resending everything

BGP prefixes

routes sent via BGP called ‘prefixes’

said to be “announced” from one router to another

because the network (e.g. 10.0.1.0/24) is the important part

(and the next hop is implied by where prefix comes from)

BGP route

adjacent routers share list of *routes* with:

IP prefix (CIDR-style, basically)

AS path — list of autonomous system the route goes through

next hop router (IP address)

multi-exit discriminator

low value = this entrance to AS is better than others for these IPs

local preference (internal-only)

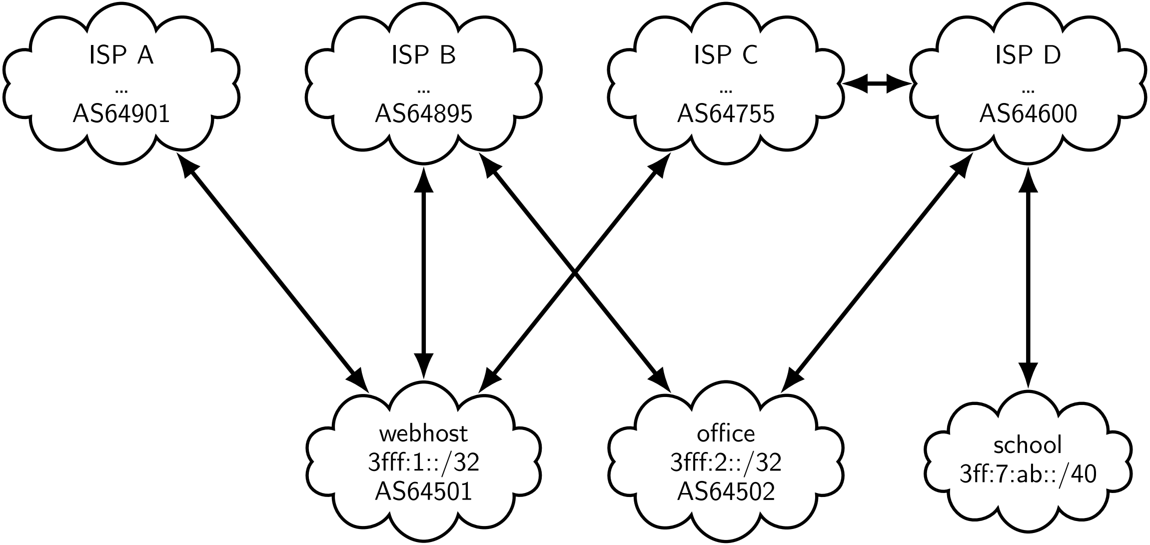
AS path

used to detect routing loops

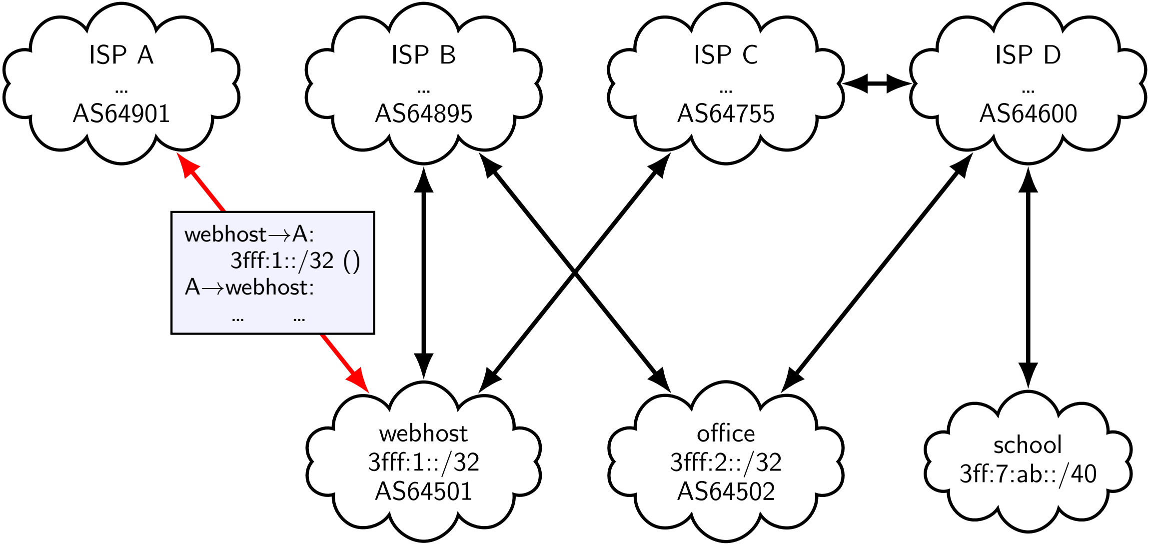
append your AS when sending route externally

always ignore external routes with your AS in their AS path already

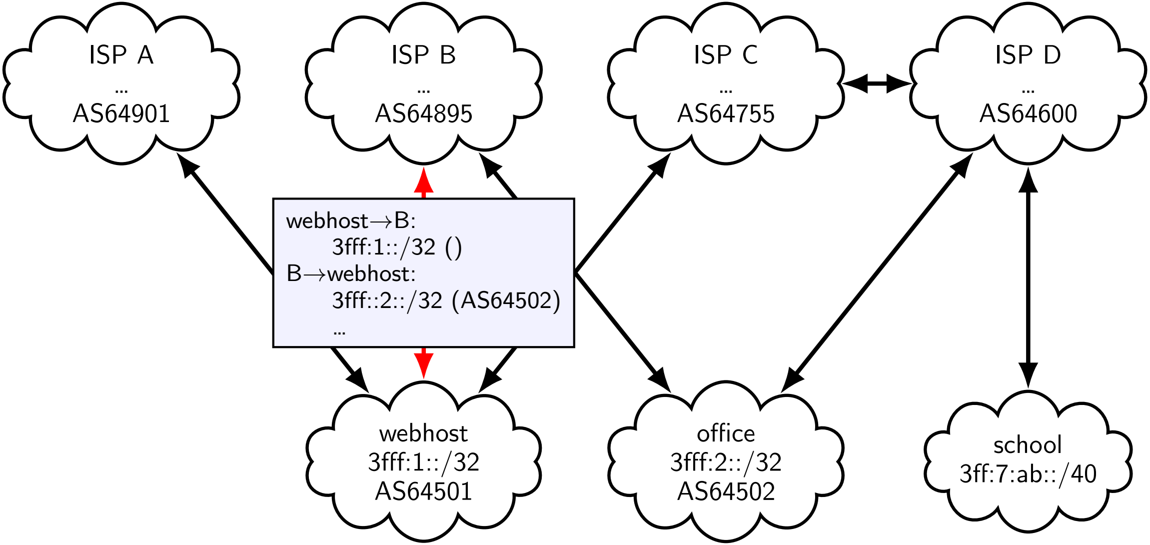
external BGP



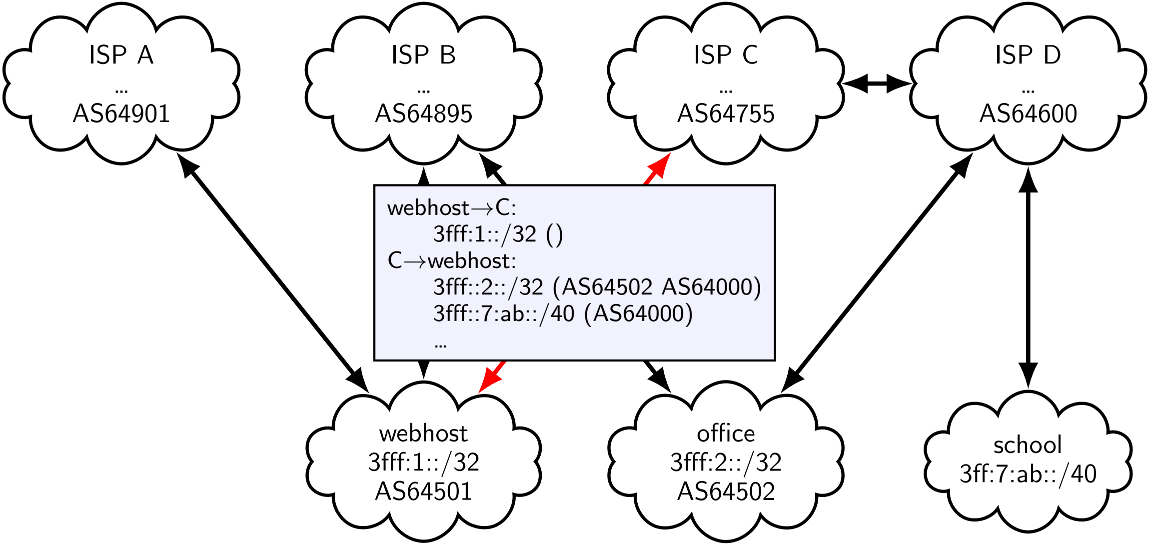
external BGP



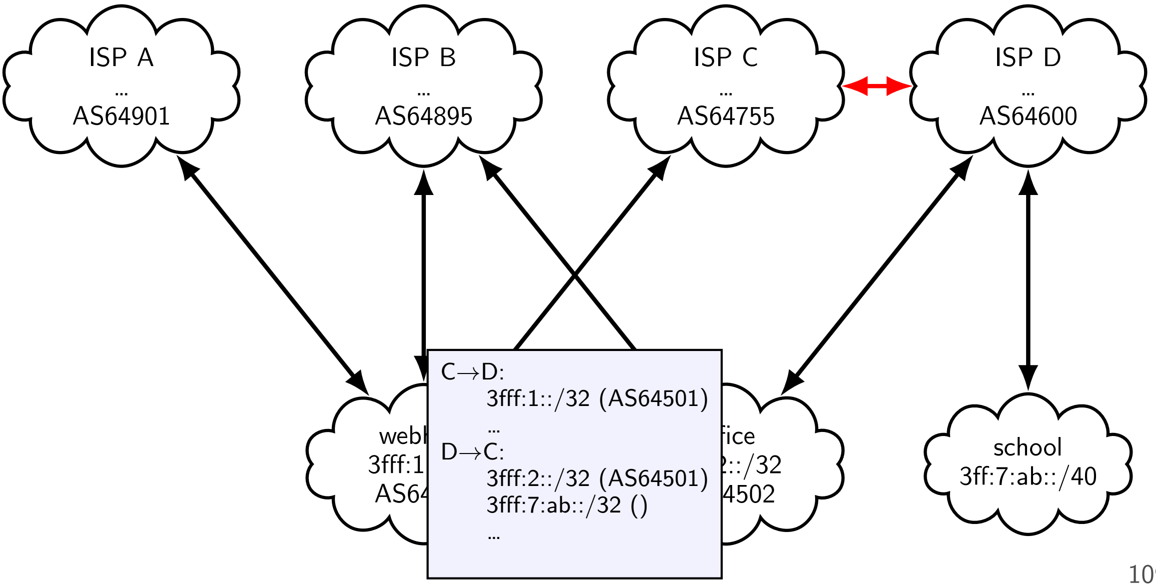
external BGP



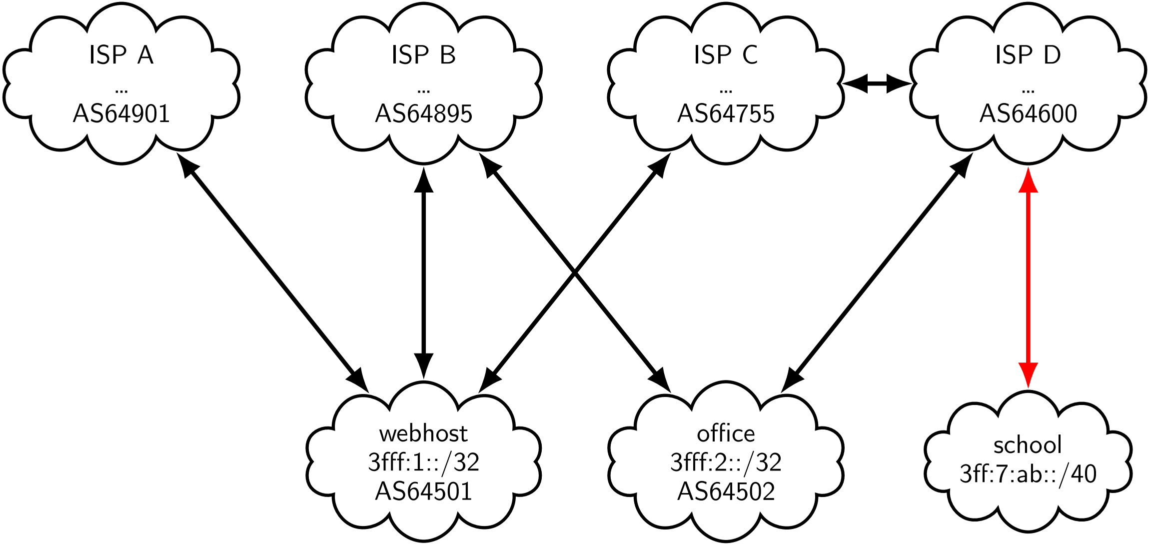
external BGP



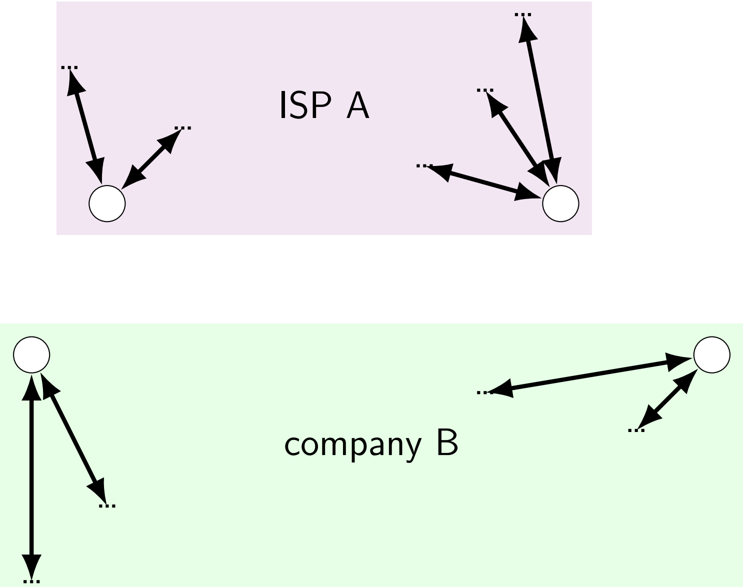
external BGP



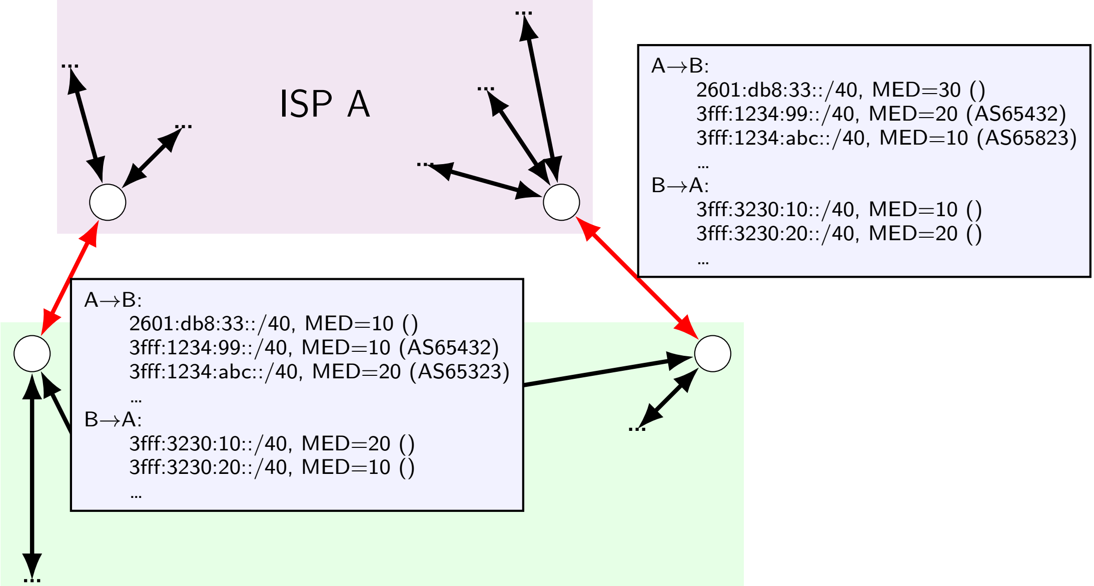
external BGP



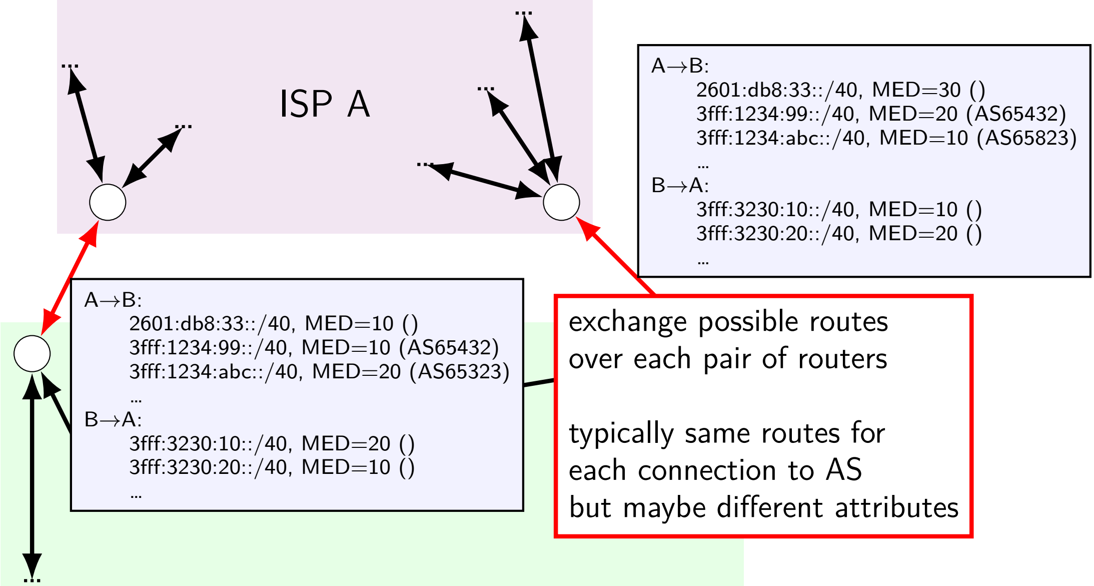
multiple BGP sessions



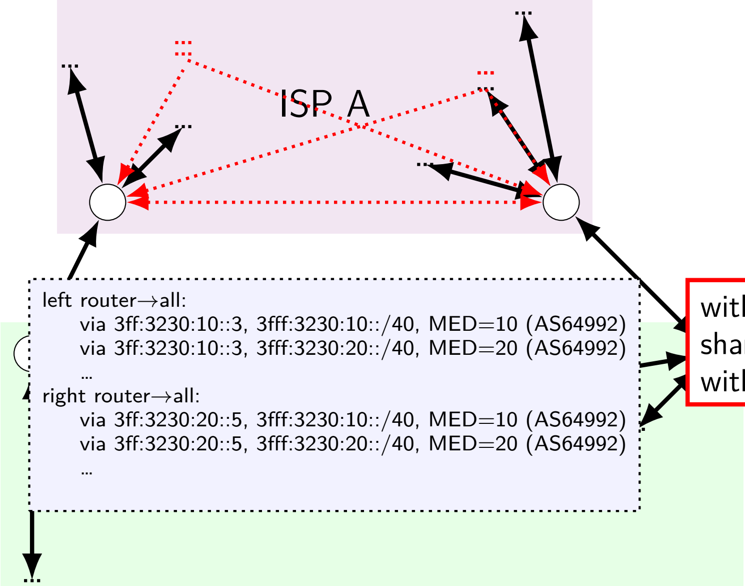
multiple BGP sessions



multiple BGP sessions



multiple BGP sessions



left router→all:

via 3ff:3230:10::3, 3fff:3230:10::/40, MED=10 (AS64992)
via 3ff:3230:10::3, 3fff:3230:20::/40, MED=20 (AS64992)
...

right router→all:

via 3ff:3230:20::5, 3fff:3230:10::/40, MED=10 (AS64992)
via 3ff:3230:20::5, 3fff:3230:20::/40, MED=20 (AS64992)
...

within ISP, use internal BGP (IBGP)
share everything learned via BGP
with all BGP routers

preference between routes

if multiple choices, most common strategy:...

should use most specific route

- use 2001:db8:1234::/40 over 2001:db8:1234::/39 if both apply
- (but usually reject very small address ranges (e.g. /31 for IPv4, /60 for IPv6))

then (if tie) local policy applies

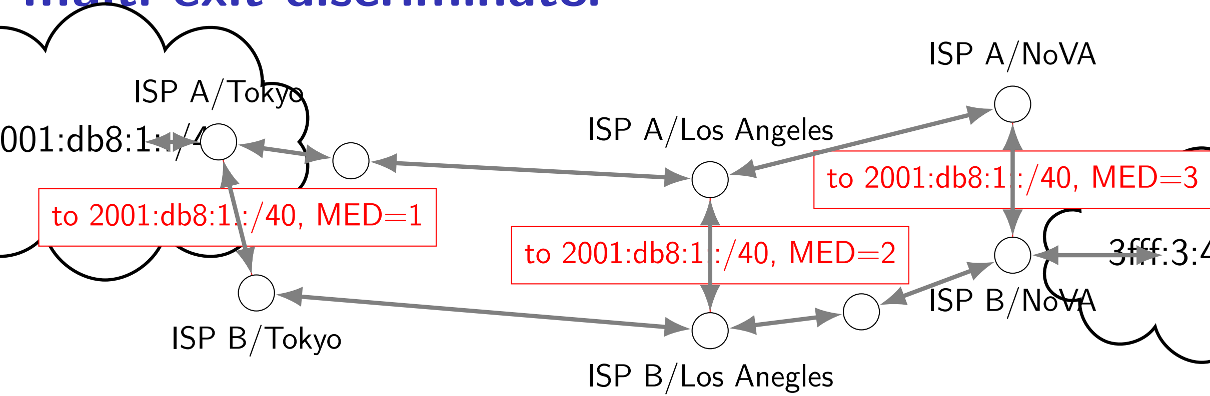
then shortest AS path

then lower AS number

then (sometimes) lower MED (multiple exit discriminator)

then best route within current AS

multi-exit discriminator



getting your preference

to affect how people route you, can...

prepend to AS path sent to make it longer

  typically add serveral copies of your AS number

only announce network from certain of your routers

  problem: won't have all 'backup' paths available

announce a large network in more specific pieces

  3fff:1234::/32 as 3fff:1234::/33 and 3fff:1234:8000::/33

get other networks to change how they forward your routes

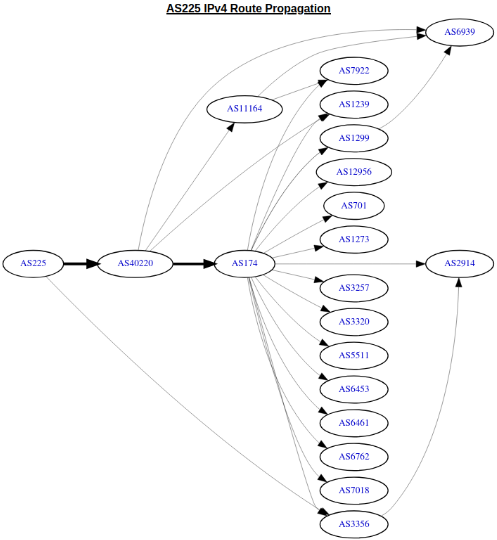
  often enabled through 'BGP communities'

https://bgp.he.net/super-lg/

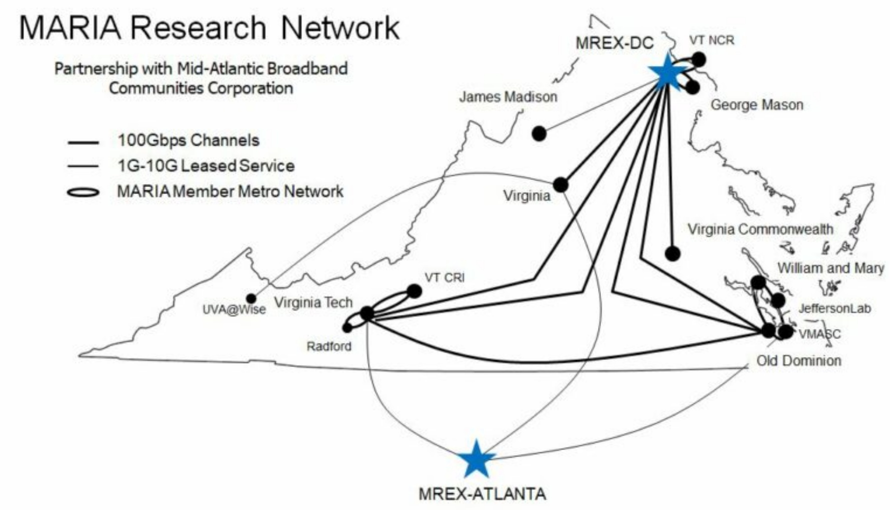
https://bgp.he.net/super-lg/#128.143.0.0/16?tob=none&mt=include&ma=6939&els=exact

https://lg.ring.nlnog.net/prefix?q=128.143.0.0/16&match=exact&peer=all

https://bgp.he.net/AS225 (University of Virginia)



AS40220


MARIA Research Network
Partnership with Mid-Atlantic Broadband Communities Corporation
100Gbps Channels
1G-10G Leased Service
MARIA Member Metro Network
MREX-DC
VT NCR
George Mason
James Madison
Virginia
Virginia Commonwealth
William and Mary
JeffersonLab
VMASC
Old Dominion
VT CRI
Virginia Tech
UVA@Wise
Radford
MREX-ATLANTA

PeeringDB

Search here for a network, IX, or facility.

Advanced Search | Legacy Search

Registeror Login

English (English)

Mid-Atlantic Terascale Partnership - MATP

EXPORT

Organization	Mid-Atlantic Terascale Partnership - MATP
Also Known As	MARIA / Virginia Tech (Virginia Polytechnic Institute and State University)
Long Name	
Company Website	
ASN	40220
IRR as-set/route-set	AS40220:AS-ALL
Route Server URL	
Looking Glass URL	
Network Types	Educational/Research
IPv4 Prefixes	90
IPv6 Prefixes	25
Traffic Levels	Not Disclosed
Traffic Ratios	Mostly Inbound
Geographic Scope	Regional
Protocols Supported	✓ Unicast IPv4 ○ Multicast ✓ IPv6 ○ Never via route servers
Last Updated	2022-07-27T05:34:22Z
Public Peering Info Updated	2024-07-24T16:31:34Z
Peering Facility Info Updated	2016-03-14T20:59:43Z
Contact Info Updated	2020-01-22T04:24:10Z
Notes	MATP (AS40220) is a consortium of research institutions in Virginia, Maryland, and Washington formed to to support research activities that require next-generation high-performance network connectivity. We have routers in Equinix Ashburn and TelX Atlanta and provide commodity and R&E network services to most research universities in Virginia. MATP is managed by Virginia

Public Peering Exchange Points

Filter

Exchange / IPv4	ASN / IPv6	Speed / Port Location	RS Peer	BFD Support
Digital Realty Atlanta 206.126.110.182	40220 2001:504:17:110::182	10G	○	○
Equinix Ashburn 206.126.236.139	40220 2001:504:0:2:0:4:220:1	20G	○	○

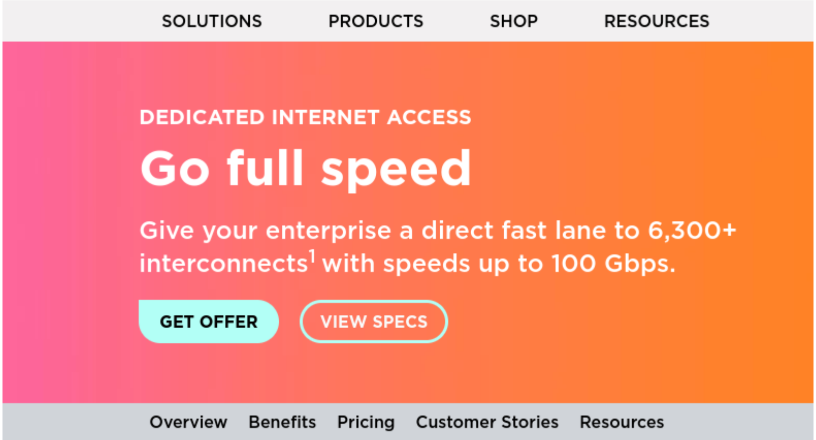
Interconnection Facilities

Filter

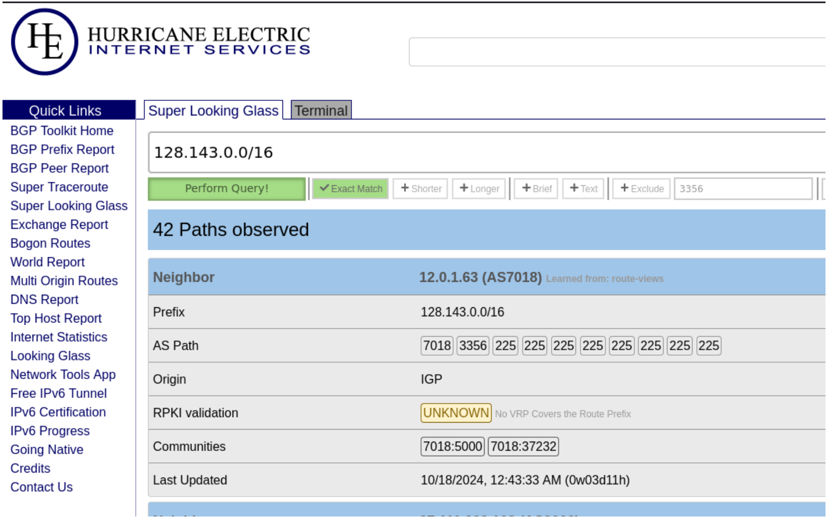
Facility / ASN	Country / City
Digital Realty ATL (56 Marietta) 40220	United States of America Atlanta
Equinix DC1-DC15, DC21 - Ashburn 40220	United States of America Ashburn

AS3356

LUMEN



AS3356 is a backup (8x AS prepending)



peeringdb

`https://peeringdb.com` — commonly used database of ASes and how to peer with them

there is also – “whois” records (from RIRs) for ASes, IP blocks with contact info

internet exchanges and route servers

internet exchange

- local network (typically within metro area) for connecting networks
- often run at and/or by ‘carrier-neutral’ datacenter
- typically high bandwidth (10-100Gbps ports to network)
- provides connections when

route servers

- BGP servers run by internet exchange
- consolidates routes from participants
- goal: only need O(n) BGP connections, not O(n^2)

BGP communities

routes sent via BGP can have ‘communities’

extra information tagged on routes sent via BGP

large ISPs have lists of communities their customers/peers can use

...and these affect how those routes are used

aside: Internet2

non-profit networking consortium

operations major US University-focused network

also makes eduroam work across different Universities

one of MARIA's major sources of connectivity (and indirectly one of UVA's major sources of connectivity)

has lots of peering relationships (incl. with big Internet companies)

(but not general Internet provider)

selected Internet2 BGP communities

Internet2 External Traffic Influencing Communities

International, Non-International, or FEDNET peers may send the below community and we will set their localpref to 460 or 560 respectively:

- Default - local-pref 500
- 11537:40 - Low (local-pref 460)
- 11537:160 - High (local-pref 560)

Connectors may send the below community and we will set their localpref to 540 or 620 respectively:

- Default - local-pref 600
- 11537:140 - Low (local-pref 540)
- 11537:260 - High (local-pref 620)

Internet2 Peers may send the following communities:

- 11537:2002 - Block prefix to commercial R&E peers.

Internet2 International (ITN) peers may send the below communities for path prepending:

- 65001:65000 - prepend x1
- 65002:65000 - prepend x2
- 65003:65000 - prepend x3

The following community combination of <CODE>:<ASN> allows you to block or prepend prefixes sent to individual international (ITN) peers. This is in the process of being deployed, once a peer has had the necessary configuration added, their ASN will be added here.

- Codes:
  - 65000 - prefixes will not be sent to ITN peer's ASN
  - 65001 - prefixes will be prepended 1 time to ITN peer's ASN
  - 65002 - prefixes will be prepended 2 times to ITN peer's ASN
  - 65003 - prefixes will be prepended 3 time2 to ITN peer's ASN
  - 65012 - prefixes will only be sent to ITN peer's ASN
- ITN Peer ASN:
  - 2603 - NORDUnet
  - 20965 - GEANT

The following community combination of <CODE>:<ASN> allows you to block or prepend prefixes sent to individual NET+ peers.

- Codes:
  - 65000 - prefixes will not be sent to NET+ peer's ASN
  - 65001 - prefixes will be prepended 1 time to NET+ peer's ASN
  - 65002 - prefixes will be prepended 2 times to NET+ peer's ASN
  - 65003 - prefixes will be prepended 3 time2 to NET+ peer's ASN
- NET+ peer ASN's
  - 16509 - Amazon
  - 62715 - Code42
  - 22556 - Blackboard
  - 16839 - ServiceNow
  - 19679 - DropBox

community options from prev slide

setting local-pref:

- you can decide how preferred your route is by Internet2
- maybe to make one primary, another secondary?

blocking route from being sent to specific place

prepending Internet2's AS before forwarding prefix

- hopefully make that route less preferred by others

prepending Internet2's AS before forwarding prefix to specific place

- hopefully make that route less preferred by that place

other things with communities

Internet2 also uses communities to mark...

what location routes were learned from

what type of organization routes were learned from

whether Internet2 is only allowed to use the route non-commerically or not

...

AS7007

https://seclists.org/nanog/1997/Apr/444

nanog mailing list archives

By Date By Thread

List Archive Search

7007 Explanation and Apology

From: "Vincent J. Bono" <vbono () MAI NET>
Date: Sat, 26 Apr 1997 19:41:35 EST

```
Dear All,

    I would like to sincerely apologize to everyone everwhere who
experienced problems yesterday due to the 7007 AS announcements.

    If anyone cares to know, here is what happened:

    At 11:30AM, EST, on 25 Apr 1997, our border router, stamped with
AS 7007, recieved a full routing view from a downstream ISP (well, a
view contacing 23,000 routes anyway).

    There was no distibute list imposed on the downstream since they
also advertise their customer AS's to us (they were also
experimenting with sending some routes out through us and some out
through the MAE).  We did filter out routes from them containing any
of our AS numbers but since they got the view from someone at
MAE-East none of our internal AS numbers showed up at all.  Not
having a filter imposed on the inbound side was our error.

    In an as yet unexplained twist of bits, the 7007 router then
began to de-aggregate the 23K route view *and* strip the AS path out
of it.  I will emphasize that we were running no IGP at the time.
Not one.  Not OSPF, not RIP, nothing.

    Our MAE-East border router, AS 6082, then got a feed of these
routes, at last count 73,000+, which set off our network monitor
system which wacthes for, among other things, route views over 45k
lines in size.  At 11:45AM we disabled the BGP peering session with
AS 1790 that was in place with the 7007 router and immediately
```

2008 Pakistan Youtube

Pakistan Telecom recieved gov't order to block youtube

implemented by inserting route for YouTube's IP in internal network

misconfiguration meant route was advertised on BGP

was more specific than YouTube's route, so made YouTube unreachablej

timeline from RIPE NCC

https://www.ripe.net/about-us/news/youtube-hijacking-a-ripe-ncc-ris-case-study/

Youtube is announcing 208.65.152.0/22

18:47Z: Pakistan Telecom starts announcing 208.65.153.0/24

20:07Z: Youtube starts announcing 208.65.153.0/24

20:18Z: Youtube starts announcing 208.65.153.0/25 and 208.65.153.128/25

20:51Z: Pakistan Telecom's ISP forwards their announcements with additional copy of Pakistan Telecom's AS number

21:01Z: Pakistan Telecom's ISP withdraws routes initiated by Pakistan Telecom (but not Pakistan Telecom's customers)

BGP Hijacking targeted cryptocurrency stuff

KLAYswap (Feb 2022), Celer Bridge (Sep 2022)

attackers intentionally redirected traffic to malicious version of services

...and stole money

both probably spoofed the final AS number in AS path

sometimes involved adding attacked IP range to routing registry

nation-states?

ars TECHNICA

AI BIZ & IT CARS CULTURE GAMING HEALTH POLICY SCIENCE SECURITY SPACE TECH FORUM | SUBSCRIBE | SIGN IN

THROWING DOWN THE GAUNTLET

Citing BGP hijacks and hack attacks, feds want China Telecom out of the US

With a history of cyber attacks, Chinese-owned telecom is a threat, officials say.

DAN GOODIN – APR 10, 2020 8:42 AM

route security

historically, no verification routes announced by "owner" of IP addresses

Internet Routing Registries — database of AS to IP address
- used automatically to filter out mistakes
- (not really designed to resist malicious attacks)

some verify which IPs they should have with RIRs
- "letter of agency/authority" to delegate

effort to deploy RPKI — public-key based scheme to verify routes
- checks that routes originated at correct AS
- doesn't verify intermediate ASes will forward correctly

Internet Engineering Task Force (IETF)
Request for Comments: 6480
Category: Informational
ISSN: 2070-1721
February 2012
M. Lepinski
S. Kent
BBN Technologies

An Infrastructure to Support Secure Internet Routing

Abstract

This document describes an architecture for an infrastructure to support improved security of Internet routing. The foundation of this architecture is a Resource Public Key Infrastructure (RPKI) that represents the allocation hierarchy of IP address space and Autonomous System (AS) numbers; and a distributed repository system for storing and disseminating the data objects that comprise the RPKI, as well as other signed objects necessary for improved routing security. As an initial application of this architecture, the document describes how a legitimate holder of IP address space can explicitly and verifiably authorize one or more ASes to originate routes to that address space. Such verifiable authorizations could be used, for example, to more securely construct BGP route filters.

partial tables

dealing with full Internet routing table is expensive

common shortcut if you have a couple ISPs:

keep 'short' routes (example: short AS path)

- to one of your "primary" ISPs
- maybe using ECMP

have default route for other cases

- special route for 0.0.0.0/0, ::/0
- to one of your ISPs

take advantage of more specific routes beating less specific

- typically also true in OSPF, RIP, etc.

backup slides