Introduction to Intra-Domain Routing

UCL Comput

COMP0023

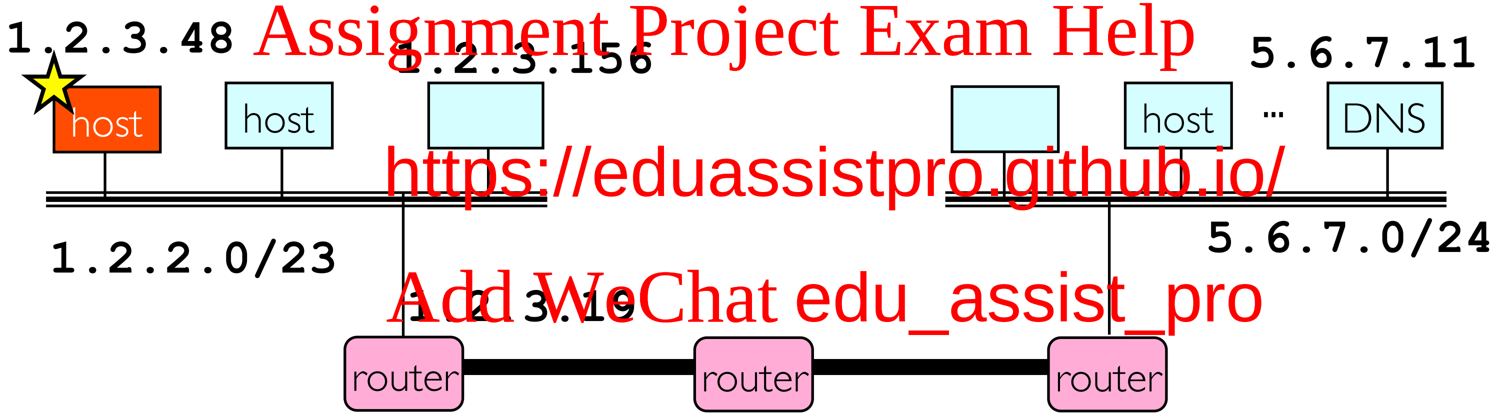
Agenda

- We delve into network layer's main functionality

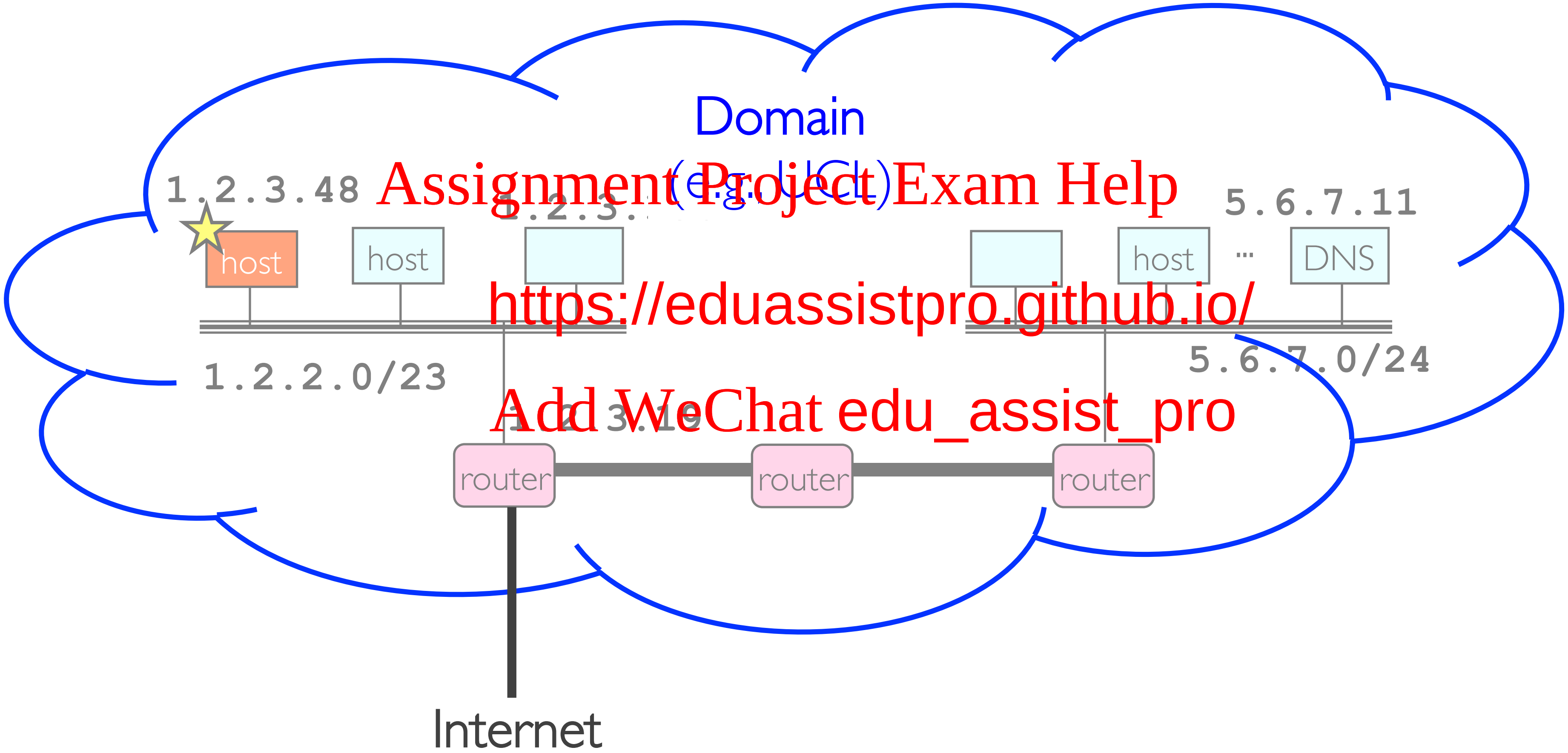
1. Setting
   - Context
   - Routing

2. Intra-domain routing problem definition

Let’s consider a network with several LANs



Several LANs can belong to a single organization



Collecting evidence from the real world

- Traceroute exposes real Internet paths
  - Unix: traceroute <destination>
  - Windows: tracert <destination>

- It displays all hops on the pa[illegible]n host where launched and <destination>
  - it sends a sequence of carefully constructed packets that "expire" after 1 hop, 2 hops, …
  - each of those packets elicits ICMP error from router that many hops from sender

A traceroute from UCL to Cambridge

traceroute to www.cl.cam.ac.uk (128.232.0.20), 64 hops max, 40 byte packets

1 cisco (128.16.64.1) 0.370 ms 0.322 ms 0.361 ms

2 128.40.255.29 (128.40.255.29) 0.483 ms 0.348 ms 0.487 ms

3 128.40.20.1 (128. 0.362 ms

4 128.40.20.62 (128 ms 0.363 ms

5 ulcc-gsr.lmn.net.uk (194.83.101.5) 0 6 ms 0.362 ms

6 london-bar1.ja.net (146.97.40.33) 0 0 ms 0.488 ms

7 po10-0.lond-scr.ja.net (146.97.35.5) 0.735 ms 0.722 ms 0.610 ms

8 po0-0.cambridge-bar.ja.net (146.97.35.10) 5.232 ms 4.964 ms 4.734 ms

9 route-enet-3.cam.ac.uk (146.97.40.50) 4.982 ms 4.841 ms 4.860 ms

10 route-cent-3.cam.ac.uk (192.153.213.194) 4.984 ms 4.964 ms 4.861 ms

A traceroute from UCL to Cambridge

traceroute to www.cl.cam.ac.uk (128.232.0.20), 64 hops max, 40 byte packets

1 cisco (128.16.64.1) 0.370 ms 0.322 ms 0.361 ms

2 128.40.255.29 (128.40.255.29) 0.483 ms 0.348 ms 0.487 ms

3 128.40.20.1 (128. 0.362 ms

4 128.40.20.62 (128 ms 0.363 ms

5 ulcc-gsr.lmn.net.uk (194.83.101.5) 0 6 ms 0.362 ms

6 london-bar1.ja.net (146.97.40.33) 0 0 ms 0.488 ms

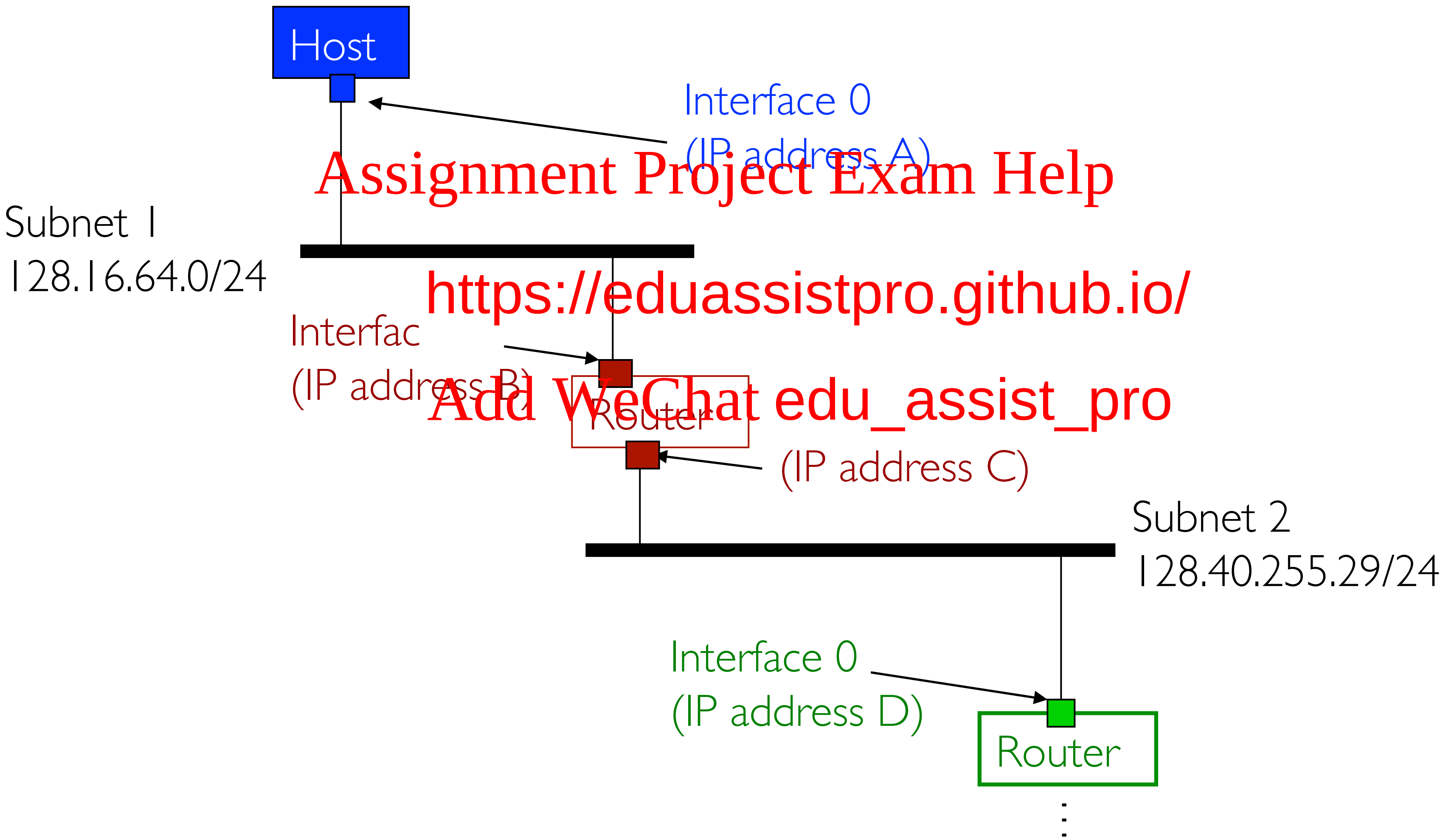
7 po10-0.lond-scr.ja.net (146.97.35.5) 0.735 ms 0.722 ms 0.610 ms

8 po0-0.cambridge-bar.ja.net (146.97.35.10) 5.232 ms 4.964 ms 4.734 ms

9 route-enet-3.cam.ac.uk (146.97.40.50) 4.982 ms 4.841 ms 4.860 ms

10 route-cent-3.cam.ac.uk (192.153.213.194) 4.984 ms 4.964 ms 4.861 ms

The big picture emerging from the traceroute



Routers keep routing tables

Three fields

- Destination: destination IP

- Outgoing interface: on which to forward packet for the given destinati

- Metric: total cost to reach t ion

  – depends on interfaces' metrics (set by net admins)

Destination	Interface	Metric
A	0	0
B	1	0

Routers' destinations are IP prefixes

- Each host (interface) has unique 32-bit IP address
  - E.g., 128.16.64.1

- Must every ro now about every other router a

  No; interfaces on same sub P prefix
    - e.g., 128.16.64/24 for 128.16.64.1

- IP routing destination is subnet's prefix
  - Not single IP addresses

Routers use routing table to forward packets

- Packet for destination D arrives
- Router searches D in destination field of routing table
  - using longest-prefix match
    - If more th... ose the route with the longest prefix
- Possible outcomes:
  - if entry for D, forward on interface in the entry
  - if no entry, no route known → drop packet

Routers use routing table to forward packets

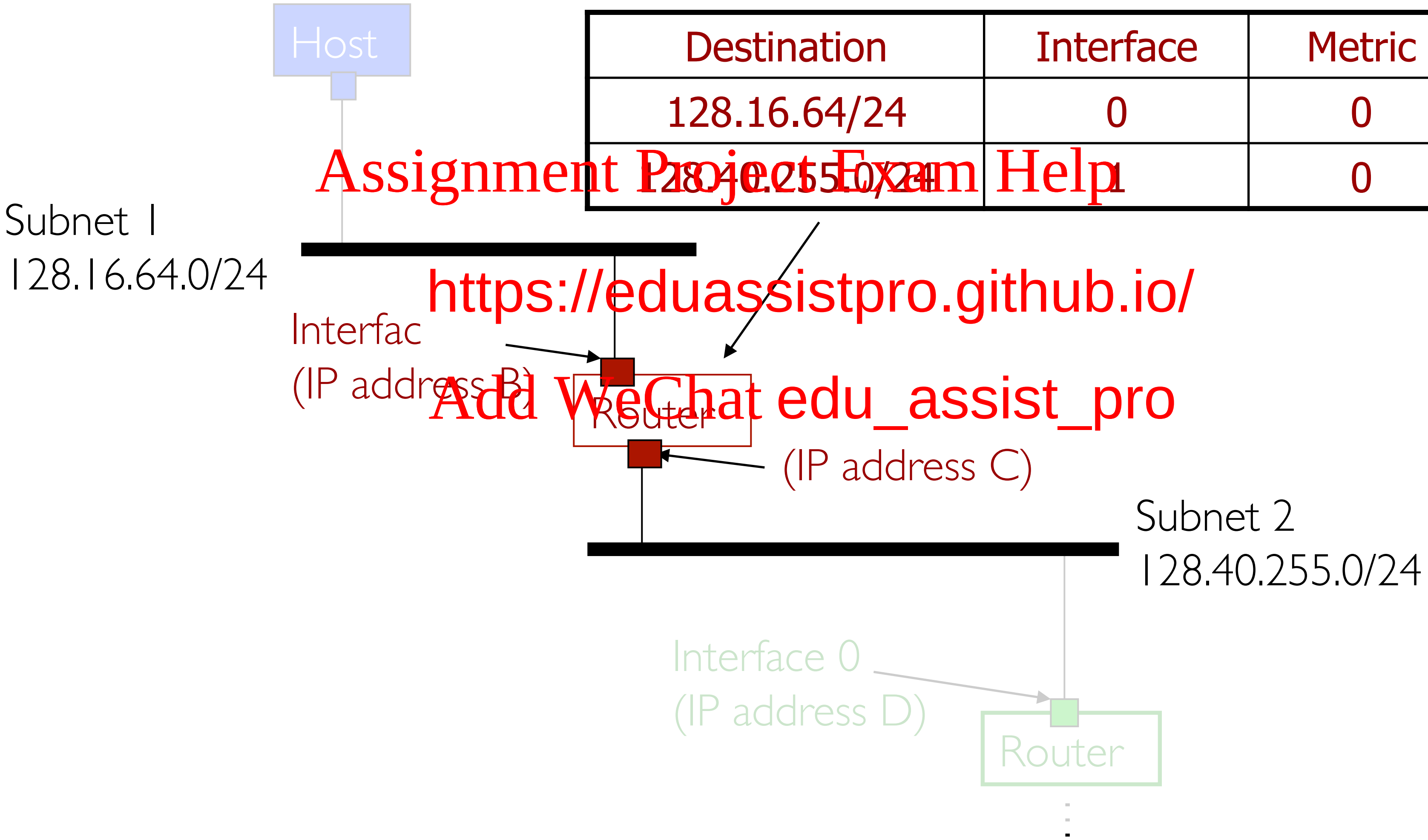
- Packet for destination D arrives
- Router searches D in destination field of routing table
  - using longest-prefix match
    - If more th... route with the longest prefix
- Possible outcomes:
  - if entry for D, forward on interface in the entry
  - if no entry, no route known → drop packet

Ho tables

f ain?

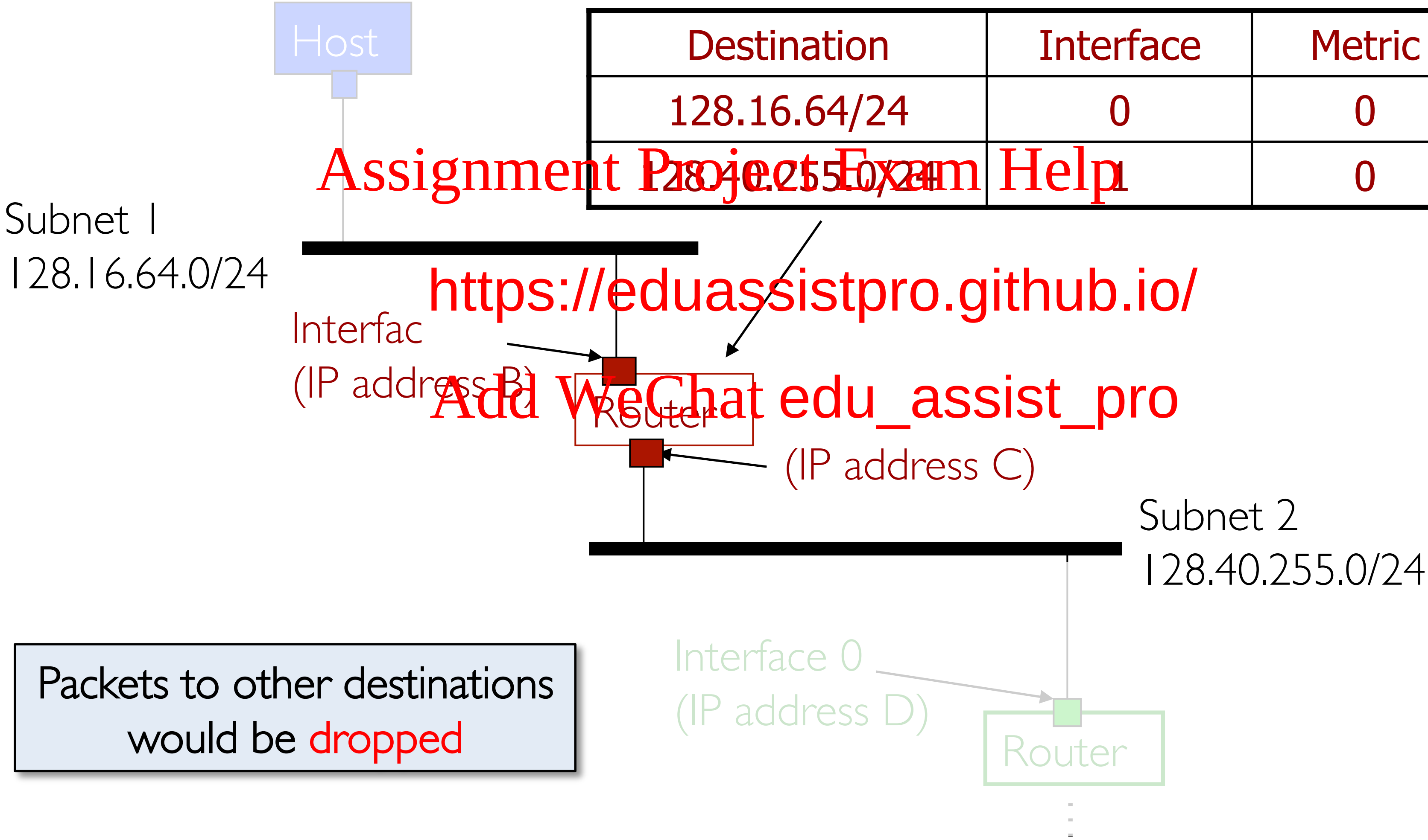
At startup, a router initializes its routing table for directly connected subnets

Destination	Interface	Metric
128.16.64/24	0	0
128.40.255.0/24	1	0



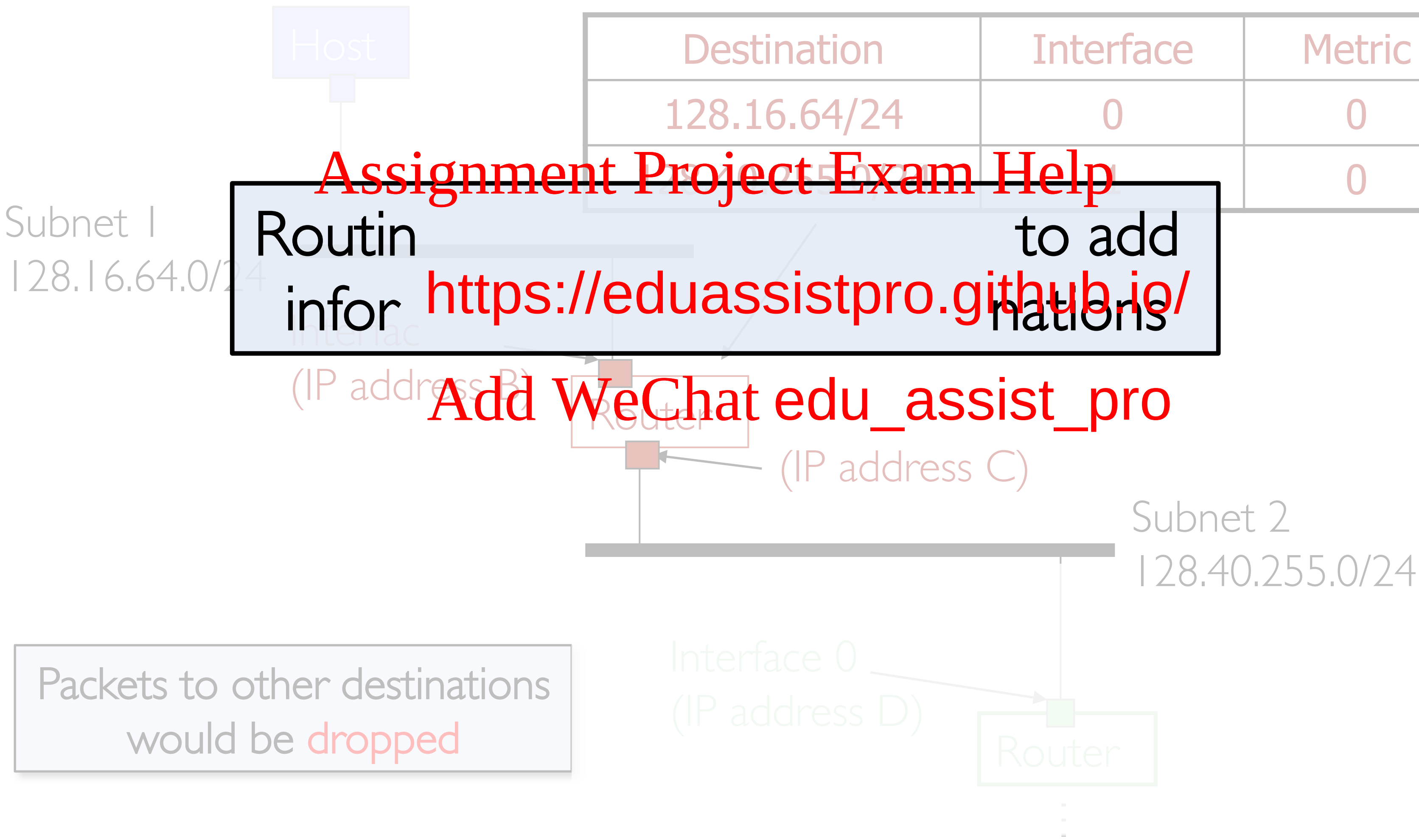
At startup, a router initializes its routing table for directly connected subnets

Destination	Interface	Metric
128.16.64/24	0	0
128.40.255.0/24	1	0



Packets to other destinations would be dropped

At startup, a router initializes its routing table for directly connected subnets



Destination	Interface	Metric
128.16.64/24	0	0
128.40.255.0/24	1	0

Routin to add infor nations

Packets to other destinations would be dropped

Agenda

- We delve into network layer's main functionality

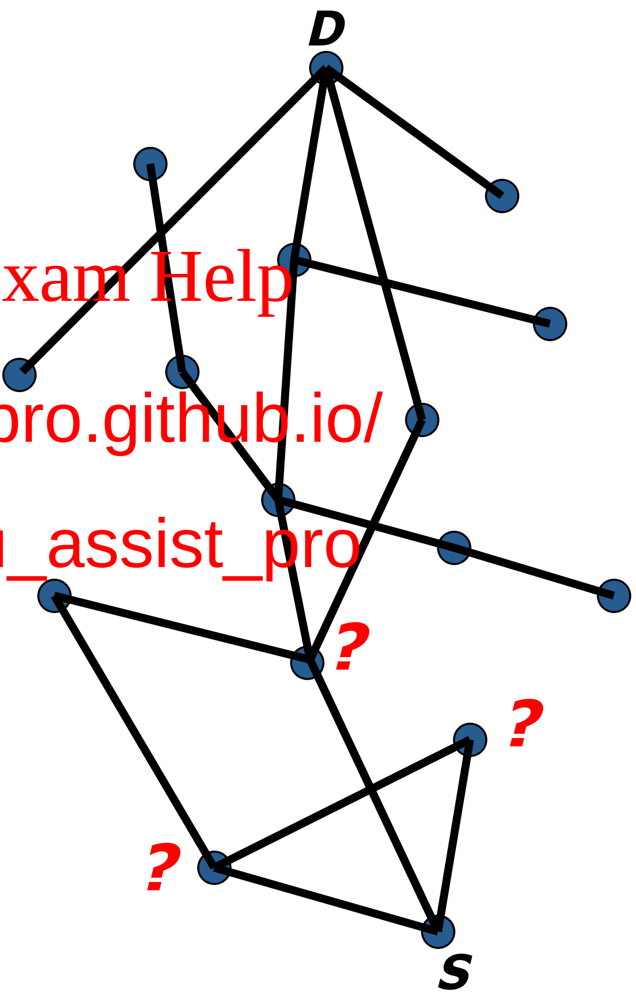
1. Setting
   - Context
   - Routing

2. Intra-domain routing problem definition

Consider an arbitrary network

- Each router has an ID and several interfaces
  - Linked to many other routers and

- Each router must choose the next-hop for received packets
  - Packets include info on destinations (IP addresses)
    - But not on intermediate hops



Routing protocols build routing tables

- Routing protocols define information and messages exchanged by routers
  - For routers t
    state (routin
    to forward packets
- Routing protocols also determine how to choose between alternative next-hops

Intra-domain routing goals

- Provide good performance
  - Typically, choose "good" paths for any pair of routers

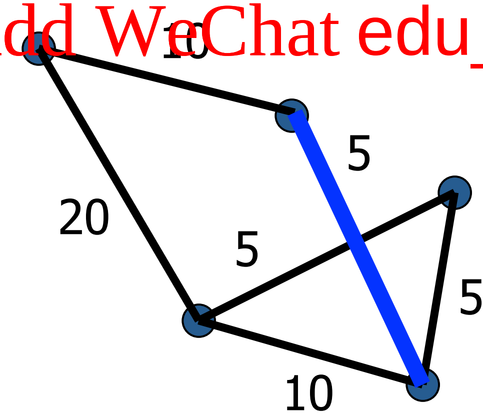
- Automated pat

  - Possibly, efficient in time an

- Deal with failures
  - Routes must be coherent with topology

Typical solution: shortest path routing

- View network as weighted graph
  - Routers are vertices, links are edges
  - Link metrics are edge weights



Typical solution: shortest path routing

Shortest paths problem:

– What path between two vertices offers minimal sum of edge weights?

- Classic graph algorithms find si shortest paths when the entire graph is know

– Dijkstra's Algorithm, Bellman-Ford Algorithm

- Typically, no central knowledge of entire graph

– Each router only knows its own interfaces' addresses

Challenge: deal with changing networks

- Network components are not reliable
  – Links may be cut, router or their interfaces may fail

- Changing netw
  – Need for (fa

aths after a failure

  – Potential for forwarding loops
    - Amplify traffic → congest links
      – TTL will eventually expire, but typically too late
  – Possibility of temporary disconnections (i.e., blackholes)

Distance Vector Routing

UCL Comput

COMP0023

Agenda

- We are now ready to study Distance Vector routing

1. Algorithm

2. Pathologie

3. Optimizations

Basic Distance Vector Algorithm

- Distributed Bellman-Ford (DBF)

- Principle: "tell everything you know to your neighbours"

  – Periodically, send all your ble entries (destination and metric fi your neighbouring routers

Basic Distance Vector Algorithm: DBF
(Failures Not Yet Considered)

Upon receipt of routing table entry for destination D with metric m on interface i:

m += *configured metric for interface* i

rt = lookup(D)

if (rt = "not fo

rt_new = *new routing ta*

$rt\_new.dst$ = D; $rt\_new.metric$ = m; $rt\_new.iface$ = i

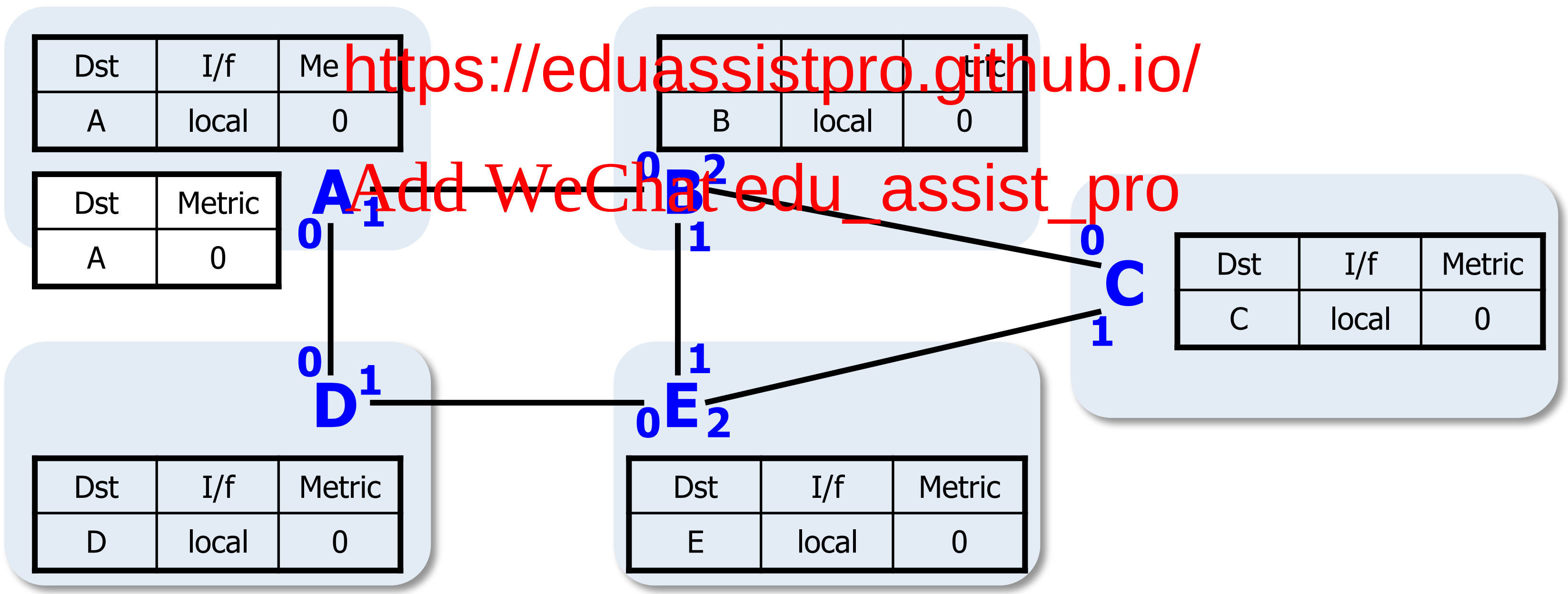
add rt_new *to routing table*

else if (m < $rt.metric$) then

$rt.metric$ = m; $rt.iface$ = i

Distance Vector: Example

- Consider simple network where all nodes are routers, addresses are simply single letters
- Initial routing tables when routers first start:



Dst	I/f	Metric
A	local	0

Dst	Metric
A	0

Dst	I/f	Metric
B	local	0

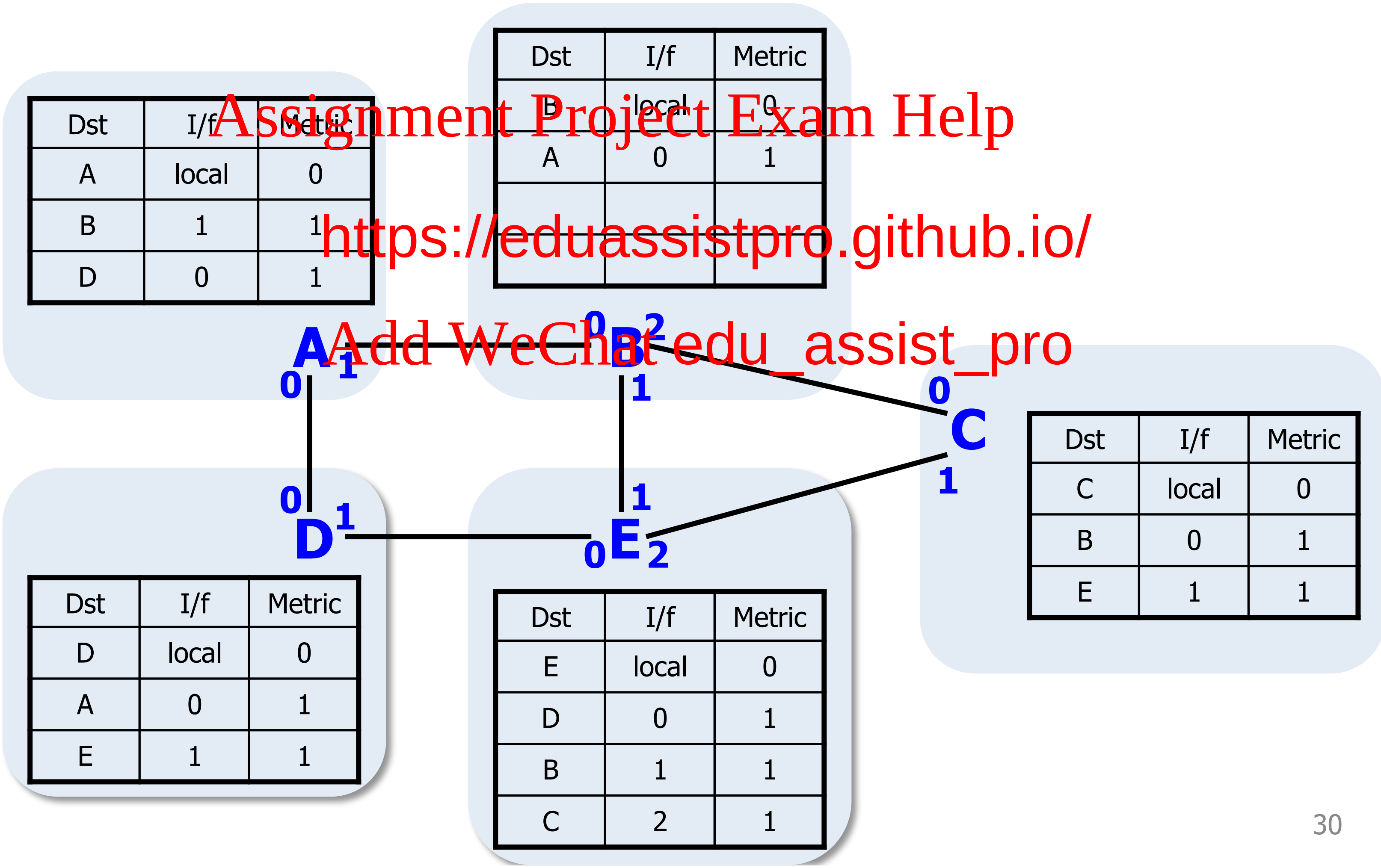
Dst	I/f	Metric
C	local	0

Dst	I/f	Metric
D	local	0

Dst	I/f	Metric
E	local	0

Distance Vector: Iteration 1

- Routers incorporate received announcements:



Dst	I/f	Metric
A	local	0
B	1	1
D	0	1

Dst	I/f	Metric
B	local	0
A	0	1

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1

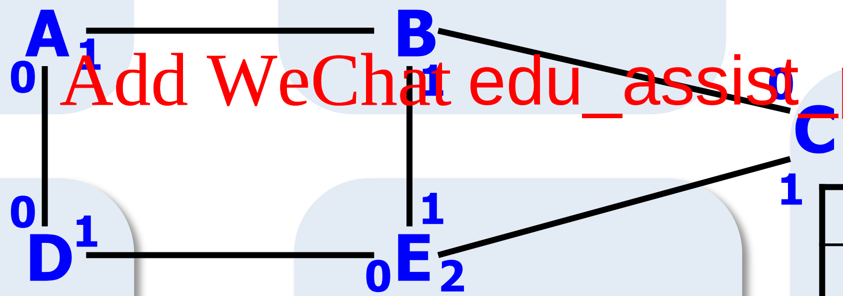
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1

Distance Vector: Iteration 2

- Routers incorporate received announcements:

Dst	I/f	Metric
A	local	0
B	1	1
D	0	1
E	0	2
C	1	2

Dst	I/f	Metric
B	local	0
A	0	1
C	2	1



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	0	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

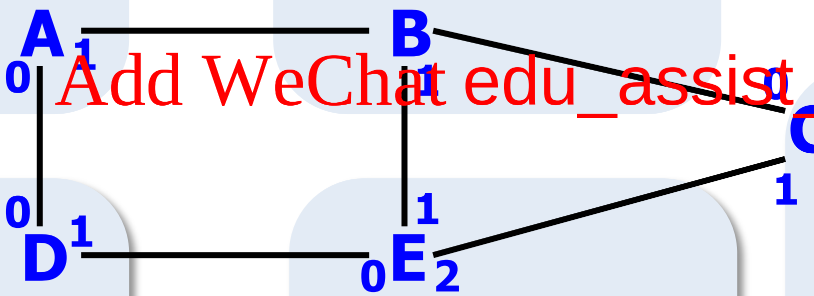
Distance Vector: Iteration 2

Convergence:

routing tables no longer changing;
routes reflect up-to-date knowledge of topology

D	0	1
E	0	2
C	1	2

C	2	1



Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

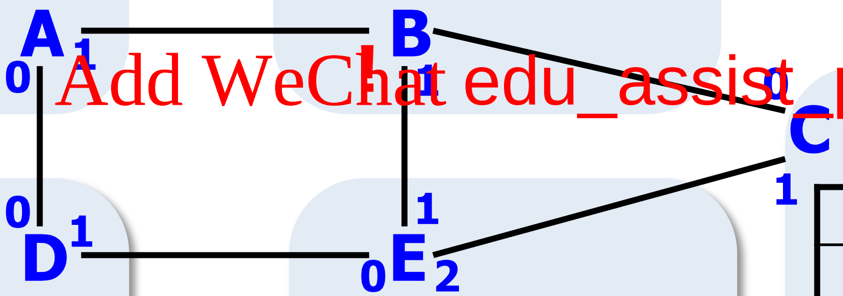
Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	0	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Link Failure (1)

Dst	I/f	Metric
A	local	0
B	1	1
D	0	1
E	0	2
C	1	2

Dst	I/f	Metric
B	local	0
A	0	1
C	2	1



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	0	2
C	1	2

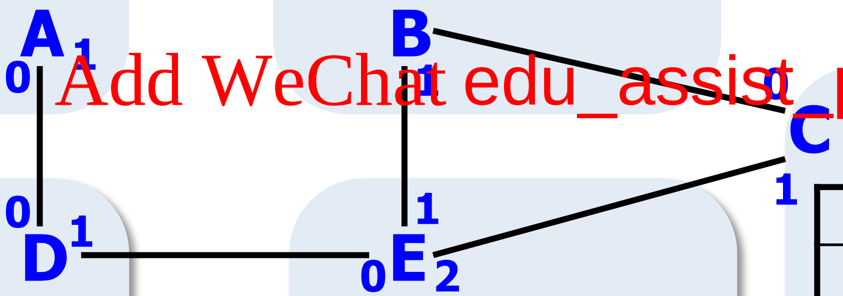
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Link Failure (1)

Dst	I/f	Metric
A	local	0
B	1	**Inf**
D	0	1
E	0	2
C	1	[illegible]

Dst	I/f	Metric
B	local	0
A	0	**Inf**
C	2	1



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	0	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

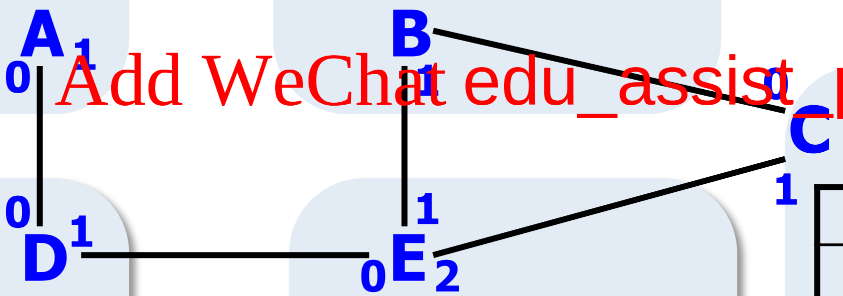
Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Link Failure (2)

Dst	I/f	Metric
A	local	0
B	1	**Inf**
D	0	1
E	0	2
	1	**Inf**

Dst	I/f	Metric
B	local	0
A	0	**Inf**
C	2	1

Dst	Metric
A	0
B	**Inf**
D	1
E	2
C	**Inf**



Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	0	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

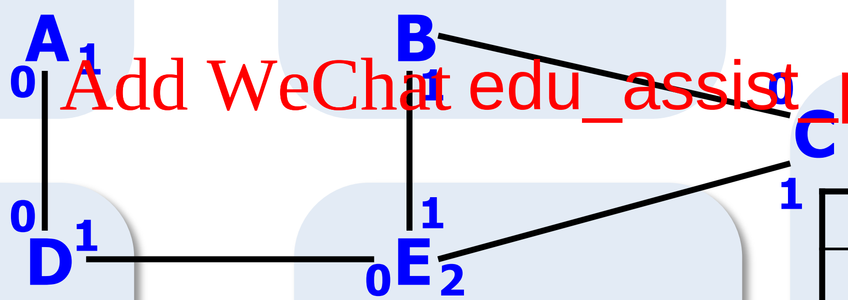
Link Failure (2)

Problem:
D ignores the update for the route to B because the metric is worse than the one it already has.

Dst	I/f	Metric
A	local	0
B	1	**Inf**
D	0	1
E	0	2
[illegible]	1	[illegible]

Dst	I/f	Metric
A	0	**Inf**
C	2	1

Dst	Metric
A	0
B	**Inf**
D	1
E	2
C	**Inf**



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	0	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

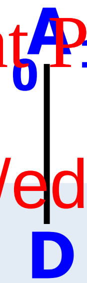
Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

DV Algorithm, Revised

- Upon receipt of routing table entry for destination D with metric m on interface i:


```
m += metric for interface i
rt = lookup(D) i
if (rt = “not fou
    r_new = new routing table
    r_new.D = D;   r_new.metri              w.iface = i
    add r_new to table
else if (i == rt.iface) then
    rt.metric = m
else if (m < rt.metric) then
    rt.metric = m;   r.iface = i
```

Link Failure (3)



Dst	Metric
A	1
B	**Inf**
D	2
E	3
C	**Inf**

+

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	0	2
C	1	2

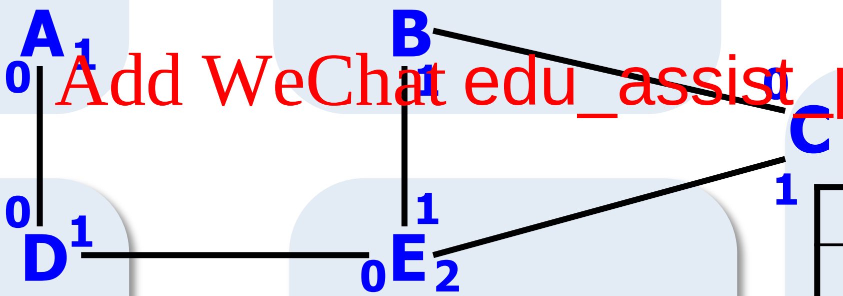
→

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	0	Inf
C	1	2

Link Failure (3)

Dst	I/f	Metric
A	local	0
B	1	Inf
D	0	1
E	0	2
C	1	Inf

Dst	I/f	Metric
B	local	0
A	0	Inf
C	2	1

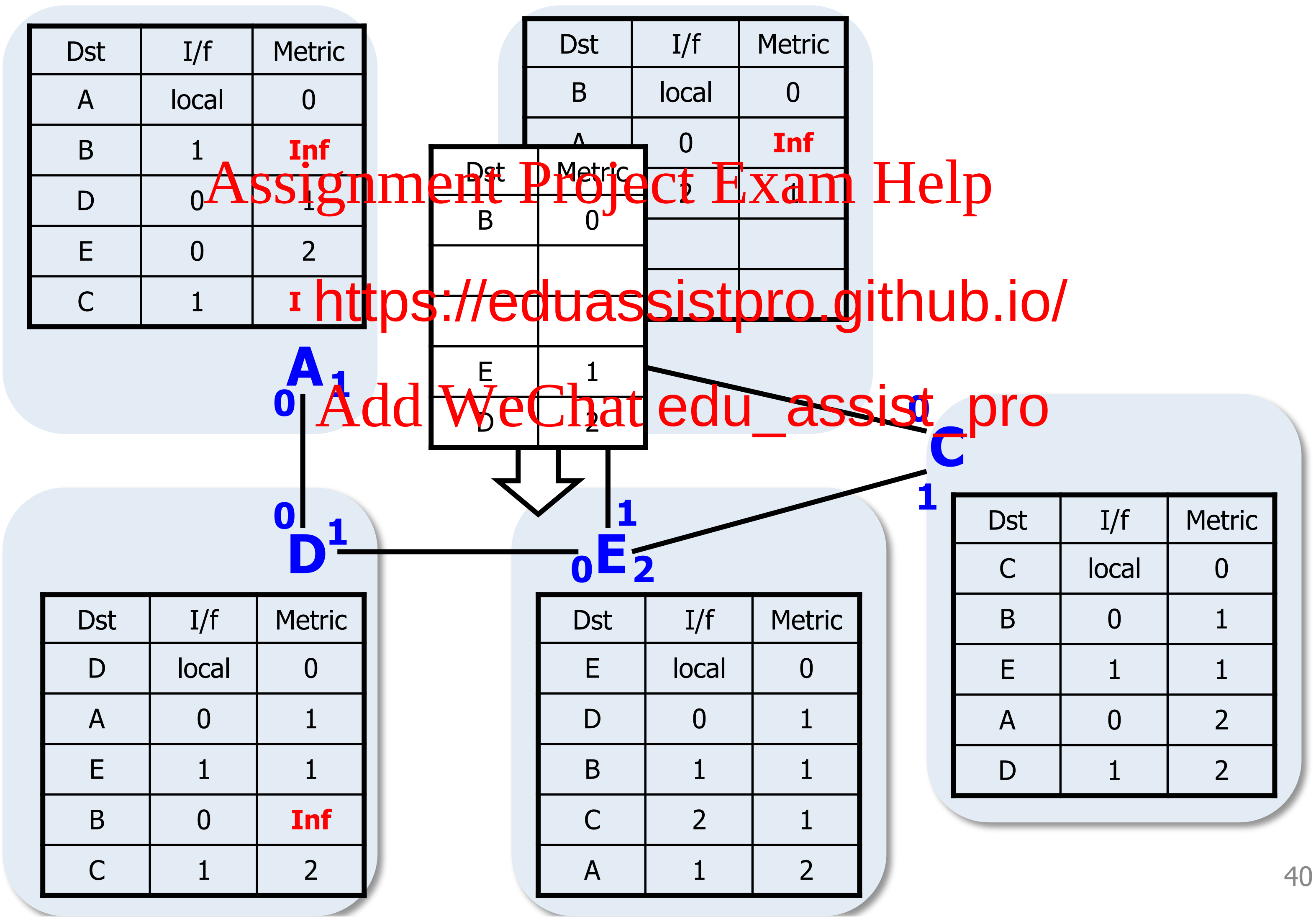


Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	0	Inf
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Link Failure (4)



Dst	I/f	Metric
A	local	0
B	1	Inf
D	0	1
E	0	2
C	1	[illegible]

Dst	I/f	Metric
B	local	0
A	0	Inf
[illegible]	2	[illegible]

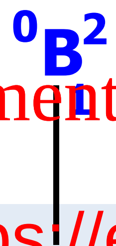
Dst	Metric
B	0
[illegible]	[illegible]
E	1
D	2

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	0	Inf
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Link Failure (5)



Dst	Metric
B	1
A	**Inf**
C	2
E	2
D	3

+

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

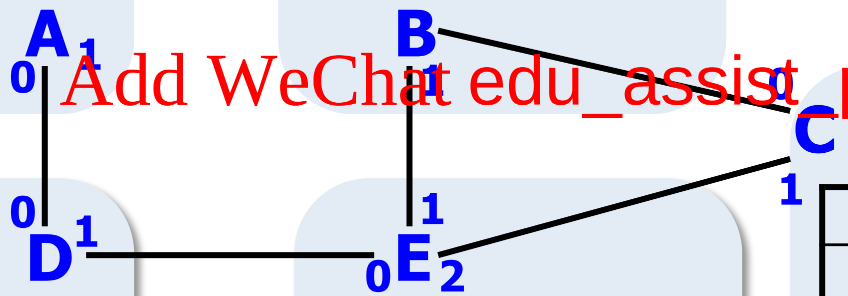
→

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	**Inf**

Link Failure (5)

Dst	I/f	Metric
A	local	0
B	1	**Inf**
D	0	1
E	0	2
C	1	[illegible]

Dst	I/f	Metric
B	local	0
A	0	**Inf**
C	2	1



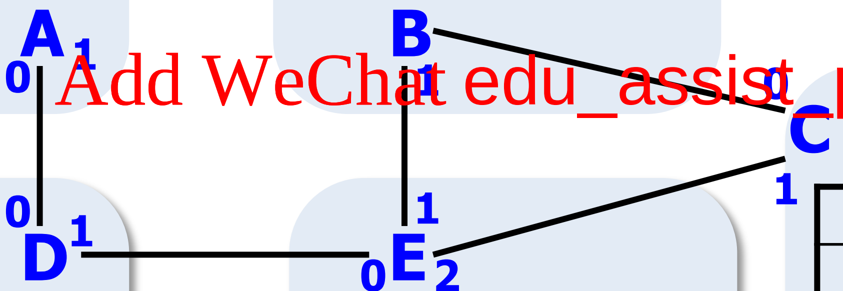
Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	0	**Inf**
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	**Inf**

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Link Failure (6)

Next round: all routers broadcast their tables to neighbours…



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	0	**Inf**
C	1	2

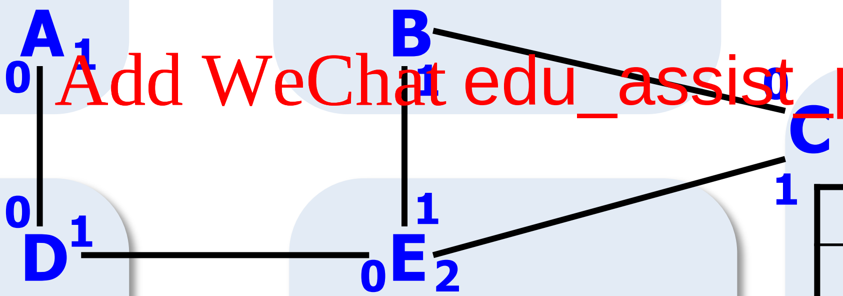
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	**Inf**

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Link Failure (7)

Dst	I/f	Metric
A	local	0
B	1	**Inf**
D	0	1
E	0	2
C	**0**	

Dst	I/f	Metric
B	local	0
A	0	**Inf**
C	2	1



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	**1**	**2**
C	1	2

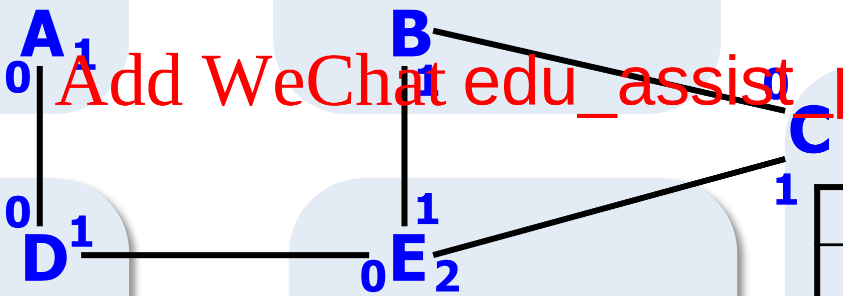
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	**0**	**2**

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	**Inf**
D	1	2

Link Failure (8)

Dst	I/f	Metric
A	local	0
B	**0**	**3**
D	0	1
E	0	2
C	**0**	

Dst	I/f	Metric
B	local	0
A	**1**	**3**
C	2	1



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	**1**	**2**
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	**0**	**2**

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	**1**	**3**
D	1	2

Agenda

- We are now ready to study Distance Vector routing

1. Algorithm

2. Pathologie

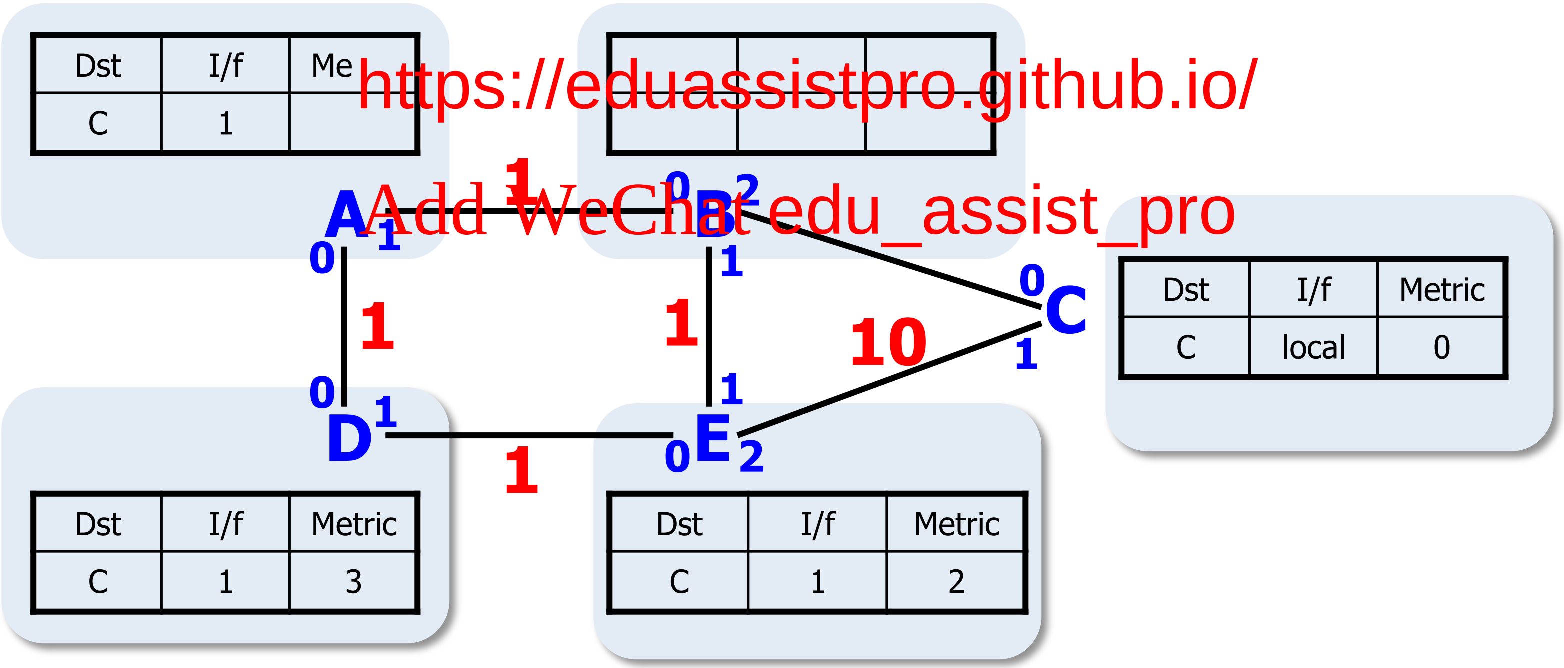
   - Bouncing

   - Counting to Infinity

3. Optimizations

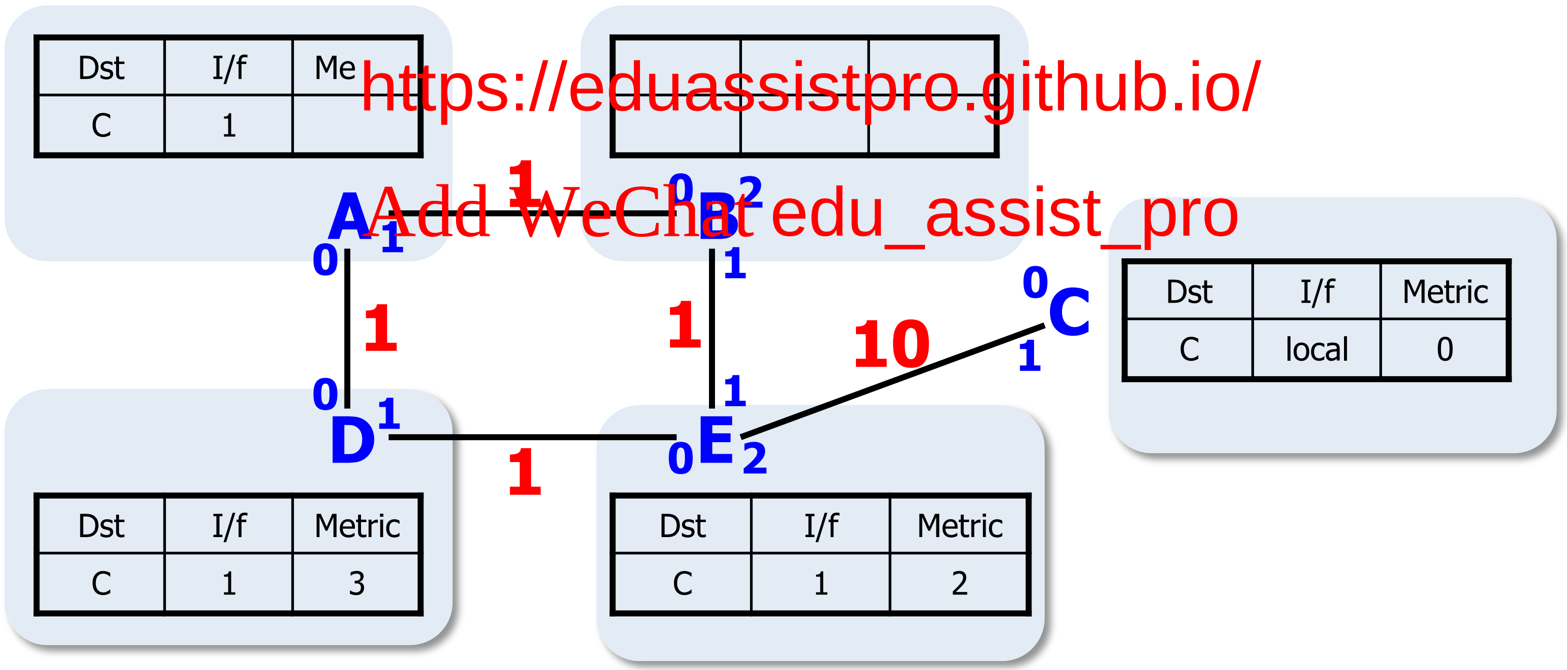
Bouncing (I)

- Consider same network, where link (C, E) has metric 10; all others have metric 1
- Consider all nodes' routes to C after convergence
- Now suppose link (B, C) breaks



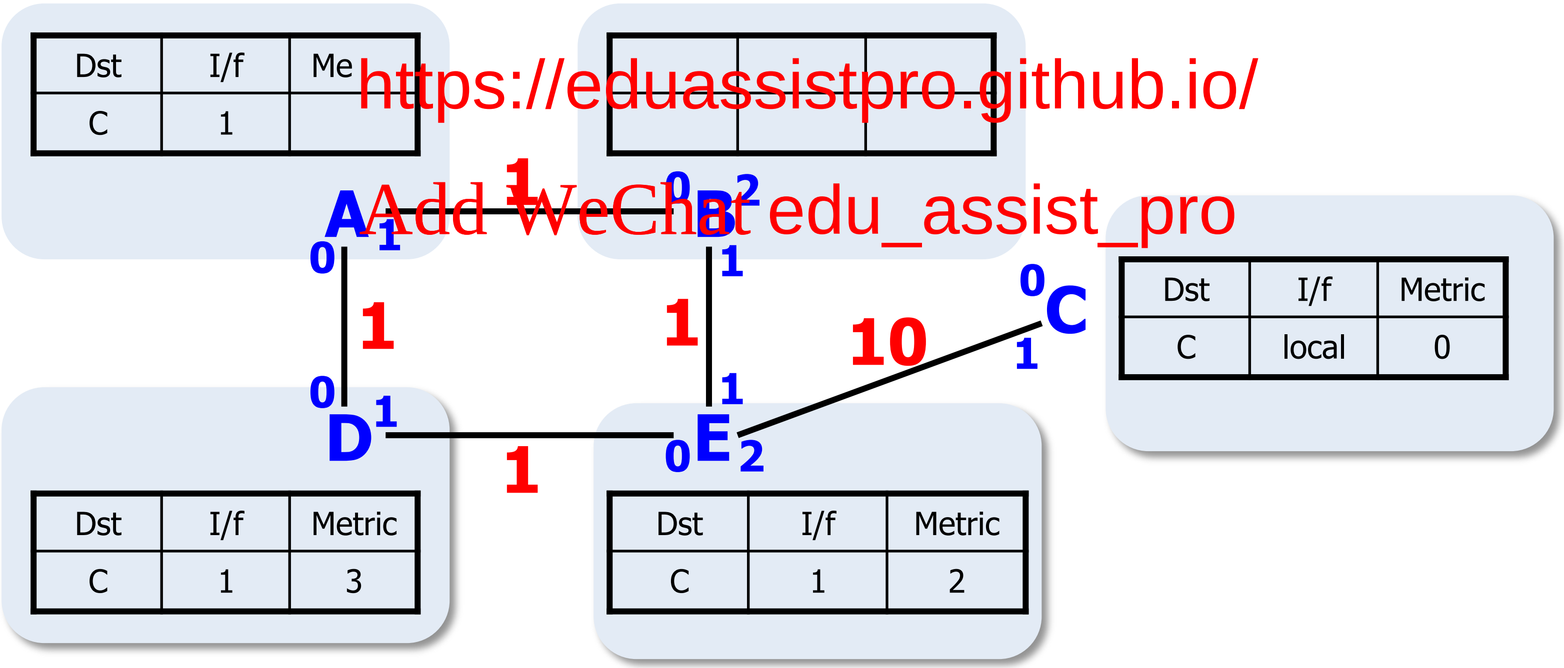
Bouncing (II)

- Suppose A advertises its table first…



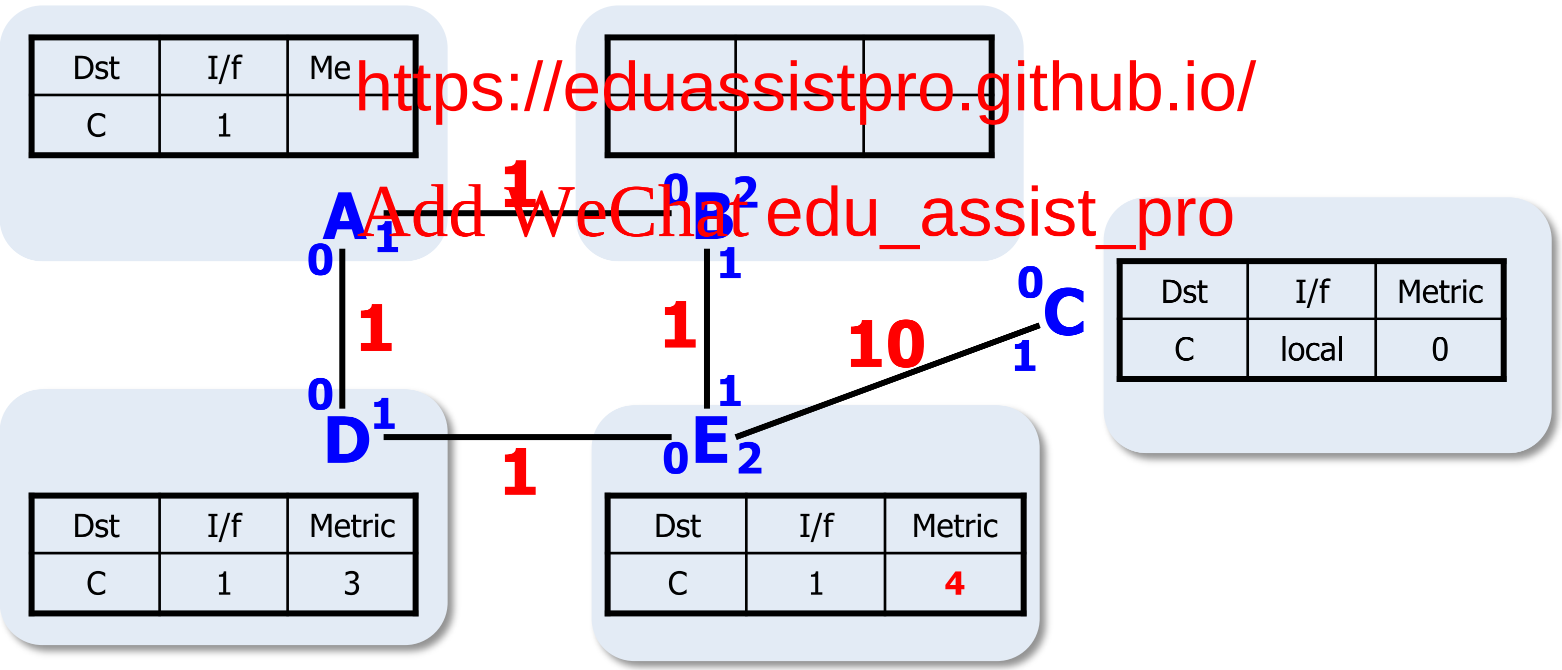
Bouncing (III)

- Suppose A advertises its table first…
- …and B advertises its table next…



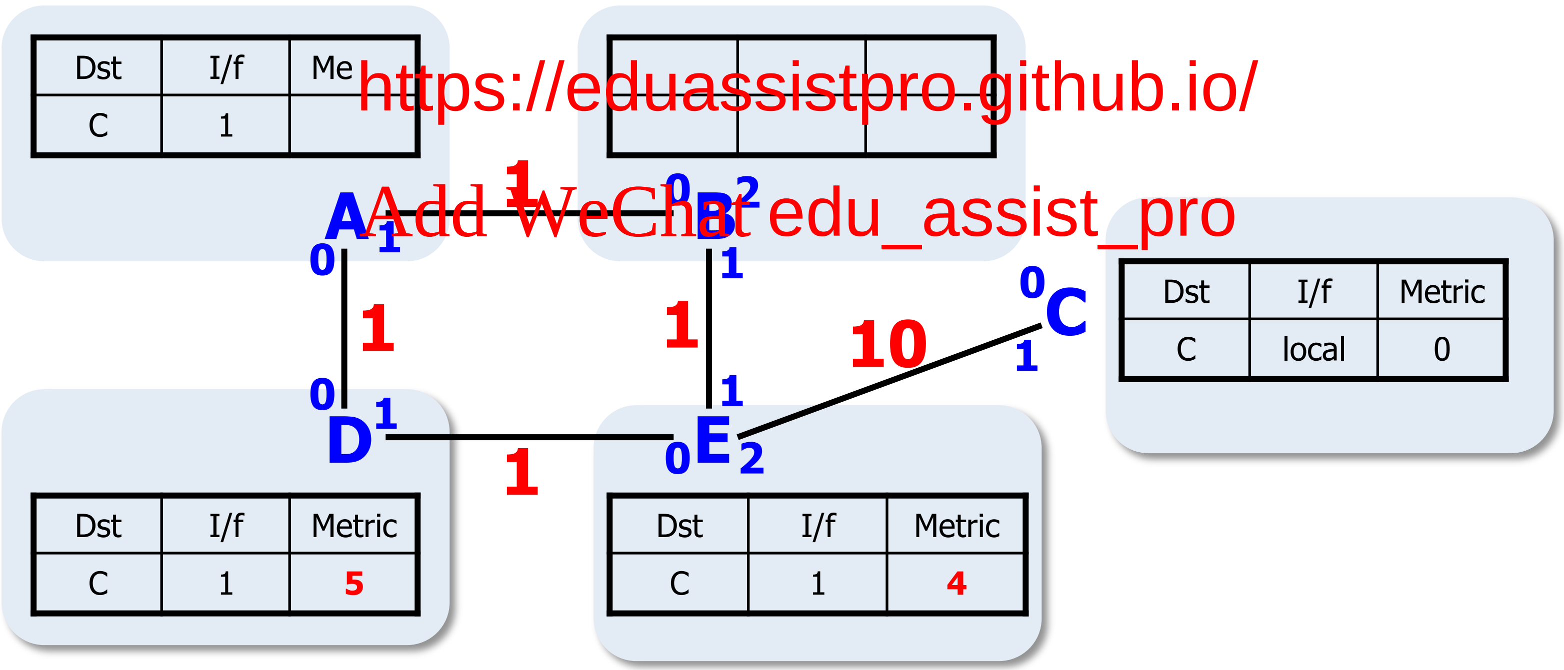
Bouncing (IV)

- Suppose A advertises its table first…
- …and B advertises its table next…
- Loop between A and B for destination C!
- If C now advertises its table, E will ignore cost 10 route!



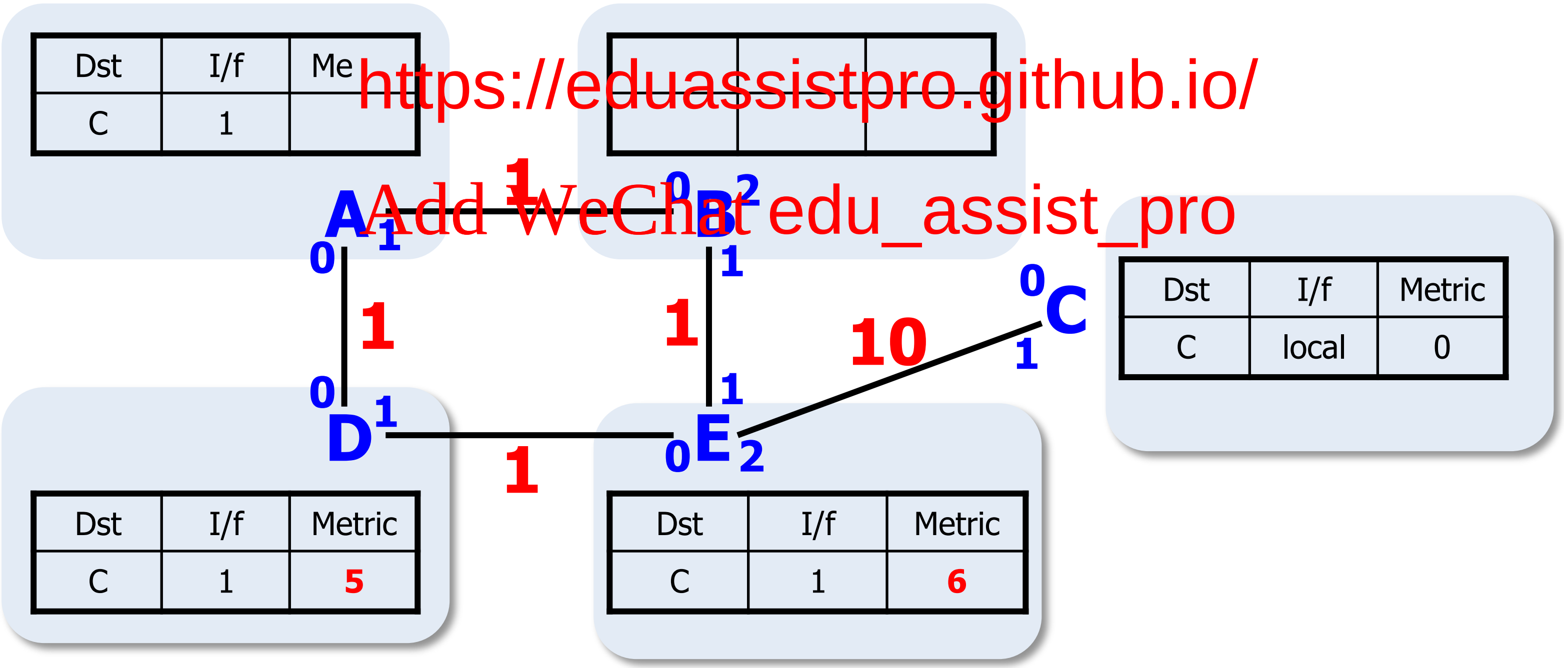
Bouncing (V)

- Suppose A and E advertise next… (B,D update metrics)



Bouncing (VI)

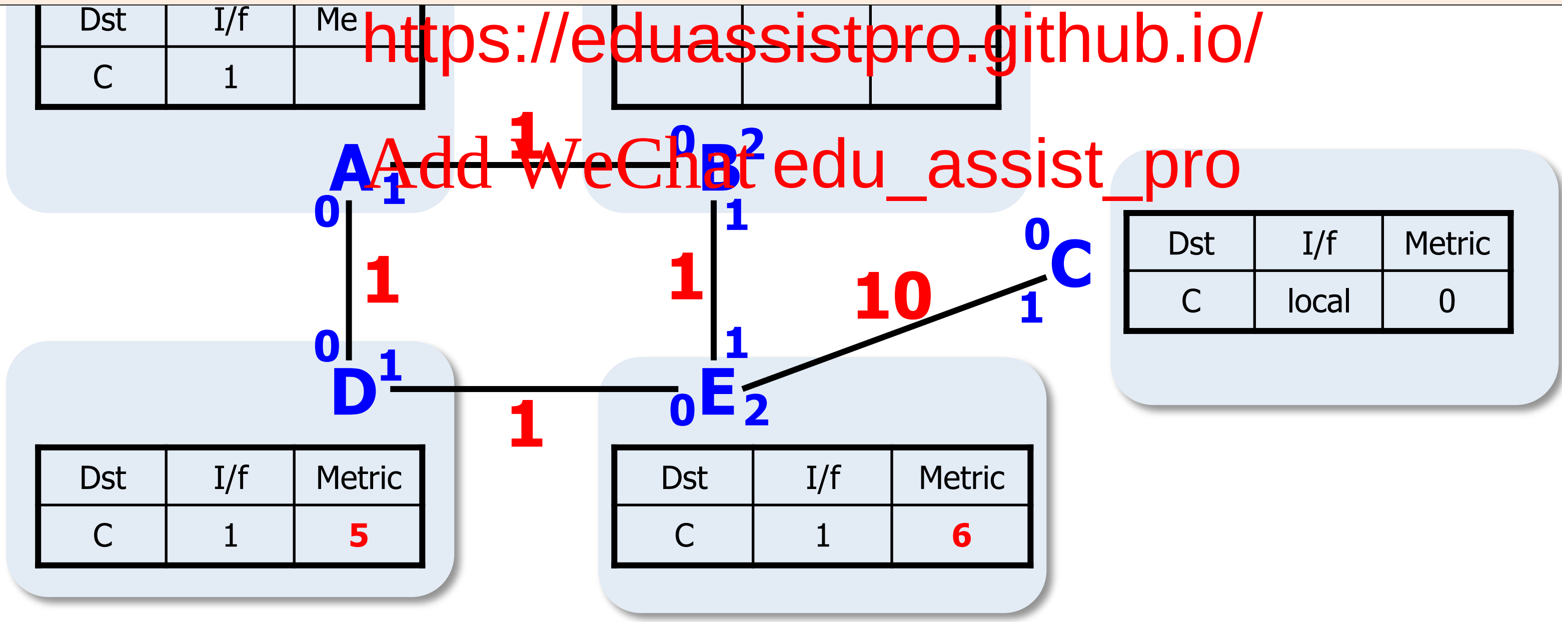
- Suppose A and E advertise next… (B,D update metrics)
- …and B advertises next (A and E update metrics)



Bouncing (VII)

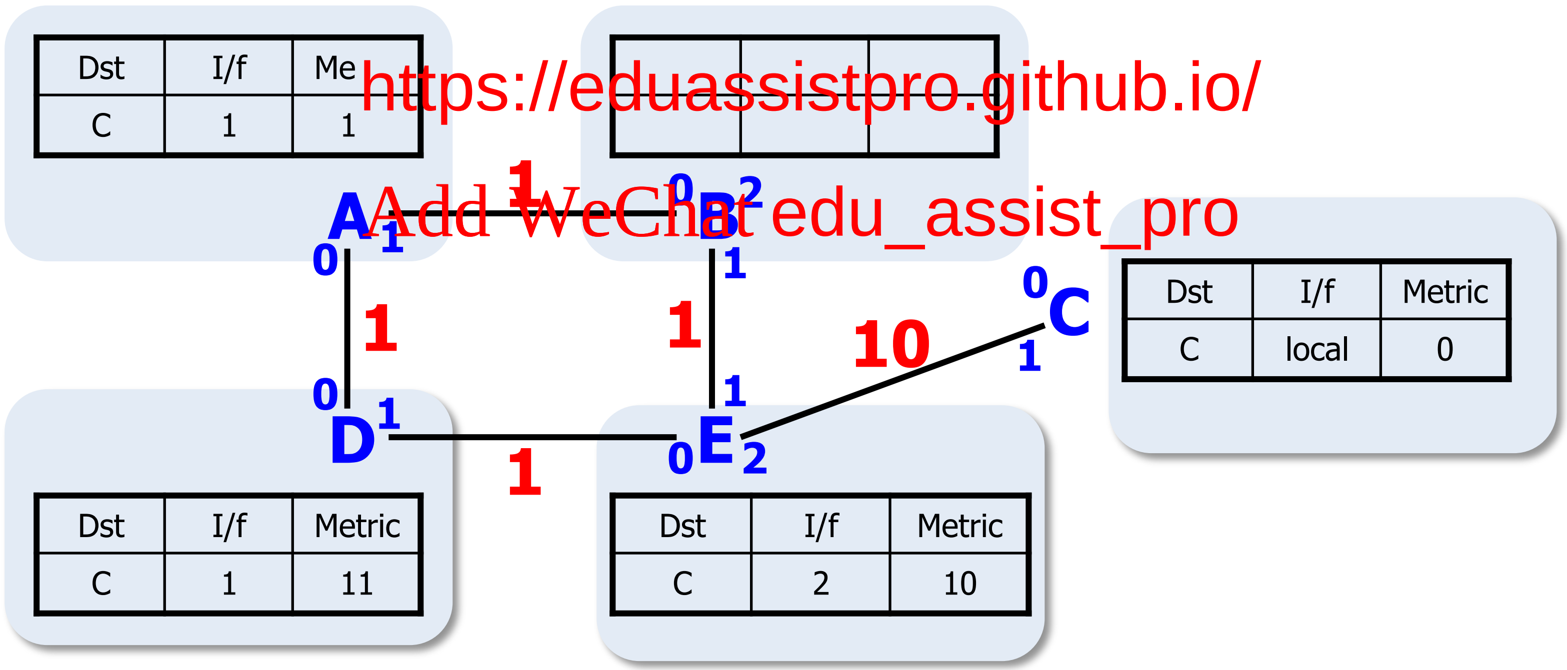
- Suppose A and E advertise next… (B,D update metrics)

And so on…



Bouncing (VIII)

- Long, painful convergence process, details dependent on message ordering
- Transient loops
- Eventually, converged state:

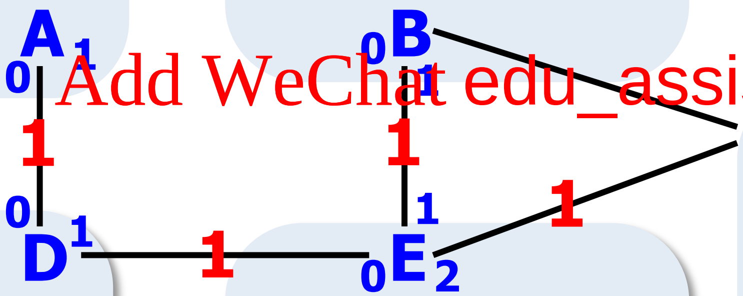


Counting to Infinity (I)

- Converged after link (A, B) breaks
- Suppose (D, E) now breaks

Dst	I/f	Metric
A	local	0
B	0	3
D	0	1
E	0	2
C	0	3

Dst	I/f	Metric
B	local	0
A	1	3
C	2	1



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	0	2

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	1	3
D	1	2

Counting to Infinity (II)

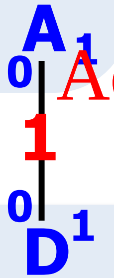
- Network partitioned
- Focus on {A, D} partition
- Suppose sequence of events:
  - D notices link failure
  - s its routing table

{C, E} between A

an

- Ho loop persist?

Dst	I/f	Metric
A	local	0
B	0	3
D	0	1
E	0	2
C	0	3



Dst	Metric
A	1
B	4
D	2
E	3
C	4

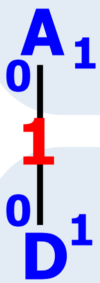
+

Dst	I/f	Metric
D	local	0
A	0	1
E	1	**Inf**
B	1	**Inf**
C	1	**Inf**

→

Dst	I/f	Metric
D	local	0
A	0	1
E	**0**	**3**
B	**0**	**4**
C	**0**	**4**

Counting to Infinity (III)



Dst	I/f	Metric
A	local	0
B	0	5
D	0	1
E	0	4
C	0	5

Dst	I/f	Metric
A	local	0
B	0	**7**
D	0	1
E	0	**6**

...

Dst	I/f	Metric
A	local	0
B	0	**Inf**
D	0	1
E	0	**Inf**
	0	**Inf**

Dst	I/f	Metric
D	local	0
A	0	1
E	0	3
B	0	4
C	0	4

Dst	I/f	Metric
D	local	0
A	0	1
E	0	**5**
B	0	**6**
C	0	**6**

...

Dst	I/f	Metric
D	local	0
A	0	1
E	0	**Inf**
B	0	**Inf**
C	0	**Inf**

- Each advertisement increments metrics for partitioned destinations by one
- Loop persists until count reaches infinity!

Agenda

- We are now ready to study Distance Vector routing

1. Algorithms
2. Pathologies
3. Optimizations
   - Split Horizon
   - Poison Reverse

Avoiding direct loops

- Bouncing and counting to infinity cause slow convergence, create loops
- Consider any link (X,Y) and destination Z
  - with X's ne Y

Split Horizon:

clearly, Y should never choose X as next hop toward Z

- Intuition: X should never announce to Y a path with short distance to Z if it's forwarding to Z via Y!

Split Horizon with Poison Reverse

- Again, consider link (X,Y), destination Z
  - with X's next hop toward Z being Y

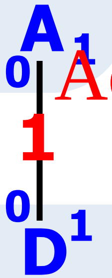
- More generall nce different routing tables
- Split horizon: don't announc destination Z on interface used as next hop toward Z!
- Poison Reverse (optional): X announces to Y its distance to Z is infinity!

Example: Split Horizon and Poison Reverse

Dst	I/f	Metric
A	local	0
B	0	3
D	0	1
E	0	2
C	0	3

- Same example as counting to infinity: {A, D} partitioned
- D detects link break, A ... ces first

p, im ... nce after one ... ent!



Dst	Metric
A	1
B	**Inf**
D	**Inf**
E	**Inf**
C	**Inf**

+

Dst	I/f	Metric
D	local	0
A	0	1
E	1	Inf
B	1	Inf
C	1	Inf

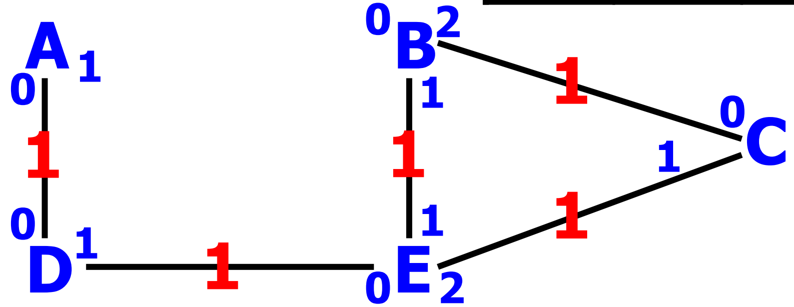
→

Dst	I/f	Metric
D	local	0
A	0	1
E	1	**Inf**
B	1	**Inf**
C	1	**Inf**

Limitations: Split Horizon and Poison Reverse

- Consider same example, but {B, C, E} partition
- Link (A, B) already failed, routing has converged
- Now link (D, E) fails
- Consider only

		Metric
		2

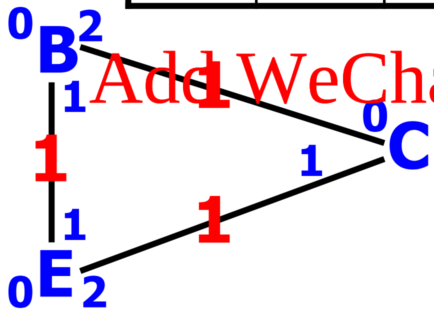


Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	0	1

Limitations (II): Split Horizon and Poison Reverse

- E notices failed link, updates local table

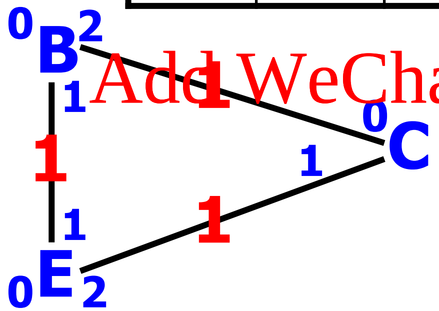


		ric
D	1	2

Dst	I/f	Metric
D	0	**Inf**

Limitations (III): Split Horizon and Poison Reverse

- E advertises its new table
  – Suppose advertisement reaches B, but not yet C



[illegible]	[illegible]	ric
D	1	2

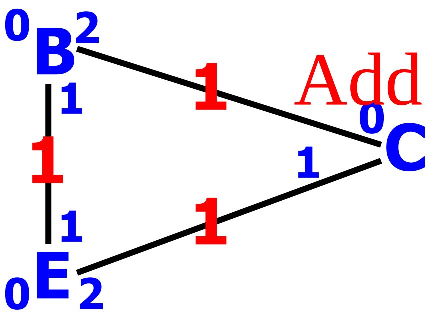
Dst	I/f	Metric
D	0	**Inf**

Limitations (IV): Split Horizon and Poison Reverse

- C advertises its table, with split horizon and poison reverse

Dst	I/f	
D	1	

Dst	I/f	Metric
D	2	**3**



Dst	I/f	
D	1	2

Dst	I/f	Metric
D	0	Inf

+

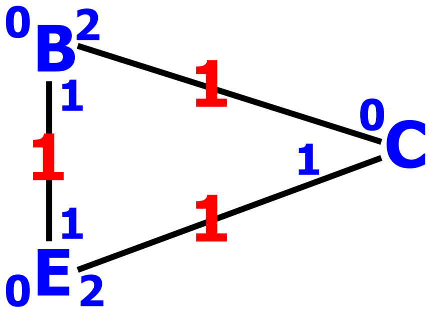
Dst	Metric
D	Inf

Dst	I/f	Metric
D	0	**Inf**

Limitations (V): Split Horizon and Poison Reverse

- B advertises its routing table, with split horizon and poison reverse
- For destination D, loop {C → E → B → C}!
  - resolved o ty

Dst	I/f	Metric
D	2	3



Dst	I/f	Metric
D	1	2

+

Dst	Metric
D	Inf

→

Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	0	Inf

+

Dst	Metric
D	4

→

Dst	I/f	Metric
D	**1**	**4**

Summary: Distance Vector routing

- DV algorithm: periodically dump routing table contents to neighbors
- Convergence: after topology change, point when routing tables

- Pros:
  - simple
  - finds correct routes after topology changes
- Cons:
  - bouncing, counting to infinity cause loops
  - slow to converge after some topology changes
  - split horizon, poison reverse only partial solutions