Routing and Forwarding Simulation on Paper

We will simulate:

- Managing routing tables using two different protocols:
  - DV - Distance vector routing (like RIP)
  - LS - Link State routing (like OSPF)

- Hop by hop forwarding
- “Ping” - Echo request, echo reply, TTL expiry

Divide into groups

- Class is divided into 5 groups: AB, CD, EF, GH, IJ
- Please do not work alone; the entire group should work together
- Each group represents a “router”
- Diagrams show the links between “routers”
  - Each group gets a different diagram, knows only what's on their diagram
  - Routing protocols should eventually learn about things that were not on the diagrams

Each group represents a “router”

- Your brains act as the software and CPU
- Large pieces of paper represent routing tables or forwarding tables
- Small pieces of paper represent messages sent or received

Thick lines on diagrams represent links

- You have a diagram showing network links from your “router” to some other “routers” in the room
- You do not (yet) know anything that's not on the diagram
- You can send or receive messages over your direct links.
- If you want to send a message to a non-neighbour, it will need hop-by-hop forwarding

Step by step instructions – Central Clocking

- All groups should work at the same pace, otherwise it gets too confusing
- Instructors will tell you what to do, step by step
- Please do not skip ahead

- Real routing protocols do not use central clocking. Instead, each router operates at its own pace with its own timers.

Reminder: Routing and forwarding

- Forwarding table lists destinations and corresponding next-hop
  - Where does forwarding table come from?
- Messages have source address, destination address, and a message body
- If destination is yourself, read message and respond
- If not addressed to you, decrement TTL and pass message on to next hop
- Discard message if destination is unknown

Distance Vector Routing Simulation

- RIP is a distance vector protocol. This exercise is a much simplified version.

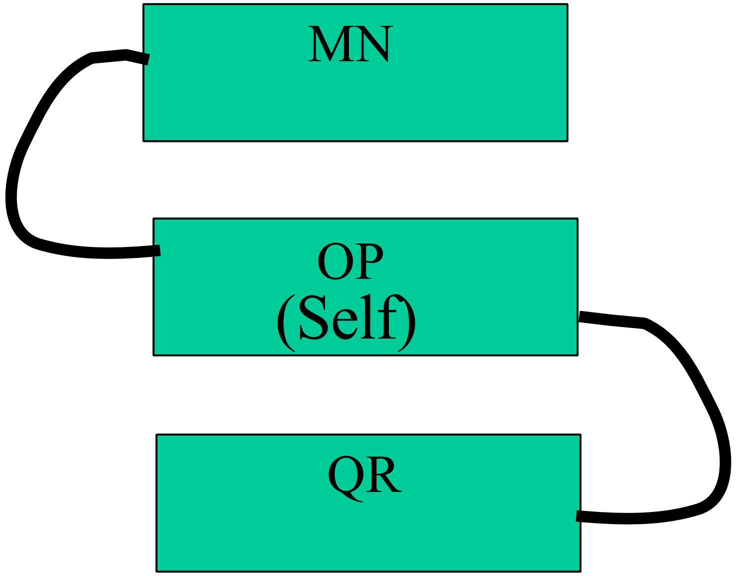
Distance Vector Routing Simulation - Initialisation

- Instructors pass out information about topology.
- Each group knows about their direct neighbours, but not distant destinations.

- Protocol will “learn” how to get to distant destinations.

DV Simulation – Examine your topology

- You have a diagram showing yourself and links to your neighbours
- You do not yet know anything else



DV Simulation - Create a distance vector routing table

- Create a distance vector table showing routes to all destinations that you know about.
- Table will contain three columns:
  - Destination
  - Next hop (yourself, or a direct neighbour, never anything else)
  - Cost (0 for yourself, 1 for your direct neighbours, more for distant destinations)
- Table will contain one row for each destination you know about.

Fill in the first version of your DV routing table

Distance Vector (DV) routing table as known by OP

Cost 1 to get to a direct neighbour

Cost 0 to get to yourself

Your name here

Time here

Time: 09:15

Destination	Next Hop	Total Cost
OP	(self)	0
MN	MN	1
QR	QR	1

Use a large sheet of paper for the routing table

DV Simulation – Prepare to send copies to neighbours

- Make copies of your distance vector table
  - but leave out the next hop. Just include the destination and cost
- You need one copy for each neighbour
- Make copies now, but do not send them to neighbours yet

Prepare updates to send to your neighbours

Message From ___OP___ to ___MN___

Type: DV routing update

Copy the “Destination” and “Total Cost” columns from your routing table. Do not copy the “Next Hop” column.

Your neighbour's name: MN

Your Time

Make a copy for each neighbour.

Time: 09:16

Destination	Total Cost
OP	0
MN	1
QR	1

Use small pieces of paper for messages

DV Simulation – Exchange DV routing updates with neighbours

- Wait for instructors to tell you to go.
- Take the update messages you made, and give them to each of your neighbours
- Expect to receive a copy of an update message from each of your neighbours

Examine the messages from your neighbours

Message From MN to OP

Type: DV routing update

Time: 09:16

This part of the message tells you which destinations are known by whoever sent the message. You do not receive their "Next Hop" information.

Your name should appear here, because the message was sent to you.

Who sent this to you. There should be one from each of your direct neighbours.

Destination	Total Cost
MN	0
OP	1
ST	1

DV Simulation - Update table using new information from neighbours

- Add 1 to the cost of everything your neighbours told you
- If there are any destinations that you did not have before, add them to your table
  - next hop is the neighbour that told you about the new destination
- If your neighbour can get to a destination for lower cost than you had before, update your table to show new cost and next hop

Add 1 to costs learned from neighbours

Message From MN to OP

Type: DV routing update

Time: 09:16

Destination	Total Cost
MN	~~0~~ 1
OP	~~1~~ 2
ST	~~1~~ 2

Add 1 to each cost. This effectively converts from “what it would cost your neighbour” to “what it would cost you if you have to forward through that neighbour”

Find new destinations or better routes

Message From ___MN___ to ___OP___

Type: DV routing update

Time: 09:16

This is a new destination. Add it to your DV routing table. Set the "Next hop" column to the name of the neighbour that told you this new information.

No change. We already know a route to "MN" with cost "1"

Destination	Total Cost
MN	~~0~~ 1
OP	~~1~~ 2
ST	~~1~~ 2

Update your DV routing table

Distance Vector (DV) routing table as known by __OP__

Add new destination learned from neighbour “MN”.

Old information does not change (at least, not the first time).

First

Repeat for other new destinations, possibly learned from other neighbours.

Time: ~~09:15~~ **09:20**

Destination	Next Hop	Total Cost
OP	(self)	0
MN	MN	1
QR	QR	1
ST	**MN**	**2**

Update the routing table on a large sheet of paper

DV Simulation - Ping

- An instructor will fill in an “Echo Request” packet with a source and destination address and TTL of their choice
- Start at the source address. Ask “I am trying to go to <destination>, what is the next hop?”
- Got to next hop. Decrement TTL. Ask again.
- Repeat until success or TTL expires
- If successful, repeat with “Echo Reply”

DV Simulation - Repeat

- Repeat the entire process once or twice more
- Observe that you learn more information the first few times
- Eventually, routing should converge, as each group learns routes to all other groups

DV Simulation – Count to infinity

- If time allows, instructors may demonstrate the “count to infinity” problem using verbal messages

DV Simulation – Differences from reality

- Real DV protocols can detect dead peers using timeouts
- Real DV protocols can delete routes
- Real DV protocols do not use central clocking
- Many other differences

Link State Routing Simulation

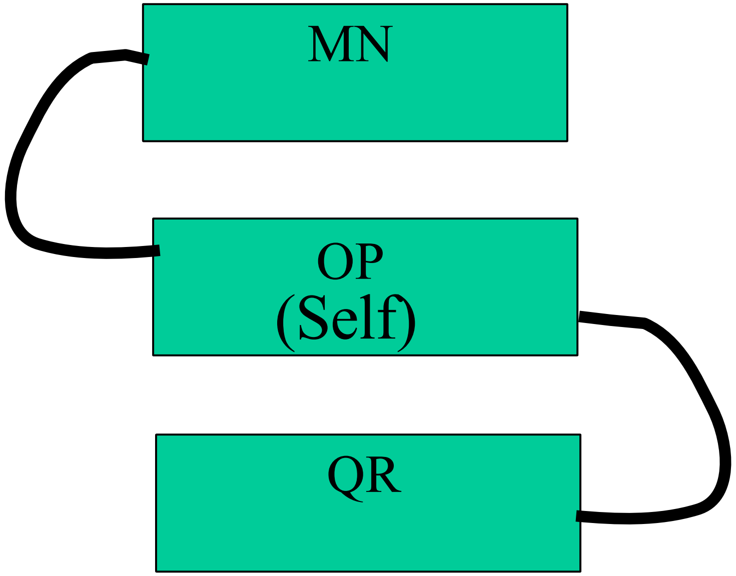
- OSPF is a link state routing protocol. This exercise is a much simplified version.

Link State Routing Simulation - Initialisation

- Forget everything from the previous exercise.
- Instructors pass out information about topology.
- Each group knows about their direct neighbours, but not distant destinations.
- Protocol will “learn” how to get to distant destinations.

DV Simulation – Examine your topology

- You have a diagram showing yourself and links to your neighbours
- You do not yet know anything else.



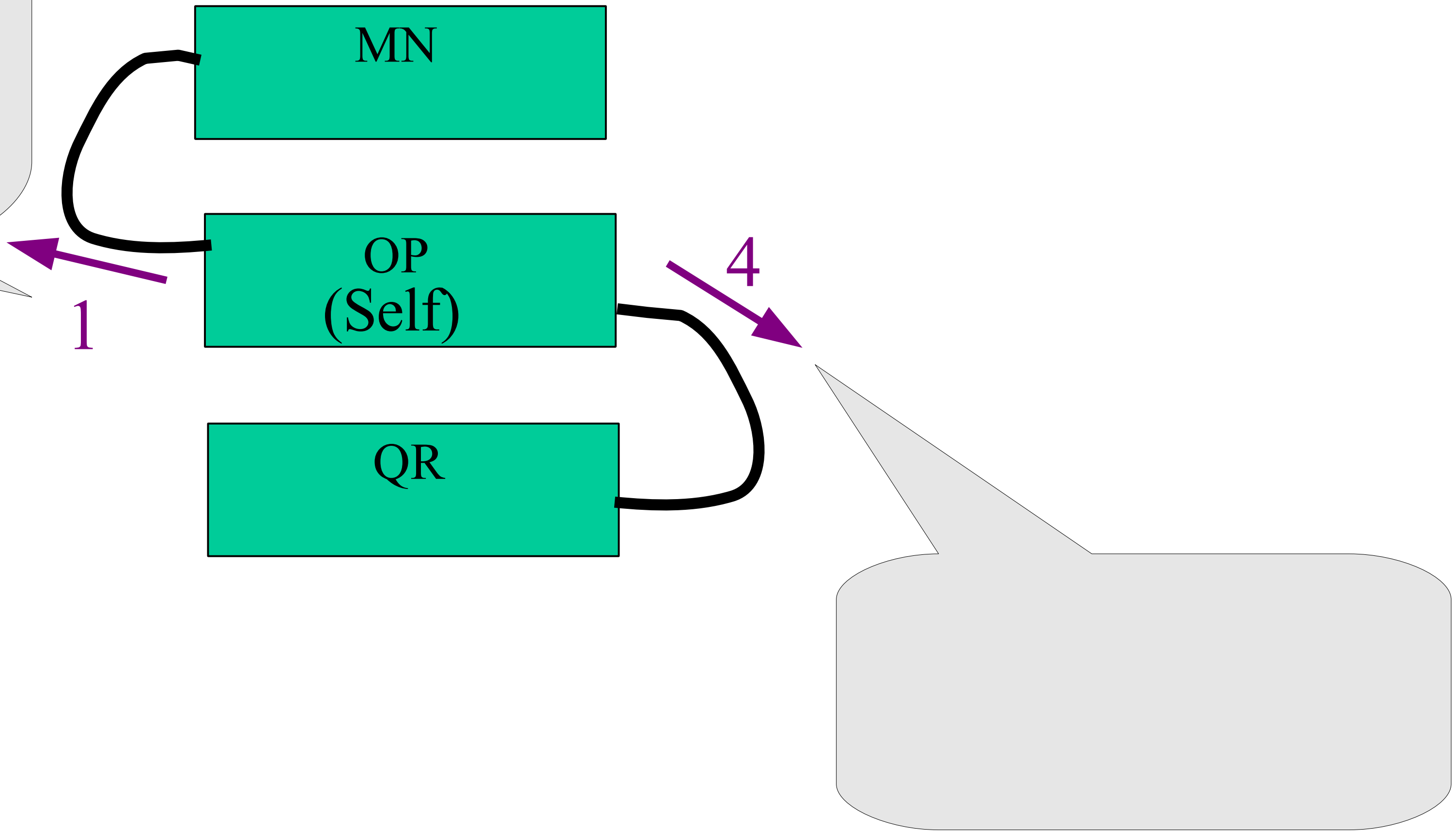
DV Simulation – Assign link costs

- Each link will have a “cost” or “distance” associated with it
  - A link may have different costs in different directions
- For each link from you to your neighbours, assign a cost (an integer from 1 to 5)
- DO NOT assign costs for the opposite direction. Your neighbours will be in charge of that.

Assign link costs – only for one direction

Cost of sending a packet across the link in the direction of the arrow (away from you, to your neighbour). Choose an integer from 1 to 5.

Similarly for all links that go from you to any directly-connected neighbours.



Update the diagram on a large sheet of paper

LS Simulation - Create a link state table

- Create a link state table showing all links that you know about
- Table will contain:
    - Link identification (PQ to RS, OP to MN, …)
        - Diagram shows your links
    - Cost of link
    - Both directions have a cost, and they might be different. “A to B” is not the same as “B to A”.
- The link state table, and the diagram with arrows, both represent exactly the same information.

Create the first version of your link state table

(LS) Table of Link States as known by OP

Cost of sending a packet over the direct link from you to your neighbour.

Repeat for all other links from you to your neighbour.

Your identity

Fill in the time

Time: 10:15

Link From-To	Link Cost
OP-MN	1
OP-QR	4

Update the link state table on a large sheet of paper

LS Simulation - Create shortest-path table

- Create a shortest-path table by analysing the link state table (or by analysing the arrows on the diagram, which represent the same information)
- For each destination that you know about, figure out the path that has the lowest cost
- Table has three columns: Destination, path to get from you to the destination, total cost to get there

Create the first version of your Shortest-Path table

(LS) Shortest-Path Table derived from Link State Table as known by OP

Set of links that have to be traversed to get to the destination. The first link in the path is also the "next hop".

Total cost to getting to the destination

Your name

The time

Time: 10:16

Destination	Shortest Path	Total Cost
OP	(self)	0
MN	OP-MN	1
QR	OP-QR	4

Update the shortest path table on a large sheet of paper

LS Simulation – Mark a checkpoint in the Link State table

- Draw a horizontal line at the end of the link state table
- Later, you will know that entries above the line are older and entries below the line are newer

Mark a checkpoint in the Link State table

(LS) Table of Link States as known by ___OP___

Draw a line here. New information will later go below the line.

Time

Time: ~~10:15~~ **10:17**

Link From-To	Link Cost
OP-MN	1
OP-QR	4

Update the link state table on a large sheet of paper

LS Simulation – Prepare the first messages to send to neighbours

- Make copies of your link state table
  - Since this is the first iteration, make exact copies; don't leave anything out
  - In the next iteration, you will send only the changes, with the help of the checkpoints
- You need one copy for each neighbour
- Make copies now, but do not send them to neighbours yet

Prepare the first updates to send to neighbours

The first message to each neighbour contains a complete copy of your link state table. Do not leave anything out. Do not accidentally use the shortest path table.

Your neighbour's name.

Make a copy for each neighbour.

Your Time

Message From OP to MN

Type: LS routing update

Time: 10:18

Link From-To	Link Cost
OP-MN	1
OP-QR	4

Use small pieces of paper for messages

LS Simulation – Exchange LS routing updates with neighbours

- Wait for instructors to tell you to go.
- Take the update messages you made, and give them to each of your neighbours
- Expect to receive a copy of an update message from each of your neighbours

Examine the messages from your neighbours

Message From __MN__ to __OP__

Type: LS routing update

This part of the message tells [illegible] [illegible] will [illegible] information into your own link state table.

Time: 10:18

Link From-To	Link Cost
MN-OP	2
MN-ST	1

LS Simulation - Update link-state table

- Merge the link states that your neighbours send you with those you already have
- Add any new links to your table
- If the costs for any old links have changed, update the cost in your table
- A real routing protocol would also have a way of reporting links that go down

Merge updates into Link State table

(LS) Table of Link States as known by OP

Add all the new information just learned from your neighbours.

New information goes below the line that you drew earlier.

“MN-OP” is not the same as “OP-MN”.

Time: ~~10:15~~ ~~10:17~~ **10:19**

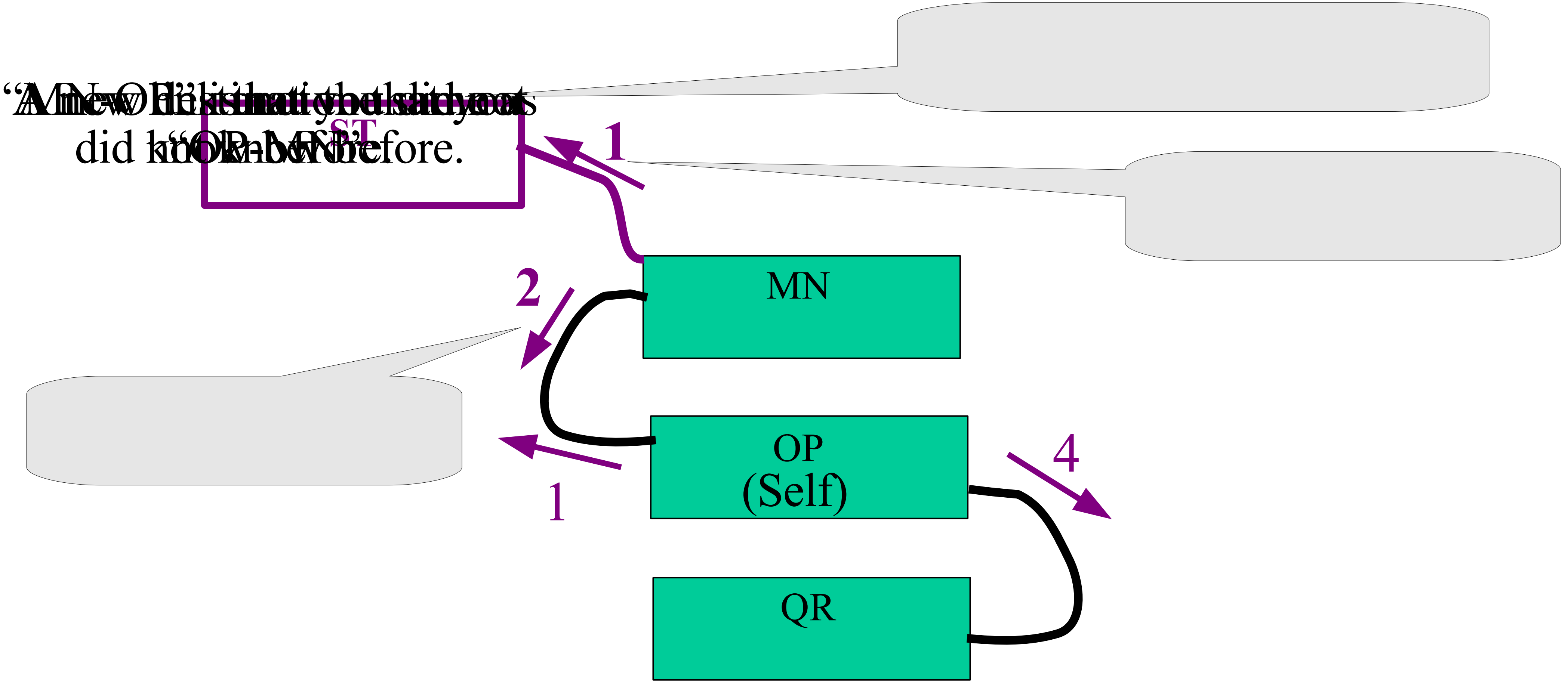
Link From-To	Link Cost
OP-MN	1
OP-QR	4
MN-OP	**2**
MN-ST	**1**

Update the link state table on a large sheet of paper

LS Simulation - Use link-state table to update diagram

- Your link-state table tells you about all the links you know about.
- Different directions are treated like different links. “A to B” is not the same as “B to A”.
- Your network diagram contains exactly the same information, just in a different format.
- Update the diagram using the table.

Update diagram from link state table



Update the diagram on a large sheet of paper

LS Simulation - Use link-state table to make shortest-path table

- Your link-state table or your network diagram tells you about all the links you know about.
- Different directions are treated like different links. “A to B” is not the same as “B to A”.
- There will often be several ways to get to a destination. Choose the path with the lowest total cost. Use the diagram to help you.
- Make a table showing all destinations, how to get there, total cost.

Update your Shortest-Path table

(LS) Shortest-Path Table derived from Link State Table as known by OP

Total cost is cost of "OP-MN" plus cost of MN-ST

The best way to get to "ST"

A new Destination

Time: ~~10:16~~ **10:20**

Destination	Shortest Path	Total Cost
OP	(self)	0
MN	OP-MN	1
QR	OP-QR	4
ST	**OP-MN-ST**	**2**

Update the shortest path table on a large sheet of paper

LS Simulation - Ping

- An instructor will fill in an “Echo Request” packet with a source and destination address and TTL of their choice
- Start at the source address. Ask “I am trying to go to <destination>, what is the next hop?”
- Got to next hop. Decrement TTL. Ask again.
- Repeat until success or TTL expires
- If successful, repeat with “Echo Reply”

LS Simulation - Repeat

- Whenever anything changes, send a copy of the changes to each neighbour
  - changes can be new links, changes in cost, or deleted links (link failure or death of peer)
  - Use the horizontal line “checkpoints” to keep track of old versus new information.
  - Also update all your tables and diagrams whenever anything changes.
- Eventually, routing should converge
  - all groups should have identical link state tables, identical diagrams, but different shortest path tables

LS Simulation – Differences from reality

- Real LS protocols send updates almost instantly
- Real LS protocols can detect dead peers and dead links using timers or direct measurements
- Many other differences

NOTE TO INSTRUCTORS

- Instructors know the complete topology.
- Students all get different diagrams, showing only their own direct neighbours, not showing more distant topology.
- When printing these notes, remember that everything after this page needs special treatment.
  - From page 1 to just before this page, treat it like a normal presentation. That probably implies printing in 6-up layout, with one copy per student.
  - This page is not printed at all.
  - Each group of students will need about 10 copies of the DV and LS routing update messages, and 1 or 2 copies of the echo request, echo reply and unreachable message templates. You can make multiple copies of the same page in the powerpoint presentation and then print 6-up.
  - Each group of students will need 1 large copy of their own partial topology, and 1 or 2 large copies of the routing tables.
  - Instructors will need 1 or 2 large copies of the complete topology.

Distance Vector (DV) routing table as known by _______

Time: ________

Destination	Next Hop	Total Cost

Instructions: Start with yourself and your neighbours. When you get updates from neighbours, update this table.

(LS) Table of Link States
as known by _______

Time: ________

Link From-To	Link Cost

Instructions:
1. Start by filling in costs of all directly-connected links.
2. When you get updates from neighbours, update this table.

(LS) Shortest-Path Table derived from Link State Table as known by _______

Time: _________

Destination	Shortest Path	Total Cost

Instructions: Use your link-state table (or diagram with arrows) to figure out the shortest path to each destination. Update this shortest-path table when something changes.

Message From _______ to _______
Type: DV routing update

Time: ________

Destination	Total Cost

Instructions for Sender:
1. Fill in source, destination, time.
2. Copy your DV routing table, but leave out the “Next Hop” column.
3. Send to your neighbour.

Instructions for Receiver:
1. Add 1 to all costs.
2. If there are any costs lower than before, update cost and Next Hop in your table.
3. Similarly for any new destinations.

Message From _______ to _______
Type: LS routing update

Time: ________

Link From-To	Link Cost

Instructions for Sender:
1. Fill in source, destination, time.
2. Copy your LS routing table; do not leave out anything.
3. Send to your neighbour.

Instructions for Receiver:
1. If any costs have changed for links that you already knew, update your routing table.
3. Similarly for any new links.

Message From _______ to _______
Type: Echo Request

TTL: ______ Request ID: _________

Instructions for Original Sender:

1. Fill in source, destination, TTL, ID.
2. Consult your routing table to choose next hop.
3. Send to next hop.

Instructions for Receiver:

1. If message is addressed to you, send back Echo Reply.
2. If message is not addressed to you: Decrement TTL,
3. If TTL is zero, send back Error: TTL Exceeded.
4. Consult your routing table to choose next hop.
5. If destination is unknown, send back Error: Host Unreachable
6. Send message to next hop.

Message From _______ to _______
Type: Echo Request

TTL: ______ Request ID: _________

Instructions for Original Sender:

1. Fill in source, destination, TTL, ID.
2. Consult your routing table to choose next hop.
3. Send to next hop.

Instructions for Receiver:

1. If message is addressed to you, send back Echo Reply.
2. If message is not addressed to you: Decrement TTL,
3. If TTL is zero, send back Error: TTL Exceeded.
4. Consult your routing table to choose next hop.
5. If destination is unknown, send back Error: Host Unreachable
6. Send message to next hop.

Message From _______ to _______
Type: Echo Reply

TTL: ______ Original ID: _________

Instructions for Original Sender:

1. Fill in source, destination, TTL, Original ID (copied from request).
2. Consult your routing table to choose next hop.
3. Send to next hop.

Instructions for Receiver:

1. If message is addressed to you: You received a reply! Congratulations!
2. If message is not addressed to you: Decrement TTL,
3. If TTL is zero, discard packet. Do not send error message.
4. Consult your routing table to choose next hop.
5. If destination is unknown, discard packet. Do not send error message.
6. Send message to next hop.

Message From ________ to ________
Type: Echo Reply

TTL: ______ Original ID: _________

Instructions for Original Sender:

1. Fill in source, destination, TTL, Original ID (copied from request).
2. Consult your routing table to choose next hop.
3. Send to next hop.

Instructions for Receiver:

1. If message is addressed to you: You received a reply! Congratulations!
2. If message is not addressed to you: Decrement TTL,
3. If TTL is zero, discard packet. Do not send error message.
4. Consult your routing table to choose next hop.
5. If destination is unknown, discard packet. Do not send error message.
6. Send message to next hop.

Message From _______ to _______
Error Type: ______________

TTL: ______ Orig ID : ______Orig Dest : ______

Instructions for Original Sender:

1. Fill in source, destination, TTL, Original ID, Original Destination.
2. Consult your routing table to choose next hop.
3. Send to next hop.

Instructions for Receiver:

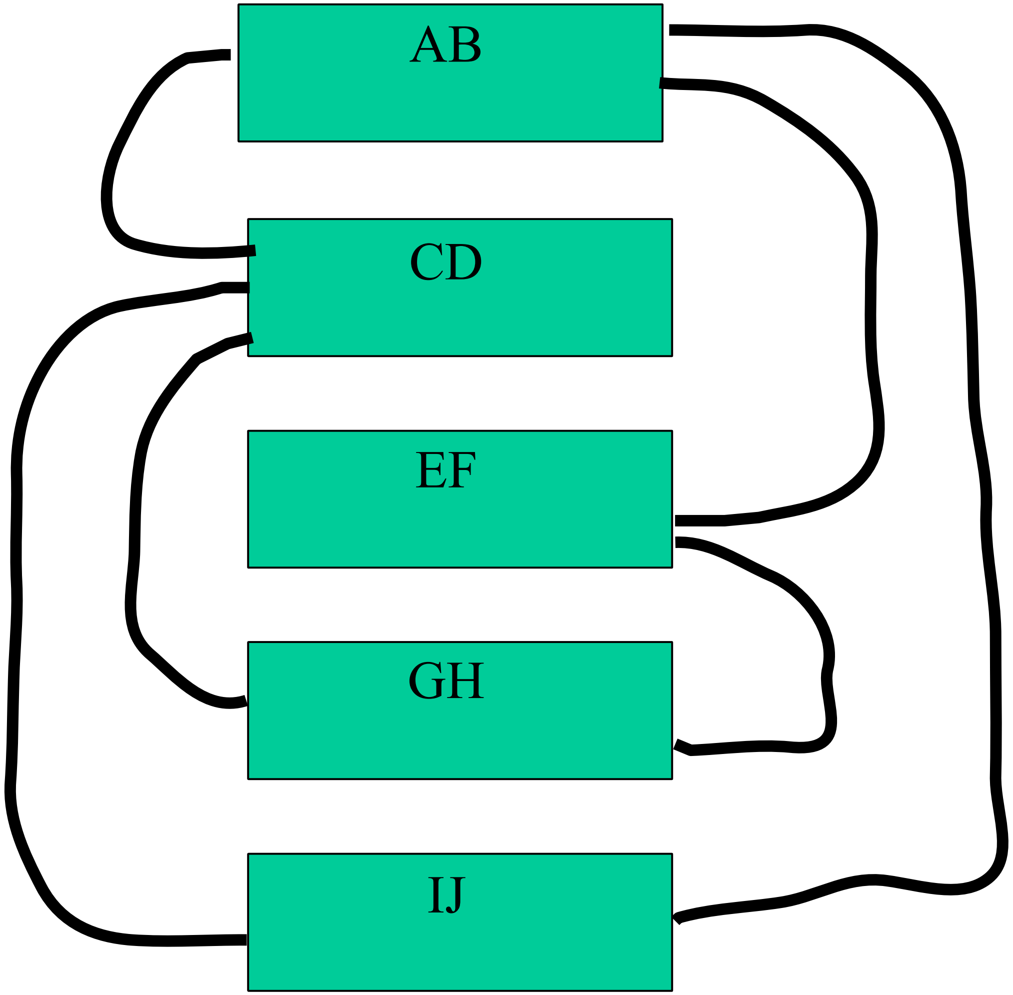
1. If message is addressed to you: There was an error! Sorry!
2. If message is not addressed to you: Decrement TTL,
3. If TTL is zero, discard packet. Do not send error message.
4. Consult your routing table to choose next hop.
5. If destination is unknown, discard packet. Do not send error message.
6. Send message to next hop.

Message From _______ to _______
Type: ______________

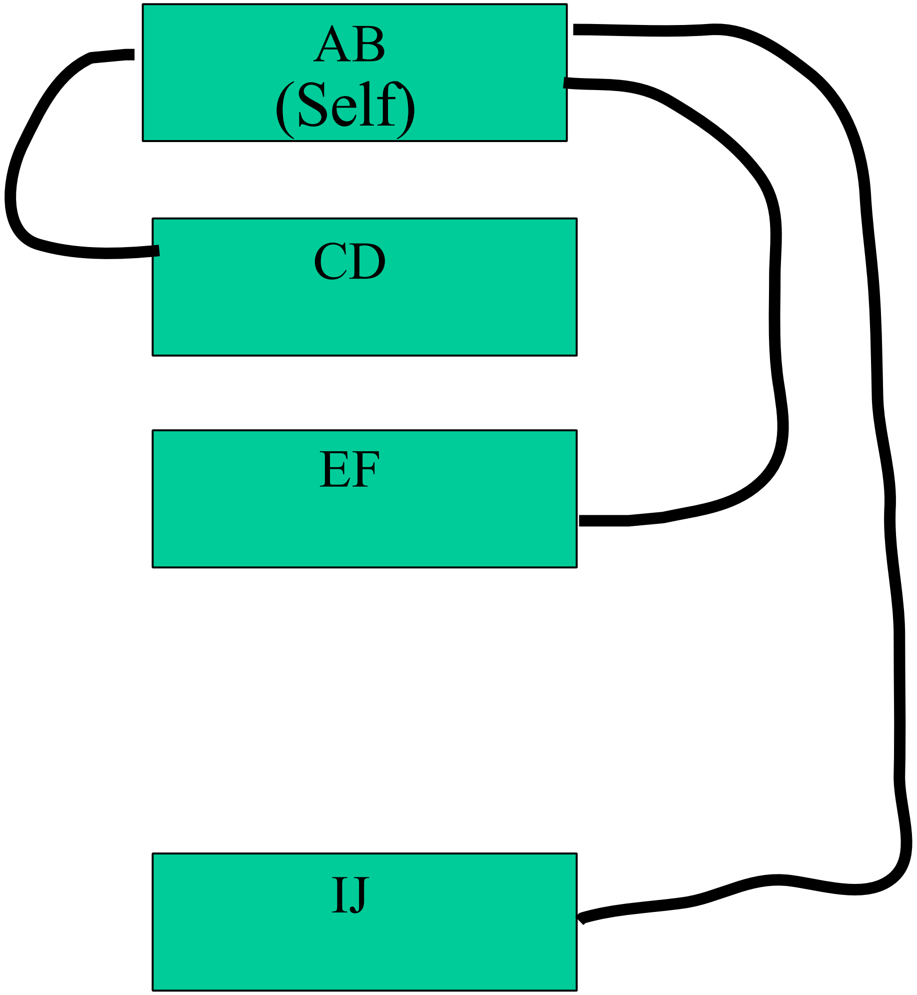
TTL: ______

Message:

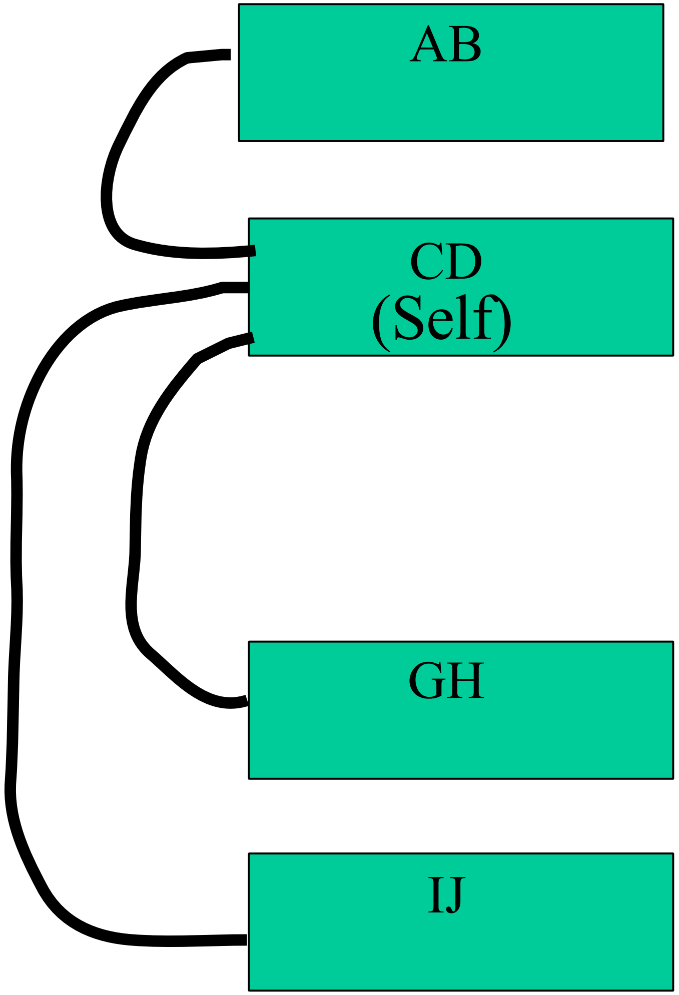
Complete Topology as known to instructors



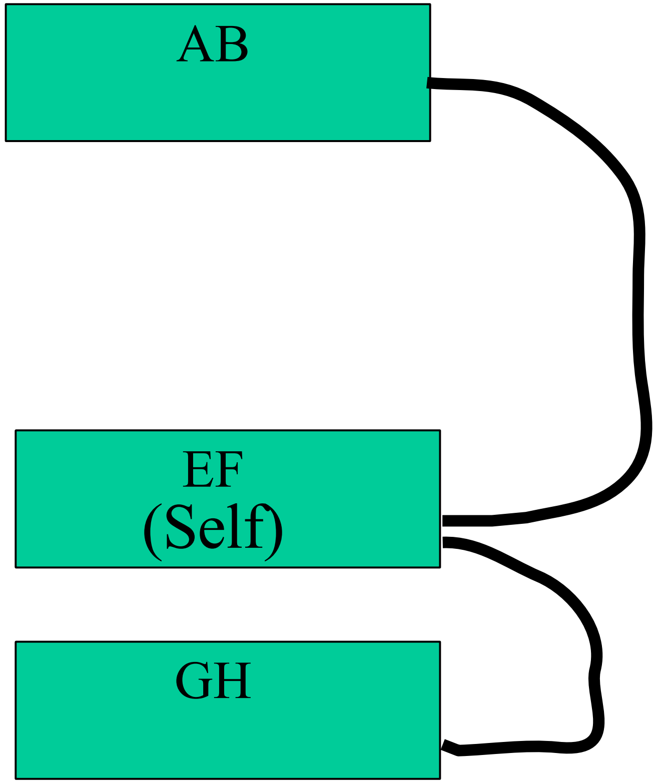
Partial Topology as known to group AB



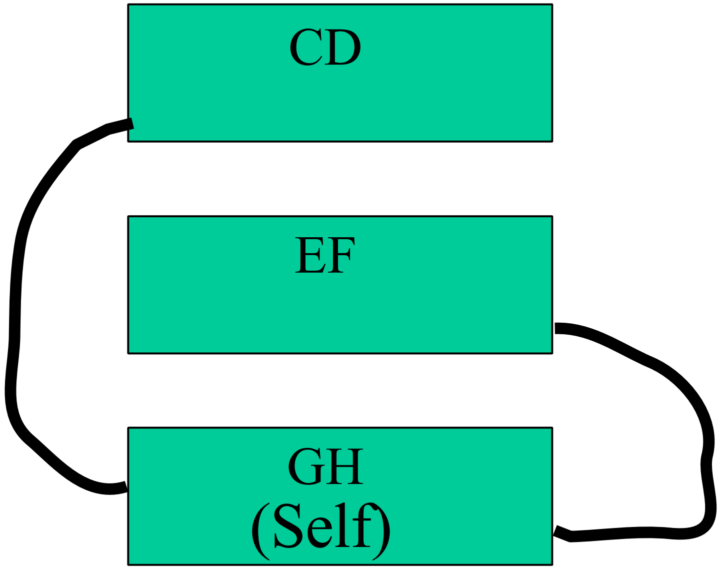
Partial Topology as known to group CD



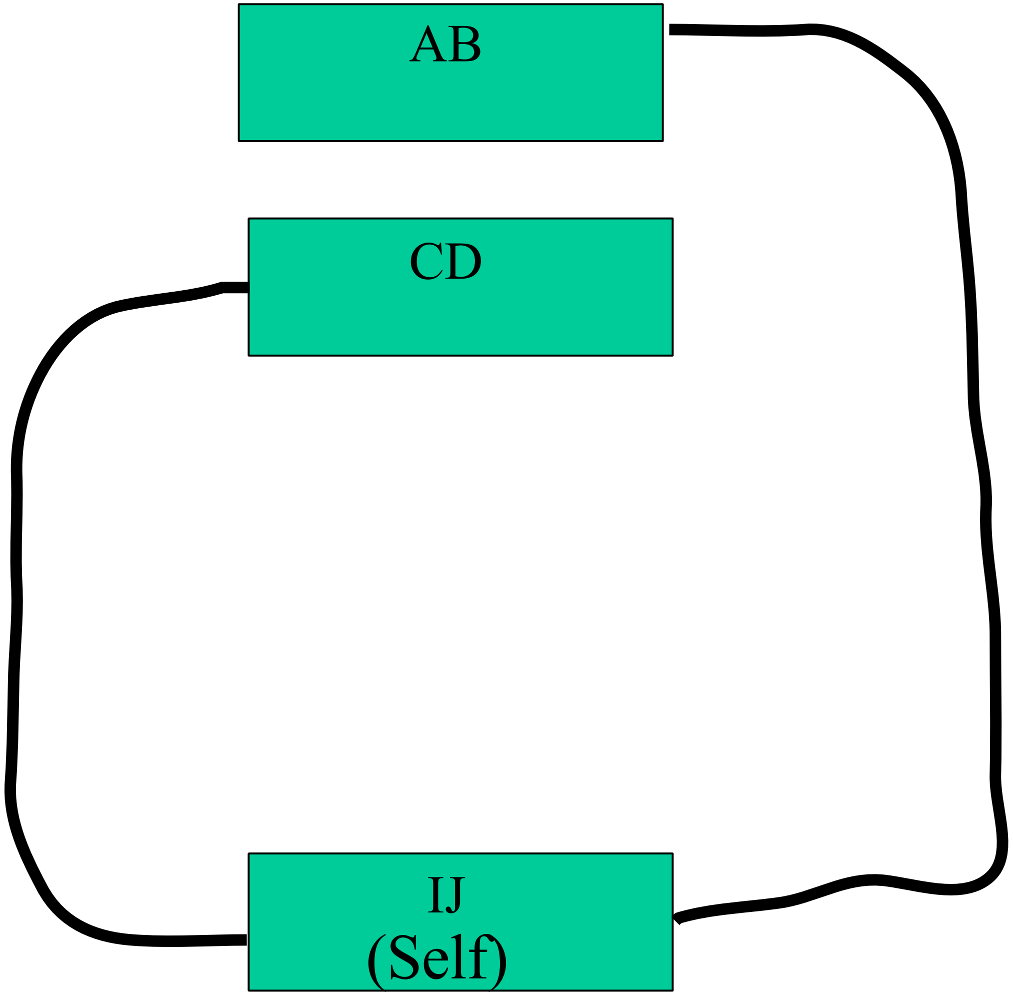
Partial Topology as known to group EF



Partial Topology as known to group GH



Partial Topology as known to group IJ


AB
CD
IJ
(Self)