Routing and Forwarding Simulation on Paper

We will simulate:

- Managing routing tables using 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

Link State Routing Simulation

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

- “Link state” protocols exchange information about the state (up, down, cost) of each link throughout the network.

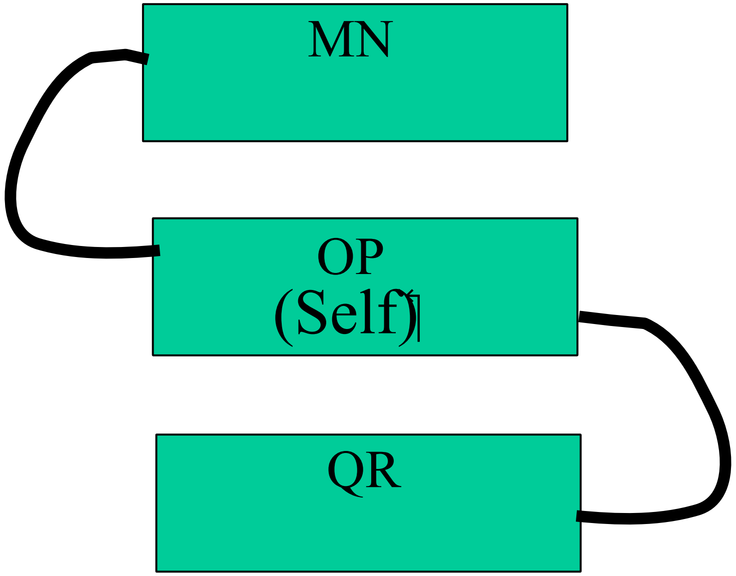
Link State Routing Simulation - Initialisation

- 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.

LS Simulation – Examine your topology

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

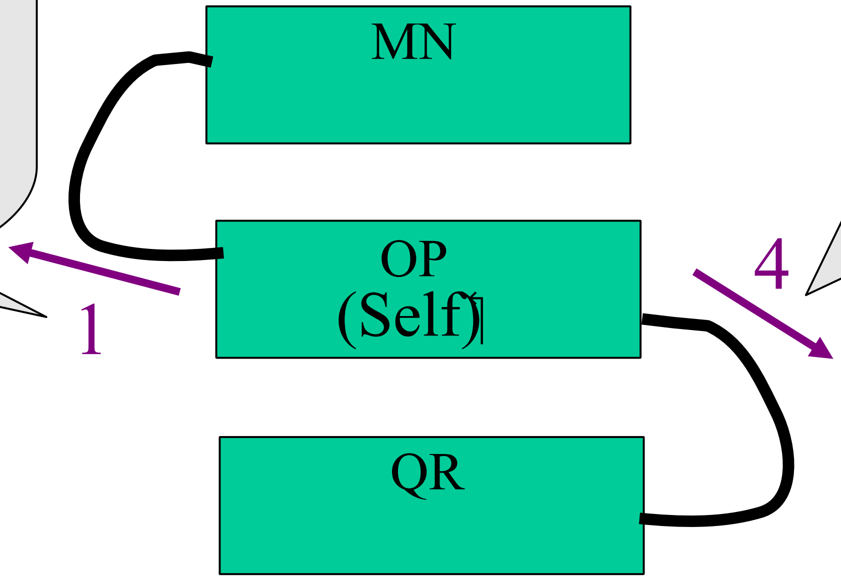


LS Simulation – Assign link costs

- 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 (in the direction going away from you), 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 of your directly-connected neighbours.

Write small to leave space for many more blocks and arrows to be added later.

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

Your name here

Time: 10:15

Time

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

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

Repeat for all other links from yourself to all your neighbours.

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

Your name here

Time: 10:16

Time

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

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

Sum of the costs of using all necessary links (in the correct direction) to get to the destination.

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

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

Time

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

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

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

Message From __OP__ to __MN__

Type: LS routing update

Time: 10:18

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

Time

Your name here

Make one copy for each neighbour.

All copies are identical except for the neighbour's name here.

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.

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

Time: 10:18

Time

Who sent this to you? It should be from one of your direct neighbours.

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

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

This part of the message tells you about all the links that your neighbour knows. You will merge this information into your own link state table.

You should have received one message for each neighbour (and sent one to each).

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

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

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

Time

New information goes below the line that you drew earlier.

Add all the new information just learned from your neighbours.

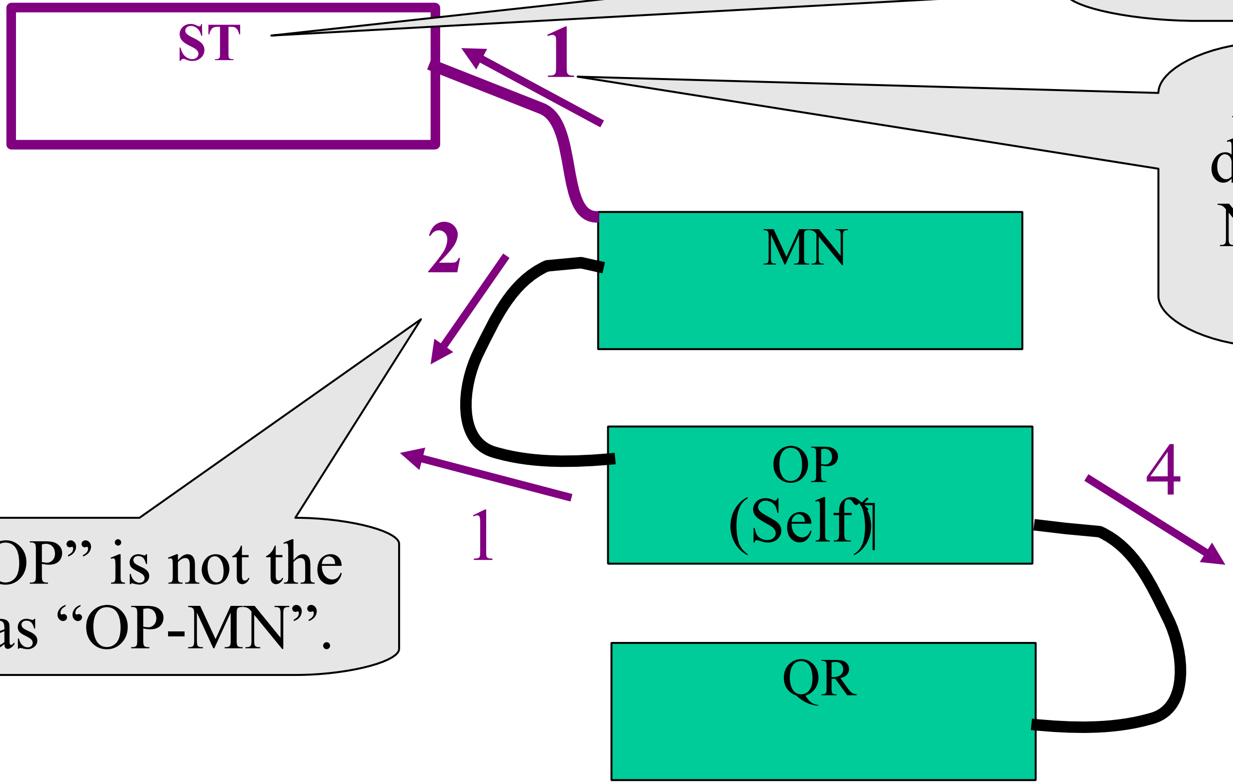
“MN-OP” is not the same as “OP-MN”.

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



A new destination that you did not know before.

A new link that you did not know before. Note the direction of the arrow.

“MN-OP” is not the same as “OP-MN”.

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

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

Time

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

A new destination that you did not know before.

The best way to get to “ST”

Total cost is cost of “OP-MN” plus cost of “MN-ST”

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