Dr SABER Takfarinas
takfarinas.saber@ucd.ie

COMP2009J
Computer Networks

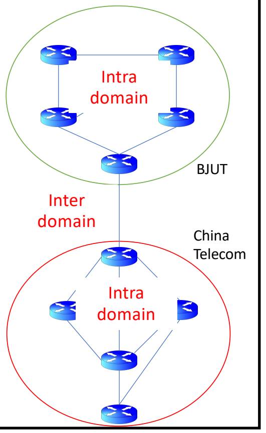
Dynamic Routing

IP Routing

- There are two approaches for calculating the routing tables
1. Static Routing
   - by hand
2. Dynamic Routing
   - automatically calculated by a routing protocol

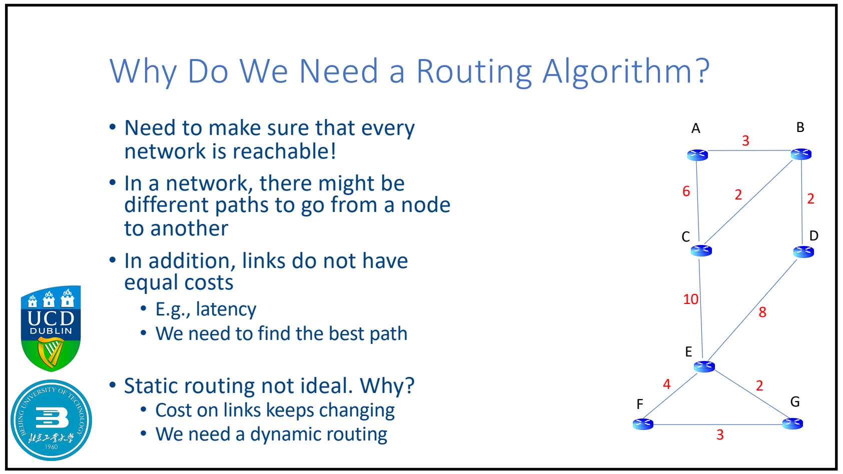
Autonomous Systems

- An **autonomous system** is a region of the Internet that is administered by a single entity
- E.g.,
  - Campus network
  - Backbone network
  - National Internet Service Provider
- Routing is done differently:
  - **intradomain routing:** within an autonomous system
  - **interdomain routing:** between autonomous system



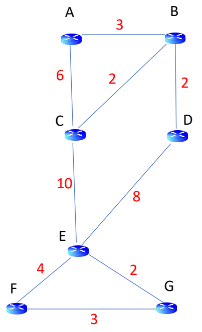
Interdomain vs Intradomain Routing

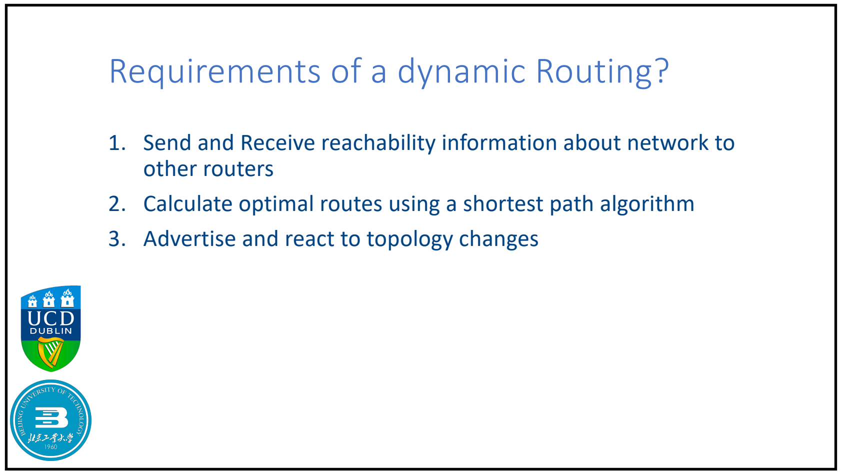
- **Intradomain Routing**
  - Routing within an AS
  - Ignores the Internet outside the AS
  - Protocols for Intradomain routing are also called **Interior Gateway Protocols** or **IGP's**
  - Popular protocols: RIP (simple, old), OSPF (better)
- **Interdomain Routing**
  - Routing between AS's
  - Assumes that the Internet consists of a collection of interconnected AS's
  - Normally, there is one dedicated router in each AS that handles interdomain traffic
  - Protocols for interdomain routing are also called **Exterior Gateway Protocols** or **EGP's**
  - Popular protocols: EGP, BGP (more recent)



Why Do We Need a Routing Algorithm?

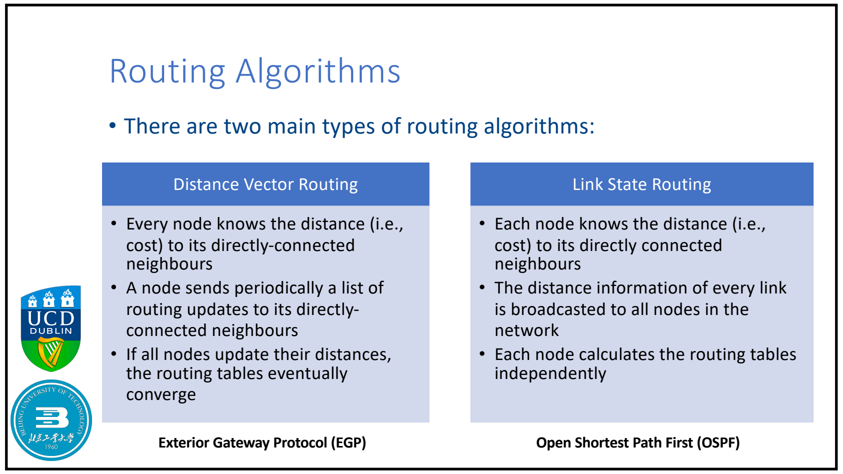
- Need to make sure that every network is reachable!
- In a network, there might be different paths to go from a node to another
- In addition, links do not have equal costs
  - E.g., latency
  - We need to find the best path
- Static routing not ideal. Why?
  - Cost on links keeps changing
  - We need a dynamic routing





Requirements of a dynamic Routing?

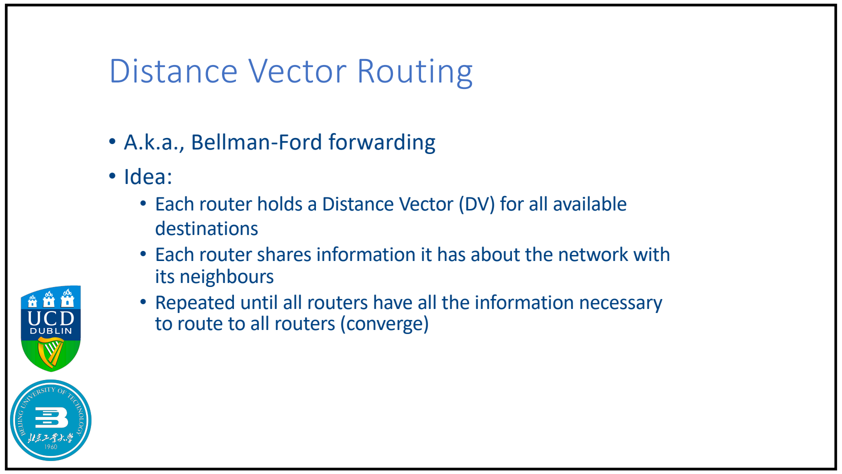
1. Send and Receive reachability information about network to other routers
2. Calculate optimal routes using a shortest path algorithm
3. Advertise and react to topology changes



Routing Algorithms

- There are two main types of routing algorithms:

Distance Vector Routing	Link State Routing
• Every node knows the distance (i.e., cost) to its directly-connected neighbours	• Each node knows the distance (i.e., cost) to its directly connected neighbours
• A node sends periodically a list of routing updates to its directly-connected neighbours	• The distance information of every link is broadcasted to all nodes in the network
• If all nodes update their distances, the routing tables eventually converge	• Each node calculates the routing tables independently
Exterior Gateway Protocol (EGP)	**Open Shortest Path First (OSPF)**



Distance Vector Routing

- A.k.a., Bellman-Ford forwarding
- Idea:
  - Each router holds a Distance Vector (DV) for all available destinations
  - Each router shares information it has about the network with its neighbours
  - Repeated until all routers have all the information necessary to route to all routers (converge)

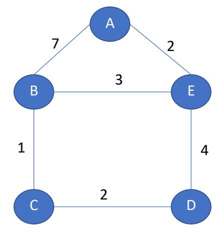
When to Update Tables in Distance Vector Routing

1. **Periodic Updates:** every time period (e.g., every 90s)
2. **Triggered Updates:** If a metric changes on a link, a router immediately sends out an update without waiting for the end of the update period
3. **Full Routing Table Update**: send the entire routing table to the neighbours (not only entries which change).
4. **Route invalidation timers:** Routing table entries are invalid if they are not refreshed for a certain number of updates (e.g., no updates received after 6 update periods)

Example

Initial State: every node knows the distance to his direct neighbours

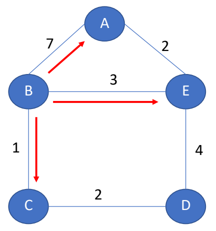
	A	B	C	D	E
A	0	7	∞	∞	2
B	7	0	1	∞	3
C	∞	1	0	2	∞
D	∞	∞	2	0	4
E	2	3	∞	4	0



Example

B sends its distance vector to its neighbours: **A**, **E**, **C**

	A	B	C	D	E
A	0	7	8	∞	2
B	7	0	1	∞	3
C	∞	1	0	2	∞
D	∞	∞	2	0	4
E	2	3	∞	4	0



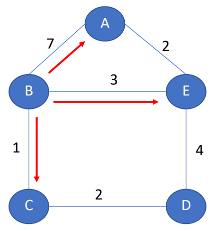
A receives **B**'s distance vector	7	0	1	∞	3
Adds the distance between **A** and **B** (i.e., 7)	14	7	7	∞	10
Compares the results to its current distance vector	0	7	∞	∞	2
Update distances with shorter ones if they exist	0	7	8	∞	2

A changed his distance vector → **A** has to inform his neighbours

Example

B sends its distance vector to its neighbours: **A**, **E**, **C**

	A	B	C	D	E
A	0	7	8	∞	2
B	7	0	1	∞	3
C	∞	1	0	2	∞
D	∞	∞	2	0	4
E	2	3	4	4	0

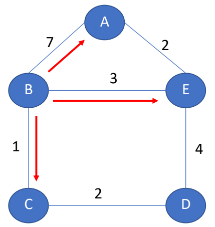


E receives **B**'s distance vector	7	0	1	∞	3
Adds the distance between **E** and **B** (i.e., 3)	10	3	4	∞	6
Compares the results to its current distance vector	2	3	∞	4	0
Update distances with shorter ones if they exist	2	3	4	4	0

E changed his distance vector → **E** has to inform his neighbours

Example

B sends its distance vector to its neighbours: **A**, **E**, **C**



	A	**B**	**C**	**D**	**E**
A	0	7	8	∞	2
B	7	0	1	∞	3
C	8	1	0	2	4
D	∞	∞	2	0	4
E	2	3	4	4	0

C receives **B**'s distance vector

7	0	1	∞	3

Adds the distance between **C** and **B** (i.e., 1)

8	1	2	∞	4

Compares the results to its current distance vector

∞	1	0	2	∞

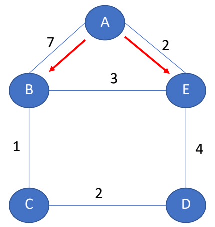
Update distances with shorter ones if they exist

8	1	0	2	4

C changed his distance vector → **C** has to inform his neighbours

Example

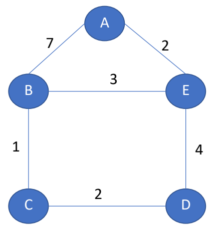
A sends its distance vector to its neighbours: **B**, **E**



	A	**B**	**C**	**D**	**E**
A	0	7	8	∞	2
B	7	0	1	∞	3
C	8	1	0	2	4
D	∞	∞	2	0	4
E	2	3	4	4	0

We keep repeating this until convergence

	A	**B**	**C**	**D**	**E**
A	0	5	6	6	2
B	5	0	1	3	3
C	6	1	0	2	4
D	6	3	2	0	4
E	2	3	4	4	0



We repeat this processes until no change in the distance vector of all nodes

DONE

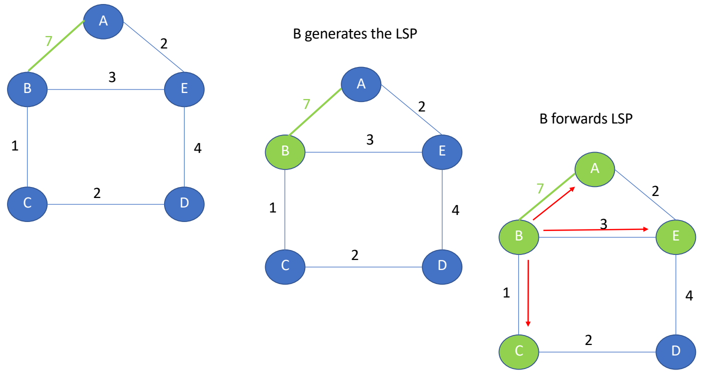
Link State Routing

- Each router shares information about its neighbours with the rest of the network
- Each router stores the complete topology of the network
- All routers on the network store the same information
- Each router uses an algorithm to solve the best path using:
  - Dijkstra's algorithm
- "Flood" link information throughout the network using Link State Packets (LSPs)
- Convergence occurs when all routers have received a LSP from each other router in the network

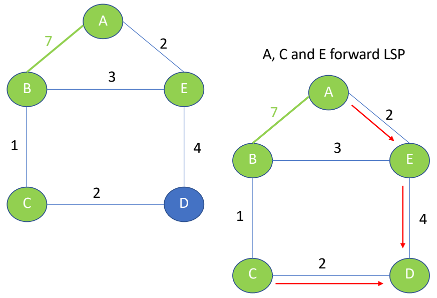
- Reliable flooding
  - Each router transmits a Link State Packet (LSP) on all links
  - A neighboring router forwards out all links except incoming
    - Keep a copy locally; don't forward previously-seen LSPs
- Challenges
  - Packet loss
  - Out-of-order arrival
- Solutions
  - Acknowledgments and retransmissions
  - Sequence numbers
  - Time-to-live for each packet

nks

Example of LSP Broadcasting



Example of LSP Broadcasting



E ignore LSP received from A as already received

D received the same LSP from C and E, so it ignores one of them.

Done with this LSP

We have to do the same with all links

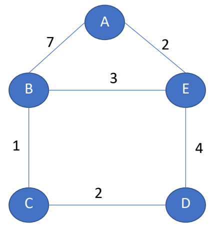
When to Flood ?

- Periodically:
  - For example: every 30 minutes
- After a topology change
  - Link or node failure
  - Modification in link metric

After Sending all LSPs

- All nodes should have the same link-state database
- Each node calculates the shortest paths
  - Using Dijkstra Algorithm
- Forwards its packets on the shortest path

Dijkstra Algorithm



Link State Database
at node C

	B	E
A	7	2

	A	C	E
B	7	1	3

	B	D
C	1	2

	C	E
D	2	4

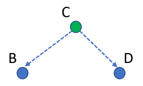
	A	B	D
E	2	3	4

Shortest Paths From C

Add distance label to each node

Take the node with the smallest distance, from reachable nodes

Explore the outgoing link



	B	E
A=∞	7	2

	A	C	E
B=∞ 1	7	1	3

	B	D
C=0	1	2

	C	E
D=∞ 1	2	4

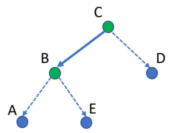
	A	B	D
E=∞	2	3	4

Shortest Paths From C

Add distance label to each node

Take the node with the smallest distance, from reachable nodes

Explore the outgoing link



	B	E
A=∞ 8	7	2

	A	C	E
B= 1	7	1	3

	B	D
C=0	1	2

	C	E
D= 2	2	4

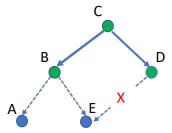
	A	B	D
E=∞ 4	2	3	4

Shortest Paths From C

Add distance label to each node

Take the node with the smallest distance, from reachable nodes

Explore the outgoing link



	B	E
A=8	7	2

	A	C	E
B=1	7	1	3

	B	D
C=0	1	2

	C	E
D: 2	2	4

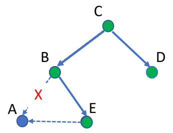
	A	B	D
E=4 6	2	3	4

Shortest Paths From C

Add distance label to each node

Take the node with the smallest distance, from reachable nodes

Explore the outgoing link



	B	E
A=8 6	7	2

	A	C	E
B=1	7	1	3

	B	D
C=0	1	2

	C	E
D=2	2	4

	A	B	D
E= 4	2	3	4

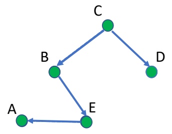
Shortest Paths From C

Add distance label to each node

Take the node with the smallest distance, from reachable nodes

Explore the outgoing link

No more nodes to visit.
DONE!



	B	E
A= 6	7	2

	A	C	E
B=1	7	1	3

	B	D
C=0	1	2

	C	E
D=2	2	4

	A	B	D
E=4	2	3	4