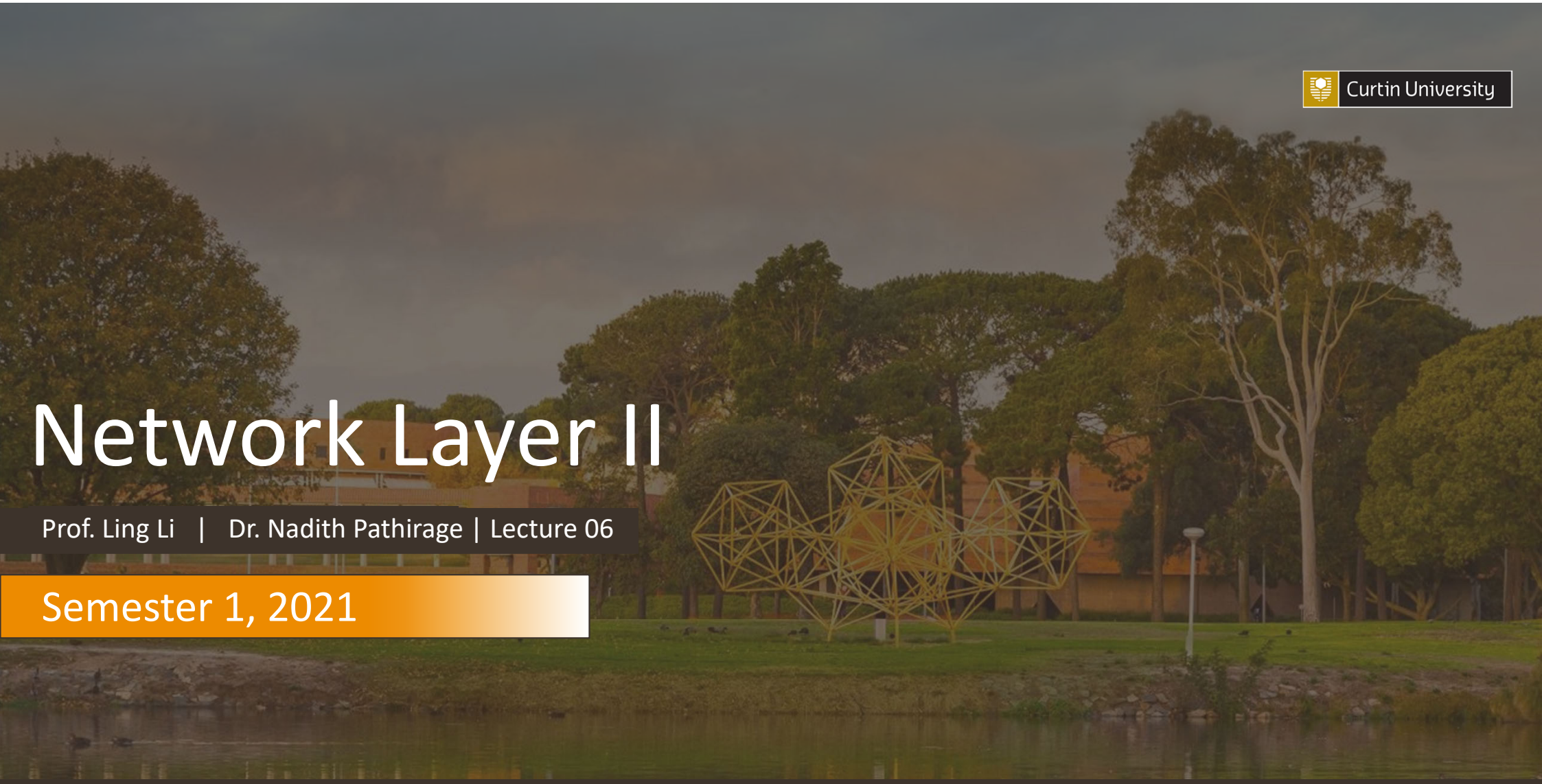


Curtin University

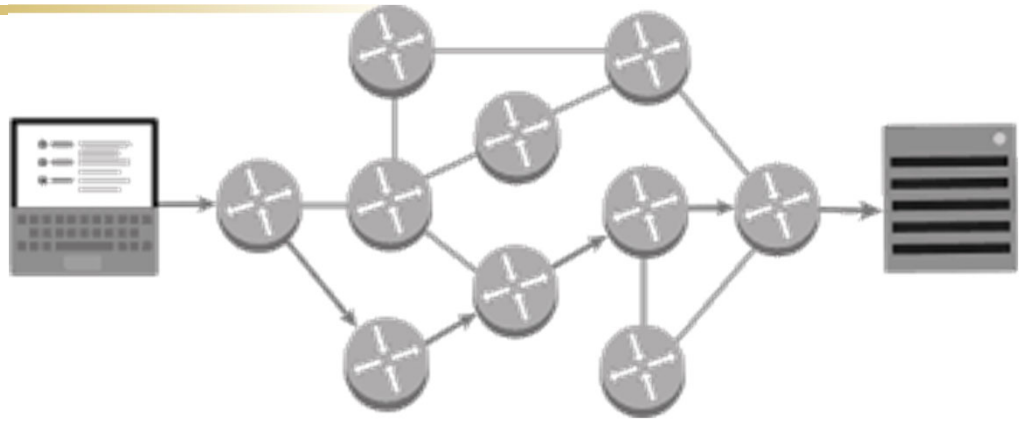
Network Layer II

Prof. Ling Li | Dr. Nadith Pathirage | Lecture 06

Semester 1, 2021

Network Layer: Routing

Curtin University



Forwarding packet towards
destination network/host

Shortest path, load balancing, etc.

Routing Classification

Curtin University

Adaptive (Dynamic) Routing

- Changes routes dynamically
  - Periodically
  - When load changes
  - When topology changes
- Gather information runtime
  - Locally
  - From adjacent routers
  - From all other routers
- Uses routing protocols
  - RIP, OSPF, IGRP

Non-Adaptive (Static) Routing

- Performed manually
- Choice of route is computed in advance, offline & downloaded to the routers when network is booted

Router: Algorithm & Forwarding

Curtin University

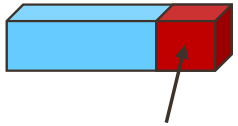
determines **end-end-path through network**

determines **local forwarding at this router**

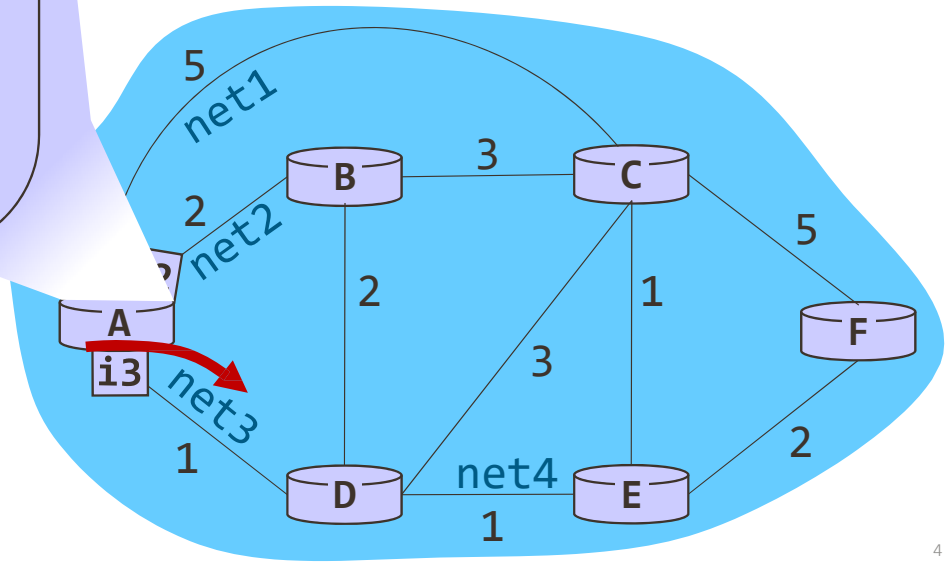
Routing Algorithm

Local Forwarding Table

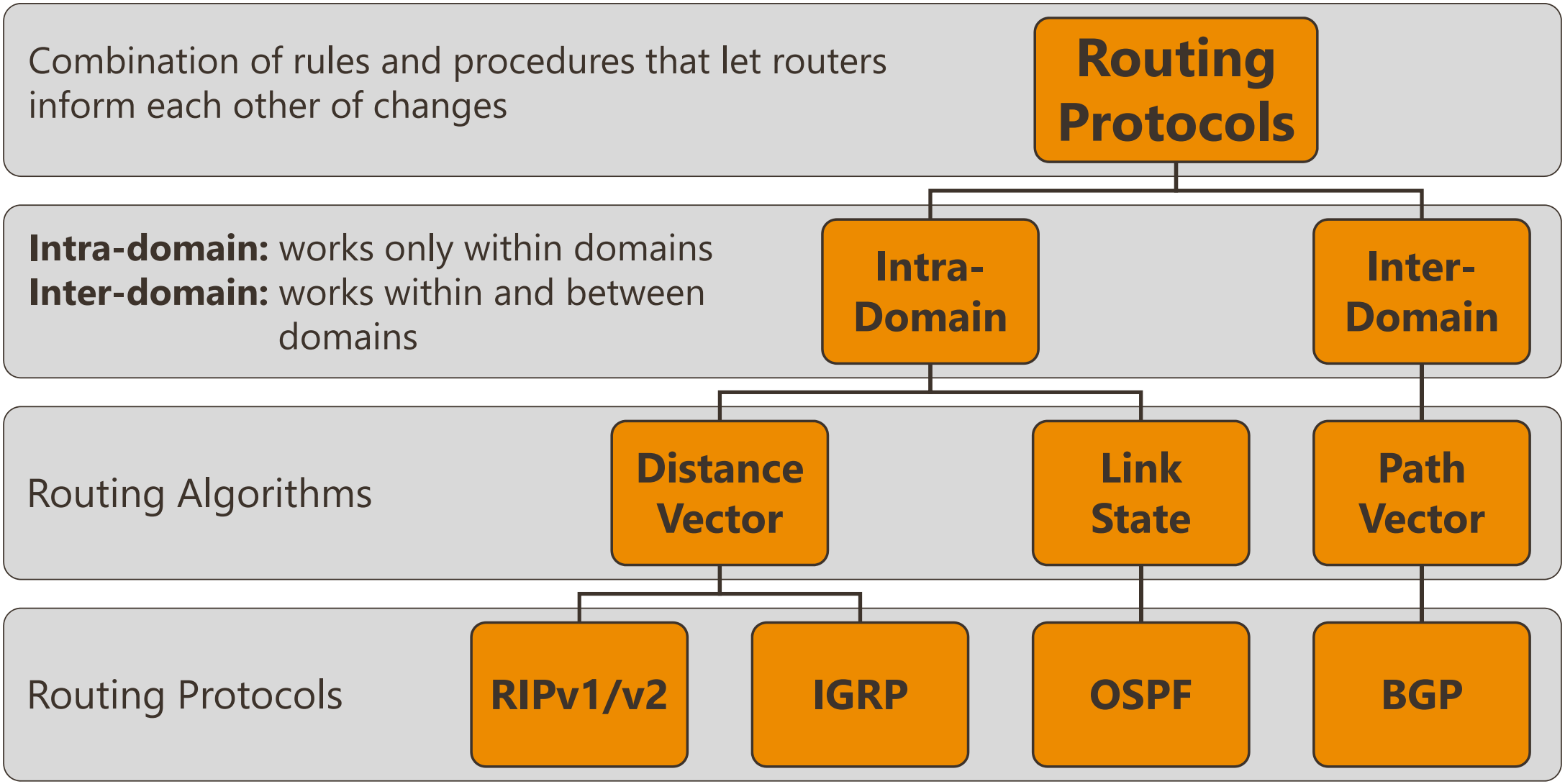
dest. address	output link
net1	i1
net2	i2
net3	i3
net4	i3



IP destination address in arriving packet's header



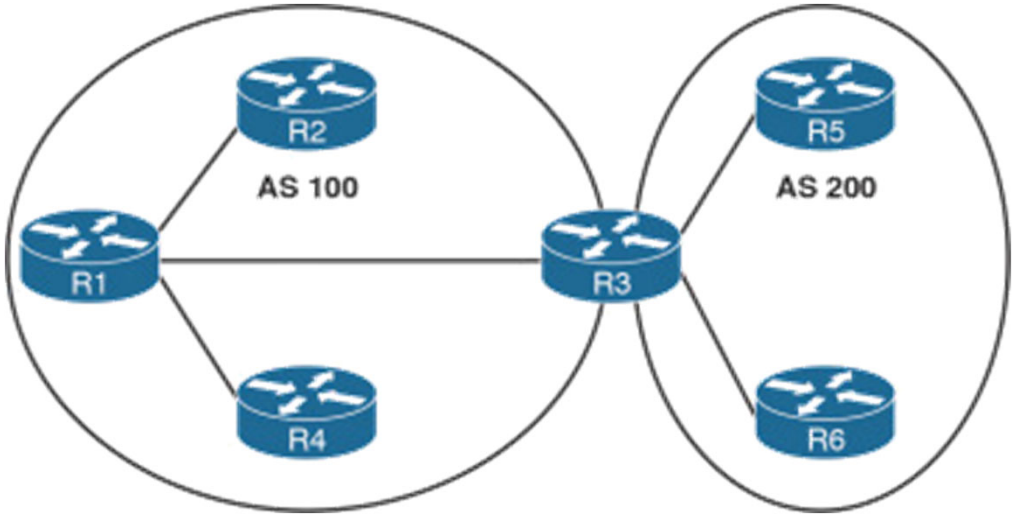
Combination of rules and procedures that let routers inform each other of changes



Intra-domain Routing

Curtin University

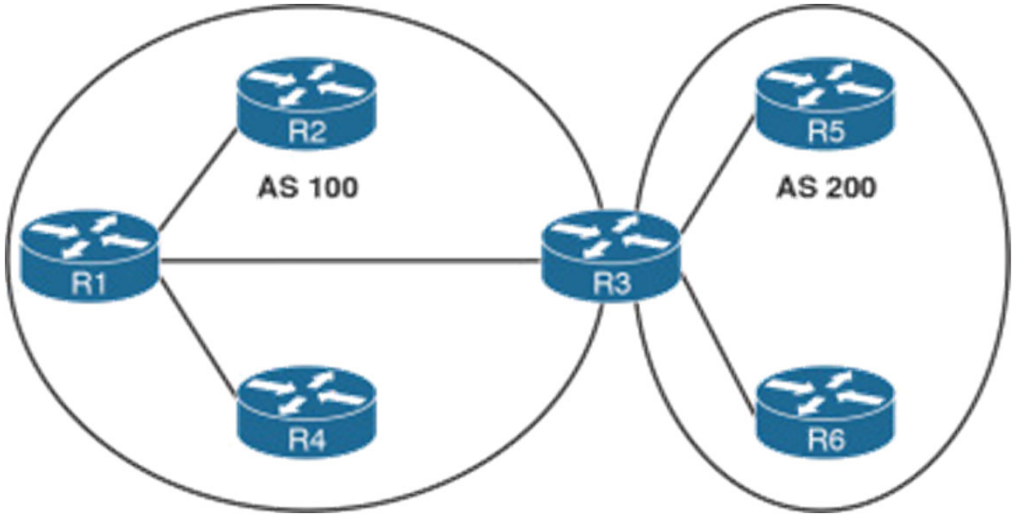
- **Routing** algorithm works **only within domains**
  - ✓ *Only aware of other routers within their domain*
- Protocols used: **Interior-gateway protocols (IGP)**
- **Routing within** an **autonomous network**
- **Ignores** the internet **outside** the **AS**(autonomous system)



Inter-domain Routing

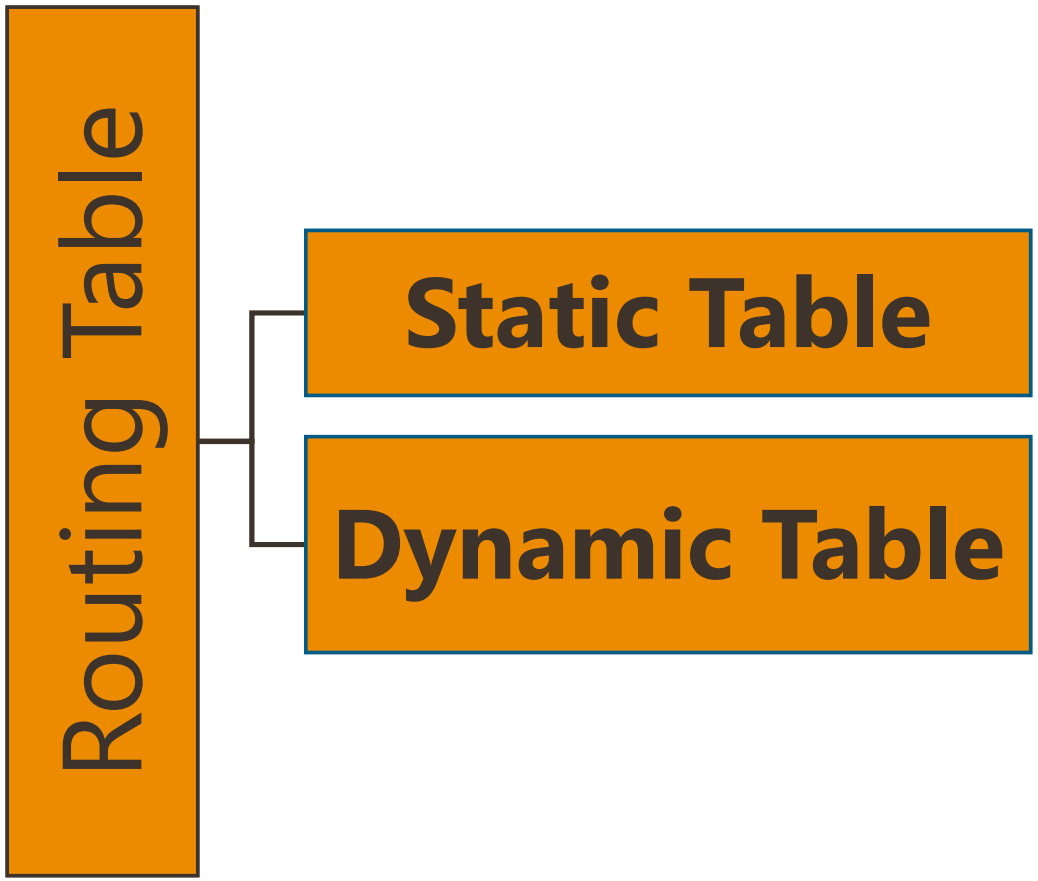
Curtin University

- **Routing** algorithm works **within and between domains**
  - ✓ *Aware of other routers within and between their domain*
- Protocols used: **Exterior-gateway protocols (EGP)**
- **Routing between** the **autonomous networks**
- **Assumes** the internet contains the **collection of interconnected AS**(autonomous systems)



Routing Table

Curtin University



Manual entries with
ip route

Automatic entries with routing protocols when there is a change in the network.
i.e. **RIP, IGRP, OSPF, etc**

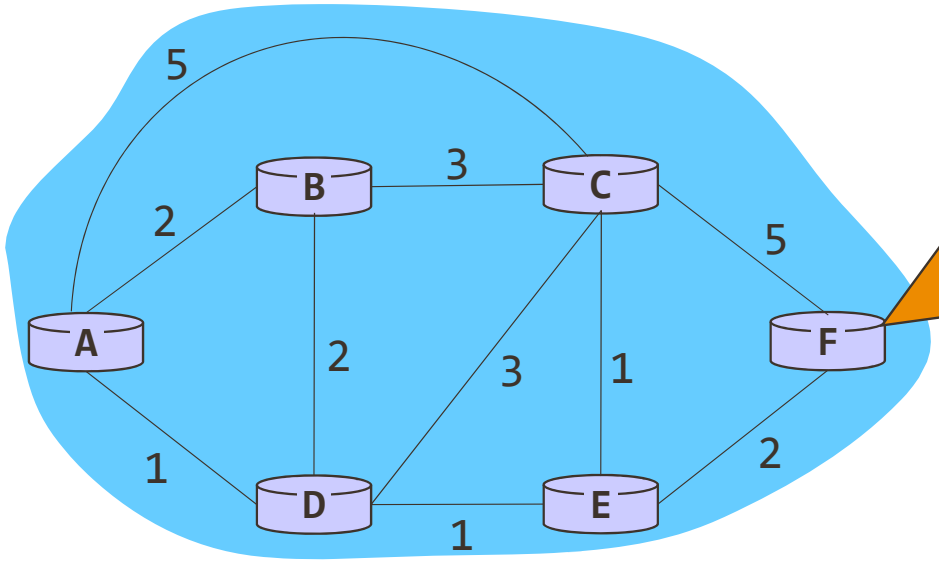
Curtin University

Routing Algorithms

- Link State Routing
  - Topology Dissemination
  - Computing Shortest Path (Dijkstra)
- Distance Vector Routing
  - Bellman Ford Algorithm
  - Distance Vector Updates

Graph Abstraction

Curtin University



cost could always be 1, or

$$\propto \frac{1}{Bandwidth}$$

$$\propto Congestion$$

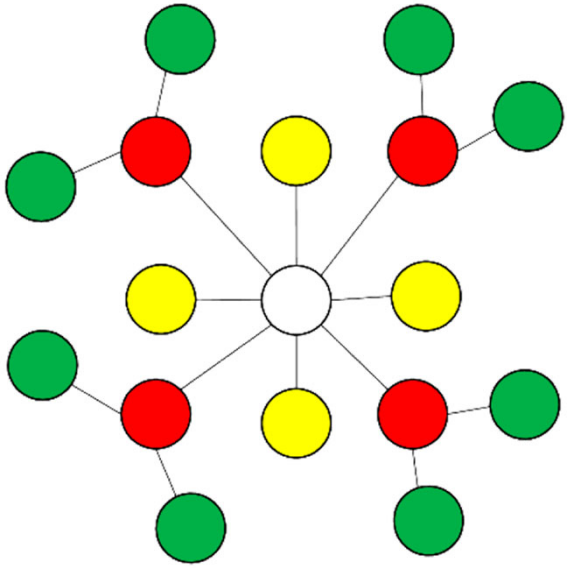
key question: what is the least-cost path between S and R?

routing algorithm: algorithm that finds that least cost path

Routing Algorithms: Packet Switched Networks

Curtin University

- **Link-State & Distance Vector** assume:
  - a router knows
    - ✓the address of each neighbor
    - ✓cost of reaching each neighbor
- **Link State:** A node tells every other node in the network its distance to its neighbors
- **Distance Vector:** A node tells its neighbors its distance to every other node in the network
- Both are **distributed**



Link State Routing

Curtin University

- Router tells every other routers in the network its distance to its neighbors
- Router knows entire network topology (global information), and computes shortest path by itself

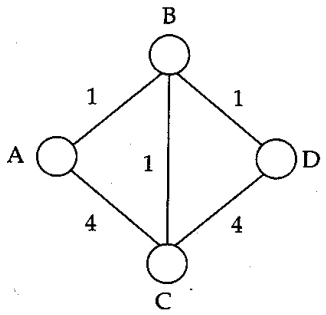
Independent computation of routes

- **Key elements**
  1. **Topology Dissemination**
  2. **Computing Shortest Routes (i.e. Dijkstra)**

1. Topology Dissemination

Curtin University

- A router describes its neighbors with a link state packet (LSP)



LSPs CREATED BY A

A	B	1
A'S ID	B'S ID	COST

A	C	4

- Use controlled flooding to distribute this info to everywhere
  - store LSPs in an LSP database
  - if new, forward to every interface other than incoming one

2. Computing Shortest Routes: Dijkstra's Algorithm

Curtin University

- Assume network topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- **Computes least cost paths** from one node ('source") to all other nodes
- **LSA exchange -> LSDB -> Dijktra -> Routing Table** for the node

Link State Advertisement

After **k iterations**, know least cost path to **k destinations**

Dijkstra's Algorithm

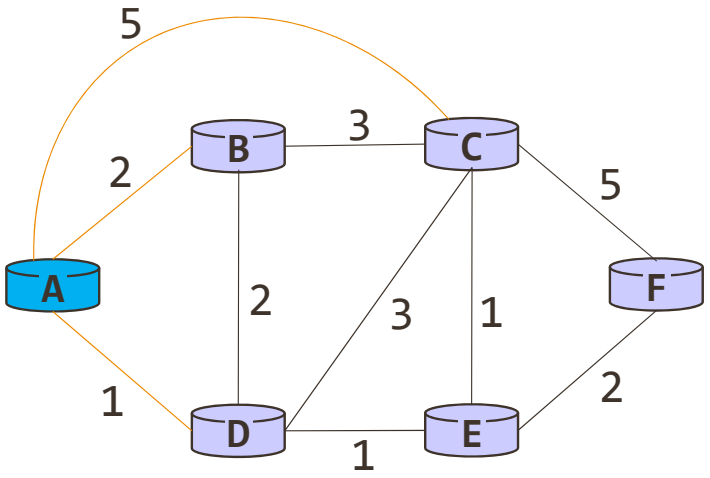
Notation:

- **c(i,j):** Link cost from node **i** to **j**; cost infinite if not direct neighbors
- **L(v):** Current value of cost of path from source to destination **v**
- **p(v):** Predecessor node along path from source to **v**, that is next **v**
- **N:** Set of nodes whose least cost path definitively known

```
1  Initialization:
2  N = {A}
3  for all nodes v
4    if v adjacent to A
5        L(v) = c(A,v)
6    else
7        L(v) = infinity
8
9  Loop
10   Find w not in N such that L(w) is a minimum
11   Add w to N
12   Update L(v) for all v adjacent to w and not in N
13   L(v) = min(L(v), L(w) + c(w,v))
14   // new cost to v is either old cost to v or
     known shortest path cost to w plus cost from w
     to v
15 Until all nodes in N
```

Dijkstra's Algorithm

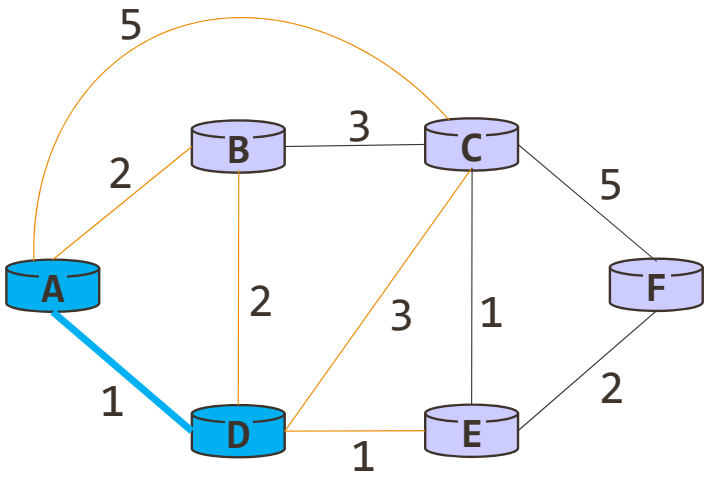
Curtin University



#	N	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path	L(F)	Path
1	{A}	2	A-B	5	A-C	1	A-D	inf	-	inf	-
2											
3											
4											
5											
6											

Dijkstra's Algorithm

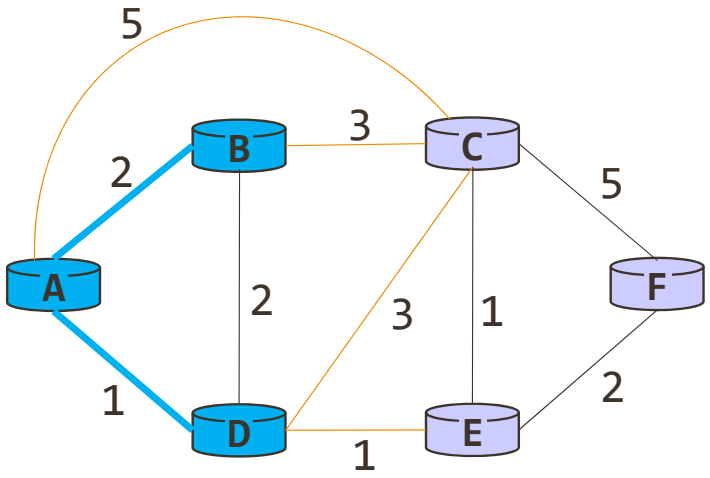
Curtin University



#	N	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path	L(F)	Path
1	{A}	2	A-B	5	A-C	**1**	**A-D**	inf	-	inf	-
2	{A,D}	2	A-B	4	A-D-C	~~1~~	~~A-D~~	2	A-D-E	inf	-
3											
4											
5											
6											

Dijkstra's Algorithm

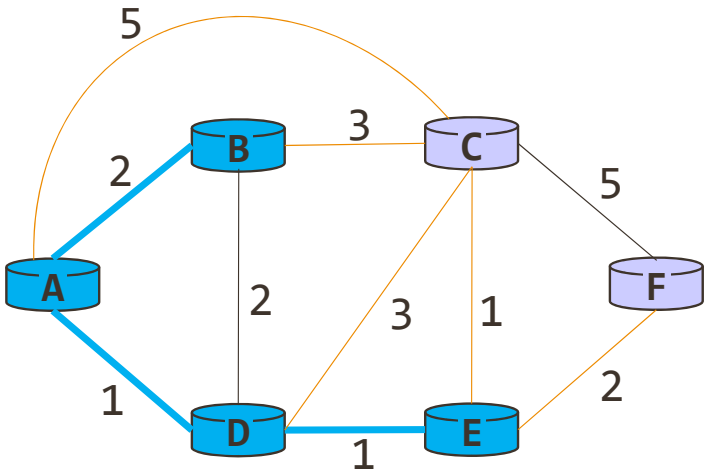
Curtin University



#	N	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path	L(F)	Path
1	{A}	2	A-B	5	A-C	**1**	**A-D**	inf	-	inf	-
2	{A,D}	**2**	**A-B**	4	A-D-C	~~1~~	~~A-D~~	2	A-D-E	inf	-
3	{A,D,B}	~~2~~	~~A-B~~	4	A-D-C	~~1~~	~~A-D~~	2	A-D-E	inf	-
4											
5											
6											

Dijkstra's Algorithm

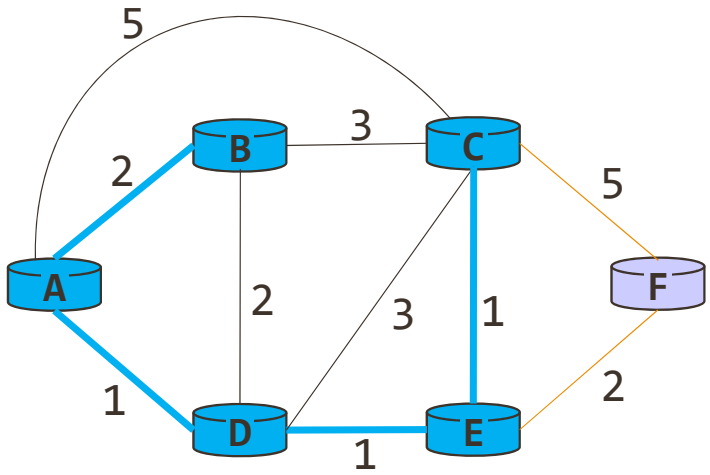
Curtin University



#	N	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path	L(F)	Path
1	{A}	2	A-B	5	A-C	**1**	**A-D**	inf	-	inf	-
2	{A,D}	**2**	**A-B**	4	A-D-C	~~1~~	~~A-D~~	2	A-D-E	inf	-
3	{A,D,B}	~~2~~	~~A-B~~	4	A-D-C	~~1~~	~~A-D~~	**2**	**A-D-E**	inf	-
4	**{A,D,B,E}**	~~2~~	~~A-B~~	3	A-D-E-C	~~1~~	~~A-D~~	~~2~~	~~A-D-E~~	4	A-D-E-F
5											
6											

Dijkstra's Algorithm

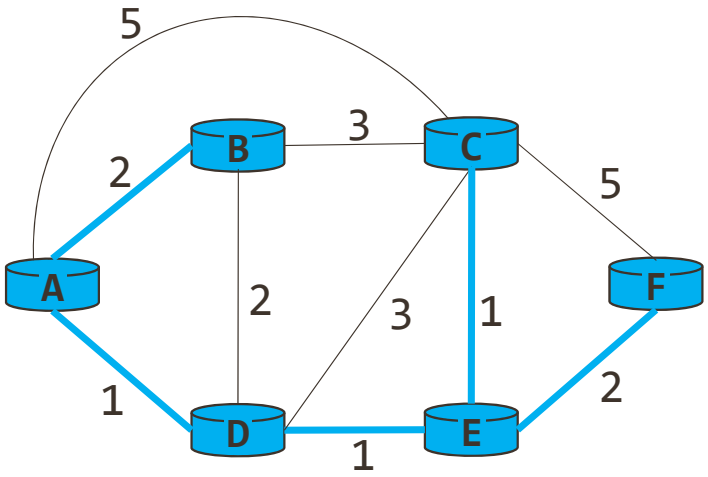
Curtin University



#	N	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path	L(F)	Path
1	{A}	2	A-B	5	A-C	**1**	**A-D**	inf	-	inf	-
2	{A,D}	**2**	**A-B**	4	A-D-C	~~1~~	~~A-D~~	2	A-D-E	inf	-
3	{A,D,B}	~~2~~	~~A-B~~	4	A-D-C	~~1~~	~~A-D~~	**2**	**A-D-E**	inf	-
4	{A,D,B,E}	~~2~~	~~A-B~~	**3**	**A-D-E-C**	~~1~~	~~A-D~~	~~2~~	~~A-D-E~~	4	A-D-E-F
5	{A,D,B,E,C}	~~2~~	~~A-B~~	~~3~~	~~A-D-E-C~~	~~1~~	~~A-D~~	~~2~~	~~A-D-E~~	4	A-D-E-F
6											

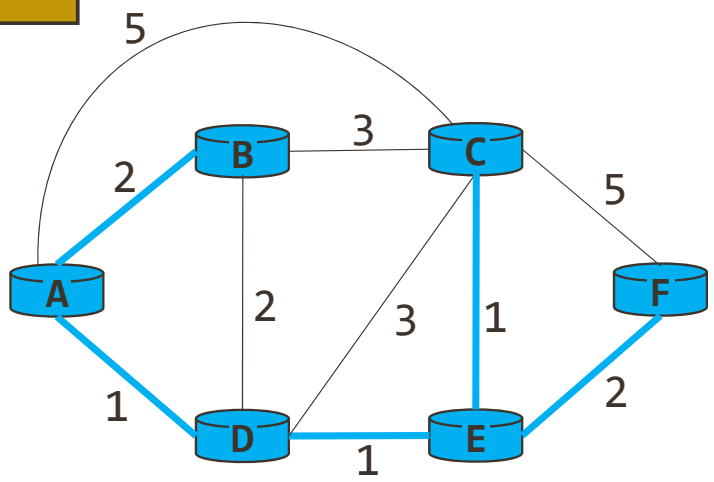
Dijkstra's Algorithm

Curtin University

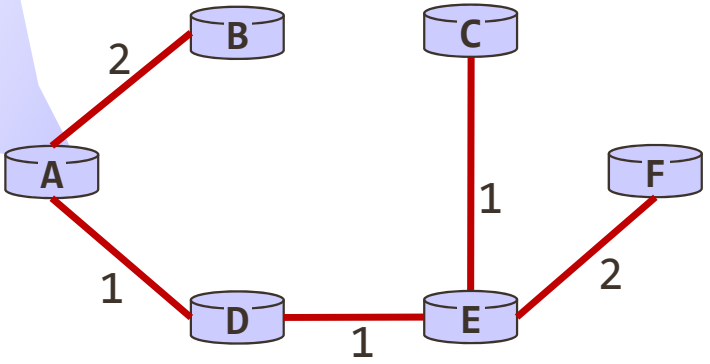


#	N	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path	L(F)	Path
1	{A}	2	A-B	5	A-C	**1**	**A-D**	inf	-	inf	-
2	{A,D}	**2**	**A-B**	4	A-D-C	~~1~~	~~A-D~~	2	A-D-E	inf	-
3	{A,D,B}	~~2~~	~~A-B~~	4	A-D-C	~~1~~	~~A-D~~	**2**	**A-D-E**	inf	-
4	{A,D,B,E}	~~2~~	~~A-B~~	**3**	**A-D-E-C**	~~1~~	~~A-D~~	~~2~~	~~A-D-E~~	4	A-D-E-F
5	{A,D,B,E,C}	~~2~~	~~A-B~~	~~3~~	~~A-D-E-C~~	~~1~~	~~A-D~~	~~2~~	~~A-D-E~~	**4**	**A-D-E-F**
6	{A,D,B,E,C,F}	~~2~~	~~A-B~~	~~3~~	~~A-D-E-C~~	~~1~~	~~A-D~~	~~2~~	~~A-D-E~~	~~4~~	~~A-D-E-F~~

Dijkstra's Algorithm



Dest.	Cost	Hop
B	2	B
C	3	D
D	1	D
E	2	D
F	4	D



#	N	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path	L(F)	Path
1	{A}	2	A-B	5	A-C	**1**	**A-D**	inf	-	inf	-
2	{A,D}	**2**	**A-B**	4	A-D-C	~~1~~	~~A-D~~	2	A-D-E	inf	-
3	{A,D,B}	~~2~~	~~A-B~~	4	A-D-C	~~1~~	~~A-D~~	**2**	**A-D-E**	inf	-
4	{A,D,B,E}	~~2~~	~~A-B~~	**3**	**A-D-E-C**	~~1~~	~~A-D~~	~~2~~	~~A-D-E~~	4	A-D-E-F
5	{A,D,B,E,C}	~~2~~	~~A-B~~	~~3~~	~~A-D-E-C~~	~~1~~	~~A-D~~	~~2~~	~~A-D-E~~	**4**	**A-D-E-F**
6	{A,D,B,E,C,F}	~~2~~	~~A-B~~	~~3~~	~~A-D-E-C~~	~~1~~	~~A-D~~	~~2~~	~~A-D-E~~	~~4~~	~~A-D-E-F~~

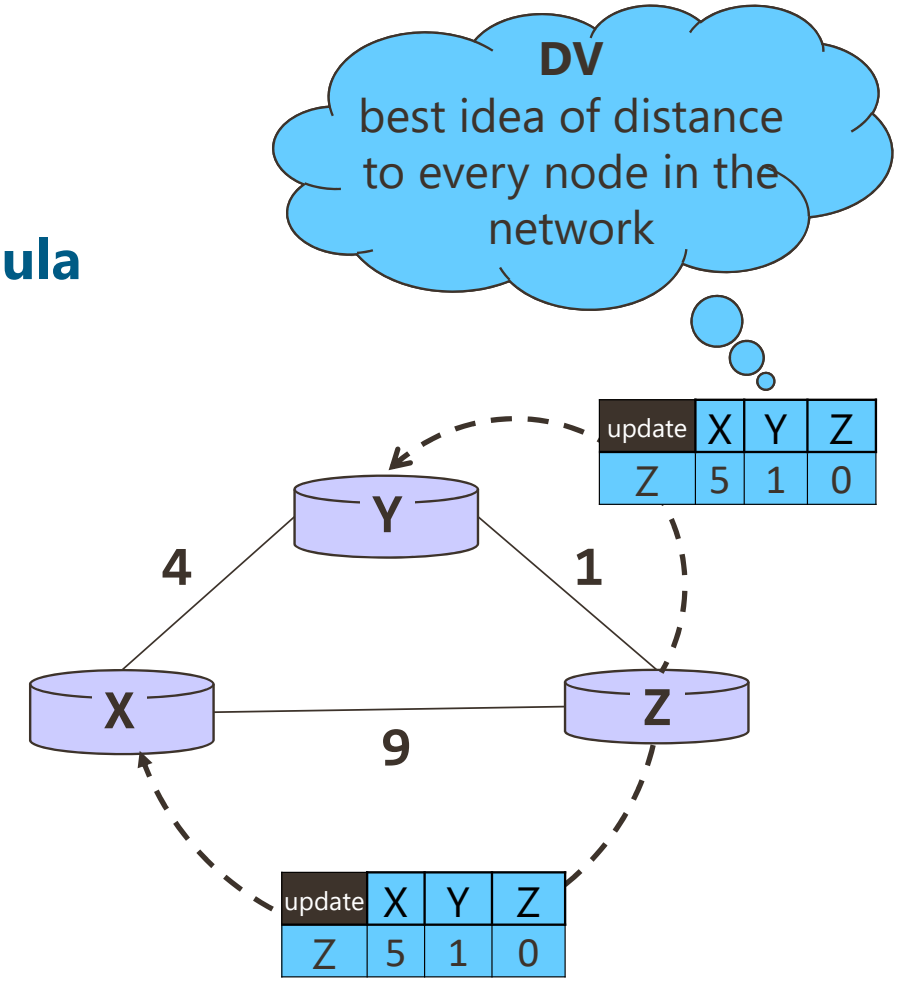
Distant Vector Routing

Curtin University

- **Basic Idea**
  1) Send DV (Distant Vector) to neighbors
  2) Update DV using **B-F (Bellman-Ford) Formula**

$$D_x(y) = min_v \{C(x,v) + D_v(y)\}, \forall y \, \varepsilon \, Nodes$$

Least cost (x->y) cost (x->v)



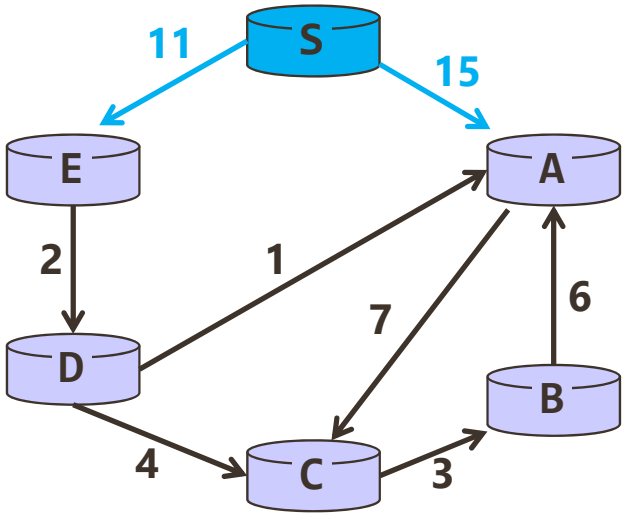
Bellman Ford Algorithm

This implementation takes in a graph, represented as lists of vertices and edges, and fills two arrays (distance and predecessor) about the shortest path from the source to each vertex

```
1  Initialization:
2  Input:  Directed graph G=(V,E);
           Edge lengths {l_e : e ε E}
3  Output:  reachable from s, dist(u) is set to the
           distance from s to u
4
5  for all u ε V:
6    dist(u) = ∞
7    prev(u) = nil
8
9  dist(s) = 0
10 repeat |V| − 1 times:
11   for all u ε V:
12     update(e)
13
14 function update((u, v) ε E)
15   dist(v) = min{dist(v), dist(u) + l(u, v)}
16   prev(v) = u // conditionally, keep track of the predecessor
```

Curtin University

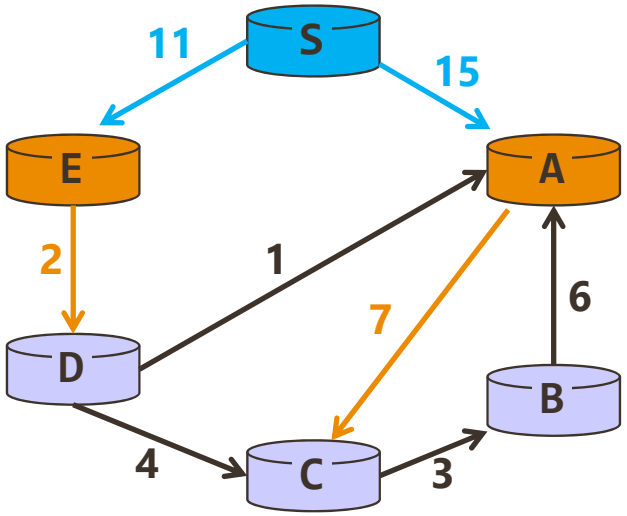
BF – 1st Hop



#	S	L(A)	Path	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path
1	0	15	S-A	∞	-	∞	-	∞	-	11	S-E

Curtin University

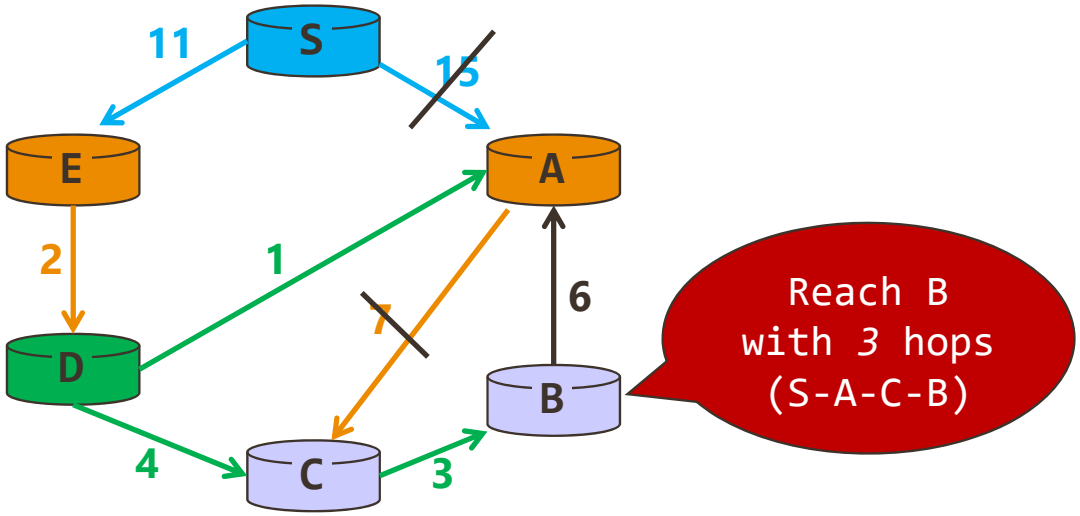
BF – 2nd Hop



#	S	L(A)	Path	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path
1	0	15	S-A	∞	-	∞	-	∞	-	11	S-E
2	0	15	S-A	∞	-	22	S-A-C	13	S-E-D	11	S-E

Curtin University

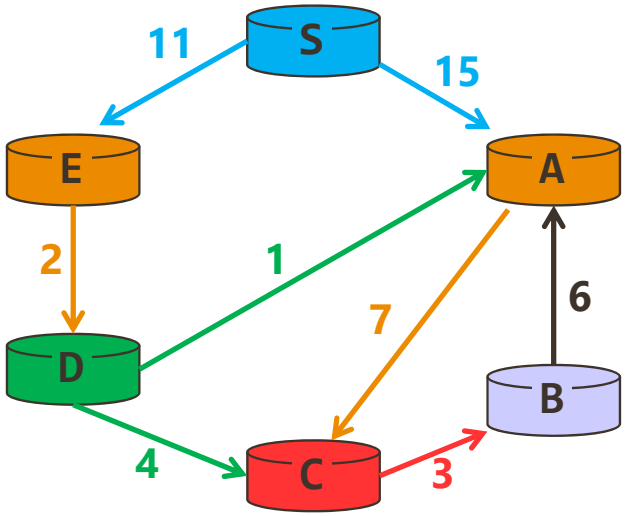
BF – 3rd Hop



#	S	L(A)	Path	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path
1	0	15	S-A	∞	-	∞	-	∞	-	11	S-E
2	0	15	S-A	∞	-	22	S-A-C	13	S-E-D	11	S-E
3	0	~~15~~ 14	~~S-A~~ S-E-D-A	25	S-A-C-B	~~22~~ 17	~~S-A-C~~ S-E-D-C	13	S-E-D	11	S-E

Curtin University

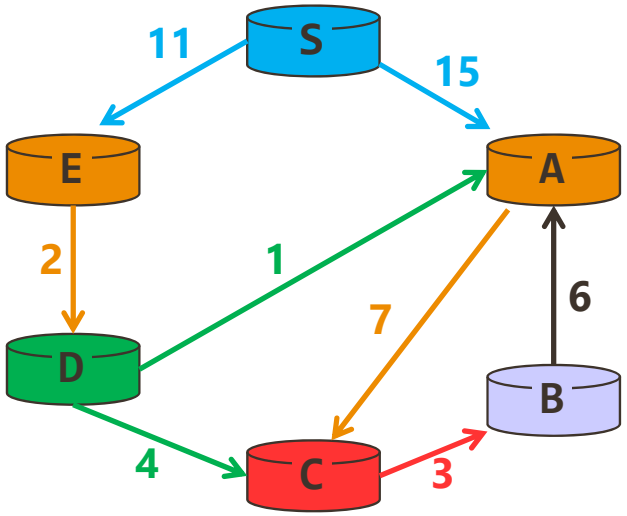
BF – 4th Hop



#	S	L(A)	Path	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path
1	0	15	S-A	∞	-	∞	-	∞	-	11	S-E
2	0	15	S-A	∞	-	22	S-A-C	13	S-E-D	11	S-E
3	0	14	S-E-D-A	25	S-A-C-B	17	S-E-D-C	13	S-E-D	11	S-E
4	0	14	S-E-D-A	~~25~~ 20	~~S-A-C-B~~ S-E-D-C-B	17	S-E-D-C	13	S-E-D	11	S-E

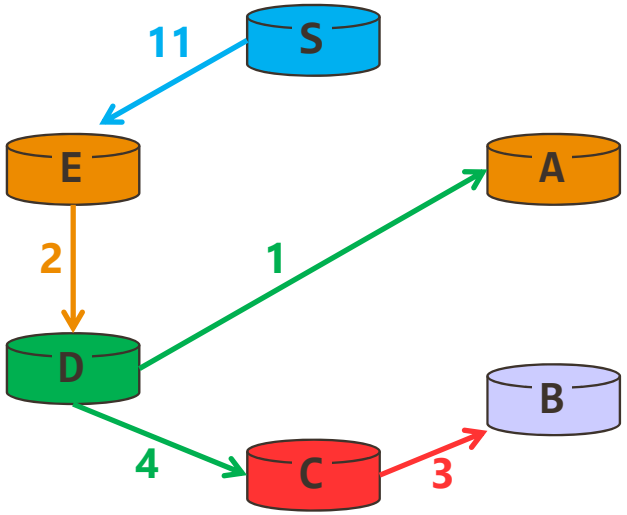
Curtin University

BF – 5th Hop

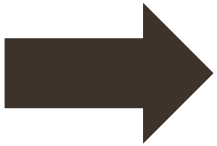


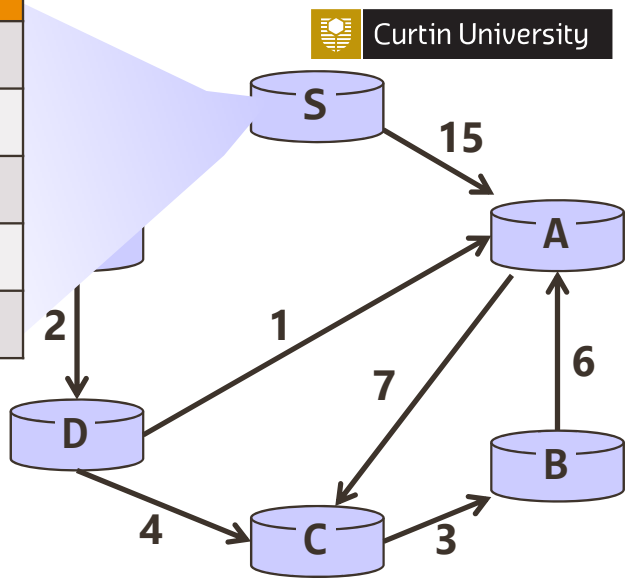
#	S	L(A)	Path	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path
1	0	15	S-A	∞	-	∞	-	∞	-	11	S-E
2	0	15	S-A	∞	-	22	S-A-C	13	S-E-D	11	S-E
3	0	14	S-E-D-A	25	S-A-C-B	17	S-E-D-C	13	S-E-D	11	S-E
4	0	14	S-E-D-A	20	S-E-D-C-B	17	S-E-D-C	13	S-E-D	11	S-E
5	0	14	S-E-D-A	20	S-E-D-C-B	17	S-E-D-C	13	S-E-D	11	S-E

Bellman Ford Algorithm



Dest.	Cost	Hop
A	14	E
B	20	E
C	17	E
D	13	E
E	11	E





#	S	L(A)	Path	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path
1	0	15	S-A	∞	-	∞	-	∞	-	11	S-E
2	0	15	S-A	∞	-	22	S-A-C	13	S-E-D	11	S-E
3	0	14	S-E-D-A	25	S-A-C-B	17	S-E-D-C	13	S-E-D	11	S-E
4	0	14	S-E-D-A	20	S-E-D-C-B	17	S-E-D-C	13	S-E-D	11	S-E
5	0	14	S-E-D-A	20	S-E-D-C-B	17	S-E-D-C	13	S-E-D	11	S-E

Distance Vector

Node X	X	Y	Z
X	0 -	4 -	9 -
Y	∞	∞	∞
Z	∞	∞	∞

Node Y	X	Y	Z
X	∞	∞	∞
Y	4 -	0 -	1 -
Z	∞	∞	∞

Node Z	X	Y	Z
X	∞	∞	∞
Y	∞	∞	∞
Z	9 -	1 -	0 -

Node X	X	Y	Z
X	0 -	4 -	5 Y
Y	4	0	1
Z	9	1	0

Node Y	X	Y	Z
X	0	4	9
Y	4 -	0 -	1 -
Z	9	1	0

Node Z	X	Y	Z
X	0	4	9
Y	4	0	1
Z	5 Y	1 -	0 -

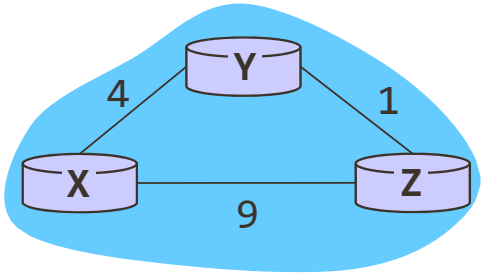
update	X	Y	Z
X	0	4	5

update	X	Y	Z
Z	5	1	0

Node X	X	Y	Z
X	0 -	4 -	5 Y
Y	4	0	1
Z	5	1	0

Node Y	X	Y	Z
X	0	4	5
Y	4 -	0 -	1 -
Z	5	1	0

Node Z	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5 Y	1 -	0 -



$$dist(v) = \min\{dist(v), dist(u) + l(u, v)\}$$

Distance Vector: Points to Note

Curtin University

- If no topology change, **convergence in a few rounds**
  - ✓ After one message exchange, node knows about nodes two hops away
  - ✓ After two message exchange, node knows about nodes three hops away
- **No node has global knowledge**
- **Fully distributed**, yet maintains correct view

Distance Vector: Updates

Curtin University

- **Triggered Updates**

  Send whenever the DV changes

Link/Node failure or cost changes

- **Periodic Updates**

  Sent even when no change in routing table

  ✓ To tell others that **"I am still alive"**

  ✓ To update others' DV in case some **route** becomes **invalid**

Good news travels fast !

Bad news travels slow !

Distance Vector

Node X	X	Y	Z
X	0 -	1 -	5 Y
Y			
Z			

Node Y	X	Y	Z
X			
Y	1 -	0 -	1 -
Z			

Node Z	X	Y	Z
X			
Y			
Z	5 Y	1 -	0 -

Node X	X	Y	Z
X	0 -	1 -	2 Y
Y	1	0	1
Z			

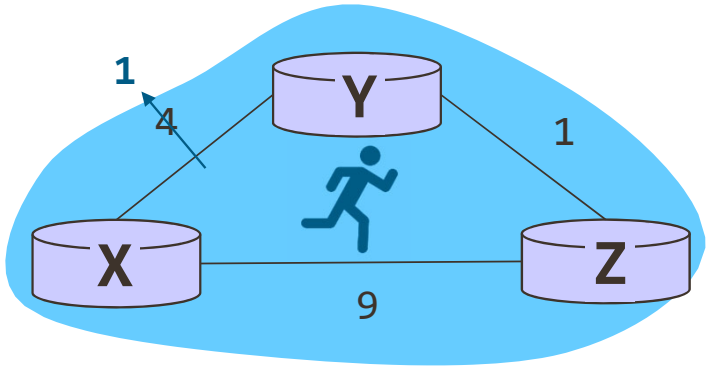
Node Y	X	Y	Z
X	0	1	5
Y	1 -	0 -	1 -
Z			

Node Z	X	Y	Z
X	0	1	5
Y	1	0	1
Z	2 Y	1 -	0 -

Node X	X	Y	Z
X	0 -	1 -	2 Y
Y	1	0	1
Z	2	1	0

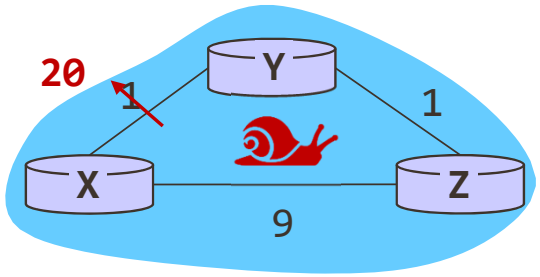
Node Y	X	Y	Z
X	0	1	2
Y	1 -	0 -	1 -
Z	2	1	0

Node Z	X	Y	Z
X	0	1	2
Y	1	0	1
Z	2 Y	1 -	0 -



Good news travels fast !

Distance Vector



Bad news travels slow !

Full Story

Node X	X	Y	Z
X	0 -	20 -	2 Y
Y			
Z			

Node X	X	Y	Z
X	0 -	10 Z	9 -
Y	20	0	1
Z	2	1	0

Node X	X	Y	Z
X	0 -	10 Z	9 -
Y	3	0	1
Z	9	1	0

Node X	X	Y	Z
X	0 -	10 Z	9 -
Y	10	0	1
Z	4	1	0

Node X	X	Y	Z
X	0 -	10 Z	9 -
Y	5	0	1
Z	9	1	0

Node Y	X	Y	Z
X			
Y	20 -	0 -	1 -
Z			

Node Y	X	Y	Z
X	0	20	2
Y	3 Z	0 -	1 -
Z	2	1	0

Node Y	X	Y	Z
X	0	10	9
Y	10 Z	0 -	1 -
Z	9	1	0

Node Y	X	Y	Z
X	0	10	9
Y	5 Z	0 -	1 -
Z	4	1	0

Node Y	X	Y	Z
X	0	10	9
Y	10 Z	0 -	1 -
Z	9	1	0

Node Z	X	Y	Z
X			
Y			
Z	2 Y	1 -	0 -

Node Z	X	Y	Z
X	0	20	2
Y	20	0	1
Z	9 -	1 -	0 -

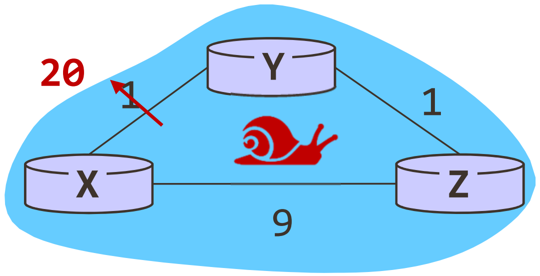
Node Z	X	Y	Z
X	0	10	9
Y	3	0	1
Z	4 Y	1 -	0 -

Node Z	X	Y	Z
X	0	10	9
Y	10	0	1
Z	9 -	1 -	0 -

Node Z	X	Y	Z
X	0	10	9
Y	5	0	1
Z	6 Y	1 -	0 -

Distance Vector

Full Story – cont



Bad news travels slow !

Node X	X	Y	Z
X	0 -	10 Z	9 -
Y	10	0	1
Z	9	1	0

Node Y	X	Y	Z
X	0	10	9
Y	7 Z	0 -	1 -
Z	6	1	0

Node Z	X	Y	Z
X	0	10	9
Y	10	0	1
Z	9 -	1 -	0 -

Node X	X	Y	Z
X	0 -	10 Z	9 -
Y	10	0	1
Z	9	1	0

Node Y	X	Y	Z
X	0	10	9
Y	10 Z	0 -	1 -
Z	9	1	0

Node Z	X	Y	Z
X	0	10	9
Y	7	0	1
Z	8 Y	1 -	0 -

Node X	X	Y	Z
X	0 -	10 Z	9 -
Y	10	0	1
Z	9	1	0

Node Y	X	Y	Z
X	0	10	9
Y	9 Z	0 -	1 -
Z	8	1	0

Node Z	X	Y	Z
X	0	10	9
Y	10	0	1
Z	9 -	1 -	0 -

Node X	X	Y	Z
X	0 -	10 Z	9 -
Y	10	0	1
Z	9	1	0

Node Y	X	Y	Z
X	0	10	9
Y	10 Z	0 -	1 -
Z	9	1	0

Node Z	X	Y	Z
X	0	10	9
Y	9	0	1
Z	9 -	1 -	0 -

Node X	X	Y	Z
X	0 -	10 Z	9
Y	10	0	1
Z	9	1	0

Node Y	X	Y	Z
X	0	10	9
Y	10 Z	0 -	1
Z	9	1	0

Node Z	X	Y	Z
X	0	10	9
Y	10	0	1
Z	9 -	1 -	0

Curtin University

Basis	Link State	Distant Vector
Summary	"**Tell the world** about neighbors"	"**Tell the neighbors** about the world"
Updates	**Triggered/Periodic Updates**	**Periodic Updates**

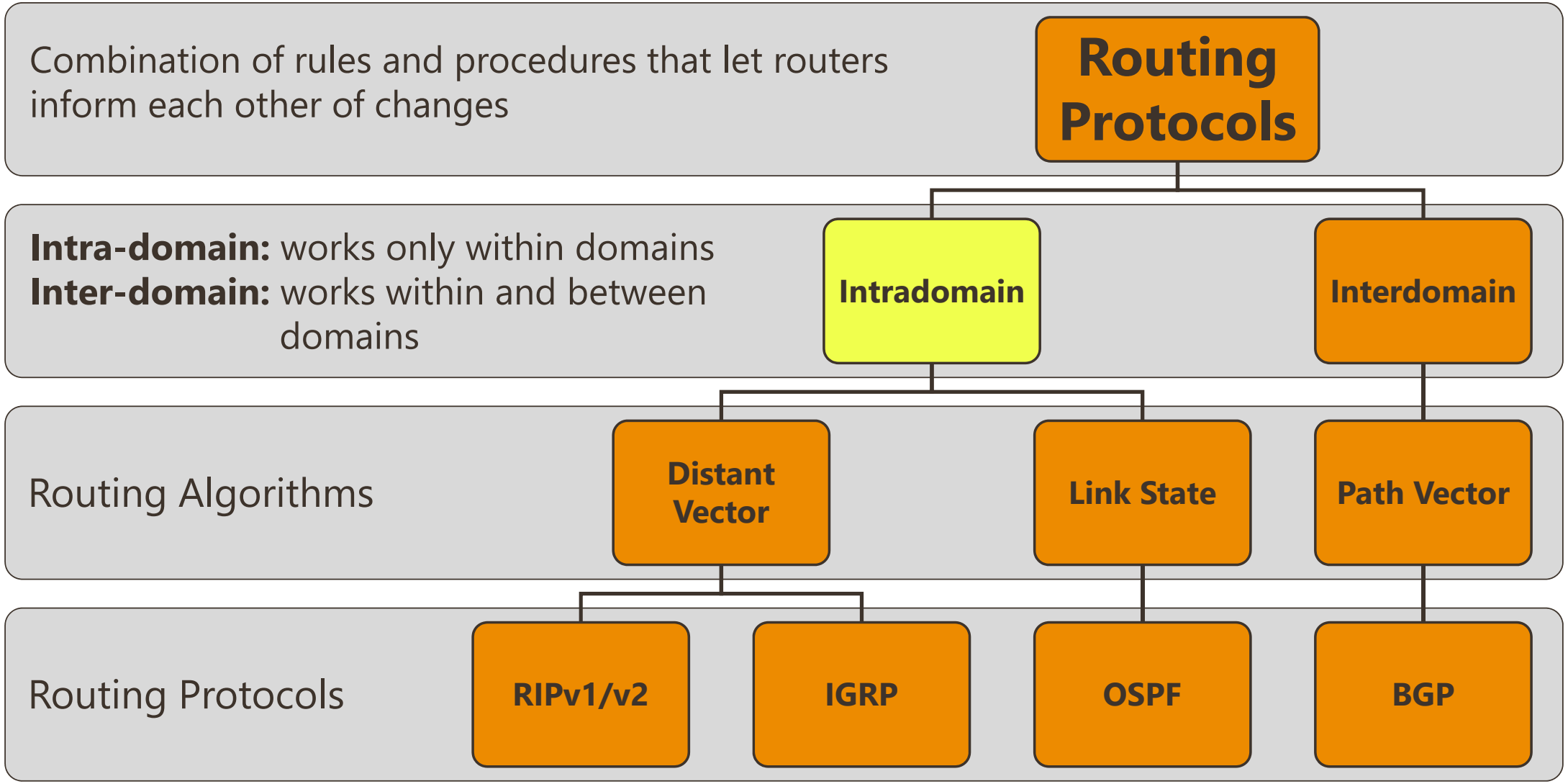
Curtin University

Intra-Domain Routing **Protocols**

- RIP (Routing Information Protocol)
- OSPF (Open Shortest Path First)

Combination of rules and procedures that let routers inform each other of changes

Intra-domain: works only within domains
Inter-domain: works within and between domains



Intradomain Routing

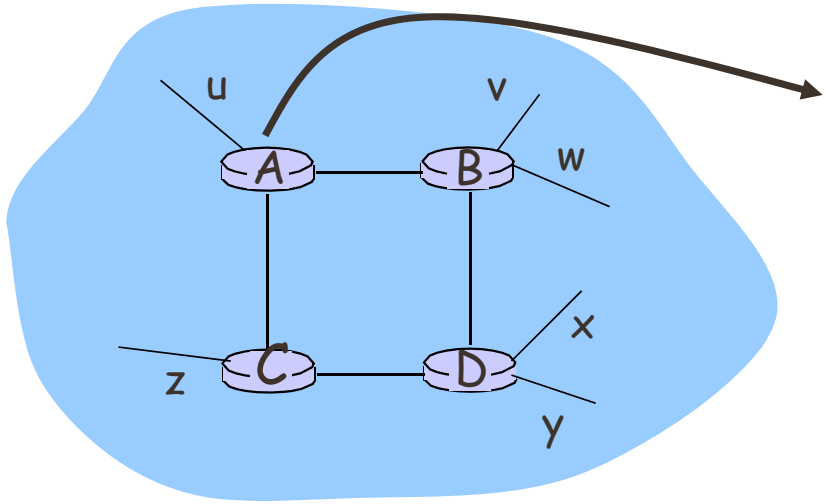
Curtin University

- **a.k.a Interior Gateway Protocols (IGP)**
- Common Protocols:
  - **RIP:** Routing Information Protocol
  - **IGRP:** Interior Gateway Routing Protocol (Cisco proprietary)
  - **EIGRP:** Extended IGRP (Cisco proprietary)
  - **OSPF:** Open Shortest Path First

RIP: Routing Information Protocol

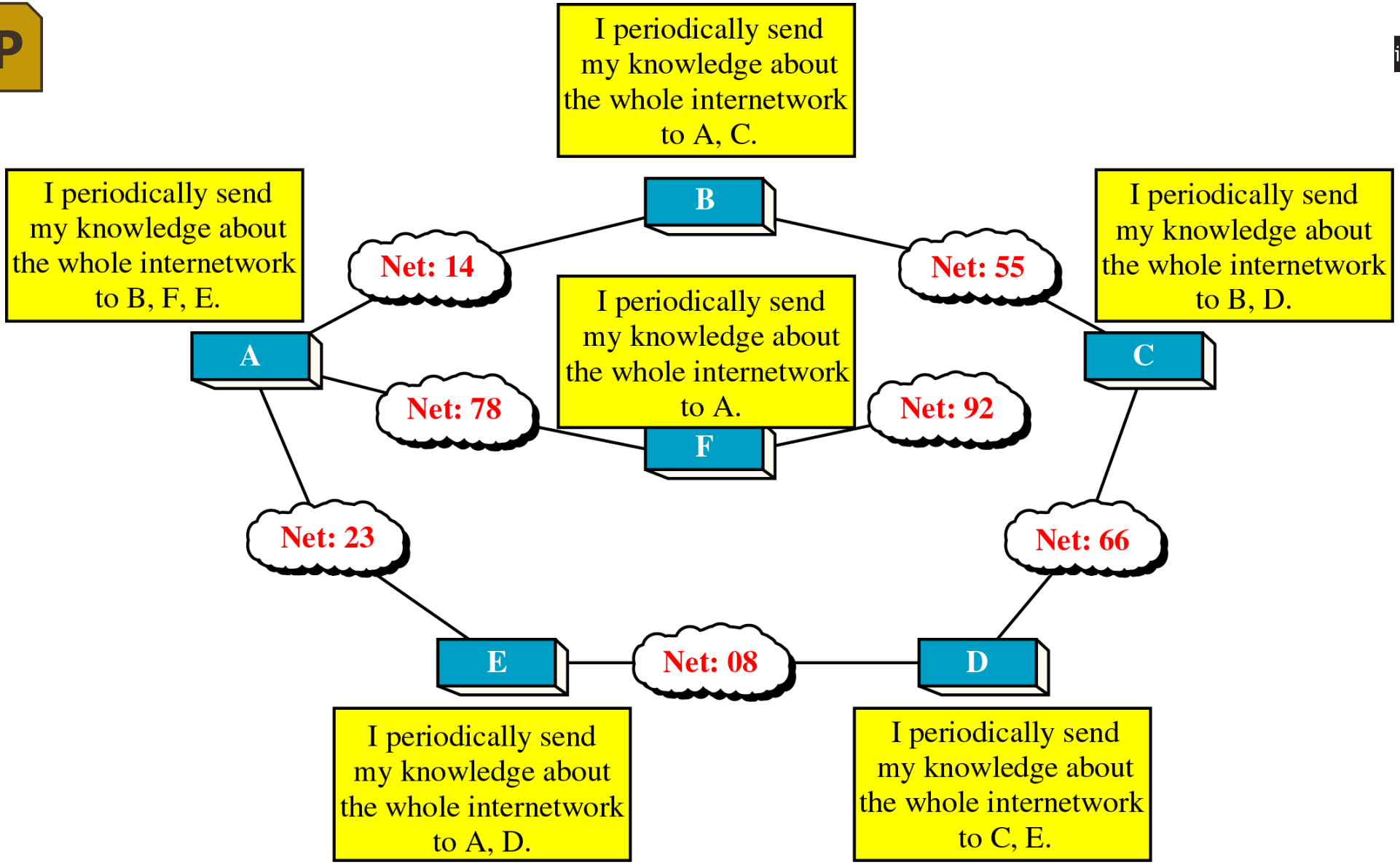
Curtin University

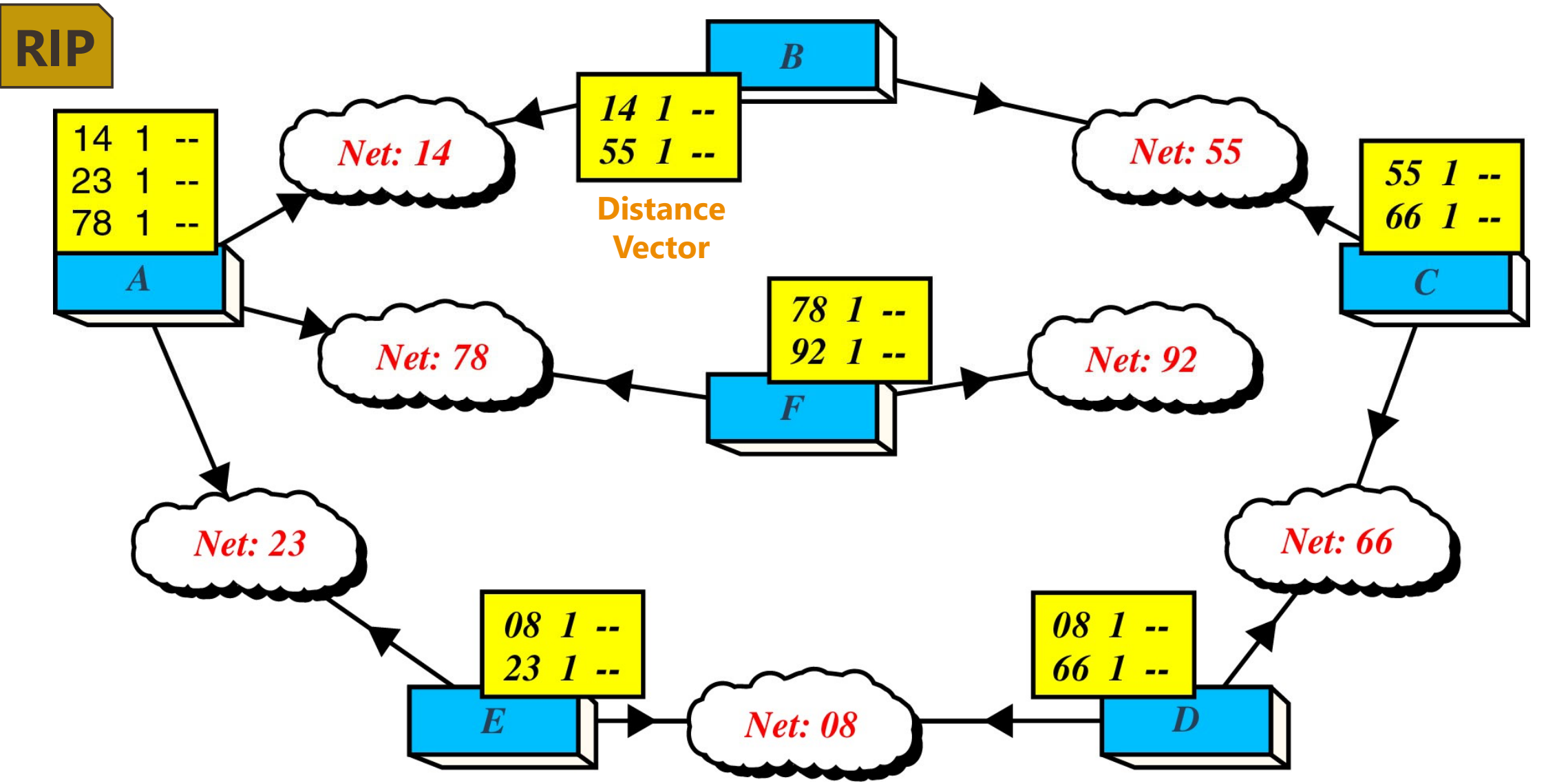
- Uses **Distance Vector** Algorithm
- **Distance Metric: # of hops** (max = 15 hops)
- **Distance Vectors** exchanged among neighbors every **30 sec**
  - *via Response Message (also called **advertisement**)*

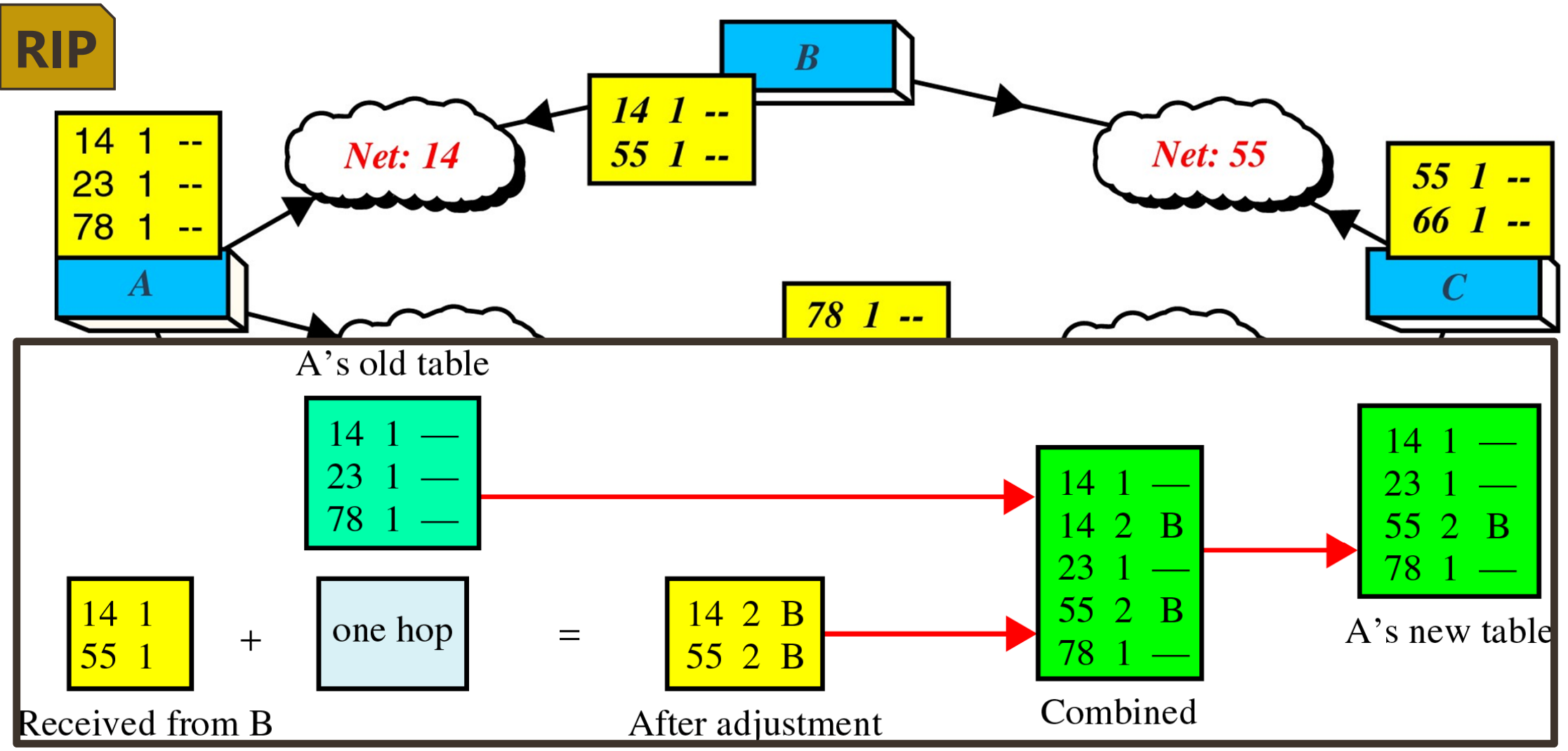


Destination	Cost (hops)	Next hop
U	1	-
V	2	B
W	2	B
X	3	B
Y	3	B
Z	2	C

RIP


I periodically send my knowledge about the whole internetwork to A, C.
I periodically send my knowledge about the whole internetwork to B, F, E.
I periodically send my knowledge about the whole internetwork to B, D.
I periodically send my knowledge about the whole internetwork to A.
B
A
C
F
E
D
Net: 14
Net: 55
Net: 78
Net: 92
Net: 23
Net: 66
Net: 08
I periodically send my knowledge about the whole internetwork to A, D.
I periodically send my knowledge about the whole internetwork to C, E.


RIP
B
14 1 --
55 1 --
Distance
Vector
Net: 14
Net: 55
14 1 --
23 1 --
78 1 --
A
55 1 --
66 1 --
C
78 1 --
92 1 --
Net: 78
F
Net: 92
Net: 23
Net: 66
08 1 --
23 1 --
08 1 --
66 1 --
E
Net: 08
D

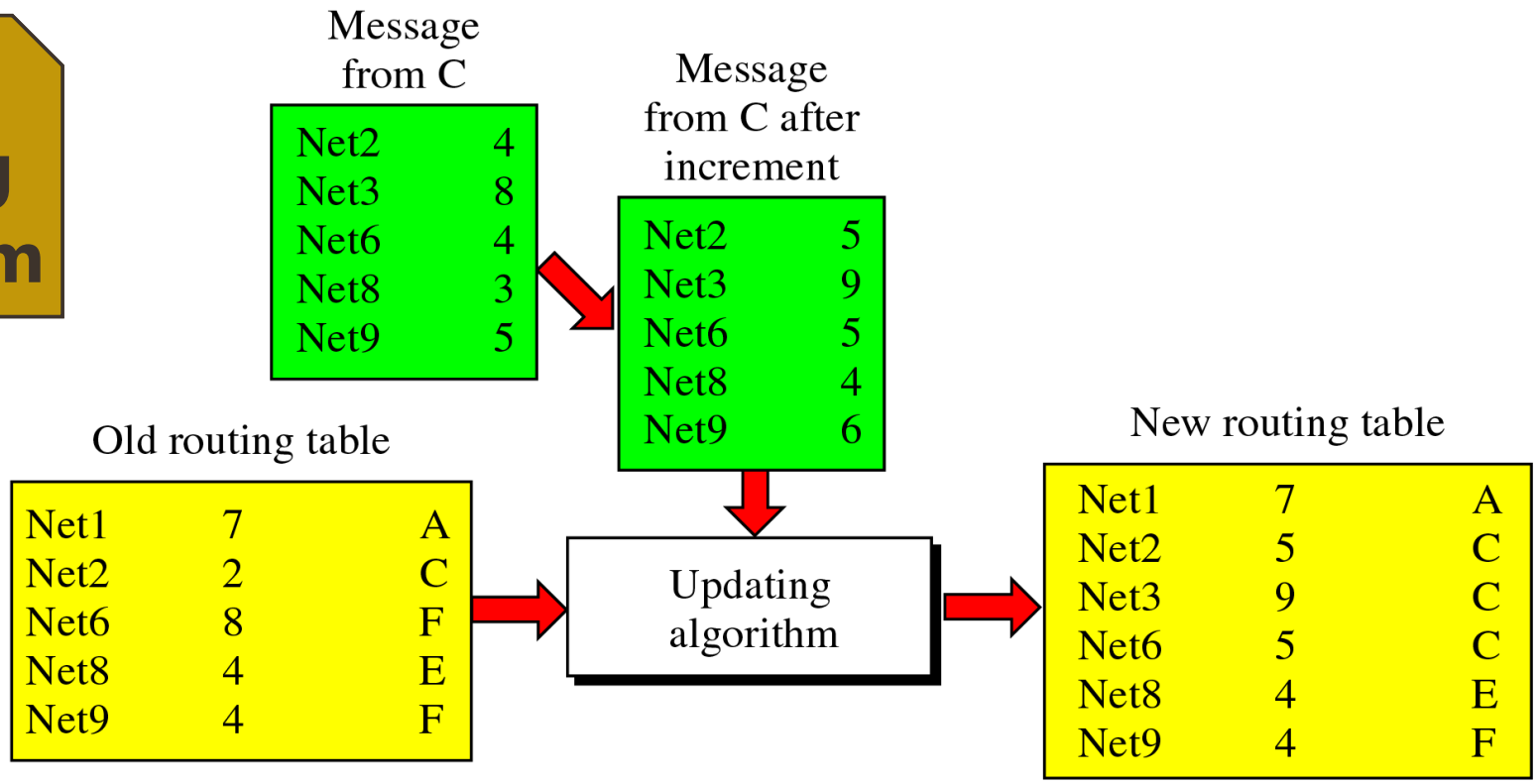

RIP
B
14 1 --
55 1 --
Net: 14
Net: 55
14 1 --
23 1 --
78 1 --
A
55 1 --
66 1 --
C
78 1 --
A's old table
14 1 —
23 1 —
78 1 —
14 1
55 1
+
one hop
=
14 2 B
55 2 B
14 1 —
14 2 B
23 1 —
55 2 B
78 1 —
14 1 —
23 1 —
55 2 B
78 1 —
A's new table
Received from B
After adjustment
Combined

RIP - Updating Algorithm

Curtin University

1. Router adds 1 hop to the hop count field for each advertised route
2. Add advertised destination if not in table
3. If destination is in the table
   a. **Advertising Node == Next hop** field, replace the entry
   b. **Advertising Node != Next hop** field
      I. smaller hop count, replace the entry
      II. Larger or equal hop count, do nothing

RIP - Updating Algorithm



Rules

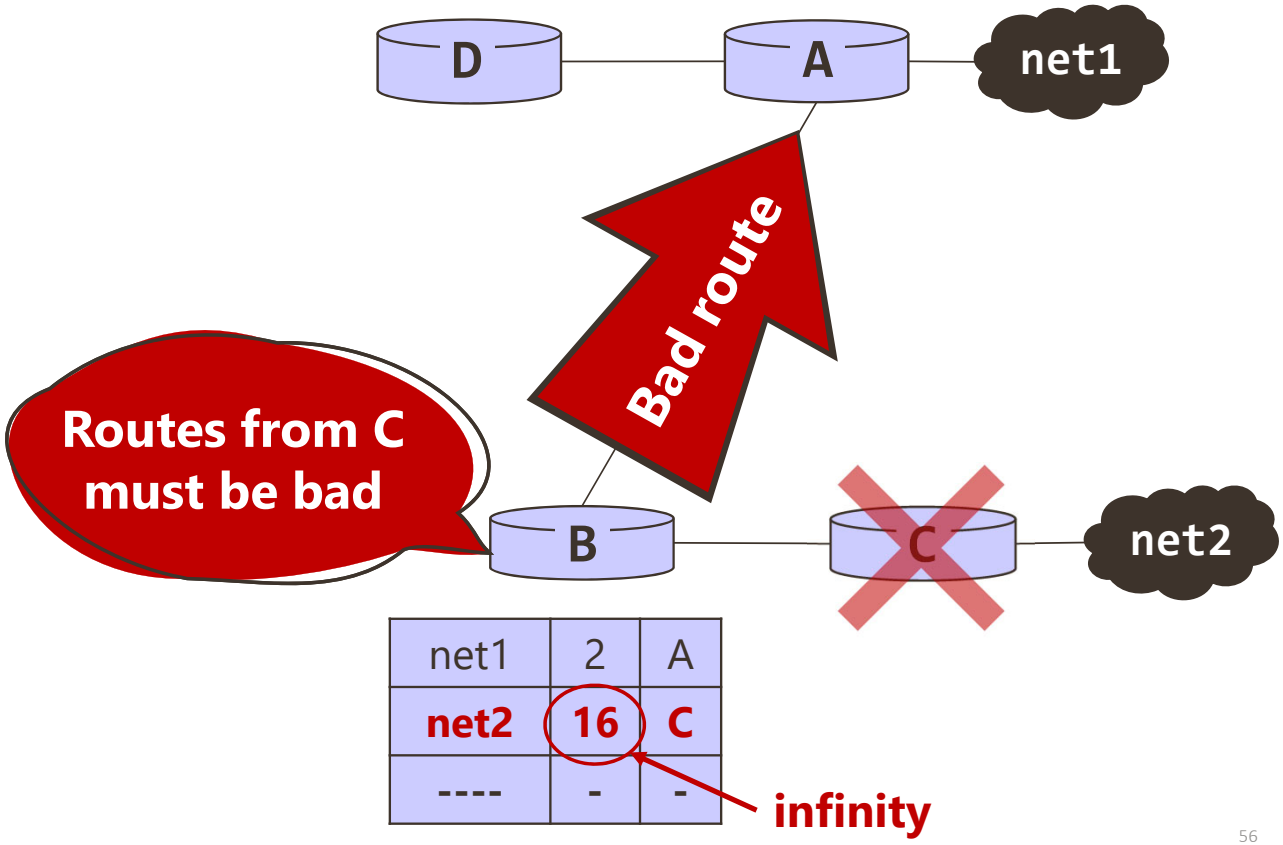
Net2: Replace **(Rule 3.a)**
Net3: Replace **(Rule 1)**
Net6: Replace **(Rule 3.b.i)**
Net8: Replace **(Rule 3.b.ii)**
Net9: Replace **(Rule 3.b.ii)**

Note that there is no news about Net1 in the advertised message, so none of the rules apply to this entry.

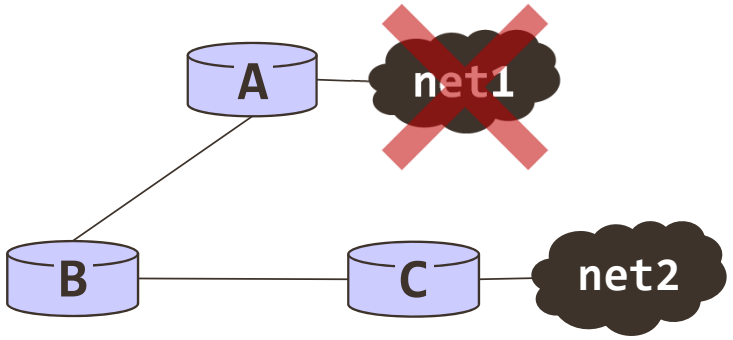
RIP - Route Poisoning

Curtin University

- **Infinity -> 16 hops** (unreachable)



RIP - Count to Infinity



Curtin University

Unstable routing
during this whole time

Node A	cost	hop
net1	16	-
net2	3	B
----	-	-

+ 1	cost
net1	16
net2	3

Node B	cost	hop
net1	2	A
net2	2	C
----	-	-

+ 1	cost
net1	2
net2	2

Node A	cost	hop
net1	3	B
net2	3	B
----	-	-

Node B	cost	hop
net1	16	A
net2	2	C
----	-	-

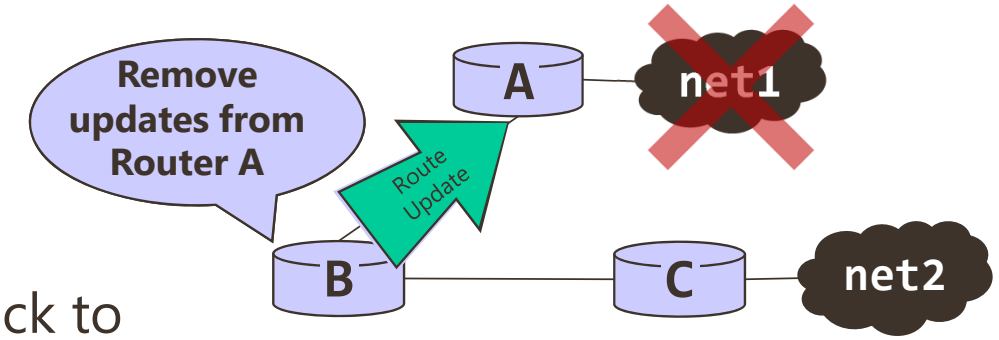
Node A	cost	hop
net1	16	B
net2	3	B
----	-	-

Node B	cost	hop
net1	4	A
net2	2	C
----	-	-

Node A	cost	hop
net1	5	B
net2	3	B
----	-	-

Node B	cost	hop
net1	16	A
net2	2	C
----	-	-

RIP - Simple Split Horizon



Curtin University

Don't advertise routes back to the router you learn them from

Node A	cost	hop
net1	16	-
net2	3	B
----	-	-

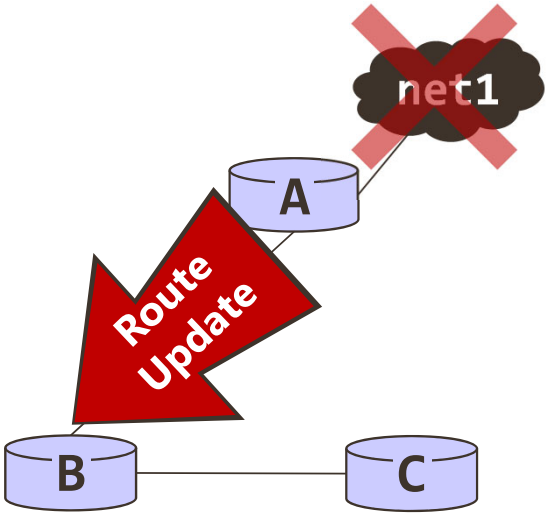
Node B	cost	hop
net1	2	A
net2	2	C
----	-	-

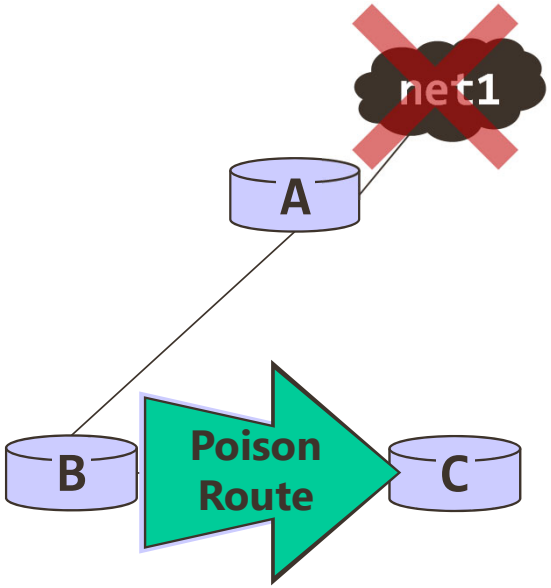
+ 1	cost
net1	16
net2	3

+ 1	cost
~~net1~~	~~2~~
net2	2

RIP - Split Horizon with Poison Reverse

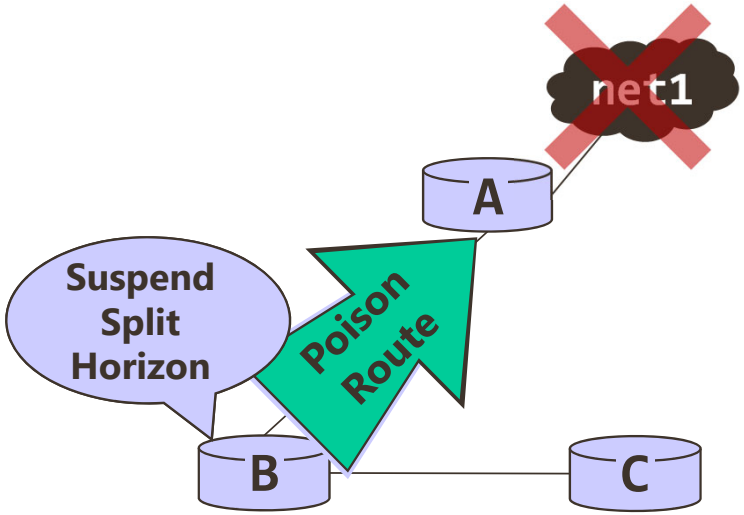
Curtin University





Triggered Partial Update

Send only the info about bad route.

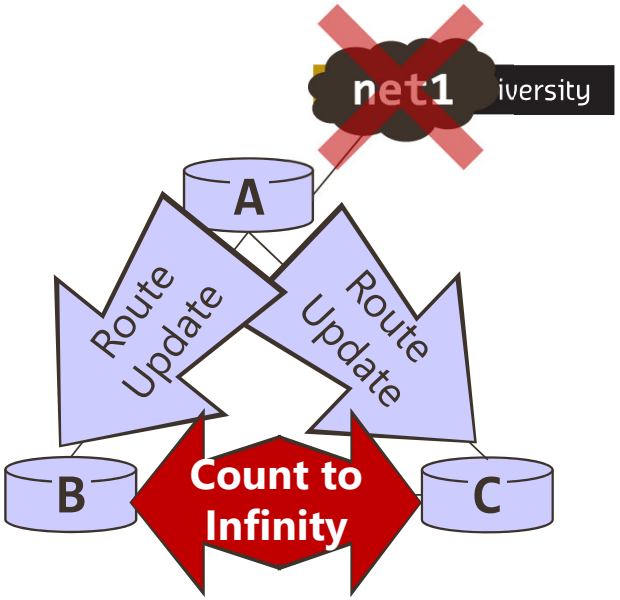


Reverse Route Poisoning

Send confirmation from B (just the poison route) immediately.

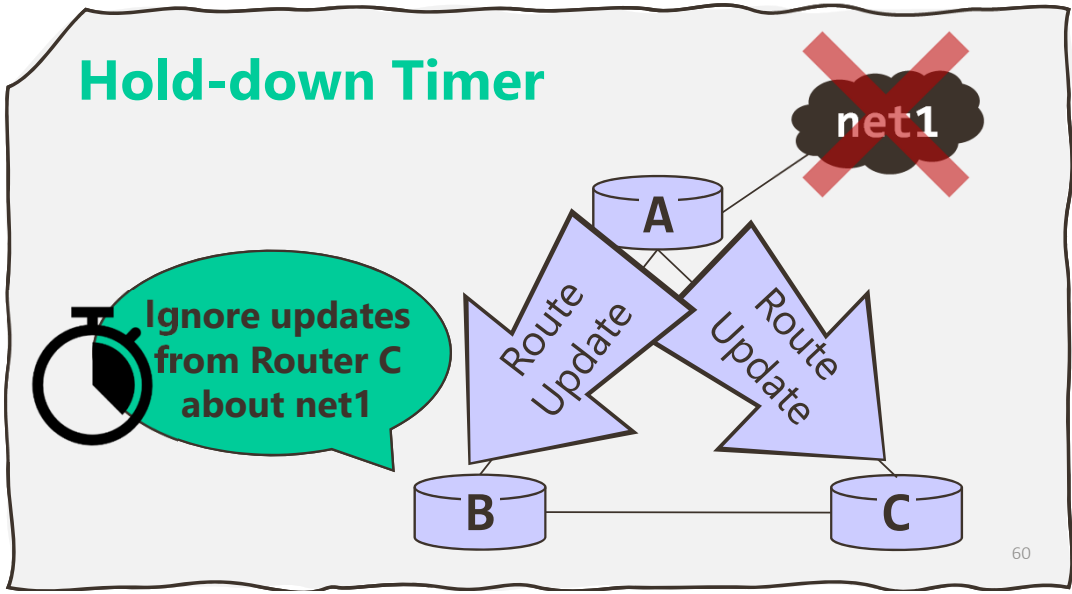
Redundancy In Networks?

- **Split horizon does not work**



- **Hold-down Timer**

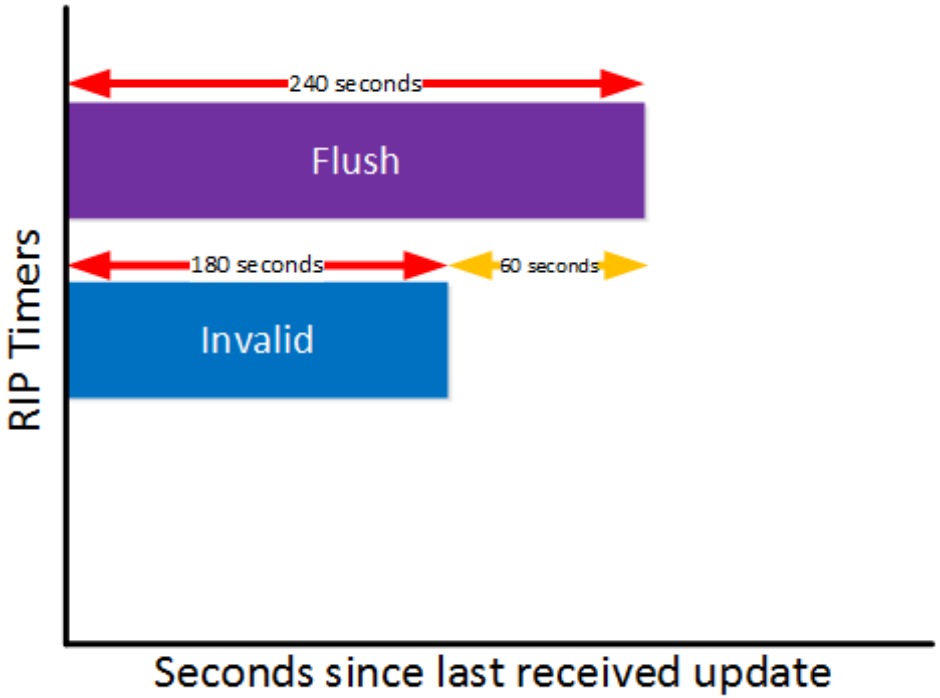
  - ✓ Suppressing any routing update after a poison route is received
  - ✓ Gives time to converge
  - ✓ But, B and C will listen to new updates of net1 from A



RIP Timers

Curtin University

- **Update:** interval to send routing updates (distance vectors), *default: 30s*
- **Invalid:** time to wait since received the last valid update, *default: 180s*
- **Flush:** time to wait since received the last valid update before throwing a route away, *default: 240s*

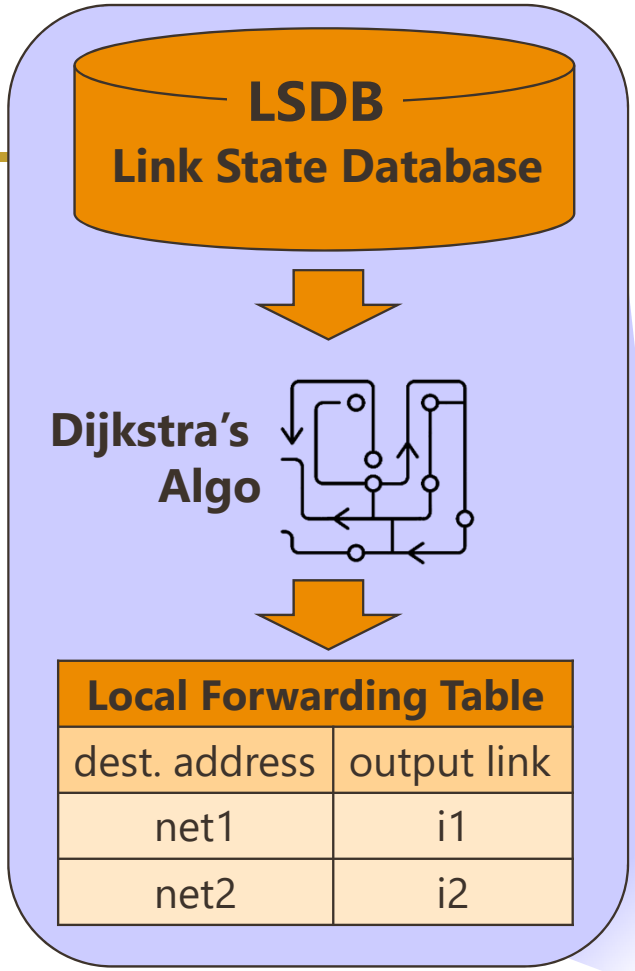


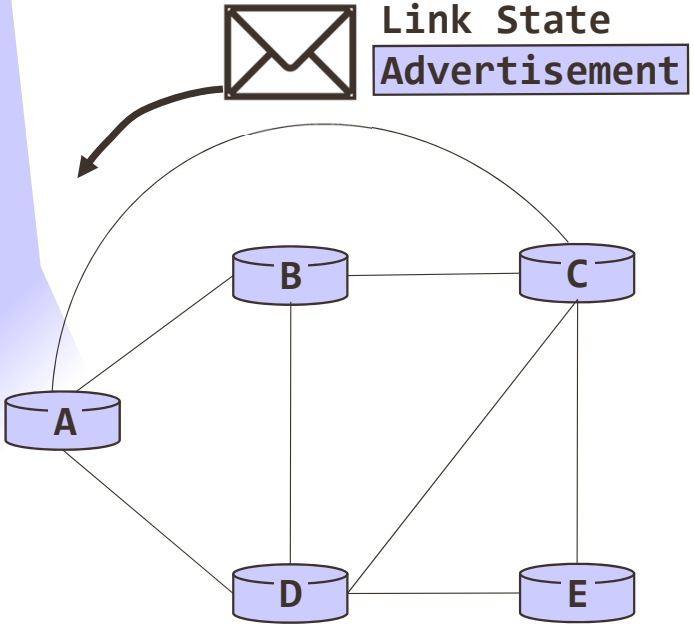
OSPF

Curtin University

Open Shortest Path First

- ✓ Uses **linked state routing**
- ✓ Faster convergence than distance vectors
- ✓ Low bandwidth requirements
- ✓ Supports **CIDR**, **VLSM** & **authentication**
- ✓ Hierarchical design using **"areas"** for larger networks





OSPF 3 Step Process

1. Initializing Bi-directional Communication

- Two routers running OSPF on the same link initialize a bi-direction communication via exchanging HELLO messages

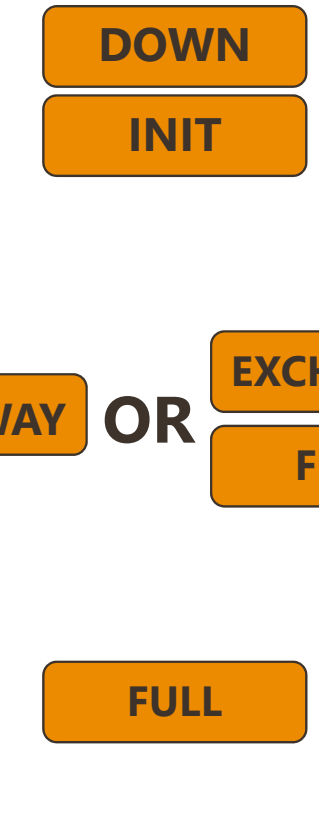
2. Become Neighbors

- Two routers running OSPF on the same link **may not** form a neighbor relationship, **remain in TWO-WAY** state
- **Or,** Become fully adjacent and **EXCHANGE LSAs**

3. Choose the Best Routes

- Each router chooses the best routes running link-state algorithm on their local database

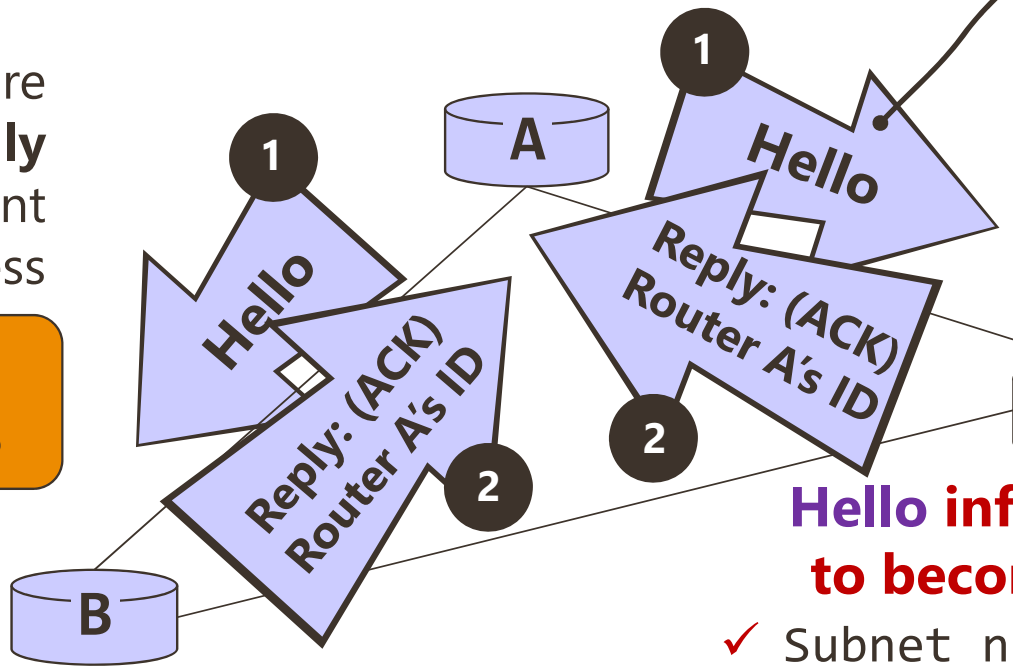
OSPF States



OSPF INIT

Neighbors are **discovered dynamically** using **hello packets** sent to multicast address

`Multicast 224.0.0.5`

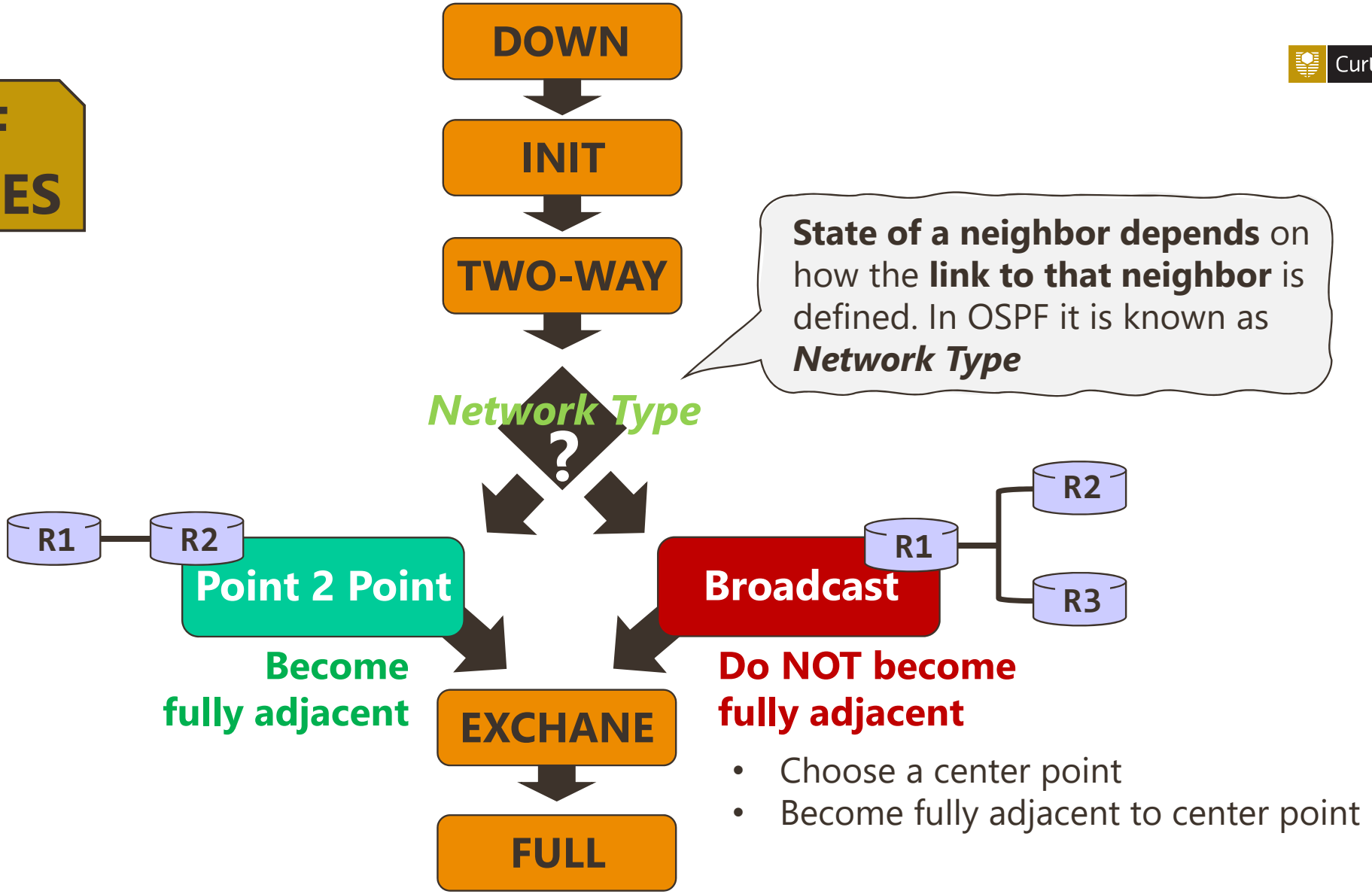


Hello info. must match to become neighbors

- ✓ Subnet number & Subnet mask
- ✓ AreaID
- ✓ Hello & Dead Interval
- ✓ Authentication
- ✓ Stub Area Flags

Curtin University

OSPF STATES

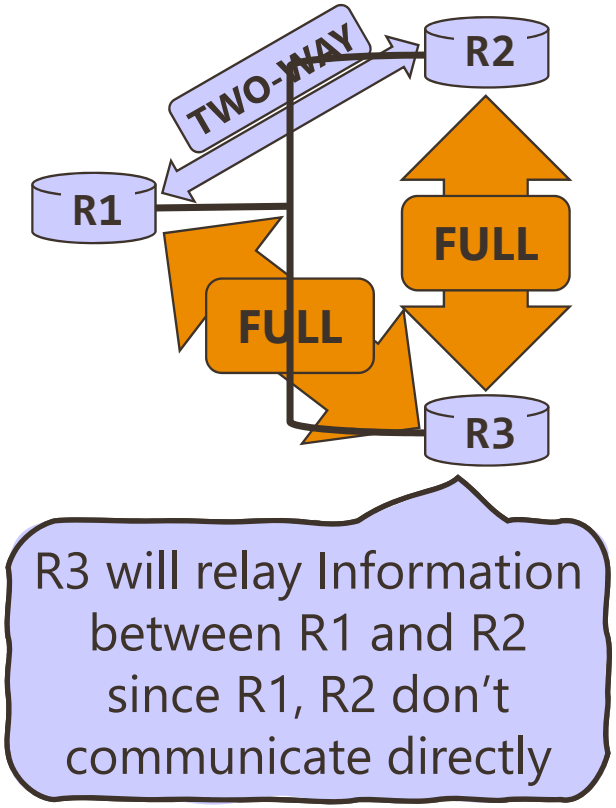


- Choose a center point
- Become fully adjacent to center point

Network Type: Broadcast

Curtin University

- **Do NOT become fully adjacent to each other**
- **Choose** a center point **(Designated Router - DR)**
  - ✓ based on HELLO message's `Router Priority` and/or `RouterID` fields
- Each router in the broadcast network segment will **become fully adjacent only to Designated Router (DR)**
- A **Backup Designated Router (BDR)** is also chosen in case primary DR fails



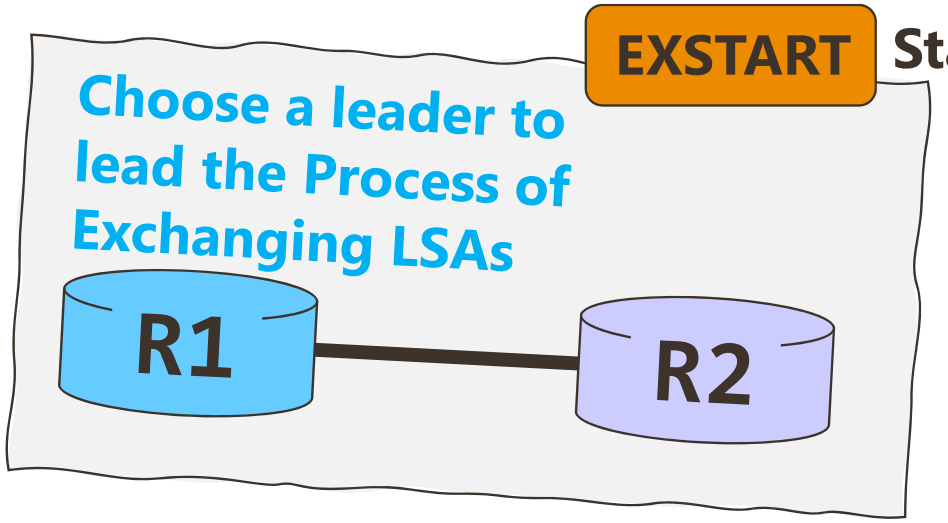
Network Type: Point 2 Point OR Broadcast

Curtin University

In **EXCHANGE** State

- Exchange database description (summary of LSDB, not entire LSDB)
- Request part of the LSAs
- Once both have the same LSDB

EXSTART State



FULL State

- Use **Dijsktra's Algorithm** on LSDB **-> Best Routes**
- **Best Routes -> Routing Table**

Curtin University

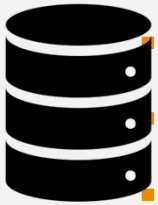
IN FULL STATE

- Continue to exchange hello packets (heartbeat) - **HELLO Interval**
- Declare dead if not heard in **Dead Interval**
- Continue to exchange ***LSAs Every 30 min***
- **If network failure**, send updates ***immediately***

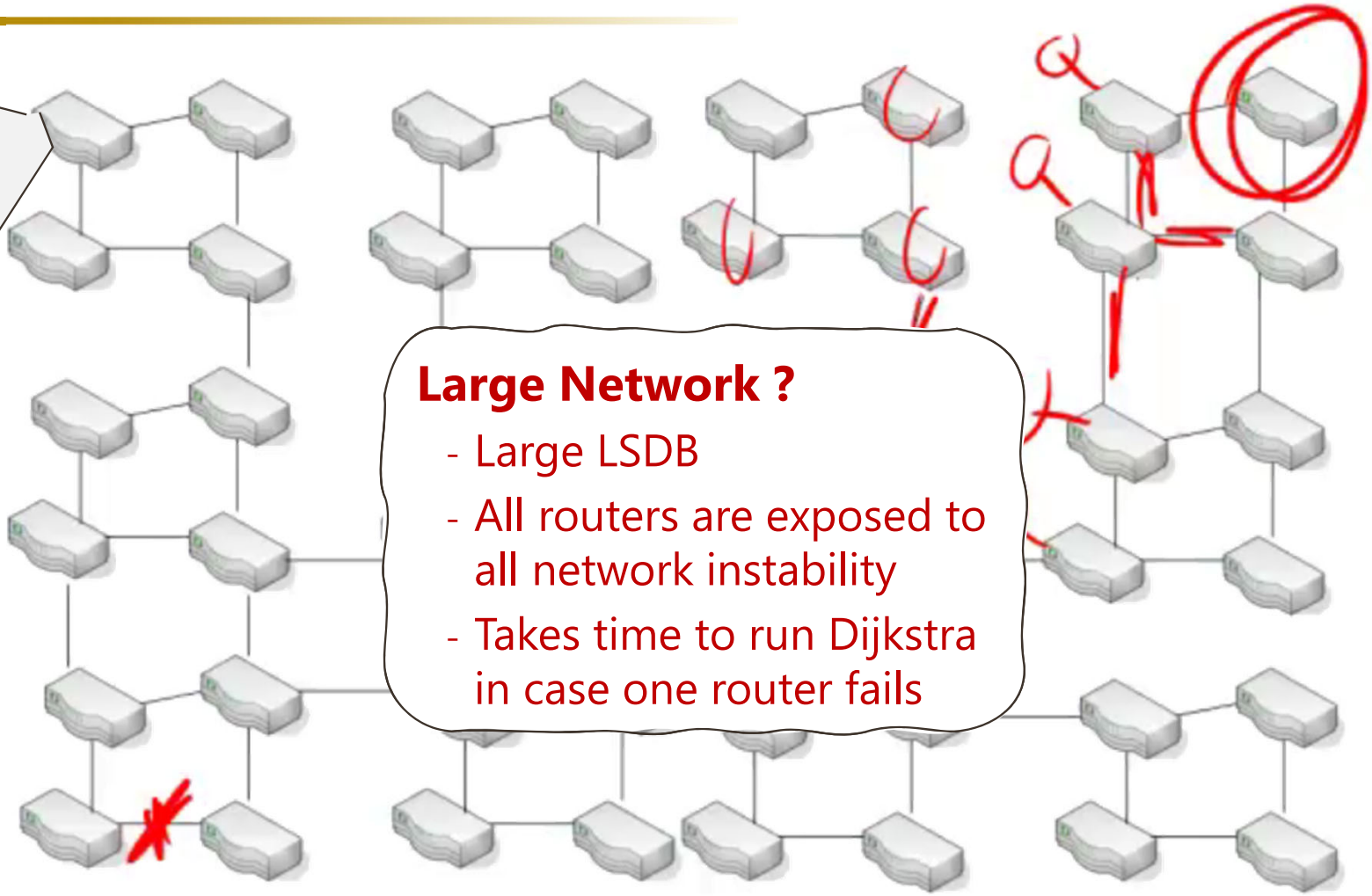
OSPF Design

Curtin University

LSDB includes



- Links
- Routers
- Subnets

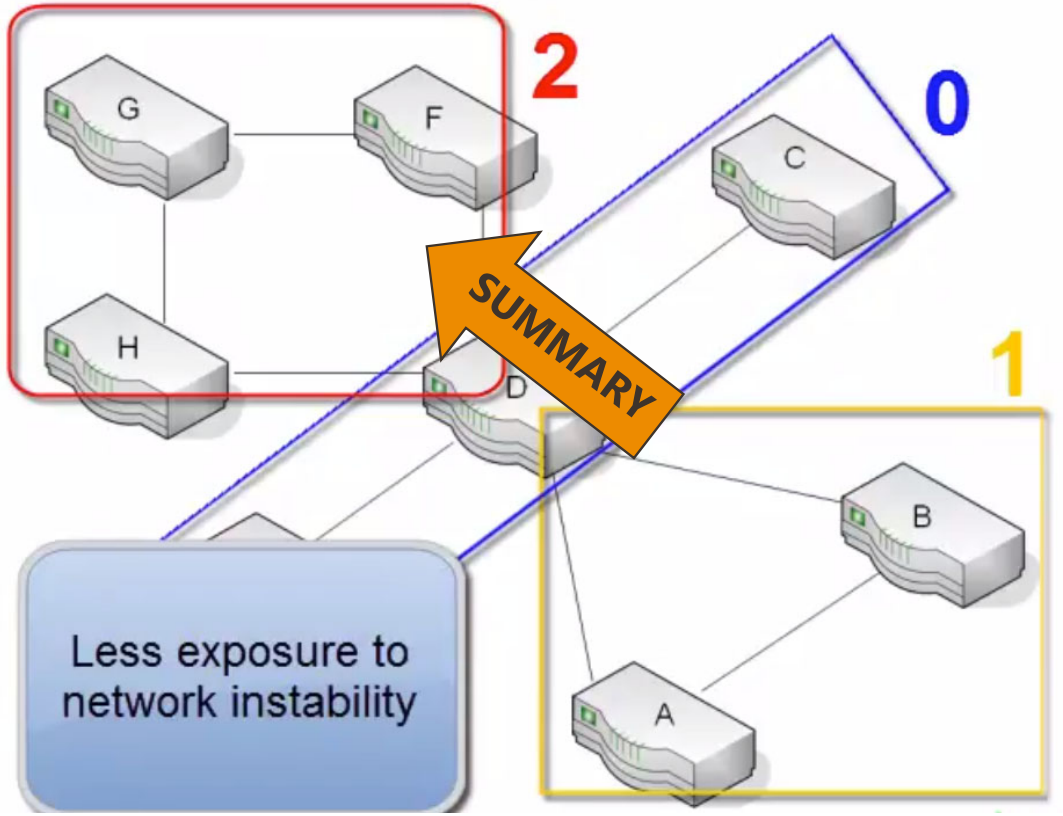


Large Network ?

- Large LSDB
- All routers are exposed to all network instability
- Takes time to run Dijkstra in case one router fails

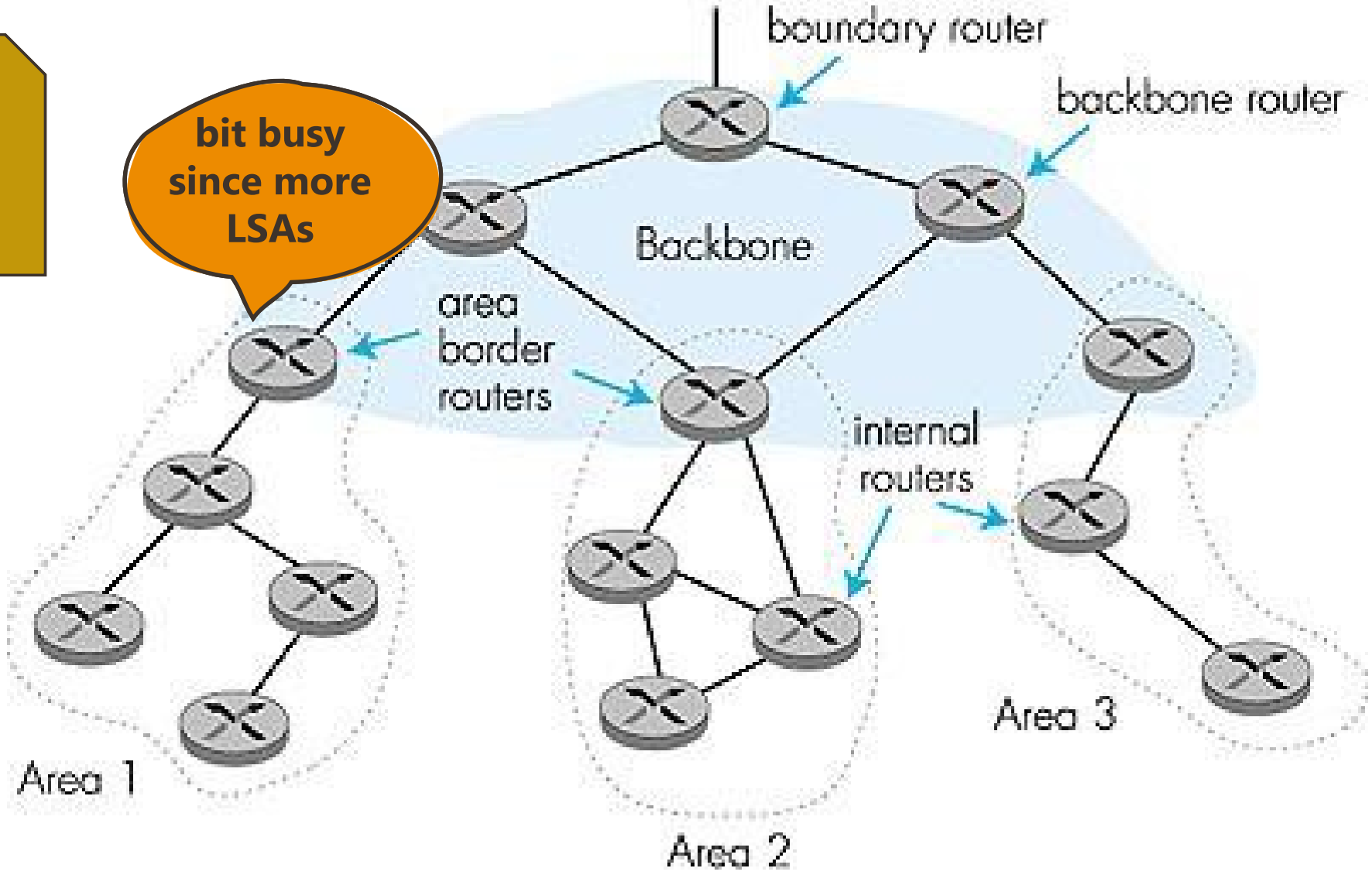
- **Use the concept of Area**
  - Modular approach to split the network
  - Area = Group of routers with same LSDB
  - **Area 0 = Backbone Area**
    - All other areas must connect via backbone

If Router A fails,
- Router B, D only will get LSAs
- Reapply Dijsktra Algorithm



- Router B is **only** fully aware of Router A, D
  - But it can still route to Router G
  - Router B has summary information of Area 2
  - Smaller LSDB

OSPF Hierarchical Design



Summary

Curtin University

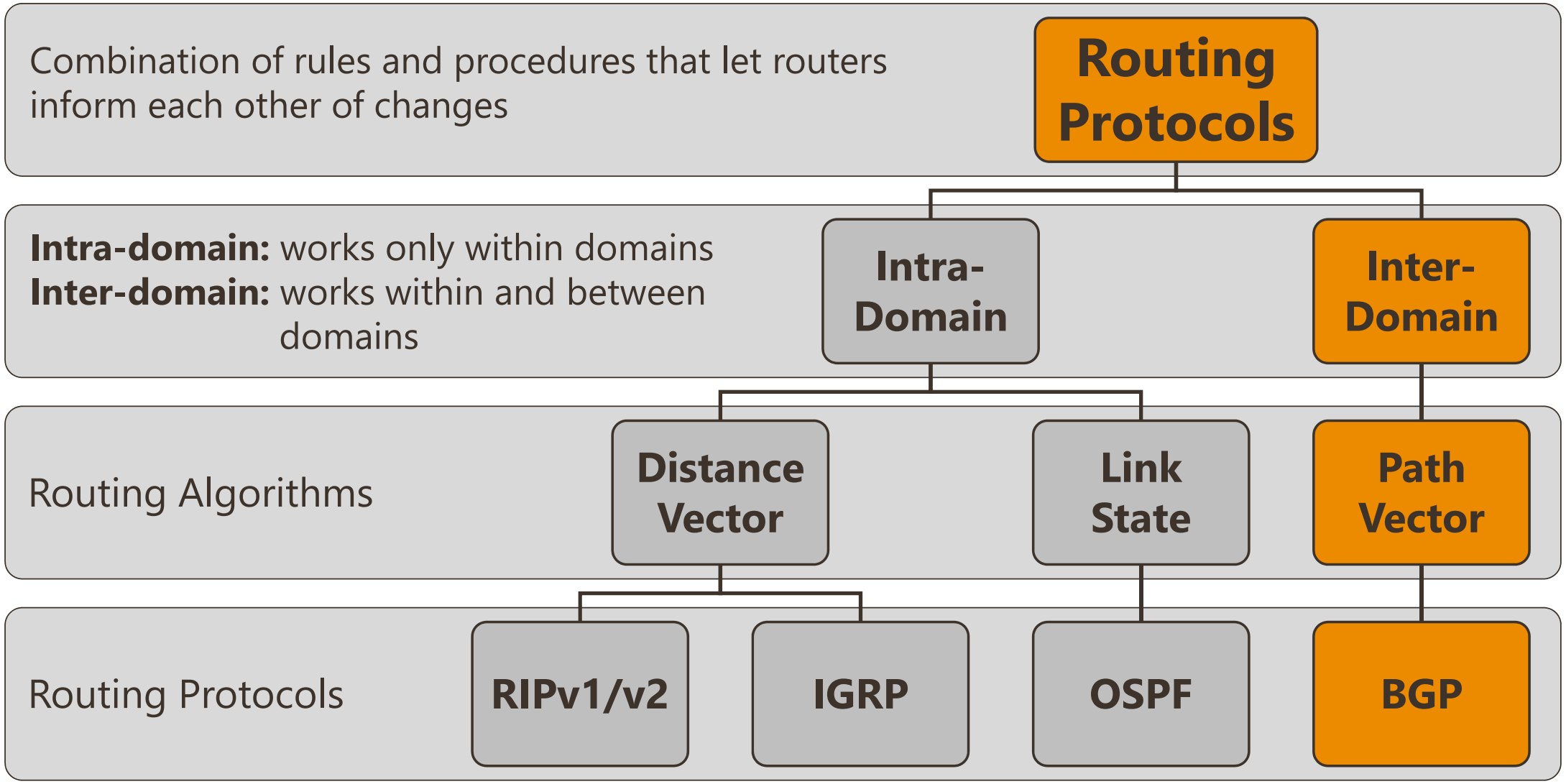
RIP	OSPF
Distance Vector (DV) Routing	**Link State Routing**
Metric: Hop Count	Bandwidth, Delay
Best Path Calculation: Bellman Ford Formula	SPF (e.g. Dijstrka) Alogorithm
Routing: Networks are not divided into areas or multiple tables	**Routing:** done with Autonomous Systems, Areas, Stub Areas and Backbone Areas
Maximum Hop Count: 15	**No hop count**

Curtin University

Inter-Domain Routing

- BGP (Border Gateway Protocol)
- IDRP (Inter-Domain Routing Protocol)

Combination of rules and procedures that let routers inform each other of changes



Hierarchical routing

Curtin University

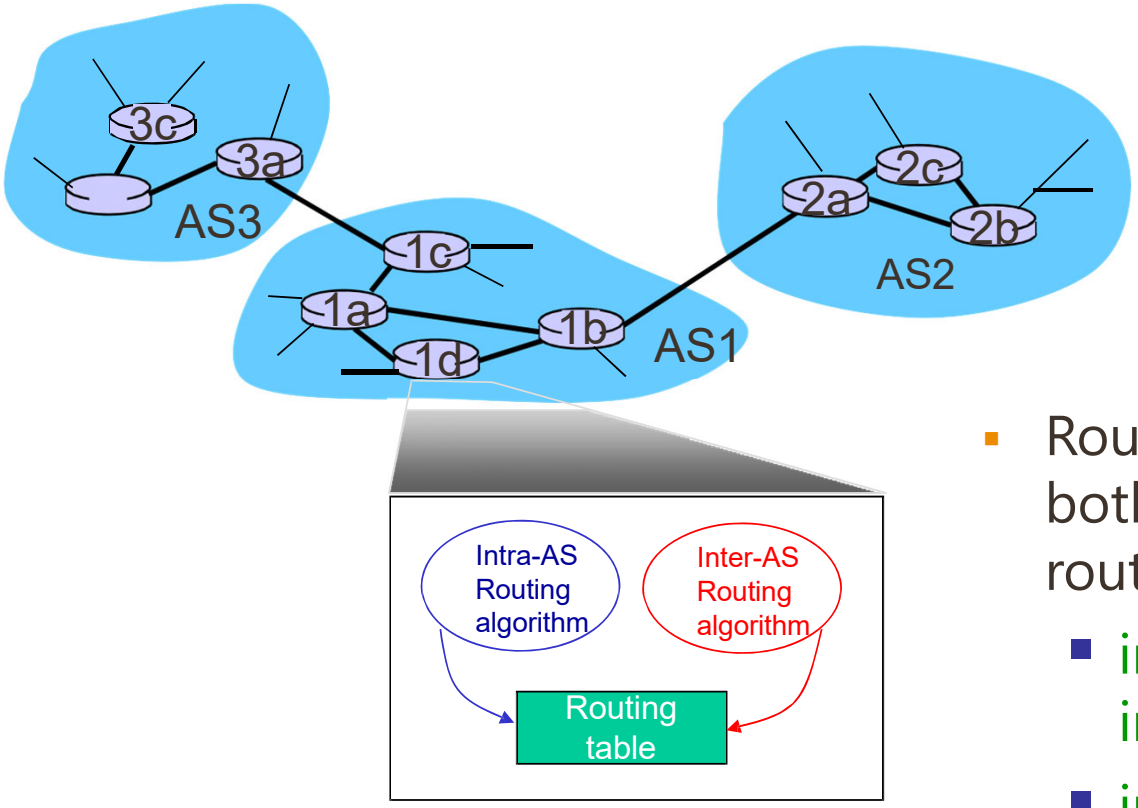
- Aggregate routers into regions, "autonomous systems" (AS)
- Routers in the same AS run the same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

Gateway router:

- at "edge" of its own AS
- has link to router in another AS

Interconnected AS's

Curtin University



- Routing table configured by both intra-AS and inter-AS routing algorithm
  - intra-AS sets entries for internal dests
  - inter-AS & intra-AS sets entries for external dests

Inter-AS tasks

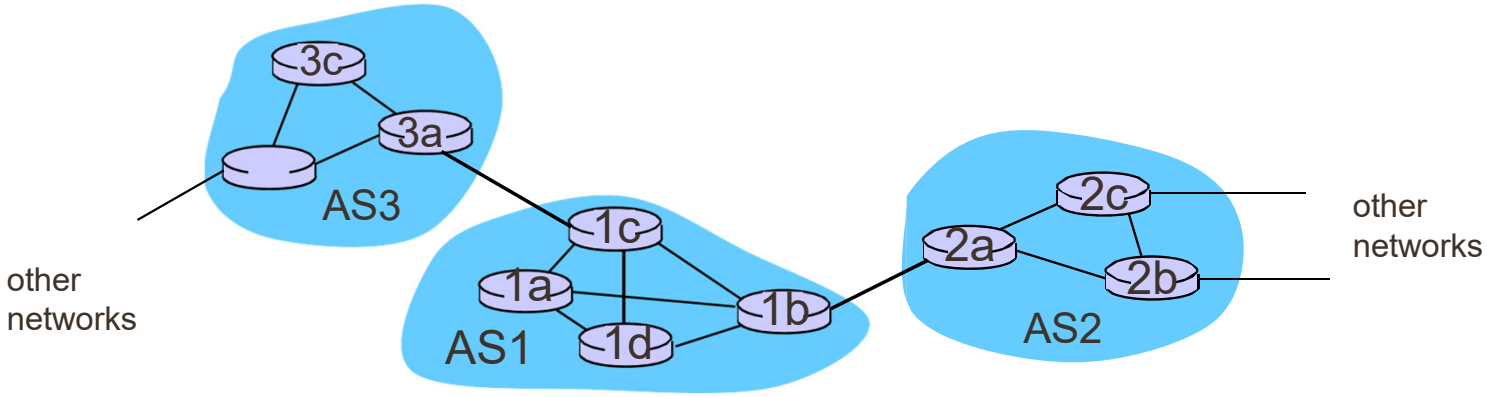
Curtin University

- Suppose router in AS1 receives datagram destined outside of AS1:
  - Router should forward packet to gateway router, but which one?

AS1 must:

1. learn which destinations are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!



Distant Vector and Link State

Curtin University

- distance-vector protocol is ineffective
  - Not all routers uses the same distance metric.
  - different AS's have different priorities and may have restrictions on other AS. No information is given about the AS that will be visited along a route.
- link-state is also unsuitable
  - flooding to all router across multiple AS's is unmanageable
  - metrics used in different AS's can be different

Path Vector Routing

Curtin University

- An alternative: path-vector routing
  - dispense with distance metrics.
  - Simply provide information about which network can be reached by a given router and the AS's that must be crossed to get there.
  - since a complete list of AS's traversed by a rou is provided in the path vector, it allows policy routing.

Inter vs Intra Domain Routing

Curtin University

Policy:

- inter-AS: admin wants control over how its traffic is routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

Scale:

- hierarchical routing saves table size, reduced update traffic

Performance:

- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance

BGP(Border Gateway Protocol)

Curtin University

- The de facto inter-domain routing protocol
- BGP provides each AS a means to:
  - **eBGP:** obtain subnet reachability information from neighboring AS's.
  - **iBGP:** propagate reachability information to all AS-internal routers.
  - determine "good" routes to other networks based on reachability information and policy.
- allows subnet to advertise its existence to the rest of the Internet: "I am here"

IDRP (Inter-Domain Routing Protocol)

Curtin University

- Designated for use with IPv6. It is a super set of BGP's functions.
- An ISO standard within the OSI family, but not dependent on OSI networking.
- Can deal with multiple internet protocols and address schemes.
- Allows confederations: aggregates of autonomous systems which can be viewed externally as a single entity. This can be done hierarchically, allowing scalable routing.

BGP vs IDRP

Curtin University

- IDRP is not tied to TCP, as BGP is.
- BGP uses 16-bit AS numbers. IDRP uses various- length identifier
- IDRP can deal with multiple internet protocols and address schemes.
- BGP specify the complete list of AS that a path visits. IDRP allows confederations. (Most important difference).

Curtin University

- **Network Layer**
  - Routing Fundamentals
  - Classification of Protocols
  - Routing Table
- **Routing Algorithms**
  - Fundamentals
  - Link State Routing
    - Topology Dissemination
    - Computing Shortest Path (Dijkstra)
  - Distance Vector Routing
    - Bellman Ford Algorithm
    - Distance Vector Updates

- **Intra-domain Routing**
  - RIP
    - rip updates
    - route poisoning
    - count-to-infinity problem
    - redundancy-in-networks problem
    - rip timers
    - RIPv1, RIPv2, RIPng
  - OSPF
    - 3-step-process
    - init-state
    - OSPF-states
    - two-way-state-to-full-state
    - full-state
    - OSPF hierarchical design

- **Inter Domain Routing**
  - Path Vector Routing
  - BGP
  - IDRP


Curtin University
314
THANK YOU
Make tomorrow better.