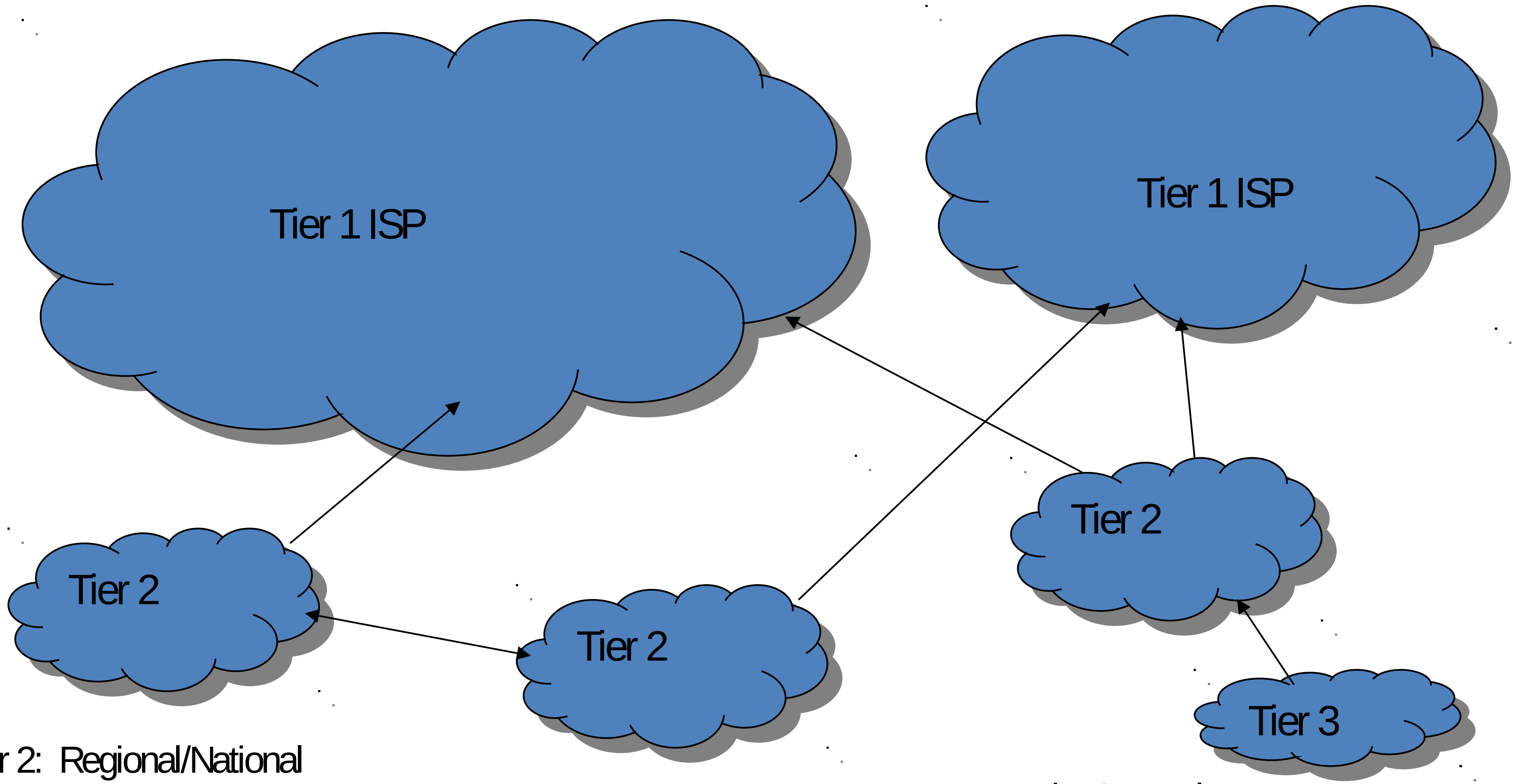
Computer Networks: Network Layer

BITS Pilani
Hyderabad Campus

Acknowledgement: Slides and Images adapted from Kurose, and Forouzan (TMH)

Chittaranjan Hota
PhD (CSE)

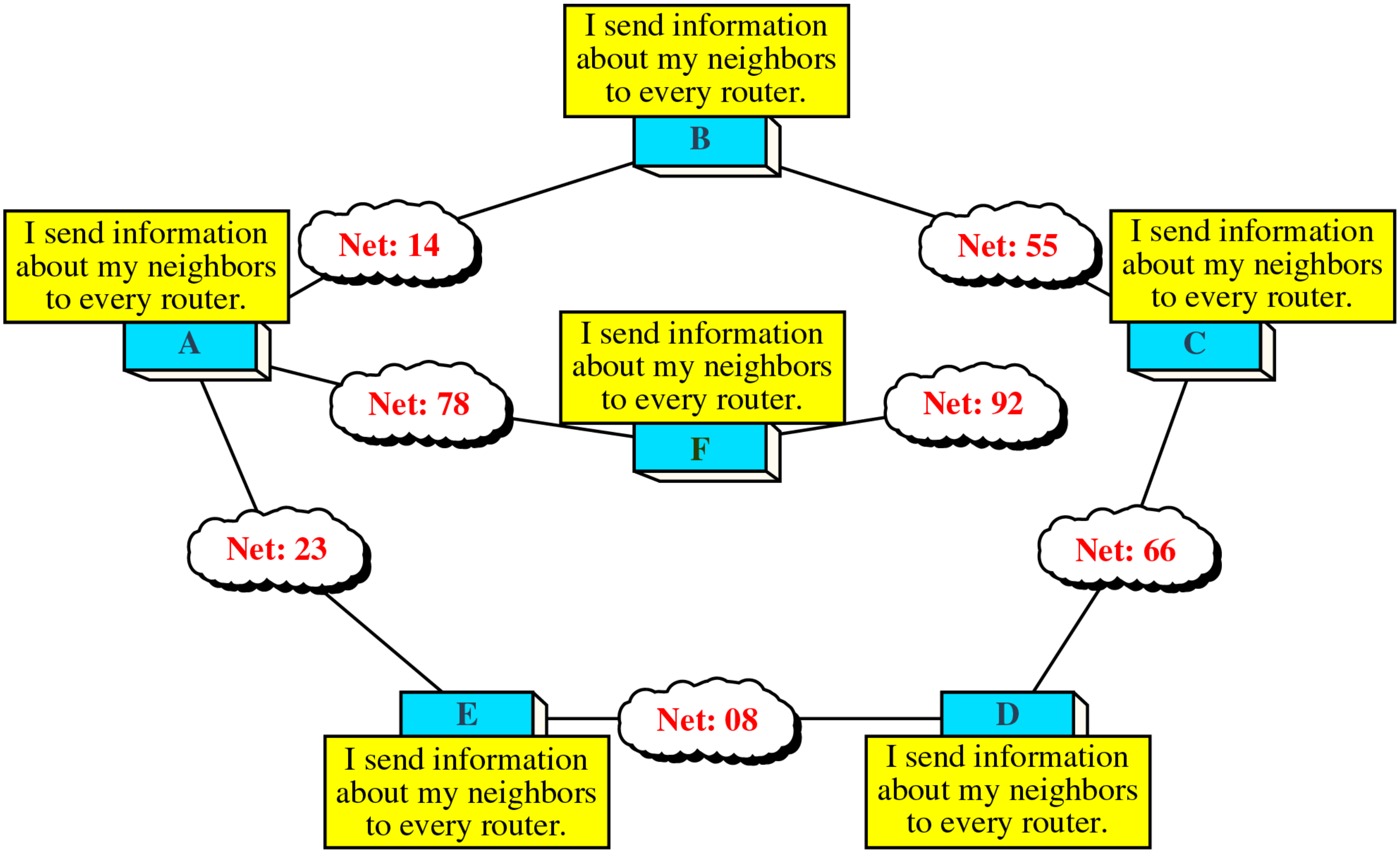
Internet Routing Architecture



Tier 2: Regional/National

Tier 3: Local

Concept of Link State Routing



Dijkstra's Shortest-Path Algorithm

- Iterative algorithm
  – After k iterations, know least-cost path to k nodes
- **S**: nodes whose least-cost path definitively known
  – Initially, **S = {u}** where u is the source node
  – Add one node to S in each iteration
- **D(v)**: current cost of path from source to node v
  – Initially, **D(v) = c(u,v)** for all nodes v adjacent to u
  – … and **D(v) = ∞** for all other nodes v
  – Continually update D(v) as shorter paths are learned

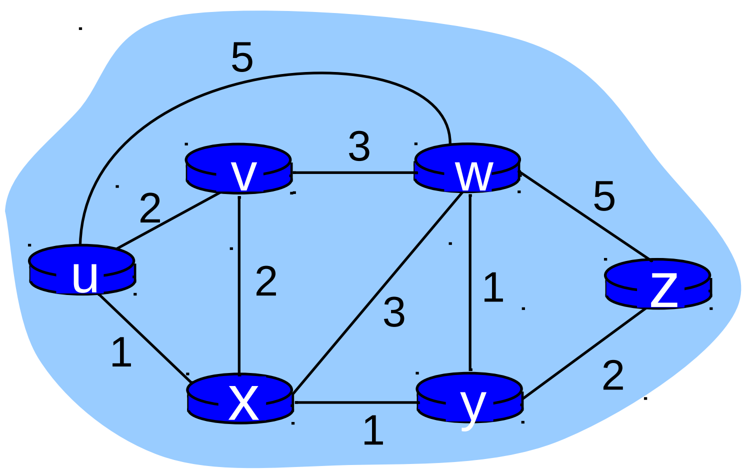
Dijsktra's Algorithm

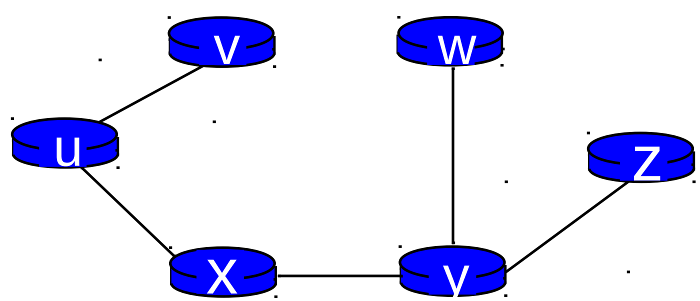
```
Initialization:
  S = {u}
  for all nodes v
    if v adjacent to u {
        D(v) = c(u,v)
    else D(v) = ∞

 Loop
   find w not in S with the smallest D(w)
    add w to S
    update D(v) for all v adjacent to w and not in S:
        D(v) = min{D(v), D(w) + c(w,v)}
 until all nodes in S
```

Dijkstra’s algorithm: example1

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					





Resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

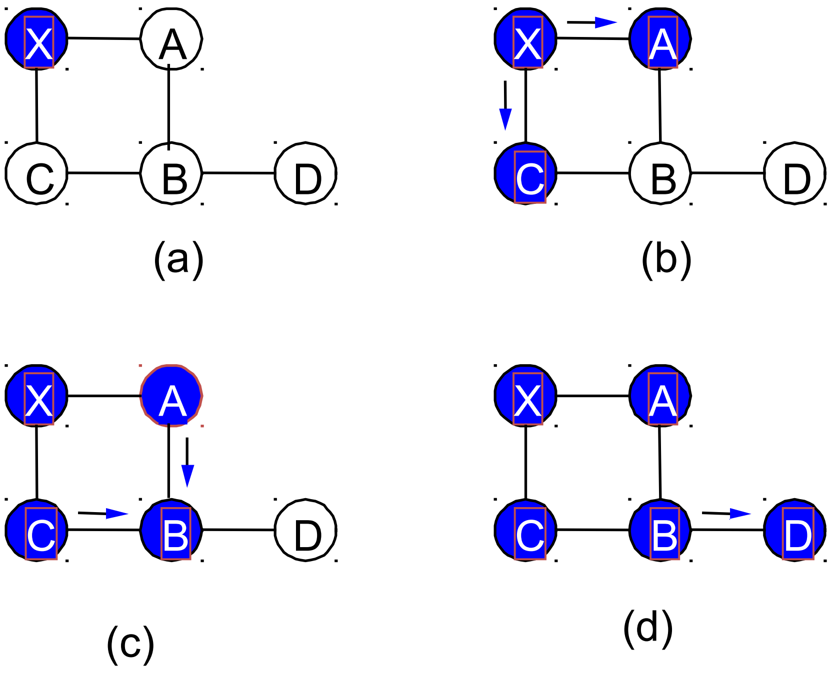
Link-State Routing

- Each router keeps track of its incident links
  - Whether the link is up or down
  - The cost on the link
- Each router broadcasts the link state
  - To give every router a complete view of the graph
- Each router runs Dijkstra’s algorithm
  - To compute the shortest paths
  - … and construct the forwarding table
- Example protocols
  - Open Shortest Path First (OSPF)

Detecting Topology Changes

- Beaconing
  - Periodic “hello” messages in both directions
  - Detect a failure after a few missed “hellos”

Broadcasting the Link State



Broadcasting the Link State

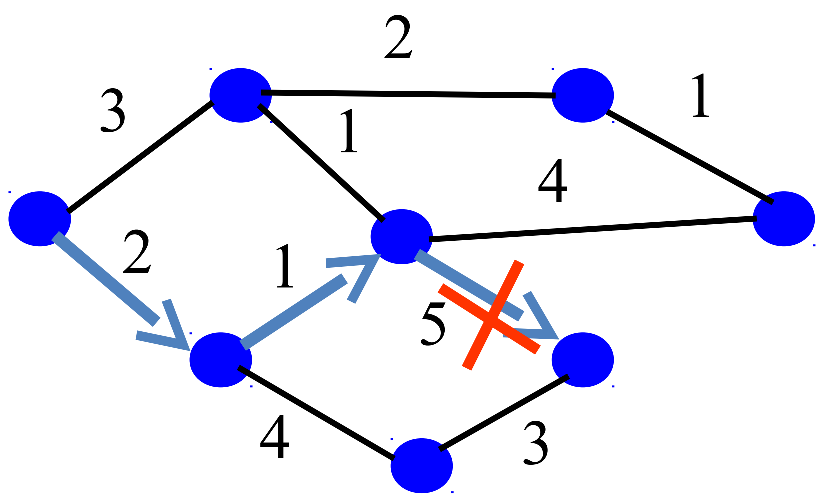
- Reliable flooding
  - Ensure all nodes receive link-state information
  - … and that they use the latest version
- Challenges
  - Packet loss
  - Out-of-order arrival
- Solutions
  - Acknowledgments and retransmissions
  - Sequence numbers
  - Time-to-live for each packet

When to Initiate Flooding

- Topology change
  - Link or node failure
  - Link or node recovery
- Configuration change
  - Link cost change
- Periodically
  - Refresh the link-state information
  - Typically (say) 30 minutes

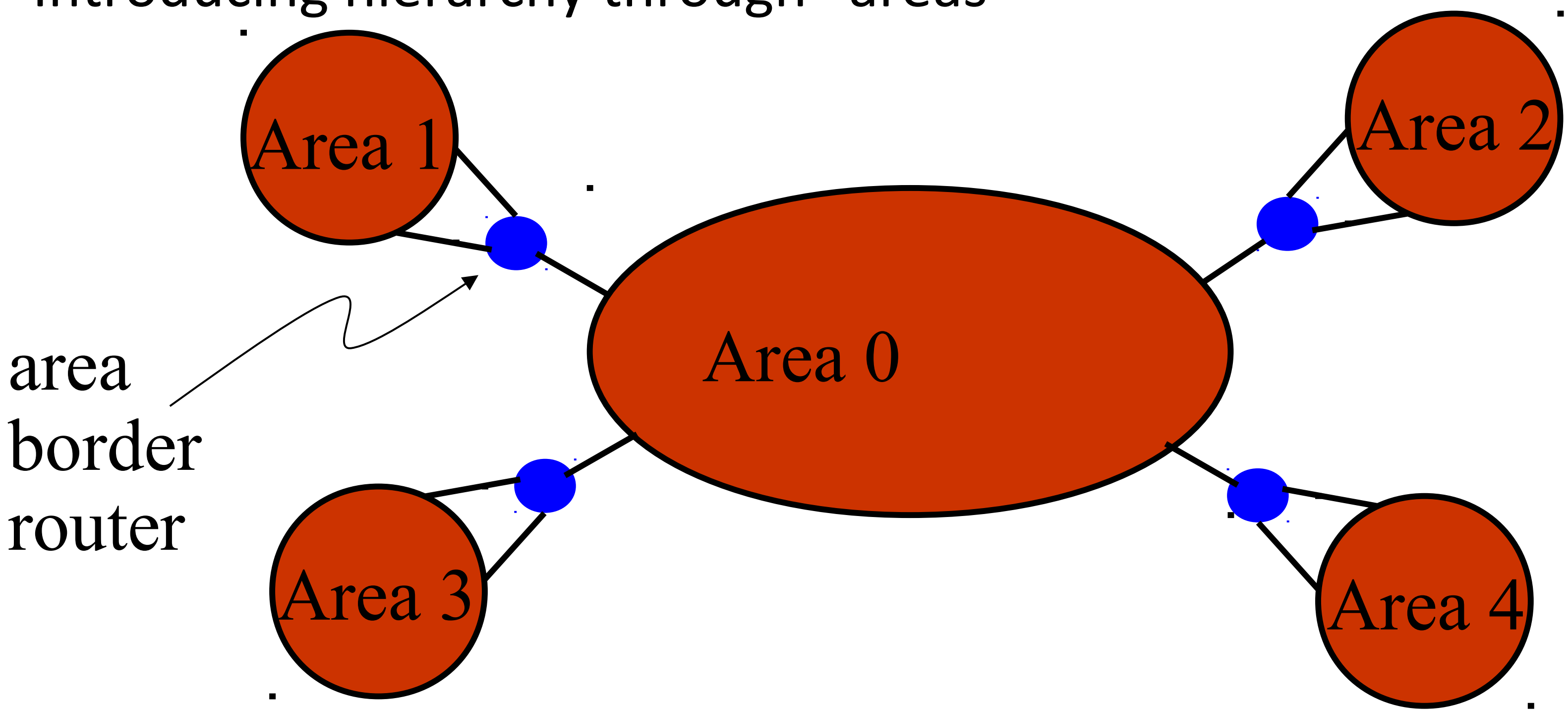
Transient Disruptions

- Detection delay
  - A node does not detect a failed link immediately
  - … and forwards data packets into a “blackhole”
  - Depends on timeout for detecting lost hellos

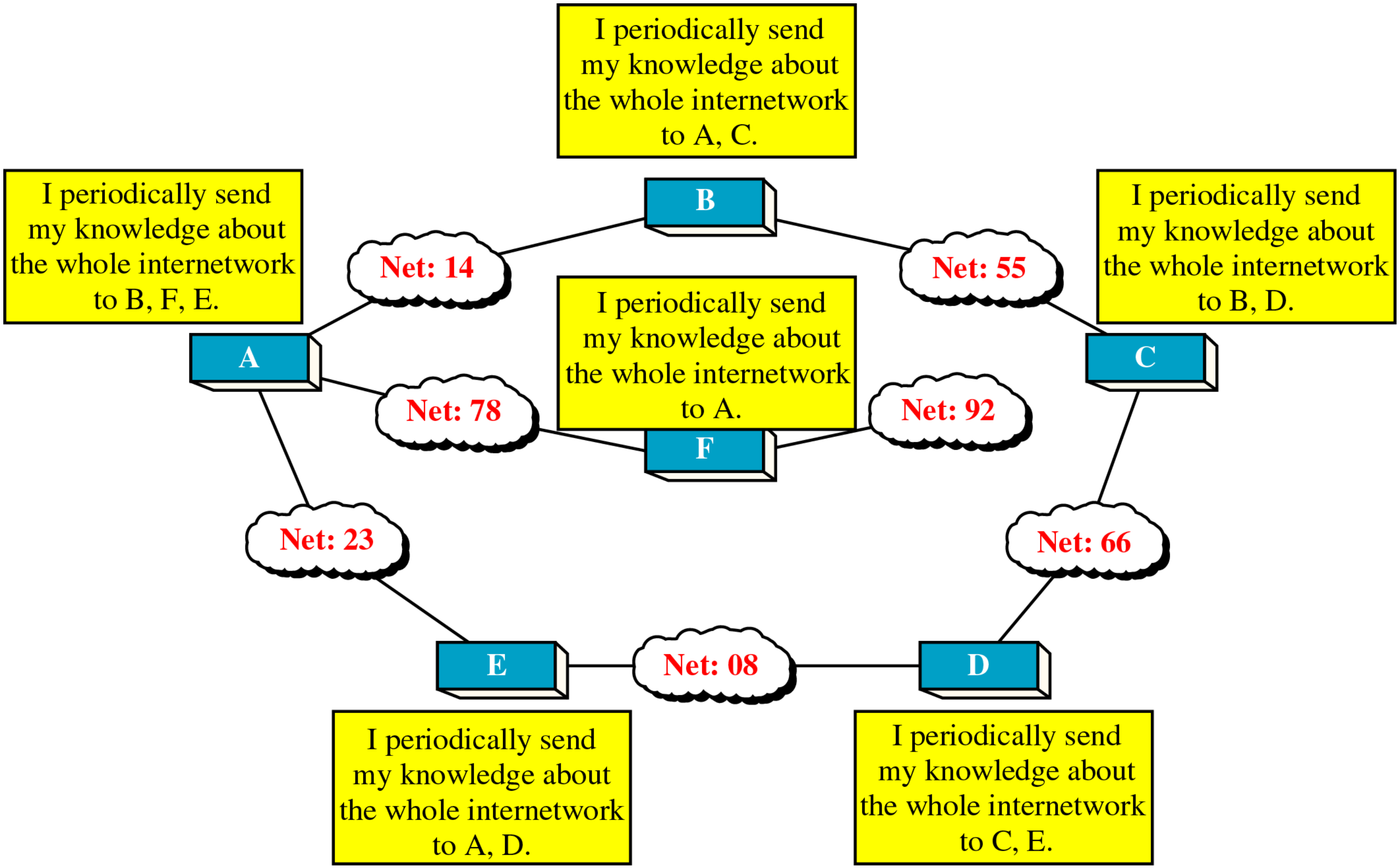


Scaling Link-State Routing

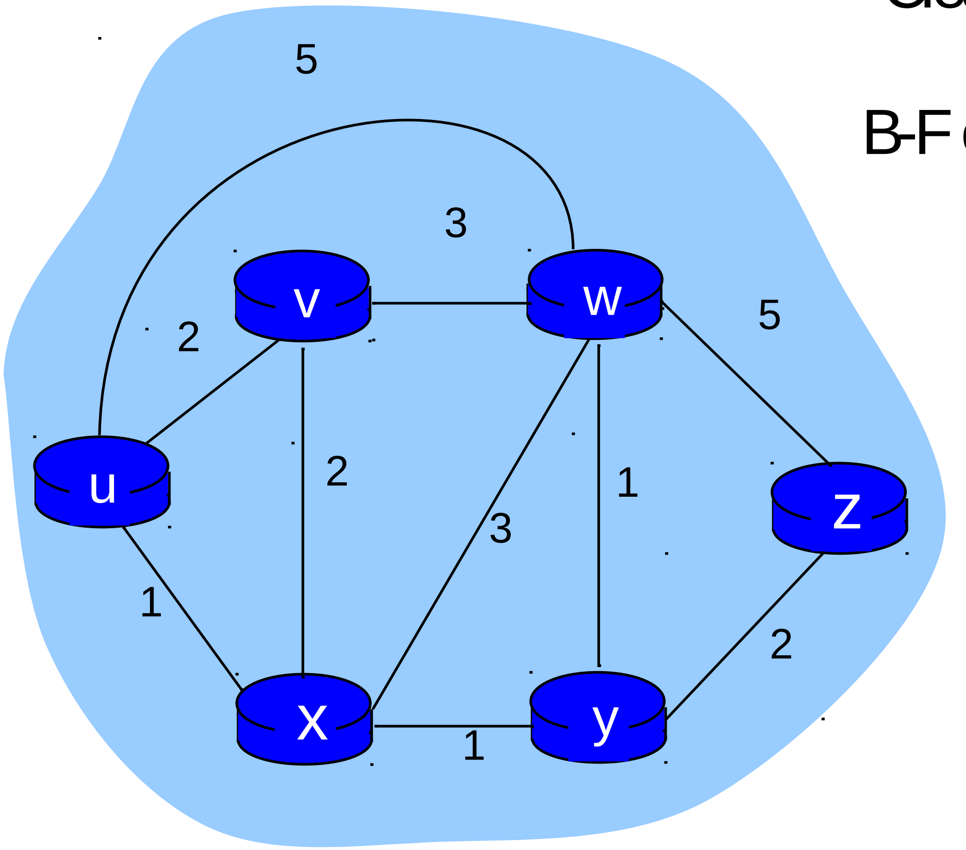
- Overhead of link-state routing
  - Flooding link-state packets throughout the network
  - Running Dijkstra's shortest-path algorithm
- Introducing hierarchy through "areas"



Concept of Distance Vector Routing



Bellman-Ford example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$
\begin{aligned}
d_u(z) = \min \{ & c(u,v) + d_v(z), \\
& c(u,x) + d_x(z), \\
& c(u,w) + d_w(z) \} \\
= \min \{ & 2 + 5, \\
& 1 + 3, \\
& 5 + 3 \} = 4
\end{aligned}
$$

Distance Vector Algorithm

- Bellman-Ford algorithm
- Repeat

  For every node x

    For every neighbor z

      For every destination y

        $d(x,y) \leftarrow \text{Update}(x,y,z)$
- Until converge

Example

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\} = \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\} = \min\{2+1\ ,\ 7+0\} = 3$$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

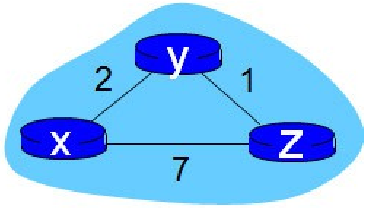
node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time



Example

$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

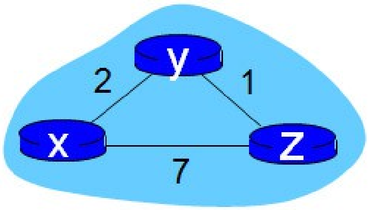
node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

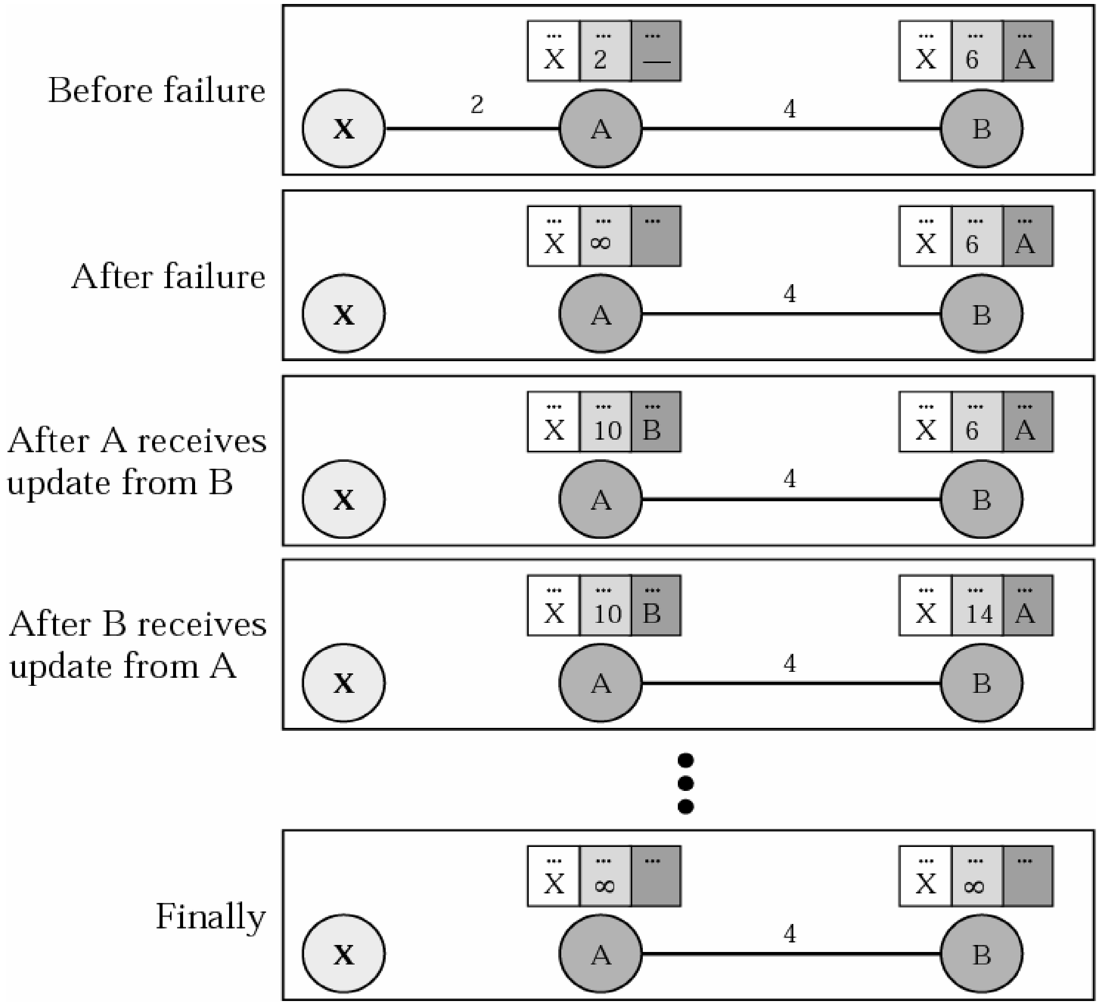
from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

time →



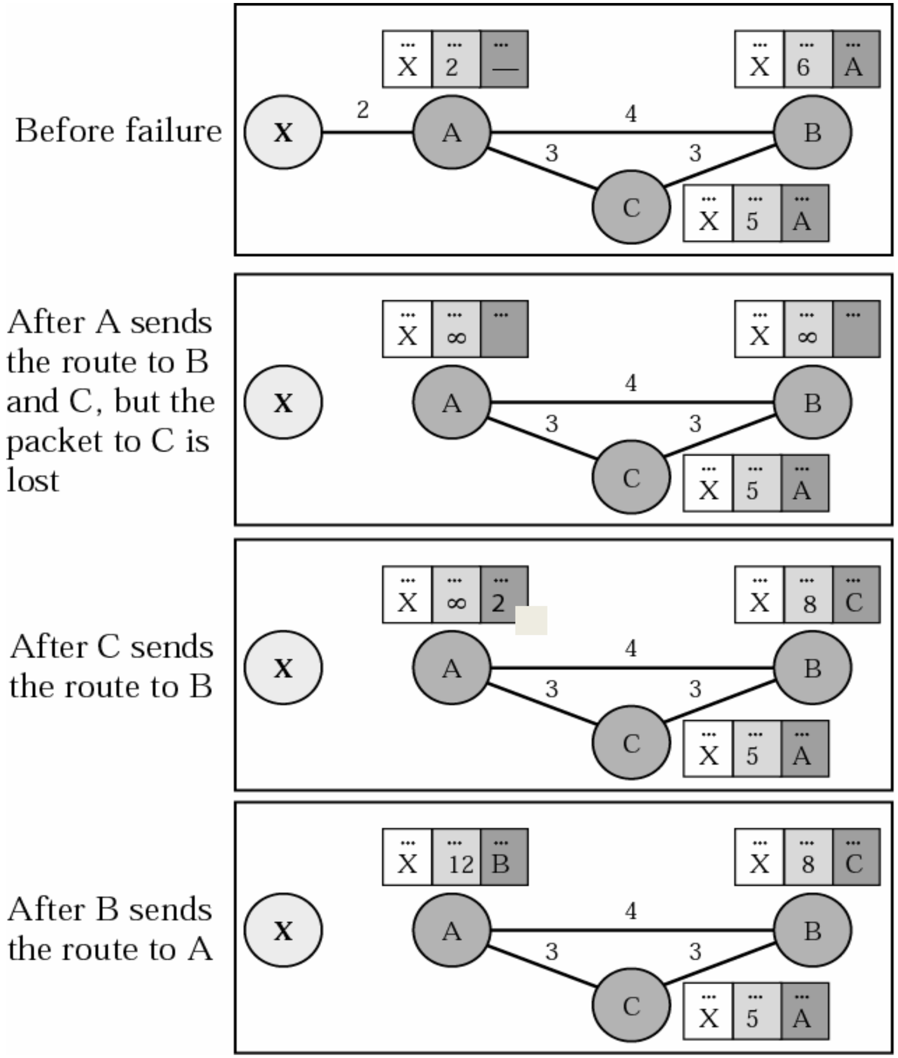
Count to infinity: Two-Node Loop Instability



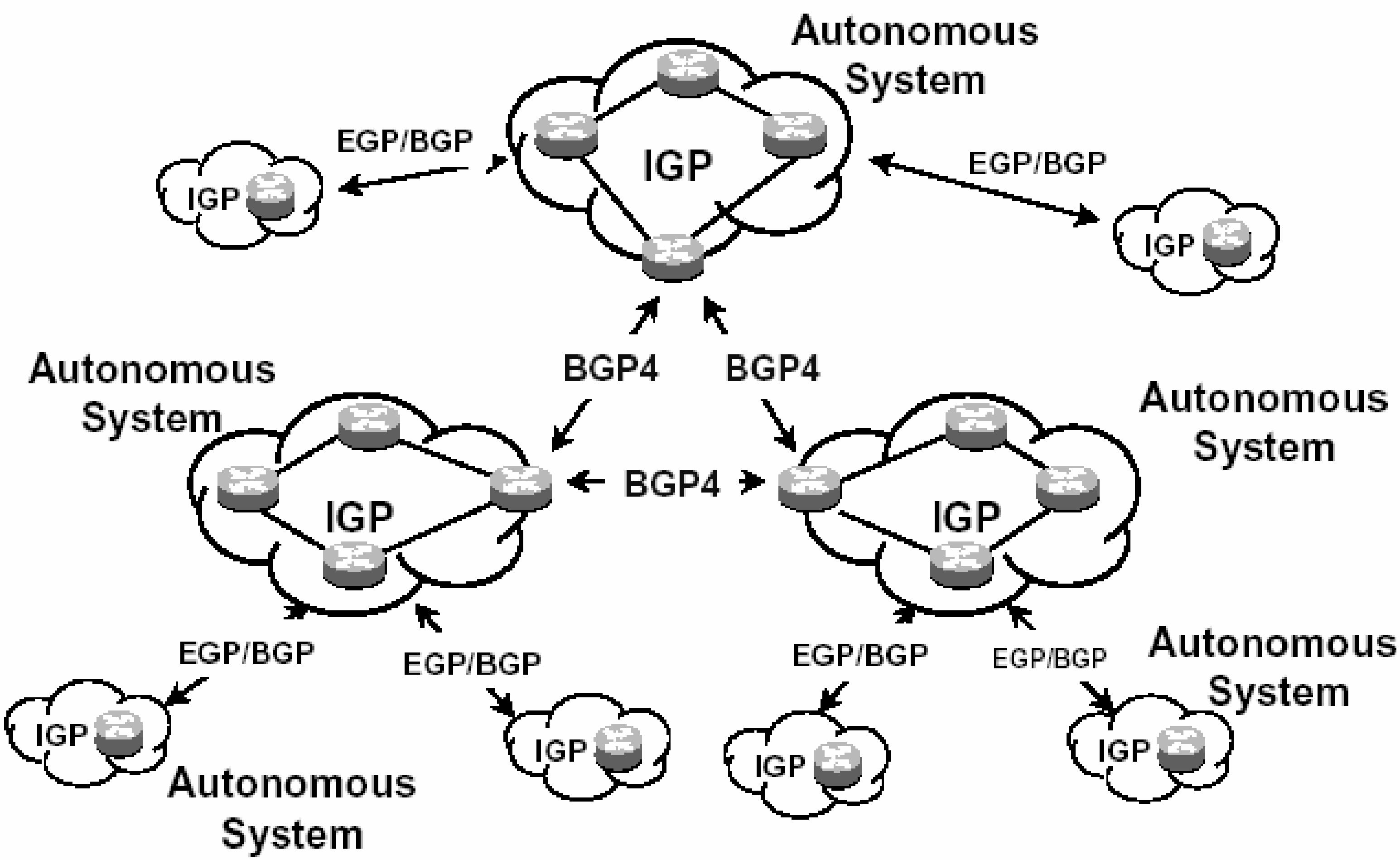
Solutions:

1. Defining infinity
2. Split horizon
3. Split horizon with Poison reverse

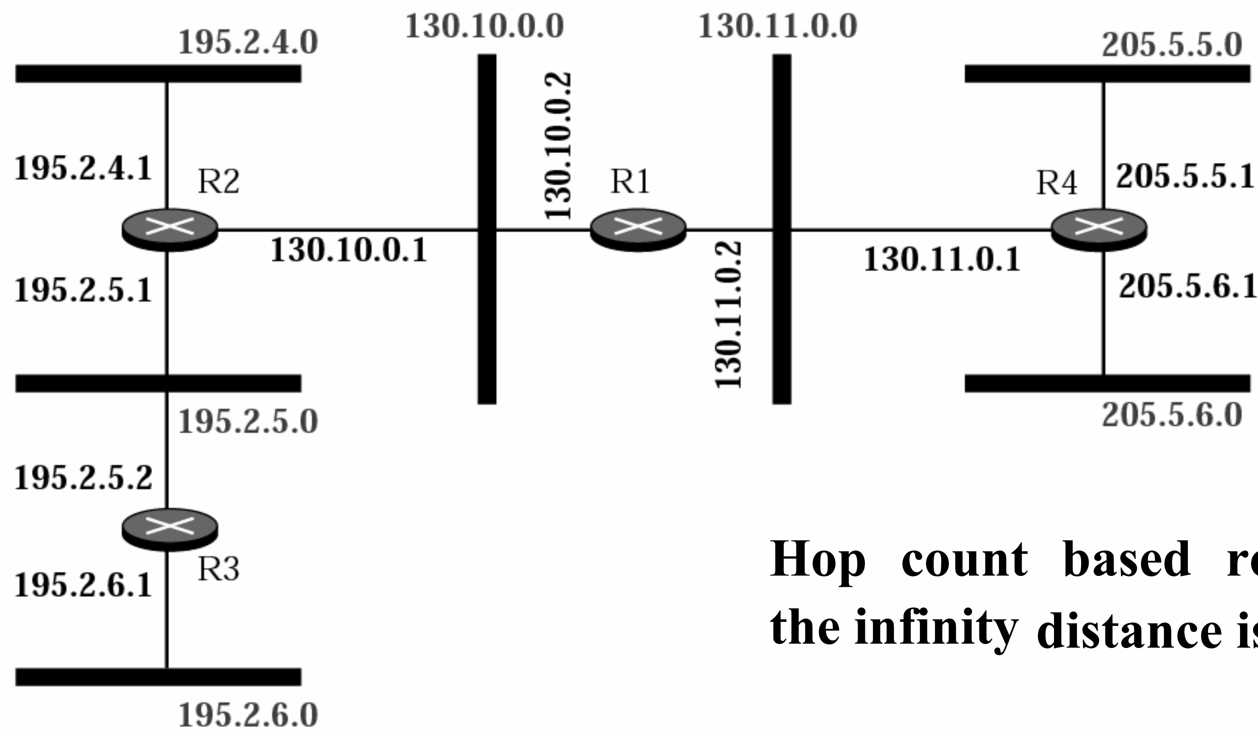
Three-node loop instability



Internet Routing Protocols



Routing Information Protocol (RIP)



Hop count based routing, the infinity distance is defined as 16

Dest.	Hop	Next
130.10.0.0	1	——
130.11.0.0	1	——
195.2.4.0	2	130.10.0.1
195.2.5.0	2	130.10.0.1
195.2.6.0	3	130.10.0.1
205.5.5.0	2	130.11.0.1
205.5.6.0	2	130.11.0.1

R1 Table

Dest.	Hop	Next
130.10.0.0	1	——
130.11.0.0	2	130.10.0.2
195.2.4.0	1	——
195.2.5.0	1	——
195.2.6.0	2	195.2.5.2
205.5.5.0	3	130.10.0.2
205.5.6.0	3	130.10.0.2

R2 Table

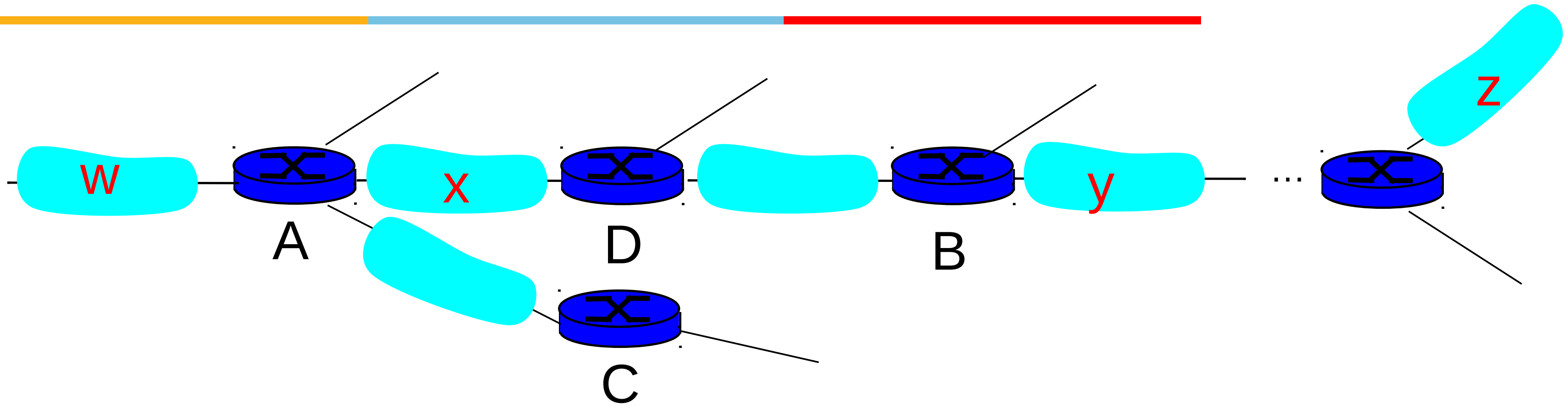
Dest.	Hop	Next
130.10.0.0	2	195.2.5.1
130.11.0.0	3	195.2.5.1
195.2.4.0	2	195.2.5.1
195.2.5.0	1	——
195.2.6.0	1	——
205.5.5.0	4	195.2.5.1
205.5.6.0	4	195.2.5.1

R3 Table

Dest.	Hop	Next
130.10.0.0	2	130.11.0.2
130.11.0.0	1	——
195.2.4.0	3	130.11.0.2
195.2.5.0	3	130.11.0.2
195.2.6.0	4	130.11.0.2
205.5.5.0	1	——
205.5.6.0	1	——

R4 Table

RIP: Example



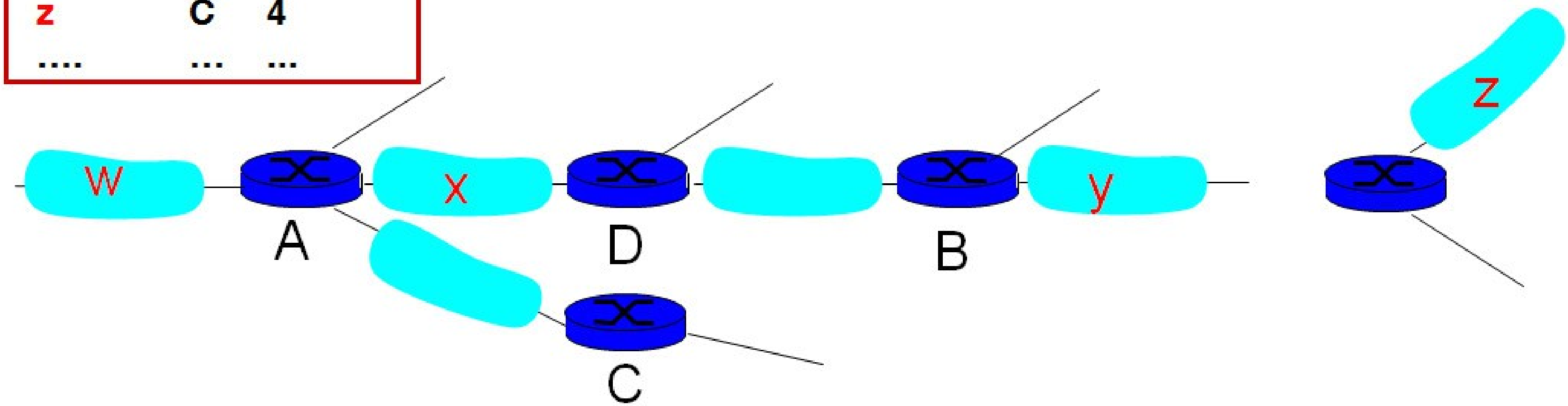
Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

Routing/Forwarding table in D

RIP: Example Continued…

Dest	Next	hops
w	-	1
x	-	1
z	C	4
….	…	…

Advertisement from A to D



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
….	….	….

Routing/Forwarding table in D