Routing Protocols

Lecture 5

Routing protocols

- Routing protocol is used to determine how routing is performed within an autonomous system (AS). Intra-AS routing protocols are also known as **interior gateway protocols**.
- The common Routing protocols are:
  - **Routing Information Protocol (RIP).**
  - **Open Shortest Path First (OSPF)**.
  - **Border Gateway Protocol (BGP).**

Broadcast and Multicast Routing

- **broadcast routing**, the network layer provides a service of delivering a packet sent from a source node to all other nodes in the network.
- **multicast routing** enables a single source node to send a copy of a packet to a subset of the other network nodes.

Broadcast Routing Algorithms

- The sending node to send a separate copy of the packet to each destination.
- Given *N* destination nodes, the source node simply makes *N* copies of the packet, addresses each copy to a different destination, and then transmits the *N* copies to the *N* destinations using unicast routing.
- This is called **N-way-unicast** approach to broadcasting packets.

Broadcast Routing Algorithms

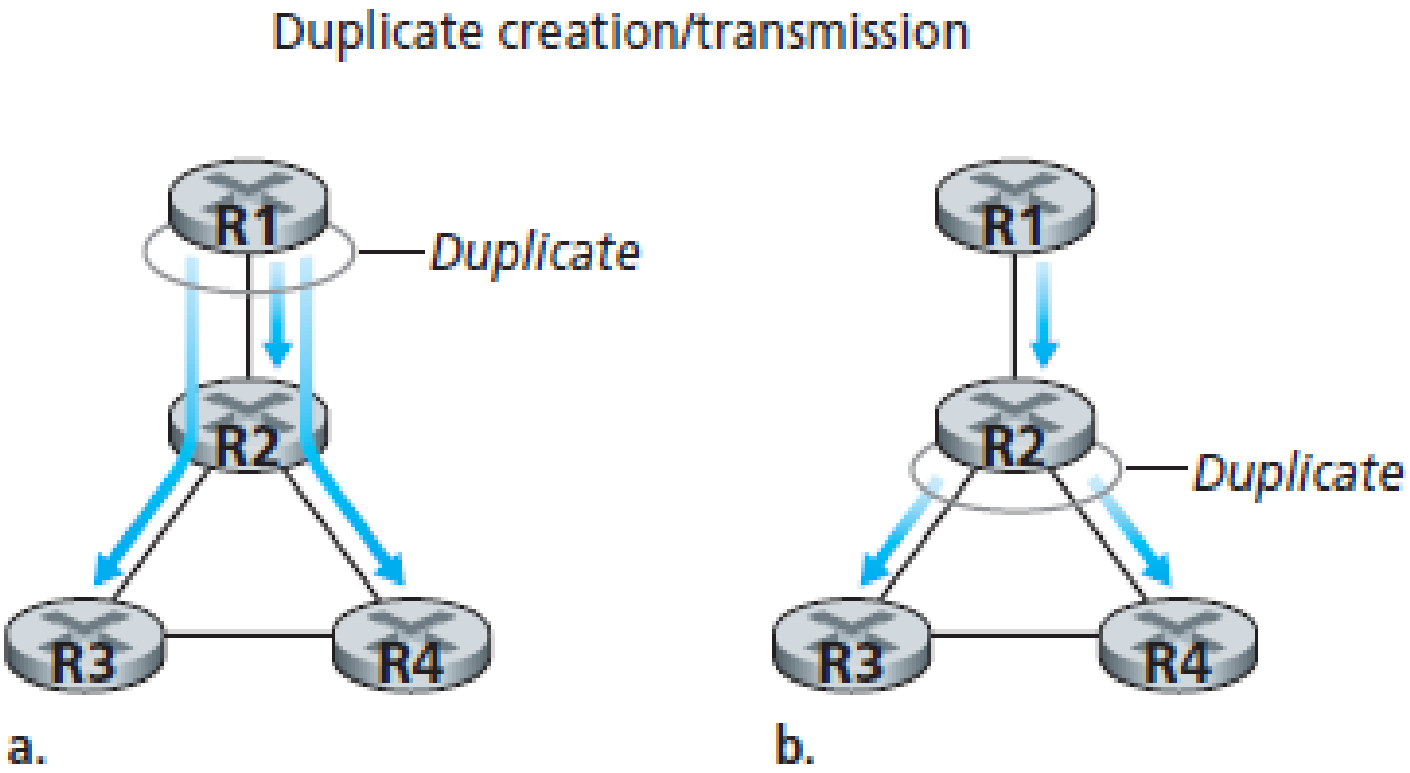


Figure 4.43 ♦ Source-duplication versus in-network duplication

Flooding

- The most obvious and popular technique for achieving broadcast is a **flooding** approach.
- There are two types of flooding:
  - **Uncontrolled Flooding**
  - **Controlled Flooding**

Uncontrolled Flooding

- The source node sends a copy of the packet to all of its neighbours.
- When a node receives a broadcast packet, it duplicates the packet and forwards it to all of its neighbours (except the neighbour from which it received the packet).

Uncontrolled Flooding

- Problems:
  - If the graph has cycles, then one or more copies of each broadcast packet will cycle indefinitely.
  - When a node is connected to more than two other nodes, it will create and forward multiple copies of the broadcast packet, each of which will create multiple copies of itself (at other nodes with more than two neighbours), and so on.
  - This is called **broadcast storm**, resulting from the endless multiplication of broadcast packets, would eventually result in so many broadcast packets being created that the network would be rendered useless.

Controlled Flooding

- To avoiding a broadcast storm problem uses controlled flooding approach.
- A node to judiciously choose **when to flood a packet** and (e.g., ***if it has already received and flooded an earlier copy of a packet***) **when not to flood a packet**.
- Two approaches:
- **sequence-number-controlled flooding**
- **reverse path forwarding (RPF) or reverse path broadcast (RPB)**

Controlled Flooding

- In **sequence-number-controlled flooding**, a source node puts its address (or other unique identifier) as well as a **broadcast sequence number** into a broadcast packet, then sends the packet to all of its neighbours.
- Each node maintains a list of the source address and sequence number of each broadcast packet it has already received, duplicated, and forwarded.

Controlled Flooding

- When a node receives a broadcast packet, it first checks whether the packet is in this list. If so, the packet is dropped; if not, the packet is duplicated and forwarded to all the node's neighbours (except the node from which the packet has just been received).

Controlled Flooding

- A second approach to controlled flooding is known as **reverse path forwarding (RPF)** [Dalal 1978], also sometimes referred to as reverse path broadcast (RPB).
- When a router receives a broadcast packet with a given source address, it transmits the packet on all of its outgoing links (except the one on which it was received) only if the packet arrived on the link that is on its own shortest unicast path back to the source.

Controlled Flooding

- Otherwise, the router simply discards the incoming packet without forwarding it on any of its outgoing links. Such a packet can be dropped because the router knows it either will receive or has already received a copy of this packet on the link that is on its own shortest path back to the sender.

Controlled Flooding

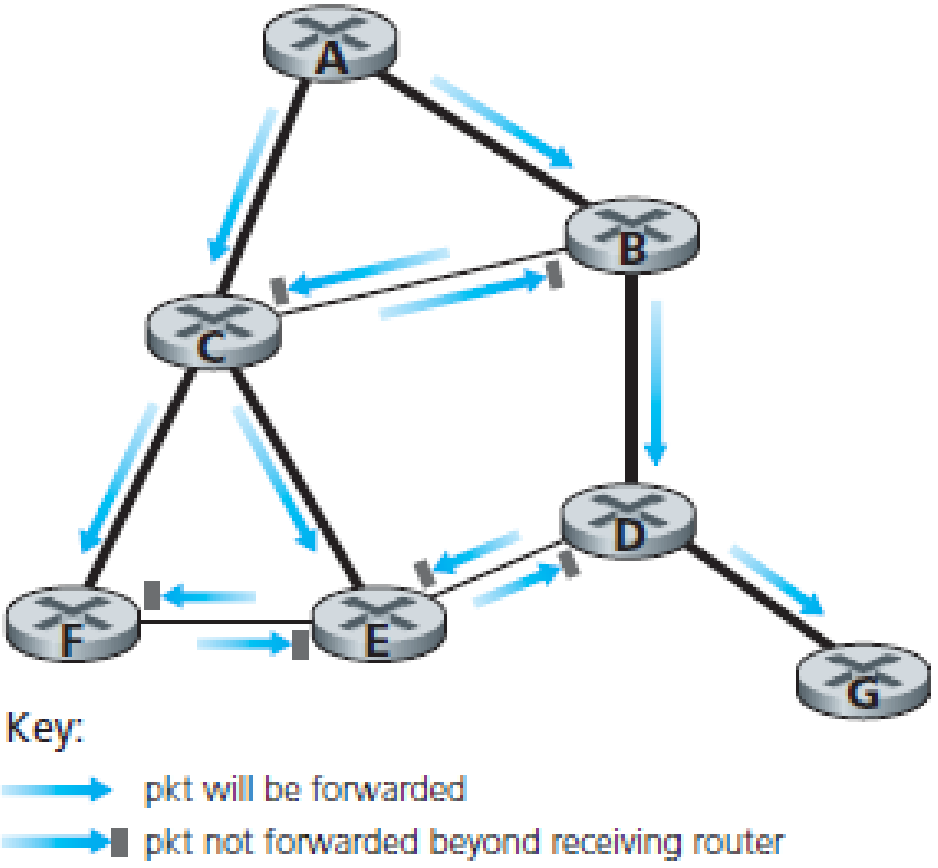


Figure 4.44 ♦ Reverse path forwarding

The Distance-Vector (DV) Routing Algorithm

- The **distance vector** (**DV**) algorithm is iterative, asynchronous, and distributed.
  - It is *distributed* in that each node receives some information from one or more of its *directly attached* neighbours, performs a calculation, and then distributes the results of its calculation back to its neighbours.
  - It is *iterative* in that this process continues on until no more information is exchanged between neighbours.
  - The algorithm is *asynchronous* in that it does not require all of the nodes to operate in lockstep with each other.

The Distance-Vector (DV) Routing Algorithm

- Let *dx*(*y*) be the cost of the least-cost path from node *x* to node *y*. Then the least costs are related by the celebrated Bellman-Ford equation, namely,

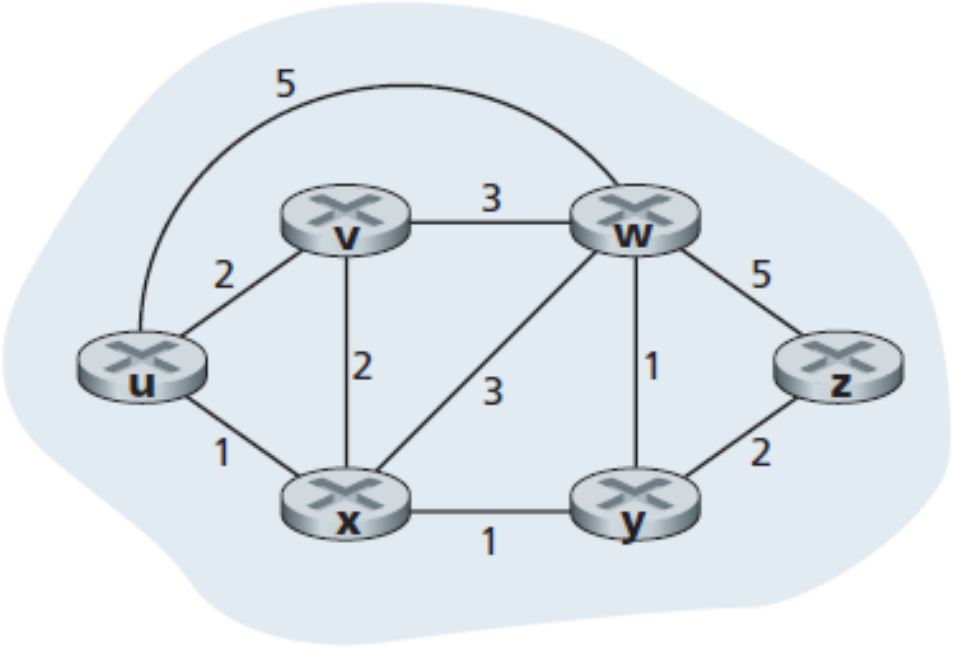
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\},$$

- where the min_v in the equation is taken over all of *x*'s neighbours. After traveling from *x* to *v*, if we then take the least-cost path from *v* to *y*, the path cost will be $c(x,v) + d_v(y)$.

The Distance-Vector (DV) Routing Algorithm

The basic idea is as follows. Each node x begins with $D_x(y)$, an estimate of the cost of the least-cost path from itself to node y, for all nodes in N. Let $D_x = [D_x(y): y \text{ in } N]$ be node x's distance vector, which is the vector of cost estimates from x to all other nodes, y, in N. With the DV algorithm, each node x maintains the following routing information:

- For each neighbor v, the cost $c(x,v)$ from x to directly attached neighbor, v
- Node x's distance vector, that is, $D_x = [D_x(y): y \text{ in } N]$, containing x's estimate of its cost to all destinations, y, in N
- The distance vectors of each of its neighbors, that is, $D_v = [D_v(y): y \text{ in } N]$ for each neighbor v of x

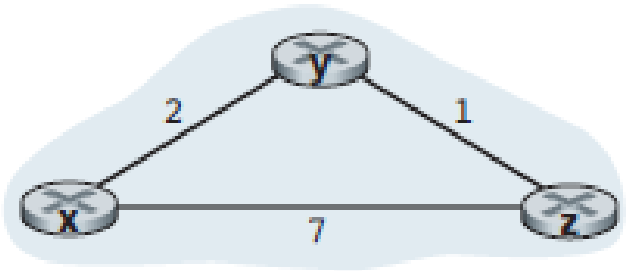


- The source node *u* has three neighbours: nodes *v*, *x,* and *w*.
- $d_u(z) = \min\{c(u,v)+d_v(z), c(u,x)+d_x(z), c(u,w)+d_w(z)\}$
- $d_u(z) = \min\{2 + 5, 1 + 3, 5 + 3\} = 4$,

Distance-Vector (DV) Algorithm

At each node, x:

```
1  Initialization:
2     for all destinations y in N:
3        Dx(y) = c(x,y)   /* if y is not a neighbor then c(x,y) = ∞ */
4     for each neighbor w
5        Dw(y) = ? for all destinations y in N
6     for each neighbor w
7        send distance vector Dx = [Dx(y): y in N] to w
8
9  loop
10    wait (until I see a link cost change to some neighbor w or
11          until I receive a distance vector from some neighbor w)
12
13    for each y in N:
14       Dx(y) = minv{c(x,v) + Dv(y)}
15
16    if Dx(y) changed for any destination y
17       send distance vector Dx = [Dx(y): y in N] to all neighbors
18
19 forever
```



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

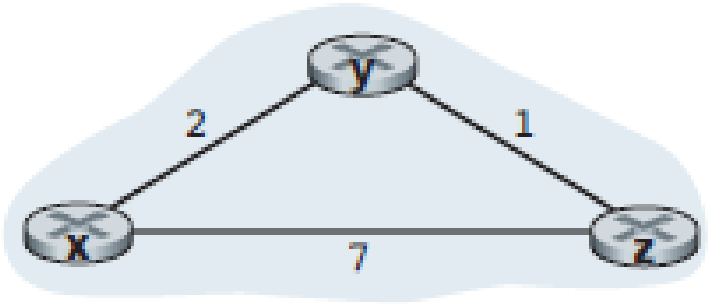
from	cost to x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

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

The Distance-Vector (DV) Routing Algorithm



$$D_x(x) = 0$$

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

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

END