Routing in Mobile Ad Hoc Networks

HACETTEPE ÜNİVERSİTESİ

Daha İleriye... En İyiye...

Two key network-layer functions

forwarding: move packets from router's input to appropriate router output

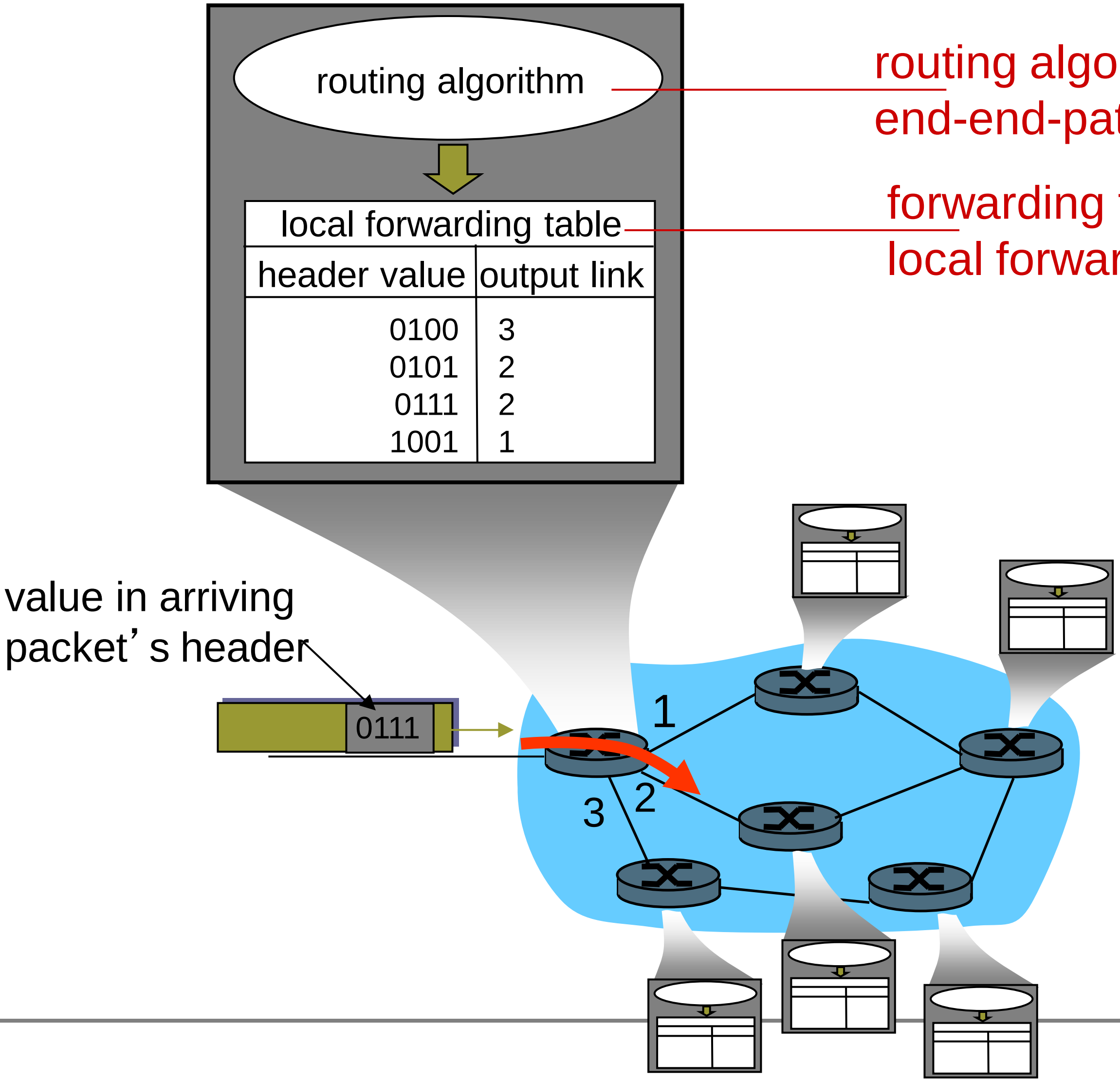
routing: determine route taken by packets from source to dest.

- *routing algorithms*

analogy:

- *routing:* process of planning trip from source to dest
- *forwarding:* process of getting through single interchange

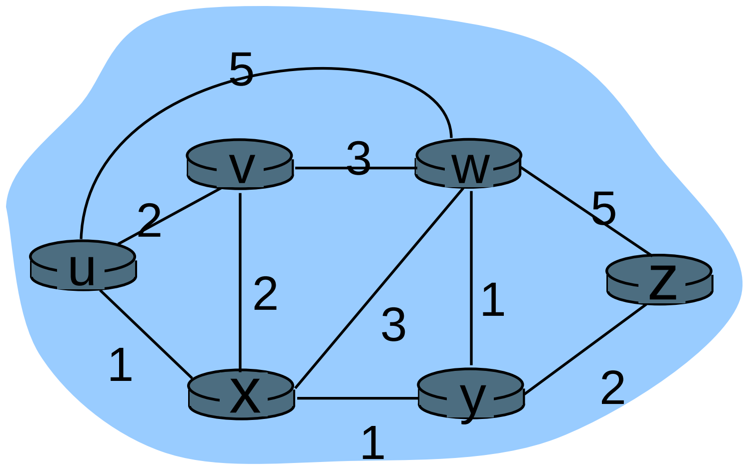
Interplay between routing and forwarding



routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router

Graph abstraction



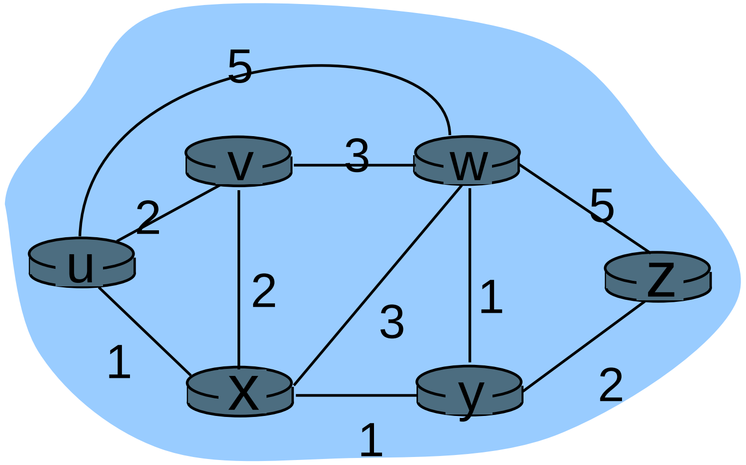
graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

aside: graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

Graph abstraction: costs



$c(x,x')$ = cost of link (x,x')
e.g., $c(w,z) = 5$

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

cost of path $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

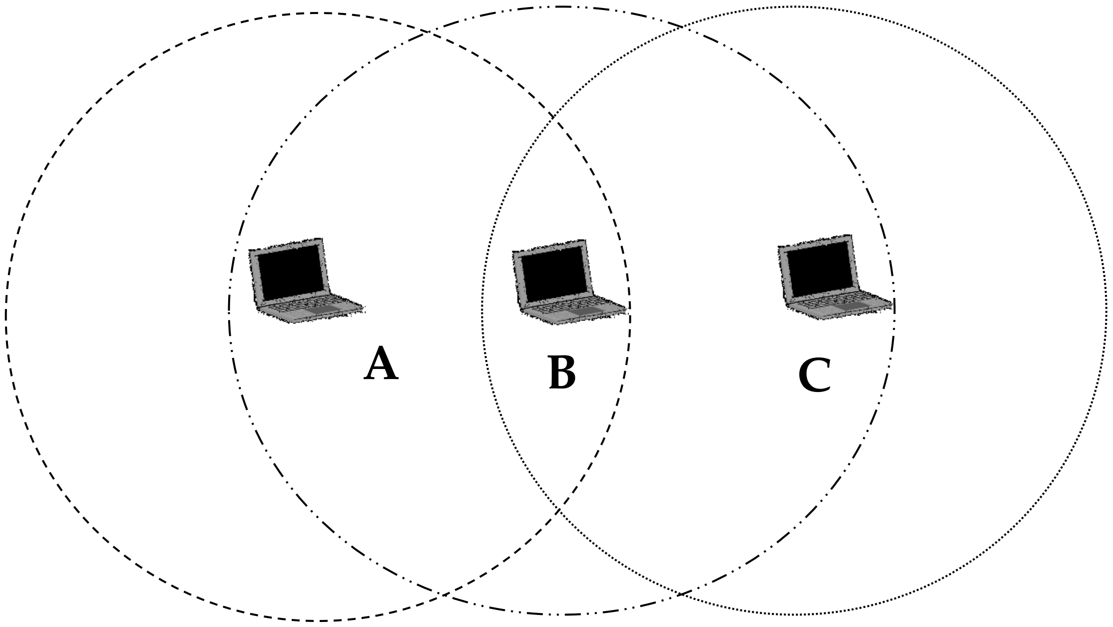
key question: what is the least-cost path between u and z ?
routing algorithm: algorithm that finds that least cost path

Routing

Two fundamental steps:

- Forwarding packets
- Determining where to forward packets
  - Reach the destination
  - Minimize path length (number of hops)
  - Minimize delay
  - Minimize packet loss
  - Minimize cost

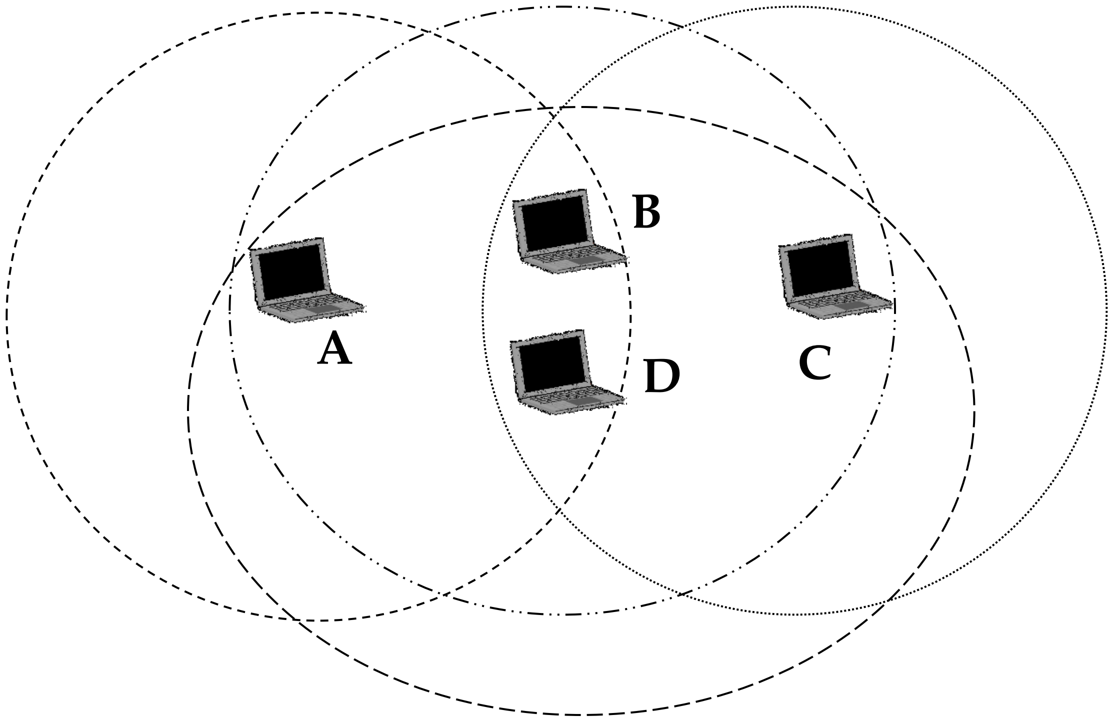
Routing Example 1



No direct communication between A and C.

A-B-C

Routing Example 2



A-B-C
A-B-D-C
A-D-B-C
A-D-C

Gets more complicated with the increase of the number of nodes in the network.

Routing Decision Point

Source routing

- Source node determines the route and specifies it in the packet.

Hop-by-hop routing

- Decision is made at each forwarding node.

Routing Table

It contains information to determine how to forward packets.

- Source routing : used to determine the route to the destination.
- Hop-by-hop routing : used to determine the route to the next hop.

- Distance vector algorithms
- Link state algorithms

Routing algorithm classification

Q: global or decentralized information?

global:

- all routers have complete topology, link cost info
- "link state" algorithms

decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

A Link-State Routing Algorithm

Dijkstra's algorithm

- net 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
  - gives *forwarding table* for that node
- iterative: after k iterations, know least cost path to k dest.'s

notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra’s Algorithm

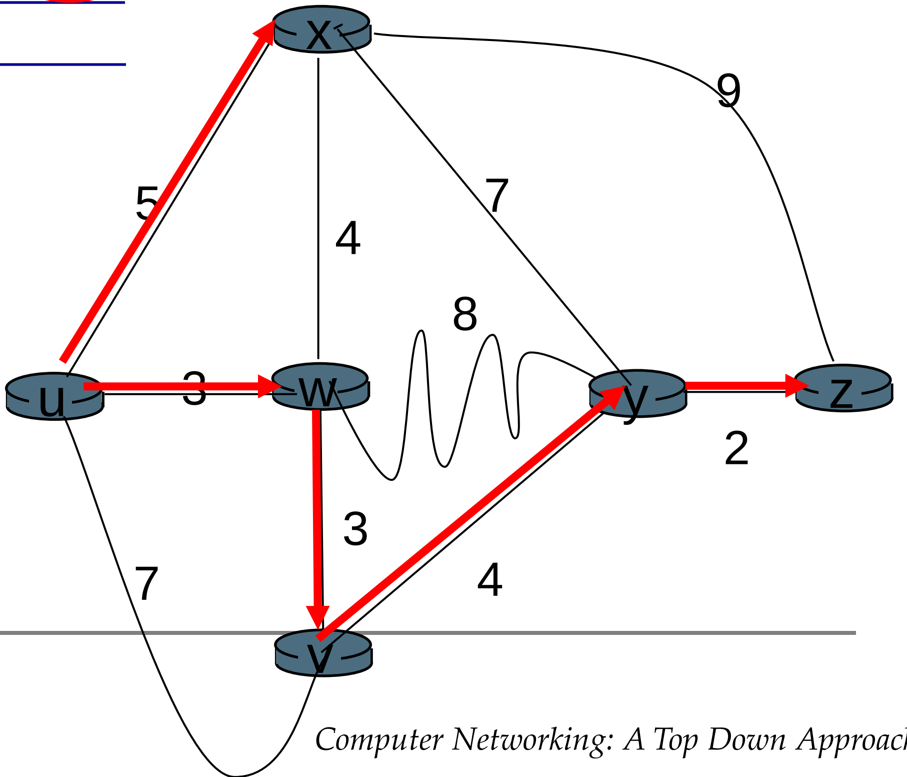
```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15  until all nodes in N'
```

Dijkstra's algorithm: example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

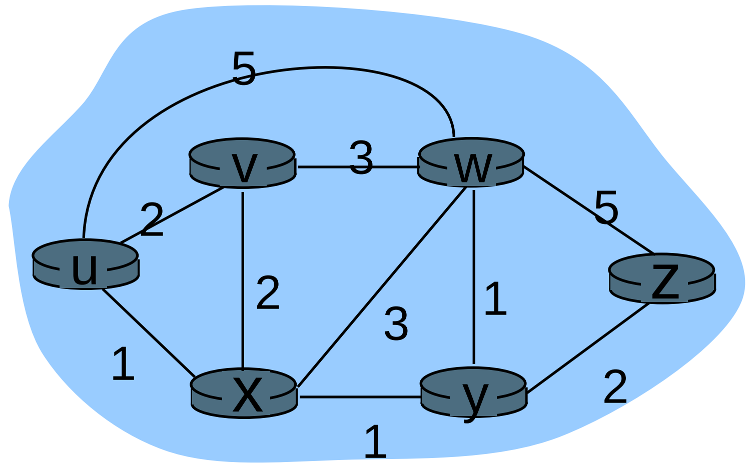
notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)



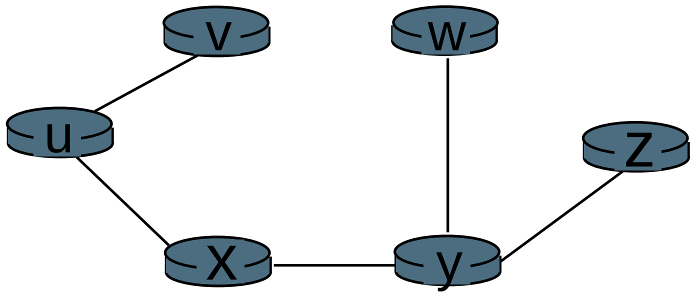
Dijkstra's algorithm: another example

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					



Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:



resulting forwarding table in u:

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

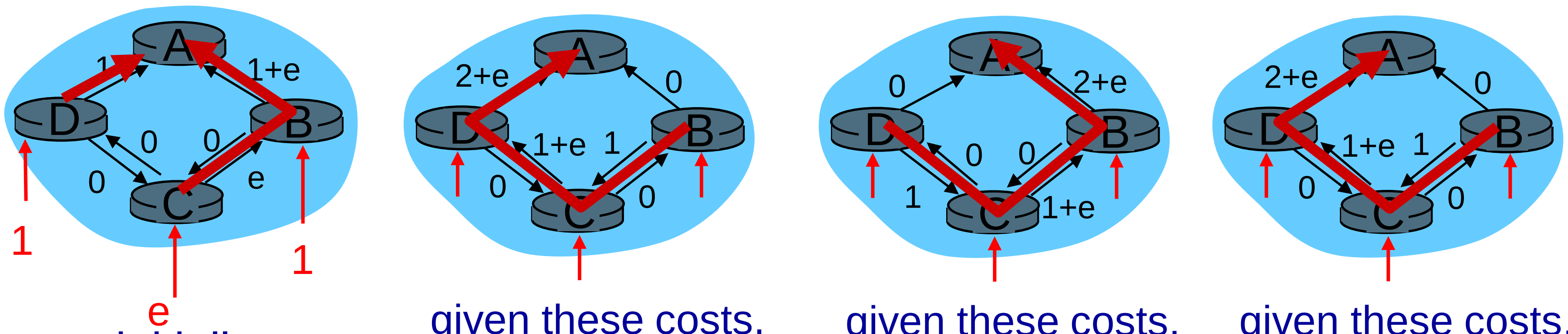
Dijkstra's algorithm, discussion

algorithm complexity: n nodes

- each iteration: need to check all nodes, w, not in N
- n(n+1)/2 comparisons: O(n^2)
- more efficient implementations possible: O(nlogn)

oscillations possible:

- e.g., support link cost equals amount of carried traffic:



initially

given these costs, find new routing…. resulting in new costs

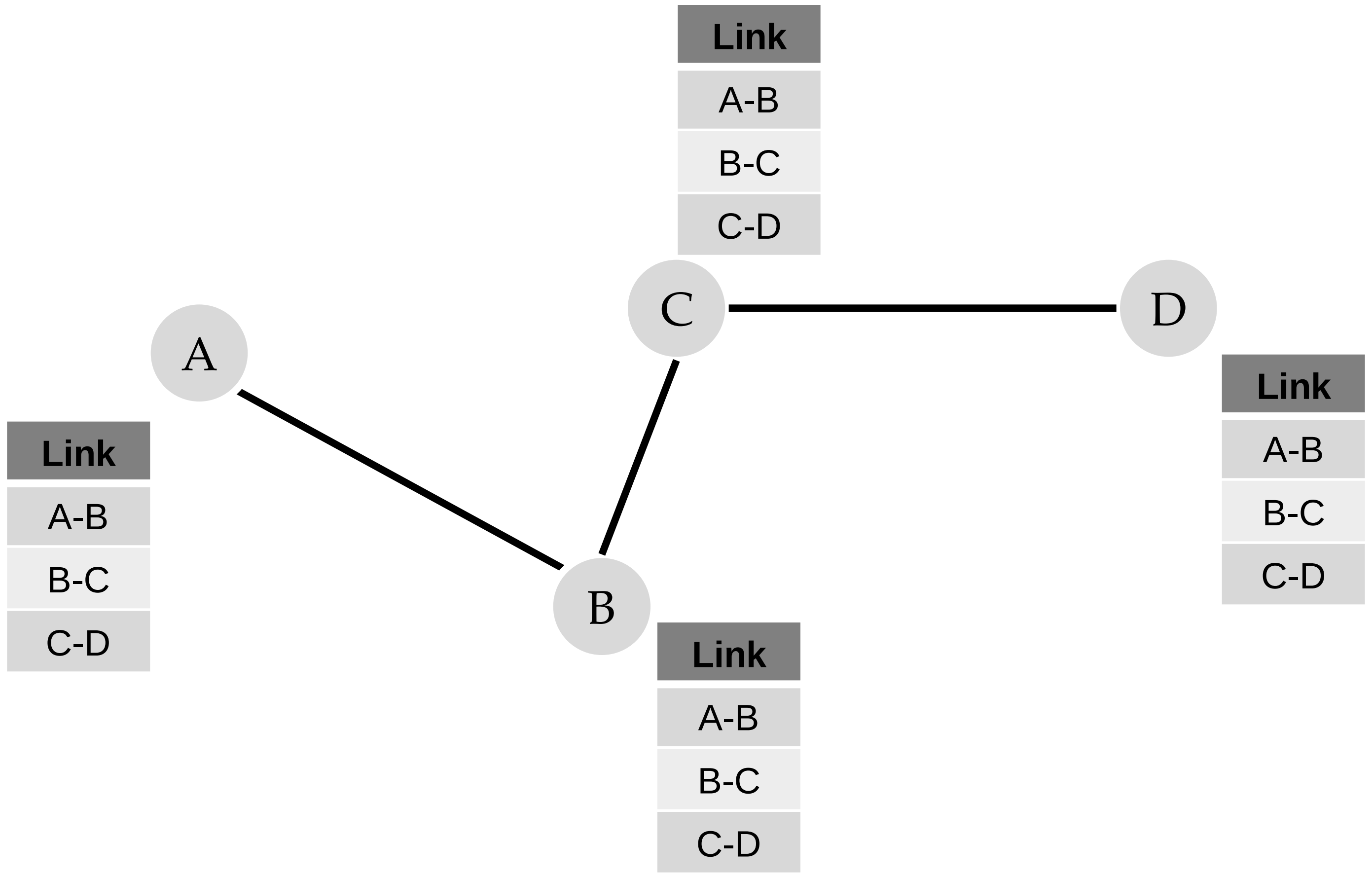
given these costs, find new routing…. resulting in new costs

given these costs, find new routing…. resulting in new costs

Link State Algorithms

- Each node shares its link information.
- All nodes can build a map of the full network topology.
- Link information is updated when a link changes its state (up or down).
- A node can determine the next best hop or a route with given full topology information.

Link State Algorithms (1)



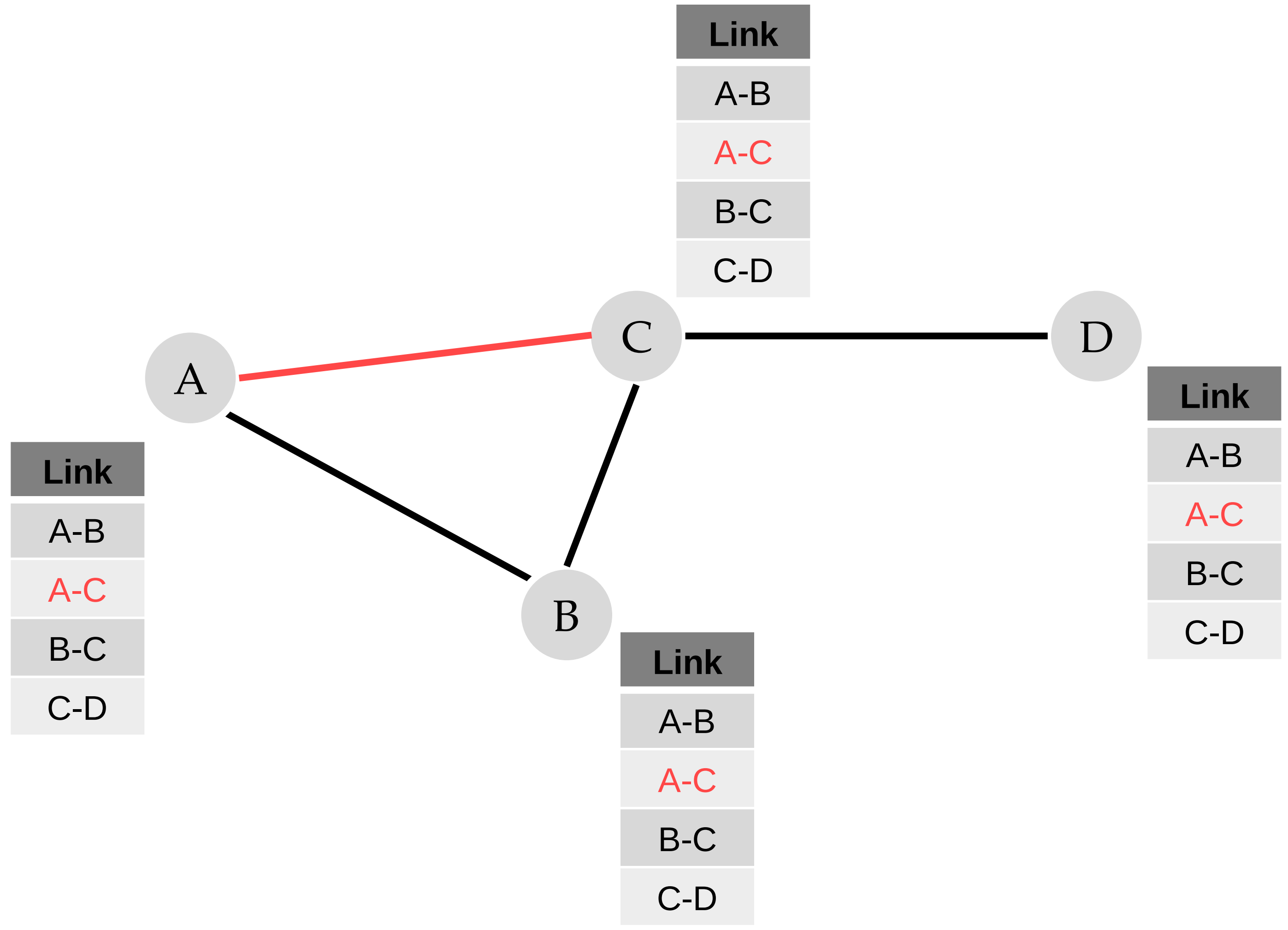
Link
A-B
B-C
C-D

Link
A-B
B-C
C-D

Link
A-B
B-C
C-D

Link
A-B
B-C
C-D

Link State Algorithms (2)



Link
A-B
A-C
B-C
C-D

Link
A-B
A-C
B-C
C-D

Link
A-B
A-C
B-C
C-D

Link
A-B
A-C
B-C
C-D

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

$d_x(y)$:= cost of least-cost path from x to y

then

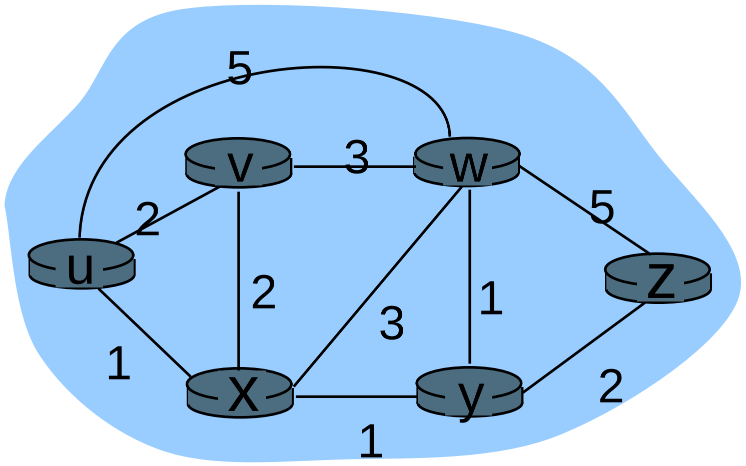
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford example



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

B-F equation says:

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

node achieving minimum is next
hop in shortest path, used in forwarding table

Distance vector algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- node x:
  - knows cost to each neighbor v: $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains
    $\mathbf{D}_v = [D_v(y): y \in N]$

Distance vector algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

  $D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$ *for each node* $y \in N$

- under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance vector algorithm

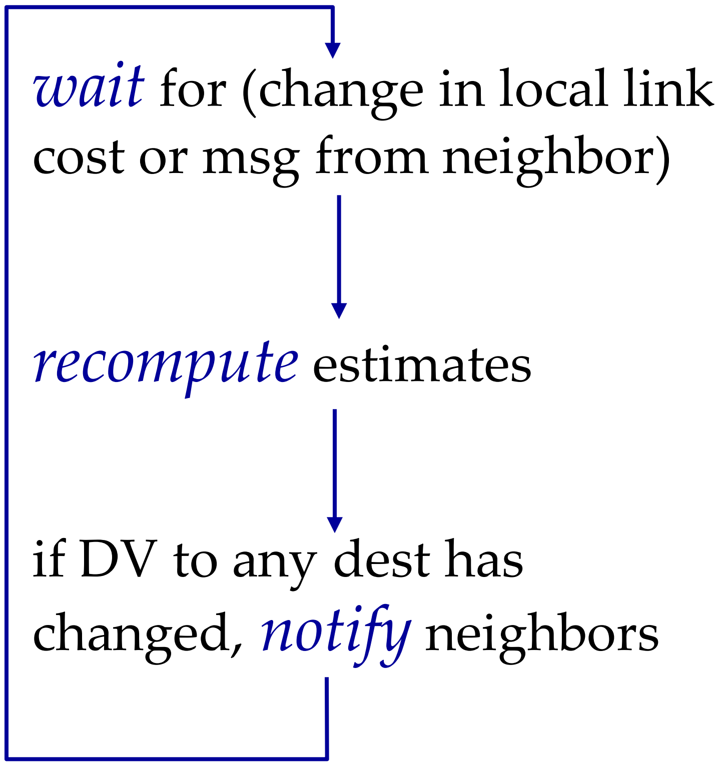
iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

each node:



wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

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

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

node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

cost to

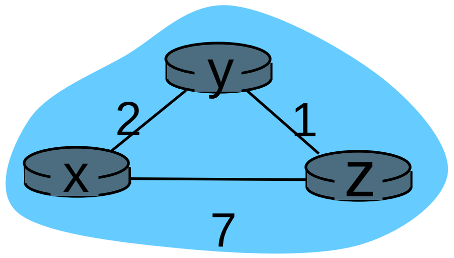
from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

cost to

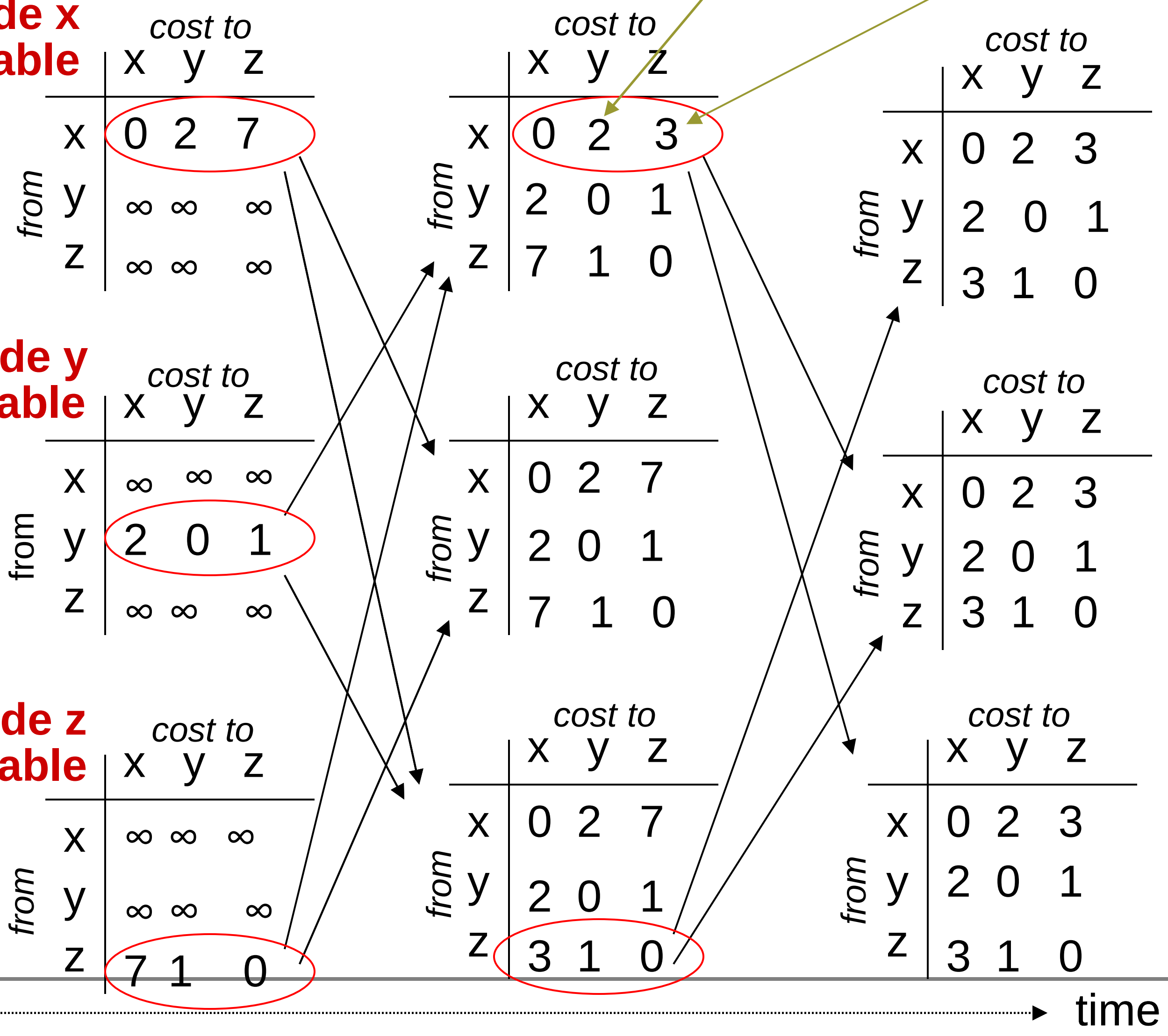
from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time



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

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



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

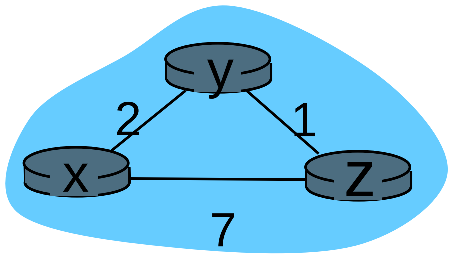
node z table

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

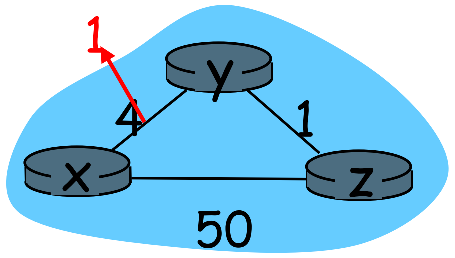
time



Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



"good news travels fast"

t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

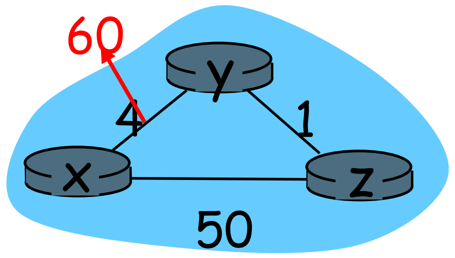
t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

t_2 : *y* receives *z*' s update, updates its distance table. *y*' s least costs do *not* change, so *y* does *not* send a message to *z*.

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see text



poisoned reverse:

- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- will this completely solve count to infinity problem?

Comparison of LS and DV algorithms

message complexity

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - convergence time varies

speed of convergence

- ***LS:*** O(n^2) algorithm requires O(nE) msgs
  - may have oscillations
- ***DV:*** convergence time varies
  - may be routing loops
  - count-to-infinity problem

robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

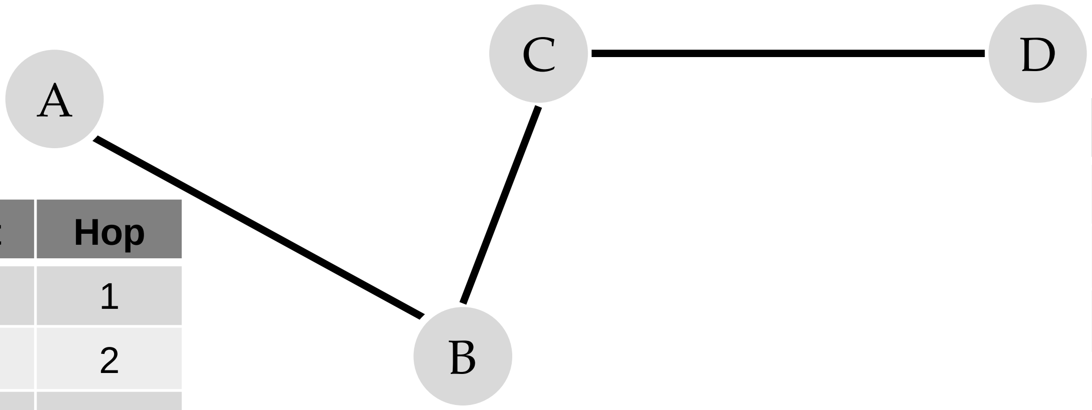
- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

Distance Vector Algorithms

- The metric ‘distance’ to be minimized.
- Routers do not have knowledge of the entire path to a destination.
- Direction in which or interface to which a packet should be forwarded.
- Distance from its destination.

Distance Vector Algorithms (1)

Dest	Next	Hop
A	B	2
B	B	1
D	D	1



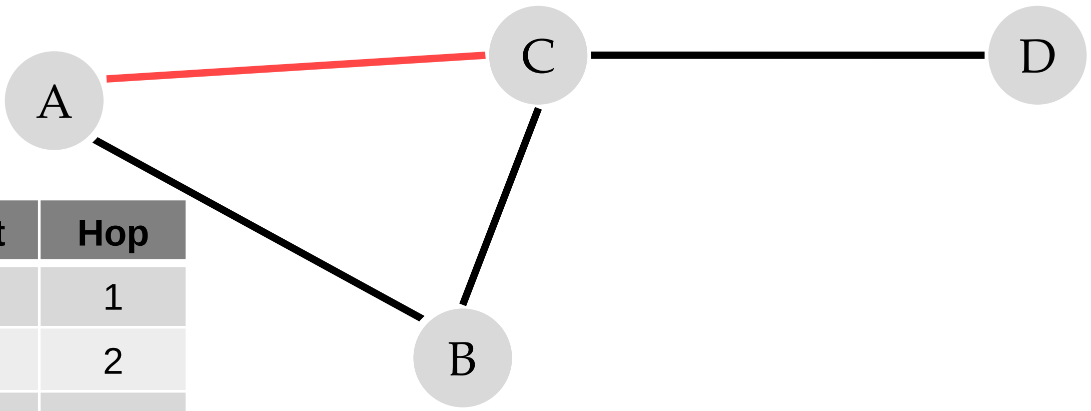
Dest	Next	Hop
A	C	3
B	C	2
C	C	1

Dest	Next	Hop
B	B	1
C	B	2
D	B	3

Dest	Next	Hop
A	A	1
C	C	1
D	C	2

Distance Vector Algorithms (2)

Dest	Next	Hop
A	B	2
B	B	1
D	D	1



Dest	Next	Hop
A	C	3
B	C	2
C	C	1

Dest	Next	Hop
B	B	1
C	B	2
D	C	2

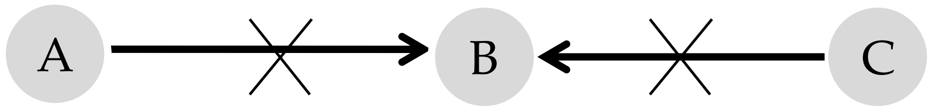
Dest	Next	Hop
A	A	1
C	C	1
D	C	2

MANET vs. Traditional Routing (1)

1. Every node behaves as a router (usually with a single interface) and a host.
2. It has a dynamic topology.
3. Routing consider both Layer 3 and Layer 2 (connectivity, interference) information.
4. It might have many more redundant links than traditional routing.
5. Channel properties (capacity, error rates) may vary.

MANET vs. Traditional Routing (2)

6. Interference is an issue in MANETs.

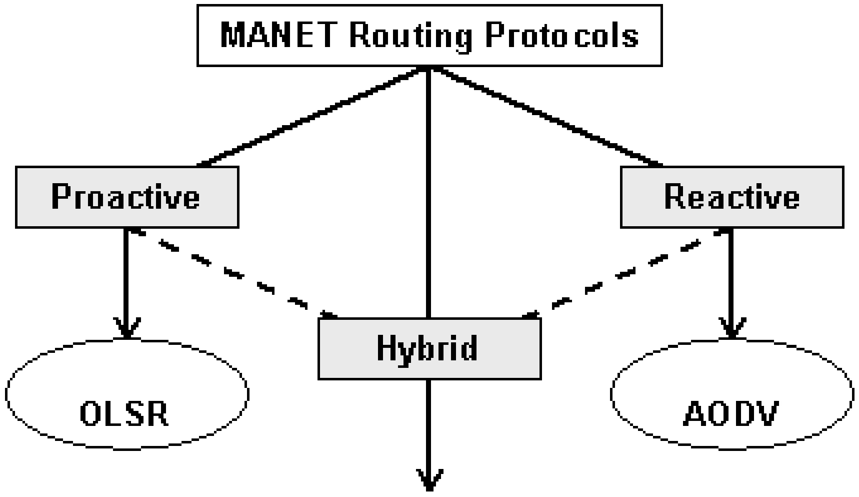


7. Channels can be asymetric (IEEE 802.11 MAC assumes symetric channels).
8. Power is an issue.
9. There is limited physical security.

Expected Properties of MANET Routing Protocols

- Be distributed.
- Assume routes as unidirectional.
- Be power-efficient.
- Consider its security.
- Combine benefits of different routing protocols.
- Be aware of QoS.
  - Reduce overhead route discovery and maintenance, *etc.*

Routing Protocols for MANETs



Other classifications are also possible :

- Secure routing protocols.

Reactive vs. Proactive

Reactive routing protocols

Reduces control traffic overhead at the cost of increased latency in finding the route to a destination.

Proactive routing protocols

Immediately provide the required routes when needed at the cost of bandwidth used in sending frequent periodic updates & topology.

Network Structure

- Flat
  - All nodes participate in routing
- Hierarchical
  - may use a proactive protocol for routing within a cluster or zone
  - may use a reactive protocol for routing between cluster/zone heads

Proactive (table-driven) Routing Protocols

- Continuously exchange topological information among the network nodes.
- When a route is needed, such route information is available immediately.
- The cost of maintaining the network is high under high mobility.
- Not power-efficient for the networks with low activity.

❖ DSDV, OLSR, *etc.*

Reactive (on-demand) Routing Protocols

- Routes are established when they are needed.
- No periodic transmission of topological information of the network.
- Resource-conserving protocols.

- AODV, DSR, and the like.

Hybrid Routing Protocols

- Include some characteristics of proactive routing protocols and some characteristics of reactive routing protocols.

- ZRP, *etc.*

IETF MANET Working Group

http://datatracker.ietf.org/wg/manet/charter/

The purpose of this working group is to standardize IP routing protocol functionality suitable for wireless routing application within both static and dynamic topologies. The fundamental design issues are that the wireless link interfaces have some unique routing interface characteristics and that node topologies within a wireless routing region may experience increased dynamics, due to motion or other factors.◎

IETF MANET Working Group

- Ad Hoc On Demand Distance Vector (AODV) protocol
- Dynamic Source Routing (DSR) protocol
- Optimized Link State Routing (OLSR) protocol
- Topology Broadcast based on Reverse-Path Forwarding (TBRPF) protocol

AODV
Ad Hoc On-Demand Distance Vector Routing

AODV – Ad Hoc On-Demand Distance Vector Routing

- On-demand routing protocol.
- Determines unicast routes to destinations.

- Quick adaptation to dynamic link conditions.
- Low processing and memory overhead.
- Low network utilization.
- Loop-free.
- Respond to link breakages and changes in a timely manner.

Message Types

Route Discovery

RREQ (Route Request) : When a route to a destination is needed.

RREP (Route Reply) : unicasting back to the source node.

Route Maintenance

RERR (Route Error) : to notify other nodes of the link break.

- Received via UDP (654).

Some definitions

<u>active (valid) route :</u>

- A route towards a destination that has a routing table entry marked as valid.

<u>invalid route :</u>

- A route that has expired, stated as invalid in the routing table entry.
- Cannot be used to forward data packets.
- Stored for an extended periof of time.
- Can be useful for route repairs, and future RREQ messages.

AODV

- Tens to thousands of mobile nodes.
- Handle low, moderate, relatively high mobility rates.
- Handle variety of data traffic levels.
- Assume that all nodes are cooperative, non-malicious.

- Designed to reduce dissemination of control traffic, eliminate overhead on data traffic.

Sequence Numbers (1)

- Monotonically increased number maintained by each node.
- Used to determine the freshness of the information contained from the source node.
- Using destination sequence numbers ensures loop-freedom.
- A route with the greatest sequence number is always chose.

Sequence Numbers (2)

A destination node increments its own sequence numbers in two circumstances :

- Before a node originates a route discovery.

(prevents coflicts with previously established route towards this node).

- Before a destination node originates a RREP in response to a RREQ.

(to the maximum of its current and the seq. number in RREQ).

Sequence Numbers (3)

A node change the sequence number in routing table entry in the following circumstances :

- It is itself the destination node.
- It receives an AODV message with new information about the seq. number for a destination node.
- The path towards the destination node expires or breaks.

Route Request (1)

- Initiated when a node wants to communicate with another node, but does not have a route to that node.
- The node broadcast a route request (RREQ) packet.
- Every neighbour receives the RREQ :
  - Returns a route reply (RREP) packet, or
  - Forwards the RREQ to its neighbors.
- (source_addr, broadcast_id) uniquely identifies the RREQ
  - broadcast_id is incremented for every RREQ packet sent.
  - Receivers can identify and discard duplicate RREQ packets.

Route Request (2)

If a node cannot respond to the RREQ

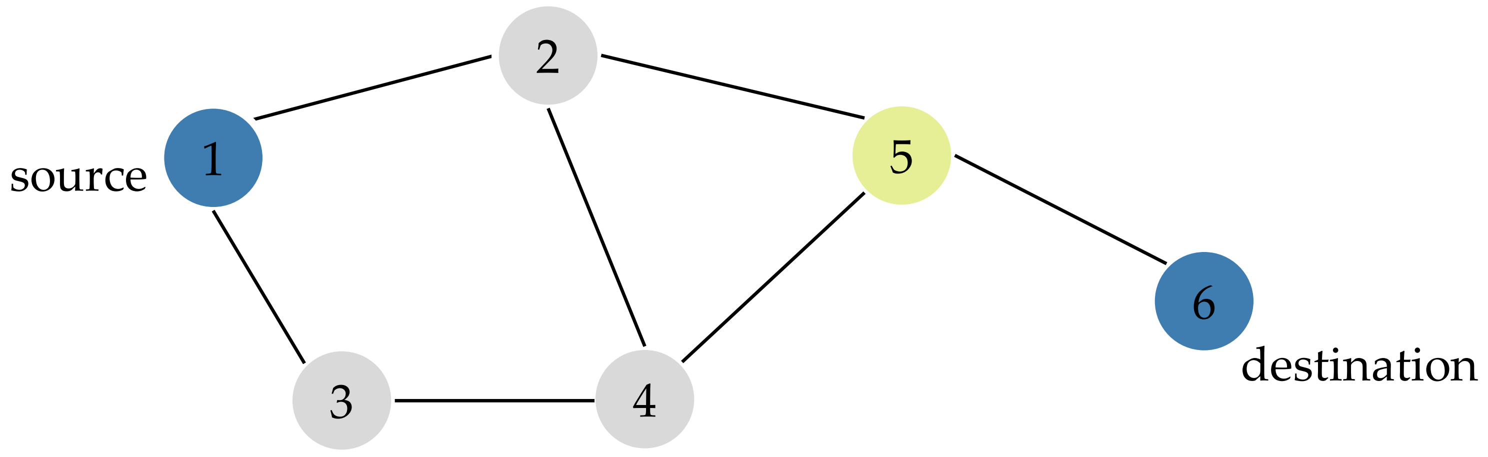
- The node increments the hop count
- The node saves information to implement a reverse path set up
  - Neighbour that sent this RREQ packet
  - Destination IP address
  - Source IP address
  - Broadcast ID
  - Source node's sequence number
  - Expiration time for reverse path entry (to enable garbage collection)

Route Request Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |J|R|G|D|U|   Reserved          |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            RREQ ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Destination Sequence Number                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Originator IP Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Originator Sequence Number                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

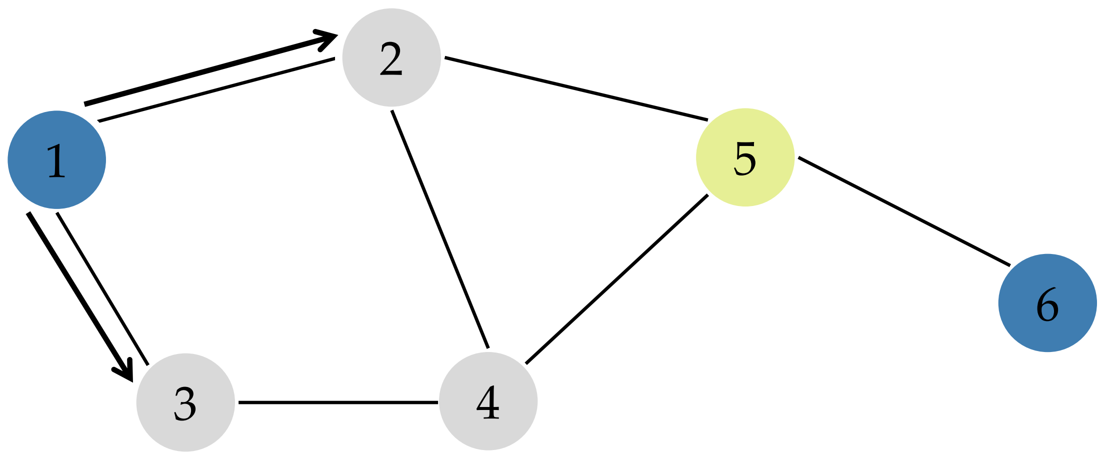
G : gratuitous
D : destination only flag
U : unknown sequence number

Route Discovery Example (1)



- Node 1 wants to send a data packet to Node 6, but has no route to Node 6 in its routing table.
- Assume Node 5 knows a route to Node 6, and no other node in the network has routing information related to Node 6.

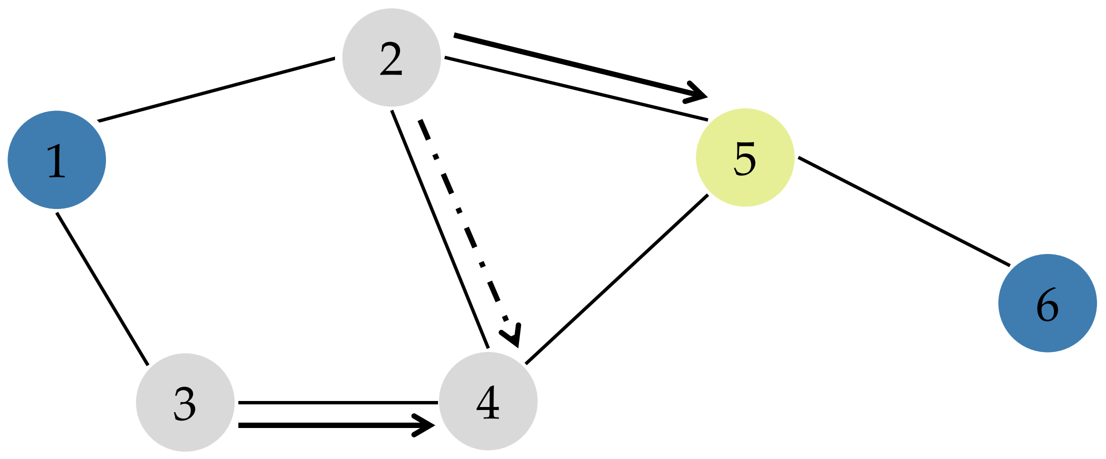
Route Discovery Example (2)



Node 1 broadcast a RREQ packet

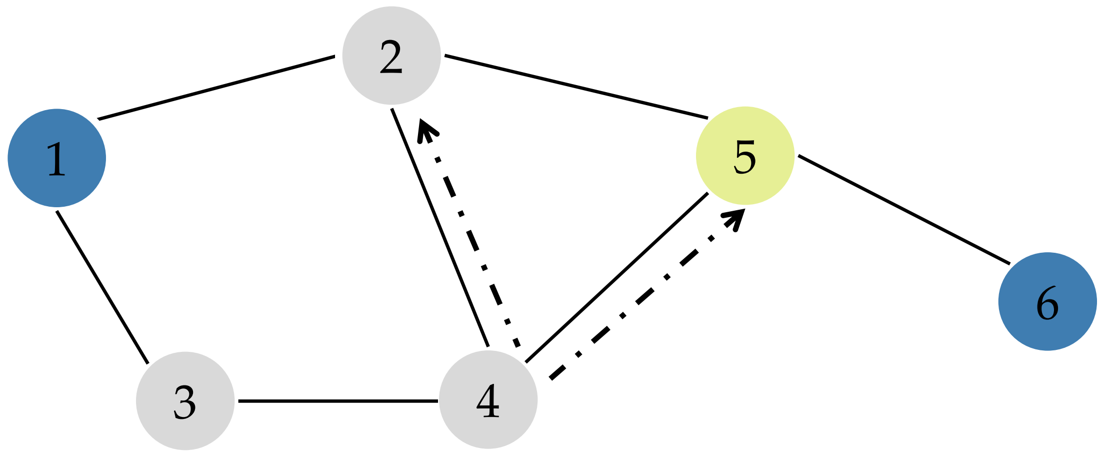
- source_addr = 1
- dest_addr = 6
- broadcast_id = broadcast_id + 1
- source_sequence_# = source_sequence_# + 1
- dest_sequence_# = last dest_sequence_# for Node 6

Route Discovery Example (3)



- Nodes 2 and 3 forward the RREQ
- Update source_sequence_# for Node 1
- Increment hop_count in the RREQ packet

Route Discovery Example (4)



- Nodes 4 will forward the RREQ packet, but the receivers recognize the packets as duplicates.
- RREQ reaches Node 5 which knows a route to 6.
  - Node 5 must verify that the destination sequence number is less than or equal to the destination sequence number it has recorded for Node 6.

Route Reply

- The destination node or a node which receives an RREQ packet and it has a route the destination, initiates a RREP packet to the neighbour node that sent the RREQ packet.

- The intermediate nodes propogate the RREP packet.

- Other RREP packets are discarded unless◎
  - dest_sequence_# number is higher than the previous, or
  - destination_sequence_# is the same, but hop_count is smaller.
- Cached reverse routes will be dropped.

RREP Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |R|A|    Reserved     |Prefix Sz|   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination IP address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Destination Sequence Number                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Originator IP address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Lifetime                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A: acknowledgement required

RREP-ACK:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Some Parameters

RREQ_RATELIMIT : RREQ messages per second.

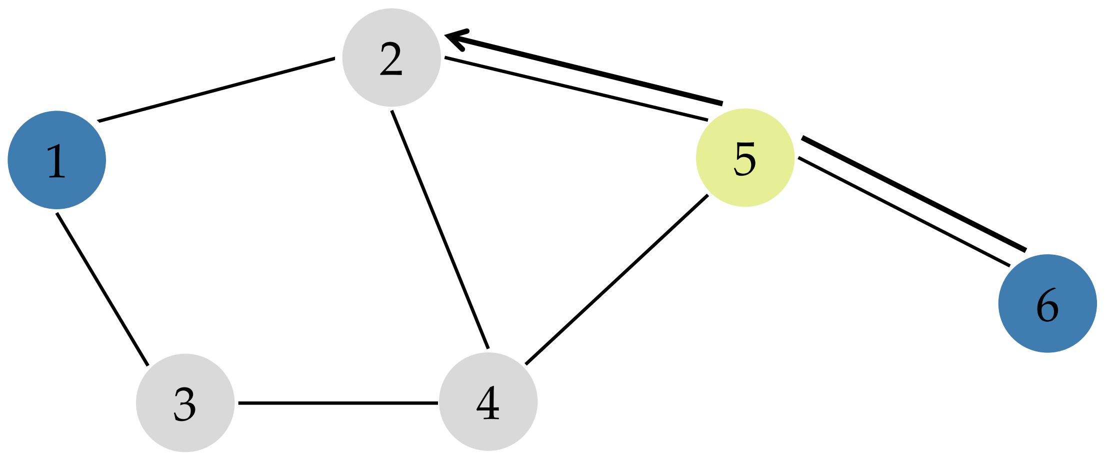
NET_TRAVERSAL_TIME :

RREQ_RETRIES

<u>Exponential backoff</u>

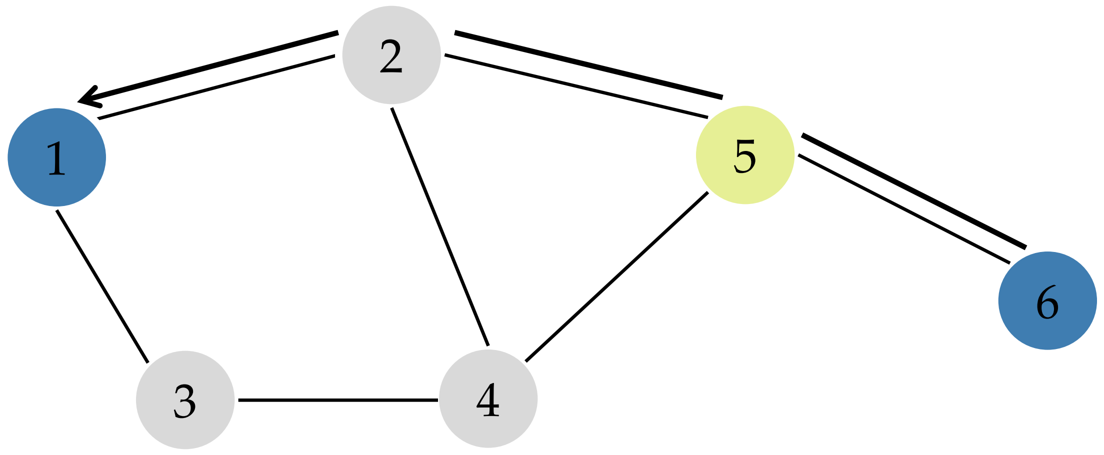
- To reduce congestion
- The first time waits NET_TRAVERSAL_TIME milliseconds for the reception of a RREP.
- The second time waits 2*NET_TRAVERSAL_TIME.

Route Discovery Example (5)



- Node 5 knows a route to Node 6 and sends an RREP to Node 2.
  - source_addr = 1
  - dest_addr = 6
  - dest_sequence_# = max(own sequence number,
    dest_sequence_# in RREQ)
  - hop_cnt = 1

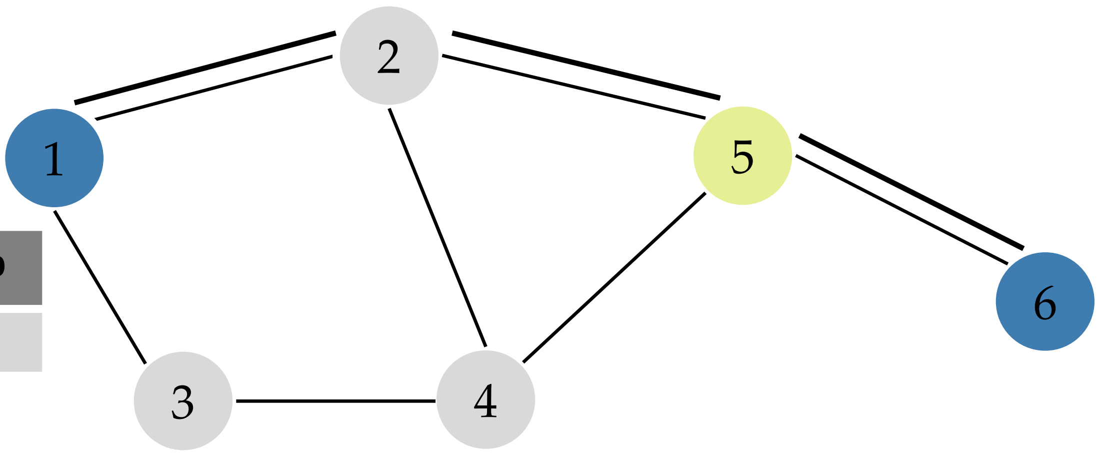
Route Discovery Example (6)



- Node 2 forwards the RREP packet to Node 1.
  - this is a new route reply (here) or
  - one that has a lower hop count and, or
  - One that has a higher sequence number.
- Increments hop_count (=2) in the RREP packet.

Route Discovery Example (7)

Dest	Next	Hop
6	2	3



- Node 1 now has a route (the shorthest) to Node 6.
- Node 1 now can use this route to send data packets.

Local Connectivity

Each forwarding node should keep track of its continued connectivity to its acive next hops (neighbours).

- Link layer notification.
  - Absence of a link layer ACK.
  - Failure to get CTS after sending RTS.
- Hello messages (allowed_hello_loss * hello_interval)
- Passive acknowledgement.
  - Failure of forwarding a packet by the next node (by listening the channel. next_hop_wait milliseconds).

Route Maintenance

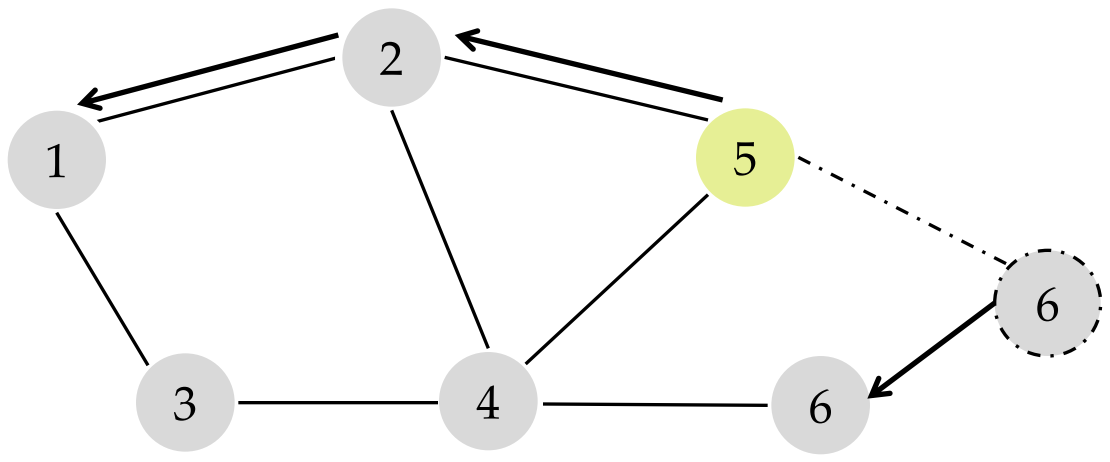
The upstream (toward the source) node detecting a failure propagates an route error (RERR) packet with a new destination sequence number.

The source (or another node on the path) can rebuild a path by sending a RREQ packet.

RERR Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |N|          Reserved           |   DestCount   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Unreachable Destination IP Address (1)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Unreachable Destination Sequence Number (1)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|  Additional Unreachable Destination IP Addresses (if needed)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Additional Unreachable Destination Sequence Numbers (if needed)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

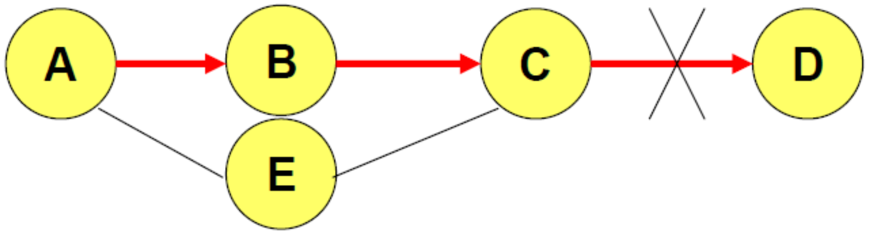
Route Maintenance Example



- Assume that Link 5-6 breaks.
- Node 5 issues an RERR packet indicating the broken path.
- The RERR propagates back to Node 1.
- Node 1 can discover a new route.

Why Sequence Numbers in AODV

- To avoid using old/broken routes
  - To determine which route is newer
- To prevent formation of loops
  - A had a route to D initially
  - Assume that A does not know about failure of link C-D because RERR sent by C is lost



  - Now C performs a route discovery for D. Node A receives the RREQ (say, via path C-E-A)
  - Node A will reply since A knows a route to D via node B
  - Results in a loop (for instance, C-E-A-B-C)

OLSR
Optimized Link State Routing Protocol

OLSR

- A proactive (table-driven) protocol.
- Not require reliable transmission of control messages.
- Not require sequenced delivery of messages.
- An optimization of the classical link state alg.
- The key concept : multipoint relays (MPRs).
- Provides optimal routes (number of hops).
- Suitable for large and dense networks.

Definitions

2-hop neighbour:

a node heard by neighbour.

strict 2-hop neighbour:

a 2-hop neighbour which is not the node itself or a neighbour of the node.

multipoint relay (mpr)

a node selected by its one-hop neighbour X to re-transmit all the broadcast messages received from X (not duplicate).

multipoint relay selector (mpr selector, ms)

a node which has selected its one-hop neighbour M as its multipoint relay.

OLSR

- Optimization of a pure link state protocol
- It reduces the size of control packets
  - Instead of all links, it declares MSs
- It minimizes flooding of the control traffic by using only selected nodes, MPRs.
- Apart from normal periodic control messages, it does not generate extra traffic in response to link failures, and additions.

Multipoint Relays (MPR) 1

- Selected nodes which forward broadcast messages during the flooding process.
- Reduces the message overhead.
- Link state information is generated only by these elected nodes.
- Minimize the number of control messages flooding in the network.

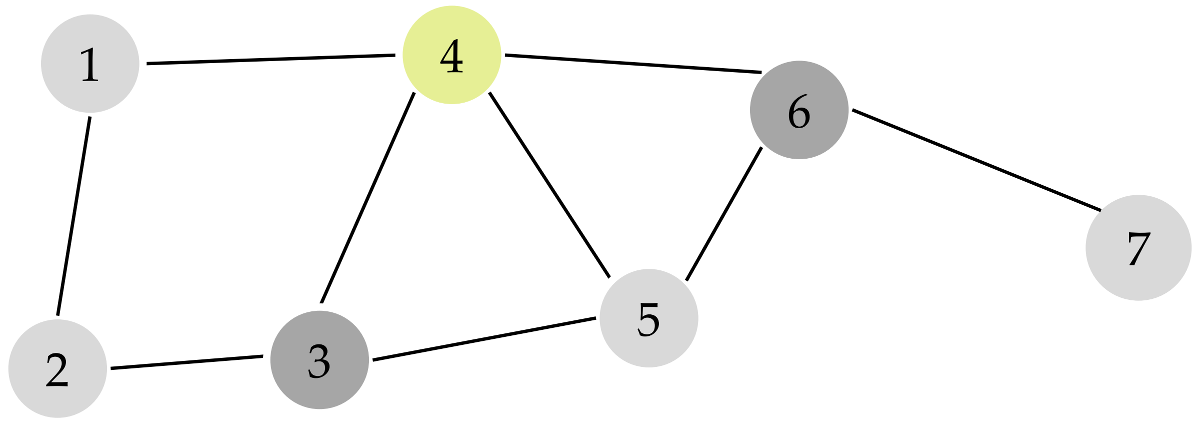
Multipoint Relays (MPR) 2

- Each node N in the network selects a set of nodes multipoint relays.
  - MPR(N) retransmit control packets from N.
  - Neighbours not in MPR(N) process control packets from N, but they do not forward the packets.

<u>MPR set</u>

- From the set of the node's one hop symmetric neighbours.
- Covers all symetric strict 2-hop nodes.
- The smaller the set, the less control traffic overhead.

Multipoint Relays

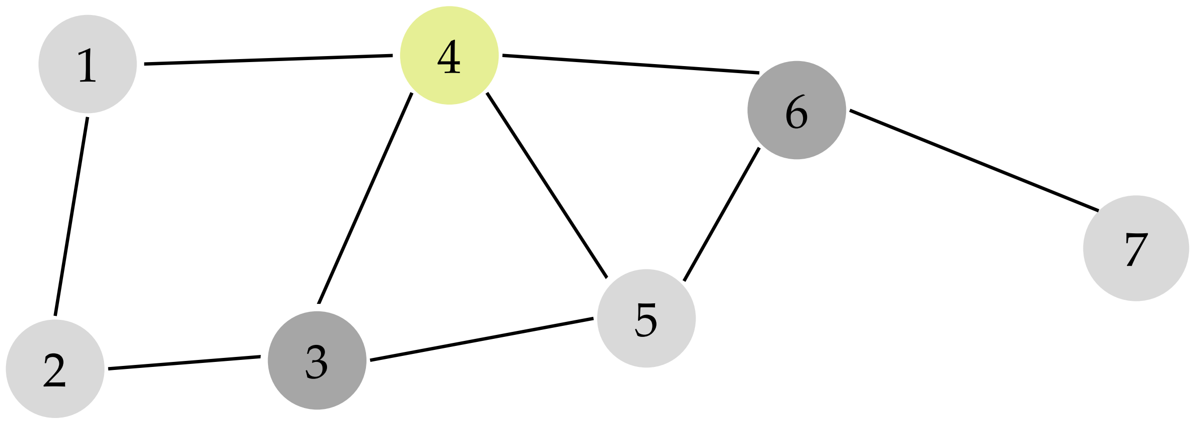


MPR (4) = {3, 6} → optimal set

Is there another optimal MPR(4) ?

Multipoint Relay Selector Set

- The multipoint relay selector set for Node N, MS(N), is the set of nodes that choose Node N in their multipoint relay set.
- A node obtains this information from periodic HELLO messages received from the neighbours.



MS(3) = {...., 4 ,.....}

MS(6) = {...., 4 ,.....}

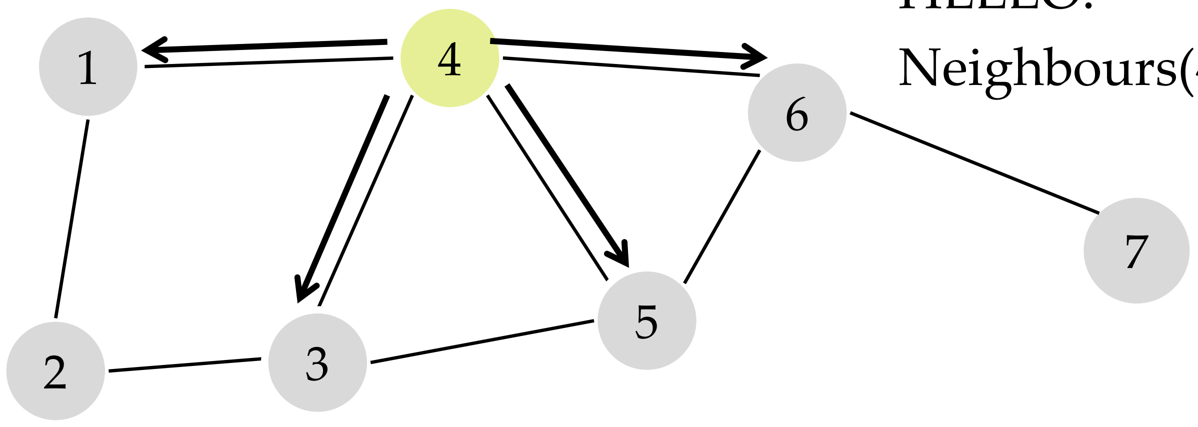
HELLO Messages

- Each node periodically broadcast HELLO messages
  - to learn the knowledge of its neighbours up to two hops.
  - and so, to determine its MPR (optimal or near-optimal) set.
- HELLO messages are not forwarded.

HELLO Messages

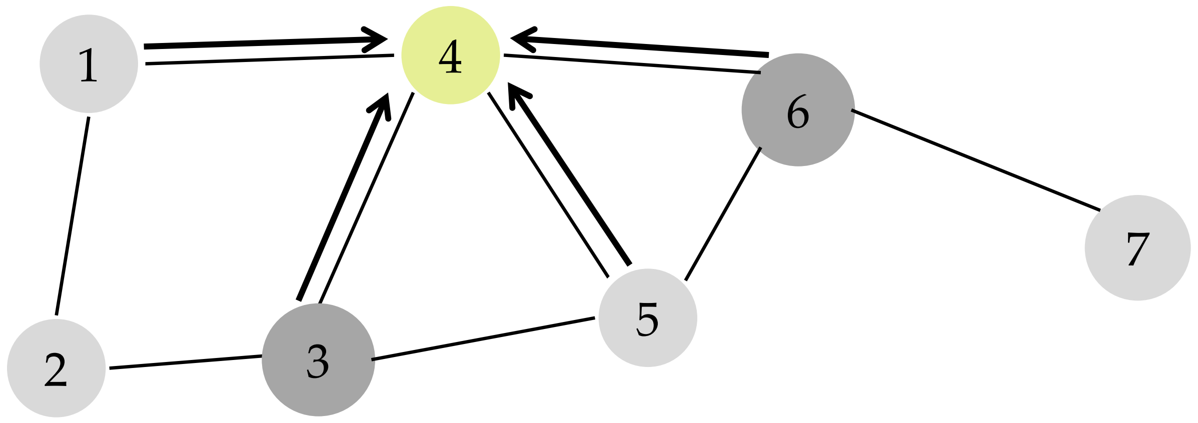
- The list of addresses of the neighbours to which there exists a valid *bi-directional* link.
- Subsequent HELLO messages also indicate the selected MPRs with the link status MPR.
- MPR set is recalculated when a change in the one-hop or two-hop neighborhood is detected.
- A sequence number is associated with this MPR set.
  - Sequence number is incremented each time a new set is calculated.

HELLO Messages



HELLO:

Neighbours(4) = {1,3,5,6}



Neighbours(1) = {2}

Neighbours(3) = {2,5}

Neighbours(5) = {3,6}

Neighbours(6) = {5,7}

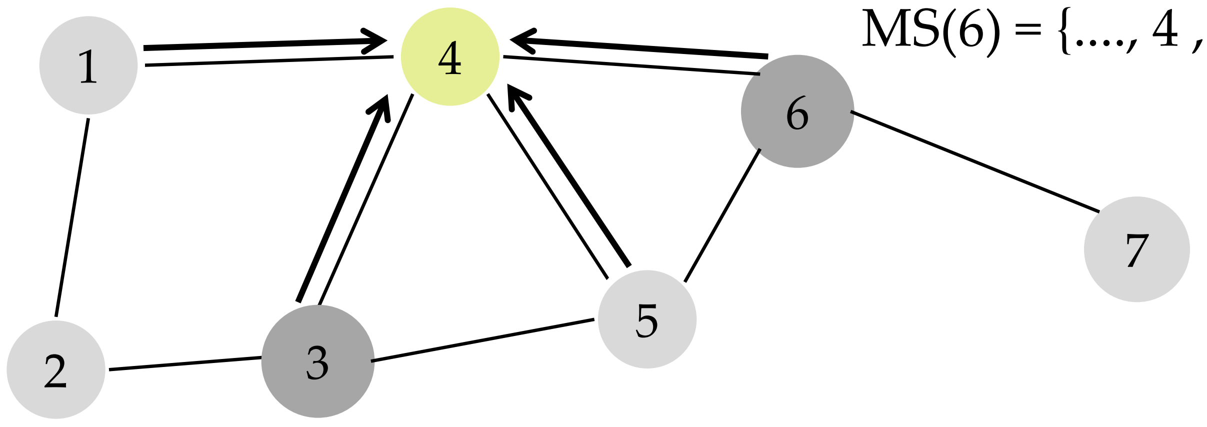
MPR(4) = {3,6}

Hello Messages

HELLO:

Neighbours(4) = {1,3,5,6}

MPR(4) = {3,6}



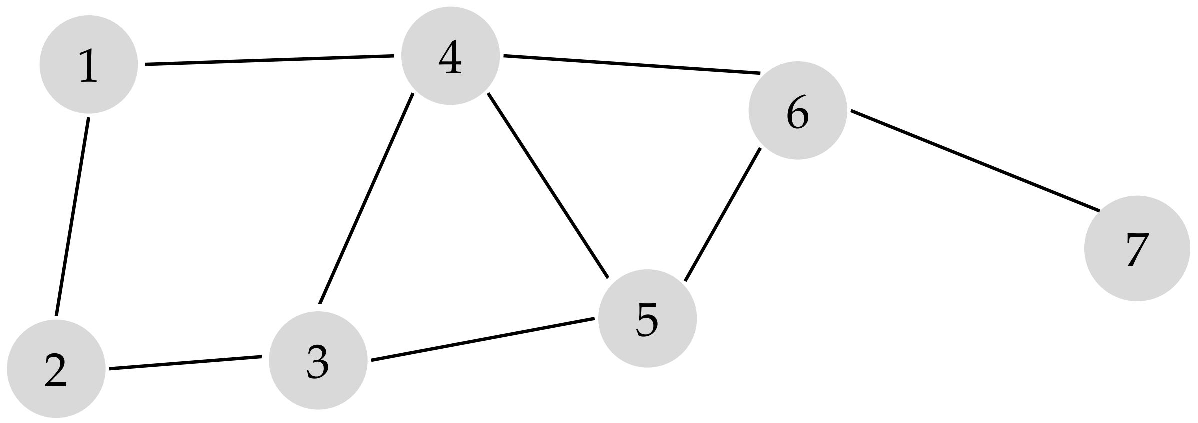
MS(6) = {…., 4 ,…}

MS(3) = {…., 4 ,…}

TC Messages

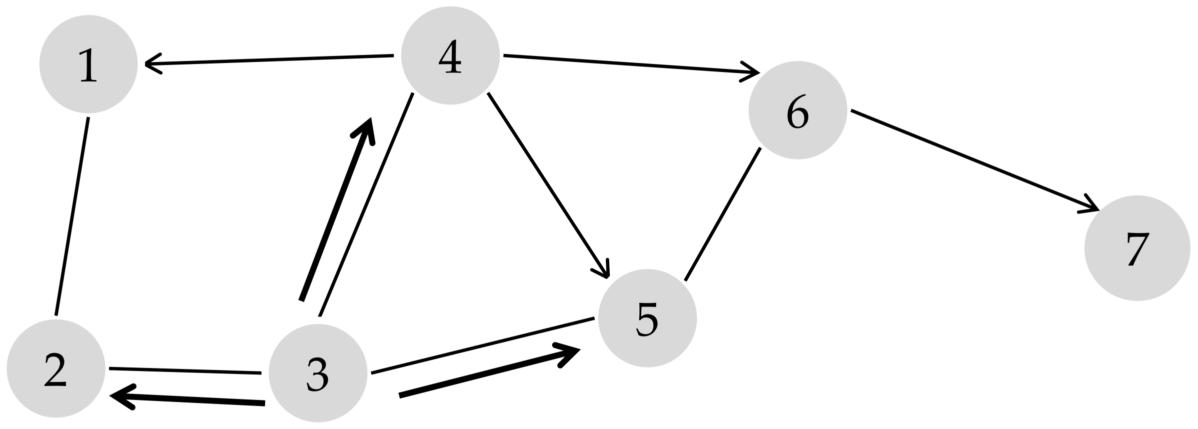
- Nodes send topology information in Topology Control (TC) messages.
  - List of advertised neighbours (link information).
  - Sequence number (to prevent use of stale information).
- A node generates TC messages only for those neighbours in its MS set.
  - Only MPR nodes generate TC messages.
  - Not all links are advertised.
- A nodes processes all received TC messages, but only forwards TC messages if the sender is in its MS set.
  - Only MPR nodes propagate TC messages.

OLSR Example (1)



MPR(1) = {4}	MS(1) = {}
MPR(2) = {3}	MS(2) = {}
MPR(3) = {4}	MS(3) = {2,4,5}
MPR(4) = {3,6}	MS(4) = {1,3,5,6}
MPR(5) = {3,4,6}	MS(5) = {}
MPR(6) = {4}	MS(6) = {4,5,7}
MPR(7) = {6}	MS(7) = {}

OLSR Example (2)



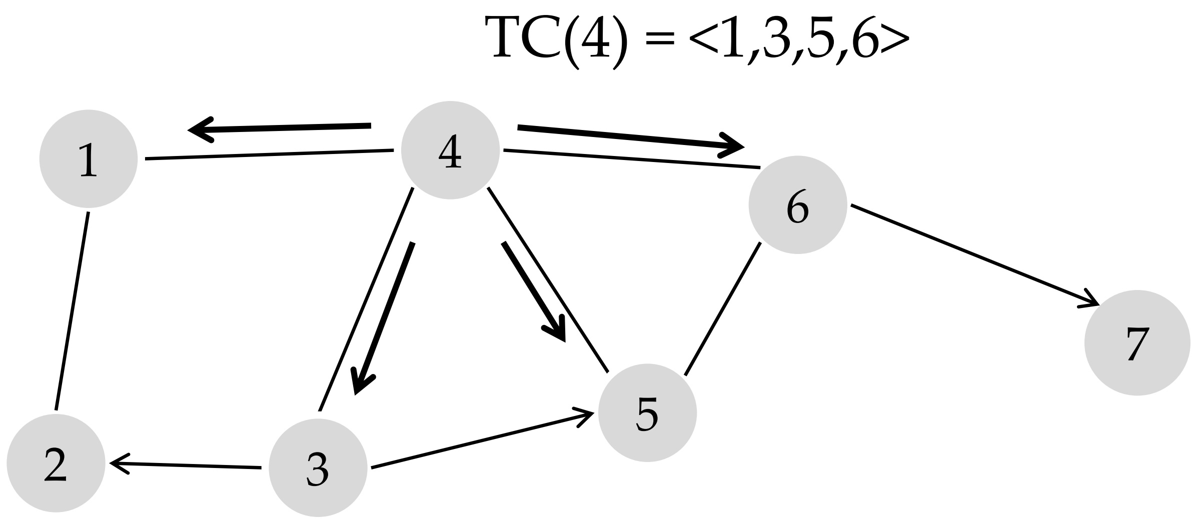
TC(3) = <2,4,5>

Node 3 generates a TC message advertising nodes in MS(3) = {2,4,5}

Node 4 forwards Node 3's TC message since Node 3 ϵ MS(4) = {1,3,5,6}

Node 6 forwards TC(3) since Node 4 ϵ MS(6)

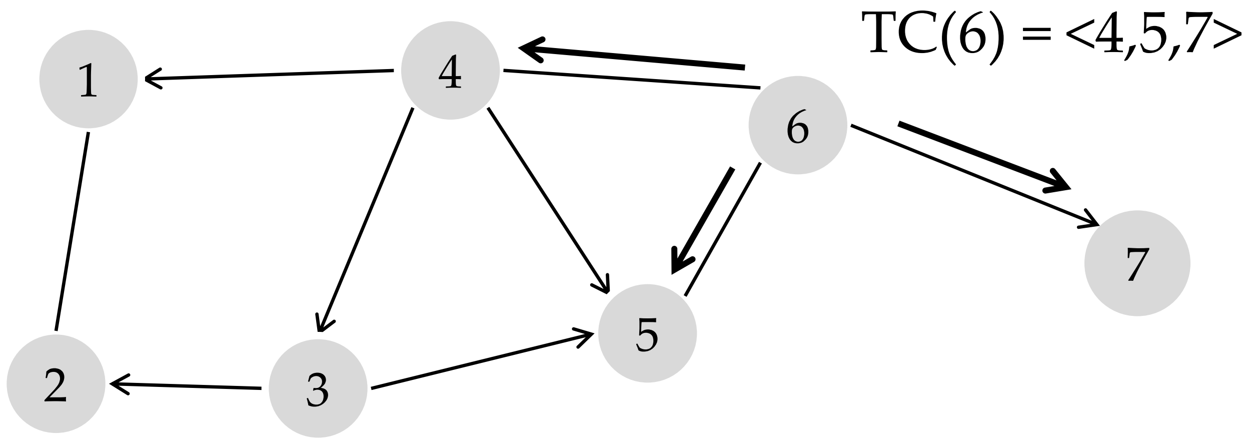
OLSR Example (3)



Node 4 generates a TC message advertising nodes in MS(4) = {1,3,5,6}

Nodes 3 and 6 forward TC(4) since Node 4 ϵ MS(3) and Node 4 ϵ MS(6)

OLSR Example (4)

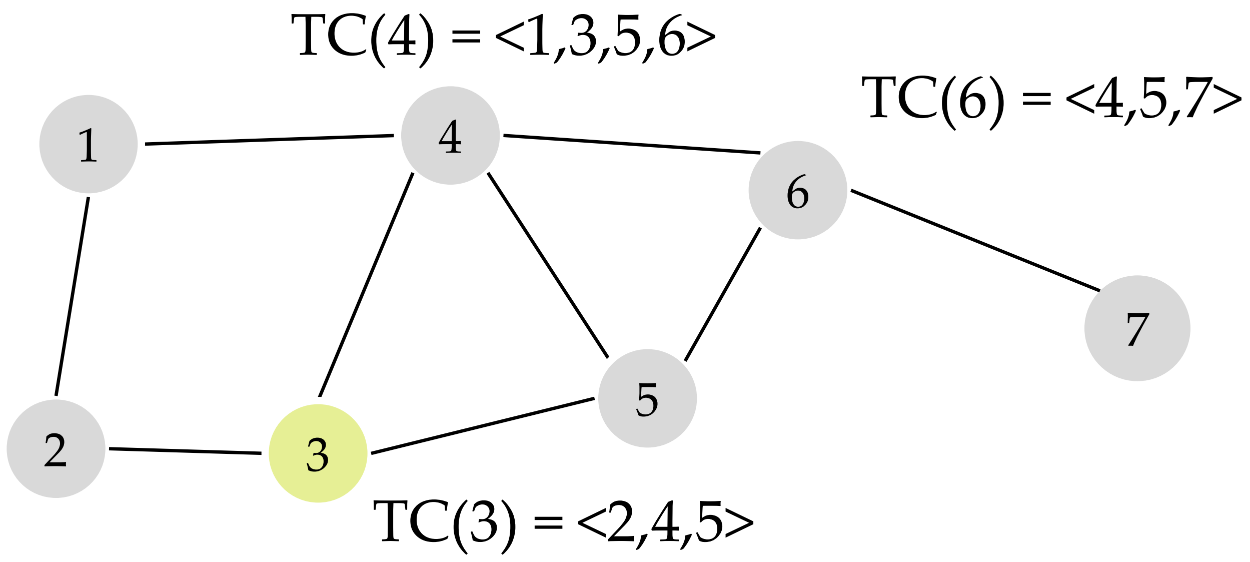


Node 6 generates a TC message advertising nodes in MS(6) = {4, 5, 7}.

Node 4 forwards TC(6) from Node 6 and Node 3 forwards TC(6) from Node 4.

After Nodes 3, 4, and 6 have generated TC messages, all nodes have link-state information to route to any node.

OLSR Example (5)



Dest	Next	Hop
1	4	2
2	2	1
4	4	1
5	5	1
6	4 (5)	2
7	4 (5)	3

- Given TC information, each node forms a topology table.
- A routing table is calculated from the topology table.

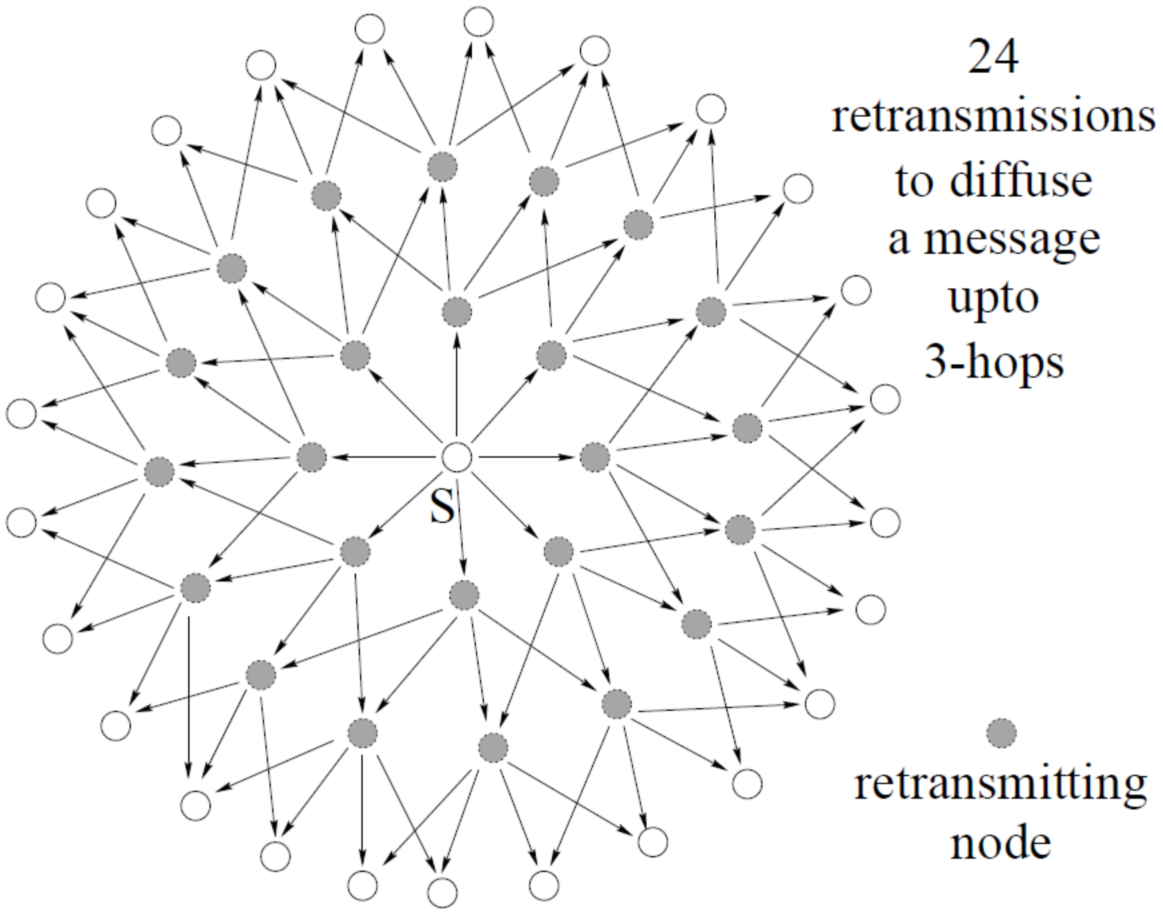


Figure 1: Diffusion of broadcast message using pure flooding

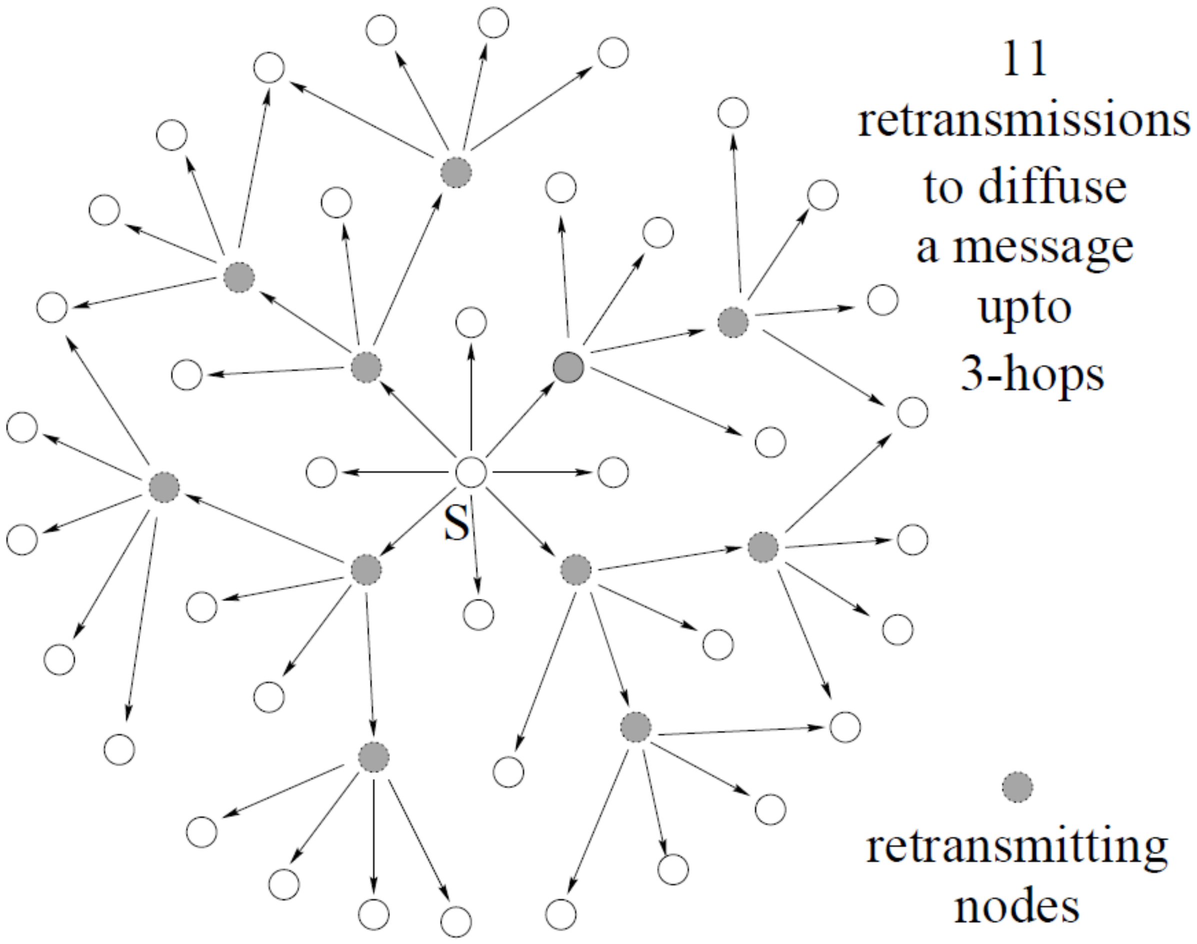


Figure 2: Diffusion of broadcast message using multipoint relays

Heuristic for the Selection of MPRs

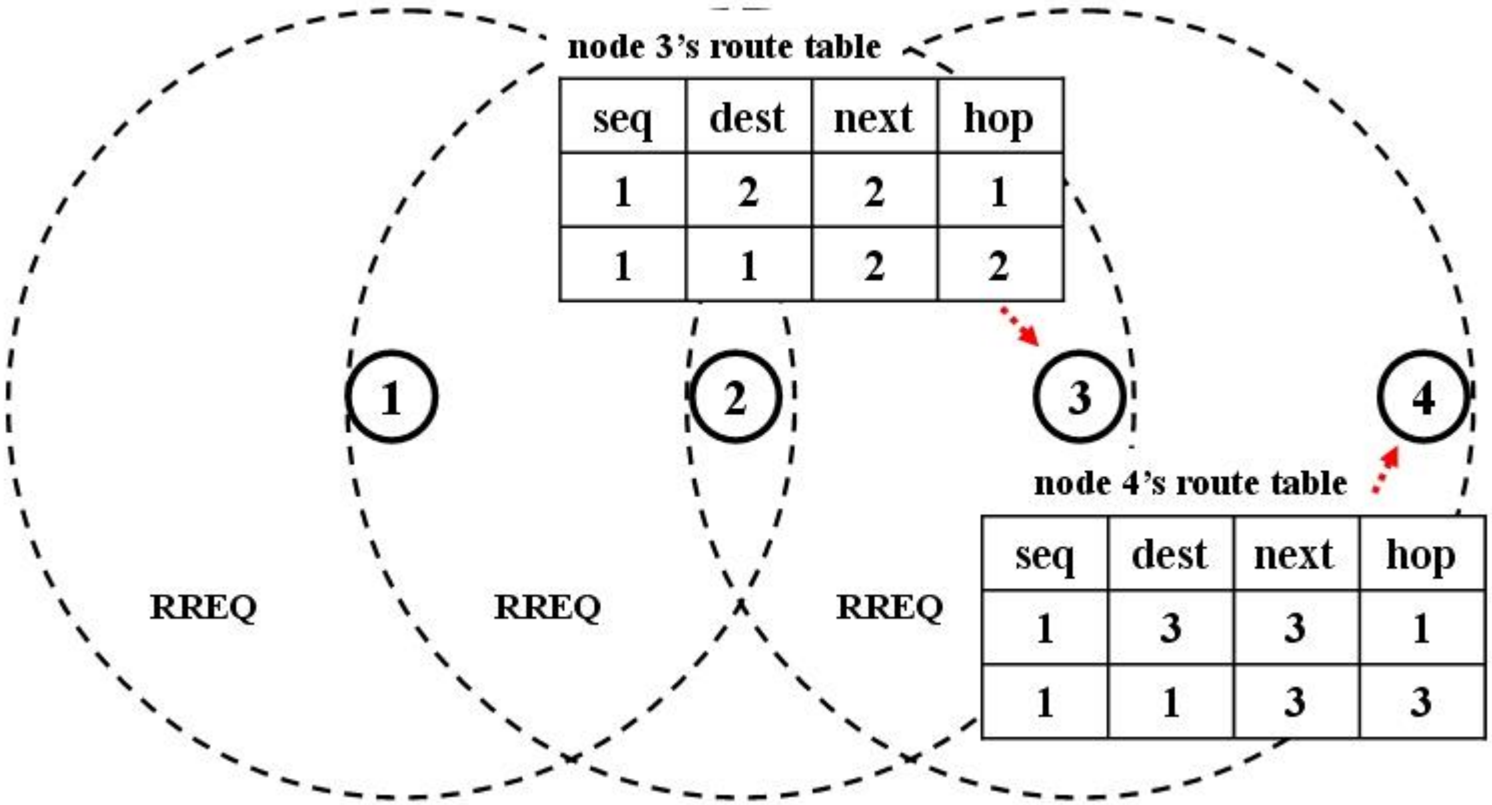
We propose here one heuristic for the selection of multipoint relays. To select the multipoint relays for the node x, lets call the the set of one-hop neighbors of node x as $N(x)$, and the set of its two-hop neighbors as $N^2(x)$. Let the selected multipoint relay set of node x be $MPR(x)$.

1. Start with an empty multipoint relay set $MPR(x)$
2. First select those one-hop neighbor nodes in $N(x)$ as the multipoint relays which are the only neighbor of some node in $N^2(x)$, and add these one-hop neighbor nodes to the multipoint relay set $MPR(x)$
3. While there still exist some node in $N^2(x)$ which is not covered by the multipoint relay set $MPR(x)$:
   (a) For each node in $N(x)$ which is not in $MPR(x)$, compute the number of nodes that it covers among the uncovered nodes in the set $N^2(x)$
   (b) Add that node of $N(x)$ in $MPR(x)$ for which this number is maximum.

References

1. A. K. Pathan, C. S. Hong, ‘Routing in Mobile Ad Hoc Networks’, Guide to Wireless Ad Hoc Networks, 2009
2. A. Quayyum, L. Viennot, A. Laouiti, ‘Multipoint Relaying : An Efficient Technique for Flooding in Mobile Wireless Networks’, 2000.
3. AODV, RFC 3561, http://www.rfc-editor.org/rfc/pdfrfc/rfc3561.txt.pdf
4. OLSR, RFC 3626, http://www.rfc-editor.org/rfc/pdfrfc/rfc3626.txt.pdf

Routing Table



node 3's route table

seq	dest	next	hop
1	2	2	1
1	1	2	2

node 4's route table

seq	dest	next	hop
1	3	3	1
1	1	3	3

Reading

C. E. Perkins, E. M. Royer, 'Ad-hoc On-Demand Distance Vector Routing', In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

D.B. Johnson, D.A. Maltz, J. Broch, 'DSR : The dynamic source routing protocol for multi-hop wireless ad hoc networks', Ad hoc networking, 2001.

D.B.Johnson, D.A. Maltz, 'Dynamic source routing in ad hoc wireless networks', Mobile computing, 1996.

P. Jacquet,et al. "Optimized link state routing protocol for ad hoc networks." *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*. IEEE, 2001.