Chapter 7

ROUTING PROTOCOLS FOR AD HOC WIRELESS NETWORKS

7.1 INTRODUCTION

An ad hoc wireless network consists of a set of mobile nodes (hosts) that are connected by wireless links. The network topology (the physical connectivity of the communication network) in such a network may keep changing randomly. Routing protocols that find a path to be followed by data packets from a source node to a destination node used in traditional wired networks cannot be directly applied in ad hoc wireless networks due to their highly dynamic topology, absence of established infrastructure for centralized administration (*e.g.*, base stations or access points), bandwidth-constrained wireless links, and resource (energy)-constrained nodes. A variety of routing protocols for ad hoc wireless networks has been proposed in the recent past. This chapter first presents the issues involved in designing a routing protocol and then the different classifications of routing protocols for ad hoc wireless networks. It then discusses the working of several existing routing protocols with illustrations.

7.2 ISSUES IN DESIGNING A ROUTING PROTOCOL FOR AD HOC WIRELESS NETWORKS

The major challenges that a routing protocol designed for ad hoc wireless networks faces are mobility of nodes, resource constraints, error-prone channel state, and hidden and exposed terminal problems. A detailed discussion on each of the following is given below.

7.2.1 Mobility

The network topology in an ad hoc wireless network is highly dynamic due to the movement of nodes, hence an on-going session suffers frequent path breaks. Disruption occurs either due to the movement of the intermediate nodes in the

path or due to the movement of end nodes. Such situations do not arise because of reliable links in wired networks where all the nodes are stationary. Even though the wired network protocols find alternate routes during path breaks, their convergence is very slow. Therefore, wired network routing protocols cannot be used in ad hoc wireless networks where the mobility of nodes results in frequently changing network topologies. Routing protocols for ad hoc wireless networks must be able to perform efficient and effective mobility management.

7.2.2 Bandwidth Constraint

Abundant bandwidth is available in wired networks due to the advent of fiber optics and due to the exploitation of wavelength division multiplexing (WDM) technologies. But in a wireless network, the radio band is limited, and hence the data rates it can offer are much less than what a wired network can offer. This requires that the routing protocols use the bandwidth optimally by keeping the overhead as low as possible. The limited bandwidth availability also imposes a constraint on routing protocols in maintaining the topological information. Due to the frequent changes in topology, maintaining a consistent topological information at all the nodes involves more control overhead which, in turn, results in more bandwidth wastage. As efficient routing protocols in wired networks require the complete topology information, they may not be suitable for routing in the ad hoc wireless networking environment.

7.2.3 Error-Prone Shared Broadcast Radio Channel

The broadcast nature of the radio channel poses a unique challenge in ad hoc wireless networks. The wireless links have time-varying characteristics in terms of link capacity and link-error probability. This requires that the ad hoc wireless network routing protocol interacts with the MAC layer to find alternate routes through better-quality links. Also, transmissions in ad hoc wireless networks result in collisions of data and control packets. This is attributed to the hidden terminal problem [1]. Therefore, it is required that ad hoc wireless network routing protocols find paths with less congestion.

7.2.4 Hidden and Exposed Terminal Problems

The hidden terminal problem refers to the collision of packets at a receiving node due to the simultaneous transmission of those nodes that are not within the direct transmission range of the sender, but are within the transmission range of the receiver. Collision occurs when both nodes transmit packets at the same time without knowing about the transmission of each other. For example, consider Figure 7.1. Here, if both node A and node C transmit to node B at the same time, their packets collide at node B. This is due to the fact that both nodes A and C are hidden from each other, as they are not within the direct transmission range of each other and hence do not know about the presence of each other. Solutions for this problem include medium access collision avoidance (MACA) [2], medium ac-

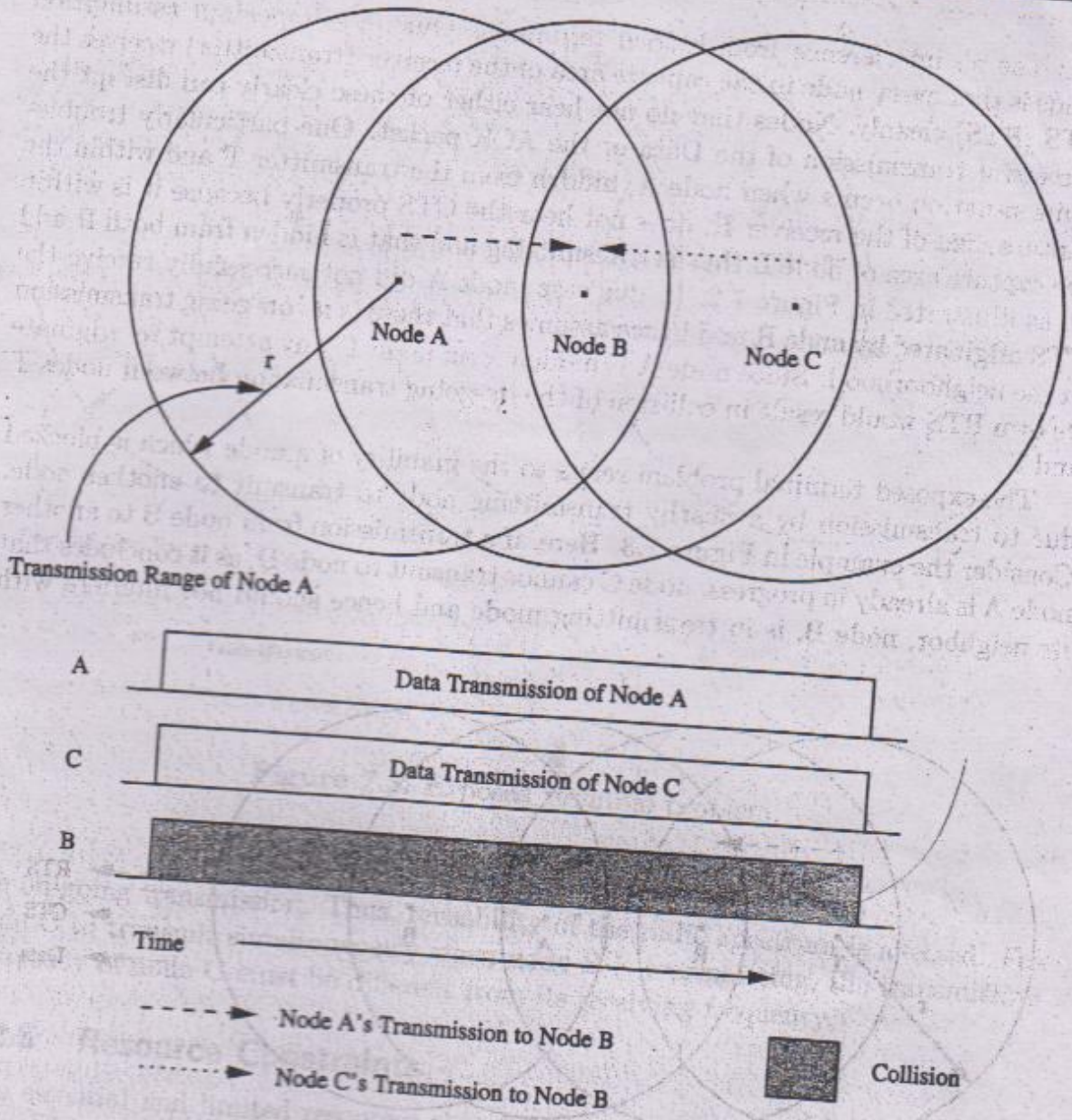


Figure 7.1. Hidden terminal problem.

cess collision avoidance for wireless (MACAW) [3], floor acquisition multiple access (FAMA) [4], and dual busy tone multiple access (DBTMA) [5]. MACA requires that a transmitting node first explicitly notifies all potential hidden nodes about the forthcoming transmission by means of a two-way handshake control protocol called the RTS-CTS protocol exchange. Note that this may not solve the problem completely, but it reduces the probability of collisions. To increase the efficiency, an improved version of the MACA protocol known as MACAW [3] has been proposed. This protocol requires that the receiver acknowledges each successful reception of a data packet. Hence, successful transmission is a four-way exchange mechanism, namely, RTS-CTS-Data-ACK. Even in the absence of bit errors and mobility, the RTS-CTS control packet exchange cannot ensure collision-free data transmission

that has no interference from hidden terminals. One very important assumption made is that every node in the capture area of the receiver (transmitter) receives the CTS (RTS) cleanly. Nodes that do not hear either of these clearly can disrupt the successful transmission of the Data or the ACK packet. One particularly troublesome situation occurs when node A, hidden from the transmitter T and within the capture area of the receiver R, does not hear the CTS properly because it is within the capture area of node B that is transmitting and that is hidden from both R and T, as illustrated in Figure 7.2. In this case, node A did not successfully receive the CTS originated by node R and hence assumes that there is no on-going transmission in the neighborhood. Since node A is hidden from node T, any attempt to originate its own RTS would result in collision of the on-going transmission between nodes T and R.

The exposed terminal problem refers to the inability of a node which is blocked due to transmission by a nearby transmitting node to transmit to another node. Consider the example in Figure 7.3. Here, if a transmission from node B to another node A is already in progress, node C cannot transmit to node D, as it concludes that its neighbor, node B, is in transmitting mode and hence should not interfere with

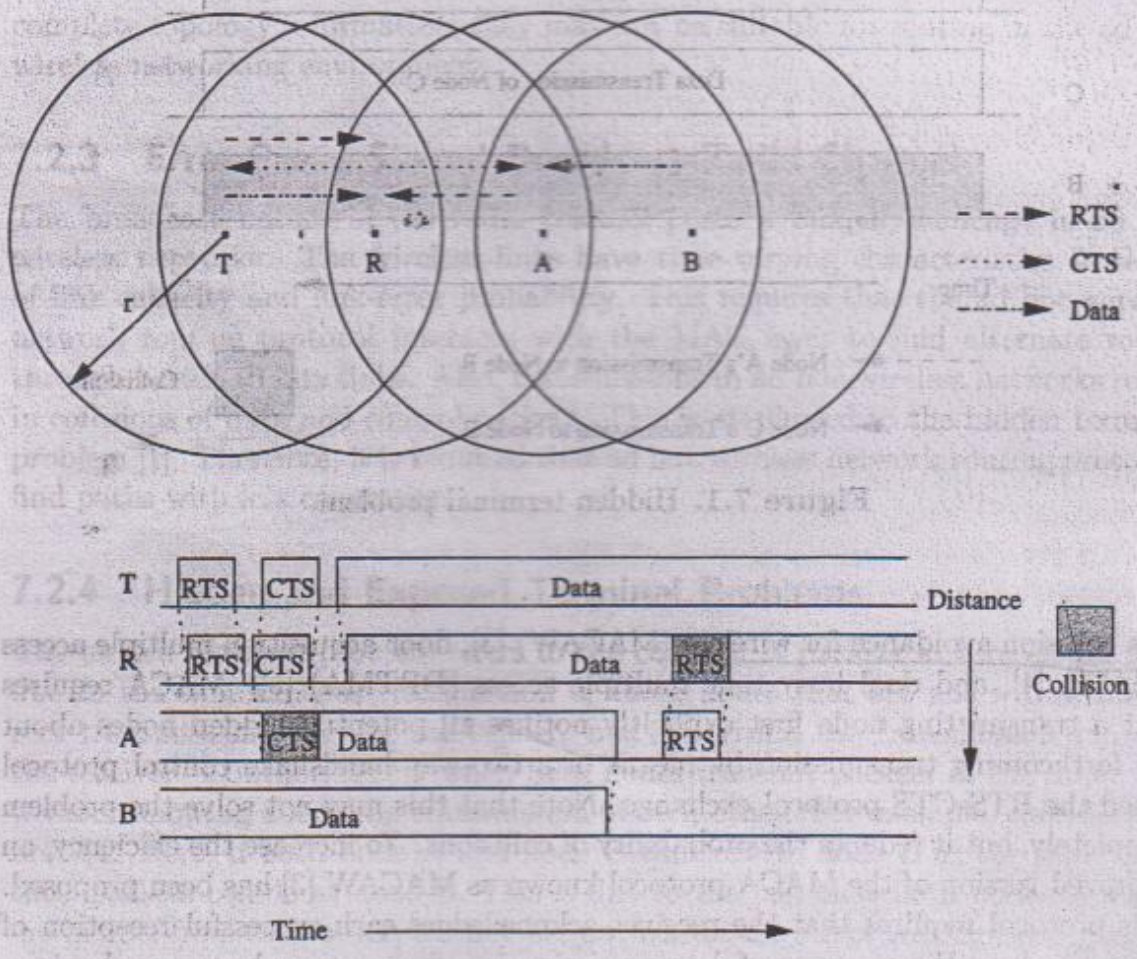


Figure 7.2. Hidden terminal problem with RTS-CTS-Data-ACK scheme.

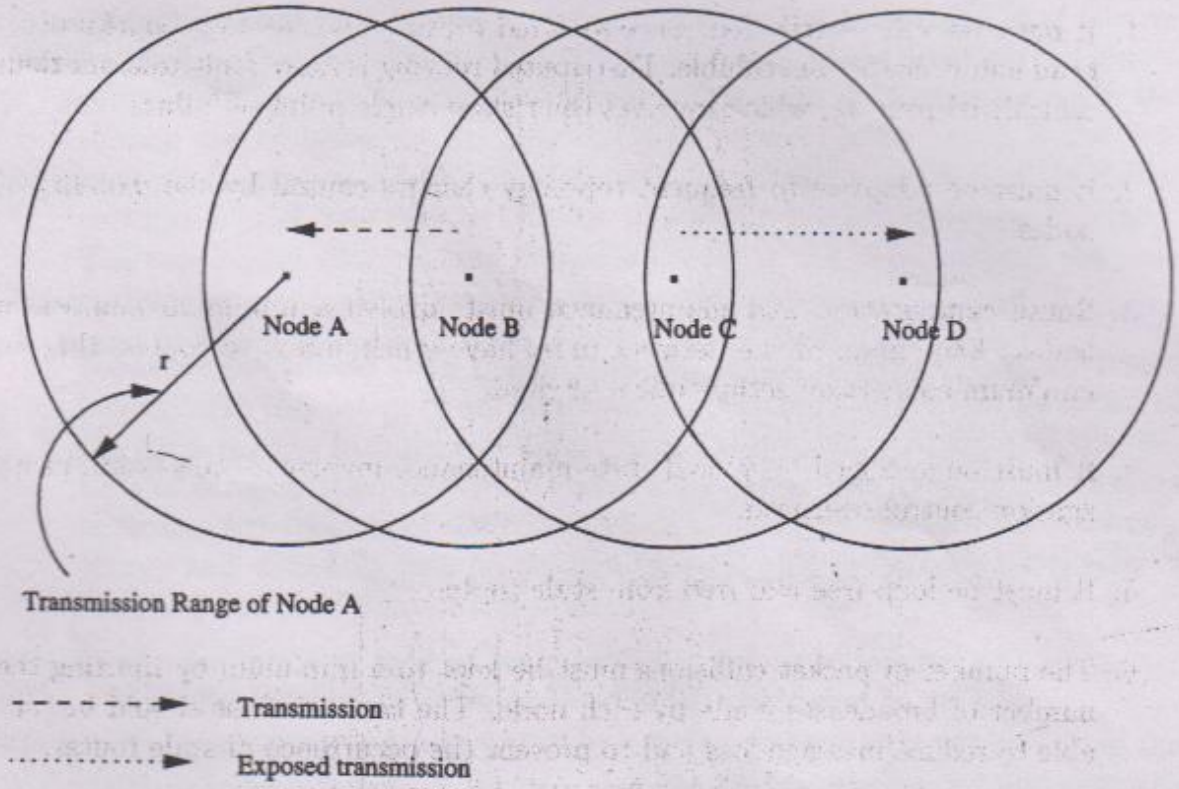


Figure 7.3. Exposed terminal problem.

the on-going transmission. Thus, reusability of the radio spectrum is affected. For node C to transmit simultaneously when node B is transmitting, the transmitting frequency of node C must be different from its receiving frequency.

7.2.5 Resource Constraints

Two essential and limited resources that form the major constraint for the nodes in an ad hoc wireless network are battery life and processing power. Devices used in ad hoc wireless networks in most cases require portability, and hence they also have size and weight constraints along with the restrictions on the power source. Increasing the battery power and processing ability makes the nodes bulky and less portable. Thus ad hoc wireless network routing protocols must optimally manage these resources.

7.2.6 Characteristics of an Ideal Routing Protocol for Ad Hoc Wireless Networks

Due to the issues in an ad hoc wireless network environment discussed so far, wired network routing protocols cannot be used in ad hoc wireless networks. Hence ad hoc wireless networks require specialized routing protocols that address the challenges described above. A routing protocol for ad hoc wireless networks should have the following characteristics:

1. It must be fully distributed, as centralized routing involves high control overhead and hence is not scalable. Distributed routing is more fault-tolerant than centralized routing, which involves the risk of single point of failure.

2. It must be adaptive to frequent topology changes caused by the mobility of nodes.

3. Route computation and maintenance must involve a minimum number of nodes. Each node in the network must have quick access to routes, that is, minimum connection setup time is desired.

4. It must be localized, as global state maintenance involves a huge state propagation control overhead.

5. It must be loop-free and free from stale routes.

6. The number of packet collisions must be kept to a minimum by limiting the number of broadcasts made by each node. The transmissions should be reliable to reduce message loss and to prevent the occurrence of stale routes.

7. It must converge to optimal routes once the network topology becomes stable. The convergence must be quick.

8. It must optimally use scarce resources such as bandwidth, computing power, memory, and battery power.

9. Every node in the network should try to store information regarding the stable local topology only. Frequent changes in local topology, and changes in the topology of parts of the network with which the node does not have any traffic correspondence, must not in any way affect the node, that is, changes in remote parts of the network must not cause updates in the topology information maintained by the node.

10. It should be able to provide a certain level of quality of service (QoS) as demanded by the applications, and should also offer support for time-sensitive traffic.

7.3 CLASSIFICATIONS OF ROUTING PROTOCOLS

Routing protocols for ad hoc wireless networks can be classified into several types based on different criteria. A classification tree is shown in Figure 7.4. Some of the classifications, their properties, and the basis of classifications are discussed below. The classification is not mutually exclusive and some protocols fall in more than one class. The deviation from the traditional routing metrics and path-finding processes that are employed in wired networks makes it worth further exploration

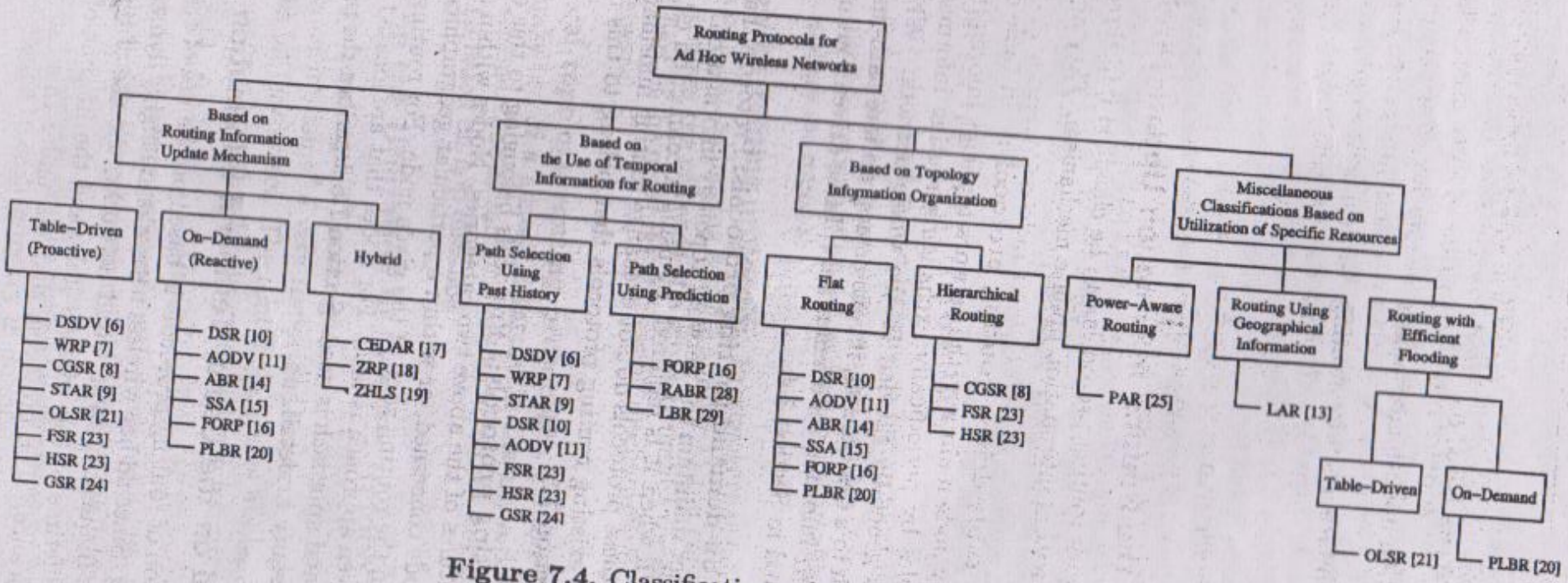


Figure 7.4. Classifications of routing protocols.

in this direction. The routing protocols for ad hoc wireless networks can be broadly classified into four categories based on

- Routing information update mechanism
- Use of temporal information for routing
- Routing topology
- Utilization of specific resources

7.3.1 Based on the Routing Information Update Mechanism

Ad hoc wireless network routing protocols can be classified into three major categories based on the routing information update mechanism. They are:

1. **Proactive or table-driven routing protocols:** In table-driven routing protocols, every node maintains the network topology information in the form of routing tables by periodically exchanging routing information. Routing information is generally flooded in the whole network. Whenever a node requires a path to a destination, it runs an appropriate path-finding algorithm on the topology information it maintains. Table-driven routing protocols are further explored in Section 7.4.

2. **Reactive or on-demand routing protocols:** Protocols that fall under this category do not maintain the network topology information. They obtain the necessary path when it is required, by using a connection establishment process. Hence these protocols do not exchange routing information periodically. Some of the existing routing protocols that belong to this category are discussed in Section 7.5.

3. **Hybrid routing protocols:** Protocols belonging to this category combine the best features of the above two categories. Nodes within a certain distance from the node concerned, or within a particular geographical region, are said to be within the routing zone of the given node. For routing within this zone, a table-driven approach is used. For nodes that are located beyond this zone, an on-demand approach is used. Section 7.6 describes the protocols belonging to this category in detail.

7.3.2 Based on the Use of Temporal Information for Routing

This classification of routing protocols is based on the use of temporal information used for routing. Since ad hoc wireless networks are highly dynamic and path breaks are much more frequent than in wired networks, the use of temporal information regarding the lifetime of the wireless links and the lifetime of the paths selected assumes significance. The protocols that fall under this category can be further classified into two types:

1. **Routing protocols using past temporal information:** These routing protocols use information about the past status of the links or the status of links at the time of routing to make routing decisions. For example, the routing metric based on the availability of wireless links (which is the current/present information here) along with a shortest path-finding algorithm, provides a path that may be efficient and stable at the time of path-finding. The topological changes may immediately break the path, making the path undergo a resource-wise expensive path reconfiguration process.

2. **Routing protocols that use future temporal information:** Protocols belonging to this category use information about the expected future status of the wireless links to make approximate routing decisions. Apart from the lifetime of wireless links, the future status information also includes information regarding the lifetime of the node (which is based on the remaining battery charge and discharge rate of the non-replenishable resources), prediction of location, and prediction of link availability.

7.3.3 Based on the Routing Topology

Routing topology being used in the Internet is hierarchical in order to reduce the state information maintained at the core routers. Ad hoc wireless networks, due to their relatively smaller number of nodes, can make use of either a flat topology or a hierarchical topology for routing.

1. **Flat topology routing protocols:** Protocols that fall under this category make use of a flat addressing scheme similar to the one used in IEEE 802.3 LANs. It assumes the presence of a globally unique (or at least unique to the connected part of the network) addressing mechanism for nodes in an ad hoc wireless network.

2. **Hierarchical topology routing protocols:** Protocols belonging to this category make use of a logical hierarchy in the network and an associated addressing scheme. The hierarchy could be based on geographical information or it could be based on hop distance.

7.3.4 Based on the Utilization of Specific Resources

1. **Power-aware routing:** This category of routing protocols aims at minimizing the consumption of a very important resource in the ad hoc wireless networks: the battery power. The routing decisions are based on minimizing the power consumption either locally or globally in the network.

2. **Geographical information assisted routing:** Protocols belonging to this category improve the performance of routing and reduce the control overhead by effectively utilizing the geographical information available.

The following section further explores the above classifications and discusses specific routing protocols belonging to each category in detail.

7.4 TABLE-DRIVEN ROUTING PROTOCOLS

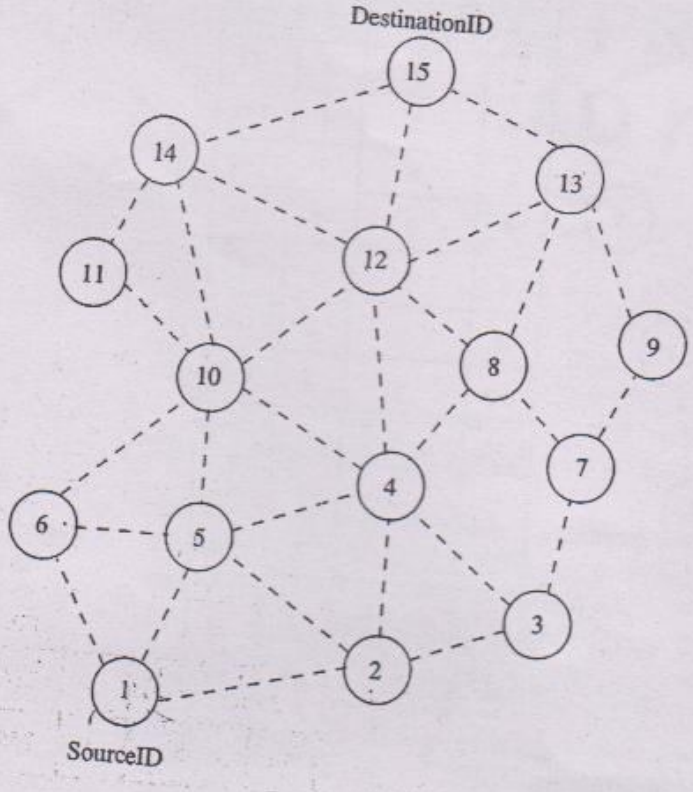
These protocols are extensions of the wired network routing protocols. They maintain the global topology information in the form of tables at every node. These tables are updated frequently in order to maintain consistent and accurate network state information. The destination sequenced distance-vector routing protocol (DSDV), wireless routing protocol (WRP), source-tree adaptive routing protocol (STAR), and cluster-head gateway switch routing protocol (CGSR) are some examples for the protocols that belong to this category.

7.4.1 Destination Sequenced Distance-Vector Routing Protocol

The destination sequenced distance-vector routing protocol (DSDV) [6] is one of the first protocols proposed for ad hoc wireless networks. It is an enhanced version of the distributed Bellman-Ford algorithm where each node maintains a table that contains the shortest distance and the first node on the shortest path to every other node in the network. It incorporates table updates with increasing sequence number tags to prevent loops, to counter the count-to-infinity problem, and for faster convergence.

As it is a table-driven routing protocol, routes to all destinations are readily available at every node at all times. The tables are exchanged between neighbors at regular intervals to keep an up-to-date view of the network topology. The tables are also forwarded if a node observes a significant change in local topology. The table updates are of two types: incremental updates and full dumps. An incremental update takes a single network data packet unit (NDPU), while a full dump may take multiple NDPUs. Incremental updates are used when a node does not observe significant changes in the local topology. A full dump is done either when the local topology changes significantly or when an incremental update requires more than a single NDPU. Table updates are initiated by a destination with a new sequence number which is always greater than the previous one. Upon receiving an updated table, a node either updates its tables based on the received information or holds it for some time to select the best metric (which may be the lowest number of hops) received from multiple versions of the same update table from different neighboring nodes. Based on the sequence number of the table update, it may forward or reject the table. Consider the example as shown in Figure 7.5 (a). Here node 1 is the source node and node 15 is the destination. As all the nodes maintain global topology information, the route is already available as shown in Figure 7.5 (b). Here the routing table of node 1 indicates that the shortest route to the destination node (node 15) is available through node 5 and the distance to it is 4 hops, as depicted in Figure 7.5 (b).

The reconfiguration of a path used by an on-going data transfer session is handled by the protocol in the following way. The end node of the broken link initiates a table update message with the broken link's weight assigned to infinity (∞) and with a sequence number greater than the stored sequence number for that destination. Each node, upon receiving an update with weight ∞, quickly disseminates it to its neighbors in order to propagate the broken-link information to the whole net-



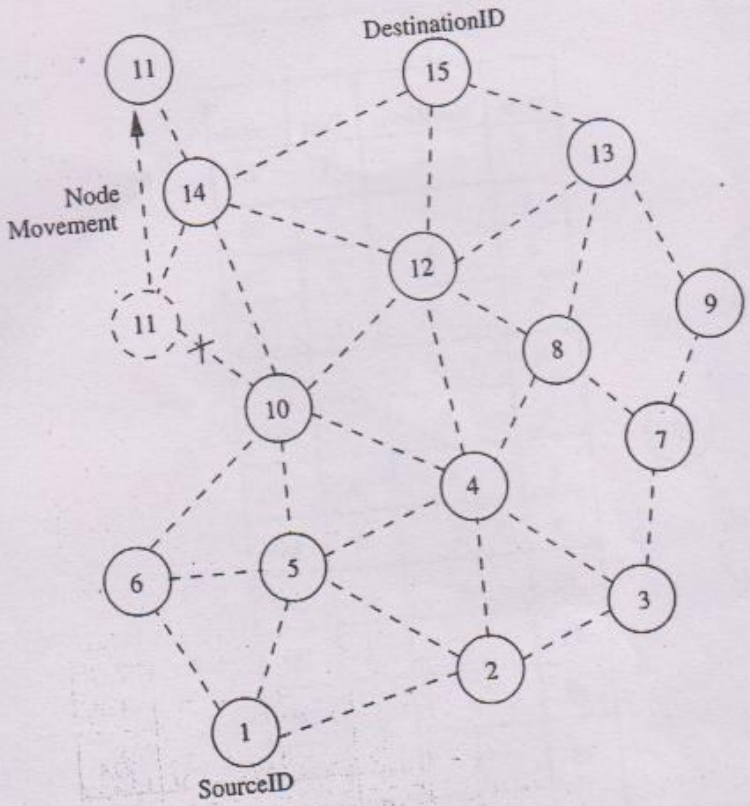
(a) Topology graph of the network

Dest	NextNode	Dist	SeqNo
2	2	1	22
3	2	2	26
4	5	2	32
5	5	1	134
6	6	1	144
7	2	3	162
8	5	3	170
9	2	4	186
10	6	2	142
11	6	3	176
12	5	3	190
13	5	4	198
14	6	3	214
15	5	4	256

(b) Routing table for Node 1

Figure 7.5. Route establishment in DSDV.

work. Thus a single link break leads to the propagation of table update information to the whole network. A node always assigns an odd sequence number to the link break update to differentiate it from the even sequence number generated by the destination. Consider the case when node 11 moves from its current position, as shown in Figure 7.6. When a neighbor node perceives the link break, it sets all the paths passing through the broken link with distance as ∞. For example, when node 10 knows about the link break, it sets the path to node 11 as ∞ and broadcasts its routing table to its neighbors. Those neighbors detecting significant changes in their routing tables rebroadcast it to their neighbors. In this way, the broken link information propagates throughout the network. Node 1 also sets the distance to node 11 as ∞. When node 14 receives a table update message from node 11, it informs the neighbors about the shortest distance to node 11. This information is also propagated throughout the network. All nodes receiving the new update message with the higher sequence number set the new distance to node 11 in their corresponding tables. The updated table at node 1 is shown in Figure 7.6, where the current distance from node 1 to node 11 has increased from three to four hops.



Routing Table for Node 1

Dest	NextNode	Dist	SeqNo
2	2	1	22
3	2	2	26
4	5	2	32
5	5	1	134
6	6	1	144
7	2	3	162
8	5	3	170
9	2	4	186
10	6	2	142
11	5	4	180
12	5	3	190
13	5	4	198
14	6	3	214
15	5	4	256

Figure 7.6. Route maintenance in DSDV.

Advantages and Disadvantages

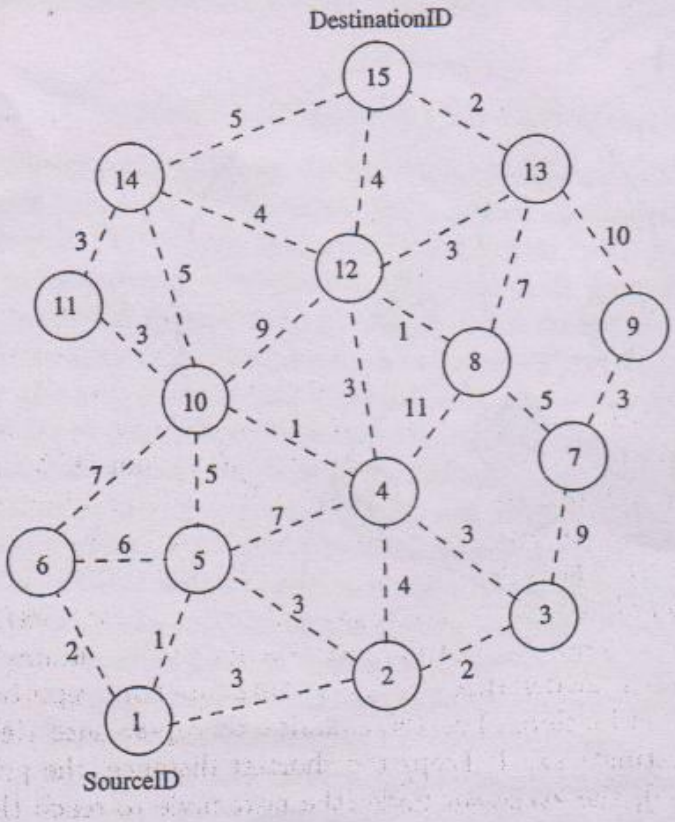
The availability of routes to all destinations at all times implies that much less delay is involved in the route setup process. The mechanism of incremental updates with sequence number tags makes the existing wired network protocols adaptable to ad hoc wireless networks. Hence, an existing wired network protocol can be applied to ad hoc wireless networks with many fewer modifications. The updates are propagated throughout the network in order to maintain an up-to-date view of the network topology at all the nodes. The updates due to broken links lead to a heavy control overhead during high mobility. Even a small network with high mobility or a large network with low mobility can completely choke the available bandwidth. Hence, this protocol suffers from excessive control overhead that is proportional to the number of nodes in the network and therefore is not scalable in ad hoc wireless networks, which have limited bandwidth and whose topologies are highly dynamic. Another disadvantage of DSDV is that in order to obtain information about a particular destination node, a node has to wait for a table

update message initiated by the same destination node. This delay could result in stale routing information at nodes.

7.4.2 Wireless Routing Protocol

The wireless routing protocol (WRP) [7], similar to DSDV, inherits the properties of the distributed Bellman-Ford algorithm. To counter the count-to-infinity problem and to enable faster convergence, it employs a unique method of maintaining information regarding the shortest distance to every destination node in the network and the penultimate hop node on the path to every destination node. Since WRP, like DSDV, maintains an up-to-date view of the network, every node has a readily available route to every destination node in the network. It differs from DSDV in table maintenance and in the update procedures. While DSDV maintains only one topology table, WRP uses a set of tables to maintain more accurate information. The tables that are maintained by a node are the following: distance table (DT), routing table (RT), link cost table (LCT), and a message retransmission list (MRL).

The DT contains the network view of the neighbors of a node. It contains a matrix where each element contains the distance and the penultimate node reported by a neighbor for a particular destination. The RT contains the up-to-date view of the network for all known destinations. It keeps the shortest distance, the *predecessor* node (penultimate node), the *successor* node (the next node to reach the destination), and a flag indicating the status of the path. The path status may be a simple path (correct), or a loop (error), or the destination node not marked (null). The LCT contains the cost (*e.g.*, the number of hops to reach the destination) of relaying messages through each link. The cost of a broken link is ∞. It also contains the number of update periods (intervals between two successive periodic updates) passed since the last successful update was received from that link. This is done to detect link breaks. The MRL contains an entry for every update message that is to be retransmitted and maintains a counter for each entry. This counter is decremented after every retransmission of an update message. Each update message contains a list of updates. A node also marks each node in the RT that has to acknowledge the update message it transmitted. Once the counter reaches zero, the entries in the update message for which no acknowledgments have been received are to be retransmitted and the update message is deleted. Thus, a node detects a link break by the number of update periods missed since the last successful transmission. After receiving an update message, a node not only updates the distance for transmitted neighbors but also checks the other neighbors' distance, hence convergence is much faster than DSDV. Consider the example shown in Figure 7.7, where the source of the route is node 1 and the destination is node 15. As WRP proactively maintains the route to all the destinations, the route to any destination node is readily available at the source node. From the routing table shown in Figure 7.7, the route from node 1 to node 15 has the next node as node 2. The predecessor node of 15 corresponding to this route is node 12. The predecessor information helps WRP to converge quickly during link breaks.

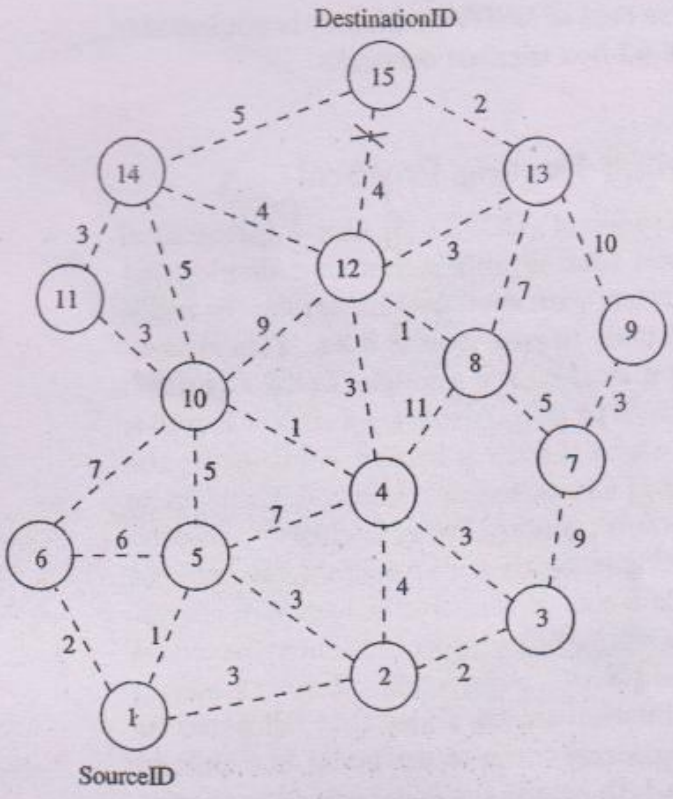


Routing Entry at Each Node for DestinationID 15

Node	NextNode	Pred	Cost
15	15	15	0
14	15	14	5
13	15	13	2
12	15	12	4
11	14	14	8
10	4	12	8
9	13	13	12
8	12	12	5
7	8	12	10
6	10	12	15
5	10	12	13
4	12	12	10
3	4	12	7
2	4	12	11
1	2	12	14

Figure 7.7. Route establishment in WRP.

When a node detects a link break, it sends an update message to its neighbors with the link cost of the broken link set to ∞. After receiving the update message, all affected nodes update their minimum distances to the corresponding nodes (including the distance to the destination). The node that initiated the update message then finds an alternative route, if available from its DT. Note that this new route computed will not contain the broken link. Consider the scenario shown in Figure 7.8. When the link between nodes 12 and 15 breaks, all nodes having a route to the destination with predecessor as node 12 delete their corresponding routing entries. Both node 12 and node 15 send update messages to their neighbors indicating that the cost of the link between nodes 12 and 15 is ∞. If the nodes have any other alternative route to the destination node 15, they update their routing tables and indicate the changed route to their neighbors by sending an update message. A neighbor node, after receiving an update message, updates its routing table only if the new path is better than the previously existing paths. For example, when node 12 finds an alternative route to the destination through node 13, it broadcasts an



Routing Entry at Each Node for DestinationID 15

Node	NextNode	Pred	Cost
15	15	15	0
14	15	14	5
13	15	13	2
12	15	13	5
11	14	14	8
10	4	13	9
9	13	13	12
8	12	13	6
7	8	13	11
6	10	13	16
5	10	13	14
4	12	13	8
3	4	13	11
2	4	13	12
1	2	13	15

Figure 7.8. Route maintenance in WRP.

update message indicating the changed path. After receiving the update message from node 12, neighboring nodes 8, 14, 15, and 13 do not change their routing entry corresponding to destination 15 while node 4 and node 10 modify their entries to reflect the new updated path. Nodes 4 and 10 again send an update message to indicate the correct path to the destination for their respective neighbors. When node 10 receives node 4's update message, it again modifies its routing entry to optimize the path to the destination node (15) while node 4 discards the update entry it received from node 10.

Advantages and Disadvantages

WRP has the same advantages as that of DSDV. In addition, it has faster convergence and involves fewer table updates. But the complexity of maintenance of multiple tables demands a larger memory and greater processing power from nodes in the ad hoc wireless network. At high mobility, the control overhead involved in

updating table entries is almost the same as that of DSDV and hence is not suitable for highly dynamic and also for very large ad hoc wireless networks.

7.4.3 Cluster-Head Gateway Switch Routing Protocol

The cluster-head gateway switch routing protocol (CGSR) [8] uses a hierarchical network topology, unlike other table-driven routing approaches that employ flat topologies. CGSR organizes nodes into clusters, with coordination among the members of each cluster entrusted to a special node named *cluster-head*. This cluster-head is elected dynamically by employing a *least cluster change (LCC)* algorithm [8]. According to this algorithm, a node ceases to be a cluster-head only if it comes under the range of another cluster-head, where the tie is broken either using the lowest ID or highest connectivity algorithm. Clustering provides a mechanism to allocate bandwidth, which is a limited resource, among different clusters, thereby improving reuse. For example, different cluster-heads could operate on different spreading codes on a CDMA system. Inside a cluster, the cluster-head can coordinate the channel access based on a token-based polling protocol. All member nodes of a cluster can be reached by the cluster-head within a single hop, thereby enabling the cluster-head to provide improved coordination among nodes that fall under its cluster. A token-based scheduling (assigning access token to the nodes in a cluster) is used within a cluster for sharing the bandwidth among the members of the cluster. CGSR assumes that all communication passes through the cluster-head. Communication between two clusters takes place through the common member nodes that are members of both the clusters. These nodes which are members of more than one cluster are called *gateways*. A gateway is expected to be able to listen to multiple spreading codes that are currently in operation in the clusters in which the node exists as a member. A gateway conflict is said to occur when a cluster-head issues a token to a gateway over a spreading code while the gateway is tuned to another code. Gateways that are capable of simultaneously communicating over two interfaces can avoid gateway conflicts.

The performance of routing is influenced by token scheduling and code scheduling (assigning appropriate spreading codes to two different clusters) that are handled at cluster-heads and gateways, respectively. The routing protocol used in CGSR is an extension of DSDV. Every member node maintains a routing table containing the destination cluster-head for every node in the network. In addition to the cluster member table, each node maintains a routing table which keeps the list of next-hop nodes for reaching every destination cluster. The *cluster (hierarchical) routing protocol* is used here. As per this protocol, when a node with packets to be transmitted to a destination gets the token from its cluster-head, it obtains the destination cluster-head and the next-hop node from the cluster member table and the routing table, respectively. CGSR improves the routing performance by routing packets through the cluster-heads and gateways. A path from any node a to any node b will be similar to $a - C_1 - G_1 - C_2 - G_2 - \ldots C_i - G_j \ldots G_n - b$, where G_i and C_j are the i^{th} gateway and the j^{th} cluster-head, respectively, in the path. Figure 7.9 shows the cluster-heads, *cluster gateways*, and normal cluster member

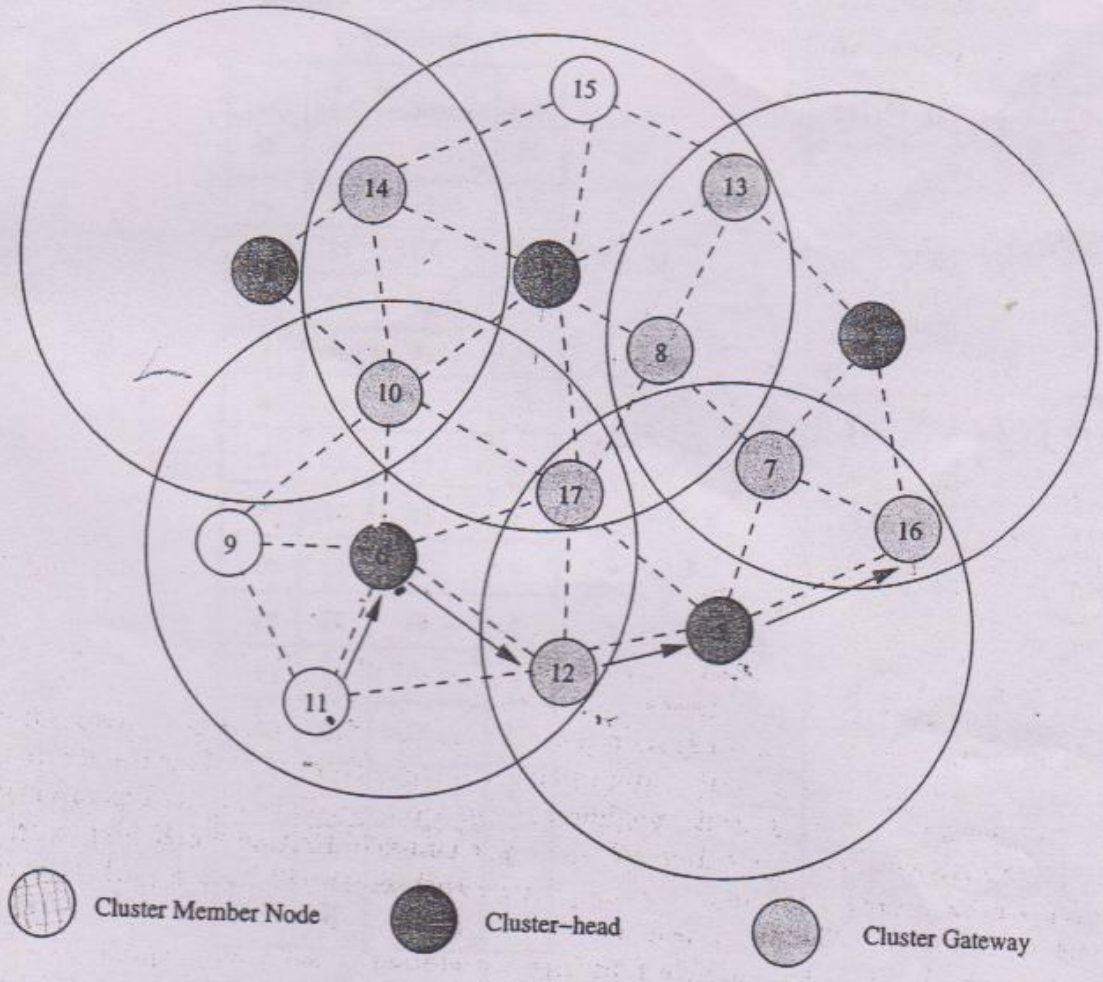


Figure 7.9. Route establishment in CGSR.

nodes in an ad hoc wireless network. A path between node 11 and node 16 would follow $11 - 6 - 12 - 5 - 16$. Since the cluster-heads gain more opportunities for transmission, the cluster-heads, by means of a dynamic scheduling mechanism, can make CGSR obtain better delay performance for real-time flows. Route reconfiguration is necessitated by mainly two factors: firstly, the change in cluster-head and secondly, the stale entries in the cluster member table and routing table. CGSR depends on the table update mechanism to handle the latter problem, while the least cluster change algorithm [8] handles the former.

Advantages and Disadvantages

CGSR is a hierarchical routing scheme which enables partial coordination between nodes by electing cluster-heads. Hence, better bandwidth utilization is possible. It is easy to implement priority scheduling schemes with token scheduling and gateway code scheduling. The main disadvantages of CGSR are increase in path length and instability in the system at high mobility when the rate of change of cluster-heads

is high. In order to avoid gateway conflicts, more resources (such as additional interfaces) are required. The power consumption at the cluster-head node is also a matter of concern because the battery-draining rate at the cluster-head is higher than that at a normal node. This could lead to frequent changes in the cluster-head, which may result in multiple path breaks.

7.4.4 Source-Tree Adaptive Routing Protocol

Source-tree adaptive routing protocol (STAR) [9] proposed by Garcia-Luna-Aceves and Spohn is a variation of table-driven routing protocols, with the least overhead routing approach (LORA) as the key concept rather than the optimum routing approach (ORA) that was employed by earlier table-driven routing protocols. The ORA protocols attempt to update routing information quickly enough to provide optimum paths with respect to the defined metric (which may be the lowest number of hops), but with LORA, the routing protocol attempts to provide feasible paths that are not guaranteed to be optimal, but involve much less control overhead. In STAR protocol, every node broadcasts its *source-tree* information. The source-tree of a node consists of the wireless links used by the node in its preferred path to destinations. Every node, using its adjacent links and the source-tree broadcast by its neighbors, builds a partial graph of the topology. During initialization, a node sends an update message to its neighbors. Also, every node is required to originate update messages about new destinations, the chances of routing loops, and the cost of paths exceeding a given threshold. Hence, each node will have a path to every destination node. The path, in most cases, would be sub-optimal.

In the absence of a reliable link layer broadcast mechanism, STAR uses the following path-finding approach. When a node s has data packets to send to a particular destination d, for which no path exists in its source-tree, it originates an update message to all its neighbors indicating the absence of a path to d. This update message triggers another update message from a neighbor which has a path to d. Node s retransmits the update message as long as it does not have a path to d with increasing intervals between successive retransmissions. After getting the source-tree update from a neighbor, the node s updates its source-tree and, using this, it finds a path to all nodes in the network. The data packet contains information about the path to be traversed in order to prevent the possibility of routing loop formation.

In the presence of a reliable broadcast mechanism, STAR assumes implicit route maintenance. The link update message about the unavailability of a next-hop node triggers an update message from a neighbor which has an alternate source tree indicating an alternate next-hop node to the destination. In addition to path breaks, the intermediate nodes are responsible for handling the routing loops. When an intermediate node k receives a data packet to destination d, and one of the nodes in the packet's traversed path is present in node k's path to the destination d, then it discards the packet and a *RouteRepair* update message is reliably sent to the node in the head of the route repair path. The route repair path corresponds to the path k to x, where x is the last router in the data packet's traversed path that is first

found in the path k to d, that belongs to the source tree of k. The *RouteRepair* packet contains the complete source tree of node k and the traversed path of the packet.

When an intermediate node receives a *RouteRepair* update message, it removes itself from the top of the route repair path and reliably sends it to the head of the route repair path.

Advantages and Disadvantages

STAR has very low communication overhead among all the table-driven routing protocols. The use of the LORA approach in this table-driven routing protocol reduces the average control overhead compared to several other on-demand routing protocols.

7.5 ON-DEMAND ROUTING PROTOCOLS

Unlike the table-driven routing protocols, on-demand routing protocols execute the path-finding process and exchange routing information only when a path is required by a node to communicate with a destination. This section explores some of the existing on-demand routing protocols in detail.

7.5.1 Dynamic Source Routing Protocol

Dynamic source routing protocol (DSR) [10] is an on-demand protocol designed to restrict the bandwidth consumed by control packets in ad hoc wireless networks by eliminating the periodic table-update messages required in the table-driven approach. The major difference between this and the other on-demand routing protocols is that it is *beacon-less* and hence does not require periodic *hello* packet (*beacon*) transmissions, which are used by a node to inform its neighbors of its presence. The basic approach of this protocol (and all other on-demand routing protocols) during the route construction phase is to establish a route by flooding *RouteRequest* packets in the network. The destination node, on receiving a *RouteRequest* packet, responds by sending a *RouteReply* packet back to the source, which carries the route traversed by the *RouteRequest* packet received.

Consider a source node that does not have a route to the destination. When it has data packets to be sent to that destination, it initiates a *RouteRequest* packet. This *RouteRequest* is flooded throughout the network. Each node, upon receiving a *RouteRequest* packet, rebroadcasts the packet to its neighbors if it has not forwarded already or if the node is not the destination node, provided the packet's time to live (TTL) counter has not exceeded. Each *RouteRequest* carries a sequence number generated by the source node and the path it has traversed. A node, upon receiving a *RouteRequest* packet, checks the sequence number on the packet before forwarding it. The packet is forwarded only if it is not a duplicate *RouteRequest*. The sequence number on the packet is used to prevent loop formations and to avoid multiple transmissions of the same *RouteRequest* by an intermediate node that receives it through multiple paths. Thus, all nodes except the destination forward

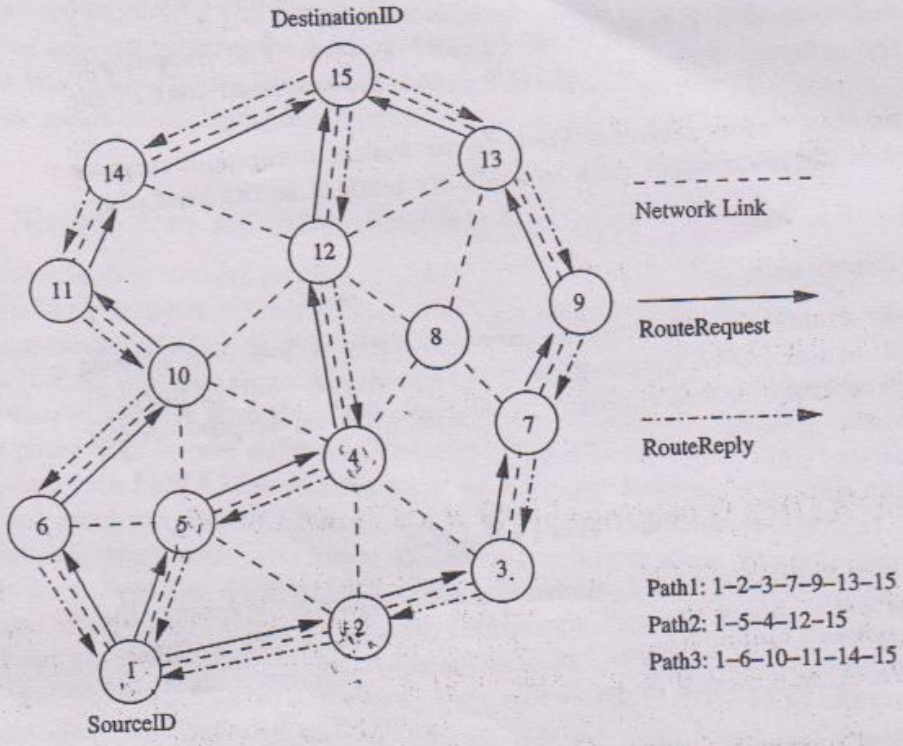


Figure 7.10. Route establishment in DSR.

a *RouteRequest* packet during the route construction phase. A destination node, after receiving the first *RouteRequest* packet, replies to the source node through the reverse path the *RouteRequest* packet had traversed. In Figure 7.10, source node 1 initiates a *RouteRequest* packet to obtain a path for destination node 15. This protocol uses a route cache that stores all possible information extracted from the source route contained in a data packet. Nodes can also learn about the neighboring routes traversed by data packets if operated in the promiscuous mode (the mode of operation in which a node can receive the packets that are neither broadcast nor addressed to itself). This route cache is also used during the route construction phase. If an intermediate node receiving a *RouteRequest* has a route to the destination node in its route cache, then it replies to the source node by sending a *RouteReply* with the entire route information from the source node to the destination node.

Optimizations

Several optimization techniques have been incorporated into the basic DSR protocol to improve the performance of the protocol. DSR uses the route cache at intermediate nodes. The route cache is populated with routes that can be extracted from the information contained in data packets that get forwarded. This cache information is used by the intermediate nodes to reply to the source when they receive a *RouteRequest* packet and if they have a route to the corresponding destination.

By operating in the promiscuous mode, an intermediate node learns about route breaks. Information thus gained is used to update the route cache so that the active routes maintained in the route cache do not use such broken links. During network partitions, the affected nodes initiate *RouteRequest* packets. An exponential backoff algorithm is used to avoid frequent *RouteRequest* flooding in the network when the destination is in another disjoint set. DSR also allows piggy-backing of a data packet on the *RouteRequest* so that a data packet can be sent along with the *RouteRequest*.

If optimization is not allowed in the DSR protocol, the route construction phase is very simple. All the intermediate nodes flood the *RouteRequest* packet if it is not redundant. For example, after receiving the *RouteRequest* packet from node 1 (refer to Figure 7.10), all its neighboring nodes, that is, nodes 2, 5, and 6, forward it. Node 4 receives the *RouteRequest* from both nodes 2 and 5. Node 4 forwards the first *RouteRequest* it receives from any one of the nodes 2 and 5 and discards the other redundant/duplicate *RouteRequest* packets. The *RouteRequest* is propagated till it reaches the destination which initiates the *RouteReply*. As part of optimizations, if the intermediate nodes are also allowed to originate *RouteReply* packets, then a source node may receive multiple replies from intermediate nodes. For example, in Figure 7.11, if the intermediate node 10 has a route to the destination via node 14, it also sends the *RouteReply* to the source node. The source node selects the latest

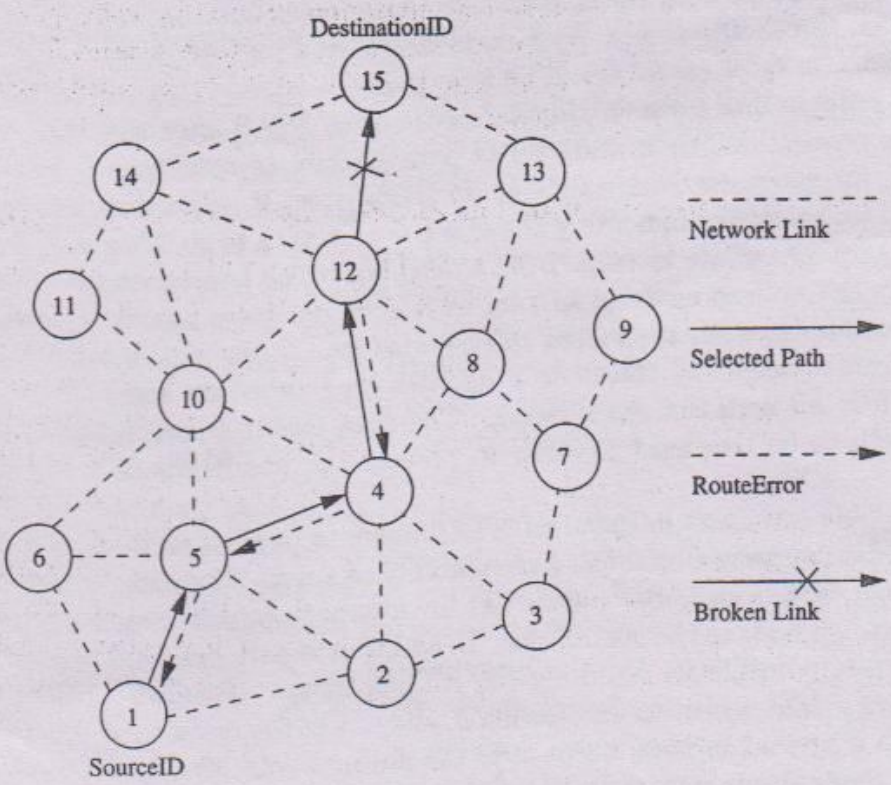


Figure 7.11. Route maintenance in DSR.

and best route, and uses that for sending data packets. Each data packet carries the complete path to its destination.

When an intermediate node in the path moves away, causing a wireless link to break, for example, the link between nodes 12 and 15 in Figure 7.11, a *RouteError* message is generated from the node adjacent to the broken link to inform the source node. The source node reinitiates the route establishment procedure. The cached entries at the intermediate nodes and the source node are removed when a *RouteError* packet is received. If a link breaks due to the movement of edge nodes (nodes 1 and 15), the source node again initiates the route discovery process.

Advantages and Disadvantages

This protocol uses a reactive approach which eliminates the need to periodically flood the network with table update messages which are required in a table-driven approach. In a reactive (on-demand) approach such as this, a route is established only when it is required and hence the need to find routes to all other nodes in the network as required by the table-driven approach is eliminated. The intermediate nodes also utilize the route cache information efficiently to reduce the control overhead. The disadvantage of this protocol is that the route maintenance mechanism does not locally repair a broken link. Stale route cache information could also result in inconsistencies during the route reconstruction phase. The connection setup delay is higher than in table-driven protocols. Even though the protocol performs well in static and low-mobility environments, the performance degrades rapidly with increasing mobility. Also, considerable routing overhead is involved due to the source-routing mechanism employed in DSR. This routing overhead is directly proportional to the path length.

7.5.2 Ad Hoc On-Demand Distance-Vector Routing Protocol

Ad hoc on-demand distance vector (AODV) [11] routing protocol uses an on-demand approach for finding routes, that is, a route is established only when it is required by a source node for transmitting data packets. It employs destination sequence numbers to identify the most recent path. The major difference between AODV and DSR stems out from the fact that DSR uses source routing in which a data packet carries the complete path to be traversed. However, in AODV, the source node and the intermediate nodes store the next-hop information corresponding to each flow for data packet transmission. In an on-demand routing protocol, the source node floods the *RouteRequest* packet in the network when a route is not available for the desired destination. It may obtain multiple routes to different destinations from a single *RouteRequest*. The major difference between AODV and other on-demand routing protocols is that it uses a destination sequence number (DestSeqNum) to determine an up-to-date path to the destination. A node updates its path information only if the DestSeqNum of the current packet received is greater than the last DestSeqNum stored at the node.

A *RouteRequest* carries the source identifier (SrcID), the destination identifier (DestID), the source sequence number (SrcSeqNum), the destination sequence num-

ber (DestSeqNum), the broadcast identifier (BcastID), and the time to live (TTL) field. DestSeqNum indicates the freshness of the route that is accepted by the source. When an intermediate node receives a *RouteRequest*, it either forwards it or prepares a *RouteReply* if it has a valid route to the destination. The validity of a route at the intermediate node is determined by comparing the sequence number at the intermediate node with the destination sequence number in the *RouteRequest* packet. If a *RouteRequest* is received multiple times, which is indicated by the BcastID-SrcID pair, the duplicate copies are discarded. All intermediate nodes having valid routes to the destination, or the destination node itself, are allowed to send *RouteReply* packets to the source. Every intermediate node, while forwarding a *RouteRequest*, enters the previous node address and its BcastID. A timer is used to delete this entry in case a *RouteReply* is not received before the timer expires. This helps in storing an active path at the intermediate node as AODV does not employ source routing of data packets. When a node receives a *RouteReply* packet, information about the previous node from which the packet was received is also stored in order to forward the data packet to this next node as the next hop toward the destination.

Consider the example depicted in Figure 7.12. In this figure, source node 1 initiates a path-finding process by originating a *RouteRequest* to be flooded in the network for destination node 15. assuming that the *RouteRequest* contains the des-

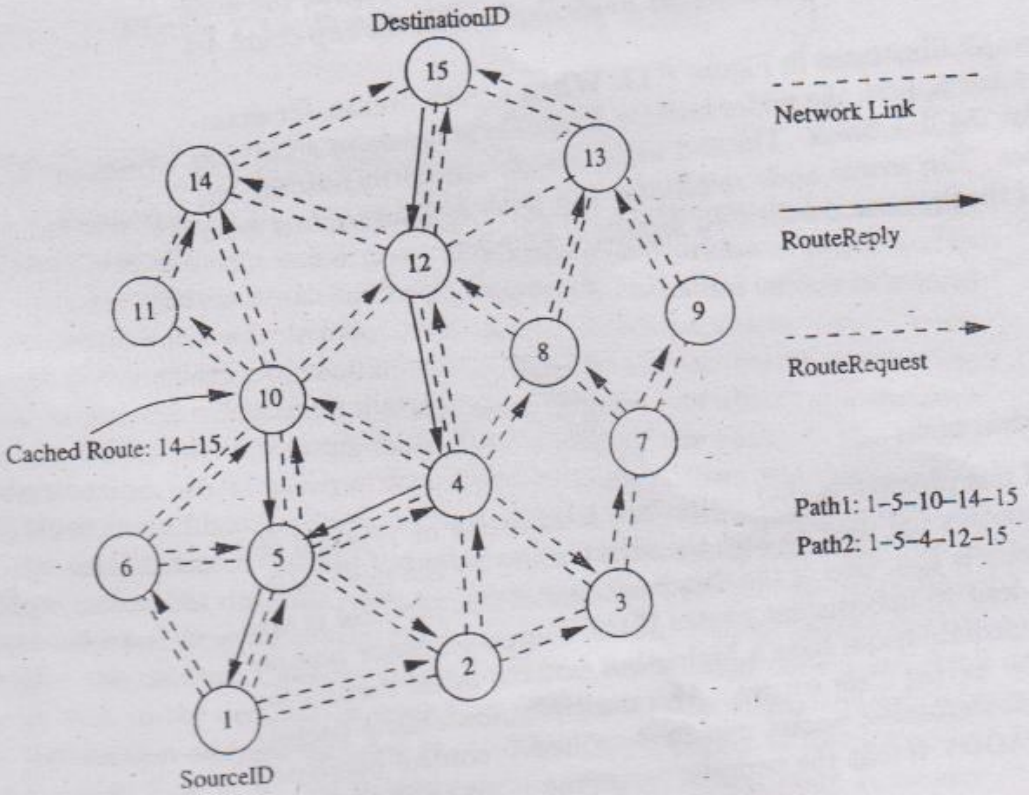


Figure 7.12. Route establishment in AODV.

tination sequence number as 3 and the source sequence number as 1. When nodes 2, 5, and 6 receive the *RouteRequest* packet, they check their routes to the destination. In case a route to the destination is not available, they further forward it to their neighbors. Here nodes 3, 4, and 10 are the neighbors of nodes 2, 5, and 6. This is with the assumption that intermediate nodes 3 and 10 already have routes to the destination node, that is, node 15 through paths 10-14-15 and 3-7-9-13-15, respectively. If the destination sequence number at intermediate node 10 is 4 and is 1 at intermediate node 3, then only node 10 is allowed to reply along the cached route to the source. This is because node 3 has an older route to node 15 compared to the route available at the source node (the destination sequence number at node 3 is 1, but the destination sequence number is 3 at the source node), while node 10 has a more recent route (the destination sequence number is 4) to the destination. If the *RouteRequest* reaches the destination (node 15) through path 4-12-15 or any other alternative route, the destination also sends a *RouteReply* to the source. In this case, multiple *RouteReply* packets reach the source. All the intermediate nodes receiving a *RouteReply* update their route tables with the latest destination sequence number. They also update the routing information if it leads to a shorter path between source and destination.

AODV does not repair a broken path locally. When a link breaks, which is determined by observing the periodical *beacons* or through link-level acknowledgments, the end nodes (*i.e.*, source and destination nodes) are notified. When a source node learns about the path break, it reestablishes the route to the destination if required by the higher layers. If a path break is detected at an intermediate node, the node informs the end nodes by sending an unsolicited *RouteReply* with the hop count set as ∞.

Consider the example illustrated in Figure 7.13. When a path breaks, for example, between nodes 4 and 5, both the nodes initiate *RouteError* messages to inform their end nodes about the link break. The end nodes delete the corresponding entries from their tables. The source node reinitiates the path-finding process with the new BcastID and the previous destination sequence number.

l
r
e
s
a
d
o
(D
its
tha
(De

Advantages and Disadvantages

The main advantage of this protocol is that routes are established on demand and destination sequence numbers are used to find the latest route to the destination. The connection setup delay is less. One of the disadvantages of this protocol is that intermediate nodes can lead to inconsistent routes if the source sequence number is very old and the intermediate nodes have a higher but not the latest destination sequence number, thereby having stale entries. Also multiple *RouteReply* packets in response to a single *RouteRequest* packet can lead to heavy control overhead. Another disadvantage of AODV is that the periodic *beaconing* leads to unnecessary bandwidth consumption.

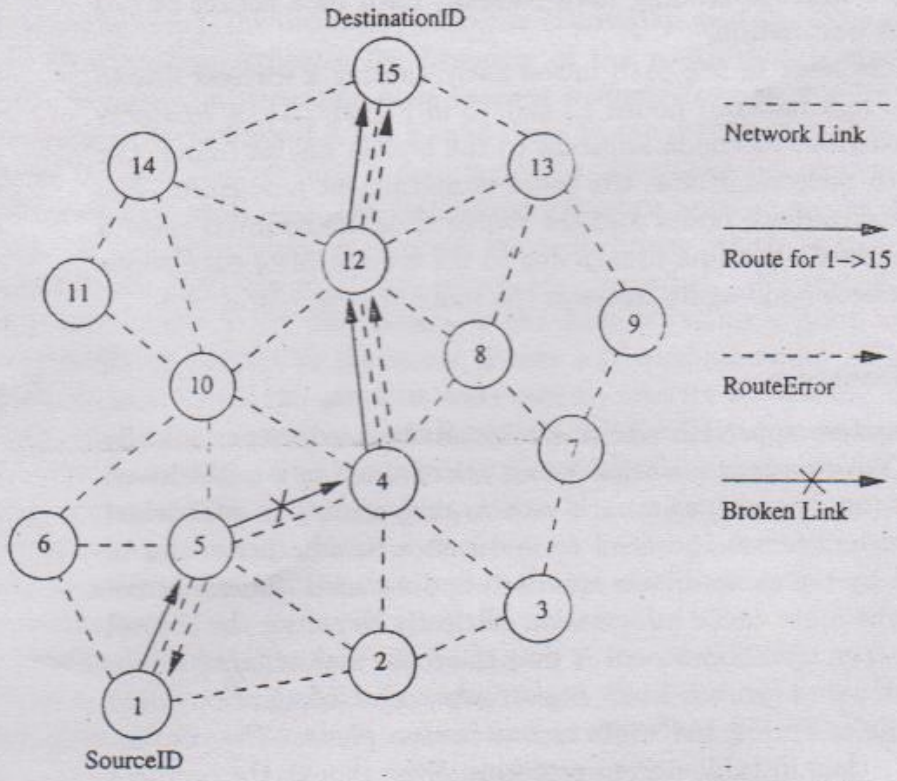


Figure 7.13. Route maintenance in AODV.

7.5.3 Temporally Ordered Routing Algorithm

Temporally ordered routing algorithm (TORA) [12] is a source-initiated on-demand routing protocol which uses a *link reversal algorithm* and provides loop-free multipath routes to a destination node. In TORA, each node maintains its one-hop local topology information and also has the capability to detect partitions. TORA has the unique property of limiting the control packets to a small region during the reconfiguration process initiated by a path break. Figure 7.14 shows the distance metric used in TORA which is nothing but the length of the path, or the height from the destination. H(N) denotes the height of node N from the destination. TORA has three main functions: establishing, maintaining, and erasing routes.

The route establishment function is performed only when a node requires a path to a destination but does not have any directed link. This process establishes a destination-oriented directed acyclic graph (DAG) using a *Query/Update* mechanism. Consider the network topology shown in Figure 7.14. When node 1 has data packets to be sent to the destination node 7, a *Query* packet is originated by node 1 with the destination address included in it. This *Query* packet is forwarded by intermediate nodes 2, 3. 4, 5, and 6, and reaches the destination node 7, or any other node which has a route to the destination. The node that terminates (in this case, node 7) the *Query* packet replies with an *Update* packet containing its distance from the destination (it is zero at the destination node). In the example.

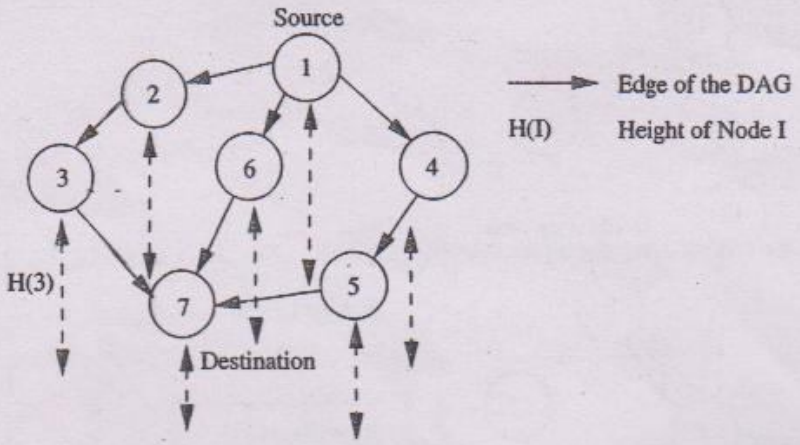


Figure 7.14. Illustration of temporal ordering in TORA.

the destination node 7 originates an *Update* packet. Each node that receives the *Update* packet sets its distance to a value higher than the distance of the sender of the *Update* packet. By doing this, a set of directed links from the node which originated the *Query* to the destination node 7 is created. This forms the DAG depicted in Figure 7.14. Once a path to the destination is obtained, it is considered to exist as long as the path is available, irrespective of the path length changes due to the reconfigurations that may take place during the course of the data transfer session.

When an intermediate node (say, node 5) discovers that the route to the destination node is invalid, as illustrated in Figure 7.15, it changes its distance value

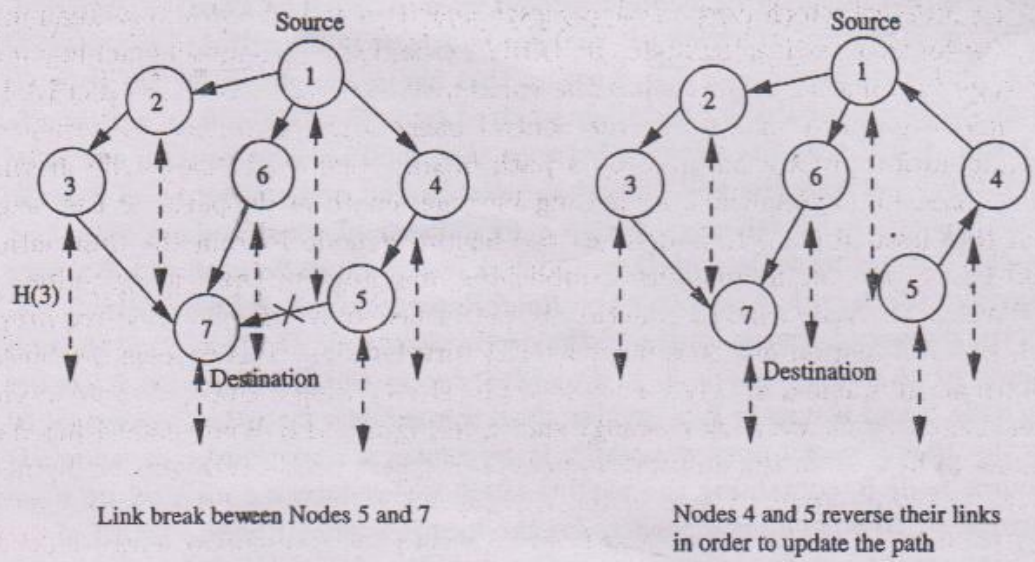


Figure 7.15. Illustration of route maintenance in TORA.

to a higher value than its neighbors and originates an *Update* packet. The neighboring node 4 that receives the *Update* packet reverses the link between 1 and 4 and forwards the *Update* packet. This is done to update the DAG corresponding to destination node 7. This results in a change in the DAG. If the source node has no other neighbor that has a path to the destination, it initiates a fresh *Query/Update* procedure. Assume that the link between nodes 1 and 4 breaks. Node 4 reverses the path between itself and node 5, and sends an update message to node 5. Since this conflicts with the earlier reversal, a partition in the network can be inferred. If the node detects a partition, it originates a *Clear* message, which erases the existing path information in that partition related to the destination.

Advantages and Disadvantages

By limiting the control packets for route reconfigurations to a small region, TORA incurs less control overhead. Concurrent detection of partitions and subsequent deletion of routes could result in temporary oscillations and transient loops. The local reconfiguration of paths results in non-optimal routes.

7.5.4 Location-Aided Routing

Location-aided routing protocol (LAR) [13] utilizes the location information for improving the efficiency of routing by reducing the control overhead. LAR assumes the availability of the global positioning system (GPS) for obtaining the geographical position information necessary for routing. LAR designates two geographical regions for selective forwarding of control packets, namely, *ExpectedZone* and *RequestZone*. The *ExpectedZone* is the region in which the destination node is expected to be present, given information regarding its location in the past and its mobility information (refer to Figure 7.16). In the event of non-availability of past information about the destination, the entire network area is considered to be the *ExpectedZone* of the destination. Similarly, with the availability of more information about its mobility, the *ExpectedZone* of the destination can be determined with more accuracy and improved efficiency. The *RequestZone* is a geographical region within which the path-finding control packets are permitted to be propagated. This area is determined by the sender of a data transfer session. The control packets used for path-finding are forwarded by nodes which are present in the *RequestZone* and are discarded by nodes outside the zone. In situations where the sender or the intermediate relay nodes are not present in the *RequestZone*, additional area is included for forwarding the packets. This is done when the first attempt for obtaining a path to a destination using the initial *RequestZone* fails to yield a path within a sufficiently long waiting time. In this case, the second attempt repeats the process with increased *RequestZone* size to account for mobility and error in location estimation. LAR uses flooding, but here flooding is restricted to a small geographical region. The nodes decide to forward or discard the control packets based on two algorithms, namely, LAR1 and LAR2.

In the LAR1 algorithm, the source node (say, S) explicitly specifies the *RequestZone* in the *RouteRequest* packet. As per LAR1, as illustrated in Figure 7.16, the

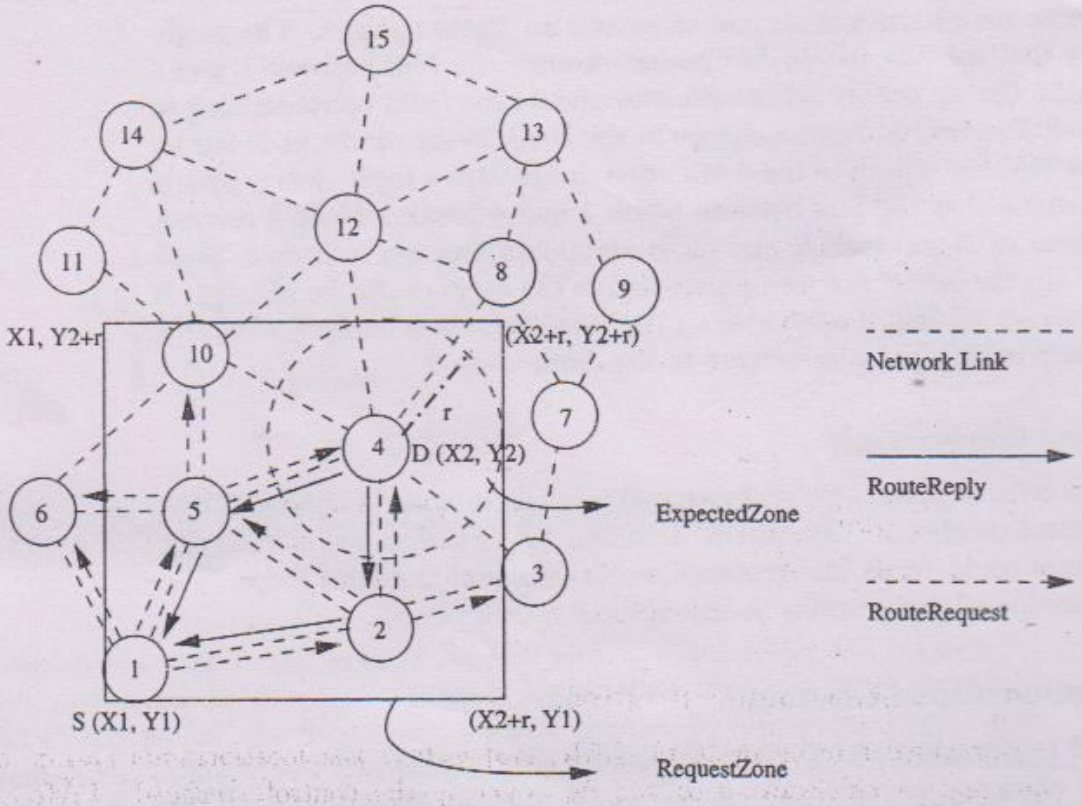


Figure 7.16. *RequestZone* and *ExpectedZone* in LAR1.

RequestZone is the smallest rectangle that includes the source node (S) and the *ExpectedZone*, the sides of which are parallel to the X and Y axes, when the node S is outside the *ExpectedZone*. When node S is within the *ExpectedZone*, then the *RequestZone* is reduced to the *ExpectedZone* itself. Every intermediate node that receives the *RouteRequest* packet verifies the *RequestZone* information contained in the packet and forwards it further if the node is within the *RequestZone*; otherwise, the packet is discarded. In Figure 7.16, the source node (node 1) originates a *RouteRequest*, which is broadcast to its neighbors (2, 5, and 6). These nodes verify their own geographical locations to check whether they belong to the *ExpectedZone*. Nodes 2 and 5 find that they are inside the *ExpectedZone* and hence they forward the *RouteRequest*. But node 6 discards the packet. Finally, when the *RouteRequest* reaches the destination node (node 4), it originates a *RouteReply* that contains the current location and current time of the node. Also, as an option, the current speed of movement can be included in the *RouteReply* if that information is available with the node. Such information included in the *RouteReply* packet is used by the source node for future route establishment procedures.

In LAR2 algorithm (Figure 7.17), the source node S (node 5) includes the distance between itself and the destination node D (node 8) along with the (X, Y) coordinates of the destination node D in the *RouteRequest* packet instead of the explicit information about the *Expected Region*. When an intermediate node re-

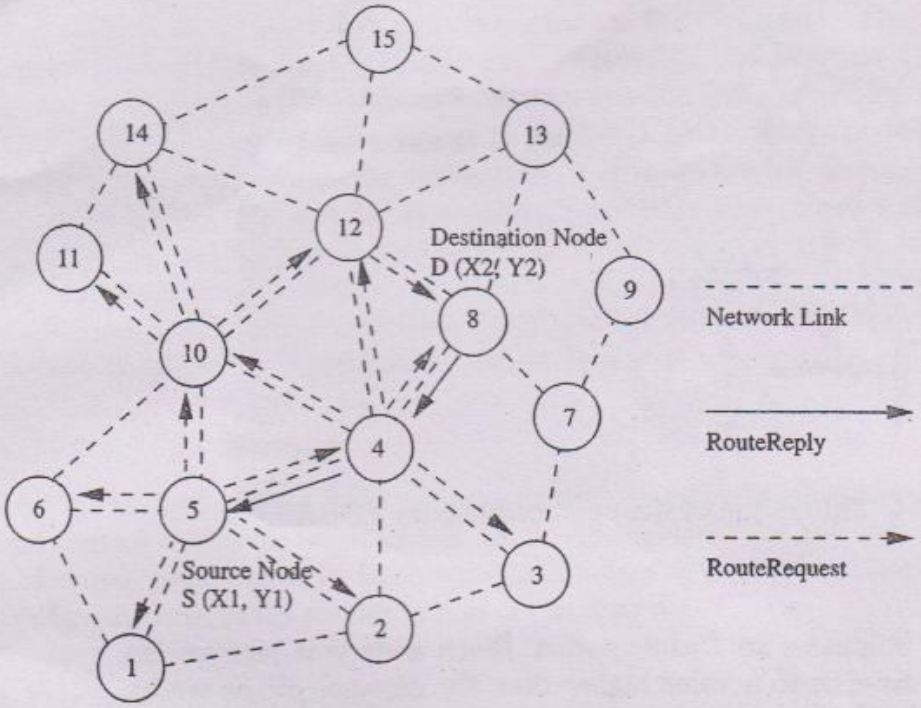


Figure 7.17. Route establishment in LAR2.

ceives this *RouteRequest* packet, it computes the distance to the node D. If this distance is less than the distance from S to node $D + \delta$, where δ is a parameter of the algorithm decided based on the error in location estimation and mobility, then the *RouteRequest* packet is forwarded. Otherwise, the *RouteRequest* is discarded. Consider the example illustrated in Figure 7.17. Assume that the value of δ is 0 here. The *RouteRequest* packet originated by node 5 is received by nodes 1, 2, 4, 6, and 10. Only nodes 4 and 10 find that the distance between them and the destination is less than the distance between the node 5 and the destination node; other nodes discard the *RouteRequest*. A *RouteRequest* packet is forwarded only once and the distance between the forwarding node and D is updated in the *RouteRequest* packet for further relaying. When node 4 forwards the *RouteRequest* packet, it updates the packet with the distance between itself and the destination node D. This packet, after being received at neighbor node 3, is discarded due to the fact that the distance between node 3 and the node 8 is greater than the distance between nodes 4 and 8. Once the *RouteRequest* reaches node 8, it originates a *RouteReply* packet back to the source node 5, containing the path through which future data packets are to be propagated. In order to compensate for the location error (due to the inaccuracy of GPS information or due to changes in the mobility of the nodes), a larger *RequestZone* that can accommodate the amount of error that occurred is considered.

Advantages and Disadvantages

LAR reduces the control overhead by limiting the search area for finding a path. The efficient use of geographical position information, reduced control overhead, and increased utilization of bandwidth are the major advantages of this protocol. The applicability of this protocol depends heavily on the availability of GPS infrastructure or similar sources of location information. Hence, this protocol cannot be used in situations where there is no access to such information.

7.5.5 Associativity-Based Routing

Associativity-based routing (ABR) [14] protocol is a distributed routing protocol that selects routes based on the stability of the wireless links. It is a *beacon*-based, on-demand routing protocol. A link is classified as stable or unstable based on its temporal stability. The temporal stability is determined by counting the periodic *beacons* that a node receives from its neighbors. Each node maintains the count of its neighbors' *beacons* and classifies each link as stable or unstable based on the *beacon* count corresponding to the neighbor node concerned. The link corresponding to a stable neighbor is termed as a stable link, while a link to an unstable neighbor is called an unstable link.

A source node floods *RouteRequest* packets throughout the network if a route is not available in its route cache. All intermediate nodes forward the *RouteRequest* packet. A *RouteRequest* packet carries the path it has traversed and the *beacon* count for the corresponding node in the path. When the first *RouteRequest* reaches the destination, the destination waits for a time period $T_{RouteSelectTime}$ to receive multiple *RouteRequests* through different paths. After this time duration, the destination selects the path that has the maximum proportion of stable links. If two paths have the same proportion of stable links, the shorter of them is selected. If more than one shortest path is available, then a random path among them is selected as the path between source and destination.

Consider Figure 7.18, in which the source node (node 1) initiates the *RouteRequest* to be flooded for finding a route to the destination node (node 15). The solid lines represent the *stable* links that are classified based on the *beacon* count, while dotted lines represent *unstable* links. ABR does not restrict any intermediate node from forwarding a *RouteRequest* packet based on the stable or unstable link criterion. It uses stability information only during the route selection process at the destination node. As depicted in Figure 7.18, the *RouteRequest* reaches the destination through three different routes. Route 1 is 1-5-10-14-15, route 2 is 1-5-4-12-15, and route 3 is 1-2-4-8-13-15. ABR selects route 3 as it contains the highest percentage of stable links compared to route 1 and route 2. ABR gives more priority to stable routes than to shorter routes. Hence, route 3 is selected even though the length of the selected route is more than that of the other two routes.

If a link break occurs at an intermediate node, the node closer to the source, which detects the break, initiates a local route repair process. In this process, the node locally broadcasts a route repair packet, termed the local query (LQ) broadcast, with a limited time to live (TTL), as illustrated in Figure 7.19 where a

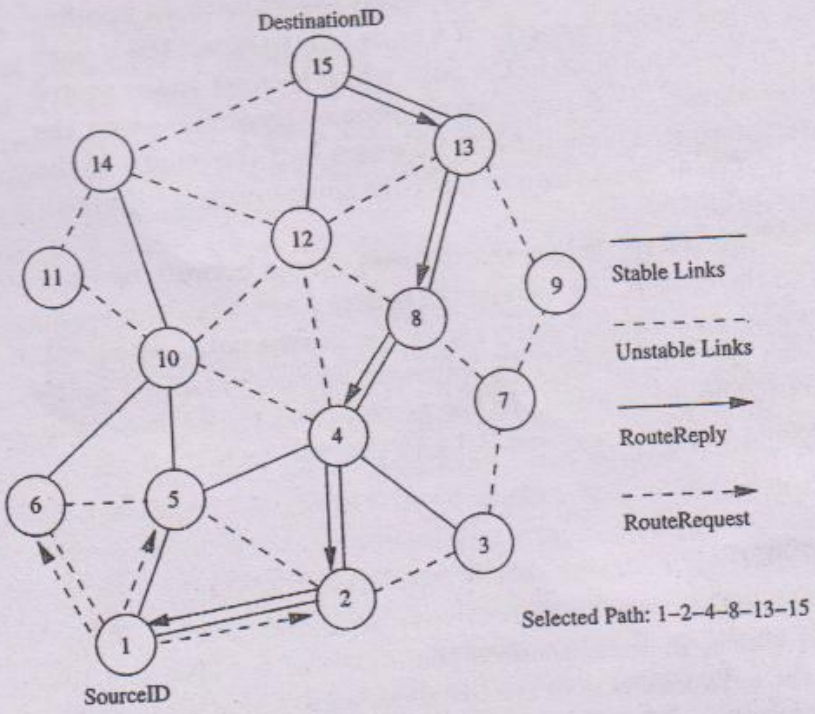


Figure 7.18. Route establishment in ABR.

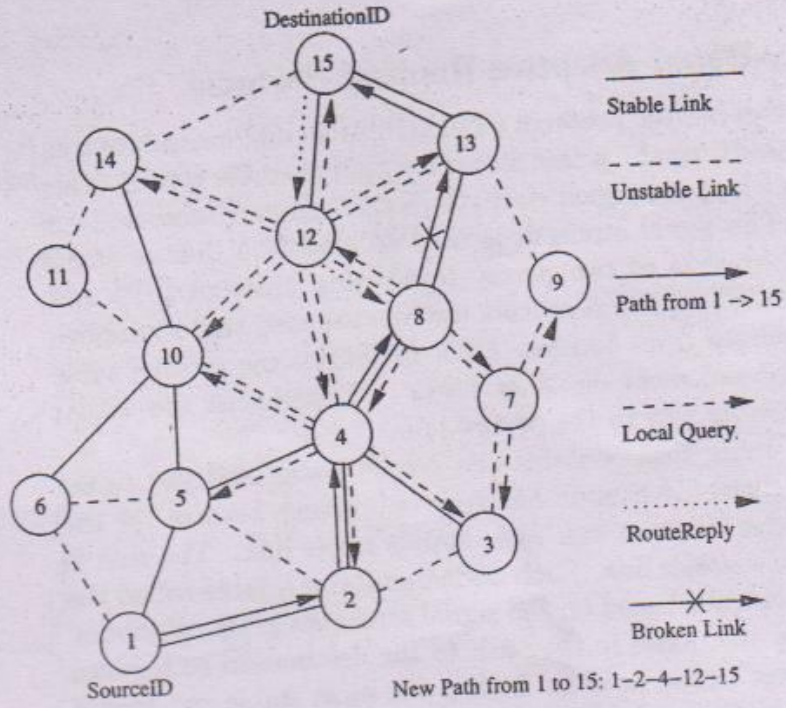


Figure 7.19. Route maintenance in ABR.

TTL value of 2 is used. This way a broken link is bypassed locally without flooding a new *RouteRequest* packet in the whole network. If a node fails to repair the broken link, then its uplink node (the previous node in the path which is closer to the source node) reinitiates the LQ broadcast. This route repair process continues along the intermediate nodes toward the source node until it traverses half the length of the broken path or the route is repaired. In the former case, the source node is informed, which initiates a new route establishment phase.

Consider the example in Figure 7.19. When a path break occurs between nodes 8 and 13, the node adjacent to the broken link toward the source node, that is, node 8, initiates a local query broadcast in order to locally repair the broken path. The local query has limited scope with the maximum TTL value set to the remaining path length from the broken link to the destination. In the same figure, the broken path is repaired locally by bypassing the path segment 8-13-15 through segment 8-12-15.

Advantages and Disadvantages

In ABR, stable routes have a higher preference compared to shorter routes. They result in fewer path breaks which, in turn, reduces the extent of flooding due to reconfiguration of paths in the network. One of the disadvantages of this protocol is that the chosen path may be longer than the shortest path between the source and destination because of the preference given to stable paths. Another disadvantage is that repetitive LQ broadcasts may result in high delays during route repairs.

7.5.6 Signal Stability-Based Adaptive Routing Protocol

Signal stability-based adaptive routing protocol (SSA) [15] is an on-demand routing protocol that uses signal stability as the prime factor for finding stable routes. This protocol is *beacon*-based, in which the signal strength of the *beacon* is measured for determining link stability. The signal strength is used to classify a link as *stable* or *unstable*. This protocol consists of two parts: forwarding protocol (FP) and dynamic routing protocol (DRP). These protocols use an extended radio interface that measures the signal strength from *beacons*. DRP maintains the routing table by interacting with the DRP processes on other hosts. FP performs the actual routing to forward a packet on its way to the destination.

Every node maintains a table that contains the *beacon* count and the signal strength of each of its neighbors. If a node has received strong *beacons* for the past few *beacons*, the node classifies the link as a *strong/stable* link. The link is otherwise classified as a *weak/unstable* link. Each node maintains a table called the signal stability table (SST), which is based on the signal strengths of its neighbors' *beacons*. This table is used by the nodes in the path to the destination to forward the incoming *RouteRequest* over strong links for finding the most stable end-to-end path. If the attempt of forwarding a *RouteRequest* over the stable links fails to obtain any path to a destination, the protocol floods the *RouteRequest* throughout the network without considering the stability of links as the forwarding criterion.

A source node which does not have a route to the destination floods the network with *RouteRequest* packets. But unlike other routing protocols, nodes that employ the SSA protocol process a *RouteRequest* only if it is received over a strong link. A *RouteRequest* received through a weak link is dropped without being processed. The destination selects the first *RouteRequest* packet received over strong links. The destination initiates a *RouteReply* packet to notify the selected route to the source.

Consider Figure 7.20, where the source node (node 1) broadcasts a *RouteRequest* for finding the route to the destination node (node 15). In Figure 7.20, solid lines represent the *stable* links, while the dotted lines represent *weak* links. Unlike ABR, SSA restricts intermediate nodes from forwarding a *RouteRequest* packet if the packet had been received over a weak link. It forwards only *RouteRequest* packets received over stable links. In Figure 7.20, when the *RouteRequest* is initiated by the source, it is to be processed by all its neighbors. But before processing, each neighbor node checks whether the *RouteRequest* packet was received through a stable link. If the *RouteRequest* had been received through a stable link and had not been sent already (*i.e.*, it is not a duplicate *RouteRequest*), it is forwarded by the node; otherwise, it is dropped. For example, when the *RouteRequest* from node 1 reaches nodes 2, 5, and 6, it is forwarded only by nodes 2 and 5 as the link between nodes 1 and 6 is weak. Similarly, the *RouteRequest* forwarded by node 2 is rejected by nodes 3 and 5, while node 4 forwards it to its neighbors, provided it

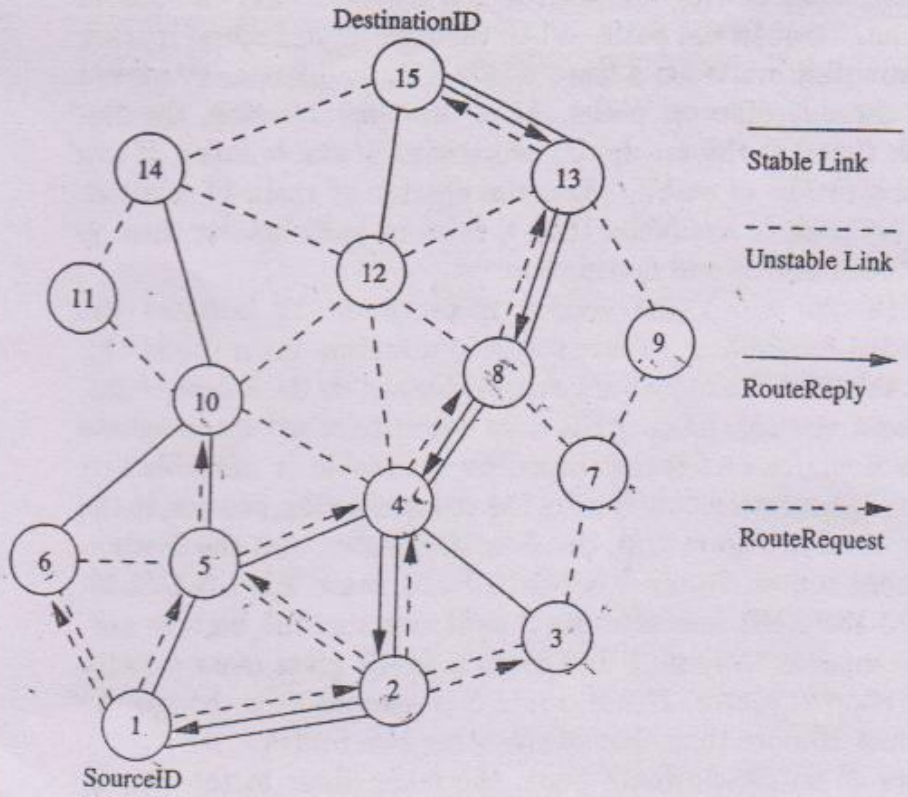


Figure 7.20. Route establishment in SSA.

has not already been forwarded. In this way, nodes forward the *RouteRequest* until it reaches the destination. In this example, only one route is established based on the strong link criterion. A *RouteRequest* propagated through path 1-5-10-14-15 is rejected by the destination as it receives a *RouteRequest* from node 14 on a weak link. The stable path consisting of strong links is 1- 2 (or 5)-4-8-13-15. The first *RouteRequest* packet that reaches the destination over a stable path is selected by the destination.

As shown in Figure 7.21, when a link breaks, the end nodes of the broken link (*i.e.*, nodes 2 and 4) notify the corresponding end nodes of the path (*i.e.*, nodes 1 and 15). A source node, after receiving a route break notification packet, rebroadcasts the *RouteRequest* to find another stable path to the destination. Stale entries are removed only if data packets that use the stale route information fail to reach the next node. If the link between nodes 2 and 4 breaks, a new strong path is established through 1-5-4-8-13-15. If no strong path is available when a link gets broken (*e.g.*, link 8-13), then the new route is established by considering weak links also. This is done when multiple *RouteRequest* attempts fail to obtain a path to the destination using only the stable links.

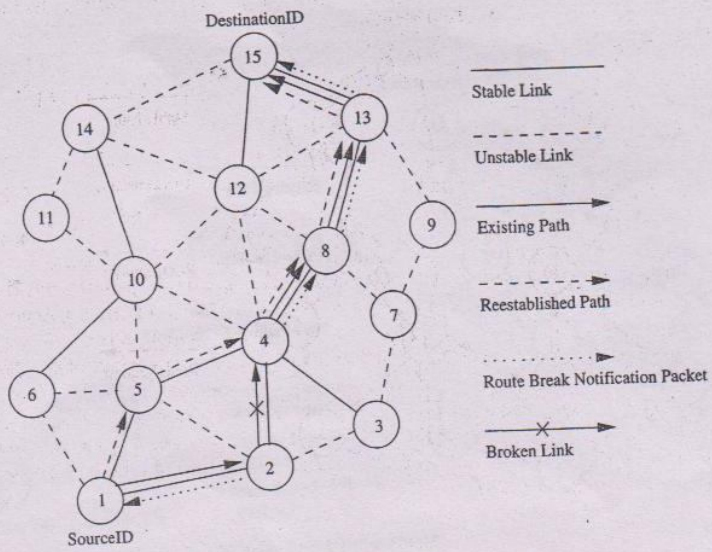


Figure 7.21. Route maintenance in SSA.

Advantages and Disadvantages

The main advantage of this protocol is that it finds more stable routes when compared to the shortest path route selection protocols such as DSR and AODV. This protocol accommodates temporal stability by using *beacon* counts to classify a link as stable or weak. The main disadvantage of this protocol is that it puts a strong *RouteRequest* forwarding condition which results in *RouteRequest* failures. A failed *RouteRequest* attempt initiates a similar path-finding process for a new path without considering the stability criterion. Such multiple flooding of *RouteRequest* packets consumes a significant amount of bandwidth in the already bandwidth-constrained network, and also increases the path setup time. Another disadvantage is that the strong links criterion increases the path length, as shorter paths may be ignored for more stable paths.

7.5.7 Flow-Oriented Routing Protocol

Flow-oriented routing protocol (FORP) [16] is an on-demand routing protocol that employs a prediction-based *multi-hop-handoff* mechanism for supporting time-sensitive traffic in ad hoc wireless networks. This protocol has been proposed for IPv6-based ad hoc wireless networks where quality of service (QoS) needs to be provided. The *multi-hop-handoff* is aimed at alleviating the effects of path breaks on the real-time packet flows. A sender or an intermediate node initiates the route maintenance process only after detecting a link break. This reactive route maintenance procedure may result in high packet loss, leading to a low quality of service provided to the user. FORP uses a unique prediction-based mechanism that utilizes the mobility and location information of nodes to estimate the link expiration time (LET). LET is the approximate lifetime of a given wireless link. The minimum of the LET values of all wireless links on a path is termed as the route expiry time (RET). Every node is assumed to be able to predict the LET of each of its links with its neighbors. The LET between two nodes can be estimated using information such as current position of the nodes, their direction of movement, and their transmission ranges. FORP requires the availability of GPS information in order to identify the location of nodes.

When a sender node needs to set up a real-time flow to a particular destination, it checks its routing table for the availability of a route to that destination. If a route is available, then that is used to send packets to the destination. Otherwise, the sender broadcasts a *Flow-REQ* packet carrying information regarding the source and the destination nodes. The *Flow-REQ* packet also carries a flow identification number/sequence number which is unique for every session. A neighbor node, on receiving this packet, first checks if the sequence number of the received *Flow-REQ* is higher than the sequence number corresponding to a packet belonging to the same session that had been previously forwarded by the node. If so, then it updates its address on the packet and extracts the necessary state information out of the packet. If the sequence number on the packet is less than that of the previously forwarded packet, then the packet is discarded. This is done to avoid looping of *Flow-REQ* packets. A *Flow-REQ* with the same sequence number as that of a *Flow-*

REQ belonging to the same session which had been forwarded already by the node, would be broadcast further only if it has arrived through a shorter (and therefore better) path. Before forwarding a *Flow-REQ*, the intermediate node appends its node address and the LET of the last link the packet had traversed onto the packet. The *Flow-REQ* packet, when received at the destination node, contains the list of nodes on the path it had traversed, along with the LET values of every wireless link on that path. FORP assumes all the nodes in the network to be synchronized to a common time by means of GPS information. If the calculated value of RET, corresponding to the new *Flow-REQ* packet arrived at the destination, is better than the RET value of the path currently being used, then the destination originates a *Flow-SETUP* packet. The LET of a link can be estimated given the information about the location, velocity, and transmission range of the nodes concerned. The LET of the wireless link between two nodes a and b with transmission range T_x, which are moving at velocity V_a and V_b at angles T_a and T_b, respectively (refer to Figure 7.22 (a)), can be estimated as described below:

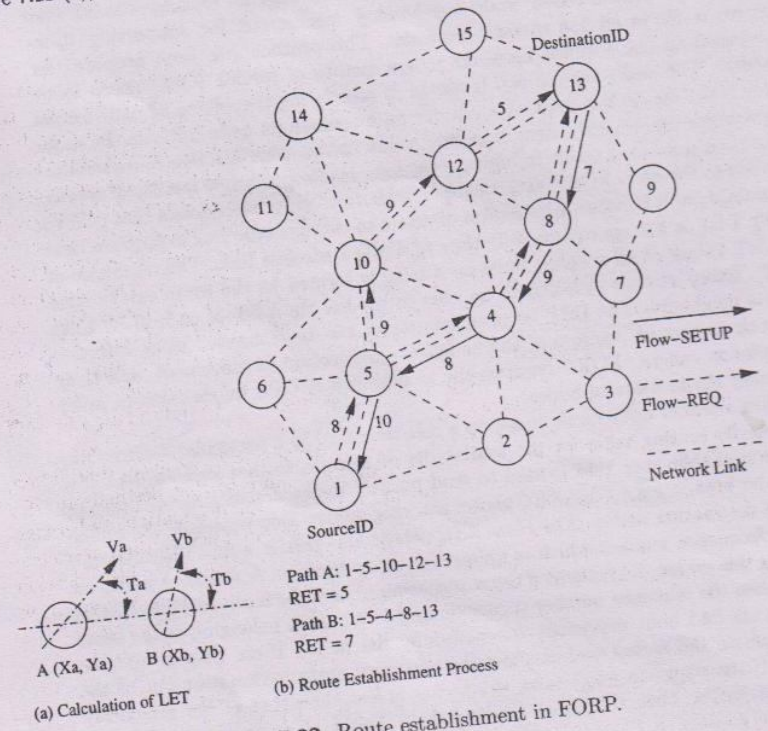


Figure 7.22. Route establishment in FORP.

$$LET_{ab} = \frac{-(pq+rs) + (p^2+r^2){T_x}^2 - (ps-qr)^2}{p^2+q^2} \tag{7.5.1}$$

where

$$p = V_a \cos T_a - V_b \cos T_b \tag{7.5.2}$$
$$q = X_a - X_b \tag{7.5.3}$$
$$r = V_a \sin T_a - V_b \sin T_b \tag{7.5.4}$$
$$s = Y_a - Y_b \tag{7.5.5}$$

The route establishment procedure is shown in Figure 7.22 (b). In this case, the path 1-5-4-8-13 (path 1) has a RET value of 7, whereas the path 1-5-10-12-13 (path 2) has a RET value of 5. This indicates that path 1 may last longer than path 2. Hence the sender node originates a *Flow-SETUP* through the reverse path 13-8-4-5-1.

FORP employs a proactive route maintenance mechanism which makes use of the expected RET of the current path available at the destination. Route maintenance is illustrated in Figure 7.23. When the destination node determines (using the RET of the current path) that a route break is about to occur within a critical time period (t_c), it originates a *Flow-HANDOFF* packet to the source node, which is forwarded by the intermediate nodes. The mechanism by which *Flow-HANDOFF*

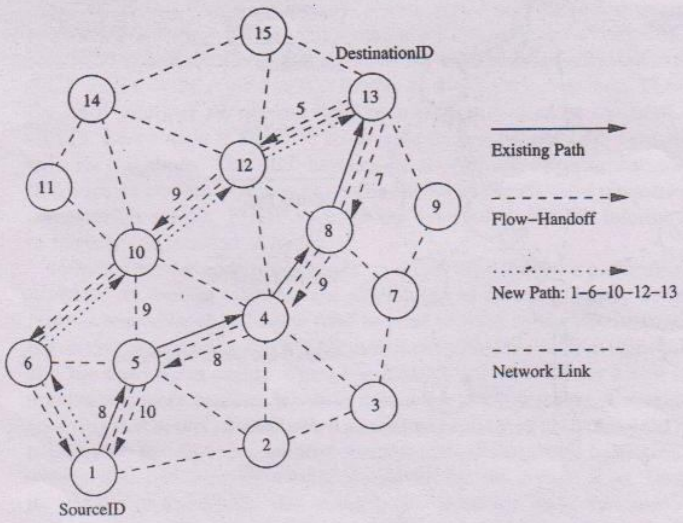


Figure 7.23. Route maintenance in FORP.

packets are forwarded is similar to the *Flow-REQ* forwarding mechanism. When many *Flow-HANDOFF* packets arrive at the source node, the source node calculates the RET values of paths taken by each of them, selects the best path, and uses this new path for sending packets to the destination. In the example shown in Figure 7.23, the *Flow-HANDOFF* packets are forwarded by every intermediate node after appending the LET information of the previous link traversed onto the packet. The existing path 1-5-4-8-13 is erased and a new path is selected by the source node based on the RETs corresponding to different paths traversed by the *Flow-HANDOFF* packets. In this case, the path 1-6-10-12-13 is chosen. The critical time (t_c) is taken as the difference between the RET and delay encountered by the latest packet which has traversed through the existing path from the source to the destination.

Advantages and Disadvantages

The use of LET and RET estimates reduces path breaks and their associated ill effects such as reduction in packet delivery, increase in the number of out-of-order packets, and non-optimal paths resulting from local reconfiguration attempts. The proactive route reconfiguration mechanism adopted here works well when the topology is highly dynamic. The requirements of time synchronization increases the control overhead. Dependency on the GPS infrastructure affects the operability of this protocol in environments where such infrastructure may not be available.

7.6 HYBRID ROUTING PROTOCOLS

In this section, we discuss the working of routing protocols termed as hybrid routing protocols. Here, each node maintains the network topology information up to m hops. The different existing hybrid protocols are presented below.

7.6.1 Core Extraction Distributed Ad Hoc Routing Protocol

Core extraction distributed ad hoc routing (CEDAR) [17] integrates routing and support for QoS. It is based on extracting core nodes (also called as dominator nodes) in the network, which together approximate the minimum dominating set. A dominating set (DS) of a graph is defined as a set of nodes in the graph such that every node in the graph is either present in the DS or is a neighbor of some node present in the DS. There exists at least one core node within three hops. The DS of the least cardinality in a graph is called the minimum dominating set. Nodes that choose a core node as their dominating node are called core member nodes of the core node concerned. The path between two core nodes is termed a virtual link. CEDAR employs a distributed algorithm to select core nodes. The selection of core nodes represents the core extraction phase.

CEDAR uses the core broadcast mechanism to transmit any packet throughout the network in the unicast mode, involving as minimum number of nodes as possible. These nodes that take part in the core broadcast process are called core nodes. In order to carry out a core broadcast efficiently, each core node must know about its

neighboring core nodes. The transfer of information about neighboring core nodes results in significant control overhead at high mobility. When a core node to which many nodes are attached moves away from them, each node has to reselect a new core node. The selection of core nodes, which is similar to the distributed leader election process, involves substantial control overhead.

Each core node maintains its neighborhood local topology information. CEDAR employs an efficient link state propagation mechanism in which information regarding the presence of high bandwidth and stable links is propagated to several more nodes, compared to the propagation of information regarding low bandwidth and unstable links, which is suppressed and kept local. To propagate link information, slow-moving increase-waves and fast-moving decrease-waves are used. An increase-wave carrying update information is originated when the bandwidth on the link concerned increases above a certain threshold. The fast-moving decrease-waves are propagated in order to quickly notify the nearby nodes (core nodes which are at most separated by three hops) about the reduction in available bandwidth. As bandwidth increase information moves slowly, only stable high-bandwidth link state information traverses long distances. If the high-bandwidth link is unstable, then the corresponding increase-wave is overtaken by fast-moving decrease-waves which represent the decrease in available bandwidth on the corresponding link. These waves are very adaptive to the dynamic topology of ad hoc wireless networks. Increase- and decrease-waves are initiated only when changes in link capacity cross certain thresholds, that is, only when there is a significant change in link capacity. Fast-moving decrease-waves are prevented from moving across the entire network, thereby suppressing low bandwidth unstable information to the local nodes only. Route establishment in CEDAR is carried out in two phases. The first phase finds a core path from the source to the destination. The core path is defined as a path from the dominator of the source node (source core) to the dominator of the destination node (destination core). In the second phase, a QoS feasible path is found over the core path. A node initiates a *RouteRequest* if the destination is not in the local topology table of its core node; otherwise, the path is immediately established. For establishing a route, the source core initiates a core broadcast in which the *RouteRequest* is sent to all neighboring core nodes as unicast data. Each of these recipient core nodes in turn forwards the *RouteRequest* to its neighboring core nodes if the destination is not its core member. A core node which has the destination node as its core member replies to the source core. Once the core path is established, a path with the required QoS support is then chosen.

To find a path that can provide the required QoS, the source core first finds a path to the domain of the farthest possible core node in the core path, which can provide the bandwidth required. Among the available paths to this domain, the source core chooses the shortest-widest path (shortest path with highest bandwidth). Assume *MidCore* is the farthest possible core node found by the source core. In the next iteration, *MidCore* becomes the new source core and finds another *MidCore* node that satisfies the QoS support requirements. This iterative process repeats until a path to the destination with the required bandwidth is found. If no

feasible path is available, the source node is informed about the non-availability of a QoS path.

Consider Figure 7.24 where the source is node 1 and the destination is node 15. The core nodes in the network are nodes 3, 5, 11, 12, and 13. In this figure, node 5 is the dominator of nodes 1 and 6. Similarly, node 12 is the dominator of node 15. When node 1 initiates a *RouteRequest* to be flooded throughout the network, it intimates its core node the <source id, destination id> pair information. If the core node 5 does not have any information about the dominator of node 15, which is the destination node, it initiates a core broadcast. Due to this, all nearby core nodes receive the request in the unicast transmission mode. This unicast transmission is done on the virtual links. For core node 5, the virtual link with core node 3 comprises of the links 5-2 and 2-3, while the virtual link between core nodes 5 and 13 is represented by path 5-4-8-13. When a core node receives the core broadcast message, it checks whether the destination is its core member. A core node having the destination as one of its core members replies to the source core node. In our case, core node 12 replies to core node 5. The path between core nodes 12 and 5 constitutes a core path. Once a core path is established, the feasibility of the path in terms of the availability of the required bandwidth is checked. If the required

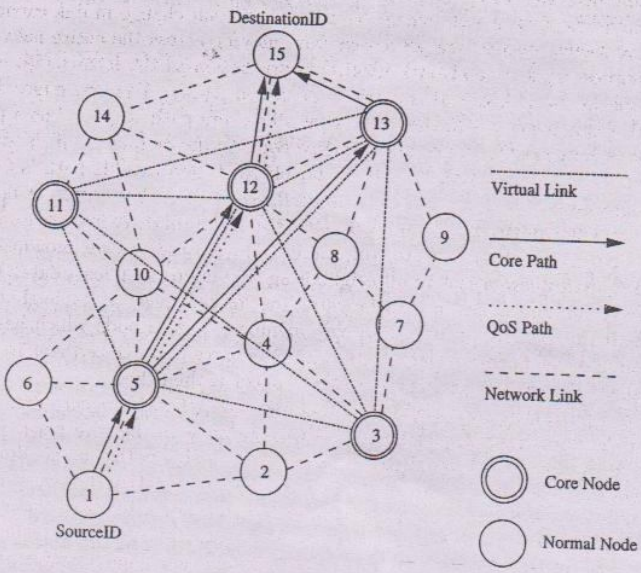


Figure 7.24. Route establishment in CEDAR.

bandwidth is available on the path, the connection is established; otherwise, the core path is rejected.

CEDAR attempts to repair a broken route locally when a path break occurs. When a link u-v on the path from source to the destination fails, node u sends back a notification to the source and starts recomputation of a route from itself to the destination node. Until the route recomputation gets completed, node u drops every subsequent packet it receives. Once the source node receives the notification sent by node u, it immediately stops transmitting packets belonging to the corresponding flow, and starts computing a new route to the destination. If the link break occurs near the source, route recomputation at node u may take a long time, but the notification sent by node u reaches the source node very rapidly and prevents large packet losses. If the broken link is very close to the destination, the notification sent by node u might take a longer time to reach the source, but the route recomputation time at node u is small and hence large packet losses are again prevented. If the link break occurs somewhere near the middle of the path, then both the local route recomputation mechanism and the route break notification mechanism are not fast enough, and hence a considerable amount of packet loss occurs in this case.

Consider the network topology shown in Figure 7.25. When the link between nodes 12 and 15 breaks, node 12 tries to reconnect to the destination using an alternate path that satisfies the bandwidth requirement. It also notifies the source

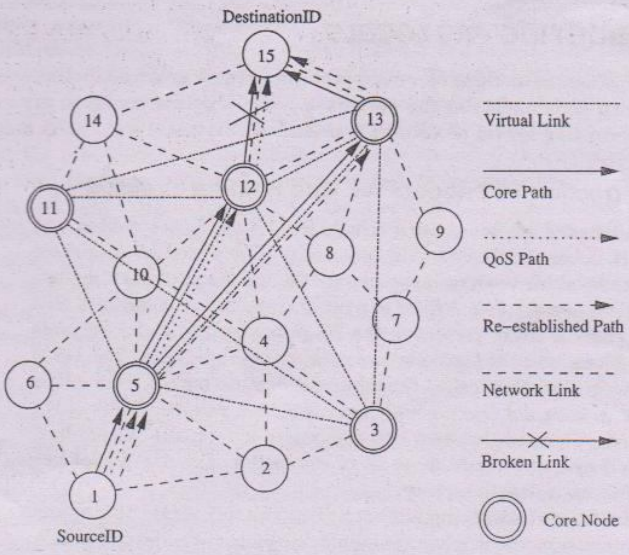


Figure 7.25. Route maintenance in CEDAR.

node about the link break. The source node tries to reconnect to the destination by reinitiating the route establishment process. In case node 12 does not have any other feasible path, then the alternate path 1-5-4-8-13-15 found by the source node is used for the further routing of packets.

Advantages and Disadvantages

The main advantage of CEDAR is that it performs both routing and QoS path computation very efficiently with the help of core nodes. The increase- and decrease-waves help in appropriate propagation of the stable high-bandwidth link information and the unstable low-bandwidth link information, respectively. Core broadcasts provide a reliable mechanism for establishing paths with QoS support. A disadvantage of this protocol is that since route computation is carried out at the core nodes only, the movement of the core nodes adversely affects the performance of the protocol. Also, the core node update information could cause a significant amount of control overhead.

7.6.2 Zone Routing Protocol

Zone routing protocol (ZRP) [18] is a hybrid routing protocol which effectively combines the best features of both proactive and reactive routing protocols. The key concept employed in this protocol is to use a proactive routing scheme within a limited zone in the r-hop neighborhood of every node, and use a reactive routing scheme for nodes beyond this zone. An *intra-zone routing protocol* (IARP) is used in the zone where a particular node employs proactive routing. The reactive routing protocol used beyond this zone is referred to as *inter-zone routing protocol* (IERP). The *routing zone* of a given node is a subset of the network, within which all nodes are reachable within less than or equal to *zone radius* hops. Figure 7.26 illustrates *routing zones* of node 8, with $r = 1$ hop and $r = 2$ hops. With *zone radius* $= 2$, the nodes 7, 4, 12, and 13 are interior nodes, whereas nodes 2, 3, 5, 9, 10, 13, and 15 are peripheral nodes (nodes with the shortest distance equal to the *zone radius*). Each node maintains the information about routes to all nodes within its *routing zone* by exchanging periodic route update packets (part of IARP). Hence the larger the *routing zone*, the higher the update control traffic.

The IERP is responsible for finding paths to the nodes which are not within the *routing zone*. IERP effectively uses the information available at every node's *routing zone*. When a node s (node 8 in Figure 7.27) has packets to be sent to a destination node d (node 15 in Figure 7.27), it checks whether node d is within its zone. If the destination belongs to its own zone, then it delivers the packet directly. Otherwise, node s bordercasts (uses unicast routing to deliver packets directly to the border nodes) the *RouteRequest* to its peripheral nodes. In Figure 7.27 node 8 bordercasts *RouteRequests* to nodes 2, 3, 5, 7, 9, 10, 13, 14, and 15. If any peripheral node finds node d to be located within its *routing zone*, it sends a *RouteReply* back to node s indicating the path; otherwise, the node reborderscasts the *RouteRequest* packet to the peripheral nodes. This process continues until node d is located. Nodes 10 and 14 find the information about node 16 to be available in their intra-zone

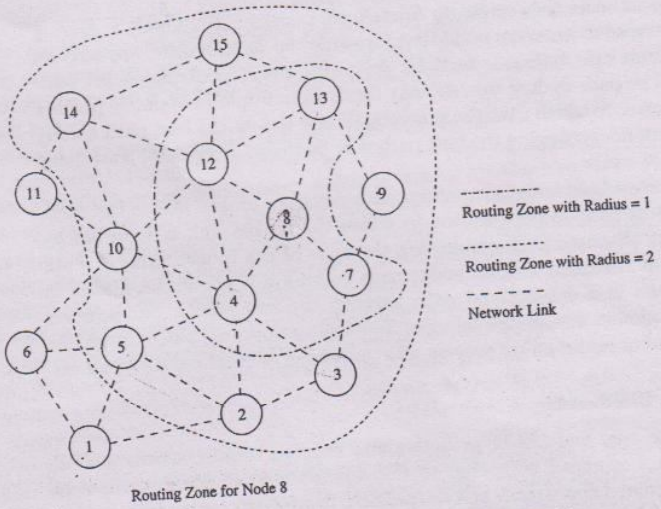


Figure 7.26. Routing zone for node 8 in ZRP.

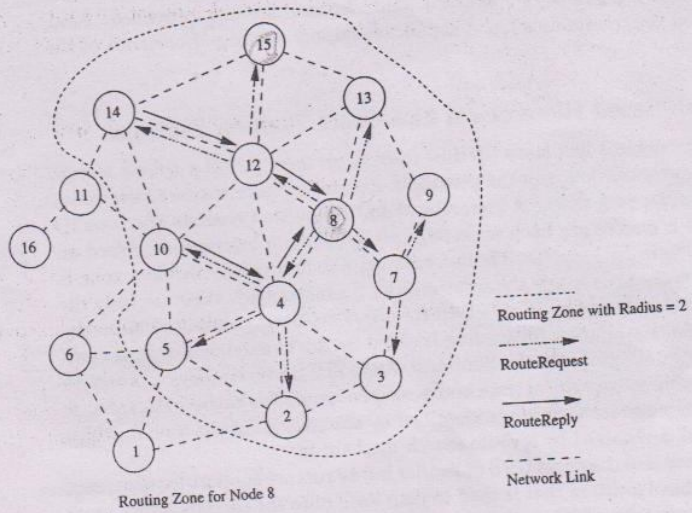


Figure 7.27. Path finding between node 8 and node 16.

routing tables, and hence they originate *RouteReply* packets back to node 8. During *RouteRequest* propagation, every node that forwards the *RouteRequest* appends its address to it. This information is used for delivering the *RouteReply* packet back to the source. The path-finding process may result in multiple *RouteReply* packets reaching the source, in which case the source node can choose the best path among them. The criterion for selecting the best path may be the shortest path, least delay path, etc.

When an intermediate node in an active path detects a broken link in the path, it performs a local path reconfiguration in which the broken link is bypassed by means of a short alternate path connecting the ends of the broken link. A path update message is then sent to the sender node to inform it about the change in path. This results in a sub-optimal path between two end points, but achieves quick reconfiguration in case of link failures. To obtain an optimal path, the sender reinitiates the global path-finding process after a number of local reconfigurations.

Advantages and Disadvantages

By combining the best features of proactive and reactive routing schemes, ZRP reduces the control overhead compared to the *RouteRequest* flooding mechanism employed in on-demand approaches and the periodic flooding of routing information packets in table-driven approaches. But in the absence of a query control, ZRP tends to produce higher control overhead than the aforementioned schemes. This can happen due to the large overlapping of nodes' *routing zones*. The query control must ensure that redundant or duplicate *RouteRequests* are not forwarded. Also, the decision on the zone radius has a significant impact on the performance of the protocol.

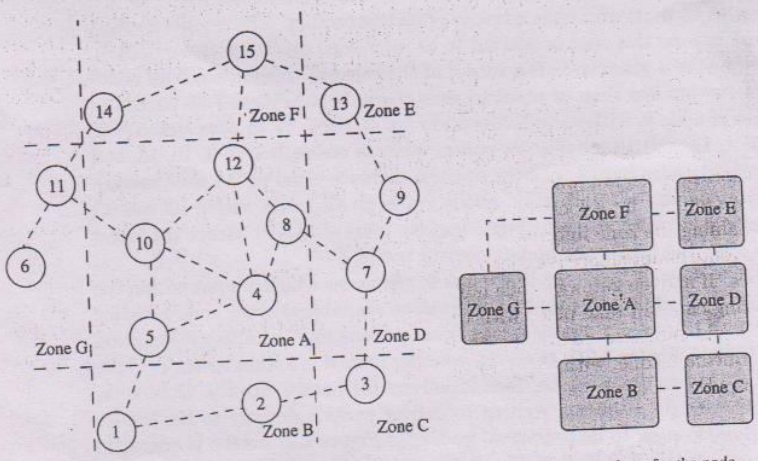
7.6.3 Zone-Based Hierarchical Link State Routing Protocol

Zone-based hierarchical link state (ZHLS) routing protocol [19] is a hybrid hierarchical routing protocol that uses the geographical location information of the nodes to form non-overlapping zones. A hierarchical addressing that consists of a zone ID and a node ID is employed. Each node requires its location information, based on which it can obtain its zone ID. The information about topology inside a zone is maintained at every node inside the zone, and for regions outside the zone, only the zone connectivity information is maintained. ZHLS maintains high-level hierarchy for inter-zone routing. Packet forwarding is aided by the hierarchical address comprising of the zone ID and node ID. Similar to ZRP, ZHLS also employs a proactive approach inside the geographical zone and a reactive approach beyond the zone. A destination node's current location is identified by the zone ID of the zone in which it is present and is obtained by a route search mechanism.

In ZHLS, every node requires GPS or similar infrastructure support for obtaining its own geographical location that is used to map itself onto the corresponding zone. The assignment of zone addresses to geographical areas is important and is done during a phase called the network design phase or network deployment phase. The area of the zone is determined by several factors such as the coverage of a single

node, application scenario, mobility of the nodes, and size of the network. For example, the ad hoc network formed by a set of hand-held devices with a limited mobility may require a zone radius of a few hundred meters, whereas the zone area required in the network formed by a set of ships, airplanes, or military tanks may be much larger.

The intra-zone routing table is updated by executing the shortest path algorithm on the node-level topology of the zone. The node-level topology is obtained by using the *intra-zone clustering* mechanism, which is similar to the link state updates limited to the nodes present in the zone. Each node builds a one-hop topology by means of a link request and link response mechanism. Once the one-hop topology is available, each node prepares link state packets and propagates them to all nodes in the zone. These update packets contain the node IDs of all nodes that belong to the zone, and node IDs and zone IDs of all nodes that belong to other zones. The nodes that receive link responses from nodes belonging to other zones are called *gateway* nodes. Data traffic between two zones will be relayed through these gateway nodes. For example, nodes 5, 8, 10, and 12 are the gateway nodes for zone A in Figure 7.28 (a). Every node in a zone is aware of the neighboring zones connected to its zone and the gateway nodes to be used for reaching them. Once the node-level link state packets are exchanged and the node-level topology is updated, every node in a zone generates a zone link state packet. The zone link state packet contains information about the zone-level connectivity. These zone link state packets are propagated



(a) Node–level topology in ZHLS

(b) Zone topology for the node–level topology in (a)

Figure 7.28. Zone-based hierarchical link state routing protocol.

Table 7.1. Zone link state packets

Source Zone	Zone Link State Packet
A	B, D, F, and G
B	C and A
C	B and D
D	A, C, and E
E	A, D, and F
F	A, E, and G
G	A and F

throughout the network by the gateway nodes. The zone-level topology is shown in Figure 7.28 (b). The zone link state packets originated by every zone are shown in Table 7.1. Using the information obtained from zone link state packets, a node can build the zone topology. The zone routing table can be formed for any destination zone by executing the shortest path algorithm on the zone-level topology. The zone link state packets are source sequence numbered and a time-stamp field is included to avoid stale link state packets. The association of the nodes to the respective zones helps in reducing routing overhead as in ZRP, but it includes the additional requirement of determining a given destination node's present location. If a source node Src wants to communicate with a destination node Dest, Src checks whether Dest resides in its own zone. If Dest belongs to the same zone, then packets are delivered to Dest as per the intra-zone routing table. If the destination Dest does not belong to the zone, then the node Src originates a location request packet containing the sender's and destination's information. This location request packet is forwarded to every other zone. The gateway node of a zone at which the location request packet is received verifies its routing table for the destination node for which the location request was originated. The gateway node that finds the destination node required by a location request packet originates a location response packet containing the zone information to the sender.

Route maintenance is easier with the presence of multiple gateway nodes between zones. If a given gateway node moves away, causing a zone-level connection failure, routing can still take place with the help of the other gateway nodes. This is due to the hierarchical addressing that makes use of zone ID and node ID. At any intermediate zone, with the most updated inter-zonal routing table, it forwards the data packets.

Advantages and Disadvantages

The hierarchical approach used in this protocol significantly reduces the storage requirements and the communication overhead created because of mobility. The zone-level topology is robust and resilient to path breaks due to mobility of nodes. Intra-zonal topology changes do not generate network-wide control packet trans-

missions. A main disadvantage of this protocol is the additional overhead incurred in the creation of the zone-level topology. Also the path to the destination is sub-optimal. The geographical information required for the creation of the zone level topology may not be available in all environments.

7.7 ROUTING PROTOCOLS WITH EFFICIENT FLOODING MECHANISMS

Many of the existing on-demand routing protocols employ flooding of *RouteRequest* packets in order to obtain a feasible path with the required packet-forwarding constraints. Flooding of control packets results in a significant amount of redundancy, wastage of bandwidth, increase in number of collisions, and broadcast storms[1] at times of frequent topological changes. Existing routing protocols that employ efficient flooding mechanisms to counter the requirement of flooding include preferred link-based routing (PLBR) protocols [20] and optimized link state routing (OLSR) protocol [21]. The former belongs to the on-demand routing protocols category and the latter belongs to the table-driven routing protocols category. These protocols utilize algorithms that require a minimum number of transmissions in order to flood the entire network.

7.7.1 Preferred Link-Based Routing Protocols

SSA [15] uses the preferred link approach in an implicit manner by processing a *RouteRequest* packet only if it is received through a strong link. Wired networks also employ preferred links mechanisms [22], but restrict themselves by selecting a single preferred link, based on heuristics that satisfy multiple constraints, for example, minimum cost and least delay required by the route. In ad hoc networks, the single preferred link model is not suitable due to reasons such as lack of topology information, continuously changing link characteristics, broadcast nature of the radio channel, and mobility of nodes.

Sisodia *et al.* proposed two algorithms in [20] known as preferred link-based routing (PLBR) protocols that employ a different approach in which a node selects a subset of nodes from its neighbors list (NL). This subset is referred to as the *preferred list* (PL). Selection of this subset may be based on link or node characteristics. Every *RouteRequest* packet carries the list of a selected subset of neighbors. All neighbors receive *RouteRequest* packets because of the broadcast radio channel, but only neighbors present in the PL forward them further. The packet is forwarded by K neighbors, where K is the maximum number of neighbors allowed in the PL. PLBR aims to minimize control overhead in the ad hoc wireless network. All nodes operate in the promiscuous mode. Each node maintains information about its neighbors and their neighbors in a table called neighbor's neighbor table (NNT). It periodically transmits a *beacon* containing the changed neighbor's

[1] Broadcast storm refers to the presence/origination of a large number of broadcast control packets for routing due to the high topological instability occurring in the network as a result of mobility.

information. PLBR has three main phases: route establishment, route selection, and route maintenance.

The route establishment phase starts when a source node (Src) receives packets from the application layer, meant for a destination node (Dest) for which no route is currently available. If Dest is in Src's NNT, the route is established directly. Otherwise, Src transmits a *RouteRequest* packet containing the source node's address (SrcID), destination node's address (DestID), a unique sequence number (SeqNum) (which prevents formation of loops and forwarding of multiple copies of the same *RouteRequest* packet received from different neighbors), a traversed path (TP) (containing the list of nodes through which the packet has traversed so far and the weight assigned to the associated links), and a PL. It also contains a time to live (TTL) field that is used to avoid packets being present in the network forever, and a $NoDelay$ flag, the use of which will be described later in this section. Before forwarding a *RouteRequest* packet, each eligible node recomputes the preferred list table (PLT) that contains the list of neighbors in the order of preference. The node inserts the first K entries of the PLT onto the PL field of the *RouteRequest* packet (K is a global parameter that indicates the maximum size of PL). The old PL of a received packet is replaced every time with a new PL by the forwarding node. A node is eligible for forwarding a *RouteRequest* only if it satisfies all the following criteria: the node ID must be present in the received *RouteRequest* packet's PL, the *RouteRequest* packet must not have been already forwarded by the node, and the TTL on the packet must be greater than zero. If Dest is in the eligible node's NNT, the *RouteRequest* is forwarded as a unicast packet to the neighbor, which might either be Dest or whose NL contains Dest. If there are multiple neighbors whose NL have Dest, the *RouteRequest* is forwarded to only one randomly selected neighbor. Otherwise, the packet is broadcast with a new PL computed from the node's NNT. PLT is computed by means of one of the two algorithms discussed later in this section. If the computed PLT is empty, that is, there are no eligible neighbors, the *RouteRequest* packet is discarded and marked as *sent*. If the *RouteRequest* packet reaches the destination, the route is selected by the route selection procedure given below.

When multiple *RouteRequest* packets reach Dest, the route selection procedure selects the best route among them. The criterion for selecting the best route can be the shortest path, or the least delay path, or the most stable path. Dest starts a timer after receiving the first *RouteRequest* packet. The timer expires after a certain *RouteSelectWait* period, after which no more *RouteRequest* packets would be accepted. From the received *RouteRequest* packets, a route is selected as follows.

For every *RouteRequest* i that reached Dest during the *RouteSelectWait* period, $\text{Max}(W^i_{min})$ is selected, where W^i_{min} is the minimum weight of a link in the path followed by i. If two or more paths have the same value for $Max(W^i_{min})$, the shortest path is selected. After selecting a route, all subsequent *RouteRequest* packets from the same Src with a SeqNum less than or equal to the SeqNum of the selected *RouteRequest* are discarded. If the $NoDelay$ flag is set, the route selection procedure is omitted and TP of the first *RouteRequest* reaching the Dest is selected as the route. The $NoDelay$ flag can be set if a fast connection setup is needed.

Mobility of nodes causes frequent path breaks that should be locally repaired to reduce broadcast of the *RouteRequest*. The local route repair broadcast mechanism used in ABR [14] has a high failure rate due to the use of restricted TTL, which increases the average delay in route reestablishment. PLBR uses a quick route repair mechanism which bypasses the down link (Dest side) node from the broken path, using information about the next two hops from NNT.

Algorithms for Preferred Links Computation

Two different algorithms have been proposed by Sisodia *et al.* in [20], for finding preferred links. The first algorithm selects the route based on degree of neighbor nodes (degree of a node is the number of neighbors). Preference is given to nodes whose neighbor degree is higher. As higher degree neighbors cover more nodes, only a few of them are required to cover all the nodes of the NNT. This reduces the number of broadcasts. The second algorithm gives preference to stable links. Links are not explicitly classified as stable or unstable. The notion of stability is based on the weight given to links.

Neighbor Degree-Based Preferred Link Algorithm (NDPL)

Let d be the node that calculates the preferred list table PLT. TP is the traversed path and OLD_{PL} is the preferred list of the received *RouteRequest* packet. The NNT of node d is denoted by NNT_d. $N(i)$ denotes the neighbors of node i and itself. Include list (INL) is a set containing all neighbors reachable by transmitting the *RouteRequest* packet after execution of the algorithm, and the exclude list (EXL) is a set containing all neighbors that are unreachable by transmitting the *RouteRequest* packet after execution of the algorithm.

1. In this step, node d marks the nodes that are not eligible for further forwarding the *RouteRequest* packet.
   (a) If a node i of TP is a neighbor of node d, mark all neighbors of i as reachable, that is, add $N(i)$ to INL.
   (b) If a node i of OLD_{PL} is a neighbor of node d, and $i < d$, mark all neighbors of node i as reachable, that is, include $N(i)$ in INL.
   (c) If neighbor i of node d has a neighbor n present in TP, mark all neighbors of i as reachable by adding $N(i)$ to INL.
   (d) If neighbor i of node d has a neighbor n present in OLD_{PL}, and $n < d$, here again add $N(i)$ to INL, thereby marking all neighbors of node i as reachable.
2. If neighbor i of node d is not in INL, put i in preferred list table PLT and mark all neighbors of i as reachable. If i is present in INL, mark the neighbors of i as unreachable by adding them to EXL, as $N(i)$ may not be included in this step. Here neighbors i of d are processed in decreasing order of their degrees. After execution of this step, the *RouteRequest* is guaranteed to reach

all neighbors of d. If EXL is not empty, some neighbor's neighbors n of node d are currently unreachable, and they are included in the next step.

3. If neighbor i of d has a neighbor n present in EXL, put i in PLT and mark all neighbors of i as reachable. Delete all neighbors of i from EXL. Neighbors are processed in decreasing order of their degrees. After execution of this step, all the nodes in NNT_d are reachable. Apply reduction steps to remove overlapping neighbors from PLT without compromising on reachability.

4. Reduction steps are applied here in order to remove overlapping neighbors from PLT without compromising on reachability.

   (a) Remove each neighbor i from PLT if $N(i)$ is covered by remaining neighbors of PLT. Here the minimum degree neighbor is selected every time.

   (b) Remove neighbor i from PLT whose $N(i)$ is covered by node d itself.

Weight-Based Preferred Link Algorithm (WBPL)

In this algorithm, a node finds the preferred links based on stability, which is indicated by a weight, which in turn is based on its neighbors' temporal and spatial stability.

1. Let $BCnt_i$ be the count of *beacons* received from a neighbor i and TH_{bcon} is the number of beacons generated during a time period equal to that required to cover twice the transmission range ($TH_{bcon} = \frac{2 \times transmission\ range}{maximum\ velocity \times period\ of\ beacon}$). Weight given to i based on time stability (WT^i_{time}) is

$$WT_{time} = \begin{cases} 1 & \text{if } BCnt_i > TH_{bcon} \\ BCnt_i/TH_{bcon} & \text{otherwise.} \end{cases}$$

2. Estimate the distance (D^i_{Est}) to i from the received power of the last few packets using appropriate propagation models. The weight based on spatial stability is $WT^i_{spatial} = \frac{R - D_{Est}}{R}$.

3. The weight assigned to the link i is the combined weight given to time stability and spatial stability. $W_i = WT^i_{time} + WT^i_{spatial}$.

4. Arrange the neighbors in a non-increasing order of their weights. The nodes are put into the PLT in this order.

5. If a link is overloaded, delete the associated neighbor from PLT. Execute *Step* 1 of NDPL and delete $\forall i,\ i \in PLT \cap i \in INL$. Also, delete those neighbors from PLT that satisfy *Step* 4 of NDPL.

Consider, for example, Figure 7.29, where the node 3 is the source and node 8 is the destination. S and U denote stable and unstable links. In WBPL and NDPL, the source that initiates that *RouteRequest* as Dest is not present in NNT and computes the preferred link table (PLT). Let $K = 2$ be the preferred list's

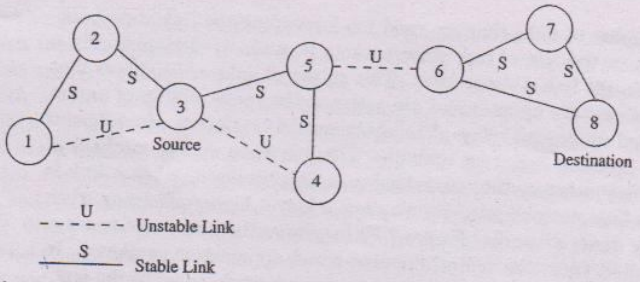


Figure 7.29. Example network. *Reproduced with permission from [20], © Korean Institute of Communication Sciences, 2002.*

size. In NDPL, after *Step* 2 the *PLT* becomes {5,1}, and after *Step* 3 also the *PLT* remains {5,1}. In reduction *Step* 4*b*, neighbor 1 is deleted from *PLT* and hence node 3 sends the *RouteRequest* only to neighbor 5. In WBPL, the weights are assigned to all neighbors according to *Steps* 1, 2, and 3, and all neighbors are in *PLT*. In *Step* 5, neighbors 1, 4, and 2 are deleted from *PLT* due to *Step* 4*a* and 4*b* of NDPL and hence only node 5 remains. Now the *RouteRequest* can be sent as a unicast packet to avoid broadcast. If it is broadcast, all the nodes receive the packet, but only node 5 can further forward it. As Dest 8 is present in node 5's *NNT*, it directly sends it to node 6 which forwards it to Dest. Here only three packets are transmitted for finding the route and the path length is 3. Now consider SSA. After broadcasts by nodes 3 and 5, the *RouteRequest* packet reaches node 6, where it is rejected and hence the *RouteRequest* fails to find a route. After timeout, it sets a flag indicating processed by *all* and hence finds the same route as WBPL and NDPL.

Advantages and Disadvantages

The efficient flooding mechanism employed in this protocol minimizes the broadcast storm problem prevalent in on-demand routing protocols. Hence this protocol has higher scalability compared to other on-demand routing protocols. The reduction in control overhead results in a decrease in the number of collisions and improvement in the efficiency of the protocol. PLBR achieves bandwidth efficiency at the cost of increased computation. Both NDPL and WBPL are computationally more complex than other *RouteRequest* forwarding schemes.

7.7.2 Optimized Link State Routing

The optimized link state routing (OLSR) protocol [21] is a proactive routing protocol that employs an efficient link state packet forwarding mechanism called *multipoint relaying*. This protocol optimizes the pure link state routing protocol. Optimizations are done in two ways: by reducing the size of the control packets and

by reducing the number of links that are used for forwarding the link state packets. The reduction in the size of link state packets is made by declaring only a subset of the links in the link state updates. This subset of links or neighbors that are designated for link state updates and are assigned the responsibility of packet forwarding are called *multipoint relays*. The optimization by the use of *multipoint relaying* facilitates periodic link state updates. The link state update mechanism does not generate any other control packet when a link breaks or when a link is newly added. The link state update optimization achieves higher efficiency when operating in highly dense networks. Figure 7.30 (a) shows the number of message transmissions required when the typical flooding-based approach is employed. In this case, the number of message transmissions is approximately equal to the number of nodes that constitute the network. The set consisting of nodes that are *multipoint relays* is referred to as *MPRset*. Each node (say, P) in the network selects an *MPRset* that processes and forwards every link state packet that node P originates. The neighbor nodes that do not belong to the *MPRset* process the link state packets originated by node P but do not forward them. Similarly, each node

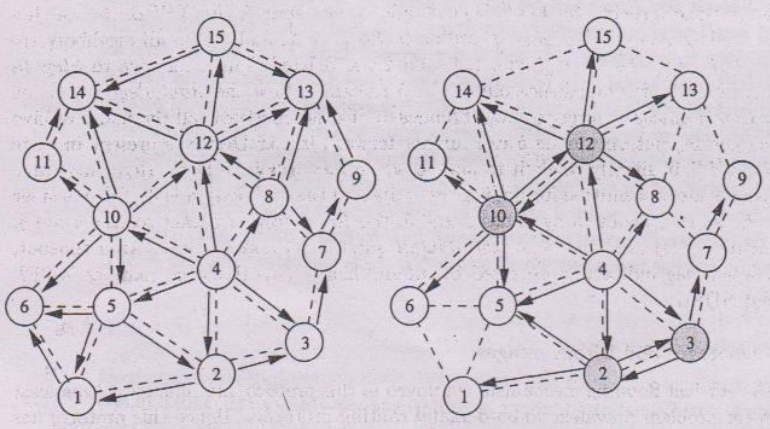


Figure 7.30. An example selection of MPRset in OLSR.

maintains a subset of neighbors called *MPR selectors*, which is nothing but the set of neighbors that have selected the node as a *multipoint relay*. A node forwards packets that are received from nodes belonging to its *MPRSelector* set. The members of both *MPRset* and *MPRSelectors* keep changing over time. The members of the *MPRset* of a node are selected in such a manner that every node in the node's two-hop neighborhood has a bidirectional link with the node. The selection of nodes that constitute the *MPRset* significantly affects the performance of OLSR because a node calculates routes to all destinations only through the members of its *MPRset*. Every node periodically broadcasts its *MPRSelector* set to nodes in its immediate neighborhood. In order to decide on the membership of the nodes in the *MPRset*, a node periodically sends *Hello* messages that contain the list of neighbors with which the node has bidirectional links and the list of neighbors whose transmissions were received in the recent past but with whom bidirectional links have not yet been confirmed. The nodes that receive this *Hello* packet update their own two-hop topology tables. The selection of *multipoint relays* is also indicated in the *Hello* packet. A data structure called *neighbor table* is used to store the list of neighbors, the two-hop neighbors, and the status of neighbor nodes. The neighbor nodes can be in one of the three possible link status states, that is, uni-directional, bidirectional, and *multipoint relay*. In order to remove the stale entries from the *neighbor table*, every entry has an associated timeout value, which, when expired, removes the table entry. Similarly a sequence number is attached with the *MPRset* which gets incremented with every new *MPRset*.

The *MPRset* need not be optimal, and during initialization of the network it may be same as the neighbor set. The smaller the number of nodes in the *MPRset*, the higher the efficiency of protocol compared to link state routing. Every node periodically originates *topology control* (TC) packets that contain topology information with which the routing table is updated. These TC packets contain the *MPRSelector* set of every node and are flooded throughout the network using the *multipoint relaying* mechanism. Every node in the network receives several such TC packets from different nodes, and by using the information contained in the TC packets, the *topology table* is built. A TC message may be originated by a node earlier than its regular period if there is a change in the *MPRSelector* set after the previous transmission and a minimal time has elapsed after that. An entry in the *topology table* contains a destination node which is the *MPRSelector* and a last-hop node to that destination, which is the node that originates the TC packet. Hence, the routing table maintains routes for all other nodes in the network.

Selection of Multipoint Relay Nodes

Figure 7.30 (b) shows the forwarding of TC packets using the *MPRset* of node 4. In this example, node 4 selects the nodes 2, 3, 10, and 12 as members of its *MPRset*. Forwarding by these nodes makes the TC packets reach all nodes within the transmitting node's two-hop local topology. The selection of the optimal *MPRset* is NP-complete [21]. In [21], a heuristic has been proposed for selecting the *MPRset*. The notations used in this heuristic are as follows: $N_i(x)$ refers to the i^{th} hop neighbor set of node x and $MPR(x)$ refers to the *MPRset* of node x.

1. $MPR(x) \leftarrow \phi$ /* Initializing empty $MPRset$ */
2. $MPR(x) \leftarrow$ { Those nodes that belong to $N_1(x)$ and which are the only neighbors of nodes in $N_2(x)$ }
3. While there exists some node in $N_2(x)$ which is not covered by $MPR(x)$
   (a) For each node in $N_1(x)$, which is not in $MPR(x)$, compute the maximum number of nodes that it covers among the uncovered nodes in the set $N_2(x)$.
   (b) Add to $MPR(x)$ the node belonging to $N_1(x)$, for which this number is maximum.

A node updates its *MPRset* whenever it detects a new bidirectional link in its neighborhood or in its two-hop topology, or a bidirectional link gets broken in its neighborhood.

Advantages and Disadvantages

OLSR has several advantages that make it a better choice over other table-driven protocols. It reduces the routing overhead associated with table-driven routing, in addition to reducing the number of broadcasts done. Hence OLSR has the advantages of low connection setup time and reduced control overhead.

7.8 HIERARCHICAL ROUTING PROTOCOLS

The use of routing hierarchy has several advantages, the most important ones being reduction in the size of routing tables and better scalability. This section discusses the existing hierarchical routing protocols for ad hoc wireless networks.

7.8.1 Hierarchical State Routing Protocol

The hierarchical state routing (HSR) protocol [23] is a distributed multi-level hierarchical routing protocol that employs clustering at different levels with efficient membership management at every level of clustering. The use of clustering enhances resource allocation and management. For example, the allocation of different frequencies or spreading codes to different clusters can improve the overall spectrum reuse. HSR operates by classifying different levels of clusters. Elected leaders at every level form the members at the immediate higher level. Different clustering algorithms, such as the one proposed in [8], are employed for electing leaders at every level. The first level of physical clustering is done among the nodes that are reachable in a single wireless hop. The next higher level of physical clustering is done among the nodes that are elected as leaders of each of these first-level clusters. In addition to the *physical clustering*, a *logical clustering* scheme has been proposed in HSR, which is based on certain relations among the nodes rather than on their geographical positions, as in the case of *physical clustering*.

Figure 7.31 illustrates the multilayer clustering defined by the HSR protocol. At the lowest level ($L = 0$), there are six cluster leaders (nodes 1, 2, 3, 4, 5, and 6). Nodes are classified as cluster leaders, or gateway nodes, or normal member nodes. A cluster leader is entrusted with responsibilities such as slot/frequency/code allocation, call admission control, scheduling of packet transmissions, exchange of routing information, and handling route breaks. In Figure 7.31, node 5 is a clusterhead marked as $L0{-}5$, which refers to the level of clustering ($L = 0$) and node ID (5). Similarly, each of the higher-level cluster leaders is also marked (*e.g.*, $L1{-}6$, $L2{-}6$, and $L3{-}6$ refer to the same node 6, but acting as leader with the given leader IDs at levels 1, 2, and 3, respectively). The spectrum reuse schemes, including spreading code assignment, can be used among the cluster leaders of the $L = 0$ clusters.

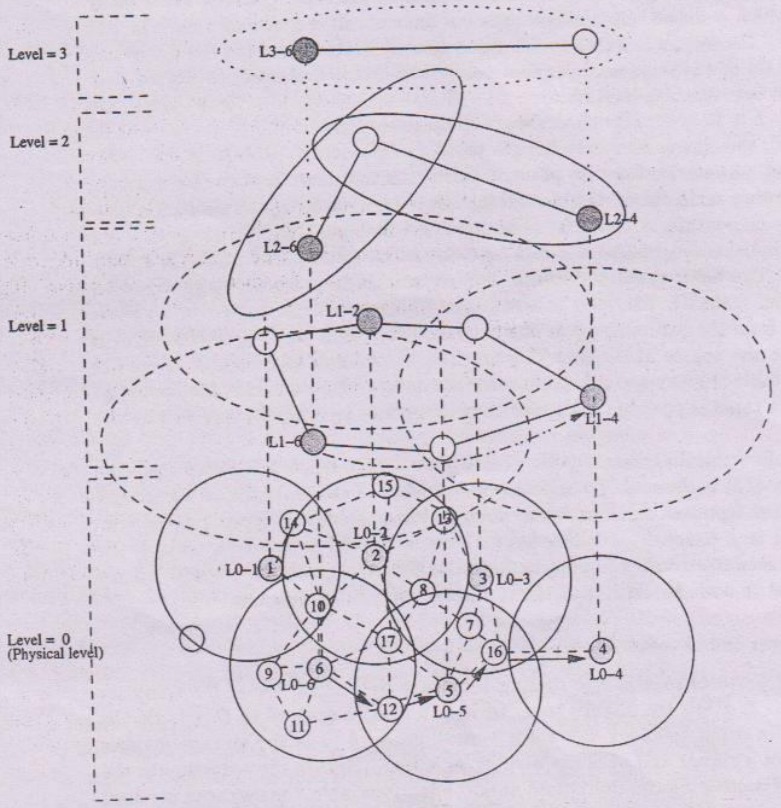


Figure 7.31. Example of HSR multi-level clustering.

For the nodes under the leadership of node 6 at level 0, the cluster members are nodes 9, 10, 11, 12, and 17. Those nodes that belong to multiple clusters are referred to as cluster gateway nodes. For the level 0 cluster whose leader is node 6, the cluster gateways are nodes 10, 12, and 17. The second level of clustering is done among the leaders of the first level, that is, the leaders of 0^{th} level clusters, $L0-1, L0-2, L0-3, L0-4, L0-5$, and $L0-6$, form the members of the first-level cluster.

Every node maintains information about all its neighbors and the status of links to each of them. This information is broadcast within the cluster at regular intervals. The cluster leader exchanges the topology and link state routing information among its peers in the neighborhood clusters, using which the next-higher-level clustering is performed. This exchange of link state information is done over multiple hops that consist of gateway nodes and cluster-heads. The path between two cluster-heads which is formed by multiple wireless links is called a *virtual link*. The link status for the *virtual link* (otherwise called *tunnel*) is obtained from the link status parameters of the wireless links that constitute the *virtual link*. In Figure 7.31, the path between first-level clusters $L1-6$ and $L1-4$ includes the wireless links $6-12-5-16-4$. The clustering is done recursively to the higher levels. At any level, the cluster leader exchanges topology information with its peers. After obtaining information from its peers, it floods the information to the lower levels, making every node obtain the hierarchical topology information. This hierarchical topology necessitates a hierarchical addressing which helps in operating with less routing information against the full topology exchange required in the link state routing. The hierarchical addressing defined in HSR includes the hierarchical ID (HID) and node ID. The HID is a sequence of IDs of cluster leaders of all levels starting from the highest level to the current node. This ID of a node in HSR is similar to the unique MAC layer address. The hierarchical addresses are stored in an HSR table at every node that indicates the node's own position in the hierarchy. The HSR table is updated whenever routing update packets are received by the node.

The hierarchical address of node 11 in Figure 7.31 is $< 6, 6, 6, 6, 11 >$, where the last entry (11) is the node ID and the rest consists of the node IDs of the cluster leaders that represent the location of node 11 in the hierarchy. Similarly, the HID of node 4 is $< 6, 4, 4, 4 >$. When node 11 needs to send a packet to node 4, the packet is forwarded to the highest node in the hierarchy (node 6). Node 6 delivers the packet to node 4, which is at the top-most level of the hierarchy.

Advantages and Disadvantages

The HSR protocol reduces the routing table size by making use of hierarchy information. In HSR, the storage required is $O(n \times m)$ compared to $O(n^m)$ that is required for a flat topology link state routing protocol (n is the average number of nodes in a cluster and m is the number of levels). Though the reduction in the amount of routing information stored at nodes is appreciable, the overhead involved in exchanging packets containing information about the multiple levels of hierarchy and the leader election process make the protocol unaffordable in the ad hoc wire-

less networks context. Besides, the number of nodes that participate in an ad hoc wireless network does not grow to the dimensions of the number of nodes in the Internet where the hierarchical routing is better suited. In the military applications of ad hoc wireless networks, the hierarchy of routing assumes significance where devices with higher capabilities of communication can act as the cluster leaders.

7.8.2 Fisheye State Routing Protocol

The table-driven routing protocols generate routing overhead that is dependent on the size of the network and mobility of the nodes, whereas the routing overhead generated by on-demand routing protocols are dependent on the number of connections present in the system in addition to the above two factors. Hence, as the number of senders in the network increases, the routing overhead also increases. ZRP uses an intra-zone proactive approach and an inter-zone reactive approach to reduce control overhead. The fisheye state routing (FSR) protocol [23] is a generalization of the GSR [24] protocol. FSR uses the *fisheye* technique to reduce information required to represent graphical data, to reduce routing overhead. The basic principle behind this technique is the property of a fish's eye that can capture pixel information with greater accuracy near its eye's focal point. This accuracy decreases with an increase in the distance from the center of the focal point. This property is translated to routing in ad hoc wireless networks by a node, keeping accurate information about nodes in its local topology, and not-so-accurate information about far-away nodes, the accuracy of the network information decreasing with increasing distance. FSR maintains the topology of the network at every node, but does not flood the entire network with the information, as is done in link state routing protocols. Instead of flooding, a node exchanges topology information only with its neighbors. A sequence numbering scheme is used to identify the recent topology changes. This constitutes a hybrid approach comprising of the link-level information exchange of distance vector protocols and the complete topology information exchange of link state protocols. The complete topology information of the network is maintained at every node and the desired shortest paths are computed as required. The topology information exchange takes place periodically rather than being driven by an event. This is because instability of the wireless links may cause excessive control overhead when event-driven updates are employed. FSR defines routing scope, which is the set of nodes that are reachable in a specific number of hops. The scope of a node at two hops is the set of nodes that can be reached in two hops. Figure 7.32 shows the scope of node 5 with one hop and two hops. The routing overhead is significantly reduced by adopting different frequencies of updates for nodes belonging to different scopes.

The link state information for the nodes belonging to the smallest scope is exchanged at the highest frequency. The frequency of exchanges decreases with an increase in scope. This keeps the immediate neighborhood topology information maintained at a node more precise compared to the information about nodes farther away from it. Thus the message size for a typical topology information update packet is significantly reduced due to the removal of topology information regarding

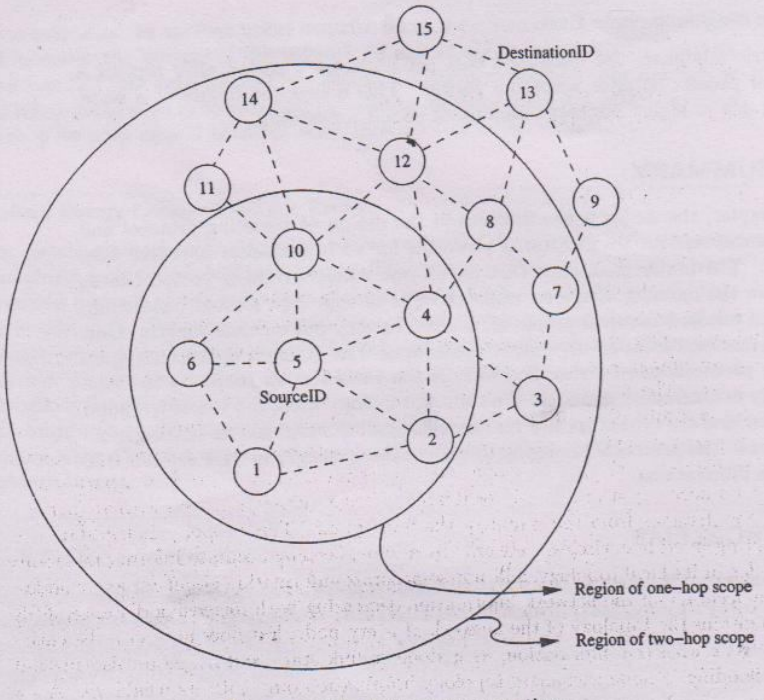


Figure 7.32. Fisheye state routing.

the far-away nodes. The path information for a distant node may be inaccurate as there can be staleness in the information. But this is compensated by the fact that the route gets more and more accurate as the packet nears its destination. FSR scales well for large ad hoc wireless networks because of the reduction in routing overhead due to the use of the above-described mechanism, where varying frequencies of updates are used.

Figure 7.33 illustrates an example depicting the network topology information maintained at nodes in a network. The routing information for the nodes that are one hop away from a node are exchanged more frequently than the routing information about nodes that are more than one hop away. Information regarding nodes that are more than one hop away from the current node are listed below the dotted line in the topology table.

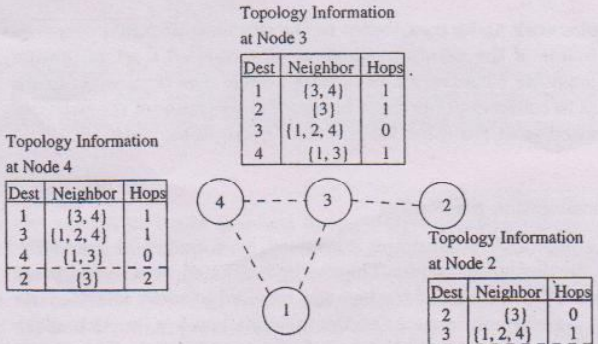


Figure 7.33. An illustration of routing tables in FSR.

Advantages and Disadvantages

The notion of multi-level scopes employed by FSR significantly reduces the bandwidth consumed by link state update packets. Hence, FSR is suitable for large and highly mobile ad hoc wireless networks. The choice of the number of hops associated with each scope level has a significant influence on the performance of the protocol at different mobility values, and hence must be carefully chosen.

7.9 POWER-AWARE ROUTING PROTOCOLS

In a deviation from the traditional wired network routing and cellular wireless network routing, power consumption by the nodes is a serious factor to be taken into consideration by routing protocols for ad hoc wireless networks. This is because, in ad hoc wireless networks, the routers are also equally power-constrained just as the nodes are. This section discusses some of the important routing metrics that take into consideration this energy factor.

7.9.1 Power-Aware Routing Metrics

The limitation on the availability of power for operation is a significant bottleneck, given the requirements of portability, weight, and size of commercial hand-held devices. Hence, the use of routing metrics that consider the capabilities of the

power sources of the network nodes contributes to the efficient utilization of energy and increases the lifetime of the network. Singh *et al.* proposed a set of routing metrics in [25] that supports conservation of battery power. The routing protocols that select paths so as to conserve power must be aware of the states of the batteries at the given node as well as at the other intermediate nodes in the path.

Minimal Energy Consumption per Packet

This metric aims at minimizing the power consumed by a packet in traversing from source node to the destination node. The energy consumed by a packet when traversing through a path is the sum of the energies required at every intermediate hop in that path. The energy consumed at an intermediate hop is a function of the distance between the nodes that form the link and the load on that link. This metric does not balance the load so that uniform consumption of power is maintained throughout the network. The disadvantages of this metric include selection of paths with large hop length, inability to measure the power consumption at a link in advance when the load varies, and the inability to prevent the fast discharging of batteries at some nodes.

Maximize Network Connectivity

This metric attempts to balance the routing load among the *cut-set* (the subset of the nodes in the network, the removal of which results in network partitions). This assumes significance in environments where network connectivity is to be ensured by uniformly distributing the routing load among the *cut-set*. With a variable traffic origination rate and unbounded contention in the network, it is difficult to achieve a uniform battery draining rate for the *cut-set*.

Minimum Variance in Node Power Levels

This metric proposes to distribute the load among all nodes in the network so that the power consumption pattern remains uniform across them. This problem is very complex when the rate and size of data packets vary. A nearly optimal performance can be achieved by routing packets to the least-loaded next-hop node.

Minimum Cost per Packet

In order to maximize the life of every node in the network, this routing metric is made as a function of the state of the node's battery. A node's cost decreases with an increase in its battery charge and vice versa. Translation of the remaining battery charge to a cost factor is used for routing. With the availability of a battery discharge pattern, the cost of a node can be computed. This metric has the advantage of ease in the calculation of the cost of a node and at the same time congestion handling is done.

Minimize Maximum Node Cost

This metric minimizes the maximum cost per node for a packet after routing a number of packets or after a specific period. This delays the failure of a node, occurring due to higher discharge because of packet forwarding.

7.10 SUMMARY

In this chapter, the major issues involved in the design of a routing protocol and the different classifications of routing protocols for ad hoc wireless networks were described. The major challenges that an ad hoc wireless routing protocol must address are the mobility of nodes, rapid changes in topology, limited bandwidth, hidden and exposed terminal problems, limited battery power, time-varying channel properties, and location-dependent contention. The different approaches upon which the protocols can be classified include the classification based on the type of topology maintenance approach, the routing topology used, the use of temporal information, and the type of specific resource utilization considered for making routing decisions. The protocols belonging to each of these categories were explained in detail with illustrations.

7.11 PROBLEMS

1. Discuss the differences in the maintenance of topology information in various protocols such as CGSR, HSR, SSA, ABR, PLBR, OLSR, and CEDAR.
2. Discuss the differences in topology reorganization in DSDV and CGSR routing protocols.
3. How is the cluster-head selected in the CGSR protocol? In the CGSR protocol, the resources of the node chosen as the cluster-head get drained very quickly, more rapidly than the other nodes in the cluster. How can this problem be overcome?
4. Is a table-driven routing protocol suitable for high-mobility environments?
5. Both ABR and SSA use stability information for routing. How do they differ in using the stability information?
6. What are the advantages of hierarchical topology-based protocols over protocols that use flat topologies?
7. Consider the topology given in Figure 7.34. Simulate DSR, SSA, and ABR protocols for path establishment from node 1 to node 10, find the paths found and the ratio of the number of *RouteRequest* packets sent in the network. (Links labeled "U" refer to unstable ones.)
8. For the sample topology given in Figure 7.35, find the Zone Link State packets for the various zones marked.

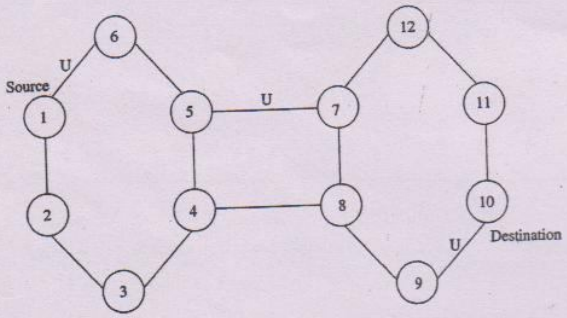


Figure 7.34. Topology for Problem 7.

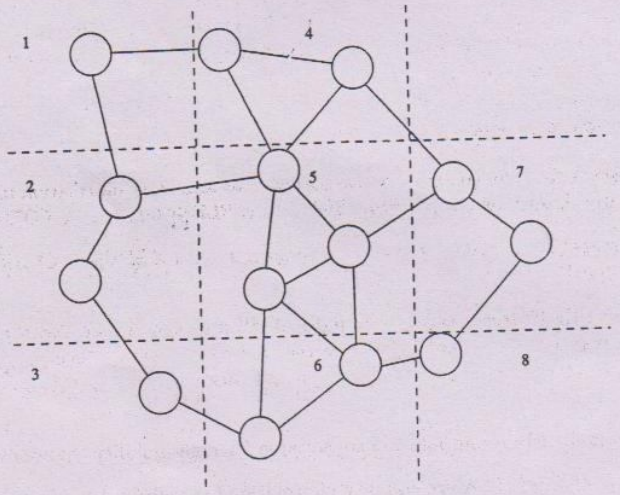


Figure 7.35. Topology for Problem 8.

9. Does the LCC algorithm (when run consistently with node degrees or node IDs) give a deterministic result? If so, prove the above fact. Otherwise, give a counter-example.

10. What are the key differences between the LAR1 and the LAR2 algorithms?

11. For the network shown in Figure 7.36, determine the fisheye routing table for nodes 7 and 5.

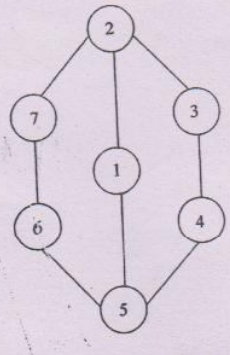


Figure 7.36. Topology for Problem 11.

12. For the topology shown in Figure 7.37, create a DAG for node 1 as the source and node 7 as the destination in TORA. If the link between nodes 4 and 6 breaks as shown in the figure, find the change in the DAG. (Also mark the distance from the destination.)

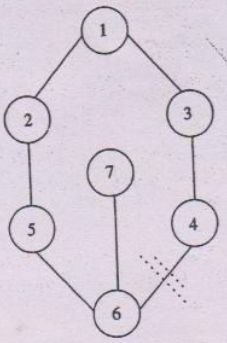


Figure 7.37. Topology for Problem 12.

13. Identify some of the key issues involved in QoS routing in ad hoc networks.