Chapter IX
Routing Protocols for Ad-Hoc Networks

Muhammad Mahmudul Islam
Monash University, Australia

Ronald Pose
Monash University, Australia

Carlo Kopp
Monash University, Australia

ABSTRACT

Ad-hoc networks have been the focus of research interest in wireless networks since 1990. Nodes in an ad-hoc network can connect to each other dynamically in an arbitrary manner. The dynamic features of ad-hoc networks demand a new set of routing protocols that are different from the routing schemes used in traditional wired networks. A wide range of routing protocols has been proposed to overcome the limitations of wired routing protocols. This chapter outlines the working mechanisms of state-of-the-art ad-hoc routing protocols. These protocols are evaluated by comparing their functionalities and characteristics. Related research challenges are also discussed.

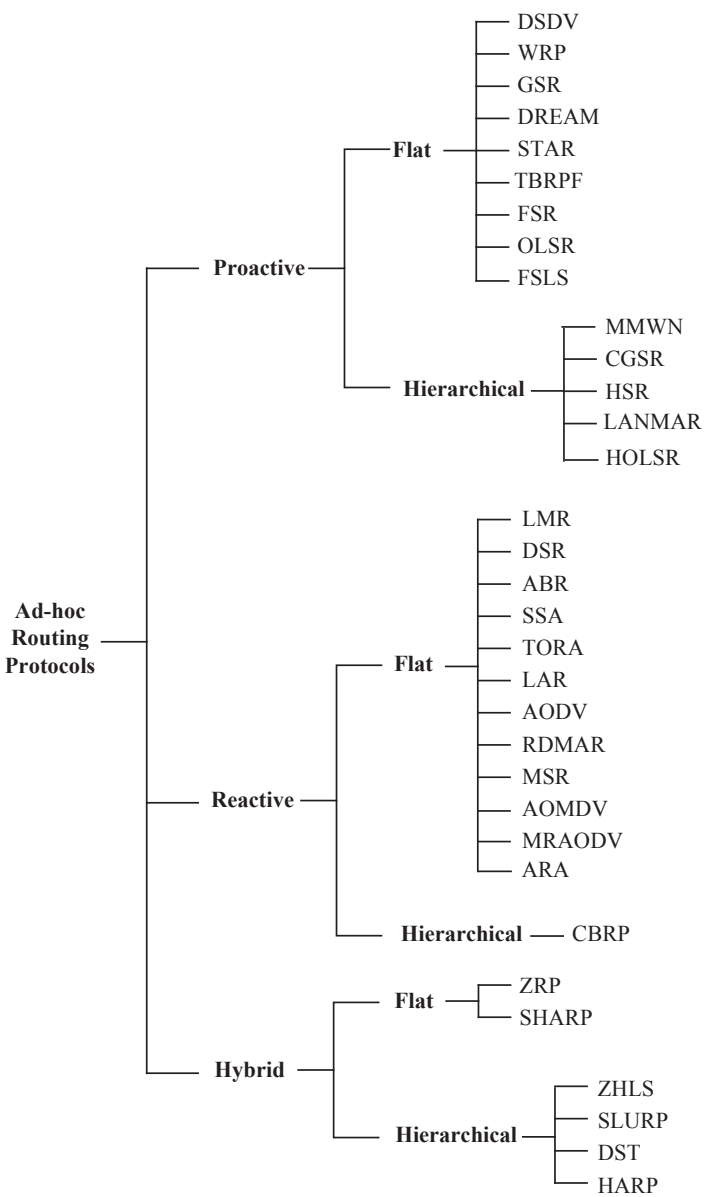
INTRODUCTION

An ad-hoc network consists of a set of nodes that communicate using a wireless medium over single or multiple hops and do not need any pre-existing infrastructure such as access points or base stations. Ad-hoc networks can comprise of mobile, static, or both types of nodes. Ad-hoc networks containing mobile nodes are known as MANETs (mobile ad-hoc networks). An example of ad-hoc networks with static nodes is SAHN (suburban ad-hoc network) (Kopp & Pose, 1998).

Since ad-hoc networks can be rapidly deployed, they are attractive for digital communication in battlefields, rescue operations after a disaster, and so forth. Ad-hoc networks are also useful in civilian forums for running/demanding multimedia applications such as video conferencing.

The topology of an ad-hoc network can change dynamically due to dynamic link failure

Figure 1. Classification of ad-hoc routing protocols based on routing strategy and network structure



and node mobility. Its size and node density can vary unpredictably since nodes can join or leave the network, or move arbitrarily from one location to another. Due to the lack of a clear physical boundary, a wireless communication channel is usually shared by more nodes than with a cabled network. Nodes in ad-hoc networks can be constrained by computation, battery, and transmission power. Thus routing in ad-hoc networks is more challenging than in wired networks.

Ad-hoc routing protocols can be classified into three major groups based on the routing strategy. These are: (1) pro-active or table driven, (2) reactive or on-demand, and (3) hybrid. In pro-active routing protocols routes to a destination are determined when a node joins the network or changes its location, and are maintained by periodic route updates. In reactive routing protocols routes are discovered when needed and expire after a certain period. Hybrid routing protocols combine the features of both pro-active and reactive routing protocols to scale well with network size and node density. Each of these groups can be further divided into two sub-groups based on the routing structure: (1) flat and (2) hierarchical. In flat routing protocols nodes are addressed by a flat addressing scheme and each node plays an equal role in routing (Hong, Xu, & Gerla, 2002). On the other hand, different nodes have different routing responsibilities in hierarchical routing protocols. These protocols require a hierarchical addressing system to address the nodes. Figure 1 depicts classification of various ad-hoc routing protocols according to these groups and sub-groups.

Reviews and comparisons of various ad-hoc routing protocols have been presented in earlier publications (Abolhasan, Wysocki, & Dutkiewicz, 2004; Hong et al., 2002; Royer & Toh, 1999). We include more routing protocols and evaluate them by comparing their functionalities and characteristics. We also outline open research challenges in this area.

PRO-ACTIVE ROUTING PROTOCOLS

Pro-active routing protocols require each node to maintain up-to-date routing information to every other node (or nodes located within a specific region) in the network. The various routing protocols in this group differ in how topology changes are detected, how routing information is updated, and what sort of routing information is maintained at each node. These routing protocols are based on the working principles of two popular routing algorithms used in wired networks. They are known as *link-state routing* and *distance vector routing.*

In the link-state approach, each node maintains at least a partial view of the whole network topology. To achieve this, each node periodically broadcasts link-state information such as link activity and delay of its outgoing links to all other nodes using network-wide flooding. When a node receives this information, it updates its view of the network topology and applies a shortest-path algorithm to choose the next hop for each destination. The well-known routing protocol OSPF (open shortest path first) is an example of a link-state routing protocol.

On the other hand, each node in distance vector routing periodically monitors the cost of its outgoing links and sends its routing table information to all neighbours. The cost can be measured in terms of the number of hops or time delay or other metrics. Each entry in the routing table contains at least the ID of a destination, the ID of the next hop neighbour through which the destination can be reached at minimum cost, and the cost to reach the destination. Thus, through periodic monitoring of outgoing links, and dissemination of the routing table information, each node maintains an estimate of the shortest distance to every node in the network. DBF (distributed Bellman Ford) (Bertsekas & Gallager, 1987) and RIP (routing information protocol) are classic examples of distance vector routing algorithms.

Due to the limitations in communication resources such as battery power, the potentially very large number of nodes, network dynamics, and node mobility, these protocols are not well suited for ad-hoc networks. The following protocols have been proposed to alleviate the problems of traditional link-state and distance vector routing strategies.

Destination-Sequenced Distance-Vector (DSDV) Routing

DSDV (Perkins & Bhagwat, 1994) is a distance vector routing protocol that ensures loop-free routing by tagging each route table entry with a sequence number.

DSDV requires each node to maintain a routing table. This routing table lists all available destinations from that node. Each entry, corresponding to a particular destination, contains the number of hops to reach the destination and the address of the neighbour that acts as a next-hop towards the destination. Each entry is also tagged with a sequence number that is assigned by the respective destination. To maintain the consistency of the routing tables in a dynamically varying topology, each node periodically broadcasts updates to its neighbours. Updates are also broadcast to neighbours immediately when significant new information, such as link breakage, is available. In order to reduce potentially large amounts of traffic generated by these updates, two modes of updates can be employed. The first type is known as "full dump" where multiple network protocol data units may be needed to carry all available routing information to the neighbours. The other mode of update is referred to as "incremental" where only routing information changed since the last "full dump" is sent in a single network protocol data unit to the neighbours. If topological change is not rapid, "full dump" can be employed less frequently than "incremental" mode to reduce network traffic.

Updated route information, broadcast by a node *X* to its neighbours, contains the address of the destination *Y*, *HC+1* where *HC* (hop count) is the number of hops to reach *Y* from *X*, the sequence number assigned to the initial updated route information broadcast by *Y* and the new sequence number assigned by *X* unique to this broadcast. Any route with the older sequence number is replaced with that of the newer sequence number. If two route updates have the same sequence number, the route with the smaller metric is chosen in order to obtain a shorter route.

Since the broadcasts of route information are asynchronous events, it is possible that a node can conceivably always receive two routes to the same destination, with a newer sequence number, one after another from different neighbours but always gets the route with higher metric first. This may lead to continuing broadcast of new route information upon receiving every new sequence number from that destination. In order to reduce the network traffic for such careless broadcasts, it has been suggested to keep track of the weighted average of the time until the route to *Y* with best metric is received, and delaying the broadcast of updated route information of *Y* by the length of the settling time.

Due to network-wide periodic and triggered update requirements, DSDV introduces excessive communication overhead. After a node or link failure DSDV may engage in prolonged exchanges of distance information before converging to shortest paths. These problems can become unacceptable if network size or node mobility increases.

Wireless Routing Protocol (WRP)

WRP (Murthy & Garcia-Luna-Aceves, 1995) is a distance vector routing protocol that aims to reduce the possibility of forming temporary routing loops in mobile ad-hoc networks. It belongs to a subclass of the distance vector protocol known as the path-finding algorithm that

eliminates the counting-to-infinity problem of DBF (distributed Bellman Ford). Each node, in a path-finding algorithm, obtains the shortest-path spanning tree to all destinations of the network from each one-hop neighbour. A node uses this information along with the cost of adjacent links to construct its own shortest-path spanning tree for all destinations.

Each node in WRP maintains a distance table, a routing table, a link-cost table, and a message retransmission list. The distance table of node *X* is a matrix that contains the distance to each destination *D* via each neighbour *N* and predecessor *P*. The second-to-last hop of a destination is referred to as a predecessor. An entry in the routing table of *X* for destination *D* contains the distance between *X* and *D*, the predecessor and successor on this route, and a tag to identify if the entry is a simple path, a loop, or invalid. The neighbour of a node is referred to as the successor for a particular destination if the neighbour offers the smallest cost and loop-free path to the destination. Predecessor and successor information are needed to detect routing loops and to prevent the counting-to-infinity problem. An entry in the link-cost table of *X* contains the cost of the link and the number of timeouts since *X* has received any error-free messages from the neighbour connected to that link. The message retransmission list (MRL) contains one or more retransmission entries where each entry enables *X* to know which update message has to be retransmitted since a neighbour has not acknowledged it in the previous transmission.

WRP requires each node to exchange routing tables with its neighbours using update messages periodically as well as after the status of one of its links changes. When a node *X* transmits an update message for the first time, it lists all its neighbours so that they can send acknowledgments. If the update message is retransmitted, *X* obtains the list of neighbours from its MRL that have not acknowledged the update message and includes them in the retransmitted message. In this way WRP can reduce network traffic by asking the neighbours, who have sent acknowledgments for the same update message previously, not to send any more acknowledgments for the retransmitted update message.

If a node does not make any change in its routing table since the last update, it has to send an idle HELLO message to ensure connectivity. On receiving an update message, a node modifies its distance table and looks for better routes using updated information. Any new route thus found is relayed back to the node from which the update message was received. On receiving an acknowledgment for an update message, a node updates its message retransmission list.

Each time a node detects any change in a link, it checks the consistency of the predecessor information reported by all neighbours. This eliminates routing loops and ensures fast convergence after a link failure or recovery that would otherwise be impossible if the consistency check was performed only for the predecessor information reported by the neighbour connected to that link.

Fewer nodes are informed in WRP than in DSDV during a link failure. Hence WRP can find shortest path routes faster than DSDV. On the other hand, WRP requires the use of HELLO packets similar to DSDV even when there is no packet to send. Thus WRP does not allow nodes to enter into a sleep mode to conserve energy (Royer & Toh, 1999).

Multimedia Support in Mobile Wireless Networks (MMWN)

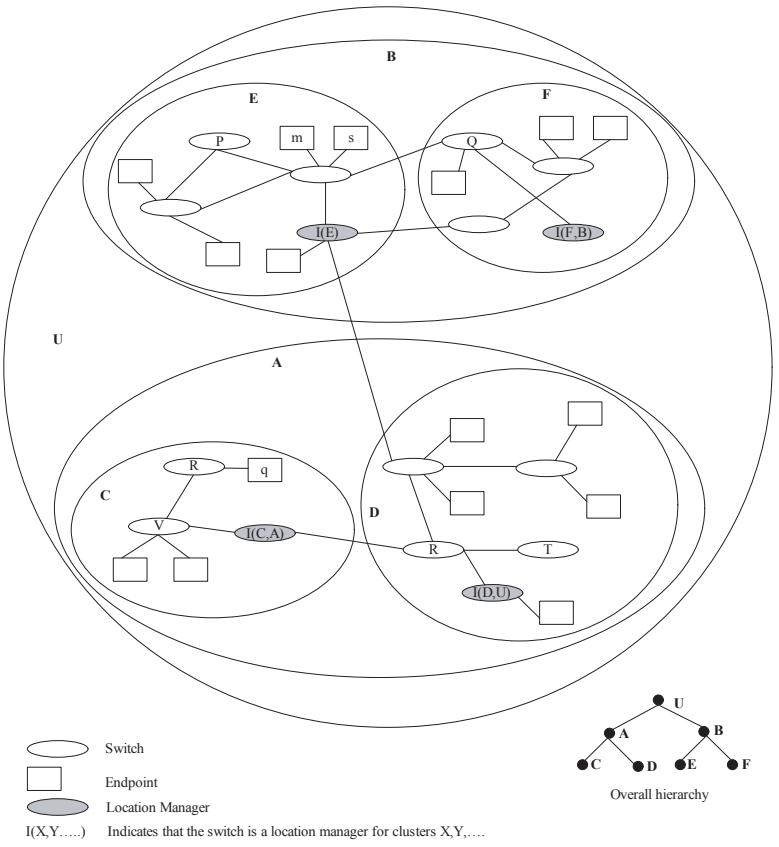
The MMWN (Kasera & Ramanathan, 1997) routing protocol maintains an ad-hoc network using a clustering hierarchy in order to reduce routing control overheads where node mobility is high or nodes do not communicate frequently.

In general each cluster contains three types of nodes: switches, (nodes *V, R* in Figure 2), endpoints

(nodes *m, s* in Figure 2), and a location manager (nodes *I(C,A), I(D,U)* in Figure 2). A location manager of a cluster is elected from among all switches in the cluster and is responsible for performing location management, that is, location updating and location finding. Only switches and location managers can route packets. Endpoints can only be sources and destinations.

At the lowest level of the hierarchy, level-0, endpoints affiliate with switches to form a cell. Multiple cells form a cluster of level-1 and so on. For example, in Figure 2, clusters *C, D, E, F* are at level-1, clusters *A, B* are at level-2 and top level cluster *U* is at level-3. Each switch and endpoint are assumed to have a globally unique identifier, referred to as the switch-id and endpoint-id respectively, which do not change over time. Every cluster, except the cluster at level-0, is identified by a cluster-id unique among its siblings. The cluster-id of a cluster at level-0 is denoted by the corresponding switch-id. The hierarchical address of a cluster C_k is $C_1.C_2...C_k$ where C_{i-1} is the parent cluster of C_i, where $1 \leq i \leq k$-1. For example, the hierarchical address of cluster *D* in Figure 2 is *U.A.D*. The hierarchical address of a switch is the hierarchical address of the cluster to which the switch belongs, suffixed by the switch-id. The hierarchical address of an endpoint is the hierarchical address of the switch with which the endpoint is affiliated. Unlike node identifiers, the

Figure 2. The clustering hierarchy used in MMWM



hierarchical addresses are autonomously acquired and may change with time.

Each endpoint is associated with two parameters referred to as the "roaming cluster" and "roaming level" for the purpose of its location updating process. The roaming cluster of an endpoint is the lowest level cluster containing the endpoint such that an update is triggered if and only if the endpoints exit this roaming cluster. The roaming level of an endpoint is the hierarchical level of its roaming cluster. For example, if the roaming level of endpoint *q* in Figure 2 is 2, then its roaming cluster is *U.A*. The roaming level of an endpoint may be changed dynamically based on its call frequency and speed. In general, the more mobile an endpoint is, the higher should be its roaming level.

The location update message, generated by an endpoint, contains four fields: its endpoint-id, old hierarchical address, new hierarchical address and roaming level. The update message is sent to the switch it has just affiliated with. The switch then forwards the message to the appropriate location managers. When a location manager receives an update message it trims the last *n* terms of the old and new hierarchical addresses contained in the message, where *n* is its hierarchical level, and compares the resultant hierarchical addresses. If they are not equal, then the message is forwarded to the parent location manager that repeats the check and this process continues until the message reaches a location manager such that the trimmed hierarchical addresses match.

Each location manager receiving an update message creates an association entry for the endpoint or updates the existing association entry for the endpoint. The last location manager, where the comparison resulted in equality, sends a cancel message to the previous location manager that was associated with the endpoint in order to delete the invalid entries for the endpoint's previous location.

When a switch changes its cluster, it also obtains a new hierarchical address. It then sends an aggregated update message, which contains its new hierarchical address and the list of endpoints affiliated with it, to the new location manager. The handling of this message is similar to that of those generated by endpoints.

When a cluster splits into two, the location manager of the original cluster remains with one of the new clusters and the new cluster gets a new location manager. The new location manager initially does not contain any association list. It fills up its list from the information obtained from the old location manager.

When a cluster merges with another cluster, one of the location managers resigns and sends its association list to the surviving location manager so that the new location manager can have a full list of the endpoints contained in the merged cluster.

A node wishing to obtain a hierarchical address of a remote endpoint sends a query message to the switch it is associated with. The switch searches its association list to see if the target endpoint is in its own cell. If the target resides within the same cell, the location finding procedure terminates. Otherwise the switch forwards the query message to its parent location manager that also searches its association list to find the target endpoint. If an entry is found, the query message is forwarded to the respective location manager contained in a child cluster. If no entry is found, the query message is forwarded to the parent location manager. This is how the query message makes its way up the hierarchy until it finds an entry for the target endpoint and then down the hierarchy until it reaches the location manager at level-0.

The final location manager may or may not contain the endpoint depending on the roaming level of the target endpoint. If the roaming level of the target endpoint is 0, the final location manager, that is, the final switch, is assumed to contain the target endpoint. In this case, the final switch sends a reply message to the originator of the query message containing the switch-id and

the hierarchical address of the target endpoint. If the roaming level of the target endpoint is greater than 1, the final switch floods a page message containing the same information as the query message throughout the cluster of level *n*, where *n* is the roaming level of the target endpoint. When a switch receives a page message it checks if it contains the target endpoint. If the endpoint is found in the association list, a reply message is sent to the originator of the query message containing the switch-id and the hierarchical address of the target endpoint.

Since the location management is closely related to hierarchical structure of the network, messages have to travel through the hierarchical tree of the location managers. For the same reason, any change in the hierarchical cluster membership of location managers will cause reconstruction of the hierarchical location management tree and introduce complex consistency management. Thus MMWN introduces implementation problems that are potentially complex to solve (Pei, Gerla, Hong, & Chiang, 1999).

Clusterhead Gateway Switch Routing (CGSR)

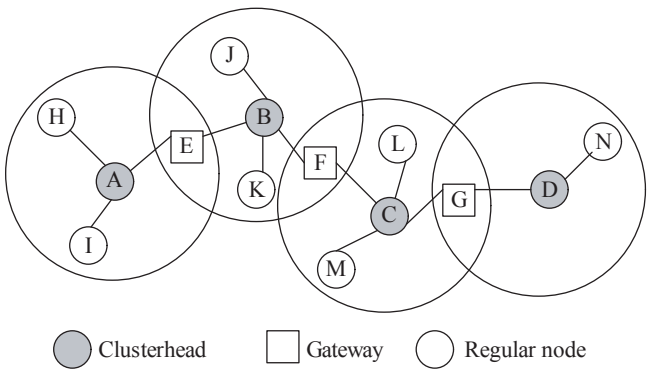
CGSR (Chiang, Wu, Liu, & Gerla, 1997) is a hierarchical routing protocol that uses DSDV (Perkins & Bhagwat, 1994) as its underlying routing algorithm but reduces the size of routing update packets in large networks by partitioning the whole network into multiple clusters. The addressing scheme used here is simpler than that of MMWM (Kasera & Ramanathan, 1997) since CGSR uses only one level of clustering hierarchy.

Each cluster in CGSR contains a clusterhead (nodes *A*, *B, C* and *D* in Figure 3) that manages all nodes within its radio transmission range. A node that belongs to more than one cluster works as a gateway (nodes *E*, *F*, and *G* in Figure 3) to connect the overlapping clusters.

CGSR requires each node to maintain two tables: a cluster member table and a routing table. The cluster member table records the clusterhead address for each node in the network and is broadcast periodically. The routing table maintains only one entry for each clusterhead, no matter how many members each clusterhead has. Thus CGSR reduces the size of the routing table as well as the size of the routing update messages. Each entry in the routing table contains the address of a clusterhead and the address of the next hop to reach the clusterhead.

A packet from a node is first sent to its clusterhead. The clusterhead then forwards the packet to its neighbouring clusterhead through the corresponding. This process continues until the packet

Figure 3. Illustration of single-level clustering hierarchy used in CGSR



reaches the clusterhead of the destination node. At this stage, the destination clusterhead simply forwards the packet to the destination.

Since each node only maintains routes to its clusterhead, routing overhead is lower in CGSR compared to DSDV or WRP. However, time to recover from a link failure is higher than DSDV or WRP since additional time is required to perform clusterhead reselection (Royer & Toh, 1999).

Global State Routing (GSR)

GSR (Chen & Gerla, 1998) improves the link-state algorithm by adopting the routing information dissemination method used in DBF. Instead of flooding GSR transmits link-state updates to neighbouring nodes only.

In GSR each node maintains a neighbour list, a topology table, a next-hop table, and a distance table. The neighbour list of a node *X* contains its neighbours that are within its radio transmission range. The topology table contains the link-state information of each destination *Y* as reported by *Y* and a timestamp indicating the time *Y* has generated this information. For each destination *Y*, the next hop table contains the next hop *Z*, which is a one-hop neighbour of *X*, to which packets must be forwarded from *X* destined for *Y*. The distance table contains the shortest distance to each destination from *X* in terms of the number of hops.

Whenever a node receives a routing message containing link-state updates from one of its neighbours, it updates its topology table if the timestamp is newer than the one stored in the table. After the node reconstructs the routing table it broadcasts the information to its neighbours with other link-state updates.

The key difference between GSR and traditional link-state algorithms is the way routing information is disseminated. A link-state algorithm floods a small packet containing a single link-state update whenever the link status changes. On the other hand, a node in GSR transmits longer packets containing multiple link-state updates to its neighbours. Therefore GSR requires fewer update messages than a traditional link-state algorithm in an ad-hoc network with frequent topology changes. Thus GSR can optimise MAC (medium access control) layer throughput since frequent smaller packets incur higher MAC layer overhead than infrequent longer packets. However, as the network size and node density increase, the size of each update message becomes larger.

Distance Routing Effect Algorithm for Mobility (DREAM)

DREAM (Basagni, Chlamtac, Syrotiuk, & Woodward, 1998) uses location information using GPS (global positioning system) to provide loop-free multi-path routing for mobile ad-hoc networks.

Each node in DREAM maintains a location table that records location information of all nodes. DREAM minimises routing overhead, that is, location update overhead, by employing two principles referred to as the "distance effect" and the "mobility rate". The "distance effect" states that the greater the distance between two nodes the slower they appear to move with respect to each other. Thus nodes that are far apart need to update their location information less frequently than the nodes closer together. This is realised in DREAM by associating an age with each location update message that corresponds to how far from the sender the message can travel. The "mobility rate" states another interesting observation that the faster a node moves, the more frequently it needs to advertise its new location information to other nodes.

When a node *X* needs to send a packet to a destination *Y*, it uses its location table to find the direction of *Y* and selects a set of one-hop neighbours in that direction. If the set is empty the packet is broadcast to all neighbours. Otherwise *X* transmits the packet to the selected set of neighbours. Each neighbour repeats this process until the packet reaches *Y*. *Y* responds to each packet with an acknowledgment that is sent

to *X*. If *X* does not receive an acknowledgment within a timeout period, it retransmits the packet by flooding in order to increase the possibility of reaching *Y*.

Source Tree Adaptive Routing (STAR)

STAR (Garcia-Luna-Aceves & Spohn, 1999) is based on a link-state algorithm that minimises the number of routing update packets disseminated into the network to save bandwidth (i.e., reduce network traffic) at the expense of not maintaining optimum routes to destinations.

STAR requires each node to maintain a source tree, which is a set of links constituting complete paths to destinations. A node knows the status of its adjacent links and the source trees reported by its neighbours. With this information the node generates a topology table and computes its own source tree. It also derives a routing table by running Dijkstra's shortest-path algorithm on its source tree. Each entry in the routing table consists of a destination address, the cost (e.g., the number of hops) of the route to destination and the next hop address towards the destination.

A node sends updates on its source tree to its neighbours only when it loses all routes to one or more destinations, when it detects new destinations, when it determines local changes to its source tree can create long-term routing loops, or when the cost of the routes exceeds a certain threshold. Instead of periodic updates for each link, the conditional dissemination of updates enables STAR to reduce the bandwidth required for link-state updates. This prevents nodes from maintaining optimum routes to destinations. The partial topology graphs of a network maintained in the nodes can change frequently as the neighbours keep sending different source trees in large and highly mobile ad-hoc networks (Abolhasan et al., 2004). In this case STAR may introduce significant memory and processing overheads.

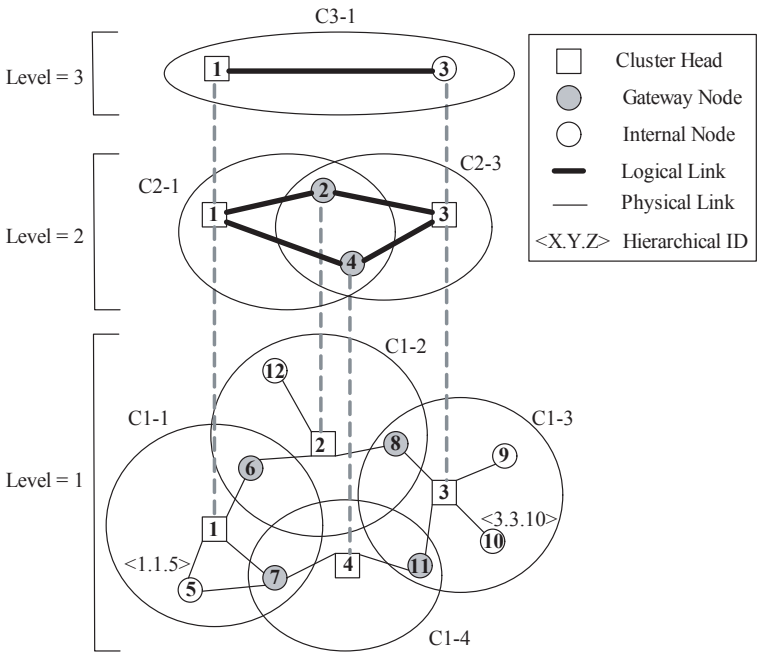
Hierarchical Star Routing (HSR)

Pei et al. (1999) have proposed a hierarchical link-state routing protocol, referred to as HSR, designed to scale well with network size. They argue that the location management (i.e., the location updating and location finding) in MMWM is quite complicated since it couples location management with physical clustering. HSR aims to make the location management task simpler by separating it from physical clustering.

HSR maintains a hierarchical topology by clustering group of nodes based on their geographical relationship. The clusterheads at a lower level become members of the next higher level. The new members then form new clusters, and this process continues for several levels of clusters. The clustering is beneficial for the efficient utilisation of radio channels and the reduction of network layer overhead (i.e., routing table storage, processing, and transmission). In addition to the multi-level clustering HSR provides multi-level logical partitioning based on the functional affinity between nodes (e.g., tanks in a battlefield or the colleagues of the same organisation). Logical partitioning is responsible for mobility management.

An example of a three-level hierarchal clustering structure is illustrated in Figure 4. The node IDs, shown in the lowest level, are physical such as MAC (medium access control) addresses. In general each cluster contains three types of nodes: a clusterhead (nodes *1*, *2*, *3*, and *4* for the lowest hierarchical level), gateway node (nodes *6*, *7*, *8*, and *11* for the lowest hierarchical level), and internal node (nodes *5*, *9*, *10*, and *12* for the lowest hierarchical level). At the lowest level of the hierarchy, each node monitors the state of each link and broadcasts the observed link-state information within the cluster. The clusterhead summarises the received link-state information and sends it to the neighbouring clusterheads through gateways. The clusterheads of a level *Cx* become the members of the cluster of level *Cx*+1

Figure 4. An example of hierarchical clustering in HSR



and they exchange their logical link information as well as their summarised lower level link-state information among each other. This process continues up to the highest level. A node at each level disseminates all the gathered link-state information up to this level to the nodes in the level below. In this way each node in the lowest level gets hierarchical topology information of all nodes. A hierarchical address, referred to as HID (hierarchical ID), of a node is defined as the sequence of MAC addresses of the nodes on the path from the top of the hierarchy to the node itself. For example, the HIDs of nodes *5* and *10* are <*1.1.5*> and <*3.3.10*> respectively. A gateway can have more than one hierarchical address. If node *5* wants to send a data packet to node *10* it sends the packet to its top hierarchy node *1*. Since node *1* has a logical link, that is, a tunnel, to node *3* through the path *1→6→5→2→8→3*, it sends the packet to node *3* through this path. Finally node *3* delivers by packet to node *10* along the downward hierarchical path that is in this case its immediate neighbour. Thus a HID is enough to ensure delivery of packets from anywhere in the network to a remote destination.

In HSR nodes are also partitioned into logical partitions, that is, subnets, in order to resolve implementation problems of MMWM. In addition to the MAC addresses, nodes are assigned logical addresses of type <subnet, host>. Each subnet contains a location management server (LMS). Each member of a logical subnet knows the HID of its LMS. All nodes in a subnet have to register their logical addresses with its LMS.

Registration is both periodic and event driven. All LMSs advertise their HIDs to the top hierarchy. Optionally the LMS HIDs can be propagated downwards to all nodes. When a node wants to send a packet to a destination, it sends the packet to its network layer with the logical address of the destination. The network layer finds the HID of the destination's LMS from its LMS and sends the packet to the destination's LMS. The destination's LMS then forwards the packet to the destination. If the source and the destination know each other's HIDs, they can communicate directly bypassing their LMSs.

Though HSR requires less memory and communication overhead than any flat pro-active routing protocol, it introduces additional overhead (like any other cluster based protocol) for forming and maintaining clusters.

Topology Broadcast Based on Reverse Path Forwarding (TBRPF)

TBRPF (Bellur & Ogier, 1999) is a link-state based routing protocol that uses the concept of reverse-path forwarding to broadcast link-state updates in the reverse direction along the spanning tree formed by minimum-hop paths from all nodes to the source of the update. Unlike a pure link-state routing algorithm, which requires all nodes to forward update packets, TBRPF requires only the non-leaf nodes in the broadcast tree to forward update packets. Thus TBRPF generates less update traffic than pure link-state routing algorithms. The use of minimum-hop tree instead of a shortest-path tree makes the broadcast tree more stable and thus results in less communication cost to maintain the tree.

Each node in TBRPF maintains a list of its one-hop neighbours and a topology table. Each entry in the topology table for a link contains the most recent cost and sequence number associated with that link. With this information each node can compute a source tree that provides shortest paths to all reachable remote nodes. Moreover, for each node $src \neq i$, node i keeps record of: (1) a parent $p_i(src)$ which is the neighbour of node i and the next hop on the minimum-hop path from node i to node src, (2) a list of children $children_i(src)$ which are the neighbours of i, and (3) the sequence number $sn_i(src)$ of the most recent link-state update originating from node src. The parents $p_i(src)$, for all $i \neq src$, form a minimum-hop spanning tree directed towards src.

Node src sends an update message to other nodes by broadcasting the update message in the reverse direction along its spanning tree. A node i accepts the update message, modifies its topology table and forwards the update message to every node in $children_i(src)$ if the update message is received from $p_i(src)$ and the update message has a larger sequence number than the corresponding entry in its topology table.

If a node i detects that the parent for node src has changed, it sends a CANCEL PARENT message, which contains the identity of src, to the current parent if it is reachable. It also sends a NEW PARENT message, containing the identity of src and $sn_i(src)$, to the newly computed parent. If the new parent receives the message, it finds out all the link-state information from its topology table that originating from src and sends it to i.

When a node i detects any change in its neighbourhood, for example, appearance of a new node or loss of connectivity with an existing neighbour, it updates the link cost and the sequence number field for the corresponding link in its topology table. It sends the corresponding link-state message to all its neighbours in $children_i(i)$. Unless the change has caused a neighbour to become inaccessible, the node recomputes its list of parents. If it detects any change in its parent list, it performs the task as outlined previously.

Ogier, Templin, and Lewis (2004) have modified TBRPF where src sends only the updates of those links to i that can result in changes to i's source tree. This modification can result in less

update traffic at the expense of having partial topology information at each node.

Fisheye State Routing (FSR)

FSR (Pei, Gerla, & Chen, 2000) is an improvement of GSR. GSR requires the entire topology table to be exchanged among neighbours. This can consume a considerable amount of bandwidth when the network size becomes large. FSR is an implicit hierarchical routing protocol that uses the "fisheye" technique (Kleinrock & Stevens, 1971) to reduce the size of large update messages generated in GSR for large networks. The scope of the fisheye of a node is defined as the set of nodes that can be reached within a given number of hops.

FSR, like GSR, requires each node to maintain a neighbour list, a topology table, a next hop table, and a distance table. Unlike GSR, entries in the topology table corresponding to nodes within the smaller scope are propagated to the neighbours with higher frequency. Thus the fisheye approach enables FSR to reduce the size of update messages.

In FSR each node can maintain fairly accurate information about its neighbours. As the distance (i.e., the scope of fisheye) from the node increases, the detail and accuracy of information also decreases. As a result a node may not have precise knowledge of the best route to a distant destination. However this imprecise knowledge is claimed to be compensated by the fact that the route becomes progressively more accurate as the packet gets closer to the destination.

Landmark Ad-Hoc Routing (LANMAR)

LANMAR (Gerla, Hong, & Guangyu, 2000; Guangyu, Geria, & Hong, 2000) is a combined link-state (i.e., FSR) and distance vector routing (e.g., DSDV) protocol that aims to be scalable. It borrows the notion of landmark (Tsuchiya, 1988) to keep track of logical subnets. Such subnets can be formed in an ad-hoc network with the nodes that are likely to move as a group such as brigades in the battlefield or colleagues in the same organisation.

When a network is formed for the first time, LANMAR only uses the FSR functionality. Gradually one of the nodes learns from the FSR tables that there it contains a certain number of nodes within its fisheye scope. It then proclaims itself as a landmark for that group (i.e., the subnet). When more than one node declares itself as a landmark for the same group, the node with the largest number of group members wins the election. In case of a tie, the node with the lowest ID breaks the tie.

A distance vector routing mechanism propagates the routing information about all the landmarks in the entire network. Within each subnet, a mechanism, similar to FSR, is used to update topology information. As a result, each node contains detailed topology information about all the nodes within its fisheye scope and the distance and routing vector information to all landmarks. Consequently LANMAR reduces both routing table size and control overhead for large MANETs.

When a source needs to send a packet to a destination within its fisheye scope, it uses the FSR routing table. If the destination is located outside the fisheye scope, the packet is routed towards the landmark of the destination. When the packet arrives within the scope of the destination, it is routed using FSR directly to the destination, possibly without going through the landmark.

LANMAR guarantees the shortest path from a source to a destination if the destination is located within the scope of the source. For a remote destination, though packets will reach the destination's landmark through a shortest path, the packets may travel through additional hops before the destination is reached (Hong et al.,

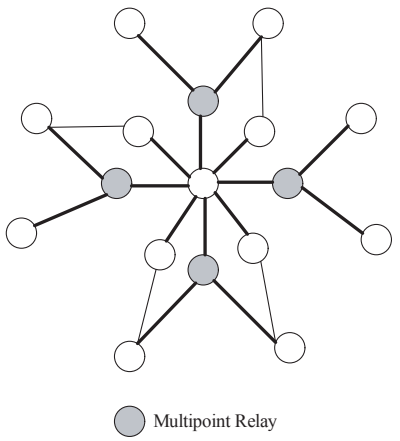
2002). LANMAR improves routing scalability for large MANETs with the assumption that nodes under a landmark move in groups.

Optimised Link-State Routing (OLSR)

OLSR (Jacquet, Muhlethaler, Clausen, Laouiti, Qayyum, & Viennot, 2001) optimises the link-state algorithm by compacting the size of the control packets that contain link-state information and reducing the number of transmissions needed to flood these control packets to the whole network.

In OLSR a node *X* selects a set of immediate (i.e., one-hop) neighbours called the multi-point relays (MPRs) of that node (see Figure 5). MPRs of *X* must cover (in terms of radio range) all the nodes that are two hops away from *X*. Every node within a two-hop neighbourhood of *X* must have bi-directional links with the MPRs of *X*. OLSR reduces the size of the control packets since in each control packet a node puts only the link-state information of the neighbouring MPRs instead of all neighbours. It minimises flooding of control traffic since only the MPRs, instead of all neighbours, of a node are responsible for relaying network-wide broadcast traffic.

Figure 5. Multi-point relays in OLSR



To select the MPRs, each node *X* periodically broadcasts HELLO messages to its one-hop neighbours. Each HELLO message contains a list of neighbours that are connected to *X* via bi-directional links and also the list of neighbours that are heard by *X* but are not connected via bi-directional links. This HELLO message can be received by all one-hop neighbours of *X*, but is not relayed to further nodes. Each node, receiving a HELLO message, can learn the link-state information of all neighbours up to two hops. This information is stored in a neighbour table and used to select MPRs.

Each node broadcasts specific control messages called the topology control (TC) messages. Each TC message, originating from a node *X*, contains the list of MPRs of *X* with a sequence number and is forwarded only by the MPRs of the network. Each node maintains a topology table that represents the topology of the network built from the information obtained from the TC messages.

Each node also maintains a routing table where each entry in the routing table corresponds to an optimal route, in terms of the number of hops, to a particular destination. Each entry consists of a destination address, next-hop address, and the number of hops to the destination. The routing table is built based on the information available in the neighbour table and the topology table.

Fuzzy Sighted Link-State (FSLS) Routing

FSLS (Santivez, Ramanathan, & Stavrakakis, 2001) is a link-state routing protocol that restricts the dissemination scope of routing updates in space and time similar to FSR (Pei et al., 2000) in order to scale well with network size.

Each node in FSLS sends a link-state update every $2^{i-1} \times T$ $(i = 1, 2, 3...)$ to all the nodes contained within a scope of s_i where T is the minimum

link-state update transmission interval and s_i is the hop distance from the node.

If s_i is set to infinity, each update message can reach the entire network and FSLS becomes similar to any standard link-state routing algorithm with the exception that a link status change is not propagated in this variant of FSLS until the current T interval finishes. If $s_i = i$, FSLS induces the same control overheads as FSR. Authors have shown that if $s_i = 2^i$, FSLS can induce the least amount of control overhead compared to other variants.

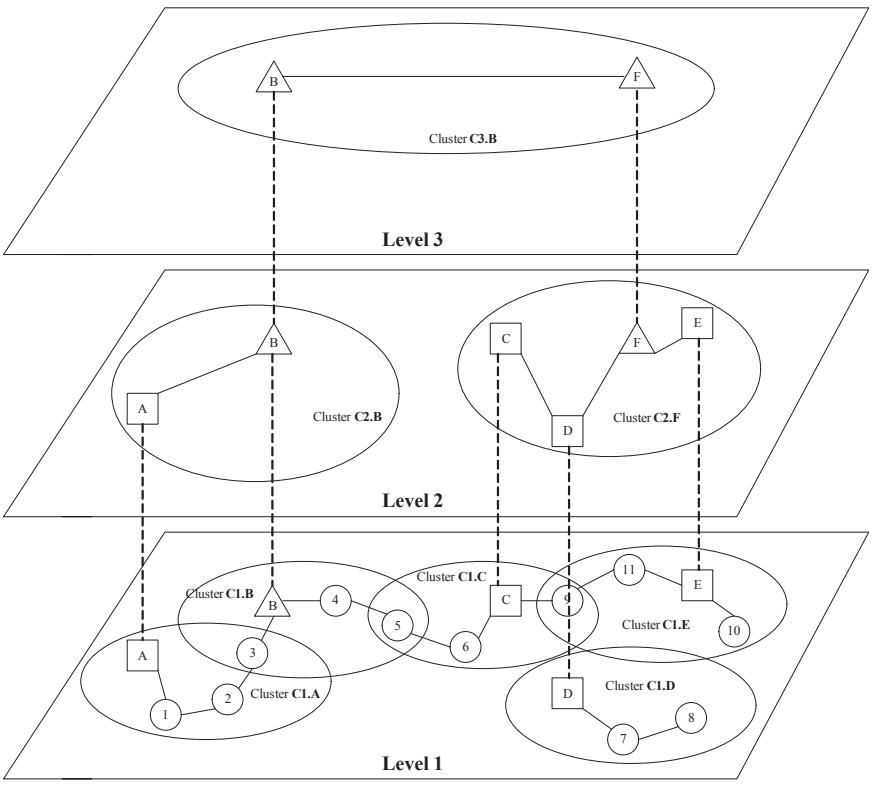
Hierarchical Optimised Link-State Routing (HOLSR)

HOLSR (Gonzalez, Ge, & Lamont, 2005) is a routing mechanism derived from the OLSR protocol. The main improvement realised by HOLSR over OLSR is a reduction in routing control overhead, for example, topology control information, in large heterogeneous mobile ad-hoc networks. A heterogeneous mobile ad-hoc network is defined as a network of mobile nodes where different mobile nodes have different communication capabilities, for example, multiple radio interfaces with varying transmission powers.

To reduce routing control overhead, HOLSR organises mobile nodes into multiple topology levels based on their varying communication capabilities. Figure 6 illustrates the network structure formed with multiple topology levels.

Nodes having only one wireless interface with low transmission power form topology level 1. These are denoted by circles in Figure 6. Nodes that have up to two wireless interfaces can form

Figure 6. Hierarchical network structure in HOLSR (Gonzalez, Ge, & Lamont, 2005)



topology level 2. These nodes are designated by squares in the figure. One of the wireless interfaces of these nodes is used to communicate with the nodes of level 1. The other interface is used to relay messages at level 2 using a frequency band or a medium access control protocol different from the one used for communication at level 1. Nodes denoted by triangles in the figure represent high capacity nodes equipped with up to three wireless interfaces. The notation, for example, *C2.B*, used to name clusters in the figure means a cluster of level 2 where node *B* is the clusterhead.

Each topology level comprises of one or more clusters. Each cluster consists of a clusterhead and other mobile nodes. A node configured as a clusterhead during the HOLSR startup process invites other nodes to join its cluster by periodically sending out CIA (cluster ID announcement) messages to neighbouring nodes. To reduce the number of packet transmissions, CIA and HELLO messages are sent together. From HELLO messages, a node gets information about its immediate and two-hop neighbours. A CIA message contains two fields: *clusterhead* and *distance*. The *clusterhead* field indicates the interface address of the clusterhead and the *distance* denotes the distance in number of hops to the clusterhead. When a clusterhead generates a CIA, it sets the value of *distance* to 0. A node receiving the CIA message joins the cluster to which the clusterhead belongs, increases the value of the *distance* by 1, and then sends the CIA to its neighbours to invite them to join the cluster. Any node can receive more than one CIA from different clusterheads. In this case the node joins the cluster that is closer in terms of hop count. If the hop count values of multiple CIA messages are the same, the node joins the cluster from which it receives the first CIA. This process is repeated at each topology level.

Due to mobility, a node might find a clusterhead closer than the one it is currently connected to. In this case the node will join the closest cluster by changing its clusterhead.

Each CIA message has a timeout value. If a node does not receive any CIA message from its existing clusterhead within the timeout period of the previously received CIA message, it can consider joining another cluster provided that it receives CIA messages from other clusters.

If no CIA messages are received, that is, the network is no longer heterogeneous, the HOLSR treats the entire network as one cluster and operates as the original OLSR.

In HOLSR, a clusterhead acts as a gateway through which messages are relayed to other clusters. This requires each clusterhead to be aware of the membership information of other clusters of the same topology level. The higher the position a node possesses in the topology level, the more information it gets about the network. In this way, the nodes at the highest topology level possess full knowledge of all the nodes of the network. Since all nodes do not contain information of all other nodes of the network, the size of the routing tables of lower-level nodes in HOLSR is less than that of OLSR.

The TC (topology control) messages used in the OLSR are usually restricted within a cluster in HOLSR. If a node is located in the overlapping regions of several clusters, it passes a received TC message of one cluster to the neighbouring nodes of other clusters. This enables nearby nodes of different clusters to communicate without directly following the clustering hierarchy which in turn decreases communication delay and reduces the load on the clusterheads.

For sending data to outside clusters, the topology hierarchy is followed. Here is an example where *Node 1* in Figure 6 wants to send data to *Node 10*. *Node 1* is a member of cluster *C1.A* and *Node 2* is a member of cluster *C1.E*. Through TC and HELLO messages *Node 1* knows that *Node 10* is not located within its cluster. So it sends the data to its clusterhead *A*. *A* does not recognise *Node 10* to be located within its cluster and therefore forwards the data to its clusterhead *B*.

Table 1. Comparison of various pro-active routing protocols

	WCC	*WTC*	*RS*	*Frequency of updates*	*Critical Nodes*	*HM*	*Advantages*	*Disadvantages*
DSDV	O*(N)*	O*(D)*	F	Periodic and on-demand	No	Yes	Loop free, simple; Computationally efficient	Excessive communication overhead; Slow convergence; Tendency to create routing loops in large networks
WRP	O*(N)*	O*(h)*	F	Periodic and on-demand	No	Yes	Loop free; Lower WTC than DSDV	Does not allow nodes to enter sleep mode
MMWN	O*(m+s)*	O*(2D)*	H	On-demand	Location Manager	No	Low WCC and WTC	Complicated mobility management and cluster maintenance
CGSR	O*(N)*	O*(D)*	H	Periodic	Clusterhead	No	Lower routing overhead than DSDV & WRP; Simpler addressing scheme compared to MMWN	Higher time complexity than DSDV and WRP for a link failure involving clusterheads
GSR	O*(N)*	O*(D)*	F	Periodic	No	No	Requires less number of update messages than a normal link-state algorithm	Update messages get larger if node density and network size increase
DREAM	O*(N)*	O*(D)*	F	On-demand	No	No	Low routing overhead	Requires GPS
STAR	O*(N)*	O*(D)*	F	On-demand	No	No	Minimises the number of routing update packets disseminated in the network	May not provide optimum routes to destinations; Significant memory and processing overheads for large and highly mobile MANETs
HSR	O*(n*l)*	O*(D)*	H	Periodic	Clusterhead	No	Requires less memory and communication overhead than any flat pro-active routing protocol	Introduces additional overhead for forming and maintaining clusters like any cluster based protocol
TBRPF	O*(N)*	O*(D)*	F	Periodic and on-demand	Parent node	Yes	Lower WCC compared to pure link-state routing	Overheads increase with node mobility and network size
FSR	O*(N)*	O*(D)*	F	Periodic	No	No	Reduces the size of update messages generated in GSR in large networks	Nodes may not have the best route to a distant destination
LANMAR	O*(N)*	O*(D)*	H	Periodic	Landmark	No	Improves routing scalability for large MANETs	Assumption of group mobility, Nodes may not have the best route to a distant destination
OLSR	O*(N)*	O*(D)*	F	Periodic	No	Yes	Reduces size of update messages and number of transmissions than a pure link-state routing protocol	Information of both 1-hop and 2-hop neighbours is required
FSLS	O*(N)*	O*(D)*	F	Periodic	No	No	Reduces control overhead required in FSR or GSR.	Nodes may not have the best route to a distant destination
HOLSR	O*(N)*	O*(D)*	H	Periodic	Clusterhead	Yes	Suitable for large heterogeneous MANETs	Information of both 1-hop and 2-hop neighbours is required; Introduces additional overhead for forming and maintaining clusters

WCC: Worst Case Communication Complexity, i.e., number of messages needed to perform an update operation in worst case; *WTC*: Worst Case Time complexity, i.e. number of steps involved to perform an update operation in worst case; *RS*: Routing Structure; *F*: Flat; *H*: Hierarchical; *HM*: HELLO Messages; *N*: Number of nodes in the network; *D*: Diameter of the network; *h*: Height of the routing tree; *n*: Average number of nodes in a cluster; *l*: number of hierarchical levels; *m*: Number of location managers in MMWN; *s*: Number of switches in MMWN.

Since *B* is located at the highest topology level, it contains information of all nodes in the network. From this information it knows that *Node 10* can be reached via *F*. So it relays the data to *F*. From *F* the data is sent to *Node 10* via *E*.

Comparisons of Pro-Active Routing Protocols

Pro-active routing protocols with flat routing structures usually incur large routing overheads in terms of communication costs and storage requirements to maintain up-to-date routing information about the whole network. Hence they may not scale well as the network size or node mobility increases. However FSR and FSLS have reduced the communication overhead by decreasing the frequency of updates for far away nodes. DREAM reduces the transmission overhead by exchanging location information rather than full or partial link-state information. OLSR reduces rebroadcasting by using multipoint relays (Abolhasan et al., 2004). Hence these flat routed protocols have better scalability potential.

The hierarchical pro-active routing protocols reduce communication and storage overhead as the network size increases since in most cases only the clusterheads are required to update their views of the entire network. However in MANETs, where group mobility is usually impossible, these protocols can introduce additional complexity and overhead for cluster formation and maintenance. Consequently these protocols may not perform better than flat pro-active routing protocols. Table 1 summarises and compares the characteristics of various pro-active routing protocols.

REACTIVE ROUTING PROTOCOLS

Unlike pro-active routing protocols, reactive routing protocols find and maintain routes when needed so that routing overheads can be reduced where the rate of topology change is very high. Route discovery usually involves flooding route request packets through the network. When a node that is a destination or has a route to the destination is reached, a route reply is sent back to the source of the request. If the links connecting the nodes are bi-directional, the reply is sent back through the path on which the route request travelled. Otherwise the reply is flooded. Thus, in the worst case the route discovery overhead grows by $O(N+M)$ when bi-directional links are available and by $O(2N)$ when only uni-directional links are possible (Abolhasan et al., 2004). Here N and M denote the total number of nodes in the network and the number of nodes in the reply path (if bi-directional links are available) respectively.

Reactive routing protocols can be classified into two groups based on the way routing information is stored at each node and carried in routing packets. These are source routing and hop-by-hop routing.

In source routing, each data packet contains a list of node addresses known as the source route that constitutes the complete path from the source to the destination. When a node wants to send data to a destination, it transmits the data packets to the first hop identified in the source route. When an intermediate node receives the packet, it simply transmits the packet to the next hop by finding it from the source route. Thus the packet propagates through the network until it reaches the destination. Source routing provides a very easy way to avoid forming loops in the network. However, the size of each packet gets bigger as the number of intermediate nodes increases for a particular source and destination pair.

On the other hand, with hop-by-hop routing, each data packet carries only the destination address and the next hop address, and each intermediate node in the routing path uses its routing table to forward the data packet to the next hop towards the destination. In this sense hop-by-hop routing is similar to pro-active routing. In this approach, each node updates its routing table when it receives updated topology information and forwards

the data packets over fresher and better routes. Hence routes can be adapted to the dynamically changing topologies of mobile ad-hoc networks. The disadvantage of the hop-by-hop routing over source routing is that each intermediate node has to store and maintain routing information for each active route and may require sending periodic beaconing messages to its neighbours to be aware of its neighbourhood.

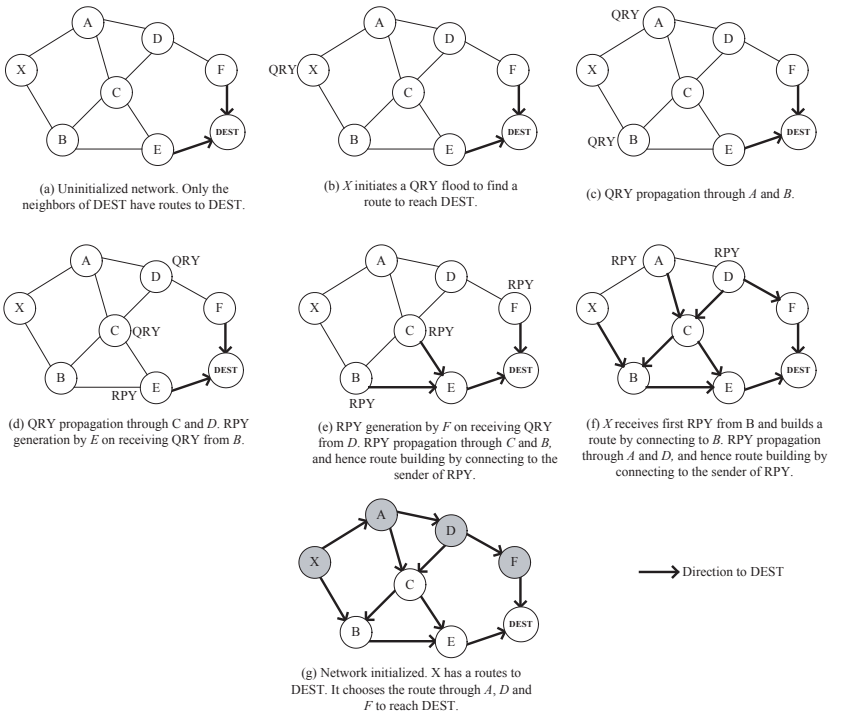
A variety of reactive protocols have been proposed based on these strategies. The rest of this section describes and compares a number of such protocols.

Light-Weight Mobile Routing (LMR)

LMR (Corson & Ephremides, 1995) maintains multiple routes to reach each destination. This feature increases the reliability of LMR since whenever a route to a particular destination fails the next available route to the destination can be used without initiating a new route construction procedure. It uses sequence numbers and inter-nodal coordination to avoid long-term loops.

Each node maintains a list of its available neighbours. When a source node needs to find routes to a destination, it initiates a route construction phase by broadcasting a query (QRY) packet to its neighbours. The QRY packet contains the address of the source, the address of the destination, a monotonically increasing sequence number maintained for each destination by the source, and the address of transmitter which is updated at each intermediate node as the QRY packet propagates through the network. The triplet <address of the source, address of the

Figure 7. Route construction using QRY and RPY packets in LMR



(a) Uninitialized network. Only the neighbors of DEST have routes to DEST.

(b) *X* initiates a QRY flood to find a route to reach DEST.

(c) QRY propagation through *A* and *B*.

(d) QRY propagation through C and *D*. RPY generation by *E* on receiving QRY from *B*.

(e) RPY generation by *F* on receiving QRY from *D*. RPY propagation through *C* and *B*, and hence route building by connecting to the sender of RPY.

(f) *X* receives first RPY from B and builds a route by connecting to *B*. RPY propagation through *A* and *D*, and hence route building by connecting to the sender of RPY.

(g) Network initialized. X has a routes to DEST. It chooses the route through *A*, *D* and *F* to reach DEST.

destination, sequence number> uniquely identifies a QRY from other queries and allows a node to remember if it has previously received the QRY. When a node receives a QRY, it rebroadcasts it to its neighbours provided that it has not received this QRY before. Thus the QRY propagates through the network and eventually reaches a node that has a route to the destination (e.g., a neighbour of the destination). This process has been illustrated in Figure 7(a)-(d).

A reply (RPY) packet is broadcast by a node which has a route to the destination, in response to the QRY packet. The RPY contains the addresses of the destination and the transmitter. The RPY is flooded back to the source in the same manner as the QRY packet with the exception that the propagation of RPY forms a directed acyclic graph that is rooted at the destination and pointed towards the origin of the RPY. Figures 7(d)-(g) illustrate this process.

When a node loses its last route to a destination due to an adjacent link failure, it enters into the route maintenance phase. If routes from other source nodes for the destination do not pass through this node, the node may enter the route construction phase if it needs to find new routes to the destination. Otherwise the node broadcasts a failure query (FQ) packet to the nodes between itself and the source node in order to inform them of the link failure and at the same time ask them if they have alternate routes to the destination. When a node receives a FQ over a link, it erases the routes containing the link. If it has any alternate route, it broadcasts an RPY. It rebroadcasts the FQ if and only if it does not have any alternate route to the destination.

LMR requires reliable delivery of its control packets. This may be an unreasonable requirement for highly dynamic networks. If reliability is not guaranteed, the protocol can suffer from temporary routing loops or may provide invalid routes temporarily in the partitioned portion of a network (Marina & Das, 2003; Park & Corson, 1997).

Dynamic Source Routing (DSR)

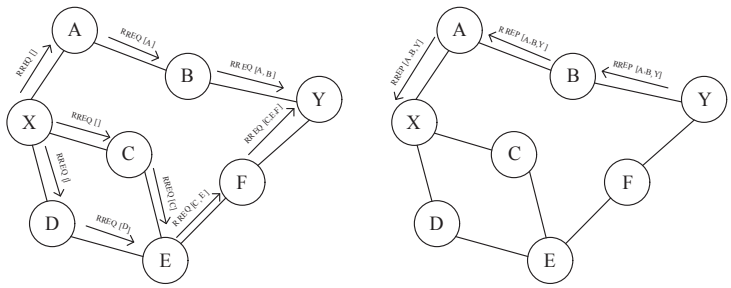
DSR (Johnson & Maltz, 1996) is based on the concept of source routing. Each node in DSR is required to maintain a route cache that contains the source routes to the destinations the node has learned recently. An entry in the route cache is deleted when it reaches its timeout.

When a source node needs to send a data packet to a destination node, it searches its route cache to determine if it already has a route to the destination. If there is a route to the destination, it uses the route to send the data packet. Otherwise it initiates a route discovery process by broadcasting a route request (RREQ) packet to its neighbours. The RREQ contains the address of the source, the address of the destination, a request id, and a route record. The request id is a sequence number maintained locally by the source node. The route record is the addresses of the intermediate nodes through which the RREQ will pass to reach the destination. At the source the route record does not contain anything.

When a node receives a copy of the RREQ, it checks the <source address, request id> pair in its list of recently seen route requests. If there is a match or the route record contains the address of the node, the RREQ is dropped. Otherwise the node checks whether it is the destination or contains a route to the destination. If it is not the destination or does not have a route to the destination it appends its address to the route record and rebroadcasts the RREQ to its neighbours. A copy of the RREQ thus propagates through the network until it reaches the destination or a node that has a route to the destination. Figure 8(a) illustrates the formation of a route record as it propagates through the network towards the destination.

A route reply (RREP) is generated when the RREQ reaches either the destination or an intermediate node that contains a route to the destination. A node does not generate more than one RREP for a particular source and destina-

Figure 8. Formation of route record through propagation of RREQ and RREP in DSR



X : Source, Y: Destination, A-F: Other Nodes

(a) Formation of route record in DSR as RREQ propagates through the network.

(b) Propagation of RREP through the network using route record.

tion pair. If the node generating the RREP is the destination itself, it copies the route record from the RREQ to the RREP. If the responding node is an intermediate node, it appends its cached route to the incomplete route record and puts the complete route record in the RREP.

To send the RREP to the source, the responding node must have a route to the source in its route cache. If the node has a route entry in its route cache for the source node, it may use this route to unicast the RREP in the same way as source routing. Otherwise the responding node may reverse the route in the route record from the RREQ and use this route to send the RREP to the source. Figure 8(b) shows the propagation of a RREP using this latter scheme. This scheme, however, will work if the neighbouring nodes, listed in the route record, can communicate equally well in both directions. As an alternative approach, the responding node can piggyback the RREP on a RREQ generated to find a route to the source.

Nodes can operate in promiscuous mode to extract route records used in the overheard packets transmitted by neighbouring nodes and thus update entries in their route caches without actually participating in any route discovery process.

If a node receives a packet in promiscuous mode and finds out its address in the unprocessed part of the source route multiple hops away from the current sender, it sends a gratuitous reply message to the packet's sender informing it that the packet can be forwarded to it directly bypassing the additional hops.

Each node monitors the operation of each route it is currently using through a route maintenance module. If it cannot send a packet to a neighbour, it declares the corresponding link to be broken and sends a route error (RERR) packet to the source of the associated route. The RERR contains the address of the node that detected the error and the address of the neighbour (i.e., the hop in error) to which the node failed to send packets. When the RERR is received, the hop in error is removed from the route cache and all routes that contain this hop are truncated at that point.

Caching route entries can be beneficial for networks with low mobility. In highly mobile or large networks aggressive use of route caching

and lack of an efficient mechanism to purge stale routes can lead to problems like stale caches and relay storm. As a result network performance can be degraded (Marina & Das, 2001a). Moreover, the use of a source route in each packet consumes extra channel bandwidth. The size of each packet gets larger as the size of the network increases. These problems, however, have been addressed in Hu and Johnson (2000, 2001) and Marina and Das (2001a).

Associativity-Based Routing (ABR)

ABR (Toh, 1996, 1997) uses the concept of source routing similar to DSR, but selects routes based on association stability, that is, connection stability, of nodes. Routes selected in this manner are likely to be long lived, resulting in requiring fewer route reconstructions and less route control traffic. However, routes selected in this way may not be the shortest in terms of the number of intermediate nodes.

Each node generates periodic beacons to notify others of its existence. When a node receives a beacon, it increments its associativity tick with respect to the neighbour from which it received the beacon. If a node observes low associativity ticks with its neighbours, it is said to exhibit a high state of mobility, that is, low association stability. On the other hand, if a node has high associativity ticks with its neighbours, it can be considered to be in a high stability state and selected for routing.

When a source needs to find a route to a destination, it broadcasts a BQ (broadcast query) packet to its neighbours. When a node, other than the destination, receives a BQ, it checks if it has previously seen the BQ. If so, the node drops the BQ. Otherwise it appends its address and associativity ticks to the BQ, and then rebroadcasts the updated BQ to its neighbours. The next succeeding node erases the associativity tick entries from the BQ that were appended by the upstream neighbour and retains only the entry concerned with itself and its upstream neighbour. Then it rebroadcasts the BQ to its neighbours after appending its address and associativity ticks to it. In this manner the BQ propagates through the network and eventually reaches the destination.

The destination, after receiving multiple BQs, selects the best route by examining the associativity ticks along each of the routes. If multiple routes have the same overall degree of association stability, the route with minimum number of intermediate nodes is selected. Once a route has been selected, the destination sends a REPLY packet back to the source along the selected route. As the REPLY passes through each intermediate node, it marks the embedded route in the REPLY packet as valid and regards all other possible routes to the destination as invalid in order to avoid duplicated packets arriving at the destination.

The route maintenance phase of ABR consists of new route discovery, partial route discovery, invalid route erasure, and valid route updates depending on node mobility along the route.

If a source node moves away from its downstream neighbour (i.e., the next hop neighbour towards the destination), it initiates a new route discovery.

When a destination moves, its immediate upstream neighbour (i.e., the next hop neighbour towards the source) erases its route to the destination. Then the upstream neighbour broadcasts a localised query (LQ [H]) packet, where H refers to the hop count from the upstream node to the destination, to find out if the destination is still reachable. If the destination receives the LQ packet, it selects the best partial route and responds with a REPLY packet. If the node, which initially generated the LQ [H], times out, it notifies the immediate upstream neighbour to erase the invalid route and invoke a LQ [H] process. If this process backtracks more than halfway towards the source, the source is notified to initiate a new route discovery phase. If an intermediate node moves, a similar process is invoked at other intermediate nodes between the point of failure

and the source. Additionally, the immediate downstream neighbour propagates a route delete message towards the destination in order to delete corresponding route entries from the route tables of all the subsequent downstream nodes.

When a route for a particular destination is no longer needed, the source broadcasts a route delete message to its neighbours. A node, receiving the route delete message, deletes the corresponding route entries from its routing table and rebroadcasts the route delete message to its neighbours. Thus a route delete message is propagated through the network until it is received by a node that does not have any entry in its routing table for the destination corresponding to the route delete message.

ABR is suitable for small MANETs. The beaconing interval should be short enough to be able to adapt quickly to spatial, temporal, and connectivity states of the neighbouring nodes (Royer & Toh, 1999). This requirement may result in extra bandwidth and power consumption.

Signal Stability-Based Adaptive (SSA) Routing

SSA (Dube, Rais, Kuang-Yeh, & Tripathi, 1997) selects routes based on signal stability, that is, the combination of signal strength and location stability, rather than using association stability as used in ABR. Like ABR, routes selected in SSA may not be shortest in terms of the number of intermediate nodes.

Each node sends out a link layer beacon to its neighbours periodically and maintains a signal stability table where each row corresponds to the signal strength and location stability of each neighbour. When a node receives a beacon, it measures the signal strength at which the beacon was received and updates the corresponding entry in its signal stability table. If the node receives a certain number of strong beacons from a neighbour for a predefined period, it classifies the neighbour as strongly connected. Otherwise the neighbour is regarded as weakly connected.

Each node maintains a routing table where each entry contains the next hop address for each reachable destination. When a source needs to send a packet to a destination, it looks up the destination in its routing table. If there is an entry, the data packet is forwarded using the hop-by-hop strategy. Otherwise the node initiates a route discovery process using a source routing strategy. Route search packets are forwarded to the next hop only if they are received from strongly connected neighbours and have not been previously processed. The first route search packet that arrives at the destination is considered to be the one arriving over the shortest or least congested path. The destination responds to the route search packet by sending a route reply packet to the source. When an intermediate node detects one of its neighbours is not available any more, for example, has moved out of its transmission range or shut down, it sends an error message to the source indicating which link has failed. The source then sends an erase message to erase the invalid route and initiates a new route discovery process to find a new route to the destination.

In SSA, intermediate nodes cannot reply to a route search packet. This incurs longer delays than DSR before a route can be found. Unlike ABR, SSA does not have any route repair mechanism at the point where link failure occurs. The source has to be notified to perform the route reconstruction. Therefore SSA may incur additional delays before a broken route is re-established (Abolhasan et al., 2004).

Temporally Ordered Routing Algorithm (TORA)

TORA (Park & Corson, 1997) is an improved variant of LMR. Like LMR it uses a directed acyclic graph, rooted at a destination, to represent multiple routes for a source and destination pair.

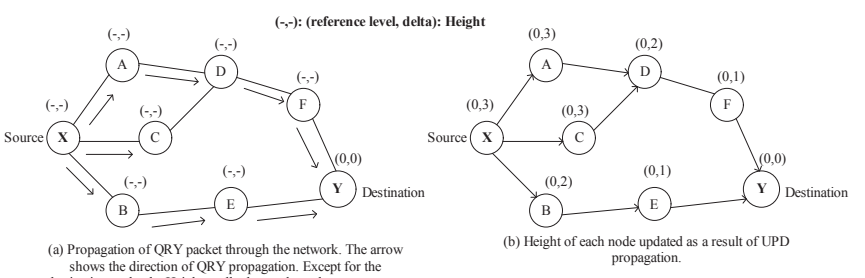
However, unlike LMR, it restricts the propagation of control messages to a very small set of nodes near the occurrence of a topological change by using the concept of link reversal proposed by Gafni and Bertsekas (1981). When a link in a directed acyclic graph breaks, the link reversal method can transform the distorted graph in finite time so that the destination becomes the only node with no outgoing links. TORA uses time stamps and internodal coordination to avoid long-term loops.

The process of route creation in TORA is similar to LMR with few exceptions. TORA uses query (QRY) and update (UPD) packets for creating new routes. The UPD packet is known as the reply packet in LMR. Unlike LMR, TORA assumes that nodes have synchronised clocks and use a height metric to establish a directed acyclic graph for each destination. The height of a node is defined by two parameters: a reference level and a delta with respect to the reference level. The height of the destination is always zero, that is, the values of the reference level and delta are both zero. The heights of other intermediate nodes increase by 1 towards the source node. This is accomplished by increasing the value of delta. Unlike LMR, a node in TORA may process multiple UPD packets for the same source and destination pair if the most recent UPD packet gives the node a lesser height. For example, in Figure 9(b), the source *X* may have received an UPD from node *A* or node *C* before the UPD from node *B*, but since the UPD from node *B* gives it lesser height it retains this height.

When a node loses its last downstream link (i.e., the link directed from this node to one of its neighbours) for a particular destination as a result of link failure, the node selects a new height so that the new height becomes a global maximum. This can be accomplished by defining a new reference level and a new delta, such as increasing the value of the current reference level and assigning zero to delta. This action results in link reversals, which may cause other nodes to lose their last downstream links for the destination. Such nodes also select a new height and perform link reversal with respect to their neighbours. Thus the new height is propagated outward from the point of the original failure and gets updated. This propagation continues only through the nodes, which have lost all the routes to the destination. As a result, the propagation of control messages becomes restricted to a very small set of nodes near the occurrence of a topological change.

If the node, which detected the link failure, receives the propagated new height, it determines that no route to the destination exists. The node then begins the process of erasing invalid routes to the destination by flooding a clear (CLR) packet throughout the network.

Figure 9. Route creation in TORA using QRY and UPD propagation



(a) Propagation of QRY packet through the network. The arrow shows the direction of QRY propagation. Except for the destination node, the Height at all other nodes values are empty.

(b) Height of each node updated as a result of UPD propagation.

TORA can falsely detect partitions because it only considers links known from previous route discovery; links that can come up later are ignored though they can be used to join the partitions (Marina & Das, 2003). It requires reliable and in-order delivery of route control packets. These requirements can degrade the network performance to such an extent that the advantage of having multiple routes can be undermined (Broch, Maltz, Hu, & Jecheva, 1998; Das, Castaneda, & Yan, 2000). Moreover it can create short-term routing loops due to the nature of its link reversal technique.

Location-Aided Routing (LAR)

LAR (Ko & Vaidya, 1998) is a flood based routing algorithm, like DSR, that uses location information in order to reduce route search space and thereby minimises route control traffic. It assumes that each node obtains its location information using a GPS (global positioning system).

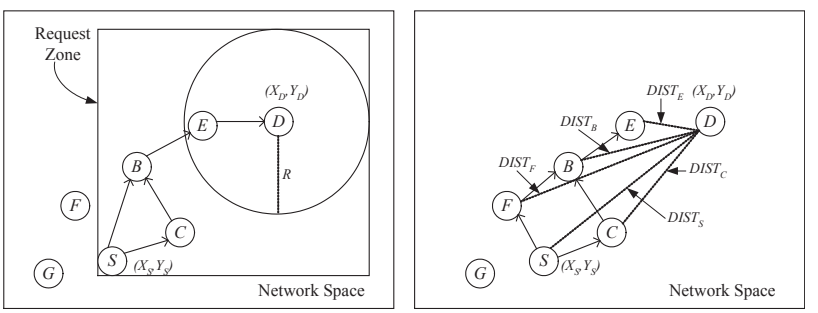
In LAR a node forwards route request packets only to the nodes that reside inside the route search space (also referred to as the request zone). Any node outside the request zone ignores such packets. If route is not discovered within a suitable timeout period, the request zone is expanded. Two schemes have been considered in LAR to determine a request zone.

Consider a node source *S*, in Figure 10(a) and Figure 10(b), wants to find a route to destination node *D*.

In *Scheme 1* (Figure 10(a)) at time t_1, *S* determines the expected zone of *D* from *D*'s location information recorded at time t_0. If where *v* is the average speed of *D*, the expected zone of *D* from *S*'s viewpoint is the circular region with radius $R = v(t_1-t_0)$, that is, *S* can assume that *D* will be in any location within that circle during the interval (t_1-t_0). Now LAR defines the request zone by the smallest rectangle that includes current location of *S* and the expected zone of *D*. *S* includes the four coordinates, that is, (X_S, Y_S) and (X_D, Y_D), with the route request packet. A node, such as *F* and *G*, ignores a route request packet if it does not belong to the rectangle defined by the four corners. If the location cannot be precisely detected then the radius of an expected zone becomes $e+v(t_1-t_0)$, where *e* denotes the maximum error in the coordinate estimated by the source node.

In *Scheme 2* (Figure 10(b)) node *S* includes $DIST_s$ and (X_D, Y_D) in each route request packet. Here (X_D, Y_D) is the location of node *D* and $DIST_s$ denotes the distance between *S*'s current location and (X_D, Y_D) recorded by *S* at time t_0. When a node *F* receives the request from *S*, it calculates its distance $DIST_F$ from (X_D, Y_D). If $DIST_S + \delta \geq DIST_F$, where δ is an error margin, *F* forwards the route request packet after updating $DIST_S$

Figure 10. Routing schemes of LAR



(a) Scheme 1 (b) Scheme 2

with $DIST_F$. If $DIST_S + \delta < DIST_F$, F discards the route request.

Though LAR aims to reduce the number of control packets in a network, it may generate control packets similar to a flooding algorithm in a highly mobile network.

Ad-Hoc On-Demand Distance Vector (AODV) Routing

AODV (Perkins & Royer, 1999) routing protocol minimises the number of required broadcasts of DSDV by creating routes on a demand basis. It uses sequence numbers to avoid long-term loops.

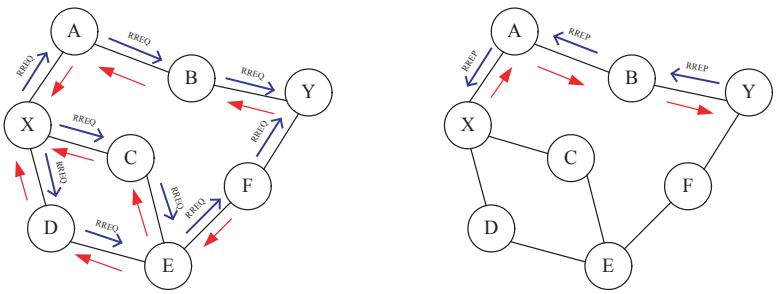
AODV requires each node to maintain a list of its active neighbours by sending periodic HELLO packets or by listening to data transmissions of neighbouring nodes in promiscuous mode.

When a source node needs to send a data packet to a destination node and does not contain any route to the destination, it initiates a path discovery process to find a route to the destination. For this purpose every node maintains two monotonically increasing counters: a sequence number and a broadcast id. The source broadcasts a route request (RREQ) packet to its neighbours containing the address of the source (i.e., the address of itself), its sequence number, its broadcast id, the address of the destination, its last known sequence number of the destination, and a hop count with a value of zero. The pair <address of the source, broadcast id> uniquely identifies a RREQ. The destination sequence number is used to determine the relative freshness of two pieces of routing information generated by two nodes for the same destination, that is, the packet with the highest destination sequence number is more recent. The broadcast id is incremented by the source every time it broadcasts a RREQ.

When a node receives a RREQ, it checks if it has received a RREQ with the same <address of the source, broadcast id> pair before. If there is a match, the node drops the RREQ and thereby limits the number of broadcast packets. Otherwise it accepts the RREQ for further processing.

During further processing the node checks if it has a route to the destination with a destination sequence number greater than the destination sequence number of the received RREQ. If such a record is found, the node can respond to the RREQ by sending a route reply (RREP) packet to the source. Otherwise it rebroadcasts the RREQ

Figure 11. Creation of reverse and forward paths in AODV



X : Source, Y: Destination, A-F: Other Nodes

(a) Propagation of RREQ and reverse path setup (shown in red coloured arrows) in AODV.

(b) Propagation of RREP and forward path setup (shown in red coloured arrows) in AODV.

to its neighbours after increasing the value of the hop count. This process repeats and eventually a RREQ is assumed to arrive at a node that is either the destination itself or has a current route to the destination.

As a RREQ travels from the source to the destination, it sets up a reverse path from the destination to the source through which the corresponding RREP will travel. To set up a reverse path, a node records in its routing table the address of the neighbour from which it received the first copy of the RREQ.

As the first copy of an RREP is sent to the source along a reverse path, each node along the path sets up a forward route entry in its routing table by recording the address of the neighbour from which it has received the RREP. This forward route entry will be used to forward the data packet from the source to the destination. If a node receives a further RREP, it updates its routing information and forwards the RREP only if the RREP contains a greater destination sequence number than the previous RREP or the same destination sequence number with a smaller hop count value. This rule ensures that the number of RREPs propagated towards the source is minimised, avoids forming loops, and restricts the source from learning multiple routes to the destination. If a route entry is not used for a certain period, it is deleted. The route discovery mechanism has been depicted in Figure 11.

When a node detects a link failure (e.g., due to node movement) it sends a link failure message to the source of the associated route along the corresponding reverse path. All the nodes receiving the link failure message erase the associated entries in their routing tables. The source may also choose to reinitiate a route discovery for the destination if required.

Since AODV does not provide any localised route repair mechanism, it introduces extra delays and consumes more bandwidth as the size of the network increases (Abolhasan et al., 2004).

Relative Distance Micro-Discovery Ad-Hoc Routing (RDMAR)

RDMAR (Aggelou & Tafazolli, 1999) minimises routing overheads by localizing query flooding into a limited area. It uses the concept of sequence numbering, similar to AODV, to prevent forming long-term loops.

If a source node does not contain a feasible route to a destination, it initiates a route discovery process. During this process the source node refers to its routing table in order to find information on its previous relative distance with the destination and the time elapsed since it last received routing information for the destination. With this information and assuming a moderate velocity and a moderate transmission range for the destination, the source node estimates the new relative distance to the destination in terms of the number of hops. The source then updates its routing table with this new relative distance. It inserts the new relative distance in the time-to-live (TTL) field of the route request (RREQ) packet so that nodes outside the range of the TTL do not process the RREQ.

The handling of RREQ and RREP (route reply) by intermediate nodes is similar to AODV. However, RDMAR allows only the destination to send an RREP packet in response to the RREQ that arrives first. It is claimed that preventing any intermediate node, which may have a route to the destination, from sending an RREP reduces the possibility of nodes receiving stale routing information.

When an intermediate node detects a link failure for a destination, it initiates a route discovery procedure by itself if it is located near to the destination. Otherwise it sends a failure notification to the source.

Each node, upon receiving the failure notification, removes the next hop information associated with the destination from its routing table. The node can then initiate a new route discovery

procedure if it has kept a copy of the data packet for which the link failure was generated. Otherwise it forwards the failure notification to the neighbour towards the source. When the source receives the failure notification, it initiates a new route discovery procedure if it still needs a route to the destination.

If a source and destination pair does not have any prior communication, RDMAR behaves like a pure flooding algorithm.

Cluster-Based Routing Protocol (CBRP)

CBRP (Jiang, Li, & Tay, 1999) is a hierarchical on-demand routing algorithm that uses source routing, similar to DSR, to avoid forming loops and route packets. Like other hierarchical routing algorithms, CBRP aims to scale well with network size. It can best perform in MANETs where nodes in each cluster move together (Abolhasan et al., 2004).

CBRP groups the nodes in a network into several clusters. Each cluster has a clusterhead that coordinates data transmission within the cluster and with other clusters. When a node is switched on, it sets its state to undecided, starts a timer, and broadcasts a HELLO message. If a clusterhead gets this HELLO message it responds immediately with another HELLO message. If the undecided node gets this message within the timeout period, it sets its state to member. If the undecided node times out, but detects some bi-directional links with some neighbours, it declares itself as the clusterhead. Otherwise, it remains in the undecided state and repeats this process to become either a clusterhead or a member.

Each node maintains a neighbour table. Each entry in the neighbour table contains information about each neighbour, that is, the status of the associated link (uni-directional or bi-directional) and the state of the neighbour (clusterhead or member). A clusterhead keeps a list of its members. It also maintains a cluster adjacency table where each entry contains information about each neighbouring cluster, that is, the gateway through which the cluster can be reached and the clusterhead of the cluster.

When a source wants to send data to a destination, it broadcasts route request packets to its neighbourhood. When a clusterhead receives the request, it checks if the destination is located within its cluster. If the destination is available within its cluster, it forwards the request to the destination. Otherwise it rebroadcasts the request to all its neighbouring clusterheads. This process continues until the destination receives the request packet and responds with a reply. The propagation of route request and route reply is similar to that of DSR. If the source does not receive a reply within a timeout period, it backs off exponentially before sending a route request again.

CBRP uses route shortening to reduce the length of a route. To do so a node receiving a source route packet tries to find the farthest node in the route that is its neighbour. This situation can arise due to a topology change. If such a neighbour is found, the node sends the packet to that neighbour directly.

While forwarding a data packet, if a node detects a link failure, it sends an error message to the source and also tries to forward the packet through a local repair mechanism. In the local repair mechanism, the node checks if the next hop or the hop after the next hop can be reached through any of its neighbours. If the node succeeds, the data packet can be delivered to the destination over the repaired path.

Like other hierarchical routing protocols, cluster formation and maintenance involve additional communication and processing overhead. CBRP may provide invalid routes temporarily as a node moves from one cluster to another (Abolhasan et al., 2004).

Multi-Path Source Routing (MSR)

MSR (Wang, Zhang, Shu, Dong, & Yang, 2001) is an extension of DSR. It tries to improve end-to-end delay, average queue size, network congestion, and path fault tolerance by employing the multi-path finding capability of DSR.

MSR allows the source to receive multiple route reply packets in a single route discovery phase. Each route discovered is stored in the route cache with a unique route index so that multiple routes for a particular destination can be distinguished properly. Each route with index *i* for a particular destination *j* is assigned a weight W_i^j based on the round trip delay of that route. The weight is measured in terms of the number of packets to be sent consecutively on the same route and is used for distributing load among multiple routes for a particular destination. If $d_{\max}^j$ is the maximum delay of all the routes to destination *j*, d_i^j is the delay of the route with index *i* for destination *j*, *U* is a bound to ensure that W_i^j does not become too large and *R* is a factor to control switching frequency between routes, then W_i^j is calculated as follows:

$$W_i^j = Min_j\left(\left\lceil \frac{d_{\max}^j}{d_i^j} \right\rceil, U\right) \times R$$

MSR sends periodic probe packets along the active routes to measure their round trip delays as well as to test their validity.

Ad-Hoc On-Demand Multi-Path Distance Vector (AOMDV) Routing

AOMDV (Marina & Das, 2001b, 2003) extends AODV to support multi-path routing in mobile ad-hoc networks. It adds some extra fields in routing tables and control packets, and requires few new rules to be followed during a route discovery phase in order to compute loop-free and link-disjoint multiple routes. Link-disjoint routes do not contain any common link among the multiple routes between a source and destination pair.

Every node maintains a variable called the advertised hop count for each destination in order to achieve loop-freedom. This variable is added in each RREQ (route request) or RREP (route reply) and in the routing table with the usual fields that are used for AODV. When a node initiates a RREQ or RREP with a particular destination sequence number, its advertised hop count field is set to the length of the longest available path to the destination expressed in terms of the number of hops. The advertised hop count remains unchanged until the associated destination sequence number is changed. The rules for loop-freedom state that if a node receives a RREQ (RREP) for a particular destination with a destination sequence number: (a) higher than the one stored in its routing table, it should update its routing information with the information obtained from the received RREQ (RREP); (b) equal to the one stored in its routing table, it can re-send the received RREQ (RREP) if the advertised hop count in the RREQ (RREP) is greater than the corresponding value in its routing table; and (c) equal to the one stored in its routing table, it can update its routing table with the information contained in the received RREQ (RREP) if the advertised hop count in the RREQ (RREP) is less than the corresponding value in its routing table.

For link-disjointness, each node maintains a route list in its routing table for each destination. A route list for a particular destination contains entries with next hop, last hop, and hop count information for the destination. The next hop refers to a downstream neighbour via which the destination can be reached. The last hop is the node immediately preceding the destination. The hop count measures the distance from the node to the destination via the associated next and

last hops. If a node can ensure that all paths to a destination from itself differ in their next and last hops, then link-disjointness among those paths can be achieved. AOMDV uses this observation to ensure link-disjointness among multiple routes for the same source and destination pair. For this purpose AOMDV adds a last hop field in each RREQ and RREP.

During route discovery AOMDV allows all copies of an RREQ to be examined for potential alternate reverse paths. When an intermediate node receives an RREQ, it creates a reverse path if the RREQ satisfies the rules for loop-freedom and link-disjointness. It then checks if it has one or more valid next hop entries for the destination. If such an entry is found, the node generates an RREP and sends it back to the source along the reverse path. If the intermediate node does not find such an entry and has not previously broadcast any copy for this RREQ, it rebroadcasts the RREQ.

When the destination receives RREQ copies, it follows the same rules for creating reverse paths. However, unlike intermediate nodes, it generates an RREP for every copy of RREQ that arrives via a loop-free path. This feature increases the possibility of finding more disjoint routes.

Multiple-route ad-hoc on-demand distance vector (MRAODV) routing (Higaki & Umeshima, 2004) is another extension of AODV that has similar aims to AOMDV. During the propagation of route request packets in AOMDV the links through which route request packets arrive are stored as backward links in order to establish potential reverse paths for the route replies. Unlike AOMDV, MRAODV switches the direction of some of these backward links so that the number of multiple routes can be increased. However this method of link reversal may not produce link-disjoint routes. Moreover, it is unclear if this method can preserve loop-freedom.

Ant-Colony Based Routing Algorithm (ARA)

ARA (Gunes, Sorges, & Bouazizi, 2002) adopts the food searching behavior of ants to find routes. When ants search for food, they start from their nest and walk towards the food. While walking they leave behind a transient trail by depositing pheromone, a substance that ants can smell. The concentration of pheromone on a certain route indicates its usage and allows other ants to follow the most commonly used route. In the course of time the concentration of pheromone is reduced due to diffusion. Like AODV, ARA uses sequence numbers to avoid forming loops. However, unlike AODV, ARA can find multiple routes between a source and destination pair.

During route discovery, a FANT (forward ant) packet is propagated through the network similar to RREQ in AODV. When a node receives a FANT for the first time, it calculates a pheromone value depending on the number of hops the FANT has traveled to reach the node. It creates an entry in its routing table with the calculated pheromone value, the address of the neighbour from which the FANT was received, and the address of the source from which the FANT originated. This entry in the routing table creates a pheromone track towards the source. Then the node forwards the FANT to its neighbours. Sequence numbers, similar to AODV, are used to avoid duplicate FANTs and prevent forming loops.

Once a FANT reaches the destination, the destination creates a BANT (backward ant) from the extracted information of the FANT and returns the BANT to the source. The BANT performs a similar task to the FANT, that is, establishing a pheromone track towards the destination. Unlike, the FANTs, the propagation of the BANTs enables ARA to establish multiple paths between a source and destination pair. When the source receives a BANT, a path is established between the source

Table 2. Comparison of various reactive routing protocols

	WCC [RD]	*WCC [RM]*	*WTC [RD]*	*WTC [RM]*	*RS*	*MR*	*PB*	*Advantages*	*Disadvantages*
LMR	O*(2N)*	O*(2a)*	O*(2D)*	O*(2D)*	F	Yes	No	Multiple routes	Requires reliable delivery of routing control packets; Can suffer from temporary routing loops
DSR	O*(2N)*	O*(2N)*	O*(2D)*	O*(2D)*	F	Yes	No	Intermediate nodes do not store route information; Can provide multiple paths	Stale caches and relay storm problems may arise in large and highly mobile MANETs; Additional communication overhead due to source routing
ABR	O*(N+r)*	O*(a+r)*	O*(D+c)*	O*(b+c)*	F	No	Yes	Stable routes; Localised route repair mechanism	Suitable for small MANETs; Frequent beacons may result in extra bandwidth and power consumptions;
SSA	O*(N+r)*	O*(a+r)*	O*(D+c)*	O*(b+c)*	F	No	Yes	Stable routes	Introduces more delays than DSR to find routes; Does not have any localised route repair mechanism
TORA	O*(2N)*	O*(2a)*	O*(2D)*	O*(2D)*	F	Yes	No	Localised route maintenance	Can falsely detect partitions; Requires reliable and in-order delivery of route control packets; Temporary routing loops
LAR	O*(2e)*	O*(2e)*	O*(2d)*	O*(2d)*	F	Yes	No	Limits the propagation of routing control packets	Flooding is used if no location information is available; Behaves like a flooding algorithm in highly mobile MANETs
AODV	O*(2N)*	O*(2N)*	O*(2D)*	O*(2D)*	F	No	Yes*	Adaptable to highly dynamic topologies; Multicast routing capability	Requires HELLO messages; Does not support multiple routes; Intermediate nodes need to store routing information; May not scale well with network size
RDMAR	O*(2e)*	O*(2e)*	O*(2d)*	O*(2d)*	F	No	No	Limits the propagation of routing control packets	Flooding is used if nodes do not have any prior communication; Suited for MANETs having low to moderate topological changes
CBRP	O*(2m)*	O*(2a)*	O*(2D)*	O*(2b)*	H	No	No	Reduces communication; Localised route maintenance	Introduces additional overhead for forming and maintaining clusters; Temporary routing loops
MSR	O*(2N)*	O*(2N)*	O*(2D)*	O*(2D)*	F	Yes	Yes#	Multi-path routing and load balancing	Requires periodic probe packets to gather information
AOMDV	O*(2N)*	O*(2N)*	O*(2D)*	O*(2D)*	F	Yes	Yes*	Link-disjoint multi-path routing	Requires periodic HELLO messages
MRAODV	O*(2N)*	O*(2N)*	O*(2D)*	O*(2D)*	F	Yes	Yes*	May provide more multiple paths than AOMDV	Requires periodic HELLO messages; May not produce link-disjoint routes; May not preserve loop-freedom
ARA	O*(N+r)*	O*(a+r)*	O*(D+c)*	O*(D+c)*	F	Yes	No	Multiple routes; Localised route maintenance	Route discovery is based on flooding

WCC: Worst Case Communication Complexity, i.e. number of messages needed to perform a route discover or an update operation in worst case; *WTC*: Worst Case Time complexity, i.e. number of steps involved to perform a route discovery or an update operation in worst case; *RD*: Route Discovery; *RM*: Route Maintenance; *RS*: Routing Structure; *F*: Flat; *H*: Hierarchical; *MR*: Multiple Routes; *PB*: Periodic Beacons; *N*: Number of nodes in the network; *D*: Diameter of the network; *a*: Number of affected nodes; *b*: Diameter of the affected area; *c*: Diameter of the directed path of RREP, BANT; *d*: Diameter of the localised region; e: Number of nodes in the localised region; *r*: Number of nodes in the route reply path; *m*: Number of clusters in CBRP; *: Beacons in terms of HELLO Messages; #: Sends periodic probe packets along active routes;

and the destination through which data packets can be sent.

Each time a node relays a data packet to the next hop toward the destination, it increases the pheromone value of the corresponding entry in its routing table. If a link of a route is not used, its pheromone value decreases over time.

While forwarding a data packet, if a node detects a link failure, it first checks its routing table to find an alternate route to the destination of that data packet. If no route is found, it requests its neighbours to find a route to the destination. If the neighbours fail to find an entry in their routing tables for the destination, the request backtracks until it reaches the source node. The source then can initiate a new route discovery phase if needed.

Since the route discovery process is based on flooding, ARA may not scale well as the numbers of nodes and flows increase (Abolhasan et al., 2004).

Comparisons of Reactive Routing Protocols

Most of the reactive routing protocols use a flat routing structure. Nodes using flat reactive routing protocols usually flood route discovery packets through the entire network to find a feasible route to the destination. LAR and RDMAR can reduce the number of route discovery packets by limiting the search space within a calculated region. However if an estimated location of a remote node is not known a priori, these protocols behave like a pure flooding based algorithm. In ABR and SSA routing overheads are minimised by selecting stable routes. Routes selected in this manner are likely to be long lived and consequently would require fewer route reconstructions and less route control traffic. However, routes selected in this way may not be the shortest in terms of the numbers of hops. ABR, TORA, and ARA provide localised route repair mechanisms to reduce delays, and limit route control packets that could otherwise be increased if alternate routes were required to be found by the source nodes.

CBRP reduces control overhead by applying a hierarchical structure to the network, since during route discovery only the clusterheads exchange routing information (Abolhasan et al., 2004). CBRP further minimises delay and the number of control packets by providing a localised route repair mechanism. However, like most other hierarchical routing protocols, CBRP may incur excessive processing and communication overheads for cluster formation and maintenance in MANETs. Therefore CBRP is most suitable for medium-sized networks with slow to moderate mobility (Abolhasan et al., 2004). Table 2 compares the main characteristics of various reactive routing protocols.

HYBRID ROUTING PROTOCOLS

These protocols combine the features of both pro-active and reactive routing strategies to scale well with the increase in network size and node density. This is usually achieved by maintaining routes to nearby nodes using a pro-active routing strategy and determining routes to far-away nodes using a reactive routing strategy. Description and comparison of a number of such protocols are provided in the rest of this section.

Zone Routing Protocol (ZRP)

ZRP (Haas, 1997; Haas & Pearlman, 1998) utilises both pro-active and reactive routing strategies in order to gain benefits from the advantages of both types.

Each node in ZRP has a routing zone centred at itself. The radius of the zone is expressed in terms of the number of hops. Nodes within the same zone can use any pro-active routing protocol to maintain routing information. If a source

needs to send packets to a destination, it looks in its routing table to find out if the destination is within its routing zone. If so, the packet can be routed using any pro-active routing protocol.

If the source does not find an entry for the destination in its routing table, it uses a route request/route reply cycle of any reactive routing protocol to determine a route to the required destination. Each zone contains some nodes on its border, referred to as border nodes. A route request packet is propagated from one zone to another through these border nodes until is reaches the zone of the required destination.

Zone-Based Hierarchical Link-State (ZHLS) Routing

Unlike ZRP, ZHLS (Joa-Ng & Lu, 1999) divides the network into non-overlapping zones and employs a hierarchical structure to maintain routes. Unlike other hierarchical protocols, ZHLS does not require any clusterheads so avoids traffic bottlenecks, single points of failure, and complicated mobility management. It is pro-active if the destination resides within the same zone of the source. Otherwise it is reactive, since location search is employed to find the zone ID of the destination. Thus it reduces communication overheads compared to any pure reactive routing protocol such as DSR and AODV.

ZHLS defines two levels of topologies: node level and zone level. The node level topology provides the information on how the nodes are connected through physical links. If there is at least one physical link connecting two zones, a virtual link is assumed to exist between those zones. The zone level topology tells how zones are connected by these virtual links.

Initially a node knows its physical position by using a GPS and determines its zone ID by mapping its physical location to the zone map. With this zone ID, the node starts the intra-zone clustering and then the inter-zone clustering procedures to build its routing tables. To have a preprogrammed zone map may not be feasible in networks where physical boundaries of the zones are dynamic.

During intra-zone clustering, each node asynchronously broadcasts a link request to which neighbouring nodes respond with their node IDs and zone IDs. When all the link request responses are received, the node broadcasts a node LSP (link-state packet) containing the node IDs of its neighbours of the same zone and the zone IDs of the neighbours of different zones. The node LSP is propagated only within its zone. After receiving all node LSPs of the same zone, each node knows the node level topology of that zone and can use a shortest path algorithm to build its intra-zone routing table.

Nodes that receive link request responses from their neighbouring zones are called the gateway nodes. During inter-zone clustering, these gateway nodes broadcast zone LSPs throughout the network. A zone LSP contains the list of all the neighbouring zones from which it is originated. After each node receives all zone LSPs, it can build an inter-zone routing table.

When a source node wants to send data to a destination, it checks if the destination exists in its intra-zone routing table. If so, the packet can be routed to the destination like any other link-state routing protocol using the information from intra-zone routing tables of the intermediate nodes. If the destination resides in a different zone, the source sends a location request according to its inter-zone routing table. The gateway of each zone will receive this request and check if the destination exists in its intra-zone routing table. If so the gateway replies with a location response containing the zone ID of the destination. When the source receives this reply, it specifies the zone ID and the node ID of the destination in the data header and routes the data according to its inter-zone routing table.

Scalable Location Update Routing Protocol (SLURP)

Like ZLHS, SLURP (Woo & Singh, 2001) organises nodes into a number of non-overlapping regions. However it does not employ a global route discovery mechanism and thereby reduces the cost of maintaining routing information.

SLURP assigns a home region for each node in the network. It assumes that each node uses a GPS to know its current location, contains a list of IDs of other nodes in the network, and uses a one-to-many static mapping function *f*, that is, *f(Node ID)* $\rightarrow$ *Region ID*, to determine the home region of other nodes. This function is known to all nodes in the network and is assumed to generate always the same home region for a specific node ID.

Each node always informs the nodes, currently present in its home region, the identity of the region it is located in by unicasting a location update message towards its home region. Once the location update message reaches the home region, it is broadcast to all nodes in the home region.

If a source node wants to send data to a destination node, it needs to find the current location of the destination. The home region of the destination contains the information about the current location of the destination. To get this information the source determines the home region of the destination using a static mapping function and then unicasts a location discovery packet to that home region using the most forwarding with fixed radius (MFR) (Hou & Li, 1986) geographic routing algorithm. In MFR the source sends the route discovery packet to one of its neighbours, which is closest to the destination in terms of physical distance. This process repeats until the route discovery packet reaches a node in the home region of the destination. This node then generates a location reply packet containing the ID of the destination's current region.

There can be cases when a home region may contain only one node and the node moves to another region from its home region. In this case the source node will not receive a reply. To address this issue, the source sends the location discovery packet, after a certain period, to the surrounding regions of the home region of the destination node, with the hope that the destination has registered itself in any of those regions. This process of expanding the search region continues until a threshold is reached.

Once the current location of the destination is found, the data packets are routed to the destination's current region using MFR. Once one of the nodes in the destination's current region gets the packets, it can route them to the destination using DSR if it contains a cached route to the destination. Otherwise that node floods the packets, using a method similar to route discovery in DSR, within its current region. Eventually the packets reach the destination. Thus SLURP uses MFR to get data packets to the current region of the destination and then DSR to get the packets to the destination.

Distributed Spanning Tree (DST) Based Routing Protocol

DST (Radhakrishnan, Racherla, Sekharan, Rao, & Batsell, 1999, 2003) uses spanning trees in regions where the topology is stable and a flooding-like scheme in highly dynamic regions of the network.

In DST, nodes are grouped into a number of disjoint trees. If two nodes of two trees come within the transmission range of each other but are likely to move away shortly, they form a bridge between the two trees. Otherwise those trees merge to form a larger tree and one of the roots of the trees becomes the root of the newly merged tree.

To determine a route, the authors have proposed two different routing techniques: hybrid tree-flooding and distributed spanning tree shuttling. In hybrid tree-flooding packets are sent to all possible neighbours and adjoining bridges. When

a node receives a packet, it stores the packet for a period, known as the *holding-time*, after which the packet is deleted. The rationale for the holding-time is that for systems, which are becoming more stable and connected, it might be useful to buffer packets and route them as the connectivity of the network increases over time. In distributed spanning tree shuttling, the routing happens in several steps. In the first step, if the source node is not the root, the packet is sent to the root. In the second step the packet is re-broadcast along the tree edges toward the leaf nodes. When a packet reaches a leaf node, the packet is sent upwards, which is the third step, until it reaches a certain height known as the *shuttling level*. Once a node in the shuttling level is reached, the packet is sent along the adjoining bridges and the algorithm continues from the second step.

Radhakrishnan et al. (1999, 2003) claim that shuttling mechanism uses fewer messages compared with tree-flooding. However, DST is prone to a single point of failure due to the reliance on the root node to configure the tree (Abolhasan et al., 2004). Moreover, the holding time may introduce extra delays for packets that may be unsuitable for some time sensitive applications.

Hybrid Ad-Hoc Routing Protocol (HARP)

HARP (Nikaein, Bonnet, & Nikaein, 2001) is a tree-based hybrid routing protocol. The trees are connected via gateway nodes, that is, the neighbouring nodes belonging to different trees, to form a forest. Unlike DST, HARP does not require the trees to have root nodes. The trees are also referred to as zones. Similar to ZHLS, the zones in HARP do not overlap. However, unlike ZHLS, HARP does not rely on a static zone map. Moreover, it does not require a clusterhead to coordinate data and control packet transmissions.

For zone creation, HARP relies on the distributed dynamic routing algorithm (DDR) (Nikaein, Labiod, & Bonnet, 2000). DDR consists of six cyclic time-ordered phases: preferred neighbour election, forest construction, intra-tree clustering, inter-tree clustering, zone naming, and zone partitioning. It uses only beacons, which are periodic messages exchanged between a node and its neighbours, to perform each of these phases. Thus it avoids global broadcasting throughout the network. In the beginning, each node starts the preferred neighbour election procedure to choose a preferred neighbour. The preferred neighbour of a node is the node that has the most neighbours. Then a forest is constructed by connecting each node to its preferred neighbour and vice versa. After that the intra-tree clustering algorithm is used to build up the structure of the zone and to build up the intra-zone routing table. Next the inter-tree clustering algorithm is executed to determine the connectivity of neighbouring zones. The gateway nodes keep this information in their inter-zone routing tables. Each constructed tree is assigned a name by executing the zone naming algorithm. Then the network is partitioned into a number of non-overlapping zones.

In HARP, routing is performed on two levels: intra-zone and inter-zone depending on whether the destination belongs to the same zone or in a different zone. The intra-zone routing mechanism uses a pro-active approach to find the destination within a zone. For inter-zone routing, HARP uses a reactive approach. During the route discovery phase, route request packets propagate from zone to zone via the gateway nodes. Inside a zone, the route request packets follow the tree structure provided by DDR. Therefore, unlike ZRP, HARP limits the flooding of route request packets to a subset of nodes. After this limited flooding, several paths may be discovered for a given destination. The destination chooses the most suitable path and unicasts a path reply packet to the source.

HARP assigns each path a refresh time, after which a new route discovery phase is triggered to avoid path failure, as the network topology may change over time. If a link failure occurs in the meantime, the node that detected the link

failure sends a path error message to the source and holds the traffic for a period. The rationale for this holding period is that HARP applies the pro-active approach inside the zone and there is a chance of receiving new routing information embedded in the periodic beacons. This feature may increase the percentage of data received successfully at the destination. When the source receives the path error message it initiates a new route discovery procedure.

Sharp Hybrid Adaptive Routing Protocol (SHARP)

SHARP (Ramasubramanian, Haas, & Sirer, 2003), unlike other hybrid routing protocols, adapts between pro-active and reactive routing strategies by adjusting the radii of pro-active zones dynamically.

SHARP creates pro-active zones automatically around the destinations that receive data from many sources. SHARP has borrowed techniques from DSDV and TORA to build an efficient pro-active routing protocol known as the SPR (SHARP pro-active routing) protocol.

SPR performs routing by building and maintaining a directed acyclic graph (DAG) rooted at the destination in each zone. Therefore nodes within a pro-active zone maintain routes pro-actively only to the destination node. The topology forming mechanism of SPR does not guarantee that a node can always find the shortest path, in terms of hop count, to a destination. SPR requires the destination to periodically initiate a DAG construction process. In addition to this, SPR uses the failure recovery mechanism of TORA to restore the DAG in response to link-failures. Each node in a pro-active zone periodically broadcasts update packets to its neighbours. Each update packet contains the height (used in TORA) of the transmitted node and also acts as a HELLO beacon.

The nodes that are not located in the pro-active zone of a given destination use a reactive routing protocol, that is, AODV in this case, to establish routes to that destination. Once a data packet enters the pro-active zone of a destination using AODV, it is routed using SPR.

In SHARP, each destination monitors the network characteristics as well as the data traffic characteristics. Monitoring network characteristics enables SHARP to vary the radius of the pro-active zone in order to reduce per-packet routing overhead (that is, route discovery, update, and maintenance costs). Monitoring data traffic characteristics is used to adjust the radius of the pro-active zone in order to reduce loss rate and delay jitter in the application layer.

In order to estimate per-packet routing overhead of SPR, the nodes in a pro-active zone measure the average link lifetime (λ) of their immediate links and the number of immediate neighbours (n). These values are sent to the destination D of a pro-active zone periodically. Let f_u be the frequency at which each node within the pro-active zone generates update packets, and let f_c be the frequency at which D performs DAG constructions. If N_r^D is the number of nodes around D with radius r then the total fixed overhead of SPR pro-active routing component is N_r^D ($f_u + f_c$) packets/second. The average frequency of event-triggered updates at a node can be approximated by

$$\frac{1}{2(\beta_n - 1)\lambda} \text{ where } \beta_n = \sum_{i=1}^{n} \frac{1}{i}$$

Therefore, the total routing overheads of SPR at node D is

$$N_r^D \left(f_u + f_c + \frac{1}{2(\beta_n - 1)\lambda} \right) \quad (1)$$

On the other hand the per-packet routing overhead of AODV for each source *S* outside the pro-active zone at a distance *h* hops from *D* can be approximated by

$$N_{h-r}^{S}\left(\frac{h-r}{\lambda}\right) \qquad (2)$$

Here N_{h-r}^{S} is the number of nodes around *S* with radius *h-r*.

D estimates the incremental difference in the overheads of the pro-active and reactive routing components using Equation (1) and Equation (2) respectively. If the reduction in the overhead of the reactive component is more than *up_threshold* times the increase in the overhead of the pro-active component, *r* is increased by 1. Conversely if the reduction in the overhead of the pro-active component is more than *down_threshold* times the increase in the overhead of the reactive component, then *r* is decreased by 1. Otherwise *r* is kept fixed.

To adjust the radius of the pro-active zone in order to reduce loss rate and delay jitter, the destination measures loss rate and delay jitter from its observed traffic pattern. If the perceived loss rate (or the delay jitter) is more than a threshold, *r* is increased by 1. On the other hand *r* is decreased by 1 if the measured loss rate (or delay jitter) is less than a threshold.

It has been shown by simulation that SHARP performs well compared to a pure reactive or pro-active routing protocol in an ad-hoc network where most of the nodes are sources with few destinations and moderate mobility. Since SHARP performs some additional steps and requires extra control packets compared to that of the constituent protocols in their original forms, there is a possibility that these extra overheads may not be compensated by the overall network performance improvement if the number of destinations or the mobility of the nodes increases for the same network.

Comparisons of Hybrid Routing Protocols

Hybrid routing protocols try to minimise routing overheads compared to any reactive and pro-active routing protocol as network size increases. However, since most of the hybrid routing protocols are hierarchical in structure, they are likely to incur excessive processing and communication overheads for forming and maintaining clusters/zones/trees in highly mobile MANETs. Large MANETs with group mobility are likely to be best served by these protocols.

DST relies on a single node to configure the associated tree. Hence it can introduce extra routing delays, and excessive processing and communication overheads when the root node of a tree becomes unreachable by its children.

In HARP, nodes have to pass their traffic through a subset of neighbours known as preferred neighbours. These preferred nodes have to transmit more routing and data packets than any other nodes and hence may get less sleep time to conserve energy than others (Abolhasan et al., 2004). Since many nodes would want to communicate with the same preferred neighbour, channel contention would increase around that preferred neighbour. In networks with high node density and traffic, channel contention around preferred neighbours would increase waiting for transmission channel. This can significantly reduce overall throughput of the network since packets have to be dropped when buffers become full (Abolhasan et al., 2004).

On the other hand, ZHLS, SLURP, and SHARP do not rely on fixed nodes to perform any critical operation such as cluster formation. Therefore these protocols may have lower processing and communication overheads for large MANETs than other hybrid routing protocols. Table 3 shows the comparison of various hybrid routing protocols based on their major characteristics.

Table 3. Comparison of various hybrid routing protocols

	WCC[I]	***WCC [I',RD]***	***WCC [I',RM]***	***WTC [I]***	***WTC [I',RD]***	***WTC [I'RM]***	***RS***	***PB***	***Advantages***	***Disadvantages***
ZRP	O*(n)*	O*(N+r)*	O*(N+r)*	O*(d)*	O*(2D)*	O*(2D)*	F	Yes*	Reduced communication compared to pure pro-active routing algorithms; Faster route discovery within a zone than any pure reactive routing protocol	For large values of routing zone it may behave like a pure reactive routing protocol; Overlapping zones
ZHLS	O*(N/M)*	O*(N+r)*	O*(N+r)*	O*(d)*	O*(2D)*	O*(2D)*	H	No	Non overlapping zone and hence capable of supporting frequency reuse	Requires GPS, Requires static zone maps;
SLURP	O*(x+N/M)*	O*(2y)*	O*(2x)*	O*(2d)*	O*(2D)*	O*(2D)*	H	No	Home region reduces cost of maintaining routing information by eliminating global route discovery	Requires static zone maps;
DST	O*(n)*	O*(2N)*	O*(N)*	O*(h)*	O*(2D)*	O*(2D)*	H	No	Holding time reduces retransmissions	Requires trees to have root nodes; Single point of failure
HARP	O*(n)*	O*(N+r)*	O*(N+r)*	O*(d)*	O*(2D)*	O*(2D)*	H	Yes	Applies early route maintenance to avoid extra delay caused by path failure during data transmission; Does not require zone map; Does not require the trees to have roots	Preferred neighbours may become bottlenecks and hence degrade network performance
SAHRP	O*(n)*	O*(2N)*	O*(y)*	O*(d)*	O*(2D)*	O*(D)*	F	Yes	Automatically finds the balance point between pro-active and reactive routing.	Moderate mobility; More overhead than pure AODV and TORA

WCC: Worst Case Communication Complexity, i.e. number of messages needed to perform a route discovery or an update operation in worst case; *WTC*: Worst Case Time complexity, i.e. number of steps involved to perform a route discover or an update operation in worst case; *RD*: Route Discovery; *RM*: Route Maintenance; *I*: Intra zone; *I'*: Inter zone; *RS*: Routing Structure; *F*: Flat; *H*: Hierarchical; *PB*: Periodic Beacons; *N*: Number of nodes in the network; *D*: Diameter of the network; *d*: Diameter of a zone, home region or cluster or tree; *n*: Number of nodes in a zone, home region, cluster or tree; *r*: Number of nodes in the route reply path; *M*: Number of zones, home regions or clusters; *x*: Number of nodes in the path to the home region; *h*: Height of the tree; *y*: Number of nodes from the source of route error to the source node; *: If pro-active and reactive routing protocols using beacons are used;

Table 4. Comparison of pro-active, reactive, and hybrid routing protocols (Abolhasan et al., 2004)

	Pro-active	Reactive	Hybrid
Routing Structure	Both flat and hierarchical	Usually flat	Usually Hierarchical
Availability of Routes	Always available.	Determined when needed. Sometimes overheard routes are stored for a limited time (e.g. in DSR).	Always available within if source and destination reside within the same zone/cluster/tree.
Volume of control traffic	Usually high. Exceptions such as FSLS and HOLSR.	Usually lower than pro-active routing.	In most cases lower than pro-active and reactive routing protocol.
Storage requirement	Usually high	Usually lower than pro-active routing protocols	Usually lower than pure pro-active and reactive routing protocols if the size of zones/ clusters/trees can be properly determined in large networks.
Delay for route discovery	Predetermined	Higher than pro-active routing protocols	Similar to pro-active routing protocols if source and destination are located within the same zone/ cluster/tree. Otherwise usually higher than pro-active but lower than reactive.
Mobility support	Low to moderate mobility is supported. Group mobility is usually required for hierarchical structured routing.	Can support higher mobility than pro-active routing protocols.	Usually supports lower level of mobility than reactive routing protocols since routing structure is mostly hierarchical in this approach.
Scalability	Usually up to 100 nodes. FSLS and HOLSR may scale higher.	Source routing protocol does not scale well, usually up to few hundred nodes. Hop by hop routing scales better than source routing.	1000 or more.

Table 4 provides an overall comparison of the pro-active, reactive, and hybrid routing strategies.

OPEN RESEARCH CHALLENGES

Ad-hoc networks have been one of the popular research fields in ubiquitous communications for more than a decade. Researchers have explored various ad-hoc routing protocols at various levels of detail. However, no routing protocol has become a winner for all scenarios. The following aspects have been considered to make routing protocols more efficient and robust:

- Quality of service (QoS)
- Flexibility and scalability
- Security
- Energy efficiency
- Multicast
- Antennas

QoS routing aims to guarantee certain performance for a flow in terms of bandwidth, delay, jitter, successful delivery probability, and

so forth. However due to the dynamics of nodes (e.g., limited power, mobility, etc.), fluctuating link characteristics, limited radio bandwidth, varying level of radio channel contention along multiple hops (Islam, Pose, & Kopp, 2005a, 2005b, 2005c), and varied user demand, routing with guaranteed end-to-end QoS is not a trivial issue.

Since no single routing protocol has been proven to be a winner, a routing protocol has to be flexible enough to switch between various pro-active and reactive routing protocols based on characteristics of the network. SHARP adapts between pro-active and reactive routing strategies by adjusting dynamically the radii of pro-active zones based on link lifetime. However other network parameters can be considered to make the routing protocol more flexible to scale adaptively with the heterogeneity of nodes' capabilities, network size, user demand, and so forth.

Usually the routing protocols assume node in the network would follow the protocol description properly. However, due to the lack of clear physical boundary, wireless networks are at more risk from attackers than their cabled counterpart. Multi-hop ad-hoc networks make the security problem even more challenging since these may lack central trusted servers. Several intrusion prevention and detection schemes (Yang, Luo, Ye, Lu, & Zhang, 2004) have been proposed. These solutions come at a cost. Implementational complexity, processing, and communication costs are involved that need to be minimised for fully automated and power conserved ad-hoc networks.

Nodes in an ad-hoc network are assumed to route traffic for other nodes if needed. These nodes may have to rely on limited battery power if they are mobile or not connected to a main power source. Examples of such nodes are nodes in a sensor network. In these situations the issue of energy efficiency becomes one of the most important problems. There are routing protocols, for example, Singh, Woo, and Vaidya (1998), that try to conserve power. However, it is still an open research challenge to determine how to conserve power and perform routing at the same time without compromising overall network performance.

Multi-cast routing enables one-to-many communication. For ad-hoc networks multi-cast routing must be able to cope with network size and node mobility (Royer & Toh, 1999). Moreover issues like uni-directional links (Gerla, Lee, Park, & Yi, 2005) and power conservation (Wan, Calinescu, & Yi, 2004) have to be considered.

Ad-hoc routing protocols usually assume each node is equipped an omni-directional antenna. In depth investigation is needed to see the effect of using smart, or multiple omni- or directional antennas at the routing layer of ad-hoc networks.

CONCLUSION

Ad-hoc network topologies can change dynamically. Similarly performance of wireless links can vary unpredictably. Hence routing in ad-hoc networks is much more difficult than in conventional networks. In this chapter we have provided a description of various routing protocols suitable for ad-hoc networks. We have evaluated their major characteristics and differences. The study suggests that no single routing protocol or class of protocols is best suited for all scenarios. In this chapter we have looked at various hybrid routing protocols with the aim of combining the most appropriate features of each for ad-hoc network situations. While this leads to various practical approaches to ad-hoc routing, there are still many open research questions to be answered before generally applicable routing protocols can be developed to suit ad-hoc networks, especially when they are used to support demanding multimedia applications. We have outlined some of these research challenges.

REFERENCES

Abolhasan, M., Wysocki, T., & Dutkiewicz, E. (2004). A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks, 2*(1), 1-22.

Aggelou, G., & Tafazolli, R. (1999). RDMAR: A bandwidth-efficient routing protocol for mobile ad-hoc networks. *Proceedings of the 2nd ACM International Workshop on Wireless Mobile Multimedia* (*WOWMOM*), ISBN 1-58113-129-1 (pp. 26-33).

Basagni, S., Chlamtac, I., Syrotiuk, V. R., & Woodward, B. A. (1998). A distance routing effect algorithm for mobility (DREAM). *Proceedings of the 4th International Conference on Mobile Computing and Networking (MobiCom)*, ISBN 1-58113-035-X (pp. 76-84).

Bellur, B., & Ogier, R. G. (1999). A reliable, efficient topology broadcast protocol for dynamic networks. *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies* (*InfoCom*), ISBN 0-7803-5417-6, 1 (pp. 178-186).

Bertsekas, D., & Gallager, R. (1987). *Data networks* (pp. 297-333). Prentice-Hall Inc.

Broch, J., Maltz, D., Hu Y.-C., & Jecheva, J. (1998). A performance comparison of multi-hop wireless ad hoc network routing protocols. *Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, ISBN 1-58113-035-X (pp. 85-97).

Chen, T.-W., & Gerla, M. (1998). Global state routing: A new routing scheme for ad-hoc wireless networks. *Proceedings of the IEEE International Conference on Communications (ICC)*, (pp. 171-175).

Chiang, C., Wu, H., Liu, W., & Gerla, M. (1997). Routing in clustered multihop, mobile wireless networks. *Proceedings of the IEEE Singapore International Conference on Networks (SICON)* (pp. 197-211).

Corson, M. S., & Ephremides. (1995). A distributed routing algorithm for mobile wireless networks. *ACM/Baltzer Wireless Networks, 1*(1), 61-81.

Das, S. R., Castaneda, R., & Yan, J. (2000). Simulation-based performance evaluation of routing protocols for mobile ad hoc networks. *ACM/Baltzer Mobile Networks and Applications (MONET), 5*(3), 179-189.

Dube, R., Rais, C. D., Kuang-Yeh, W., & Tripathi, S. K. (1997). Signal stability-based adaptive routing (SSA) for ad-hoc mobile networks. *IEEE Personal Communications, 4*(1), 36-45.

Gafni, E., & Bertsekas, D. (1981). Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, 11-18.

Garcia-Luna-Aceves, J. J., & Spohn, M. (1999). Source-tree routing in wireless networks. *Proceedings of the 7th Annual International Conference on Network Protocols* (*ICNP*), ISBN 0-7695-0412-4 (pp. 273-282).

Gerla, M., Hong, X., & Guangyu, P. (2000). Landmark routing for large ad hoc wireless networks. *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, 3 (pp. 1702-1706).

Gerla, M., Lee Y.-Z., Park, J.-S., & Yi, Y. (2005). On demand multicast routing with unidirectional links. *IEEE Wireless Communications and Networking Conference, 4*, 2162-2167.

Gonzalez, L. V., Ge, Y., & Lamont, L. (2005). HOLSR: A hierarchical proactive routing mechanism for mobile ad hoc networks. *IEEE Communication Magazine, 43*(7), 118-125.

Guangyu, P., Geria, M., & Hong, X. (2000). LANMAR: Landmark routing for large scale wireless ad hoc networks with group mobility. *Proceedings of the 1st Annual Workshop on Mobile and Ad Hoc Networking and Computing (MobiHoc)* (pp. 11-18).

Gunes, M., Sorges, U., & Bouazizi, I. (2002). ARA: The ant-colony based routing algorithm for MANETs. *Proceedings of the International Conference on Parallel Processing (ICPP) Workshops* (pp. 79-85).

Haas, Z. J. (1997). A new routing protocol for the reconfigurable wireless networks. *Proceedings of the 6th IEEE International Conference on Universal Personal Communications (ICUPC)*, 2 (pp. 562-566).

Haas, Z. J., & Pearlman, M. R. (1998). The performance of a new routing protocol for the reconfigurable wireless networks. *Proceedings of the IEEE International Conference on Communications (ICC)*, 1 (pp. 156-160).

Higaki, H., & Umeshima, S. (2004). Multiple-route ad-hoc on-demand distance vector (MRAODV) routing protocol. *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS).*

Hong, X., Xu, K., & Gerla, M. (2002). Scalable routing protocols for mobile ad hoc networks. *Kluwer Wireless Networks, 16*(4), 11-21.

Hou, T.-C., & Li, V. (1986). Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications, 34*(1), 38-44.

Hu, Y.-C., & Johnson, D. (2000). Caching strategies in on-demand routing protocols for wireless ad hoc networks. *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom)*, ISBN 1-58113-197-6 (pp. 231-242).

Hu, Y.-C., & Johnson, D. (2001). Implicit source routes for on-demand ad hoc network routing. *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, ISBN 1-58113-428-2 (pp. 1-10).

Islam, M. M., Pose, R., & Kopp, C. (2005a). Challenges and a solution to support QoS for real-time traffic in multi-hop ad-hoc networks. *Proceedings of the 2nd IEEE and IFIP International Conference on Wireless and Optical Communications Networks (WOCN)*, ISBN 0-7803-9019-9.

Islam, M. M., Pose, R., & Kopp, C. (2005b). MAC layer support for real-time traffic in a SAHN. *International Conference on Information Technology: Coding and Computing (ITCC)* (pp. 639-645).

Islam, M. M., Pose, R., & Kopp, C. (2005c). Making SAHN-MAC independent of single frequency channel and omnidirectional antennas. *IASTED International Conference on Networks and Communication Systems (NCS)*, 6pp.

Jacquet, P., Muhlethaler, P., Clausen, T., Laouiti, A., Qayyum, A., & Viennot, L. (2001). Optimized link state routing protocol for ad hoc networks. *Proceedings of the IEEE National Multi-Topic Conference (INMIC)* (pp. 62-68).

Jiang, M., Li, J., & Tay, Y. C. (1999). *Cluster based routing protocol.* Draft-ietf-manet-cbrp-spec-01.txt, Internet Draft, August.

Joa-Ng, M., & Lu, I.-T. (1999). A peer-to-peer zone-based two-level link state routing for mobile ad-hoc networks. *IEEE Journal on Selected Areas in Communications, 17*(8), 1415-1425.

Johnson, D., & Maltz, D. (1996). Dynamic source routing in ad hoc wireless networks. In T. Imielinski, & H. Korth (Eds.), *Mobile computing* (vol. 353, pp. 151-181). Kluwer Academic Publishers.

Kasera, K. K., & Ramanathan, R. (1997). A location management protocol for hierarchically organized multihop mobile wireless networks. *Proceedings of the IEEE 6th International Conference on Universal Personal Communications (ICUPC)*, 1 (pp. 158-162).

Kleinrock, L., & Stevens, K. (1971). *Fisheye: A lens like computer display transformation.* Technical Report, UCLA Computer Science Department.

Ko, Y. B., & Vaidya, N. H. (1998). Location-aided routing (LAR) in mobile ad hoc networks. *Proceedings of the 4th annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, ISBN 1-58113-035-X (pp. 66-75).

Kopp, C., & Pose, R. (1998). Bypassing the home computing bottleneck: The suburban area network. *3rd Australasian Computer Architecture Conference (ACAC), 20*(4), 87-100. Springer-Verlag, ISBN: 981-3083-93-X.

Marina, M. K., & Das, S. R. (2001a). Performance of route caching strategies in dynamic source routing. *Proceedings of the International Workshop on Wireless Networks and Mobile Computing (WNMC) in Conjunction with International Conference on Distributed Computing Systems (ICDCS)* (pp. 425-432).

Marina, M. K., & Das, S. R. (2001b). On-demand multipath distance vector routing in ad hoc networks. *Proceedings of the International Conference for Network Protocols (ICNP)* (pp. 14-23).

Marina, M. K., & Das, S. R. (2003). *Ad-hoc on-demand multipath distance vector routing.* Technical Report, Computer Science Department, Stony Brook University.

Murthy, M., & Garcia-Luna-Aceves, J. J. (1995). A routing protocol for packet radio networks. *Proceedings of the 1st Annual International Conference on Mobile Computing and Networking (MobiCom)*, ISBN 0-89791-814-2 (pp. 86-95).

Nikaein, N., Bonnet, C., & Nikaein, N. (2001). HARP: Hybrid ad hoc routing protocol. *Proceedings of the International Symposium on Telecommunications (IST).*

Nikaein, N., Labiod, H., & Bonnet, C. (2000). DDR: Distributed dynamic routing algorithm for mobile ad hoc networks. *Proceedings of the 1st Annual Workshop on Mobile and Ad Hoc (MobiHoc)* (pp. 19-27).

Ogier, R., Templin, F., & Lewis, M. (2004). *RFC 3684 on topology dissemination based on reverse-path forwarding (TBRPF).* Draft-ietf-manet-tbrpf-11.txt, Internet Draft, Network Working Group, February 2004.

Park, V. D., & Corson, M. S. (1997). A highly adaptive distributed routing algorithm for mobile wireless networks. *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom)*, ISBN 0-8186-7780-5, 3 (pp. 1405-1413).

Pei, G., Gerla, M., & Chen, T. (2000). Fisheye state routing in mobile ad hoc networks. *Proceedings of the ICDCS Workshop on Wireless Networks and Mobile Computing* (pp. 71-78).

Pei, G., Gerla, M., Hong, X., & Chiang, C. C. (1999). A wireless hierarchical routing protocol with group mobility. *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 3 (pp. 1538-1542).

Perkins, C., & Bhagwat, P. (1994). Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *Proceedings of the ACM Conference on Communications Architectures, Protocols and Applications (SIGCOMM)* (pp. 234-244).

Perkins, C. E., & Royer, E. M. (1999). Ad-hoc on-demand distance vector routing. *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)* (pp. 90-100).

Radhakrishnan, S., Racherla, G., Sekharan, C. N., Rao, N. S. V., & Batsell, S. G. (1999). DST-A routing protocol for ad hoc networks using

distributed spanning trees. *Proceedings of the IEEE Wireless Communications and Networking Conference, (WCNC),* 3 (pp. 1543-1547).

Radhakrishnan, S., Racherla, G., Sekharan, C. N., Rao, N. S. V., & Batsell, S. G. (2003). Protocol for dynamic ad-hoc networks using distributed spanning trees. *Kluwer Wireless Networks, 9*(6), 673-686.

Ramasubramanian, V., Haas, Z. J., & Sirer, E. G. (2003). SHARP: A hybrid adaptive routing protocol for mobile ad hoc networks. *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, ISBN 1-58113-684-6 (pp. 303-314).

Royer, E. M., & Toh, C.-K. (1999). A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications, 6*(2), 46-55.

Santivez, C., Ramanathan, R., & Stavrakakis, I. (2001). Making link-state routing scale for ad hoc networks. *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking Computing (MobiHoc)*, ISBN 1-58113-428-2 (pp. 22-32).

Singh, S., Woo, M., & Vaidya, N. H. (1998). Power-aware routing in mobile ad hoc networks. *Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, ISBN 1-58113-035-X (pp. 181-190).

Toh, C. K. (1996). A novel distributed routing protocol to support ad-hoc mobile computing. *Proceedings of the 15th IEEE International Performance, Computing, and Communications Conference (IPCCC)* (pp. 480-486).

Toh, C. K. (1997). Associativity-based routing for ad hoc mobile networks. *Journal of Wireless Personal Communications, 4*(2), 103-139.

Tsuchiya, P. F. (1988). The landmark hierarchy: A new hierarchy for routing in very large networks. *Computer Communication Review, 18*(4), 35-42.

Wan, P.-J., Calinescu, G., & Yi, C.-W. (2004). Minimum-power multicast routing in static ad hoc wireless networks. *IEEE/ACM Transactions on Networking, 12*(3), 507-514.

Wang, L., Zhang, L. F., Shu, Y. T., Dong, M., & Yang, O. W. W. (2001). Adaptive multi-path source routing in wireless ad-hoc networks. *Proceedings of the IEEE International Conference on Communications (ICC)* (pp. 867-871).

Woo, S.-C. M., & Singh, S. (2001). Scalable routing protocol for ad hoc networks. *Journal of Wireless Networks, 7*(5), 513-529.

Yang, H., Luo, H., Ye, F., Lu, S., & Zhang, L. (2004). Security in mobile ad hoc networks: Challenges and solutions. *IEEE Wireless Communications, 11*(1), 38-47.