Survey of Routing in Ad Hoc Networks

Dragos Niculescu (dniculesc@paul.rutgers.edu)

December 23, 1998

Abstract

An *ad hoc* network consists of a number of mobile hosts who communicate with each other over a wireless channel without any centralized control. The basic problem is to obtain a distributed routing scheme so that under the network connectivity assumption any mobile host can transmit/receive data from any other host in the network. Due to the lack of any infrastructure, a host may need the aid of other hosts to route a packet to the destination. The algorithms should provide loop-free routes, multiple routes (wireless bandwidth is usually scarce), allow scalability, and run in a timely fashion such that the topology does not change before the routes are created. In this paper we survey the existing algorithms of ad hoc routing and attempt a classification based on their capacity to adapt under conditions such as: high/low mobility, high/normal node density, different traffic patterns, average per message latency.

1 Terminology

- Ad hoc network: a collection of hosts placed in a graph with changing topology and no infrastructure support, so that the nodes should distributedly implement routing.
- upstream link: a link on which a node is able to receive data
- downstream link: a link on which a node can transmit data to a given destination
- host, mobile host, node, router: a mobile computer
- flooding: the process by which a message is broadcast hop by hop to a number of hosts

2 Introduction

The most important issue in an ad hoc network is how a mobile host can communicate with another mobile host which is not in its direct transmission range. Usually, the measure of distance is the number of hops in the path, but some algorithms may take into consideration other measures, as delay. Some of the facts that complicate ad hoc routing are: hosts are moving and MAC links are only temporary, path delays are dependent on events which happen at different physical locations and on the topology of the network, traffic patterns cannot be predicted.

In a mobile environment, it is not clear that shortest-path algorithms, however efficient, are worth the time and communication complexity they entail. In a rapidly changing topology, it is likely that the topology will change again before a shortest-path computation can converge. Thus, the communication overhead expended in the computation is partially, and at times, completely wasted. Also, in a congested mobile network, quickly discovering multiple routing options is likely preferable to computing a single, shortest-path route. At very high rates of topological changes, no routing protocol can adapt quickly enough and "flooding" of data packets is the only option. Before this point is reached, there is a region where the rate of topological changes is too fast to allow protocols appropriate for a nearly-static topology to converge, but not so fast as to make flooding the only possible routing option. In this region, changes may occur too frequently to allow shortest-path protocols to converge, yet enough consistency remains to suggest that a more efficient routing than flooding is possible. This is the realm of "highly adaptive" routing protocols, and their goals [CMB96]are to:

- execute distributedly
- create loop-free, multiple routes
- to build routes quickly so that they may be used before the topology changes, and
- to react quickly, reestablishing routing (if possible and desired) when topological changes destroy existing routes.

Here, routing optimality of secondary importance; what matters is simply finding one or more routes quickly, with minimal overhead. A fundamental result of optimal routing [BG92] shows that "*for low input traffic, one should use only one path (the fastest in terms of transmission time), and as traffic input increases, additional paths are used to avoid overloading the fastest* *path*". Therefore, because in ad hoc networks bandwidth is scarce, congestion is high and the topology is rapidly changing, it is necessary to have alternative paths available.

Although the area of ad hoc networks is relatively new, we already can choose from a wide diversity of algorithms. One feature we can split categories of algorithms on is the way they build and maintain routes: on demand or global. On demand algorithms discover the route to a destination only when that route is needed. Some routing algorithms in this category are: Lightweight Mobile Routing (**LMR**)[CMB96], Ad Hoc On-Demand Distance Vector (**AODV**)[PR97], Dynamic Source Routing (**DSR**)[JM96], Temporally-Ordered Routing Algorithm (**TORA**)[PC97]. Global routing schemes maintain an estimate image of the entire topology in each node. Maintaining the accuracy of this estimation involves periodically sending (or flooding) query messages in the network. Some algorithms in this category are: Destination Sequenced Distance-Vector (**DSDV**)[PB94], Global State Routing (**GSR**)[CG96], Wireless Routing Protocol (**WRP**)[MG96]. Hybrid algorithms combine, such as Zone Routing Protocol (**ZRP**)[Haas96] and Hierarchical State Routing (**HSR**)[CG98]combine combine features of the two mentioned categories.

The rest of the paper is organized as follows: section 3 presents a few global state algorithms, section 4 reviews on-demand algorithms mentioned above, section 5 presents two hybrid protocols, section 7 comments features and performance of the algorithms and compares performances in the two categories, and section 8 presents the conclusions of the survey.

3 Global state algorithms

They are also known in literature as next hop algorithms, and they can be further classified in *link-state* and *distance-vector.*

Link-state algorithms are closer to the centralized version of shortest path computation method. Each node stores locally an estimate of the topology, meaning a table with costs of all links. To maintain consistency, every node periodically floods the network with the status of its outgoing links. Based on this, each node may apply a shortest path method locally in order to decide for the best next hop. Loops may appear due to inconsistencies resulted from propagation delay, but they are short-lived.

Distance vector algorithms are basically variants of Bellman Ford algorithm, whose distributed version is presented in the next subsection.

3.1 Distributed Bellman Ford (DBF)

A very commonly used algorithm in wired networks is Distributed Bellman-Ford (RIP is based on DBF). Being a distance vector algorithm, every host maintains the length of the shortest path from each of its neighbor hosts to every destination in the network. It can be summarized as follows:

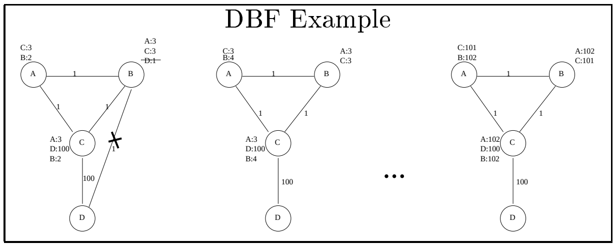
$D_i^0 = \infty, \forall i \neq 1$

$D_1^h = 0, \forall h$

$D_i^{h+1} = \min_j [d_{ij} + D_j^h], \forall i \neq 1$

D_i^h is the estimation of the shortest path from node i to node 1, at step h of the algorithm, and d_{ij} is the length of the link between node i and its neighbor j. The algorithm terminates when $D_i^h = D_i^{h+1}$. A separate copy of the algorithm is run for each destination.

In order to maintain up-to-date distance information, every host monitors its outgoing links and periodically broadcasts to neighboring hosts its current estimate of the shortest distance to every network destination. The algorithm is guaranteed to converge in a finite time when the topology is fixed, but it may generate loops, due to inconsistent estimates in changing topologies. To stabilize oscillations, we could add a large constant to the length of each edge, but this would reduce the sensitivity to changes.



In the above example in the initial configuration, link B-D fails, and triggers a long process of updates. In fact, the algorithm converges after 100 steps, the length of the edge C-B. If now this link fails too, nodes A,B and C will continue to increment their estimates, not realizing that in fact the destination D is not reachable anymore – this problem is known as "counting to infinity".

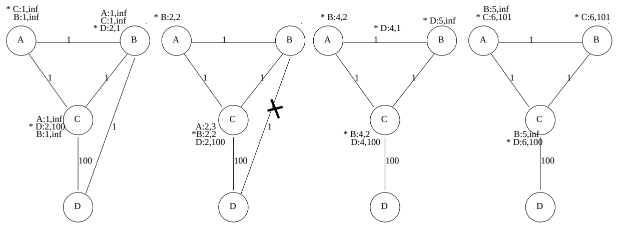
Drawbacks: generating long-lived loops, and counting to infinity (it takes a large number of update messages to detect that a node is unreachable).

3.2 Destination-Sequenced Distance-Vector Protocol (DSDV)

This algorithm is basically a DBF with timestamps. Each host maintains a forwarding table with an entry for each possible destination, estimated number of hops to that destination, and an even sequence number associated with the route (stamped by the destination). This table is periodically broadcast to all neighbors within the range of reception, together with a new even timestamp generated by the host. Other hosts receiving a table update will prefer routes with more recent sequence number, or for the same sequence number, routes that are shorter in number of hops. In the same manner in which tables are propagated in DBF, sequence numbers are sent to all hosts which may decide to maintain a routing entry for the originating host.

Whenever an outgoing link breaks (this is hopefully detected by the MAC level), any route through that hop is assigned an infinite distance, and an odd timestamp. This is the only situation when a route has a stamp originating at a node which is not the destination node of that route. In this way, any route with an even ("real") sequence number will be able to replace the broken link (∞ metric).

DSDV Example



In the above figure, we are considering tables for routing to destination D. Only entries marked with a "*" are actually kept in the routing tables, the other ones are possible routes obtained from periodical advertisement of neighbors. A route is replaced when the new route has a newer sequence number, or, for the same sequence numbers, when the new route has a better metric.

In order to damp the possible fluctuations, each host is in fact maintaining two routing tables - one that is advertised, and another one that is actually used. The host estimate the time between the arrival of the *first* route and the arrival of the *best* route, and this is the interval at which a route is updated from the actual table to the advertised table.

Advantages: It solves the main problem of DBF - loops - by using the timestamps; has a moderate memory complexity $O(n)$, n being the number of nodes (DBF has $O(nd)$, d=degree of a node).

Drawbacks: slow convergence, lack of multiple routes, parameter selection(settling time, update interval) may be critical

3.3 Global State Routing (GSR)

Classic LS floods each change to the entire network, which is not scalable. GSR is based on LS, but avoids the problem of flooding. For that it uses an idea similar to DBF to communicate the link state table of a node only to the immediate neighbors. GSR uses timestamps to help the neighbors always maintain the most recent estimates, in this respect being similar to DSDV. Information about topology changes is propagated through the network as each hop advertises its view of the network to its neighbors.

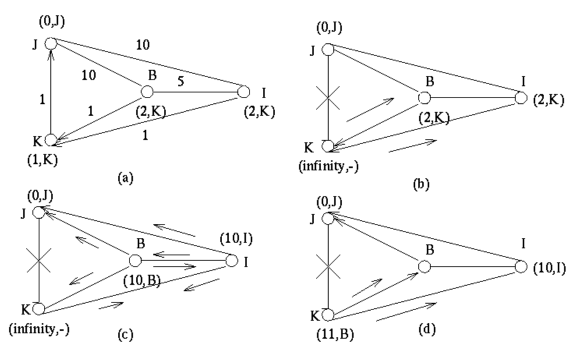
Advantages: GSR has more accurate image of the topology in each node, thus aiming for the best routing path, and also providing several possible paths for the same destination. It is good for slowly changing topologies and high loads.

Drawbacks: The complexity of computation for each update is high $O(|V|^2)$, while finding the best route may not be worth the effort. Large size updates consume a large amount of bandwidth, especially at a high mobility rate.

3.4 Wireless Routing Protocol (WRP)

WRP enhances DBF by maintaining for each destination the last node before the destination on the preferred path. Also, when the best distance is broadcast from a node to its neighbors, the first node on that best path is also specified. For example, in figure below (b), node B, after losing the link with D, beside broadcasting 3 as the new estimated best distance to D, would mention also that this best route is estimate through A. WRP does not rely on the MAC level protocol to sense the links with the neighbors, therefore it sends positive ACKs to confirm receiving of the route updates.

WRP Example



In the figure, node I is the source, and node J is the destination. When link (J,K) fails, node K reports an infinite distance to J, broadcasting this information to B and I. They will discard their routes to J through K, and use the other, now shorter routes. These new routes will be broadcast to K, which in turn will broadcast its estimate back.

Advantages: gets rid of the loops created by DBF

3.5 System and Traffic Adaptive Routing Algorithm (STARA)

STARA is in fact a distance-vector based algorithm, which takes care of uniformly distributing the load on all downstream links of a node. It is based on the following assumptions:

- The network topology and traffic rates are such that queueing delays are dominant over propagation delays;
- Each node has an accurate clock, even though clocks at different nodes need not be synchronized;
- There exists a mechanism to prioritize packets

A source s that has to route a packet to a destination d, will estimate the delay of sending this packet through all the immediate neighbors. With this information, s allocates its traffic among its neighbors, such that all utilized nodes have an equal mean delay to d. $D_s^d(t)$ is the mean estimated delay from source s to destination d, $D_{sk}^d(t)$ is the estimated delay from source s to destination d over outgoing link k, $p_{sk}^d(t)$is the flow on outgoing link k of source s that is allocated for destination d.

$$p_{sk}^d(t) = p_{sk}^d(t-1) + \alpha(t) \cdot (D_s^d(t) - D_{sk}^d(t))$$

$$D_s^d(t) = \sum_{k \in N_s^R(t)} p_{sk}^d(t) \cdot D_{sk}^d(t),\ k \in N(t)$$

It can be proven [GK98] that if each pair of nodes *(s,d)* applies this policy of minimizing the mean delay, then all intermediate nodes to d also have equal mean delays. In a network with increasing traffic-delay characteristics where all nodes apply the policy, there is a unique traffic allocation.

Advantages: It uses multiple routes, with the load distributed uniformly, such as to minimize delay, therefore optimal utilization of all links. Takes into consideration not only the changing topology, but changes in traffic as well.

Drawbacks: The price paid by a static estimation algorithm is longer delay for some packets when there is no congestion, this being exactly the opposite of the current Internet design. The scheme used for estimating the delay is *exponential forgetting*, which may lead to an even slower convergence that DBF.

4 On-demand algorithms

4.1 Lightweight Mobile Routing (LMR)

In LMR a set of paths is determined by broadcasting a query packet from the source S to all the immediate neighbors of S. This query packets are further broadcast until they hit nodes which have routes to D. The reply of these queries will carry a distance estimate to the destination (which is incremented on the return path to the source S). When the reply propagation ceases, each node knows its distance to D on each downstream link. Consider the example in the figure (a) below: Directed edges mean that, with respect to destination D, only the immediate neighbors have routes to D.

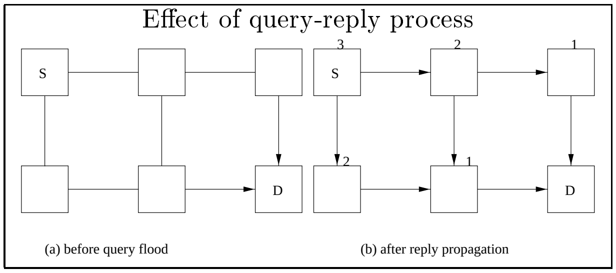
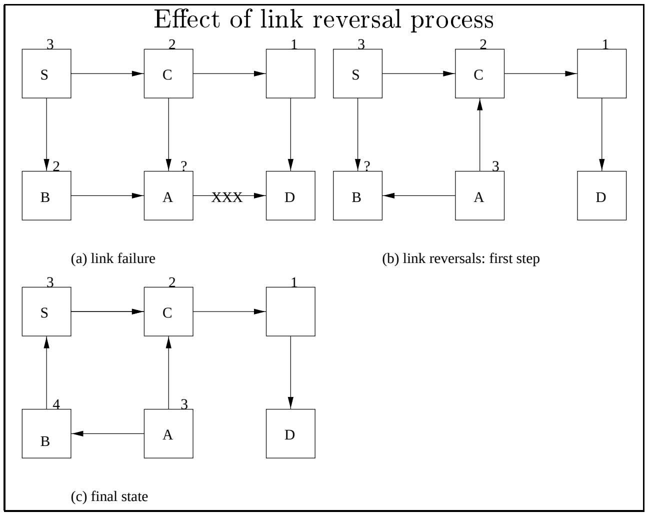


Figure (b) above shows the state of the graph after the route discovery - reply propagation builds a directed acyclic graph rooted at the destination. The link failures are taken care of by the link reversal method. To verify the state maintained by the mechanism, the protocol uses an infrequent, destination-initiated propagation wave to update the routers' distance estimates.

Route updates: When a node A that has D immediately downstream detects a link failure with D, it should redirect all the routes for D. This is done using a link reversal method: A assumes that, because the algorithm finds several routes, some nodes upstream of A will have alternative routes for D, therefore it exchanges all upstream links for D in downstream links for D. The neighbors of A will notice the loss of the downstream link for D and will propagate the same process.



Advantages: LMR has a low computation and communication complexity. As routing is done over multiple paths, traffic congestion is avoided.

Drawbacks: routing is not optimal as the routes are chosen without any cost consideration. LMR requires the links to be bidirectional.

4.2 Temporally-Ordered Routing Algorithm (TORA)

TORA is similar to LMR in building a DAG rooted at the destination from an initial undirected graph. A height is assigned to each node, and a downstream link is directed from a higher node to a lower node, as opposed to an upstream link. Height is defined as a quintuple $(\tau_i, oid_i, r_i, \delta_i, i)$, and allows complete lexicographic ordering between nodes. The first three values represent a reference level:

- τ_i is the time of link failure - TORA needs synchronized clocks in all nodes
- oid_i is the id of the node that generated the new reference level
- r_i distinguishes between original and higher reflected reference levels

The last two values are:

- δ_i-a value to order nodes with respect to the same reference level
- i- the unique id of the node

Nodes have initially NULL heights=*(_,_,_,_,id),* and the height of the destination is ZERO=*(0,0,0,0,did).* Each node also maintains a link-state array, where each link to a neighbor may be UNdirected, UPstream, or DowNstream.

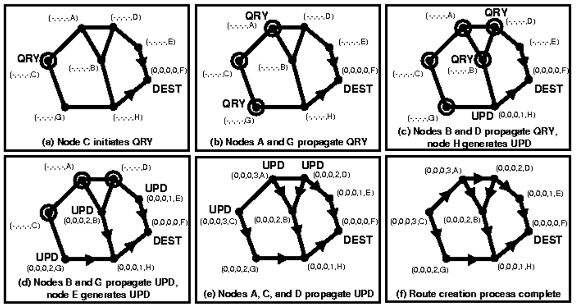
Creating routes: every node has an associated RR_i(route-required flag) and timers for UPD packets and for each link. A node with no downstream links broadcasts a QRY packet, containing the destination *id.* Upon receiving of a QRY packet, a node reacts as follows:

- if it has no downstream links, and RR is not set, it sets RR and broadcasts the QRY packet
- if it has no downstream links, and RR is set, it discards the QRY packet
- if it has at least one downstream link, and the height is NULL, it sets the height to be a local maximum among its non-NULL neighbors
- if an UPD packet was already sent in response for this source, it discards the QRY packet

Upon receiving a UPD packet, a node i updates the height of that neighboring node, and

- if RR is set, sets its δ_i 1 above the lowest non-null neighbor, resets RR, broadcasts the UPD packet with the new height
- if RR is not set, the node simply updates the state of the link for that neighbor

Creating routes



(a) Node C initiates QRY

(b) Nodes A and G propagate QRY

(c) Nodes B and D propagate QRY, node H generates UPD

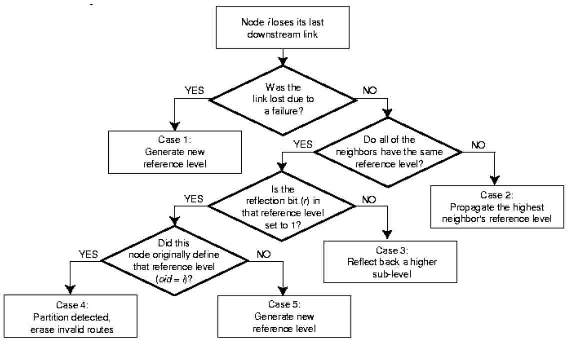
(d) Nodes B and G propagate UPD, node E generates UPD

(e) Nodes A, C, and D propagate UPD

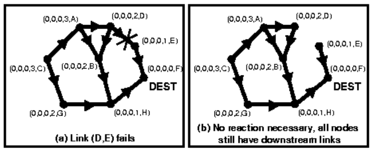
(f) Route creation process complete

Maintaining routes is performed only by nodes having non-NULL height. Here is the route maintenance pseudo-code:

Route maintenance


Node *i* loses its last downstream link
Was the link lost due to a failure?
YES
NO
Case 1: Generate new reference level
Do all of the neighbors have the same reference level?
YES
NO
Case 2: Propagate the highest neighbor's reference level
Is the reflection bit (*r*) in that reference level set to 1?
YES
NO
Case 3: Reflect back a higher sub-level
Did this node originally define that reference level (*oid* = *i*)?
YES
NO
Case 4: Partition detected, erase invalid routes
Case 5: Generate new reference level

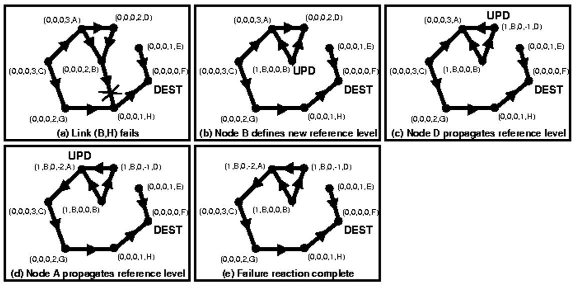
Link failure with no reaction



(a) Link (D,E) fails

(b) No reaction necessary, all nodes still have downstream links

Reestablishing routes after link failure



(a) Link (B,H) fails

(b) Node B defines new reference level

(c) Node D propagates reference level

(d) Node A propagates reference level

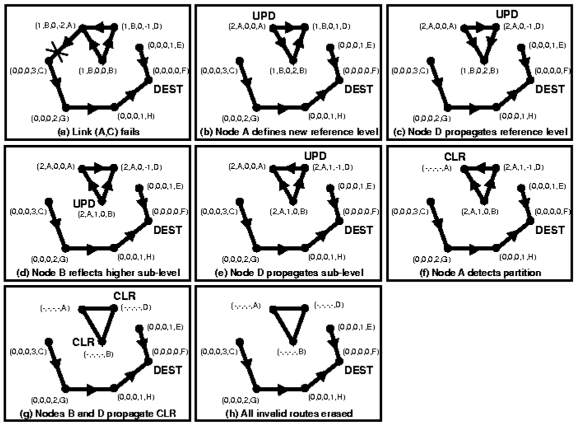
(e) Failure reaction complete

Erasing routes is performed in case 4 of the above diagram, when a partition is detected - a CLR message is broadcast.Upon receiving a CLR message, a node i

- if the CLR message has the same reference level, it sets the heights of the neighbors to NULL; broadcasts the CLR packet

- otherwise sets to NULL those neighbors having the same reference level as the CLR packet - actually sets to null the portion of the network that was partitioned

Erasing invalid routes after partitioning



(a) Link (A,C) fails
(b) Node A defines new reference level
(c) Node D propagates reference level
(d) Node B reflects higher sub-level
(e) Node D propagates sub-level
(f) Node A detects partition
(g) Nodes B and D propagate CLR
(h) All invalid routes erased

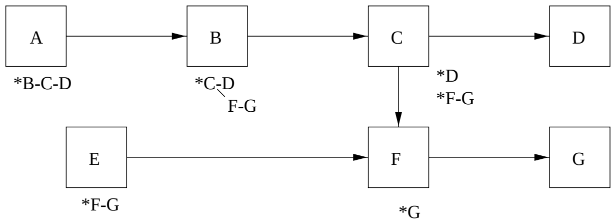
Advantages: loop-free, multiple paths; reactions to topological changes are maintained local; when portions of the mesh are disconnected, the new state is rapidly and locally updated.

Drawbacks: more overhead than other on-demand algorithms, high sensitivity to packet loss.

4.3 Dynamic Source Routing (DSR)

In DSR algorithm, the source of a data packet determines the complete sequence of hosts through which the packet is routed before it reaches the destination. Since the network connectivity is changing with time, the source routes are dynamically constructed using a *route-discovery* protocol, i.e. whenever a host needs a route to another host and it does not have one in its cache, it dynamically determines one by flooding the network with route discovery packets, in a similar manner to LMR. The difference is that on the way back, reply packets cumulate the route to the destination. *route-maintenance* is the mechanism by which a packet's sender, S detects if the route to D is broken at some point. This is detected by a data packet that is not able to pass the hop. A route error packet is generated by a forwarding node that is not able to use a downstream link. To send this packet, the forwarding node will need a path to the source, or, it may use the reverse path from the data packet. On the way back the hop in error will be cut from all caches. S can then use another route to D from its cache, or invoke *route-discovery* again. Caches are organized as partial spanning trees, and they are incrementally updated based on information from route-discovery packets, route reply packets, and overheard packets caught in promiscuous mode. Entries in cache have associated expiration periods. In the following example, each tree entry in the cache is marked by a "*".

State of the caches after routes discovery:
$A \to D$, $B \to G$, $F \to G$

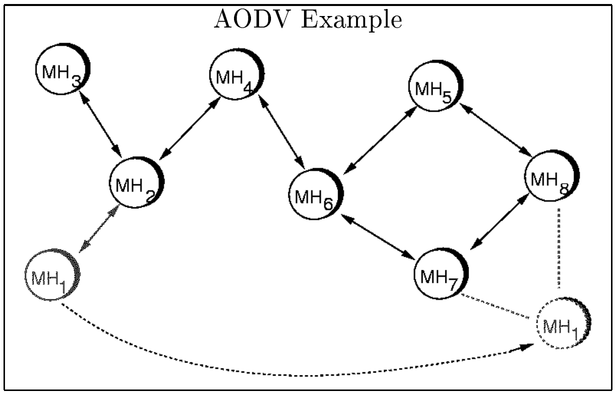


Advantages: this routing scheme is appropriate for communication with infrequent access destinations, when it does not pay to spend the overhead to maintain routes from all sources to such destinations (like in LMR). It does not use periodic routing advertisement messages, and it has low overhead provided that movement rate is low. Route caches are specific for the direction of communication in Figure 3, even if E is in the neighborhood of A, it does not need to store routes for B, C or D.

Drawbacks: The most important one is that DSR does not use multiple routes (although it could), which may lead to congestion and underutilization. Even in moderately dynamic networks DSR may result in a large communication overhead as it may have to invoke the *route-discovery* protocol often. It is not delay optimal because the cached routes continued to be used in spite of host movements and are removed only when a failure occurs. Control packets, which accumulate routes may impact the scalability.

4.4 Ad Hoc On-Demand Distance Vector (AODV)

AODV is the on-demand version of DSDV. It is using the same destination sequenced scheme that is ensuring loop-freeness, and eliminates DBF counting to infinity problem. AODV uses three types of packets: RREQ (Route Request), RREP (Route Reply) and MINV (Multicast Route Invalidation). RREQ *(src_ seqno, src_ ip, src_ hops, dest_ seqno, dest_ ip)* is broadcast when a node does not have a route to a given destination, or the route expired. *dest_ seqno* is the last sequence number route for that destination. A node receiving a RREQ message, may unicast back a RREP if it has a route to the requested destination and the sequence number for that route is greater than the sequence number of the request. Otherwise, it just rebroadcasts the request as if it didn't have the route. When a link breaks, a notification is broadcast to all neighbors using that route. The notification contains an ∞ metric and an odd timestamp.

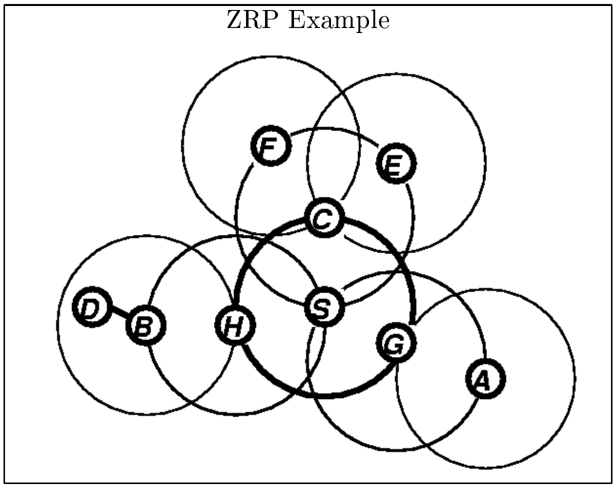

AODV Example

In this example, *MH1,* which has active connections with *MH3* and *MH6,* moves away from *MH2* in the vicinity of *MH7* and *MH8.* Noticing that its link with *MH1* is broken, *MH2* broadcasts an ∞ metric to *MH3* and *MH4.* *MH3* may subsequently issue a new route request for *MH1.* *MH4* also forwards the ∞ metric to *MH6*, which may later issue a route request for *MH1.*

5 Hybrid protocols

5.1 Zone Routing Protocol (ZRP)

Is an algorithm that takes advantage of both schemes - on demand, and global state: it maintains a global state in each node, but only for the circular zone surrounding the node. between zones, the algorithm uses on-demand strategies to find paths. The radius of the zone can be set to 1, which would lead to a completely on-demand algorithm, or can be set to the radius of the network, which would lead to a completely global state algorithm.


ZRP Example

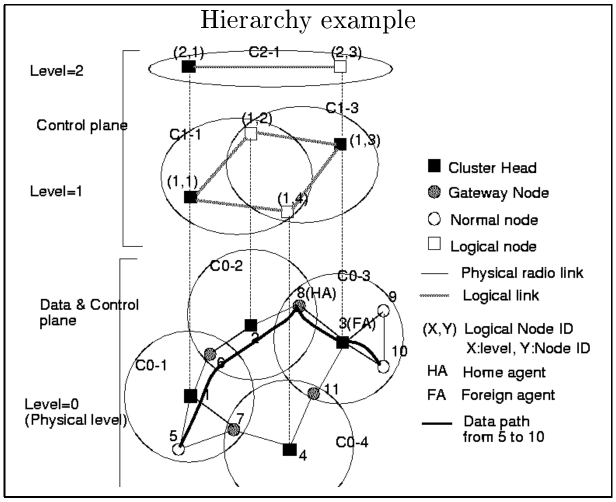
Example: S wants to send a message to *D*. First *S* verifies that *D* is not within its routing zone, then *S* sends a query message to all the nodes in the periphery of its routing zone - *G, G* and *H.* These nodes in turn broadcast the query to their peripheral nodes. In this example, *H* sends the query to *B*, which knows *D* to be in its routing zone. *B* then responds to the query, indicating the path *S-H-B-D.* The path is cumulated on the forward path with a mechanism similar to DSR - store the hops seen so far in the query message.

Advantages:

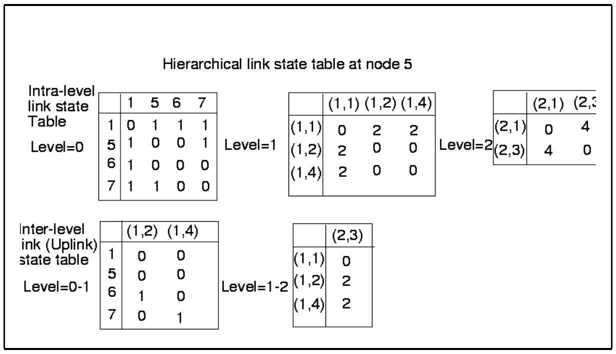
- discovers multiple routes;
- allows fine tuning, cumulating advantages from both categories
- ZRP can use any global state algorithm(or rather, a truncated version of it) to manage the zone.
- good scalability: limits the propagation of local changes in topology.

5.2 Hierarchical State Routing

HSR is a link state algorithm that tries to improve scalability by maintaining physical and logical clustering. Elected cluster heads at each level become members in the higher levels. The lowest is the physical level and clusters at this level are sized by the range of the radio link. Logical nodes within the cluster exchange logical link state and low level state information. Communication between logical nodes is established with tunnels between lower level clusters. Logical state of a node is continuously advertised to nodes in lower level cluster.


Hierarchy example

A node maintains state information for all the clusters it belongs to, at all levels. For example, node 5 would have states of the clusters $C0-1, C1-1, C2-1$. In fact, all nodes maintain both inter-level and intra-level information. For node 5, that would mean the link between levels $0-1$and $1-2$.



Hierarchical link state table at node 5

Intra-level link state Table

Level=0

	1	5	6	7
1	0	1	1	1
5	1	0	0	1
6	1	0	0	0
7	1	1	0	0

Level=1

	(1,1)	(1,2)	(1,4)
(1,1)	0	2	2
(1,2)	2	0	0
(1,4)	2	0	0

Level=2

	(2,1)	(2,3
(2,1)	0	4
(2,3)	4	0

Inter-level ink (Uplink) state table

Level=0-1

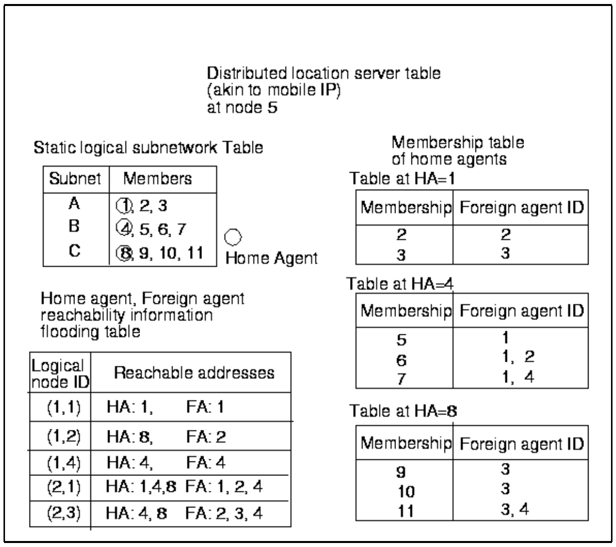
	(1,2)	(1,4)
1	0	0
5	0	0
6	1	0
7	0	1

Level=1-2

	(2,3)
(1,1)	0
(1,2)	2
(1,4)	2

Each node has a MAC address and an IP address in the form <*subnet, host*>. The subnet address identifies a group that may span over several physical clusters and has at least one (roaming) home agent. The home agent reachability is monitored by neighboring cluster heads which propagate this information along with the routing tables.

Foreign agents are the lower level cluster heads advertised by each destination to the home agent. Routing is done as follows: the source has the destination IP from which it extracts the subnet address in order to find the home agent. The home agent sets up a tunnel between the source and the foreign agent of the destination. If any intermediate node between the source and the home agent has information about the destination, it will forward the request directly to the foreign agent of the destination.



Distributed location server table
(akin to mobile IP)
at node 5

Static logical subnetwork Table

Subnet	Members
A	①, 2, 3
B	④, 5, 6, 7
C	⑧, 9, 10, 11

○ Home Agent

Home agent, Foreign agent reachability information flooding table

Logical node ID	Reachable addresses
(1,1)	HA: 1, FA: 1
(1,2)	HA: 8, FA: 2
(1,4)	HA: 4, FA: 4
(2,1)	HA: 1,4,8 FA: 1, 2, 4
(2,3)	HA: 4, 8 FA: 2, 3, 4

Membership table of home agents

Table at HA=1

Membership	Foreign agent ID
2	2
3	3

Table at HA=4

Membership	Foreign agent ID
5	1
6	1, 2
7	1, 4

Table at HA=8

Membership	Foreign agent ID
9	3
10	3
11	3, 4

Advantages: Has good scalability due to hierarchy - a logical node keeps only information related to its direct neighbors, and to the other logical nodes on the same level.

Drawbacks: Maintaining the hierarchy may incur high communication overhead. This drawback is augmented by high mobility.

6 Other Algorithms

- Location Aided Routing is an on-demand algorithm which uses information provided by a GPS (Global Positioning System) to reduce the area flooded with query requests [KV98].
- Fish-eye State Routing is an optimization of GSR that employs the fish-eye technique to reduce the level of detail at points that are far from a certain point of interest. The reduction is achieved by updating the network information for nearby nodes at a higher frequency than for remote nodes [CG98].
- Associativity Based Routing is a protocol that does not attempt to consistently maintain routing information in every nodes, but employs a new associativity-based routing scheme where a route is selected based on nodes having associativity states that imply periods of stability. In this manner, the routes selected are likely to be long-lived and hence there is no need to restart frequently, resulting in higher attainable throughput.

7 Comparison between the presented algorithms

Trying to answer the question “Which algorithm is the best?”, we must consider advantages drawbacks specific to each class of algorithms. On demand algorithms have lower memory requirements because routes are computed on the fly, although there are exceptions: DSR maintains trees with cached routes. On the other hand, their main drawbacks are flooding which may incur high overhead, and high latency in discovering routes. Also, on-demand algorithms, sometimes referred as “reactive” drop a considerable number of packets during route discovery - until a new route is found. If the traffic is not very high, these algorithms are better suited for higher rates of mobility than the global state algorithms. Global state algorithms have better responsiveness because the routes are precomputed, but have the problems of slow convergence, and poor scalability. [CG98] argues that QoS guarantees would be better supported by global state routing. Periodic updates may enhance the congestion when density is high. Hybrid algorithms (HSR and ZRP) combine the quality of the two classes and drop some of the drawbacks. ZRP for example is very flexible, in allowing the size of the zone to vary between 1 and the radius of the network. In these extreme cases, ZRP’s behavior is on demand when $r_{zone} = 1$, or global, when $r_{zone} = r_{net}$. Performance of ZRP depends heavily on chosen parameters (r_{zone}), but then the radius could be adjusted dynamically to adapt traffic and network conditions. Most routing algorithms, even those which find multiple routes do not try to adapt to traffic load. STARA makes an attempt to approximate the optimal routing using all the known paths to distribute the load such as to reduce the total transmission time, and improve utilization. Hierarchical algorithms (HSR)are better suited for large populations, when on-demand algorithms would have high latency and flood overhead, and global state algorithms would have high memory requirements.

[DCY97] and [BMJ98] present results of actual routing simulations that underline the strengths and the weaknesses of each algorithm with respect to performance parameters like: fraction of packets delivered, end-to-end delay, routing load and mobility. As expected, global state algorithms behave drop fewer packets than on-demand ones, mainly due to the availability of alternate paths. Surprisingly, TORA, which also provide multiple paths, does not perform as well in this respect, probably because of the longer route discovery process. [DCY97] mentions that, in case of on-demand algorithms, "early quenching" (a node that receives a query and knows a route to the destination will send it to the source) is not always performing well, and AODV is an example in which early quenching picks up stale routes. As for the end-to-end delay, the same study confirms that this parameter is higher for on-demand algorithms (they tested TORA, AODV, DSR, DSDV, SPF-Shortest Path First, and EXBF - Extended Bellman Ford). In terms of load (routing overhead), on demand algorithms perform better. TORA, in spite of the multi-path and on-demand features, performed poorly on all of the above tests. Tests in [BMJ98] revealed generally the same conclusions placing AODV and DSR above DSDV and TORA.

We are not aware about studies that would experimentally compare hybrid algorithms with the two categories - on demand and global. Also, these studies didn't take into account other important dimensions of the problem, like the size of the population and the amount of resources required at each node. This is the area in which hybrid protocols may have a significant advantage. In the context of a large population, ZRP would be able to face high mobility better than global state, or even than on-demand algorithms. This is because route discovery is shorter, by a factor of r_{zone}. On the other hand, if the mobility is low, HSR would have certain advantages for large populations of hosts, both in terms of latency (it is LS based) and scalability (it is cluster based). In these particular conditions, on-demand would have high message latency, while global algorithms state would have high memory requirements.

8 Conclusions

- Routing protocols for ad-hoc networks should react quickly to reestablish routes when topology changes. The issue of route optimality is of less importance than the one of route maintenance with low time and communication overhead. Existence of alternate paths is an advantage in a changing topology, and allows approximation of optimal routing.
- On-demand algorithms have more protocol overhead, but most of them find multiple paths, thus allowing better utilization of the bandwidth. They have an initial latency for route finding, therefore being less appropriate for interactive applications, but have better scalability because of the reduced size of the routing tables. They are better for topologies that change faster.
- Global state algorithms have less overhead, per message (or per session), because they tend to use the already created, slowly changing routes. On the other hand, due to periodic updates on the outgoing links, they may enhance congestion when density is high (degree of the graph is high). Global state routing maintains tables with all the nodes in the network, which makes it less scalable, but more responsive.
- Hybrid algorithms combine features from both classes, also diminishing the drawbacks of each class. They address issues of scalability and flexibility better than plain on demand or global.

References

[CMB96] S. Corson, J. Macker, S. Batsell, 1996, *Architectural Considerations for Mobile Mesh Networking*

[BG92] D. Bertsekas, R. Gallagher, *Data Networks*, Prentice Hall 1992

[MG96] Shree Murthy, J.J Garcia-Luna-Aceves, 1996, *An Efficient Routing Protocol for Wireless Networks*

[CG96] Tsu-Wei Chen, Mario Gerla, 1996, *Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks*

[PB94] Charles E. Perkins, Pravin Bhagwat, 1994, *Highly Dynamic Destination-Sequenced Distance-Vector Routing(DSDV) for Mobile Computers*

[PC97] Vincent D. Park, M.S. Corson, 1997, *A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Network*

[JM96] David B. Johnson, David A. Maltz, *Dynamic Source Routing in Ad-Hoc Networks*

[PR97] Charles E. Perkins, Elizabeth M. Royer,1997, *Ad Hoc On Demand Distance Vector (AODV) Routing.*

[Haas96] Zygmunt J. Haas, 1996, *A New Routing Protocol for the Reconfigurable Wireless Networks*

[BMJ98] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, Jorjeta Jetcheva, 1998, *A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols*

[GK98] Piyush Gupta and P.R Kunar, 1998, *A System and Traffic dependent Adaptive Routing Algorithm for Ad Hoc Networks*

[CG98] Tsu-Wei Chen, Mario Gerla et al., 1998, *Scalable Routing Strategies for Multi-Hop Ad-hoc Wireless Network*

[KV98] Young-Bae Ko, Nitin H. Vaidya, 1998, *Location-Aided Routing (LAR) in Mobile Ad GHoc Networks*

[DCY97] Samir R. Das, Robert Castaneda, Jiangtao Yan, 1997, *Comparative Performance Evaluation of Routing Protocols for Mobile, Ad-Hoc Networks*

[Toh97] C.-K. Toh, 1997, *Associativity Based Routing For Ad Hoc Mobile Networks*