Massachusetts Institute of Technology | Lecture 24
6.895: Advanced Distributed Algorithms | December 3, 2002
Professor Nancy Lynch

Routing and end-to-end communication

Papers

- Vincent D. Park and M. Scott Corson. A Highly Adaptive Routing Algorithm for Mobile Wireless Networks. In *Proceedings of IEEE INFOCOM*, Kobe, Japan, April 1997.
- Charles E. Perkins and Elizabeth M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, pages 90-100, February 1999.
- Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *ACM/Baltzer Wireless Networks (WINET)*, Vol.6-4, 2000.
- Young-Bae Ko and Nitin H. Vaidya. Geocasting in mobile ad-hoc networks: location-based multicast algorithms. In *Proceedings of the 2nd IEEE Workshop on Mobile Computer Systems and Applications, (WMCSA)*, New Orleans, LA, February 1999.

1 Design Strategies for Routing Protocols

1.1 Hierarchy Based Protocols

The best example of this is today's internet. Without hierarchy, Internet routing could not scale to support today's number of Internet leaf networks. Autonomous Systems run an internal protocol, internal Border Gateway Protocol (BGP), inside their borders and to communicate with each other employ an external protocol (BGP). This hierarchy is based on well defined and rarely changing administrative and topological boundaries. Tables of routing information are maintained in such protocols like Destination Sequenced Distance Vector (DSDV). It is therefore not easily applicable to freely moving ad-hoc wireless networks, where topology has no well-defined AS boundaries, and routers may have no common administrative authority.

1.2 Caching

Caching has come to prominence as a strategy for scaling ad-hoc routing protocols. Dynamic Source Routing (DSR), Ad-hoc On-Demand Distance Vector Routing (AODV), and the Zone Routing Protocol (ZRP) all avoid constantly pushing current topology information network-wide. Instead, routers running these protocols request topological information in an *on-demand* fashion as required by their packet forwarding load, and cache it aggressively. When their cached topological information becomes out of date, these routers must obtain more current topological information to continue routing successfully. Caching reduces the routing protocols' message load in two ways:

- It avoids pushing topological information where the forwarding load does not require it (e.g. at idle routers)

- It often reduces the number of hops between the router that has the needed topological information and the router that requires it (i.e., a node closer than a changed link may already have cached the new status of that link).

The routing algorithms discussed today are essentially applicable to three kinds of networks:

- Ad-hoc networks: These mobile nodes have no fixed infrastructure.
- Sensor networks: Comprised of small sensors, these mobile networks can be deployed with a very large number of nodes, and have very impoverished per-node resources.
- "Rooftop" networks: These wireless networks are not mobile, but are deployed very densely in metropolitan areas as an alternate to wired networking. A self-configuring routing system for hundreds of thousands of such nodes in metropolitan areas is a significant scaling challenge.

2 Temporally Ordered Routing Algorithm - TORA

2.1 Overview:

The protocol can be divided into three basic functions:

- **Creating Routes:** Creating a route from a given node to destination essentially corresponds to assigning directions to links in an undirected portion of the network. A query/reply procedure described later builds a DAG rooted at the destination. Uses packets of type query (QRY) and update (UPD).
- **Maintaining Routes:** A version of GB algorithm used here to bring DAG to again be rooted at the destination. The algorithm also detects network partitions. Implemented by UPD packets.
- **Erasing Routes:** Upon detection of a network partition, all links, in the portion of the network which has become partitioned from the destination, must be undirected, to erase invalid routes. Clear (CLR) packets used for erasing routes.

2.2 Basic Idea:

2.2.1 Creating Routes:

When a node with no directed links and with no pending route request, requires a route to the destination, it broadcasts a QRY packet.

- If the receiver also has no route to the destination and has no pending request, it rebroadcasts QRY.
- If the receiver also has no route to the destination and has a pending request it discards the QRY.
- If the receiver has at least one downstream node and its own height is NULL, it makes its height 1 more than the minimum height of that of its downstream links and broadcasts an UPD packet.

- If the receiver has at least one downstream node and its own height is not NULL, it broadcasts an UPD packet if one has already not been broadcast. If an UPD has already been broadcast over that link, then the QRY is discarded.
- If a new link appears while a node has a pending QRY, it broadcasts a QRY packet again.
- When a node gets an UPD, it updates that neighbor's height information.
  1. If the node has height NULL, then it makes its height 1 more than the minimum height of that of its downstream links and broadcasts an UPD packet.
  2. If it already has its height, then it updates the link state information.

2.2.2 Maintaining Routes:

The basic idea of route maintenance is as follows: (Please look at Fig. 2 in the Park Corson paper)

- **Generate:** When a node loses last downstream link (i.e. becomes a local minimum) as a result of a link failure, the node selects a new height such that it becomes a *global* maximum by defining a new "reference level". This action results in link reversals which may cause other nodes to lose their last downstream link.

- **Propagate:** Any such node executes a partial reversal with respect to its neighbors that have heights already associated with the newest (height) reference level. In this manner, the new reference level is propagated outward from a point of original failure. This propagation will only extend through nodes which have lost all routes to the destination.
- **Reflect:** Any node which initially had all downstream links and now experiences link reversals from all neighbor nodes, chooses a new height to become a local maximum. This is accomplished by defining a higher sub-level associated with the new reference level. This node essentially "reflects" this higher sub-level back toward the node which originally defined the new reference level.
- **Detect:** Should this reflected reference level be propagated back to the originating node from all of its neighbors, then it is determined that no route to the destination exists.
- **Generate:** If a node gets reflects from all neighbors, but they have references not originating from it then it starts with a new reference level. This happens when there is a link failure which did not require reaction between the time UPD is propagated and the reflect returns. However, now as reflects have come from every neighbor, a new route needs to be established.

2.2.3 Erasing Routes:

Following detection of a partition, a node sets it height and that of neighbors to NULL (except if destination is neighbor in which case its height is 0) and broadcasts a CLR packet. The CLR contains the destination-id and reflected reference level of the node. The nodes which have the same reference level change their height to NULL but other nodes do not. Thus only invalid routes are erased.

2.3 Examples

(Please look at Fig. 1,3,4,5 in the Park Corson paper)

2.4 Details

Each node has a quintuple associated to it for reference (first 3) and height(last 2 parameters):

- Time Tag: Time of link failure and initialization of new reference.
- Originator ID: of the node which defines the new reference level.
- r: To decide whether it is a basic reference or a reflected sub-level.
- δ : Height of the node
- i : UID of the node.

Each node also maintains Link State Information for each link (Upstream or Downstream) depending upon the height information of the neighbors which is also maintained. For NULL heights (initialization) the links are considered undirected.

3 Ad-hoc On-Demand Distance Vector Routing - AODV

3.1 Overview

AODV uses a broadcast route discovery mechanism as is also used by DSR and TORA. Instead of source routing however, AODV relies on dynamically establishing route table entries at intermediate nodes. AODV has the following sub-protocols:

3.1.1 Path Discovery:

(Please see Figures 1 and 2 in the AODV paper by Royer and Perkins) When a source node needs to communicate with another node for which it has no routing information in its table, it initiates path discovery by broadcasting a route request (RREQ). Each neighbor either sends a route reply (RREP), or rebroadcasts the RREQ after increasing hop count of RREQ. If a node gets the same request from two neighbors, it drops the later RREQ. If a node cannot satisfy the RREQ, it keeps track of the query and caches the information when an eventual RREP passes via it.

Reverse Path Setup:

As the RREQ travels from a source to various destinations, it automatically sets up the reverse path from all nodes back to the source. For this, a node records address of the neighbor from which it received the first copy of RREQ.

Forward Path Setup:

Eventually a RREQ will arrive at a node, that possesses a fresh route, and if the RREQ has not been processed previously, the node then unicasts a route reply packet (RREP) back to its neighbor from which it received the RREQ.

As the RREP travels back to the source, each node along the path sets up a forward pointer to the node from which the RREP came. Nodes that are not along the path determined by the RREP will timeout and will delete the reverse pointers.

A node receiving an RREP propagates the first RREP and later RREPs for that source only if the route is shorter or the destination information is more recent.

3.1.2 Route Table Management

The reverse routes which do not lie from source to destination are timed-out depending upon a *route request expiration timer*. Routing entries also time out depending upon the *Route Caching Timeout*. A neighbor is considered active *active* if it originates or relays at least one packet for that destination within the *active_timeout* period. This information is maintained so that all active sources can be notified when a link along the path to the destination breaks. If a path does not transmit data for some time, the entry for that link times out.

A mobile node maintains a route table entry for each destination with the following information:

- Destination
- Next Hop
- Number of Hops (metric)
- Sequence number of the destination (Similar to Time Tag to choose Latest information about a node)
- Active neighbors for this route
- Expiration time for the route table entry

3.1.3 Path Maintenance

The following things are done to maintain routes:

- If the source node moves during an active session, it re-initiates route discovery.
- When destination or intermediate node moves, a special RREP is sent to the affected source nodes.
- Periodic *hello* messages to detect link failures.
- Once the next hop becomes unreachable, the node upstream of the break propagates an unsolicited RREP with a hop count of ∞. This message is relayed to the active neighbors and so on.
- Upon receiving notification of a broken link, source node can restart the discovery process.

3.1.4 Local Connectivity Management

Nodes learn of their neighbors in one of two ways:

- Whenever a node receives a broadcast from a neighbor, it updates local connectivity information to ensure that it includes this neighbor.
- If a node has not sent any packets to all of its downstream neighbors, it broadcasts a *hello* message (a special unsolicited RREP).
- If a stipulated number of consecutive *hello* messages are not received from active neighbors, then a notification of link failure is sent as described in Section 3.1.3.

3.2 Details:

A RREQ contains the following fields:

- Source Address
- Source Sequence Number : To decide upon the latest information about source (similar to logical time).
- Broadcast ID : To differentiate between RREQ's.
- Destination Address
- Destination Sequence Number
- Hop Count: The number of links traversed, increased by 1 each time a new node rebroadcasts it.

A RREP contains the following information:

- Source Address
- Destination Address
- Destination Sequence Number
- Hop Count
- Lifetime: Nodes not on the source to destination route will time out after this time.

4 Location-Aided Routing (LAR)

The previous MANET (Mobile Adhoc NETwork) routing protocols do not take into account the physical location of a destination node.

4.1 Preliminaries

Location information used in LAR protocol may be provided by Global Positioning System (GPS). This paper assumes that the mobile nodes are moving in a two dimensional plane.
Expected Zone This is the region in which node S expects to find node D at time t_1 if it knew the location of D at time t_0. The size of the zone is dependent upon:

- The average speed (or some other measure of speed distribution) of D.
- The time elapsed since S knew of the precise location of D.

If node S does not know a previous location of D, then the algorithm reduces to a basic flooding mechanism.
Request Zone Node S needs to define a request zone for the route request. A node forwards a route request *only if* it belongs to the request zone. The request zone needs to include the *Expected Zone* and a path to the expected zone. Request Zones are increased in coverage if no path is found in the previous search. To decide upon an adequate request zone so that at least a path from S to

D is included is a trade-off between latency of route determination and message overhead. (Fig 3 of LAR paper).

Thus implementing LAR algorithm requires that a node be able to determine if its is in the request zone for a particular request - the two LAR algorithms present here take different approaches for that:

4.2 LAR Scheme 1

(Fig 4 from LAR Paper) The first scheme uses a request zone that is rectangular in shape. Assume that node S knows that D was at location (X_d, Y_d) at time t_0. At time t_1, node S initiates a new route discovery for destination D. Lets assume S also knows the speed v with which D can move. So node S defines the expected zone at time t_1 to be the circle of radius $R = v(t_1 - t_0)$ centered at location (X_d, Y_d).

In this approach, the request zone is defined as the smallest rectangle that includes current location of S and the expected zone such that the sides of the rectangle are parallel to the X and Y axes. S includes their coordinates with the route request message transmitted when initiating route discovery. When a node receives a route request, it discards the request if the node is not within the rectangle specified by the four corners included in the route request, otherwise it forwards it. When D receives the route request message, it replies by sending a route reply message (as in the flooding algorithm). However, in case of LAR, node D includes its current location and current time in the route reply message. When node S receives this route reply message, it records the location of node D.

In general, a smaller request zone may occur at speeds that are neither too small, nor too large. This is because of the dependence of request zone size on elapsed time and velocity of nodes.

4.3 LAR Scheme 2

(Please see Figure 5 of LAR paper) In this scheme, S includes two pieces of information with its route request:

- Assume that node S knows the location (X_d, Y_d) of node D at some time t_0. Node S calculates its distance from location (X_d, Y_d), denoted as $DIST_s$ and includes this distance with the route request message initiated at time t_1.
- The coordinates (X_d, Y_d) are also included with the route request.

When a node I receives the route request from sender node S, node I calculates its distance from location (X_d, Y_d), denoted as $DIST_i$, and for some parameter α and β:

- If $\alpha(DIST_s) + \beta \geq DIST_i$, then node I forwards the request to its neighbors. Node I also replaces $DIST_s$ with $DIST_i$.
- Else $\alpha(DIST_s) + \beta < DIST_i$. Then node I discards the route request.

Thus a node forwards a route request if it is "closer" to or "not much farther" from (X_d, Y_d) that the previous node. Nonzero α and β may be used to trade-off the probability of finding a route on the first attempt with the cost of finding the route.

4.4 Variations and Optimizations

1. **Alternative definitions of request zone** One may define larger rectangles, or remove the condition that the sides of rectangle are parallel to X and Y axes. (Figure 14 of LAR paper)
2. **Adaptation of request zone** More accurate information about D can be obtained by intermediate nodes. So the intermediate node might change the request zone with the help of newer information.
3. **Another adaptation of request zone** Thus any shape for request zone can be used. An interesting one could be the cone-shaped zone as in Figure 15 of the LAR paper.
4. **Propagation of location and speed information** The nodes between source and destination can also cache the location and speed information passed through them for future reference.
5. **Local search** If any intermediate node detecting a route error is allowed to initiate a route discovery, a smaller request zone would be required to search for a functioning route.
6. **Combining with Time-to-live (TTL)** By setting the TTL to some reasonable number, the source can bound packet propagation geographically and by hops.
7. **Use of directional antennas** Packets can be sent only towards nodes which are in the request zone without broadcast if directional antennas are used.
8. **Clock Synchronization** LAR Scheme 1 needs synchronized clocks. This can be avoided by using local clocks which timestamp packets before communication. This would help to set up a logical clock.

5 Geocasting in Mobile Ad-hoc Networks

This paper uses completely the same ideas as the LAR paper. In this case, the request zone is called the forwarding zone. The expected zone is considered to be the multicast zone (where each node needs to get the packet).