Selective Flooding in Ad Hoc Networks

by

Ming-Yee Iu

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Mathematics

in

Computer Science

Waterloo, Ontario, Canada, 2002

Abstract

An ad hoc network is a collection of mobile wireless devices that cooperate with each other to route packets amongst themselves. The main difficulty in designing routing algorithms for such a network is the large number of topology changes that the network undergoes due to device movement.

Selective flooding is a routing technique that is more resilient to topology changes than traditional algorithms but is more bandwidth efficient than pure flooding. An on-demand selective flooding algorithm has been designed and tested on the ns-2 simulator. In scenarios involving a large number of topology changes, selective flooding outperforms other ad hoc network routing techniques. Unfortunately, selective flooding is much more bandwidth hungry and is unable to scale to handle reasonable traffic loads.

Interestingly, the analysis of selective flooding reveals major problems with traditional ad hoc networking techniques. Many current algorithms demonstrate shortcomings when dealing with bursty traffic, and current wireless hardware cannot handle ad hoc networking traffic in an efficient manner. These issues need to be addressed before ad hoc networking technology can become feasible for widespread use.

Acknowledgements

I would like to thank the Natural Sciences and Engineering Research Council of Canada for supporting my research through a Postgraduate Scholarship, and I would also like to express my gratitude to the Nortel Networks Institute and the Ontario Graduate Scholarship in Science and Technology for their additional assistance. The University of Waterloo also supported my research through a University of Waterloo Graduate Scholarship.

I would also like to thank my supervisor and readers—Gordon Cormack, Raouf Boutaba, and Johnny Wong.

Contents

List of Figures

Chapter 1

Introduction

Over the past decade, wireless technology has improved at a dramatic rate. The ubiquity, low cost, and high functionality of the newest wireless devices have opened the door for researchers to consider new ways of using this technology. One of the most promising such areas being researched today is ad hoc networking. Ad hoc networking allows wireless mobile devices to form networks without the need for additional infrastructure or centralized management. Unfortunately, routing data in such a network is extremely difficult, requiring specialized routing algorithms. Selective flooding is an approach to routing that is simpler to implement and potentially more suitable to the dynamic topologies of ad hoc networks than existing techniques.

Current wireless networks (Ramanathan and Steenstrup 1996) are designed with the assumption that wireless devices have little computing power. As such, the devices are not designed to be able to communicate directly with or to coordinate their activities with other wireless devices. Instead, powerful immobile base stations supervise all communication. These base stations (figure 1.1), typically in the form of radio towers, must be erected throughout areas where wireless coverage is desired. The region surrounding a base station is called a cell, and each base station is responsible for all communication that occurs within its cell. Base stations detect all wireless devices within their cells and configure them as to when and how they can communicate. When one wireless device wants to send data to another, it will send the data to the nearest base station instead. The base station will then forward this data across wired links to the base station closest to the destination wireless device. That final base station will then transmit the data to the destination wireless device. Because current networks require a large number of base stations with interconnecting wired links, they can be expensive to construct and maintain.

By contrast, ad hoc networks require no such infrastructure. In a pure ad hoc network abstraction (figure 1.2), all wireless devices in the network are considered to be equally powerful nodes in the network. There are no base stations to coordinate activities in the network. The nodes themselves are responsible for coordinating between themselves to start and maintain the network. When one node wants to communicate with another, the other node might be outside its broadcast radius, so it must cooperate with other nodes to have them act as intermediaries and forward data towards the destination. The choice of intermediaries will change constantly because the network topology is dynamic with nodes always in movement.

The advantages of such a network over traditional networks is that ad hoc networks can be used in locations where no wireless infrastructure exists. Military and disaster-relief applications are particularly well-suited to them. The lack of

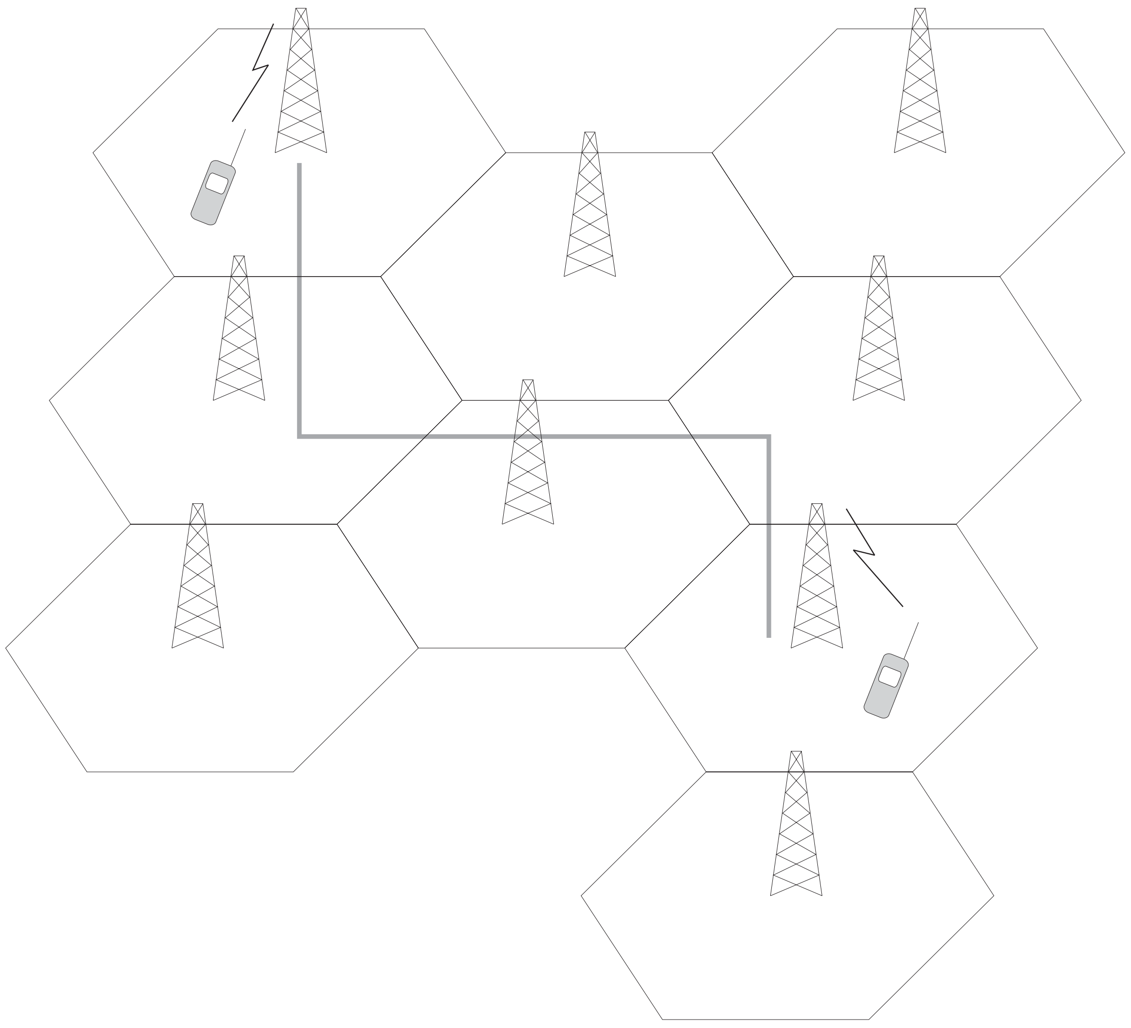


Figure 1.1: Current mobile networks rely on base stations and cells

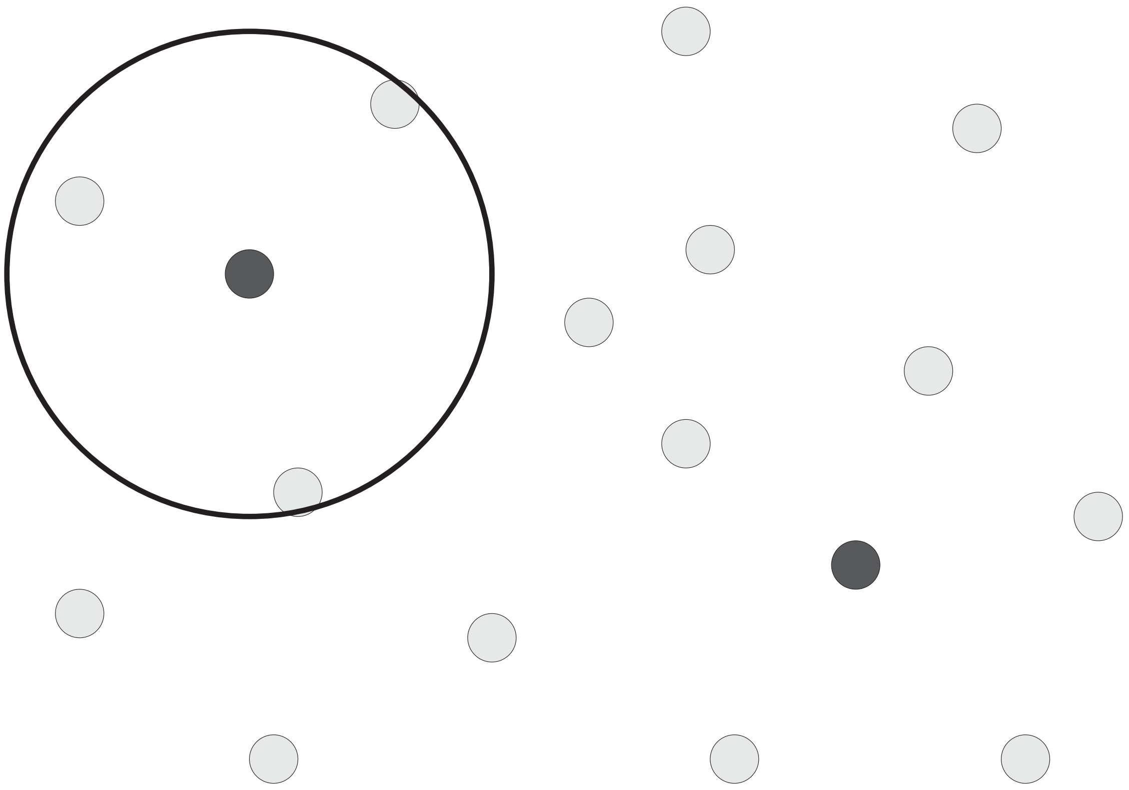


Figure 1.2: Ad hoc networks are composed of nodes with broadcast ranges that are too short to reach all the other nodes in the network

infrastructure in an ad hoc network also holds the promise of users being able to create free, public wireless networks. Since ad hoc networks are self-starting and self-maintaining, users can create and join such networks without having to pay fees to cover infrastructure costs.

Although much research has been devoted to ad hoc networks, the central problem in ad hoc networking—how does a node coordinate with others nodes to find others and to route data to them despite a dynamic network topology—has not been solved in a scalable and robust fashion. Even though some toy implementations have been constructed, no real usable networks have been made.

This thesis examines current ad hoc network routing techniques and then proposes a new approach based on selective flooding. While comparing the algorithm to others, the thesis uncovers some interesting properties of ad hoc networks in general.

Chapter 2

Current Routing Techniques

Although the field of ad hoc networking is relatively young, much of the field is influenced by the more mature areas of packet radio and the Internet. In both those areas, solutions to the problems of routing and network maintenance have been developed and implemented. Unfortunately, those solutions are not directly applicable to ad hoc networks because the high mobility of nodes in an ad hoc network results in a network topology that is too unstable for traditional techniques to be effective.

Routing protocols for mobile ad hoc networks, by contrast, are specifically designed to deal with extremely dynamic network topologies. Nodes detect topology changes quickly and rapidly disseminate this information to affected parts of the network, allowing for a swift reaction. Because this dissemination may put an excessive strain on network resources, many of these routing algorithms attempt to localize the effect of topology changes and to minimize the amount of information that must be disseminated. Already, a large number of ad hoc network routing protocols have been developed, and researchers continue to refine existing techniques and develop new ones.

Understanding the current state of ad hoc network routing requires an examination of existing Internet protocols, early packet radio protocols, and a selection of ad hoc networking protocols.

2.1 Distance Vector

The distance vector algorithm is a routing algorithm used in the Internet for routing within subnets (Comer and Droms 1999). Although it cannot be used directly in ad hoc networks, many ad hoc networking algorithms are derived from it.

It is based on the distributed Bellman-Ford algorithm, an algorithm for finding shortest paths between vertices in a mathematical graph. A network is converted to a graph by changing every router and device in the network into a vertex and drawing an edge between vertices if the corresponding network devices have a communication link between them.

Distance vector routing uses next-hop forwarding where each router in the network only maintains information about the next hop a packet should follow along its path to its destination. Each router has a routing table listing which link packets for a given destination should be forwarded along and the minimum distance that packets must travel to reach that destination. Table entries exist for all possible destinations. Initially, the table lists every destination as being an infinite distance away. The distance to directly connected routers is known, so entries for them are then updated with this information. Periodically, all routers will send a copy of their table to their neighbours. Upon receiving a copy of a neighbour's routing table, a router can update its own routing table to reflect the new topology information that it has received. By examining each entry of the neighbour's table, a router can determine whether it would be more efficient to forward packets meant for a given destination through that neighbour or to keep forwarding packets along the route listed in its own routing table. The router will revise its table to incorporate any improved routes.

Through continual updating, the routing table entries will converge to a point where they all contain the optimal paths for forwarding packets in the network. Although this convergence occurs relatively quickly and routers respond quickly to the creation of new shorter routes, the distance vector algorithm reacts poorly to link breakages (Tanenbaum 1996). When a link breaks in the network, routers that are adjacent to the link will adjust their routing tables so that any route that uses that link will be set to a distance of infinity. Ideally, this information will be disseminated throughout the network and routers will find alternate routes. Unfortunately, routers can receive routing tables from their neighbours listing an apparent alternate route that, in reality, also goes through the link that has just been broken. Because of this problem, routers in a network can often take considerable time to find new routes after a link breaks. Even worse, during this convergence period, the routes specified in the routers' tables are often completely invalid, specifying loops or other incorrect routes.

In wireless networks, a link is considered to exist between two nodes when they are within broadcast range of each other. Since the high node mobility in an ad hoc network results in many link breakages, the distance vector algorithm is inappropriate for use in an ad hoc network.

2.2 DARPA Packet Radio

Much of the foundation and initial research for current ad hoc networks was developed for the Defense Advanced Research Projects Agency (DARPA) for military packet radio (Jubin and Tornow 1987). The creation of the DARPA Packet Radio Network (PRNET) provided invaluable experience and data about the issues involved in wireless networks. Although much of the research was focused on hardware topics, discoveries were also made in routing.

The PRNET uses an algorithm similar to the distance vector algorithm for its routing though with modifications to prevent the spreading of incorrect routing information or the creation of routing loops after a link breakage. It also contains a mechanism for assessing link quality and avoiding the use of links that cannot ensure reliable packet delivery. One novel feature of the routing in the PRNET is that if a network node is unable to forward a packet to a next hop due to link failure, it will initiate a small flood of the packet to any node that is at the same distance or closer to the destination than itself. Hence, even if a link breakage occurs and node routing tables have not converged to a new route, network communication can still continue.

2.3 DSDV

The Destination-Sequenced Distance-Vector (DSDV) protocol (Perkins and Bhagwat 2001) is one of the earliest ad hoc networking protocols. It attempts to adapt distance vector routing for ad hoc networking by improving its handling of high node mobility and link breakages.

In a network with high node mobility, links between nodes are constantly being created and destroyed. Unlike a classical distance vector algorithm where topology information is only updated when nodes periodically exchange routing table information, DSDV propagates information about new and invalid routes as soon as they are detected. To prevent storms of topology change packets, nodes may delay propagating topology change information if they expect to receive alternate topology data soon after.

DSDV also adds a sequence number to each routing table entry to prevent routing loops from forming in the network. By examining this sequence number, nodes can compare the staleness of its own routing information with that of incoming routing information; therefore, old routing data describing broken routes cannot overwrite new routing data that shows that the route is invalid.

Although DSDV can route data in an ad hoc network, it is considered to be less efficient and less reliable than newer algorithms.

2.4 DSR

Dynamic Source Routing (DSR) (Johnson, Maltz, and Broch 2001), another early algorithm, was inspired by the Address Resolution Protocol though similar routing algorithms for packet radio had been developed earlier in 1978 (Kahn et al. 1978). Unlike other algorithms where nodes must periodically exchange topology information for the entire network (referred to as a table-driven or proactive approach), DSR nodes only exchange topology information for active communication links (referred to as an on-demand or reactive approach). In particular, if no nodes within a network are communicating, no control packets need be exchanged anywhere in the network.

In DSR, the path that a packet must travel to reach its destination is explicitly encoded in the packet itself. As such, routing packets is easy because nodes merely forward packets to the next hop specified in the packet. But sending packets requires more work because a sender must somehow find a route to the desired destination in order to create the packets. To find a route, a sender floods the entire network with a packet requesting a route to the destination. As this request is propagated, the route it follows is recorded in it. Eventually, the destination will receive a copy of the request, and it will send a reply packet back to the sender that contains the route from the request packet. This reply can follow the route taken by the request packet in the reverse direction, or the destination can initiate a search for the sender (which is required if the links between nodes are unidirectional).

Once a sender knows of a route, it can send packets to the destination. Each node along the route is responsible for forwarding packets to the next hop on the route and must verify the receipt of packets using acknowledgments. If after several retransmissions, a node is unable to verify receipt of a packet, it will assume that the link is broken and will inform the sender of the packet that the route is invalid. The sender will then have to reflood the network with a new route request packet. Interestingly, since a route is never changed unless a link breakage occurs, DSR ignores newer, shorter routes after an initial route has been chosen.

This basic algorithm can be inefficient if many link breakages occur resulting in many network floods. To counter this problem, DSR includes many optimizations for storing alternate routes, caching routes to alternate destinations, and managing potential storms of packet exchange.

DSR is the quintessential example of an on-demand ad hoc network routing algorithm. All on-demand routing algorithms tend to exhibit properties similar to those of DSR. Because on-demand algorithms only exchange control packets for active links, they are considered more efficient than table-driven algorithms and are preferred by the ad hoc networking community today. Many table-driven algorithms have been reformulated as on-demand algorithms. In particular, DSDV has an on-demand counterpart called the Ad Hoc On-Demand Distance-Vector protocol (AODV) (Perkins and Royer 2001).

2.5 TORA

Although algorithms such as DSDV, DSR, and AODV are able to route packets efficiently in small, mostly static networks, different techniques are needed for larger, more dynamic networks. Topology changes occur more often, and if information about these changes need to be disseminated throughout the network, the network will quickly become flooded. Instead, algorithms need to minimize the amount of topology information that must be exchanged to react to one of these changes.

Although several such techniques based on clustering, landmarks, and other methods exist, the Temporally Ordered Routing Algorithm (TORA) (Corson and Park 2001) uses a particularly interesting approach. TORA belongs to a class of routing algorithms known as link reversal algorithms, which are designed to perform well in networks that are too unstable for traditional shortest path routing algorithms. Instead of trying to create optimal paths, TORA merely creates directed acyclic graphs (DAGs) indicating the directions that packets must travel along links to eventually reach their destinations. Because a rooted DAG for a particular destination will always end at that destination, a node routes packets by forwarding them along any downward link, and the packets will eventually reach their destinations. This DAG structure also has the interesting property where if a link is broken, there are often several alternate downward links available at the node where the link breakage occurred providing redundancy, and if all the downward links are broken, compromising the DAG structure, reversing the direction of the links in the vicinity of the link breakage is often sufficient to restore the structure (figure 2.1).

In TORA, each node stores a measure of its "distance" to different destinations. Communication links point downwards from nodes with a higher distance to nodes with a lower distance. This distance measure has no relation to the actual physical distances between nodes but is a completely artificial construction composed of five

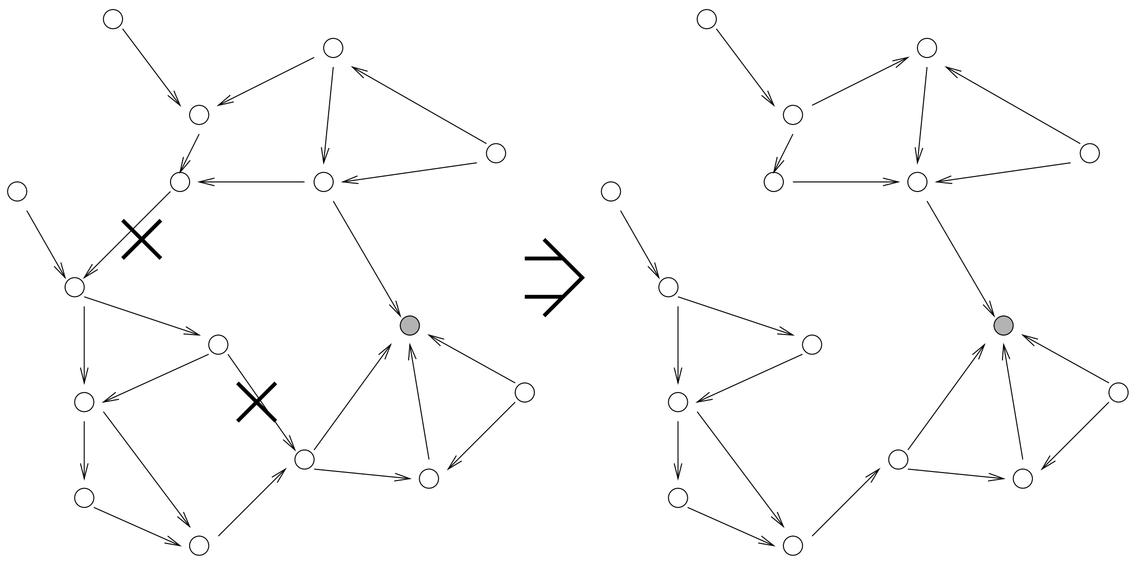


Figure 2.1: TORA can recover from link failures by using redundancy and link reversals

values including time, base level, relative level, and node id, which can be adjusted quickly to reverse link directions and cancel invalid paths. A fairly involved process is used to set-up the initial distance values and to maintain them.

The algorithm does not guarantee optimal routing paths, but the unique design of TORA allows it to quickly compensate for changes in network topology without the need for costly route searches. TORA is a good example of the types of approaches that can be used to minimize the effect of topology changes in large dynamic networks.

2.6 Flooding

The topologies of some networks are so dynamic and unstable that no reliable routes can be found in the network. Although not normally considered to be a routing algorithm, network flooding is often the only viable routing option in these situations. It has one interesting property that distinguishes it from other routing protocols: it is able to route packets without any knowledge of the network topology, so no control packets ever need to be exchanged in the network.

In flooding, when a node wants to send a packet to a destination, it broadcasts the packet to all of its neighbours. Whenever any node receives any packet, it simply rebroadcasts it. This process continues until the packet has been rebroadcasted throughout the entire network, resulting in every node in the network receiving a copy of the packet. To prevent the packet from being rebroadcasted forever, nodes do not rebroadcast packets that they have already seen before. Often a random delay is inserted between packet broadcasts to lower the potential of packet collisions from two nodes receiving and rebroadcasting the same packet simultaneously.

Despite its poor scalability and inefficient bandwidth usage, flooding is useful to study because it is an extreme example of how to minimize the effect of topology changes on routing. By completely eliminating any reliance on knowledge about the network topology, flooding performance does not degrade with increased node mobility.

2.7 Other Algorithms

Many other ad hoc networking protocols have been developed (*Mobile Ad-hoc Networks Charter* 2002), but they generally share many characteristics with the routing algorithms described in this chapter. Certain specialized routing protocols do exist that are optimized for different situations such as improving battery life (Singh, Woo, and Raghavendra 1998) or improving scalability (Steenstrup 2001). In all of these algorithms though, the primary difficulty in the design of the algorithms is still in efficiently handling changes in network topology.

Chapter 3

Selective Flooding

Although many different ad hoc network routing algorithms exist, few are suitable for use in networks with highly mobile nodes, and those that are tend to be fairly complex. Selective flooding holds the potential of being simple to implement and well-suited to highly unstable networks yet sufficiently efficient in bandwidth usage to scale to reasonably large networks. It works by restricting the scope of network floods to only those parts of the network where packet destinations are likely located. This chapter examines the rationale for building an algorithm based on selective flooding, describes one technique for controlling the scope of a network flood, and develops two approaches for actually implementing the technique.

As noted in the previous chapter, the main difficulty in developing ad hoc network routing algorithms is in maintaining up-to-date topology information at each node. After a link breakage, nodes must be informed of this event lest they try to route packets through the link. Although many algorithms attempt to lower the amount of topology information that each node must be aware of or try to route data through more stable parts of the network, these algorithms are still ultimately subject to the constraint that if the topology information in the network becomes stale, they cannot route packets and must spend time probing the current topology.

By contrast, flooding is able to route packets in a network without any knowledge of the network topology. Although it is effective in networks where the topology changes too quickly for traditional ad hoc network routing algorithms, network flooding uses far too much bandwidth to be practical for any large-scale network. The fact that network flooding and traditional routing techniques occupy opposite ends of the spectrum in terms of the amount of topology knowledge they need to route packets suggests that there might be an intermediate algorithm that can route packets using stale or incomplete topology information but is more bandwidth efficient than network flooding.

To route packets efficiently, network nodes need some knowledge about the topology of the network, but information about actual routing paths through the network is too specific to allow for resilience to stale topology information.

One alternative is for nodes to know their geographical coordinates (Ko and Vaidya 1998) and to use this knowledge as a substitute for topology information. Unfortunately, this information is usually obtained through the global positioning system (GPS), which adds to the cost of nodes, may not be available in certain military scenarios, and may not provide the position accuracy needed. An additional problem is that routing based on geographical coordinates cannot handle certain network topologies such as nodes arranged vertically in a building or nodes arranged in a circle around the base of a large mountain.

A better alternative is for nodes to store only their distances to other nodes. In such a situation, a packet traveling along the optimal path to its destination will pass through nodes where each successive node is one hop closer to the destination than the previous node.

In figure 3.1, each node knows its distance from node B. The number beside each node is the distance in hops that the node is from node B. If node A wants to send a packet to node B, the optimal route for the packet is indicated by the arrows. Note that the distance of each node along the optimal route is one less than the distance of the previous node along the route.

This observation suggests a possible approach for routing packets given that this limited topology information is available. If every network packet has a hop count entry, then when a node sends a packet, it should set the hop count of the packet to be its own distance from the packet's destination. An intermediate node receiving a copy of the packet should forward the packet on to other nodes if the node has a hop count smaller than that of the packet. The intermediate node should also decrement the hop count of the packet before forwarding it. If this process continues, the packet will eventually reach its destination. Figure 3.2 gives a more detailed description of the algorithm.

This approach to routing packets has some interesting properties. Most importantly, this algorithm routes packets along the optimal path to their destinations. In fact, if more than one optimal path exists between a sender and destination, a packet will travel along all of these paths simultaneously.

Figure 3.3 shows an example of this behaviour. In the figure, Node A is sending

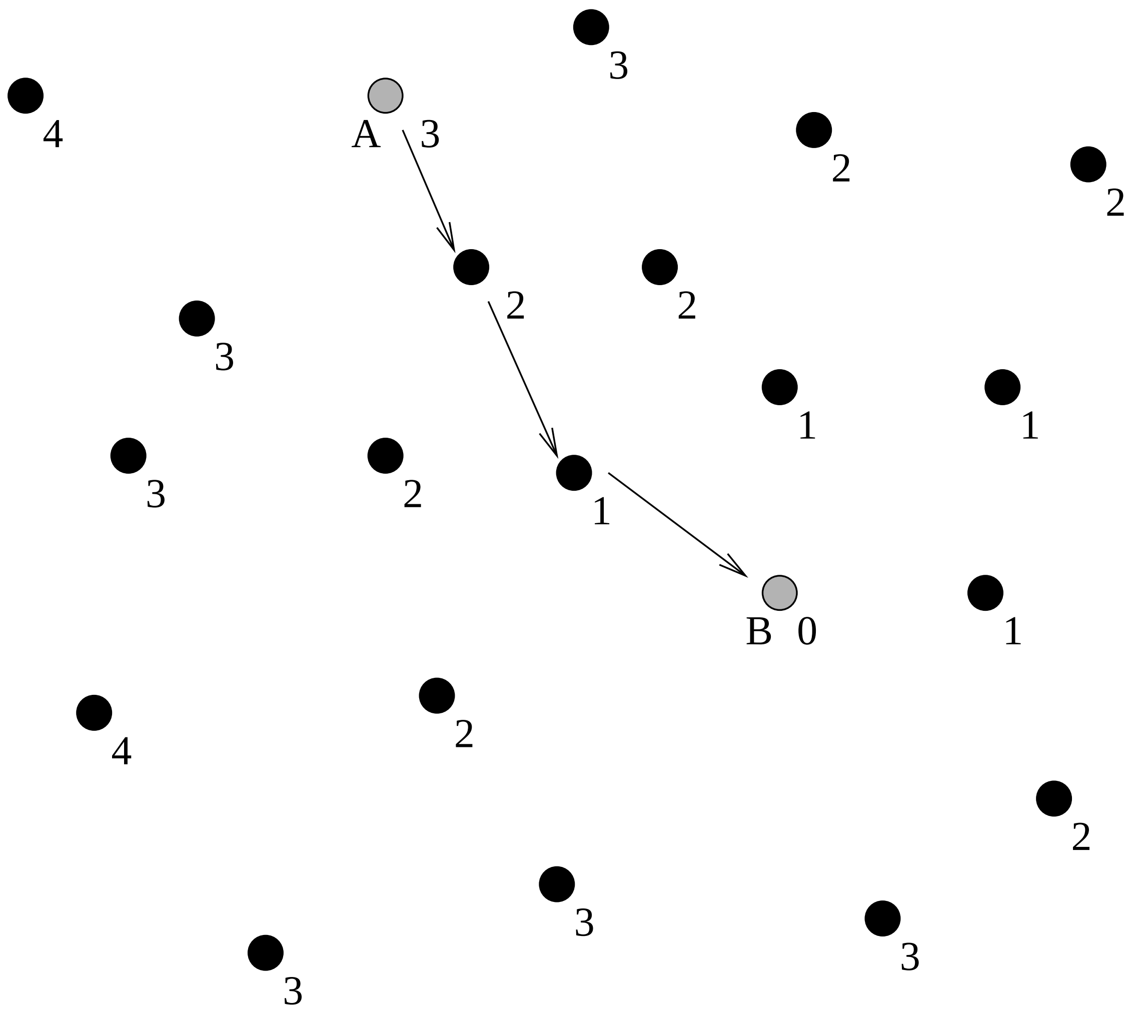


Figure 3.1: The nodes along an optimal path will have successively smaller hop counts

To send a packet

- Get the number of hops between the node and the destination
- Set the hop count of the packet to that value
- Set the destination address of the packet
- Broadcast the packet

On receiving a packet

- Decrement the hop count of the packet by one
- Compare hop count of the packet to the number of hops between the node and the destination
- If it is greater or equal, rebroadcast the packet
- Otherwise, discard the packet

Figure 3.2: Simple selective flooding algorithm

a packet to node B. Since node A is four hops away from node B, it sets the hop count of the packet to four and broadcasts it. The large circle around node A denotes the broadcast radius of that node. Although there are two nodes within the broadcast radius of node A, one of them has a hop count greater than that of the packet, so it does not forward the packet, thus preventing the packet from traveling in the wrong direction. The other node within broadcast radius of node A is three hops away from the destination, which is less than the hop count of the packet, so the node will receive the packet, decrement the hop count of the packet to three, and then rebroadcast it. This process continues until the packet reaches its destination. The packet takes two paths to the destination because there are two nodes with a hop count of one that are within broadcast range of the previous node in the path. Both of these nodes will receive a copy of the packet when the packet has a hop count of two. These two nodes will then both decrement the hop count of the packet by one and rebroadcast it, resulting in the destination node receiving the packet twice.

Another interesting property of this approach to routing results when the initial sender of a packet sets the packet's hop count to a value higher than its own hop count. In figure 3.4, the nodes in the network have moved, but the distance information at each node has not been updated, so the topology information at the nodes is stale. Node A then sends a packet to node B with a hop count of five instead of a hop count of four. The packet follows two paths to its destination. When the packet takes the upper path, it has a hop count of four when it passes through the node with distance three, a hop count of three when it passes through

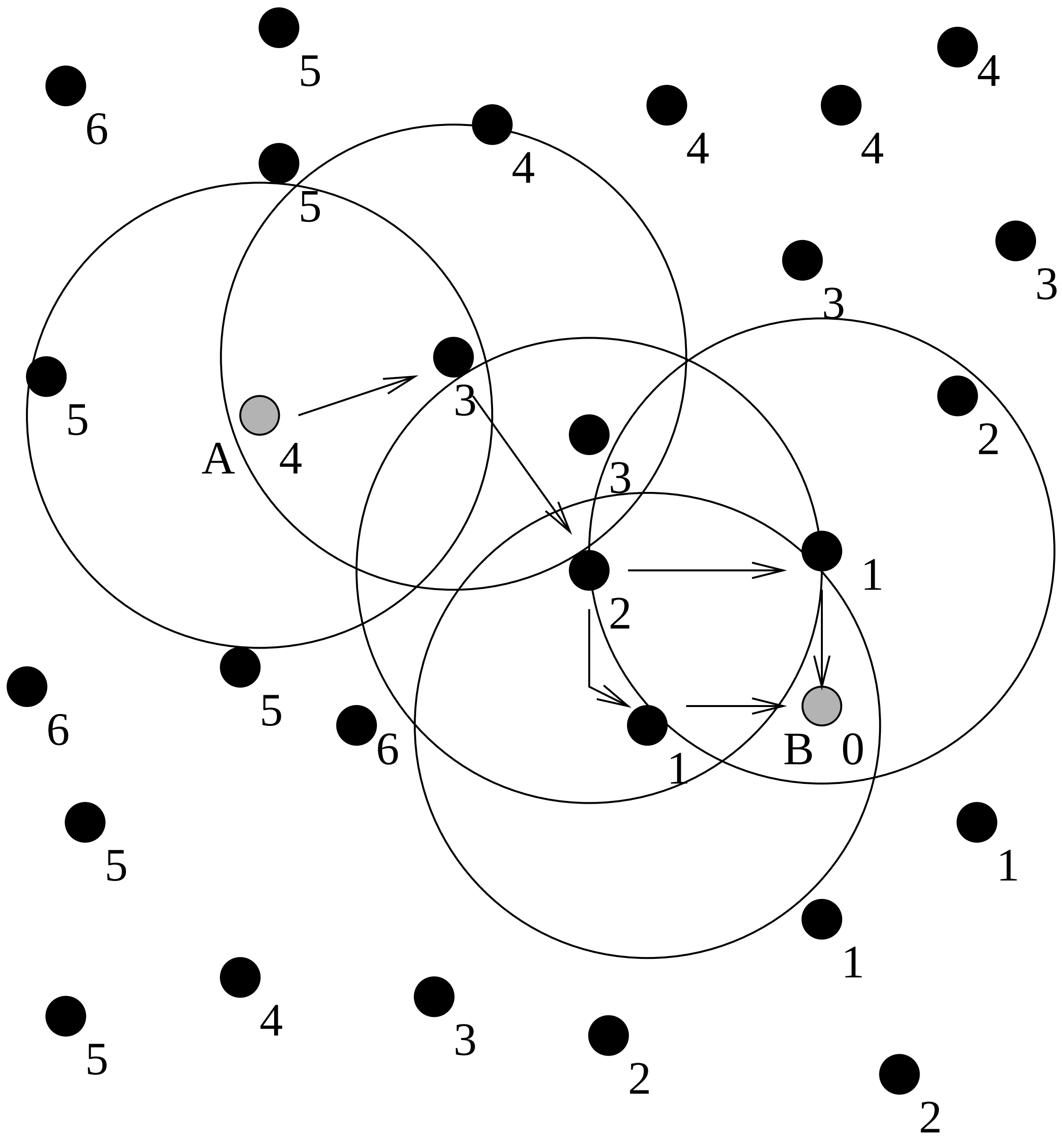


Figure 3.3: Node A using selective flooding to route a packet to node B

the node with distance one, and a hop count of two when it passes through the node with distance two. When the packet takes the lower path, it passes through two nodes that both have a distance of three to the destination. The packet has a hop count of four when it passes through the first node of distance three, and it has a hop count of three when it passes through the other node. So despite the incorrect distance information at the nodes, the packet is still able to reach its destination. If the initial hop count of the packet is set to a value greater than five, the packet will find more alternate paths to the destination. The packet essentially floods a channel in the network between the sender and the destination. As the initial hop count of the packet is increased, the size of the channel flooded by the packet becomes larger. As the size of the channel becomes larger, the algorithm behaves increasingly like a network flood and becomes increasingly resilient to incorrect topology information at the nodes. In fact, if the initial hop count of the packet is set to infinity, the algorithm will ignore the topology information completely and behave exactly like a network flood. The distance information stored at the nodes essentially acts as topology information used to control the scope of a flood, and the choice of initial hop count for packets determines how selective the flood is. This approach to routing is obviously a selective flooding technique.

Another way to view this approach is to consider the distance metric stored at a node to be a “height” for the node much as in link reversal routing algorithms (Corson and Park 2001). The network can then be viewed as a funnel with the destination at the bottom of the funnel (figure 3.5). The routing of packets behaves like water flowing from the sender down the funnel to the destination. Increasing

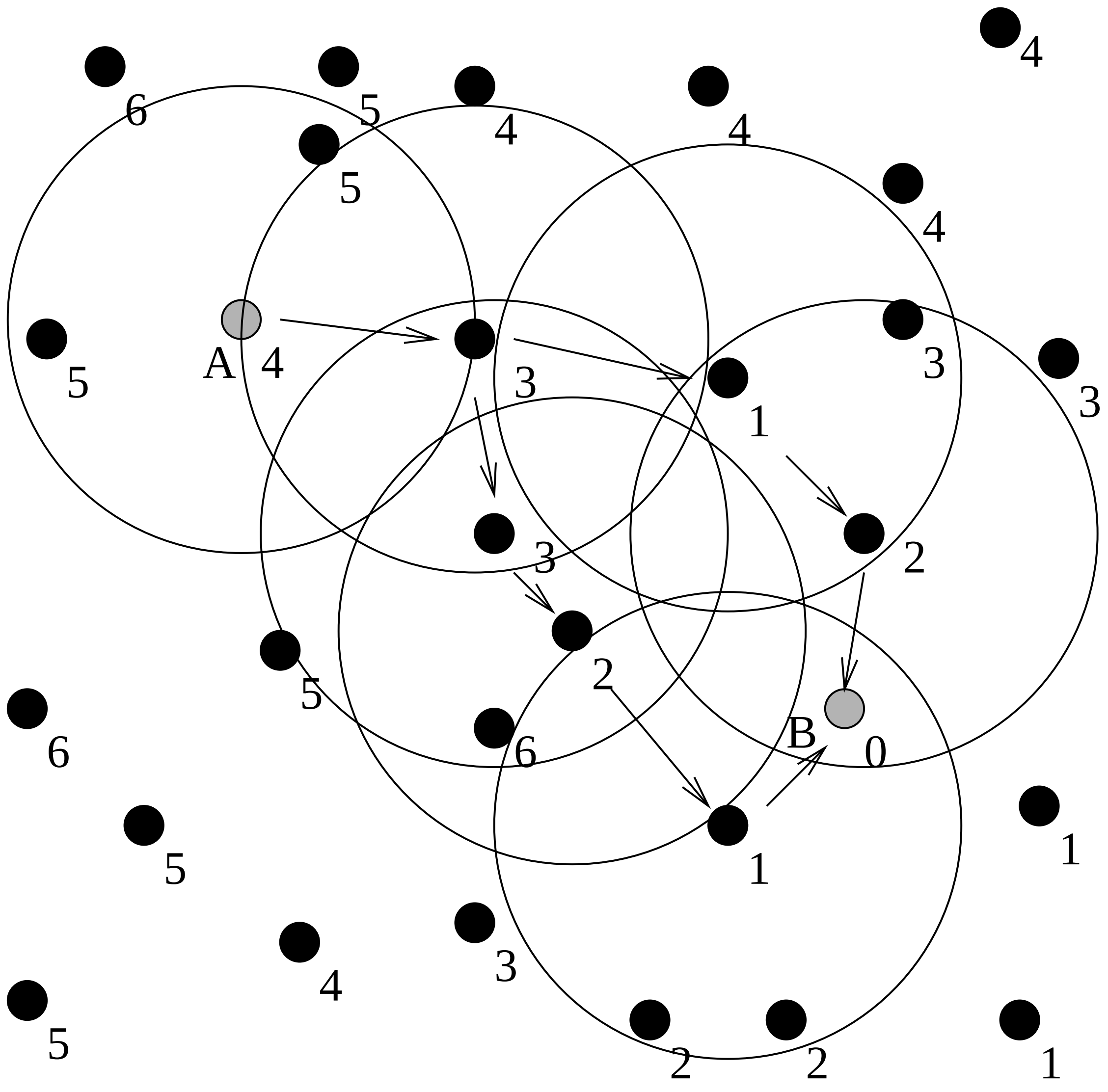


Figure 3.4: Increasing the size of a selective flood improves resilience to stale topology information

the initial hop count of packets is equivalent to starting with more water at the sender. With more water, a greater area of the funnel is flooded as it flows to its destination.

Because selective flooding uses broadcast communication instead of traditional point-to-point communication, the role of nodes in the network is different than in traditional networks. Instead of nodes receiving a packet and then having to decide to whom the packet should be forwarded, nodes receive all packets broadcast in the vicinity and then have to decide whether the packet should be rebroadcast to a different part of the network or not. Nodes simply make a binary decision: should they participate in rebroadcasting the packet or not? This decision is much simpler than having to choose a particular neighbour to serve as a next hop and is consistent with the fact that the algorithm requires less topology information to route packets. Unfortunately, the use of broadcast packets is less bandwidth efficient than point-to-point communication and makes it more difficult to detect when neighbours fail to receive a packet.

Nonetheless, this selective flooding approach seems feasible. The discussion so far, however, has ignored certain details that need to be addressed. For the algorithm to be complete, these questions must be answered:

- How does the distance information for the nodes initially get propagated throughout the network?
- Once this distance information is propagated, how is the distance information maintained and kept up-to-date as the nodes in the network move?
- When a packet is sent, how does the sender decide what to set the initial hop

Figure 3.5: Selective flooding resembles water flowing down a funnel

count to?

In addition, because selective flooding relies heavily on broadcast communication, it is also important to have a MAC layer that handles broadcast communication efficiently. Overall though, these questions deal with how to manage the topology information stored in a network. In traditional ad hoc network routing, there are two approaches to managing this information: the table-driven and on-demand approaches. Algorithms based on both approaches have been created.

3.1 A Table-Driven Approach

In the table-driven selective flooding algorithm, nodes store distance information about all the nodes in the network. Upon being activated, a node will flood the network with a "hello" packet. Nodes receiving one of these hello packets will know their distance to the sender since all packets track the number of hops they have visited while traveling. Nodes store distance information in a routing table-like structure and exchange tables with neighbours every 15 seconds. A node merges neighbours' tables with its own by exponentially averaging the distance values of each table entry. Additionally, headers are added to data packets listing the sender of the packet, the distance that the packet has traveled, the forwarder of the packet, and the forwarder's distance to the destination. Nodes that overhear a data packet will use that information to update their distance table entries for the packet's sender, forwarder, and destination, also using exponential averaging.

The initial hop count of packets is always two more than the sender's distance to the destination. Although the hop count is larger than is necessary to route packets properly and to correct small link breakages, it is not so large as to flood the entire network.

The use of exponential averaging for updating distance tables is intended to cause the distance information at each node to converge to correct values as nodes move. Unfortunately, simulation reveals that this is not the case. Several situations exist which cause the distance information to converge to incorrect values. In particular, incorrect distance values often develop in nodes at the edge of the network, and these incorrect values eventually propagate back to the interior of the network, causing the distance information in the entire network to be incorrect. This effect is fairly significant as evidenced by the fact that initial simulations show a significant improvement in the percentage of successfully routed packets after the updating of distance information at nodes is disabled.

Although these problems can be surmounted using sequence numbers or other schemes, table-driven algorithms are generally frowned upon by the ad hoc networking community, so further refinement of the table-driven algorithm has been abandoned.

3.2 An On-Demand Approach

Compared to the table-driven algorithm, the on-demand algorithm uses a simpler approach to maintaining distance information at the nodes. It assumes that the only reliable way for a node to determine the number of hops between it and another node is to receive a packet from the other node and to count the number of hops that the packet has traveled. If a node receives a packet from another node that has traveled three hops, then there must exist a path to the other node of length three. By periodically receiving packets from other nodes, a node can then maintain its distance tables.

In the algorithm, when a sender wants to send a packet to a destination node but does not know its distance to that node, it initiates a search for the destination. The search consists of an expanding ring search using broadcast packets with time-to-live values (figure 3.6). Nodes within the search region will receive a packet from the sender, so they will know their distance to the sender. If the destination is within the search region, it will send a reply to the sender (figure 3.7). This communication is possible because all the nodes between the sender and destination will have been within the search region and will all know their distance to the sender. As such, they are able to properly route a packet from the destination to the sender. The act of forwarding this reply packet from the destination to the sender will update the distance tables of nodes along the route with distance information about the destination. Afterwards, the sender, destination, and intermediate nodes will have the correct hop counts of both the destination and sender, so packets can flow between the two. Sending packets between them will update the distance information along the route, so they can maintain the route by simply sending packets to each other periodically (figure 3.8). The longer the amount of time that passes since a sender receives a packet from a destination, the more stale the distance information at the nodes become, and the higher the sender must set the initial hop counts of packets to compensate.

More information about implementation issues and optimizations for this algo-

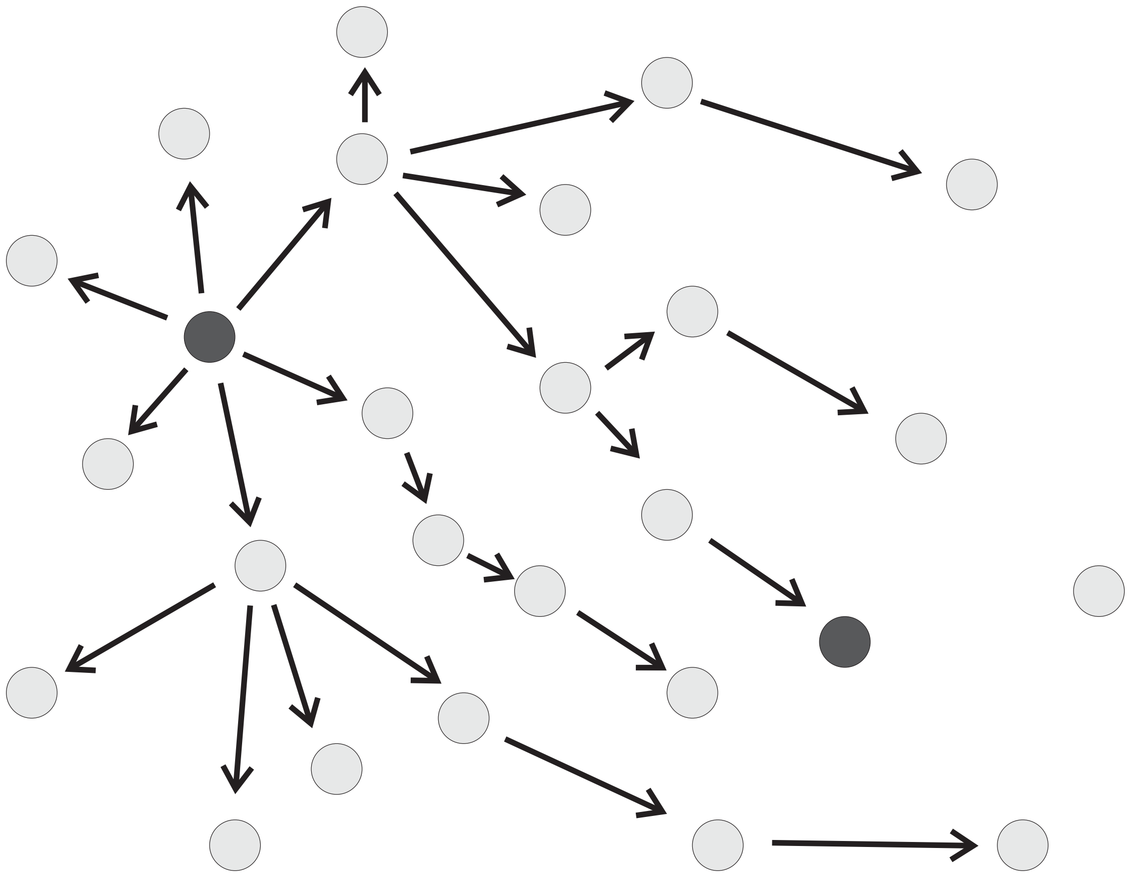


Figure 3.6: A limited depth search for a destination will update distance information in the search region

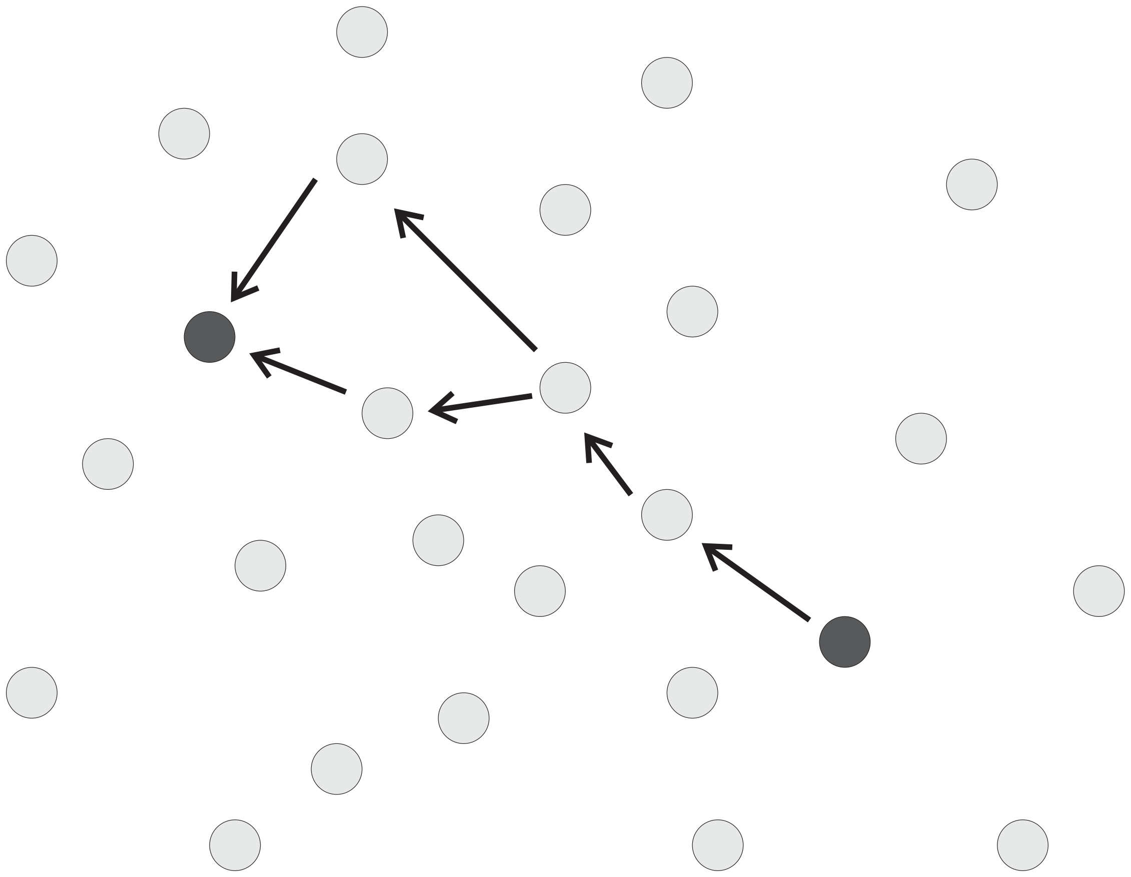


Figure 3.7: A reply by the destination will update distance information for the opposite direction

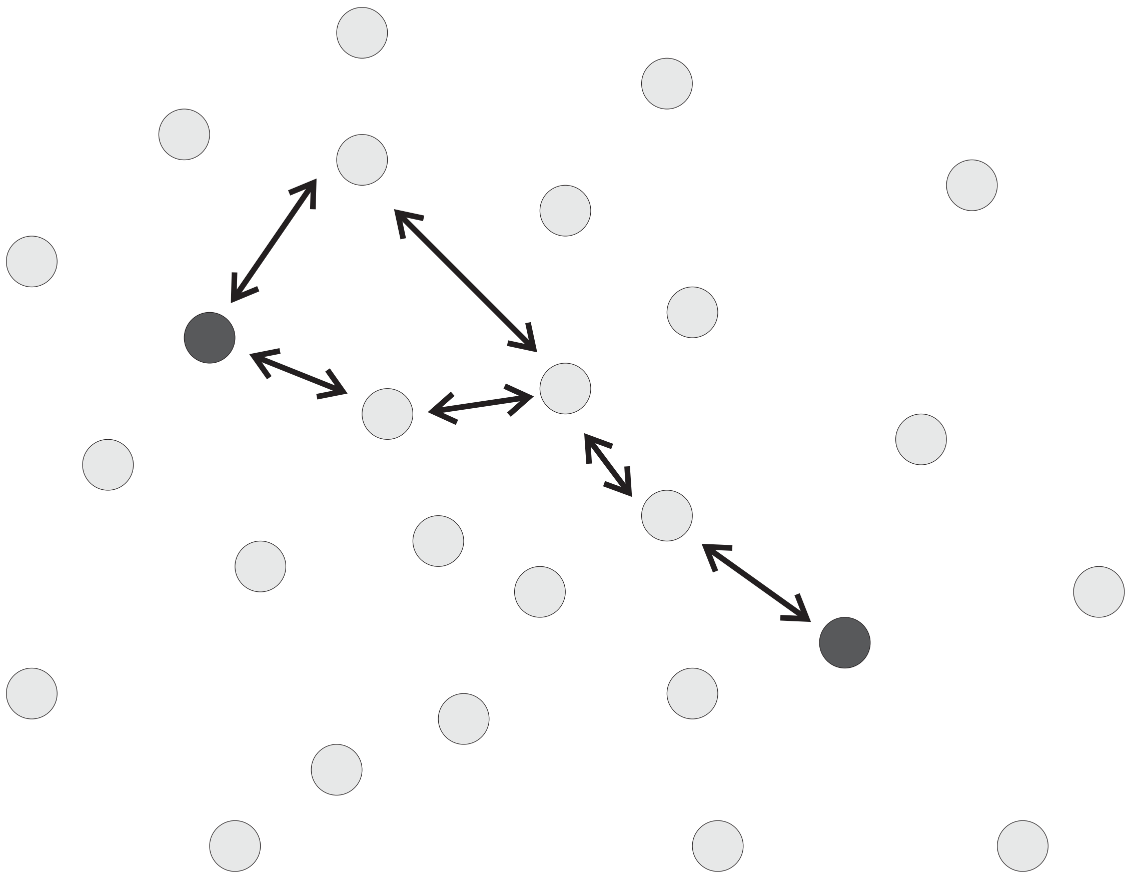


Figure 3.8: Periodic exchanges of packets will update distance information

rithm appear in the next chapter.

Chapter 4

Experiments and Results

Because creating large scale wireless networks is expensive and time-consuming, wireless network routing algorithms are evaluated by simulation. The implementation of on-demand selective flooding used in simulations contains several optimizations not discussed in the general algorithm description. Simulated on a variety of different routing scenarios, it achieves good performance on some but exhibits very poor performance on others. Unfortunately, the poor performance is caused by excessive bandwidth usage and has no obvious solution.

The simulator used to test the performance of on-demand selective flooding is the Network Simulator (ns-2) version 2.1b9 (*The Network Simulator: ns-2* 2002). Ns-2 is a discrete event simulator that can simulate the behaviour of all the protocol stack layers of an ad hoc network node (*The CMU Monarch Project's Wireless and Mobility Extensions to ns* 1999). Because the 802.11 MAC layer (ANSI/IEEE Std 802.11-1999) used in standard ad hoc networking simulations handles broadcast packets poorly, the selective flooding algorithm uses a different MAC layer, the details of which are discussed in appendix B. Also, various protocol stack layers such as the ARP and out queue are absent or merged with other layers because of certain cross-layer optimizations used by the selective flooding implementation. Although these changes to the protocol stack and MAC layer can be used with other routing algorithms, they are of little use to traditional routing algorithms, so the other routing algorithms still use a normal ad hoc networking protocol stack and a 802.11 MAC layer in the simulations.

4.1 Implementation Details

The selective flooding algorithm implemented in the simulation environment contains optimizations to improve its performance and exhibits certain behaviours specific to its implementation. Although pseudocode for the algorithm is included in appendix A, a more general discussion of implementation details is included here:

4.1.1 Repeat Packets

When a network is flooded with a packet, every node in the network will eventually receive this packet and rebroadcast it. Since a node may have multiple neighbours, each of whom will be rebroadcasting that packet, a node may receive the same packet multiple times. To prevent nodes from redundantly rebroadcasting the same packet, each packet is uniquely identified with a source address and sequence number allowing each node to track which packets it has received or broadcasted before. Nodes can then identify packets that are repeats of packets that they have already received and will not bother reprocessing them.

4.1.2 Neighbour Heuristic

In a wireless network, network nodes sometimes congregate in a cluster. For example, multiple people carrying wireless devices might decide to ride in a car together or gather in a meeting room. Because these nodes are so close together, their broadcast zones will largely overlap. As a result, if, during a flood, all of these nodes rebroadcast the same packet, much of their effort will be redundant. To prevent this wastage, a heuristic is implemented whereby if a node hears a nearby node broadcasting a packet, the node will not rebroadcast the same packet. In fact, if the node has that same packet queued in its output buffer for eventual transmission by the MAC layer, the node will remove the packet from that buffer. Nodes can determine the proximity of their neighbours based on the strength of received radio transmissions. In the simulator, the standard broadcast radius for a node is 250m. When nodes are within 100m of each other, they are considered a cluster. Although there are situations where this heuristic will cause packets to be lost, this loss of packets will be rare in networks with medium or high node density.

4.1.3 Request for Echoes

To maintain a route between two nodes, the on-demand selective flooding algorithm requires that the two nodes periodically send packets to each other. Because the typical simulation load for ad hoc networks consists of one-way communication only, the implementation of the algorithm uses a simple scheme to decide when nodes should send packets to each other for route maintenance. In this scheme, the source node of a communication connection maintains the route from the destination to itself by sending actual data packets to the destination. The destination will only send a packet to the source node, thereby maintaining the route from source to destination, if the source node specifically requests that a packet be sent. As a result, it is ultimately the responsibility of the source node to maintain the entire route.

4.1.4 Types of Packets

In order to exchange routing control messages, this algorithm implementation uses four different types of messages: search messages, echo request messages, echo reply messages, and empty messages. Since these messages are mutually exclusive to each other and all packets carry one of these four messages, there are, in essence, four types of packets: search packets, echo request packets, echo reply packets, and normal packets. The search packet is used during limited depth floods to find a specific node. An echo request packet is routed using normal selective flooding routing procedures and is sent when one node wants another node to send back a packet to refresh the route between them. An echo reply packet is routed normally and is sent in reply to a search or echo request packet. Finally, a normal packet is a packet routed normally and has no control instructions. All packets carry application data except for echo reply packets.

4.1.5 Search Behaviour

Due to the architecture of the algorithm implementation, when a node is executing an expanding ring search for a target, the searching node can receive and forward packets but cannot send any packets originating from itself. Instead, these packets will be queued until the search has ended. One consequence of this behaviour is that a node can only execute one search at any one time. The expanding ring search starts with a depth of two, and the depth is increased in increments of three until a maximum depth of 25 is reached. At that point, the search is terminated and the target node will be marked as unreachable. Packets sent to unreachable nodes are discarded.

4.1.6 Search Optimization

If a node receives a search packet and knows its distance to the target of the search, the node will accelerate the search by sending an echo request packet, on behalf of the searcher, to the target. As the echo request travels to the target, nodes along the route will update their distance information about the searcher. Then when the target replies to the echo request, the reply will travel back along this route, through the search region, to the searcher, establishing a route between the target and searcher. Although this route may not initially be optimal, as more packets flow between the two nodes, the route will gradually improve until it converges to an optimal route.

4.1.7 Invalidation of Distance Information

If the distance table at a node has not been updated in a significant amount of time, its entries will likely have become stale or invalid. This can lead to incorrect routing. Table entries for a destination that have not been updated for over four seconds are considered stale. A node sending data to a destination for which it has stale distance information will send the data in the form of an echo request packet so as to refresh its distance information. If a node's table entry for a destination has not been updated for over twenty seconds, that entry is considered invalid and will be removed from the node's distance table. The node will then behave as if it has no distance information for the destination at all.

4.1.8 Distance Aging

The amount of time that has passed since a node's distance information was last updated is also used by senders to determine the initial hop count of packets. A packet's initial hop count is set to be one more than the distance between the sender and the destination, plus one additional hop for every five seconds since the sender's table entry for the destination was last updated; therefore, as the sender's distance information becomes more stale, the hop count of packets is gradually increased to compensate.

4.2 Simulation Scenarios

The on-demand selective flooding algorithm has been tested against a variety of routing scenarios. These scenarios are consistent with those used in other ad hoc networking research (Perkins and Royer 2001). The scenarios consist of 50 or 100 network nodes in a 1000m by 1000m world simulated for 400 seconds. All of the nodes are constantly in motion, based on a random waypoint movement model (Johnson and Maltz 1996). In this model, nodes randomly select a point to move to, and then move towards that point at a constant speed. Once that point is reached, they will pause at that location for up to one second, then select another point to move to, and so on. The speed at which a node moves is set to a new random value each time a node reaches a waypoint, but the speed must be below a certain maximum speed. The different maximum speeds simulated are 0m/s, 2m/s, 10m/s, and 30m/s.

All nodes are silent at the start of scenarios. As scenarios unfold, ten communication connections are established at random times. These connections stay active until the end of the simulation. Each of them consists of one randomly chosen node unicasting constant bit rate (CBR) data in 512 byte chunks using the user datagram protocol (UDP) (Comer and Droms 1999) to one other randomly chosen node. This communication is one-way. Two different traffic loads are simulated: a sparse load where nodes only send 0.25 packets per second over each connection and a more typical load where nodes send 4 packets per second over each connection.

Although various statistics can be gathered during simulations, the primary statistic of significance is the packet delivery ratio. This ratio is the percentage of data packets sent from a node's application layer that actually arrive at their destinations. The results reported in this thesis are from single simulation runs. The nodes in all simulation runs involving the same number of nodes and having the same maximum speed follow identical movement patterns. Similarly, simulation runs with the same traffic rates use identical communication connection patterns.

4.3 Results

Simulation results show that on-demand selective flooding is able to route data in spite of stale topology information but suffers from serious scalability problems.

Figure 4.1 compares the packet delivery ratios of selective flooding and several popular on-demand ad hoc networking algorithms in simulations involving 50 nodes, a sparse traffic load, and various movement speeds. The use of a sparse traffic load is significant. Because packets are only sent once every four seconds and all of the algorithms in the simulation are on-demand algorithms, the topology information for routes is only updated once every four seconds. As the movement rate increases, the number of topology changes that occur during these four seconds also increases. Algorithms that aren't resistent to stale topology information must then resort to more drastic topology updates to recover correct routes. Although the performance of selective flooding is weaker than the other routing algorithms when there is no movement in the network, its performance remains relatively constant as node movement increases because of its tolerance for stale topology information. On the other hand, the performance of other on-demand algorithms deteriorates because they have difficulty recovering from having stale topology information. As a result, selective flooding outperforms the other algorithms by a wide margin at the highest movement rate. Selective flooding can likely outperform the other algorithms at 10m/s as well, but the algorithm complained of many search failures, suggesting that the network might have become partitioned or that the neighbour heuristic might have interfered with the algorithm.

The neighbour heuristic is also thought to be the reason why selective flooding

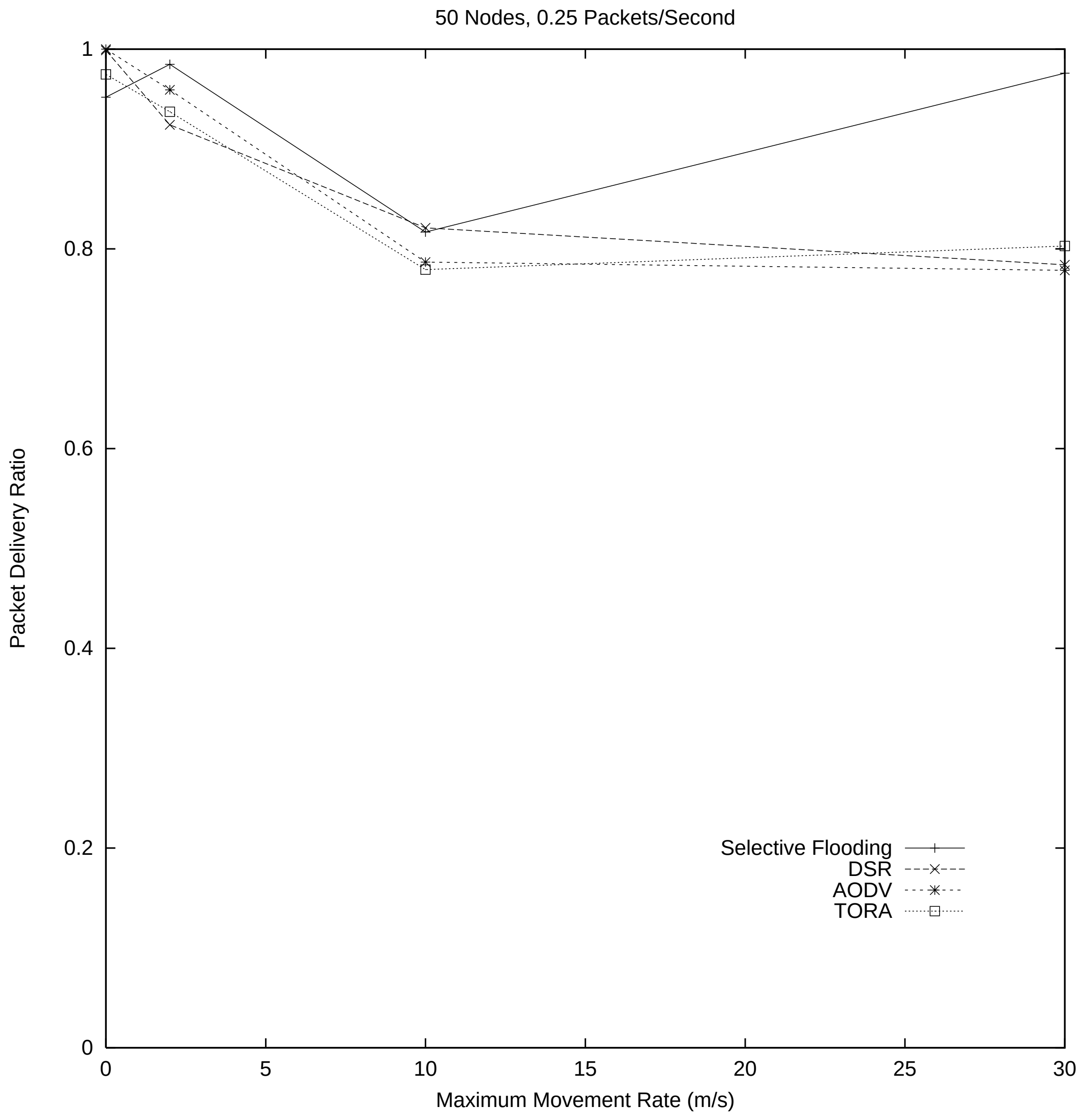


Figure 4.1: Selective flooding achieves superior performance at high movement rates

is not able to achieve 100% packet delivery ratios even when there is no node movement. Networks with low node density may cause problems for this heuristic. Figure 4.2 shows simulation results for 100 nodes and a sparse traffic load. With the higher node density, the selective flooding results are actually similar but more consistent. Disabling the neighbour heuristic does allow selective flooding to achieve 100% packet delivery ratios.

Although selective flooding performs well when packets are only sent once every four seconds, standard ad hoc networking simulations involve traffic loads of four packets every second. Figure 4.3 shows the simulation results of 50 nodes under a normal traffic load. The packet delivery ratio for selective flooding is much worse than that of the other routing algorithms. As expected, the performance of some of the algorithms improves compared to their performance under a sparse traffic load. This phenomenon occurs because on-demand algorithms update the topology information along their routes only when data packets are actually being sent. Since the packet rate in these scenarios is 16 times higher than in the previous scenarios, the topology information is updated 16 times more often, so the routing information is less likely to be stale.

In contrast, the performance of selective flooding is much worse with a normal traffic load than its performance with a sparse traffic load. One factor that contributes to this behaviour is the MAC layer. Although the MAC layer for the selective flooding algorithm is designed for handling broadcast communication (as described in appendix B), it still suffers from collision problems. Figure 4.3 also shows the effect of disabling packet collisions on selective flooding. Although the

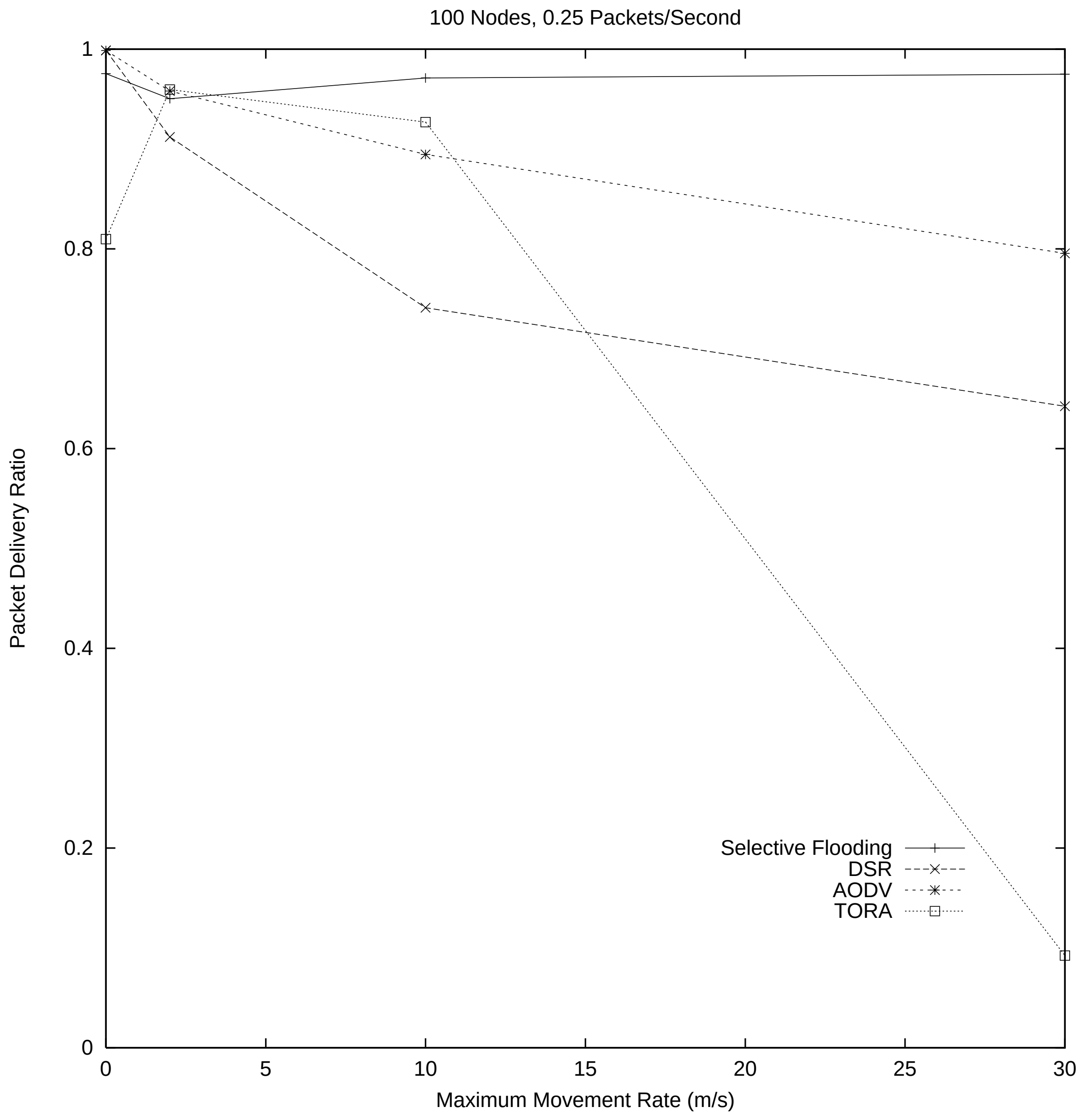


Figure 4.2: Selective flooding performance stabilizes with increased node density

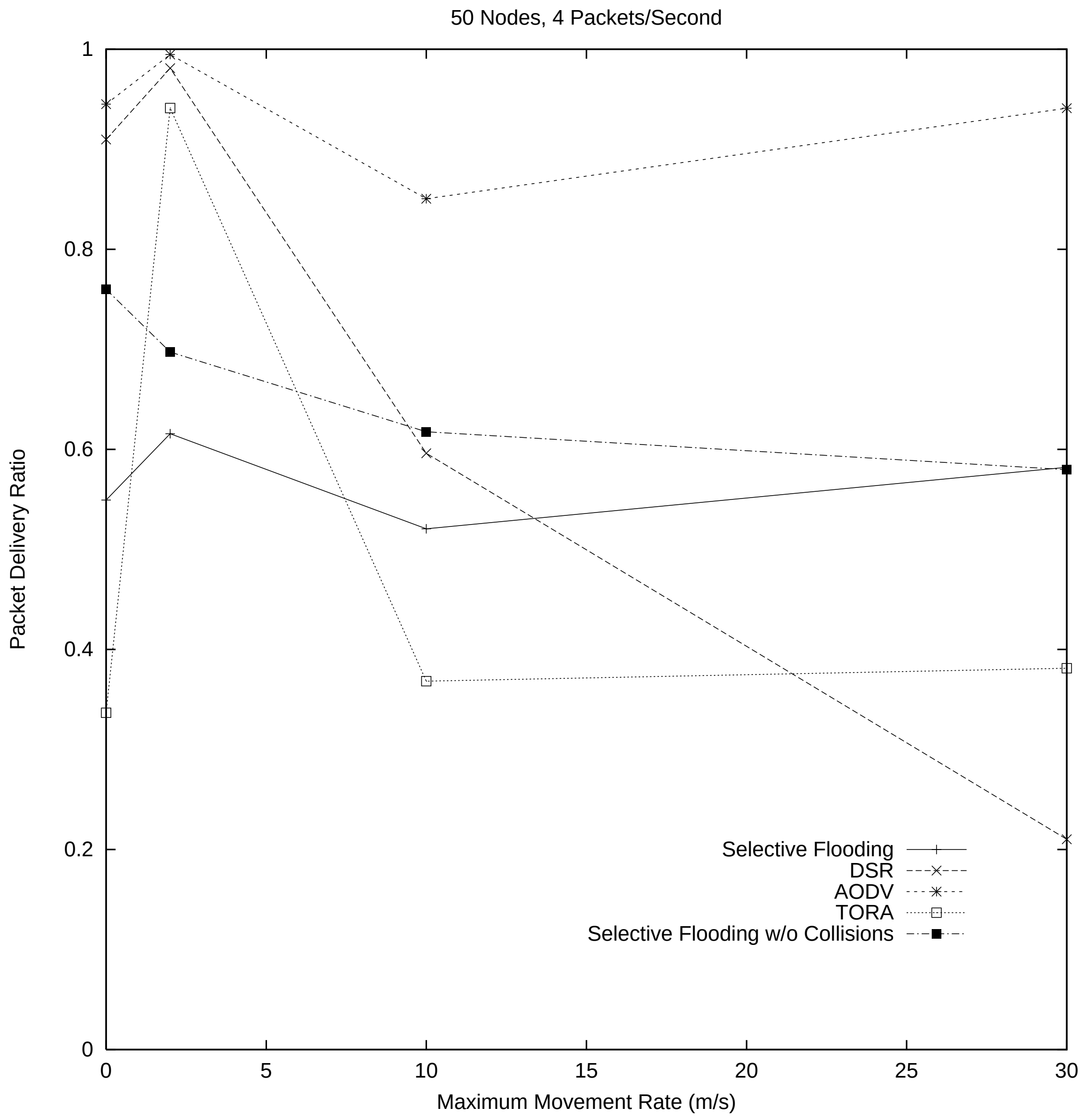


Figure 4.3: Selective flooding cannot handle normal traffic loads

MAC layer still uses the same collision avoidance procedure it normally uses, it ignores all collisions and process packet data even if the data should not have been received correctly due to interference. The performance of selective flooding improves when collisions are disabled but is still poor.

The main factor behind this poor performance is probably insufficient bandwidth. Because selective flooding routes packets along multiple paths to their destinations, it uses more bandwidth than other routing algorithms. During simulations, many of the nodes discard packets because their send queues become full. Attempts to lower the bandwidth usage by giving priority to packets that are traveling along the optimal path to their destinations do not provide any improvement in performance. Although it is difficult to determine the scale of this problem, analytic results show that selective flooding requires more bandwidth than is available in the network.

The results for DSR are somewhat unusual. Published DSR results typically show DSR outperforming AODV in similar simulations (Das, Perkins, and Royer 2000). Running the scenarios on two older versions of the ns-2 simulator, versions 2.1b7a and 2.1b8a, produces data consistent with the published results. The latest ns-2 simulator contains changes to DSR and the 802.11 MAC layer that are likely causing this discrepancy, but it is unclear whether these newer results should be considered more accurate than the published results.

4.4 Analytic Observations about the Behaviour of Selective Flooding

During high data rate simulations, nodes using selective flooding report overflowing send queues whereas no such problem occurs with other algorithms. This behaviour suggests that selective flooding uses more bandwidth than other algorithms, but how much of a problem is this extra bandwidth usage? Is it possible to estimate the extent of this problem?

Most of the congestion problems likely occur near the center of the network. This claim is reasonable because on a 1000m by 1000m map where nodes broadcast with a range of 250m, if a sender and destination are randomly chosen, the communication between these two nodes will likely pass through the center of the network (figure 4.4). By assuming that all traffic in the network must pass through the center of the network, one can easily calculate the amount of traffic there.

At the peak of a scenario involving normal traffic loads, there are ten nodes simultaneously sending packets into the network. Each will transmit four packets per second with 512 bytes of data in each packet. The MAC layer used in the simulation reserves the wireless channel for 1000 bytes worth of data for each packet regardless of a packet's actual size. Thus, the bandwidth needed in the center of the network to forward all the data is 10 connections$\times$1000 bytes$\times$4 packets per second or 40 KBps. Since the channel supports a bandwidth of 2 Mbps or 250 KBps, a node in the center of the network should be able to handle all of the data passing through the network easily.

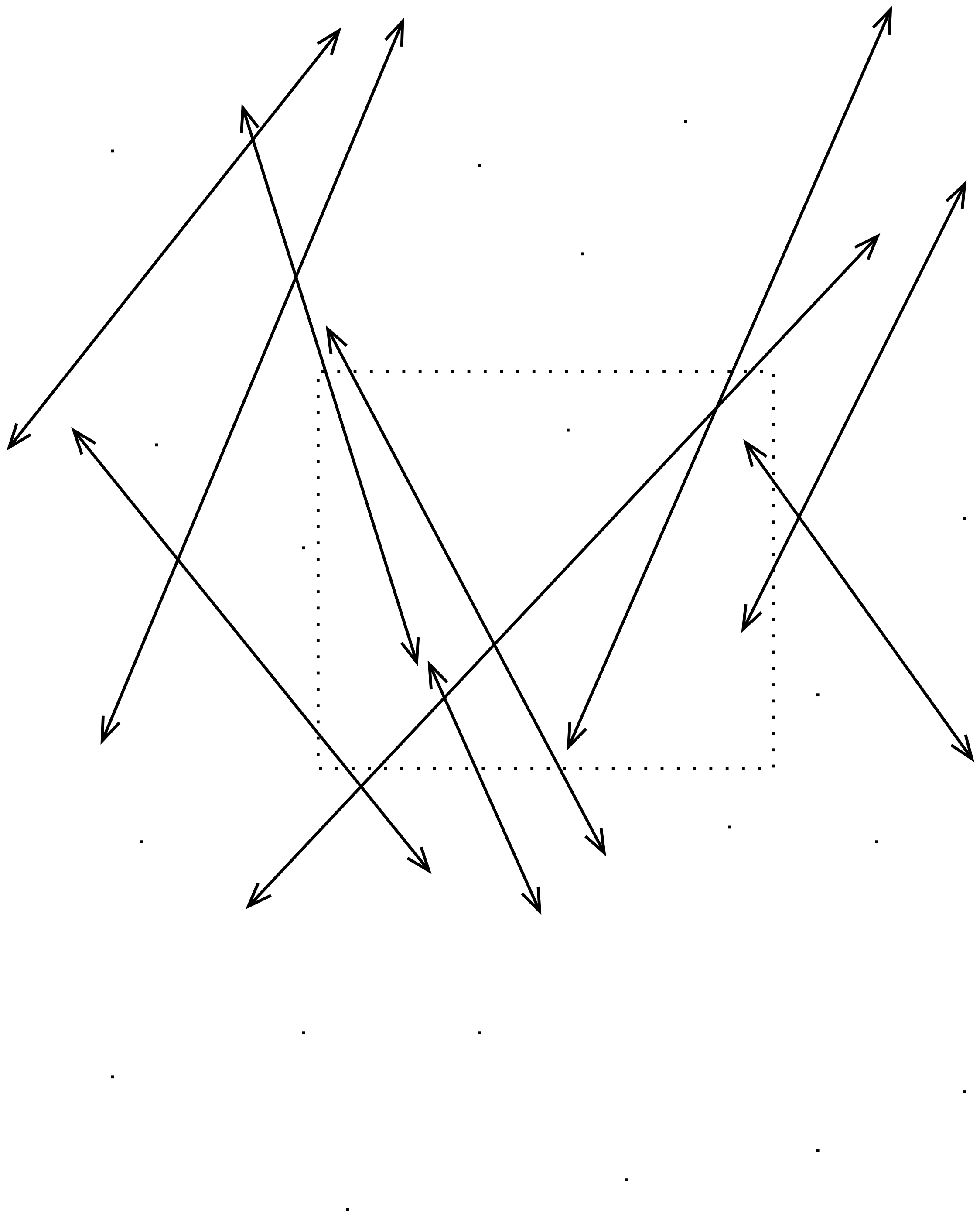
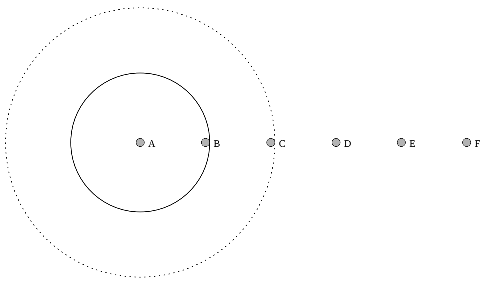


Figure 4.4: Most traffic passes through the center of the network

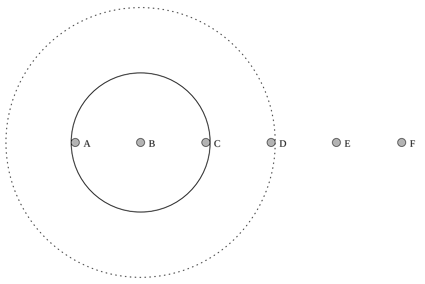
Nodes in the center of the network cannot transmit constantly though. They must remain silent while their neighbours are transmitting or receiving. Because selective flooding relies on broadcast communication, when one node is transmitting, any node within a radius of twice the node's broadcast radius must stay silent or risk interfering with the transmission. Figure 4.5 shows the forwarding of a packet along a chain of nodes. In the figure, Node C must stay silent four times during the forwarding of the packet: when node A is broadcasting to node B, when node B is broadcasting to node C, when node D is broadcasting to node E, and when node E is broadcasting to node F. In addition, node C must also consume bandwidth itself when it transmits the packet to node D. So even in an optimal MAC layer for broadcast communication, the bandwidth consumed by forwarded packets is at least five times the size of the packet itself.

So assuming an optimal MAC layer, the bandwidth available in the center of the network must be at least five times the amount of data being sent through the center of the network. This value is 5×10 connections $\times 1000$ bytes $\times 4$ packets per second or 200 KBps. This is 80% of the total 250 KBps bandwidth available in the channel. Once the overhead of contention and handshaking protocols is factored in, it is clear that there is simply not enough bandwidth available in the center of the network to handle all the communication connections in the network.

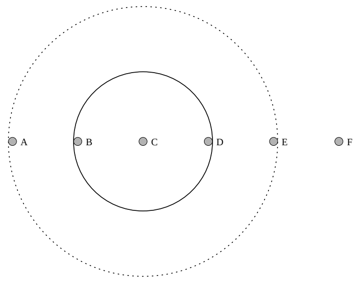
Additionally, the selective flooding algorithm used in the scenarios sets the initial distance of packets to be one higher than what is needed to reach the destination. As a result, packets will not only take the optimal path to the destination, but any adjacent path as well. If one assumes that there is at least one adjacent path



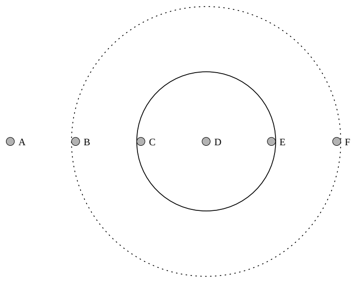
(a) A broadcasts to B



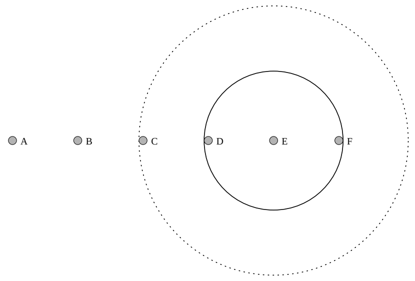
(b) B broadcasts to C



(c) C broadcasts to D



(d) D broadcasts to E



(e) E broadcasts to F

Figure 4.5: A forwarded packet will consume node C's bandwidth at least five times

on either side of an optimal path, selective flooding will require at least three times more bandwidth in the center of the network. Clearly, there is not enough bandwidth available for selective flooding to handle the routing of a normal traffic load.

4.5 Results Summary

Overall, simulations show that selective flooding is indeed able to route packets successfully in an ad hoc network. Its routing is resilient to stale topology information. Even when there is considerable node movement and the topology information stored at nodes is rarely updated, selective flooding is still able to route packets successfully while traditional ad hoc network routing schemes often lose a significant number of packets. Unfortunately, selective flooding uses significantly more bandwidth than traditional routing under more normal circumstances.

Chapter 5

Conclusion

Although much research has been undertaken in the field of ad hoc network routing, current algorithms are generally all vulnerable to stale topology information in the network. The algorithms deal with this problem by lowering the cost of updating the topology information, thereby making it possible to update the information more often.

Selective flooding can route packets in the presence of stale topology information by using flooding techniques. When compared to traditional routing techniques, selective flooding is superior at routing packets when networks are very dynamic and few topology information updates are allowed. Unfortunately, most simulated ad hoc networks are actually fairly stable with connections suffering from link breakages at most about once every four seconds.

Additionally, selective flooding requires more bandwidth than traditional routing. In fact, it may be more bandwidth efficient to constantly update the topology information in a network than to use selective flooding. Selective flooding uses so much bandwidth that it scales poorly, and at high transfer rates, it discards many packets due to insufficient bandwidth.

Consequently, selective flooding is largely ineffective as an alternative to traditional ad hoc network routing techniques. Analysis of selective flooding does reveal interesting properties though about the behaviour of traditional routing algorithms when dealing with bursty traffic, the bandwidth efficiency of ad hoc networks in general, and factors that must be considered in designing scalable ad hoc network routing algorithms.

5.1 CBR Traffic in Ad Hoc Networking Simulations

One interesting observation revealed during networking simulations is that although traditional ad hoc network routing techniques achieve extremely high packet delivery ratios when nodes send packets at a fast rate, they exhibit much worse performance when nodes send packets at a low rate. Although low data rate traffic is rare, many applications do produce bursty network traffic, which resembles low rate CBR traffic more than it resembles high rate CBR traffic. For example, interactive data applications such as e-mail, instant messaging, or web browsing will produce no network traffic for several seconds while users read data but then produce considerable amounts of traffic when users start accessing or sending additional data. Even voice communication, which is commonly associated with CBR traffic, often generates bursty data. If one considers only one direction of a voice communication connection, the connection alternates between having CBR traffic when a user is talking and having no traffic when a user is listening silently.

Despite the existence of a wide variety of interactive data applications, ad hoc network simulations use high data rate CBR traffic almost exclusively. Ns-2 exponential traffic loads characterize the bursty traffic of these applications far more accurately than high data rate CBR traffic. Figures 5.1 and 5.2 show the packet delivery ratios of various algorithms in simulations involving 50 nodes under a normal CBR traffic load and a ns-2 exponential traffic load. Under the exponential traffic load, senders alternate between five seconds of silence and two seconds of four packet per second traffic.

Although AODV reports a packet delivery ratio of over 90% in high movement scenarios under CBR traffic, its packet delivery ratio drops to below 40% under an exponential traffic load. The results for TORA are not reported since its ns-2 implementation cannot complete exponential traffic simulations.

Although some may argue that the poor performance of algorithms under exponential traffic is merely caused by the poor selection of distance aging parameters in these algorithms, bursty traffic is common enough that the poor tuning of these parameters indicate a major deficiency in those algorithms.

In the future, ad hoc network routing papers should report both CBR traffic and exponential traffic results to provide a more accurate portrayal of routing performance. The need for this reporting is especially important for those algorithms submitted to the Internet Engineering Task Force (IETF) for standardization (*Mobile Ad-hoc Networks Charter* 2002).

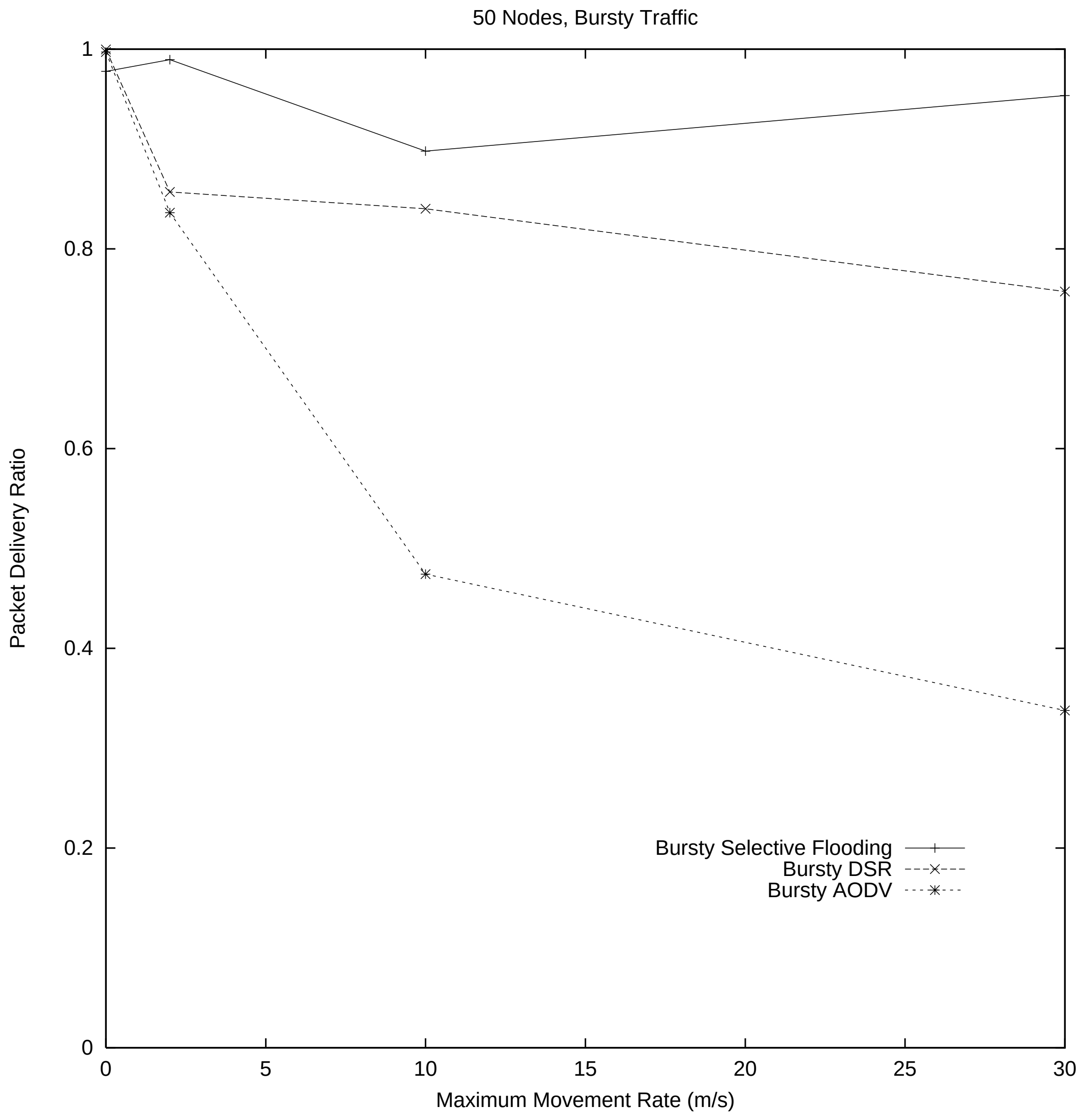


Figure 5.1: AODV exhibits poor performance with bursty traffic

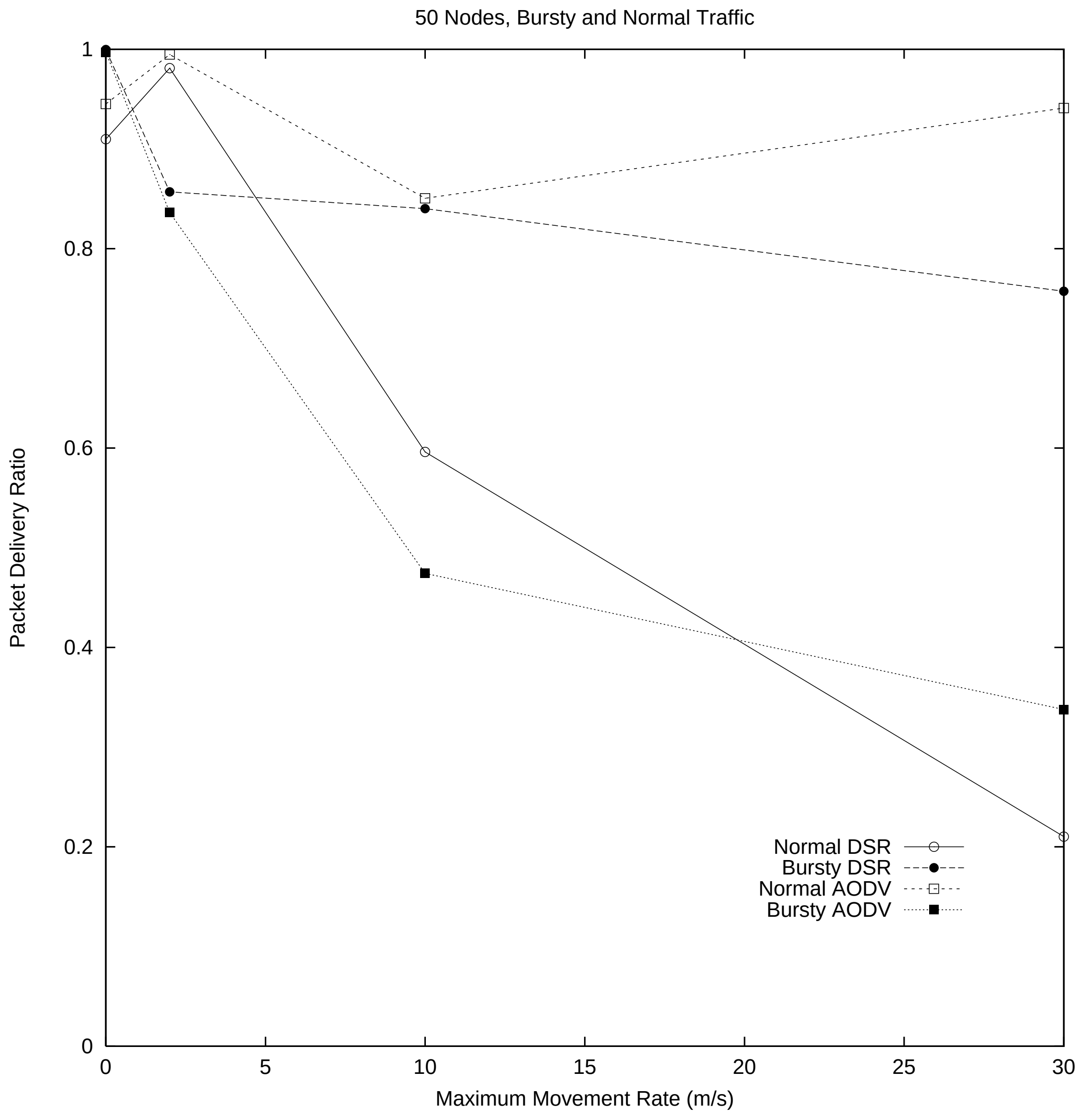


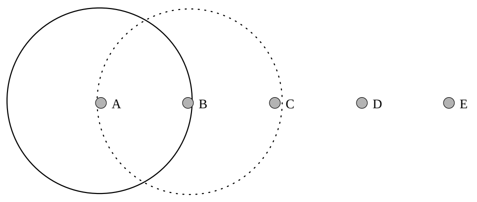
Figure 5.2: AODV's poor performance with bursty traffic is not reflected in its excellent performance with CBR traffic

5.2 The Efficiency of Forwarding Packets in Wireless Networks

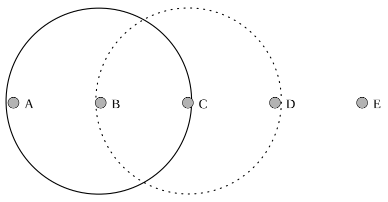
The second interesting ad hoc networking property revealed during the analysis of selective flooding relates to the bandwidth efficiency of ad hoc networks in general. Do the fundamental bandwidth inefficiencies found in selective flooding also affect traditional ad hoc networking algorithms?

Fortunately, since traditional ad hoc network routing algorithms primarily use point-to-point communication, they make more efficient use of wireless channels than the broadcast communication of selective flooding. In an optimal MAC layer for point-to-point communication, a packet forwarded through a network will prevent nodes along its route from being able to receive or transmit three times. Figure 5.3 shows the effect of the forwarding of a packet along a chain of nodes. Node C cannot transmit when node A is forwarding the packet to node B or it will interfere with the transmission; it cannot transmit when it is receiving the packet from node B; and it cannot receive when node D is transmitting to node E (node C can potentially transmit during this time, but doing so will interfere with node D's ability to receive control signals from node E). Additionally, node C must also use its bandwidth to actually forward the packet to node D.

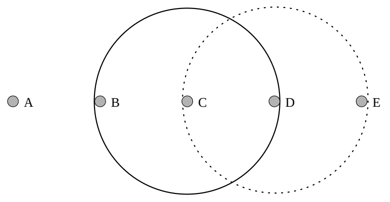
As a result, simulation scenarios involving a normal traffic load require a minimum bandwidth of 4×10 connections$\times512$ bytes per packet (without header overhead)$\times$ 4 packets per second or 80 KBps in the center of the network; thus, the 250 KBps capacity of a simulated wireless channel is ample even taking into account control



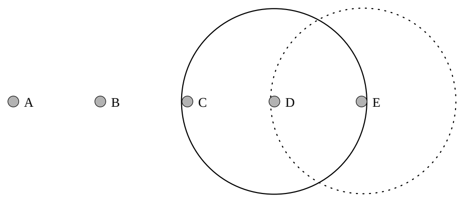
(a) Node A transmits to B



(b) Node B transmits to C



(c) Node C transmits to D



(d) Node D transmits to E

Figure 5.3: A forwarded packet will consume node C's bandwidth four times

overhead and inefficiencies of the MAC layer.

Ad hoc networks may have difficulty scaling to more bandwidth hungry applications though. In bandwidth requirements calculations for ad hoc networks, packets have to be accounted for four times. In a sense, when a node participates in forwarding packets for neighbours, it must assume that packets will consume at least four times their size in bandwidth. Or, from a different perspective, when a node participates in an ad hoc network, its channel capacity should be reduced by 75%.

In many applications, a loss of 75% of channel capacity is unacceptable. Especially in commercial sectors where bandwidth is limited and expensive, ad hoc networking likely cannot compete with traditional cell-based networks. In fact, the use of ad hoc networks can likely only be justified in situations where the amount of data transferred is small, where the amount of bandwidth in plentiful, or where it is infeasible to build an infrastructure of base stations.

5.3 Routing for Scalability

Lastly, the analysis of selective flooding reveals that a major bottleneck of shortest path routing algorithms is the limited channel capacity at the center of ad hoc networks. When packets are routed along shortest paths, more packets have to pass through the center of the network than other parts of the network. As such, the center will be the most congested. For example, if an ad hoc network were to be established in a city, the most congested part would likely be the city center.

Therefore, for better scalability, algorithms should attempt to avoid this bottleneck by routing packets through less congested outer areas of the network. Although shortest path algorithms can be modified to take congestion into account in their calculations, the resulting algorithms will not distribute traffic in an optimal fashion. Compared to algorithms that attempt to improve scalability by reducing the number of control packets exchanged by nodes, routing to avoid bottlenecks may provide superior scalability.

The amount of extra scalability that can be unlocked by specialized scalable algorithms is limited though. Since ad hoc networks are largely bounded in size, simple constant factor or better improvements in the bandwidth efficiency of algorithms can provide equivalent gains. Scalability techniques such as clustering are irrelevant if networks cannot grow to a size where clustering becomes effective.

The bounded size of common ad hoc networks can be demonstrated easily using "back-of-the-envelope" approximate calculations. For example, consider a scenario where as more nodes are added to a network, the geographical area occupied by the network is increased but the node density of the network remains constant:

If there are n nodes in such network and b is the maximum bandwidth available to any single node in the network, then under ideal conditions, the total bandwidth available in the network (defined as the total amount of data that may be transferred between nodes in a set interval) will scale linearly with the number of nodes, resulting in an upperbound for the total bandwidth in the network of nb.

To keep the calculations simple, assume that the nodes are arranged randomly within a square-shaped area. Therefore, if the density of the network is to remain constant regardless of the number of nodes in the network, the area of the square must increase as the number of nodes increase. If each node occupies A square meters, then the area occupied by the network will be An square meters, and the dimensions of the network will be $\sqrt{An} \times \sqrt{An}$.

If communication connections are established between randomly chosen nodes, then because nodes are arranged uniformly and randomly throughout the network, communication connections can be assumed to be between random points in the network.

Since the network is square-shaped, the distance between any two random points in the network is $\sqrt{An}\Delta(2)$ where $\Delta(2)$ is the average distance 2 points in a $[0,1] \times [0,1]$ square.

If the broadcast radius for all communication in the network is r, then the number of hops required for a communication connection is

$$\frac{\sqrt{An}\Delta(2)}{r}$$

Assuming that at any time, the percentage of users involved in communications is a constant μ, μn communication connections will be active in the network at any time. If d is the amount of data sent over each connection, the total bandwidth required in the network is

$$\mu dn \frac{\sqrt{An}\Delta(2)}{r}$$

Or, approximating $\Delta(2)$ with 0.5214 and rearranging the terms, the bandwidth required is

$$\frac{0.5214\mu d\sqrt{A}}{r} \cdot n^{\frac{3}{2}}$$

Notice that the left side of the expression is constant, so the bandwidth required to handle the communication in the network grows faster than a linear rate. Since the total bandwidth in the network only grows linearly, the size of the network is fundamentally bounded.

Another common ad hoc networking scenario involves networks where as more nodes are added the geographical area of the network remains constant but the density of the network increases. In such a network, as the nodes become more closely packed, they will broadcast with less power. Again, under idealized and optimal conditions, the total bandwidth available in the network can be assumed to scale linearly with the number of nodes in the network, so the total bandwidth available in the network is nb.

If the network is in the shape of a square and has an area of A square meters, the density of the network is n/A. If nodes are arranged uniformly throughout the network, then the average area occupied by a node is A/n, so the average distance between nodes is $c\sqrt{A/n}$ where c is some constant.

Because the network maintains a constant area and is square-shaped, the average distance between two nodes will be $\sqrt{A}\Delta(2)$, a constant value. The nodes in the network have a reduced broadcast radius though, so the average number of hops required for a random communication connection is

$$\frac{\sqrt{A}\Delta(2)}{c\sqrt{\frac{A}{n}}}$$

If there are μn nodes broadcasting at a data rate of d, the total bandwidth required in the network to handle this communication is

$$\mu dn\frac{\sqrt{A}\Delta(2)}{c\sqrt{\frac{A}{n}}}$$

Substituting an approximation for $\Delta(2)$ and rearranging the terms gives

$$\frac{0.5214\mu d}{c} \cdot n^{\frac{3}{2}}$$

Again, the bandwidth required for the network grows at a faster than linear rate, whereas the bandwidth grows linearly, meaning that the maximum number of nodes possible in such a network is bounded.

Some networks do exist that are not bounded in size such as those where closer nodes have increased bandwidth, those where nodes have directional antennae, those where nodes can delay forwarding a packet until they move closer to the packet's destination, or those where the number of senders in the network always remain fixed. Nonetheless, most common networks are bounded in size, which reduces the justification for scalable routing. A more thorough examination of related results can be found in a paper by Gupta and Kumar (Gupta and Kumar 1999).

5.4 Future Work

Although selective flooding does show some promise in its ability to route packets in the presence of stale topology information, its large consumption of bandwidth resources make it infeasible for practical use with traditional application contexts. Although modifications to the algorithm can likely be made to reduce this problem, these modifications will likely require major reworking of the algorithm.

Instead, it might be more useful to try to use selective flooding techniques as optimizations to existing algorithms. For example, selective flooding is likely more reliable and more bandwidth efficient at finding new routes after a link failure than the caching and flooding approach used by AODV and DSR.

Selective flooding might also prove useful for networks composed of highly mobile, densely packed nodes with short broadcast radii where routing routes are long and unstable. No application requiring such a network has been developed yet, but the existence of feasible routing for such a network may encourage the creation of such applications.

Overall though, the field of ad hoc network routing algorithms seems to be mature. For most applications, existing algorithms achieve very respectable performance. Most new developments will likely be only incremental improvements over these algorithms. As such, it may be more useful to focus research on some of the more fundamental limitations of ad hoc networks.

Current ad hoc networks are only practical for niche applications such as short messaging services or search and rescue where the amount of data being sent is small compared to the number of nodes in the network. Until the inefficiency of ad hoc networks is resolved, they will remain commercially uncompetitive with other wireless technologies.

New developments in hardware might mitigate this problem however. Better use of spread spectrum technologies might be one solution. For example, ad hoc networks might be able to reserve one channel for signalling but switch to different channels for actual data transmission, reducing the amount of channel contention. The ability to receive multiple communications from multiple channels simultaneously might provide similar performance improvements. Of course, ultra-wide band communication research might be able to supply nearly unlimited bandwidth, making the whole problem of bandwidth inefficiency moot.

The future of ad hoc networking lies in improved MAC layers and improved wireless hardware. Researchers should focus their attentions on these areas to realize the most dramatic improvements in ad hoc networking performance.

Bibliography

ANSI/IEEE Std 802.11-1999. *IEEE Standards for Local and Metropolitan Area Networks: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.*

The CMU Monarch Project's Wireless and Mobility Extensions to ns. CMU Monarch Proejct. 5 August 1999. <ftp://ftp.monarch.cs.rice.edu/pub/monarch/wireless-sim/ns-cmu.ps>

Comer, Douglas and Ralph Droms. *Computer Networks & Internets.* 2nd ed. Prentice Hall Inc., 1999.

Corson, Scott and Vincent Park. "Link Reversal Routing" *Ad Hoc Networking.* Ed. Charles Perkins. Addison Wesley, 2001.

Das, Samir, Charles Perkins, and Elizabeth Royer. "Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks." *Proceedings of the IEEE Conference on Computer Communications (INFOCOM).* March 2000.

Gupta, Piyush and P. R. Kumar. *Capacity of Wireless Networks.* Technical Report, University of Illinois, Urbana-Champaign, 1999.

Johnson, David and David Maltz. "Dynamic Source Routing in Ad Hoc Wireless Networks." *Mobile Computing.* Eds. Tomasz Imielinski and Hank Korth. Kluwer Academic Publishers, 1996.

Johnson, David, David Maltz, and Josh Broch. "DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks." *Ad Hoc Networking.* Ed. Charles Perkins. Addison Wesley, 2001.

Jubin, John and Janet Tornow. "The DARPA Packet Radio Network Protocols." *Proceedings of the IEEE* Vol. 75 No. 1: January 1987.

Kahn, Robert, Steven Gronemeyer, Jerry Burchfiel, and Ronald Kuzelman. "Advances in Packet Radio Technology." *Proceedings of the IEEE* 66(11). November 1978.

Ko, Young-Bae and Nitin Vaidya. "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks." *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM).* October 1998.

Mobile Ad-hoc Networks Charter. IETF. 13 August 2002. <http://www.ietf.org/html.charters/manet-charter.html>

The Network Simulator: ns-2. VINT Project. 25 July 2002. <http://www.isi.edu/nsnam/ns/index.html>

Perkins, Charles and Elizabeth Royer. "The Ad Hoc On-Demand Distance-Vector Protocol." *Ad Hoc Networking.* Ed. Charles Perkins. Addison Wesley, 2001.

Perkins, Charles and Pravin Bhagwat. "DSDV Routing over a Multihop Wireless Network of Mobile Computers." *Ad Hoc Networking.* Ed. Charles Perkins. Addison Wesley, 2001.

Singh, Suresh, Mike Woo, and C. S. Raghavendra. "Power-Aware Routing in Mobile Ad Hoc Networks." *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM).* October 1998.

Steenstrup, Martha. "Cluster-Based Networks." *Ad Hoc Networking.* Ed. Charles Perkins. Addison Wesley, 2001.

Tanenbaum, Andrew. *Computer Networks.* 3rd ed. Prentice-Hall Inc., 1996.

Ramanathan, S. and Martha Steenstrup. "A Survey of Routing Techniques for Mobile Communications Networks." *Mobile Networks and Applications* 1(2). October 1996.

Appendix A

Pseudocode

The pseudocode is divided into five parts: the main loop, a send function, a search function, a receive function, and a forwarding function.

MAIN LOOP()

```
searching := false
do
  if searching == true
    handle-search()
  if node wants to send a packet
    send(packet)

  if node has received a packet meant for this node
    receive(packet)
  else if node has received a packet meant for another node
    forward(packet)
loop
```

SEND(packet)

```
if destination is unreachable
  discard packet
else if distance to destination is known
  if distance data is older than 20 seconds
    broadcast packet as an echo request packet ...
      with distance = node distance + 1 + age / 4
  else
    broadcast packet as a normal packet ...
      with distance = node distance + 1 + age / 4
else // distance to destination is unknown
  broadcast packet as a search packet with distance = 3
  set search timer to trigger later
  searching := true
```

HANDLE-SEARCH()

```
if we know distance to node we are searching for
  searching := false
```

```
    cancel search timer
    rebroadcast search packet as a normal packet ...
      with distance = node distance + 1 + age / 4

  if search timer has expired
    if already searching at the maximum distance
      set destination as being unreachable
      searching := false
    else
      rebroadcast search packet ...
        with distance = old distance + 3
      set search timer to trigger later
```

RECEIVE(packet)

```
  if packet is not a repeat
    set distance to packet's sender to number of hops ...
      traveled by the packet
    if packet is a search packet or echo request packet
      broadcast an echo reply packet to sender ...
        with distance = distance to packet's sender + 1
    else if packet is an echo reply and searching == true ...
      and echo reply satisfies our search
      searching := false
      cancel search timer

    if packet contains data
      pass data up to a higher network layer
```

FORWARD(packet)

```
  if packet is a repeat
    if packet is from a nearby node
      remove any identical packets from the out queue
  else
    set distance to packet's sender to be number hops ...
      traveled by the packet
    if packet is from a nearby node
      discard it
    else
      if distance to packet's destination is unknown
        increment number of undirected hops traveled by the packet
        decrement distance of the packet
        if packet is a search packet and distance > 0
          rebroadcast packet
        else if distance > 0 and ...
          number of undirected hops traveled by packet < 2
          rebroadcast packet
        else
          discard packet
      else // distance to packet's destination is known
        reset number of undirected hops traveled by the packet to 0
        decrement distance of the packet
        if packet is a search packet
          rebroadcast packet
          rebroadcast packet as an echo request packet ...
            with distance = node distance + 1 + age / 4 ...
            + number of hops traveled by packet
        else if packet is not a search packet and ...
          distance of packet >= distance ...
            of node to packet's destination
          rebroadcast packet
        else
          discard packet
```

Appendix B

MAC Layer Issues

Most ad hoc network research revolves around the use of 802.11 hardware, but the 802.11 MAC layer is designed primarily for point-to-point communication, making it ill-suited for the broadcast communication used in selective flooding. As a result, a different MAC layer is used in network simulations of selective flooding.

Although wired networks commonly use Carrier Sense Multiple Access / Collision Detection (CSMA/CD), wireless devices are unable to use such a scheme because they are not able to transmit and receive data at the same time. The 802.11 MAC layer uses Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) to avoid packet collisions instead (ANSI/IEEE Std 802.11-1999). Unfortunately, straight-forward collision avoidance schemes are insufficient for wireless devices because of a problem known as the hidden-terminal problem (figure B.1). This problem occurs when two nodes that are not within broadcast range of each other both believe that the channel is clear and simultaneously start transmitting data to the same target node. Although this target node is within broadcast range of the

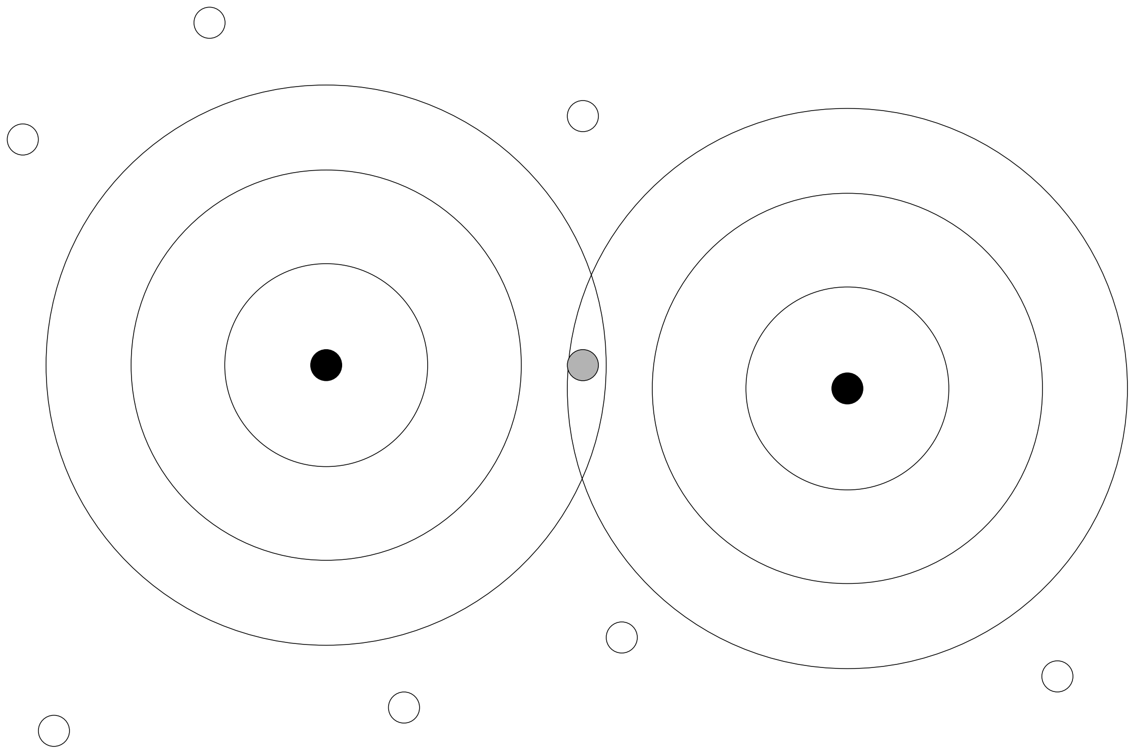


Figure B.1: The hidden terminal problem

senders, it will not correctly receive either of the transmissions due to collisions. Because the two senders are outside of each others' broadcast range, they are not aware of each other and are unable to negotiate access to the channel.

The MAC layer of 802.11 avoids this problem by using a virtual carrier sense strategy (figure B.2). In this system, a sender initially sends out a Request to Send (RTS) packet when it wants to send some data. This RTS packet specifies the destination that the data is intended for. When the destination hears this RTS packet, it replies with a Clear to Send (CTS) packet, authorizing the sender to send. The sender then sends its data packet. Upon receipt of this data packet, the destination acknowledges receipt of it. If during this process the sender does

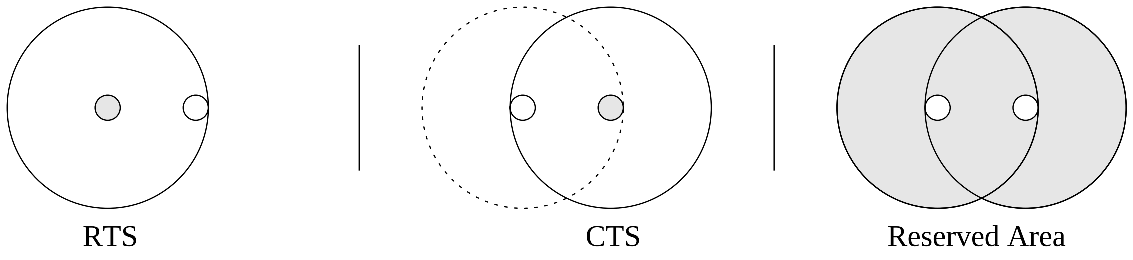


Figure B.2: 802.11 RTS/CTS scheme

not receive a CTS or acknowledgment, it will pause for a random amount of time and then restart the process. Nodes other than the sender and destination are supposed to suppress their own broadcasts upon hearing a RTS or CTS packet, so they will not interfere with the transmission. Although the retransmissions in this algorithm occasionally interfere with the timing of TCP retransmissions, usually this behaviour is not a significant problem.

Unfortunately, this scheme is only suitable for point-to-point communication. For broadcast communication, 802.11 cannot use the virtual carrier sense scheme described above but must instead rely on traditional physical carrier sense techniques. A node that wants to send data will listen to check if the channel is clear. If it is clear, it will transmit, but if it is not clear, it will delay transmission for a random amount of time. This scheme is vulnerable to the hidden terminal problem. Since flooding techniques rely heavily on broadcast communication, the use of the 802.11 MAC layer with such algorithms seriously hinders their performance due to the excessive number of collisions that occur.

As a result, a different MAC layer is needed in simulations of flooding-based algorithms. One alternative is a null MAC layer that allows all transmissions to be received despite collisions, but the results produced are too unrealistic to allow for

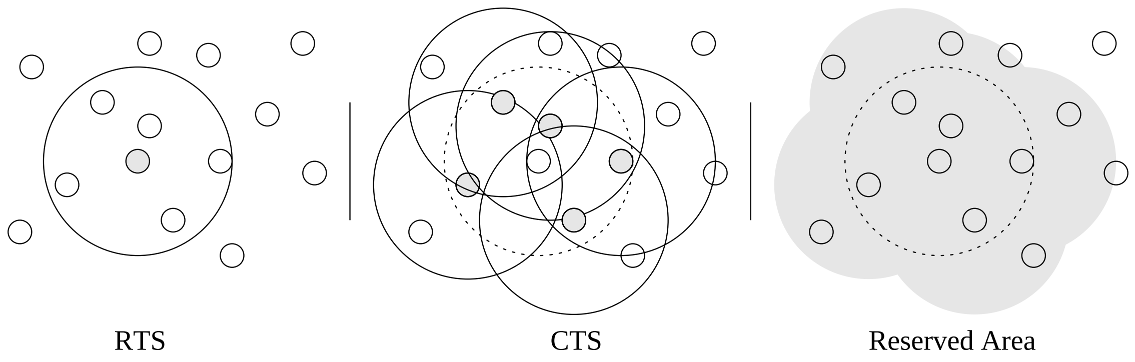


Figure B.3: Behaviour of the new MAC layer

useful comparisons with existing algorithms.

Instead, a new MAC layer is used. This MAC layer is designed to be more reliable for broadcast communication. In this alternate MAC layer, a sender signals that it wants to send by sending out a RTS packet (figure B.3). Any node that receives this RTS packet then sends out a special CTS signal that jams all communication within broadcast range. Any node that detects this jam signal then stays silent to allow the sender to send its packet. Essentially, any node within two times the broadcast radius of a transmitting node will stay silent during the transmission, allowing any node within the immediate broadcast radius of the transmitting node to receive the signal. Unlike 802.11, there is no packet acknowledgment meaning that detection of lost packets must occur at a higher layer in the network stack. A simple finite state machine diagram of the ns-2 implementation of this algorithm is shown in figure B.4. Although this MAC layer is more reliable than the 802.11 MAC layer for broadcast communication, it is still vulnerable to the hidden terminal problem. The length of the vulnerable period per transmission is twice the propagation time plus the length of a RTS packet. In addition, if a node receives certain sequences of signalling packets, it will not be able to control access to the channel properly.

Although this MAC layer may not actually be implementable in practice, it provides a better indication of the type of performance a flooding algorithm can achieve with an appropriate MAC layer than the use of 802.11 or null MAC layers.

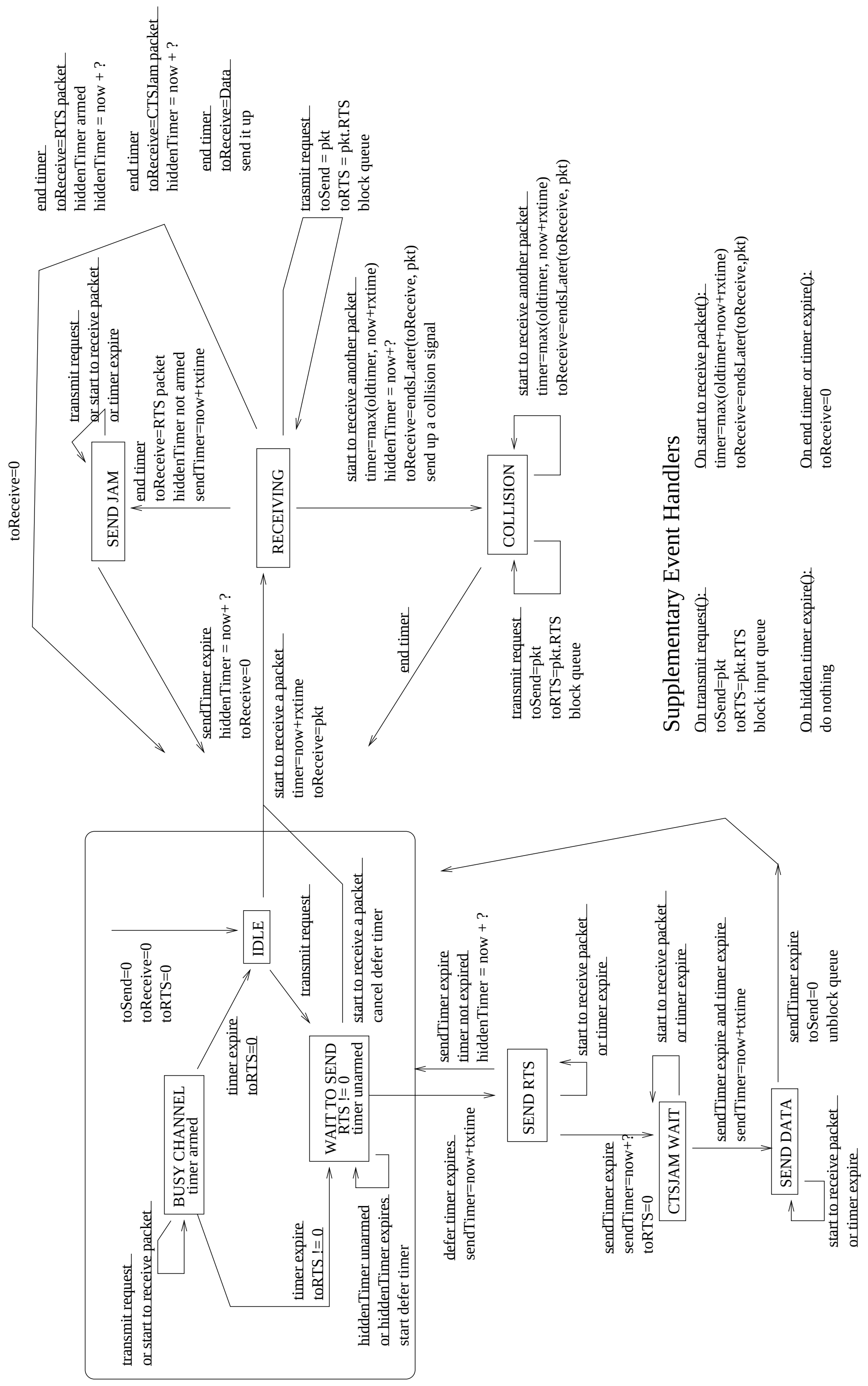

IDLE
BUSY CHANNEL timer armed
WAIT TO SEND RTS != 0 timer unarmed
RECEIVING
SEND JAM
COLLISION
SEND RTS
CTSJAM WAIT
SEND DATA
toSend=0 toReceive=0 toRTS=0
transmit request
timer expire toRTS=0
timer expire toRTS != 0
transmit request or start to receive packet
hiddenTimer unarmed or hiddenTimer expires start defer timer
start to receive a packet cancel defer timer
start to receive a packet timer=now+rxtime toReceive=pkt
defer timer expires sendTimer=now+txtime
sendTimer expire timer not expired hiddenTimer = now + ?
start to receive packet or timer expire
sendTimer expire sendTimer=now+? toRTS=0
start to receive packet or timer expire
sendTimer expire and timer expire sendTimer=now+txtime
sendTimer expire toSend=0 unblock queue
start to receive packet or timer expire
toReceive=0
transmit request or start to receive packet or timer expire
end timer toReceive=RTS packet hiddenTimer not armed sendTimer=now+txtime
sendTimer expire hiddenTimer = now+ ? toReceive=0
end timer toReceive=RTS packet hiddenTimer armed hiddenTimer = now + ?
end timer toReceive=CTSJam packet hiddenTimer = now + ?
end timer toReceive=Data send it up
trasmit request toSend = pkt toRTS = pkt.RTS block queue
start to receive another packet timer=max(oldtimer, now+rxtime) hiddenTimer = now+? toReceive=endsLater(toReceive, pkt) send up a collision signal
start to receive another packet timer=max(oldtimer, now+rxtime) toReceive=endsLater(toReceive, pkt)
transmit request toSend=pkt toRTS=pkt.RTS block queue
end timer
Supplementary Event Handlers
On transmit request():
toSend=pkt
toRTS=pkt.RTS
block input queue
On start to receive packet():
timer=max(oldtimer+now+rxtime)
toReceive=endsLater(toReceive,pkt)
On hidden timer expire():
do nothing
On end timer or timer expire():
toReceive=0

Figure B.4: State diagram for the new MAC layer