On-demand Routing in Disrupted Environments

J. Boice
J.J. Garcia-Luna-Aceves
K. Obraczka

UCSC-CRL-06-16
October 25, 2006

Department of Computer Engineering
University of California at Santa Cruz

ABSTRACT

While current on-demand routing protocols are optimized to take into account unique features of mobile ad hoc networks (MANETs) such as frequent topology changes and limited battery life, they often do not consider the possibility of intermittent connectivity that may lead to temporary partitions. In this work, we introduce the Space-Content-adaptive-Time Routing (SCaTR) framework, which enables data delivery in the face of both temporary and long-lived MANET partitions. SCaTR takes advantage of past connectivity information to effectively route traffic towards destinations when no direct route from the source exists. We show through simulations that, when compared to traditional on-demand protocols, as well as Epidemic routing, SCaTR increases delivery ratio with lower signaling overhead in a variety of network scenarios with intermittent connectivity. We also show that SCaTR performs as well as on-demand routing in scenarios that are well-connected, and/or have no mobility predictability (e.g., scenarios with random mobility).

1. Introduction

The price, performance, and form factors of sensors, processors, storage elements, and radios today are enabling the development of network-supported applications in very disrupted environments, i.e., environments where end-to-end connectivity is not guaranteed at all times because of either the characteristics of the environment or the normal operation of the network nodes. Examples of such applications and environments include monitoring of disrupted phenomena (e.g., wild fires), object tracking, establishment of on-demand network infrastructure for disaster relief or military purposes (in which case, the ad hoc network can be disrupted by terrain, weather, and other natural phenomena, as well as jamming, interference, etc.), peer-to-peer vehicular or interpersonal networks [4] with very sparse connectivity, and mobile ad hoc networks (MANETs) that need not be connected at all times in order to limit interference and contention. In these scenarios, network disconnection is the normal state of operation rather than an exception.

The demand for networking in environments prone to intermittent connectivity poses a challenge because the architects of the IP Internet and even MANETs have assumed that physical connectivity exists on an end-to-end basis between sources and destinations for extended periods of time, or at least for the duration of a transaction among communicating parties. This assumption has had profound implications on how communication bandwidth is shared, how routing is accomplished, and how messages are disseminated across computer networks. In particular, routing in packet-switching networks has been based on routing tables that specify the next hop to one or more destinations. Such routing information is derived entirely from topology (or connectivity) information that represents only a snapshot of the state and characteristics of network links at particular instants.

Regardless of the specific mechanisms used in a routing protocol today (e.g., proactive or on-demand routing), computing the routing table entry for a given destination can be viewed as a particular form of searching a database. The routing database can be replicated (as it is done in such topology broadcast approaches as TBRPF [15] and OLSR [3]) or distributed, as in AODV [17] or DSR [9]. Depending on whether the routing database is replicated or distributed, the search algorithm can be centralized (e.g., using Dijkstra's shortest path first algorithm) or distributed (e.g., using a flood search based on route requests and route replies). The routing databases constructed by traditional routing algorithms specify the instantaneous status of a link (up or down), and the value of its parameters such as delay and bandwidth at some specific point in time. The search for routes in such databases produces snapshot paths that have no temporal dimension. Hence, if the network connectivity or link parameters change, multiple paths to destinations may be affected; the only way most current routing protocol can recover is to search for new paths. This time-independent, reactive approach to changes in network connectivity and link parameters works well as long as the disruptions in network connectivity due to environmental or operational reasons

are not so frequent and/or long-lived that they prevent the routing protocol from obtaining time-independent paths to intended destinations.

Starting with the work in the Interplanetary Internet Research Group (IPNRG) [2] of the IRTF (Internet Research Task Force), considerable effort has recently been devoted to the study of networks with intermittent connectivity or very long latencies. Perhaps most prominent in this area is the work by the DTNRG (Delay Tolerant Networking Research Group) [6], which started in 2002 under the IRTF. Section 2 summarizes prior related work on routing in disrupted environments. From our summary of related work, it becomes apparent that no complete solution exists for on-demand routing that incorporates the network topology's time dependency.

In this paper we describe the SCaTR (Space-Content-adaptive-Time Routing) framework to enable on-demand routing in MANETs with intermittent connectivity. Section 3 describes SCaTR which we currently implement by extending the On Demand Ad Hoc Distance Vector Routing Protocol [17] (AODV). Our current instantiation of SCaTR is such that, if the network is connected, it operates exactly as regular on-demand routing, in this case AODV. However, if no direct route is available from source to destination, a node that is deemed closer to the destination than the source will advertise itself as a *proxy*. In this manner, we are assured that the resulting protocol will do no worse than standard AODV in well-connected environments, and better in partitioned networks.

Section 5 addresses the performance of SCaTR compared to on-demand and Epidemic routing. Scenarios involving both random and predictable node mobility are investigated. In predictable mobility scenarios, node schedules or trajectories are not assumed to be global knowledge. Instead, the routing algorithm in SCaTR uses mobility histories to improve performance. Our simulation results show that the added functionality of proxies in SCaTR improves delivery rates in both predictable and random mobility situations, while incurring lower signalling overhead. Given enough time, the protocol delivers all possible packets to their intended destinations, achieving optimal reliability. Section 6 summarizes our contributions and discusses ideas for future work.

2. Related Work

Many routing protocols [9, 16] exist for ad hoc and sensor networks. Most traditional protocols do not account for networks that are frequently disconnected, and thus do not exhibit adequate performance in such scenarios. For instance, AODV [17] uses a feature called *local repair*, which allows intermediate nodes to buffer data packets in the event that a route is lost. However, this feature is only activated if a route existed at one point in time, which is unlikely in a highly partitioned network.

Recently, *Message Ferrying* [22] has been investigated for use in highly partitioned networks. The approach utilizes special nodes called ferries whose mobility can be controlled to maintain communication between partitions. Much of the work has focused on route scheduling of the ferries, and synchronization between their routes, since a well-chosen schedule will have a great impact on timely and reliable message delivery. It has also been shown that the ferries can be used as an energy saving device for other nodes in the network; if there are no ferries nearby, nodes can be turned off to conserve energy. This work makes the assumption that controllable nodes exist in the network, however, there are situations in which these nodes are not available. Our work addresses these situations.

Data *MULES* [18] address data delivery in static sensor networks. Like *ferries*, *MULES* are also controllable mobile devices that move throughout the sensor network to receive and deliver data where appropriate. They are a solution to data delivery in a sparse sensor network whose data could otherwise not be reported. Their motion is assumed to be controllable as well.

In the *Epidemic routing* [14] approach, data are transmitted through message exchanges between neighboring nodes. The premise behind the work is that by duplicating a message and sending it to all neighbors, it can quickly and reliably reach its destination. Similar work [11] has been done to improve Epidemic routing by limiting the number of nodes to which messages are sent. Epidemic routing, however, relies on data duplication to deliver messages which can be prohibitively expensive in terms of energy consumption.

Delay Tolerant Networking [6] has attracted considerable interest from the network research community for several years now. Additional layers atop the routing and transport layers have been proposed to provide store and forward capabilities, as well as interoperability services to partitioned networks. This work has also examined the idea of *custody* [7], which is a hop-by-hop method of reliability for disconnected networks. This is a much-needed approach to the issue of reliability, as end-to-end reliability is unmanageable in these scenarios.

A more general framework for routing using knowledge of network mobility has also been proposed [12]. This approach adds the *time* dimension to routing tables, and selects routes based on a combination of the data's destination and the time of message arrival. It uses global knowledge of node mobility schedules to construct these routing tables. In some scenarios, these schedules may not be available, or too expensive to maintain.

Spray and Wait [20] is a recent improvement over a pure flooding protocol such as Epidemic routing. In this work, only the source can replicate a message, and the amount of replication is proportional to the number of nodes in the network. It is shown that the method can bound delay proportional to the optimal delay. After 'spraying' several copies of a message, the host 'waits' until one is delivered.

Location information is a useful tool for routing in disrupted networks. Both MobySpace [10] and MV routing [1] are methods that use location information to aid routing decisions. They assume that a node who has visited a particular location is likely to revisit it, and therefore is a good candidate to carry messages to that location. MV routing uses location information to facilitate buffer management, while MobySpace is a framework for generating probabilities that a nodes will move to specific locations in the future. Both methods require some sort of localization method such as GPS.

Work has also been done to predict the future topology of a network based on the current properties of nodes [21]. The main motivation behind this work is to determine for *how long* nodes that are connected will remain connected. This allows routing protocols to pro-actively search for a new route when a link is expected to break. Metrics such as node speed, direction, and radio propagation range are factored into the estimate, and these metrics could also be good indications of future node meetings.

While there has been significant prior work on the topic of partitioned networks, most of the approaches have made one of three assumptions. One assumption is global topology knowledge, which has proved to be useful in determining future connectivity. Another assumption is the existence of controllable nodes, which can be extremely useful in aiding delivery in sparse networks. The third assumption is that data can be duplicated freely among nodes, which can lead to excellent delivery ratios. Our work is an attempt to do away with these assumptions, yielding a more general framework applicable to any network scenario where any such assumption may be unrealistic.

3. SCaTR

The SCaTR framework extends on-demand routing, taking action only when direct routes cannot be established by the underlying protocol. In the case of a route discovery failure, i.e., the source and destination are in separate partitions, SCaTR tries to route data to the node or nodes in the source's partition that are likely to have a route to the destination in the near future. These nodes act as *proxies* for the destination and buffer messages until either the destination is discovered, or another node is selected as a better proxy for those messages. Messages are replicated at most one time, resulting in minimal data replication and duplicate message filtering overhead.

Proxies are selected based on past connectivity information which nodes keep in content-adaptive *contact tables*. As it will become clear, contact tables, which are the equivalent of traditional routing tables, use time-dependent and space-dependent routing metrics. These metrics differ for different types of content or local constraints such as buffer size. For instance, if a proxy is running low on buffer space, it may decide to select as the next proxy for that destination the first node it hears from that has been in contact with the destination; this is done even if the node's *contact value* is lower that its own; however, if the node has higher buffer availability, it can carry the data for a longer interval.

Because SCaTR takes no action if routes are successfully established, we are guaranteed that it will perform *no worse* than the underlying on-demand routing protocol in any situation. This diversity makes it well suited for adoption in any network scenario. SCaTR consists of several phases: contact table maintenance, route discovery, route selection, and proxy rediscovery. The general behavior of the SCaTR framework is shown in Figure 3.1. Each phase is described in detail below.

3.1 Contact Table Maintenance

Each node in the network maintains a contact table containing a measure of time-dependent distances to other nodes in the network. Each entry in the table consists of a destination address and its current contact value. Nodes maintain these tables with information that is piggybacked onto hello messages; when a node receives a hello message from a neighbor, it also receives that node's contact table. It uses this table to make changes to its own contact table.

Because it can be very expensive to maintain contact information about all nodes in a network, SCaTR initializes its contact tables on-demand. Each node starts with an empty table, and adds destinations only when it receives a request for that destination, or meets another node who has an entry in its table for that destination. This method of initialization results in a delay for the first messages introduced into the network because the contact table request must propagate to the destination and then back to the source before proxies are advertised. In large

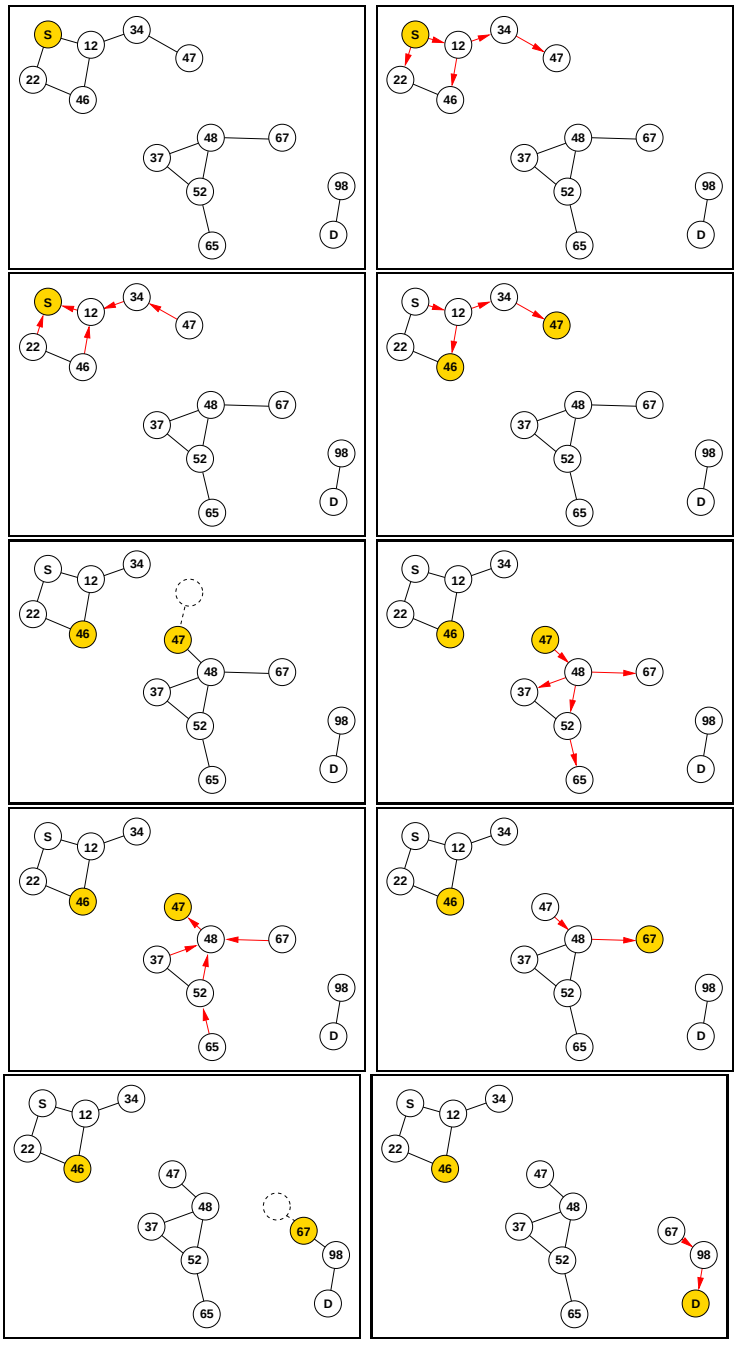

Figure 3.1: An overview of the SCaTR framework. The source node, at top left, has data (indicated by shading) to send to the destination, at bottom right. The source node first initiates proxy discovery in its local partition. After finding the two nodes with the best contact values for the given destination, the source selects them as proxies and sends them data to be buffered. When one of the proxies joins a new partition, it initiates the proxy discovery process, and selects the best available proxy. Finally, a proxy reaches the partition containing the destination and delivers the data.

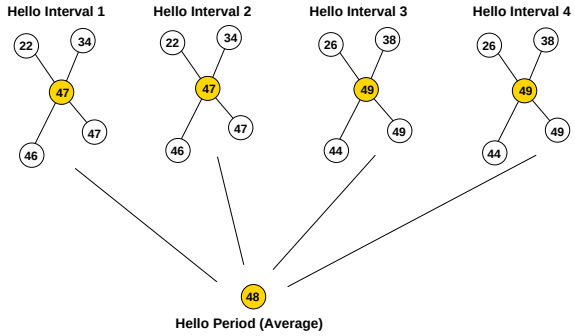

Figure 3.2: The value for a hello period is found by averaging τ successive hello intervals.

networks with predefined sinks or a limited number of destinations, however, this method saves significant computational and communication overhead.

The mechanisms used to maintain and update contact values have several important characteristics that must be addressed:

- *Ordering*: The highest contact value advertised for any node *must* be advertised by the node itself. A node that will reach a destination sooner than another node should have a higher contact value for that destination.
- *Stability*: Contact values should not fluctuate greatly over time in order to minimize 'leapfrogging' of contact values (the case in which nodes alternately select each other as proxies due to asynchronous contact value updates).
- *Simplicity*: Contact value calculation should not be computationally complex, as it must be frequently invoked.

To address these issues, we propose an algorithm that is initiated with a maximum contact value advertised by destinations themselves. Time is broken into *hello intervals*. During a hello interval, nodes maximize their neighbors' advertised contact values for each destination. These values are averaged over a *hello period*, and each node maintain a window of periods, used to generate the next interval's contact value for each destination. A more detailed description of contact value maintenance follows.

3.1.1 Hello Intervals and Periods

The length of a hello interval is the amount of time between hello message advertisements. During each hello interval, a node maintains the maximum contact value it receives from its neighbors for each destination. The hello period consists of a sequential group of hello intervals as defined by the parameter τ. At the end of a hello period, a node averages the values obtained during each hello interval to obtain a single value for the entire period for each destination. Figure 3.2 illustrates a hello period with $\tau = 4$; the center node averages its hello values for each interval to determine its value for the entire period. The selection of τ is discussed in Section 3.6.

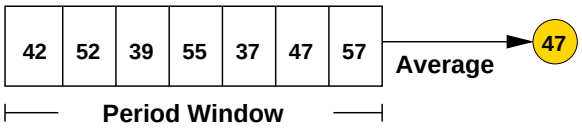

Figure 3.3: A node's current contact value is determined by averaging its values for the past κ hello windows.

3.1.2 Period Window

The length of time for which a node retains contact information is specified by the period window κ. At the end of each hello period, the value obtained during the most recent period is added to a queue, and the oldest value is removed. Each node's contact values are recalculated at the end of each period by averaging the most recent κ periods. Until the node's contact value is updated again, it will advertise this value to its neighbors. Figure 3.3 illustrates a node's contact value when calculated at the end of a hello period with $\kappa = 7$. Section 3.6 discusses the selection of κ.

3.2 Route Discovery

The main mechanisms for route discovery in the SCaTR framework are the Proxy Request (PREQ) and Proxy Reply (PREP) messages. When the underlying routing protocol is unable to establish a route to a destination, the source node will issue a PREQ to its current partition. This is a request for a candidate node or nodes in the source's partition to buffer messages and carry them towards their destination.

The PREQ can request one or more destinations to reduce signaling overhead. Instead of sending out an individual PREQ for each destination, the node will send a single PREQ with all destinations for which it has buffered data. The PREQ also contains the source node's contact value for each destination that is being requested. The contact value is included so that only nodes with a significantly better value reply to the request. In our implementation, we have used a threshold of 5% to determine whether or not a proxy is better; a proxy is only selected if its contact value is 5% higher than the node at which the messages currently reside. Different threshold values could be used and will determine how selective a node is in choosing proxies. In preliminary experiments, we found 5% to be an effective threshold to balance overhead and delivery ratios.

The PREP is a message that advertises the responding node as a possible proxy for one or more of the destinations in the PREQ. The source can then decide which proxy or proxies to use for each destination. The PREP contains the proxy's contact value for the destination, as well as its remaining buffer space, and the number of messages it has already buffered for the source-destination pair. It is important to note that the PREP specifies the destination, not the proxy itself. The source need not know the address of the proxy; it must only know the

next hop towards the destination. The source-destination pair obtained from the PREQ packet is entered into a buffering table at the proxy, and any packets that are later received by the proxy with this source-destination pair are buffered.

Each node receiving a PREQ takes action based on its contact values for the requested destinations and the source's contact values that are included in the request. For each destination included in the PREQ, there are several possible actions that can be taken by a receiving node. These are outlined below:

3.2.1 PREQ Case 1

The receiver has no entry for the destination in its contact table. If a receiving node has no contact information for a destination, the destination is added to the receiver's contact table. The receiver will now begin to maintain contact information for that destination. The PREQ is rebroadcast as received.

3.2.2 PREQ Case 2

The receiver's contact value for the destination is not better than the source's by the specified threshold. In this situation, the receiver does not reply to the request and rebroadcasts the PREQ as received. It is not a valid proxy for the destination.

3.2.3 PREQ Case 3

The receiver has an improved contact value for the destination. In this case, the receiver responds to the source as a possible proxy for that destination. It adds the destination to its buffer table, and any subsequent data packets that are received are buffered. It is possible that there are other proxies for this destination in the partition, so the PREQ must be rebroadcast. It is rebroadcast with the *receiver's* contact value for the destination to reduce the number of nodes that reply to the request.

3.2.4 PREQ Case 4

The receiver has an active route to the destination. In the case that the receiver has an existing route to the destination, it advertises the route to the source. This indicates that the destination is in the same partition as the source, and SCaTR functionality is not needed. The source will add this destination to its routing table as an ordinary route. Because there is no need to find proxies for the destination, the PREQ is not rebroadcast.

3.2.5 PREQ Case 5

The receiver has already processed a PREQ for the given destinations in the recent past. In this case, the receiver takes no action, and the PREQ is not rebroadcast. To suppress duplicate requests, we use the same timer value as AODV and uniquely identify PREQs with the time at which they were issued as well as the included source-destination pairs.

3.3 Route Selection

As the source collects PREPs from the nodes in its partition, it compares contact values for each of the destinations. If a new PREP has a higher contact value than the one currently in its routing table, it replaces the entry. Routing tables may now contain two types of routes. One is an active, connected route, while the other is a route to a destination that was advertised by a proxy. These are distinguished only because a reply that advertises a direct route will always take precedence over a proxy route.

3.4 Proxy Rediscovery

After a proxy buffers packets, it must ensure that those packets reach their destination. Proxies have several methods of initiating the route discovery process for buffered messages. One method is 'listening' as updates to its contact table are received. If it receives an improved contact value for a destination for whom it has buffered packets, the node assumes that a better route to the destination is available in its partition, or that the destination itself is nearby. This information initiates route discovery behavior in the proxy. The proxy sends out a single PREQ for all nodes for which it has buffered packets. In the same manner as the route discovery process described above, nodes reply to this request, and the proxy selects the best next destination for the data. The proxy also listens to any relevant RREPs or PREPs that it relays, and if route information for destinations in its buffer is relayed, a route is established along the advertised path. In the case that a proxy does not make any updates to the contact value for a buffered destination, or relay a reply for that destination, it sends a PREQ periodically. The length of time used for this timeout is explored further in Section 3.6.

3.5 Message Replication

There are many approaches to message replication, ranging from the minimal approach of a single copy of each message, to the unlimited duplication (up to the number of nodes in the network) of Epidemic routing. More message redundancy requires increased overhead in the form of replicated data, as well as the computational and memory requirements required to maintain buffers and detect duplicate messages.

One approach to the problem is to allow each proxy to replicate a message a certain number of times. However, as the network scales, it is clear that this number will grow exponentially. In addition, a proxy holding a message has no indication of whether another proxy has successfully delivered the message already. Thus, any replicas made by this proxy are in waste. This clearly is not a feasible solution. It is also possible to control replication in a tree fashion with the message source as the root of the tree. There can be a defined branching factor or replication degree that each level of the tree is permitted to use. Using the depth of a message in the tree and the branching factor, one can specify an exact number of duplications for each message. This reduces message duplication, but has the same problem of communicating message deliveries as mentioned above.

In SCaTR, we have decided to minimize the amount of message replication. Prior work by one of the authors [5] has indicated analytically that under constrained conditions, the optimal number of message duplications is one. The overhead resulting in any further duplication is not justified by the increase in delivery ratio. For these reasons, the source can select at most two initial proxies for message transmission, and proxies may not duplicate messages.

In addition, it is possible to introduce messages into the system that request the deletion of packets that have already been delivered, sometimes referred to as antipackets [19]. While these are useful with a high degree of message replication, they are not necessary in SCaTR.

3.6 Parameter Selection

As in any protocol, parameter selection in SCaTR can have significant performance implications. In particular, the parameters τ (the length of the hello period), κ (the length of the period window), and the PREQ frequency should be set such that the protocol retains only useful history information, avoids the leapfrogging problem, and minimizes transmission overhead, computation, and memory requirements.

A larger value of τ indicates more coarsely-grained time periods, while a smaller value of τ maintains finer time periods. Shorter periods have the advantage of increased sensitivity to topology changes, since each hello period represents fewer time-based snapshots of the surrounding topology. However, shorter periods also result in more computation and a higher chance of leapfrogging. For the experiments we present in this paper, we use $\tau = 10$, which we found to be an effective value through preliminary simulations. Of course, depending on the network and target application, the value of *tau* can be tuned accordingly.

Similarly, a larger value of κ will maintain more mobility history, resulting in routing decisions based on behavior further into the past. While more history has the advantage of increased information, it requires more memory and may result in the use of outdated information for route selection. Thus, it is important to approximate for how long information should be maintained, and use this approximation, combined with the available memory resources at the node to

define κ. It is difficult to determine, from the contents of the history queue itself, how much history is necessary. For the simulations presented here, we set the queue size to 1000.

The amount of time a proxy waits before sending a PREQ is also an important feature in the protocol. A wait time that is too short will result in wasted request messages being flooded to the proxy's partition. If the wait time is too long, however, a node risks missing a particularly transient route. We propose an adaptive timeout similar to that of Ethernet's binary exponential backoff [13]. Each node is seeded with an initial time to wait before initiating a request, and if the request fails, it waits for twice that amount of time. For example, if a node is seeded with a 5 second timer, it will successively wait 5, 10, 20, 40, 80, etc. seconds until its buffer is empty. This timer is reset once a proxy has no messages remaining in its buffer.

4. Simulation Setup and Mobility Models

The SCaTR framework is implemented as an extension to the AODV routing protocol and simulated in GloMoSim [8] with several mobility scenarios and connectivity models. Epidemic routing was implemented for comparison to SCaTR, in which messages are passed to all neighbors. Throughout the experiments, 18 CBR data flows generate messages at ten second intervals for the first 400 seconds of the simulation. The experiments run for 2000 seconds to give messages time to propagate to their destination, and use an 802.11 MAC layer. All experiments were run with 5 seed values and the results averaged. The two mobility scenarios described below, in addition to the random waypoint model, show the effectiveness of the protocol over varied topologies and connectivity models.

4.1 Gridded Random Waypoint Setup

The gridded random waypoint scenario has been included to illustrate a network in which nodes move randomly but in predefined areas. The scenario provides a situation in which connectivity is limited, and nodes must be selected as proxies due to their distance to the destination.

Nodes are arranged in a square field with a 377m radio propagation range. Within each square, a fixed number of nodes move according to the random waypoint mobility model; random locations are selected within the square, and a node moves there at a rate of between 5m/s and 10m/s. After reaching its destination, a node pauses for 20s. The flows exist between nodes on opposite ends of the grid to provide maximum route lengths. Throughout the experiments with this mobility model, the number of nodes and flows are fixed, while the dimensions of the scenario are varied to provide more or less connectivity in the network. A sample topology is shown in Figure 4.1.

4.2 Scheduled Routes Setup

The scheduled routes scenario was generated to illustrate predictable motion in a network. Because each node follows a predefined path, it provides a situation where past topologies are a good indication of future connectivity. In this case, the contact tables of SCaTR can accurately represent the distance to destinations.

For this setup, 50 nodes are arranged in a square field with a radio propagation range of 377m. Ten nodes are positioned around the perimeter of the network, and act as sources and destinations. All other nodes are assigned a randomly sized and positioned rectangle over which to travel at a random speed of between 5m/s and 20m/s. Using these parameters, links will be somewhat predictable, although they will not always occur at the same interval. Throughout the experiments, the size of the scenario is increased to provide less connectivity in the network. A sample topology is shown in Figure 4.2.

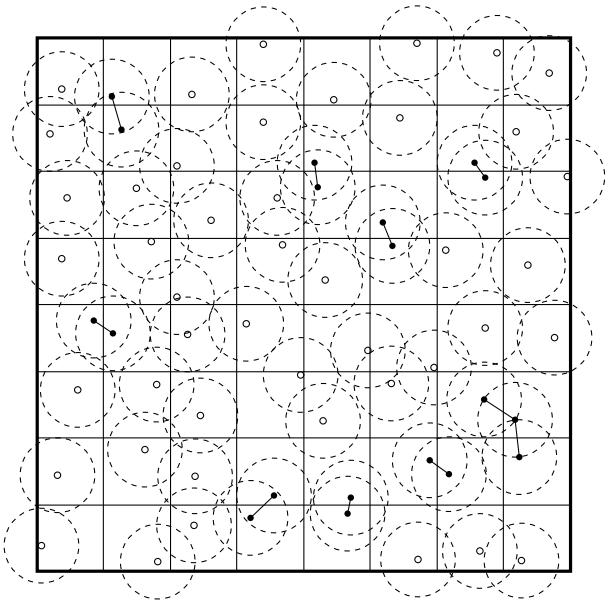

Figure 4.1: The gridded random waypoint mobility model: each node moves randomly within its square in a grid. The dotted circles indicate the radio propagation limits. Source and destination nodes are all at the edges.

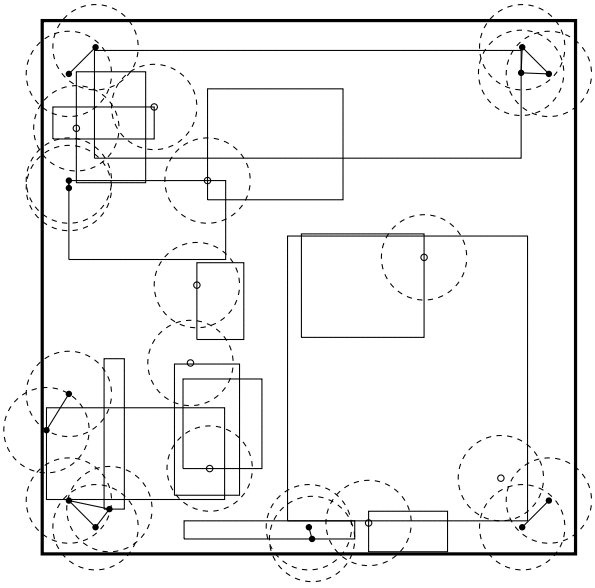

Figure 4.2: Scheduled routes mobility model: the rectangles indicate scheduled node mobilities and the dotted circles indicate their radio propagation limits. Source and destination nodes are placed around the perimeter. The network shown is very sparse for clarity.

4.3 Random Waypoint Setup

We also run simulations with the random waypoint mobility model to illustrate the performance of SCaTR in a purely random network; in this case, past mobility information gives no indication of future topologies. This scenario is especially interesting as it is not favorable to SCaTR. 40 nodes move according to the random waypoint mobility model within a square area 3000m x 3000m. Due to the sparseness of the network, the connectivity is poor. The experiments vary the speed of each node to change the duration of connected paths in the network. Pause time remains a constant 30s throughout the simulations.

5. Performance Comparison

In this section we show that the addition of SCaTR to an on-demand routing protocol results in significantly improved performance in terms of delivery ratio and signaling overhead in all mobility scenarios that were tested. In these experiments, signaling overhead includes control messages such as route requests and replies, as well as data packet replicas. Because the networks simulated are sparse, it is often not possible to deliver all messages in the allotted time. All simulations were run for 2000 seconds, and it should be noted that undelivered packets after this time were not lost; they have not yet reached their destination. It would be possible to run much longer simulations, and have much higher delivery ratios, but the time selected gives a good indication of the relative performance between protocols. In a subsequent experiment, we show how lengthening simulation time affects performance.

5.1 Scheduled Routes Results

We first compare routing methods in the scheduled routes mobility scenario. Connectivity is varied by increasing the size of the scenario; clearly a larger scenario size results in less connectivity. Experiments with scheduled routes show that SCaTR takes advantage of predictability to provide high delivery ratios with low signaling overhead. Figure 5.1 illustrates delivery rates for the various protocols. At small scenario sizes with good network connectivity, all protocols successfully deliver some messages. Epidemic routing delivers, as expected, 100% of messages, while SCaTR delivers approximately 96%. AODV, due to frequent route disruptions, delivers only 6% of messages. Epidemic routing maintains a 100% delivery ratio until the scenario size reaches 2500m x 2500m. Most notably, SCaTR sustains similar delivery ratio to Epidemic routing throughout the experiment. At 4500m x 4500m, connectivity is extremely poor, and none of the protocols deliver more than 40% of messages.

The plot of signaling overhead in Figure 5.2 shows that SCaTR maintains its high delivery rates with little overhead. Since those packets that reach their destination with Epidemic routing likely reach all nodes in the network, overhead remains at nearly 50 messages per delivery. The addition of SCaTR enables AODV to maintain fairly constant overhead, much lower than that of Epidemic routing. The 'hump' in the plot for AODV occurred because partitions at the start of the simulations surrounding the sources were large. Thus, as AODV tried its route establishment attempts, they reached a high number of nodes (but not the destination). This behavior had a large impact on overhead since so few packets were delivered.

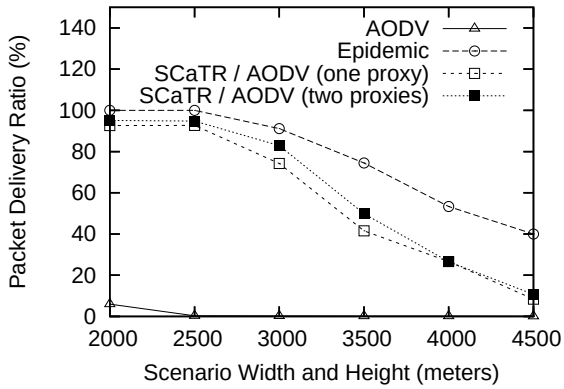

Figure 5.1: Delivery ratio for the scheduled routes mobility model with varied network connectivity.

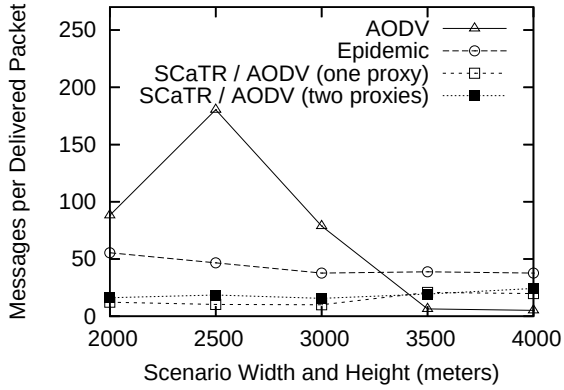

Figure 5.2: Signaling overhead for the scheduled routes mobility model with varied network connectivity.

5.2 Gridded Random Waypoint Results

The results for the gridded random waypoint model, shown in Figure 5.3, are similar to that of the scheduled routes scenario. AODV's performance decreases sharply as connectivity diminishes, while SCaTR maintains delivery ratios similar to Epidemic routing. In addition, Epidemic routing and SCaTR both show very high delivery ratios until the network is extremely disconnected. Notably, in both scenarios, the results show that adding a second proxy to SCaTR does not have a significant impact on its delivery ratio, although it does incur additional overhead. It is likely that a network with many more nodes would show the benefit of the second proxy more clearly, however these results indicate that it is not a significant factor.

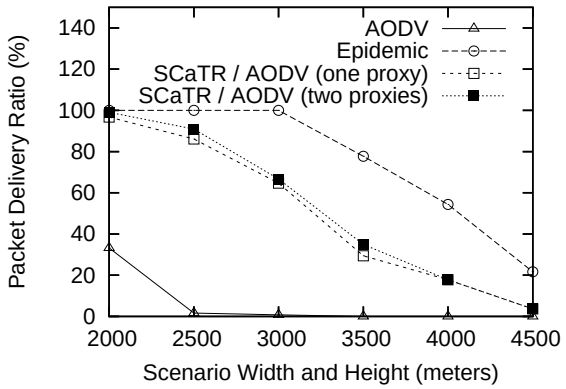

Figure 5.3: Delivery ratio for the gridded random waypoint mobility model with varied network connectivity.

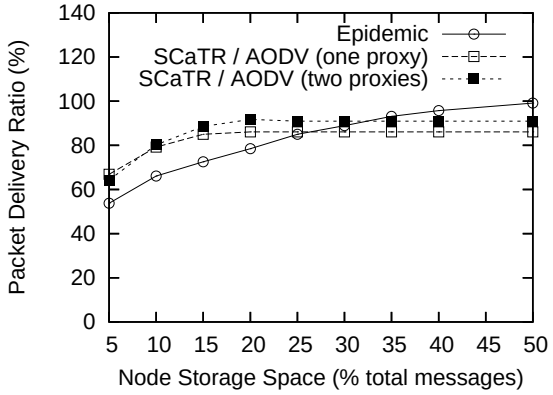

Figure 5.4: Delivery ratio for scheduled routes mobility model with limited buffer space.

5.3 Limited Buffer Results

When buffer limitations are introduced, SCaTR's performance significantly improves compared to that of Epidemic routing. All protocols employed a FIFO drop strategy for buffered messages. As shown in Figure 5.4, the duplicate messages in Epidemic routing have a detrimental effect with a small buffer. Only once the size of the buffer reaches approximately 25% of the number of messages originated in the network does Epidemic routing's delivery ratio surpass that of SCaTR. This improvement, however, is still offset by the high overhead of Epidemic routing. We can also see that with a very constrained buffer of 5% to 10%, it is beneficial to select only one proxy for each message.

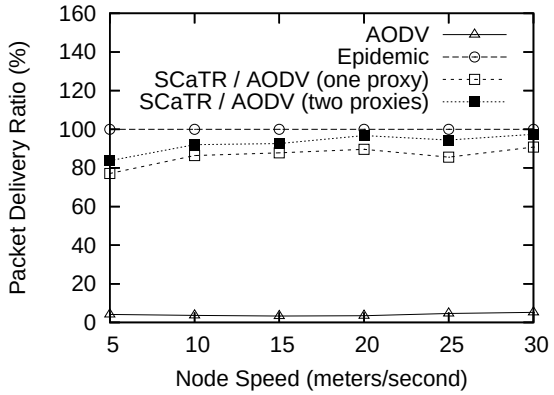

Figure 5.5: Delivery ratio for random waypoint mobility model with varied node speeds.

5.4 Random Mobility Results

SCaTR, because it utilizes mobility predictability, was not designed with random mobility in mind, but we include this scenario to illustrate that it will perform at least as well as the underlying on-demand protocol in any situation. Figure 5.5 illustrates performance when node speeds were varied to provide more or less connectivity in the network. SCaTR shows a higher delivery ratio than standard AODV despite the fact that it is unable to take advantage of historical information to predict future topologies. In addition, this experiment shows the benefit of additional message replication that can be provided with SCaTR. Epidemic routing has excellent delivery ratios at the expense of very high overhead. In power constrained environments, this would likely not be acceptable. It should be noted that the overhead of AODV is extremely high because it delivers so few messages.

5.5 Performance Over Time

Figure 6.1 shows that SCaTR's performance improves as the length of simulation increases. As the scenario time increases, delivery rates increase and signaling overhead remains constant after an initial drop. Epidemic routing quickly achieves nearly a 100% delivery rate, while SCaTR is able to achieve better than 80% after approximately 2000 seconds. Despite the fact that SCaTR takes longer than Epidemic routing to deliver messages, it does so with far less signaling. After an initial drop in signaling overhead due to low delivery rates (most packets are en route to their destination at this point), SCaTR settles at approximately 10-20 messages per delivered packet. Epidemic routing settles at approximately 38 messages per delivered packet, since each message reaches a large portion of the network.

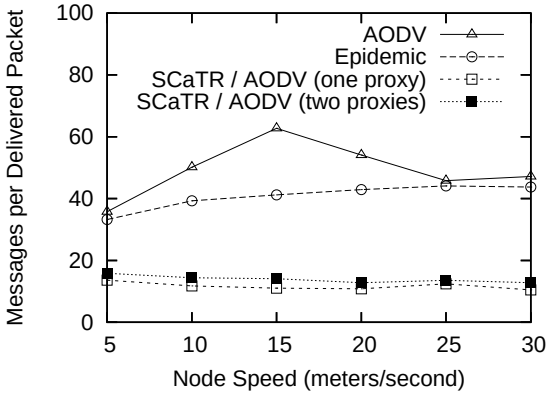

Figure 5.6: Signaling overhead for random waypoint mobility model with varied node speeds.

6. Discussion and Future Work

We have introduced the Space-Content-adaptive-Time Routing (SCaTR) framework to efficiently and effectively route data in networked environments where connectivity is intermittent. SCaTR extends on-demand routing to operate in environments where, often times, there may not be a direct route between source and destination. Thus, if the network is connected, SCaTR operates exactly as regular on-demand routing. However, if source and destination do not have a connected route, SCaTR chooses a node that is deemed closer to the destination than the source as a *proxy* for that destination. The proxy will either deliver the data to the destination directly, or choose another proxy closer to the destination than itself. In summary, the resulting protocol will do no worse than standard AODV in well-connected environments, and far better in partitioned networks. Through extensive simulations in environments with varying connectivity and mobility patterns, we showed that SCaTR can yield considerably higher delivery ratio with lower signaling overhead than traditional on-demand routing.

In this paper, we have made no assumptions regarding nodes that can broadcast schedules; our routing decisions are based solely on information gleaned from past topologies. It is conceivable, however, that a heterogeneous network may be able to provide different routing information depending on the type of device. It would be useful to combine a-priori mobility knowledge with our predicted model to improve routing decisions; when no definite schedule information is available, a node will fall back to forwarding data based on predicted mobility. In addition, nodes could broadcast trajectory or location information if it is available. We have explored routing without these assumptions, with the possibility of adding them later to improve performance.

In our approach to on-demand routing in disrupted networks, we first attempt to establish an active route with the underlying protocol, and, as an alternative, forward messages towards the destination. Depending on the scenario, it may be more efficient to first try forwarding data, and if a connected route is found to exist between the source and destination, establish the route. In this manner, we would effectively be using the data as route request messages.

These ideas we leave as future work, along with the refinement of the contact heuristic and further scenario experimentation.

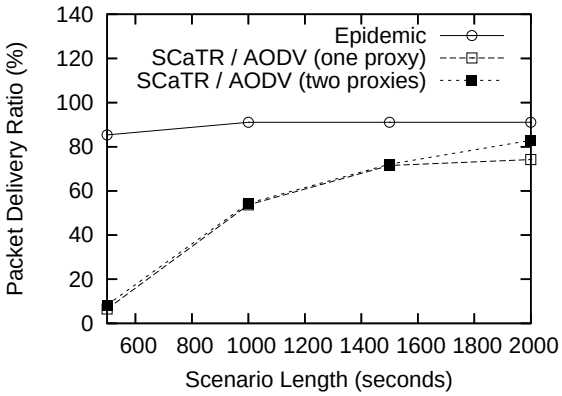

Figure 6.1: Delivery ratio for scheduled routes mobility model with varied scenario length.

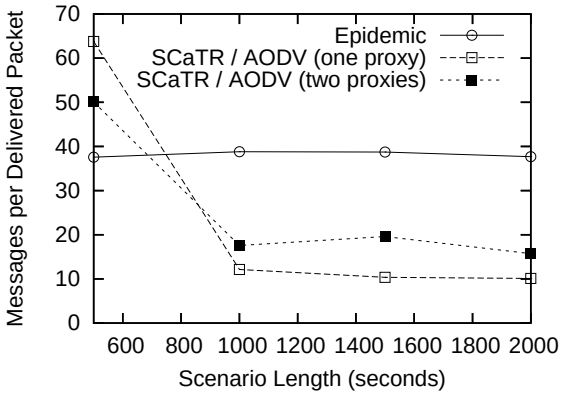

Figure 6.2: Signaling overhead for scheduled routes mobility model with varied scenario length.

References

[1] B. Burns, O. Brock, and B. N. Levine. Mv routing and capacity building in disruption tolerant networks. In *IEEE INFOCOM 2005*, 2005.

[2] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-tolerant network architecture, 2004.

[3] T. Clausen and P. Jacquet. Rfc 3626: Optimized link state routing protocol (olsr), 2003.

[4] J. Davis, A. Fagg, and B. Levine. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. In *Proceedings of the International Symposium on Wearable Computing, Zurich, Switzerland, October 2001*, pages 141–148, 2001.

[5] R. de Moraes, H. Sadjadpour, and J. Garcia-Luna-Aceves. Throughput-delay analysis of mobile ad-hoc networks with a multi-copy relaying strategy. In *Proc. IEEE SECON 2004: The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, 2004.

[6] Delay tolerant networking research group.

[7] K. Fall, W. Hong, and S. Madden. Custody transfer for reliable delivery in delay tolerant networks.

[8] Global mobile information systems simulation library.

[9] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[10] J. Leguay, T. Friedman, and V. Conan. Dtn routing in a mobility pattern space. In *Proc. SIGCOMM 2005*, 2005.

[11] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. In *The First International Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR 2004)*, 2004.

[12] S. Merugu, M. Ammar, and E. Zegura. Routing in space and time in networks with predictable mobility.

[13] R. Metcalfe and D. Boggs. Ethernet: Distributed packet switching for local computer networks. In *Communications of the ACM*, volume 19. ACM Press, 1976.

[14] W. Mitchener and A. Vahdat. Epidemic routing for partially connected ad-hoc networks.

[15] R. Ogier, F. Templin, and M. Lewis. Rfc 3684: Topology dissemination based on reverse-path forwarding (tbrpf), 2004.

[16] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM (3)*, pages 1405–1413, 1997.

[17] C. Perkins and E. Belding-Royer. Rfc 3561: Ad hoc on-demand distance vector (aodv) routing, 2003.

[18] R. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks.

[19] T. Small and Z. J. Haas. Resource and performance tradeoffs in delay-tolerant wireless networks. In *ACM SIGCOMM 2005*, 2005.

[20] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proc. SIGCOMM 2005*, 2005.

[21] W. Su, S. Lee, and M. Gerla. Mobility prediction and routing in ad hoc wireless networks, 2000.

[22] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of ACM Mobihoc 2004*, 2004.