Data-Centric Routing for Mobile Ad Hoc Networks

by

Edward Woodrow

Submitted in Partial Fulfillment

of the

Requirements for the Degree

Master of Science in Electrical Engineering

Supervised by

Assistant Professor Wendi Beth Heinzelman

Department of Electrical and Computer Engineering
The College
School of Engineering and Applied Sciences

University of Rochester
Rochester, NY

2002

Curriculum Vitae

The author was born in Syracuse, NY on July 11, 1963. He attended the Rochester Institute of Technology from 1981 to 1986, and graduated with a Bachelor of Science degree in 1986. He came to the University of Rochester in the Fall of 1999 and began graduate studies in Electrical Engineering. He pursued his research in ad hoc wireless networking protocols under the direction of Assistant Professor Wendi Heinzelman.

Acknowledgments

My graduate study has been supported through the Eastman Kodak Company's Special Opportunities Graduate Program. I would like to thank all the people who were vital in making this thesis come to fruition, especially my advisor Assistant Professor Wendi Heinzelman, who showed great patience and was always willing to help.

Abstract

In this thesis, we propose and analyze two routing protocols for mobile ad hoc networks that support efficient data and route discovery based on metadata queries. Most ad hoc routing protocols assume a node seeks to find a route to a peer node of known identity. As illustrated by the variety of proposed ad hoc service discovery architectures, more often a node is likely to seek specific data, or a specific service, with no knowledge of where it exists on the network. Our protocols, called SPIN-IT (Sensor Protocols for Information via Negotiation - Information Transfer) and SPIN-DL (- Directory Listing) use descriptive metadata to search the network for data of interest, and to establish a multi-hop route to the source. The SPIN-IT protocol is based on a distributed approach, using a controlled flooding mechanism. A centralized look-up is employed in SPIN-DL to respond to queries. Using simulations, we compare the performance of these protocols in terms of their relative overhead cost and latency. We find that each protocol performs well under different conditions, suggesting a hybrid approach might be ideal.

Contents

List of Figures

Chapter 1

Introduction

Recent technological advancements in mobile computing and wireless communications, coupled with widespread availability of devices employing such technology, has lead to heightened interest in networking protocols for these devices. Such protocols are governed by a different set of constraints when compared to their wired counterparts.

Often, these devices come together to form mobile ad hoc networks. A primary function of these networks is to share data and/or services without taking advantage of an existing infrastructure. Many routing protocols for mobile ad hoc networks have been published, but none are based on the concept of data-based route discovery.

1.1 Example Scenario

To aid in understanding mobile ad hoc network architectures, we shall first describe an example scenario. Consider a simple network made up of mobile wireless devices which may act as repositories of data. An example device might be a radio-enabled digital camera. The location of the ad hoc network might be a disaster site, say the aftermath of a train derailment. There might be twenty cameras randomly deployed on the network, moving with the users as they roam the site. A user in this network might want to locate images of interest, as defined by image metadata. For example, a user might want to find all images tagged with the descriptor "wheel." Furthermore, the network might include devices other than cameras, such as computers and printers. Using the same data exchange architecture, a user might be able to locate a print service to create a hard copy of an image, or an Internet bridge, to allow images to be uploaded. A central thesis of our work is that an efficient routing protocol can be defined to enable this type of functionality.

1.2 Network Protocols

A Mobile Ad Hoc Network (MANET) is defined by the Internet Engineering Task Force (IETF) as "an autonomous system of mobile routers (and associated hosts) connected by wireless links" [8]. This seems like a relatively broad definition, but an abundance of published research in this field in recent years has narrowed the scope. The IETF MANET group points out that MANETs have four salient characteristics, namely dynamic topologies, bandwidth-constrained variable capacity links, energy-constrained operation, and limited physical security [8].

A similar class of networked devices is referred to in the literature as sensor networks [29]. It is important to note, however, that the applications, networking principles and protocols for these systems are significantly different than in MANETs [19]. Although in this paper we will borrow routing concepts from the domain of sensor networks, we will limit our application space to that of the MANET.

The fundamental motivation behind MANET routing protocols is to lower overhead costs while supporting mobility. Traditional link-state and distance-vector-type protocols (also referred to as "shortest path" or "proactive" protocols) do not apply well to MANETs because of the high message complexity and potentially frequent topology changes. Many successfully demonstrated protocols for ad hoc routing have been based on an on-demand (or "reactive") approach, where route discovery and maintenance is done on an as needed basis [9]. These protocols are designed to handle the dynamics of the mobile ad hoc environment with minimal overhead. Two of the most commonly cited examples are Dynamic Source Routing (DSR) [4] and Ad hoc On-demand Distance Vector (AODV) [28].

Route discovery in DSR is based on the concept of source routing, where packet headers include the ordered list of nodes through which the packet will pass. The nodes themselves need not maintain route information. To make the protocol more efficient, however, nodes do cache route information they overhear. When a node has a requested route in its cache, it can complete the source route in the packet header and return it to the requesting node. AODV combines the route discovery mechanism of DSR with periodic *Hello* messages that each node sends to its neighbors. Nodes use these beacons to maintain a table of one hop route entries for each neighboring node, allowing the nodes to perform route maintenance.

Despite the large amount of effort that has gone into the development and analysis of protocols like DSR and AODV, research in this area actively continues. Even the best of the proposed on-demand MANET protocols can, in some situations, approach proactive protocols in overhead due to their costs in setting up and maintaining routes [9].

1.3 Data Discovery

As mentioned in [33], mobile ad hoc networks not only challenge routing protocols, but also network applications as well. One of the most common applications on a network is information access, or interaction with data or service providers. This interaction, often referred to as resource or service discovery, includes searching for and browsing available services and data, selecting the service or data that meets predefined criteria, and utilizing the service or directly accessing the data. The dynamic behavior of ad hoc networks makes this process more difficult than in a traditional wired network in several ways. Existing providers freely move out of a network and may or may not return. Entirely new providers may join the network at any time. Nodes may provide intermittent access due to power constraints. Furthermore, techniques like information caching are less efficient as mobile nodes break links, leave the network, and as new nodes join the network.

The concepts of data, resource, information, and service can all be abstracted to be the same thing in terms of discovery. In our context, discovery means distributing or submitting a description of what you are looking for, using metadata and matching rules, and getting back a response that directs you to your goal. The "thing" you are looking for could be "data, resource, information, or service." A discovery protocol is best designed to be agnostic towards the format and content of the metadata and matching rules. An example of such a protocol is the sensor network protocol called Sensor Protocols for Information via Negotiation (SPIN) [14]. SPIN allows an application to define and interpret its own metadata, and the authors point out that the benefit of having a succinct representation for large data messages far outweighs the costs associated with the storage, retrieval, and general management of metadata.

There may exist some differentiating characteristics between the broad metadata concepts. For example, metadata that describes a print service on a network (printer attributes) may be relatively static when compared to a network storage service's metadata (a list of files). But the performance impact of these differences is beyond the scope of this thesis. For simplicity, from this point forward in this thesis we will use the term "data" as a generic term for "data, resource, information, or service," and the term metadata as the descriptors for that data.

The primary objective of our research is to employ protocols that efficiently perform data discovery on a mobile ad hoc network. Discovery protocols exist, but these are not directly applicable to mobile ad hoc networks due to their high overhead requirements.

As described in [3][30], three of the primary service discovery protocols are the Service Location Protocol (SLP), the Simple Service Discovery Protocol (SSDP), and Jini.

In SLP, each Service Agent (SA) uses multicast messages to register services with a Directory

Agent (DA), which stores the network address and characteristics of the service. A client node wishing to employ a service first needs to find a DA, by either multicasting a service request to a specific address or by listening for periodic advertisements sent out by the DA. Client nodes “discover” a SA by connecting to and querying the DA. In an optional configuration, the service agents themselves monitor the multicast address for service requests and reply directly to the client.

The second protocol, SSDP, is essentially the same as the SLP case in which there is not a directory agent. The only significant difference is that SSDP supports multicast advertisements from the service agents. SSDP is used within the Universal Plug and Play (UPnP) architecture for service discovery.

The major issue with applying SLP and SSDP to mobile ad hoc networks is their dependence upon TCP/IP. The large amount of network traffic and complexity required to support the TCP/IP protocol stack is not an issue with wired networks, or even wireless networks of relatively static devices (like notebook computers). But for a MANET, where power consumption, computational simplicity, and mobility are major drivers, TCP/IP is simply too much of a burden.

The third referenced protocol, Jini, claims to be transport layer independent, but all practical implementations have been developed on TCP/IP. In fact, a research group within the Jini organization has been developing the concept of using a surrogate to connect devices without TCP/IP to a Jini-based system on the assumption that Jini will be running on a TCP/IP network [17].

MOCA [2] presents a Java-based framework using a distributed service discovery model. The authors claim the framework is targeted towards mobile devices with limited resources. Although similar in some ways to Jini, MOCA assumes a purely distributed architecture where no particular device becomes central to the system. MOCA is based on multicast advertisements of exported services. Nodes listen to advertisements to discover available services. If a node chooses to, it adds the service to an internal registry of available services, employing a timeout mechanism to remove stale entries. To minimize memory and network overhead, the class files associated with the remote service are only downloaded if and when the service is invoked. Unfortunately, the authors base their implementation on protocols like TCP/IP and Remote Method Invocation, which are again inherently resource intensive and do not scale well to small devices [31].

Query routing, as described in [11], is a type of informed distributed search used for resource discovery. It depends on nodes having forward knowledge of network resources in order to prune the search space. Forward knowledge is essentially a summary of a node's contents. Requests are only forwarded to nodes likely to be able to satisfy the query. The authors acknowledge that this protocol may suffer performance problems caused by network dynamics, both in terms of topology and changes to data, implying that it will not translate well to a MANET.

1.4 Motivation

Our motivation is to perform data discovery with route discovery, i.e., to find data or a service of interest along with a route to that data or service, in an energy constrained dynamic environment. No previously defined protocols meet this need. In this thesis we develop two such data-centric routing protocols for ad hoc wireless networks and model their performance. The first protocol combines the energy conscious characteristics of a traditional reactive MANET protocol (DSR) and a data-centric sensor network protocol (SPIN) to create an efficient distributed system. We call this protocol SPIN-IT, for Sensor Protocols for Information via Negotiation, Information Transfer. Concerned that the purely reactive SPIN-IT would be less efficient in communication intensive situations, we also created a protocol that borrows from proactive protocols. This second protocol, called SPIN-DL (Sensor Protocols for Information via Negotiation, Directory Listing), takes the opposite approach, creating a centralized data directory, similar to popular service discovery architectures. The results of our modeling illustrate the different conditions under which each protocol excels, suggesting that a third hybrid approach might be optimal.

The remainder of this thesis is structured as follows. We define our network models in Chapter 2. Chapter 3 describes our proposed protocols in some detail. A description of our simulation and the experimental results observed is presented in Chapter 4. Related work is described in Chapter 5, and Chapter 6 presents our conclusions.

Chapter 2

Network Models

Before describing our protocols, we need to define the MANET architecture in which they will be expected to be deployed. The network nodes will be battery powered hand-held devices with wireless transceivers and minimal computational capability. We will assume the devices have bidirectional links and will be constrained to a fixed transmit power level. The distribution of the devices could take on a range of configurations, from two devices in close proximity, to dozens of devices within a radius of hundreds of meters.

We shall assume that nodes do not have location information, unlike the models assumed in several other routing techniques [18][20][23]. This allows us to avoid the added complexity of on-board location sensors such as GPS, in devices that otherwise would not need it.

The primary form of communication between these devices is the exchange of data. The network is designed to allow nodes to discover data of interest and to transfer that data between nodes. This contrasts with other network architectures labeled as data-centric, like a sensor network running directed diffusion [16]. Directed diffusion is designed to treat a sensor network like a distributed computing system. The network is considered task-specific, that is, it supports a user's task through distributed coordination of network nodes. Our MANET network model is significantly different in that to the user, the network looks like a distributed database. The nodes on the network maintain a collection of data of potential interest to the user. Our protocols must facilitate searching the network to establish the existence of data, while also providing a multi-hop network route to the corresponding data source node.

Chapter 3

The Protocols

In this section we describe the elements of our protocols, with references to our design choices and how they tie in to our network model. We will also draw attention to the similarities our protocols share with other network protocols, while emphasizing the unique attributes of SPIN-IT and SPIN-DL.

3.1 SPIN-IT (Sensor Protocols for Information via Negotiation, Information Transfer)

SPIN-IT is an on-demand protocol that combines attributes of SPIN and DSR to efficiently locate and establish a route to distributed data of interest. When we set out to define our network, we did so from a pragmatic perspective based on the desire to use metadata to find data. We looked to previously published routing protocols, but could not find one that met our needs.

Interest in the use of metadata as a tool for discovery stems from the author's prior research in image processing and the networking of imaging devices. Images are a classic example of something that is difficult to parameterize in a meaningful way, but with the proliferation of digital images, that is exactly what users want to do. Metadata facilitates the description of images for the purpose of communication. Taken more generally, metadata can be used to describe and communicate essentially anything. The authors of the SPIN family of protocols discovered that metadata offered their sensor network protocol a tool for route negotiation before data transmission [14]. This concept overcomes the problems of implosion and overlap that occur if a sensor network uses classic flooding of sensor data. Route negotiation based on metadata eliminates redundant data transmissions, and overlap is minimized because a sink node can specify the data it is interested in. In addition, SPIN is inherently efficient because the exchange of metadata about sensor data need not be as expensive as exchanging the data itself. The relevance of this prior work to our MANET protocol was obvious early on in our research, and we named our protocol SPIN-IT with respect to the SPIN family of protocols.

An examination of the strengths and limitations of existing ad hoc protocols led to our choice of an on-demand protocol versus a proactive one. Most of our example scenarios involved random spontaneous requests for data against an ever-changing collection of data and sources. Broadcasting a request to the network seemed a logical place to begin. We quickly realized that blind flooding could result in serious problems with redundancy, contention, and collision, as was also observed in [24]. This is where we began to look to other protocols like DSR for techniques to optimize our approach. Over time, SPIN-IT evolved to be similar to DSR in many ways, with one very significant exception. Whereas DSR is node-centric, SPIN-IT is data-centric. DSR does routing based on the desire of one node to communicate with another specific node. Routing in SPIN-IT is based on a node's desire to access data, with no a priori knowledge as to the data source node's identity . The following paragraphs describe SPIN-IT in more detail.

Assume a node seeks to find a route to data of interest. The data may reside on any node within the network. The node initiates an REQ message that identifies the definition of the data of interest. The REQ message is also tagged with a unique request sequence number, generated by the requesting node. The message is broadcast to all nodes within radio range.

Each node receiving the message first checks a local database to see if it has already received this message by comparing the node identifier/sequence number tuple against those previously received. If it is redundant, the message is dropped.

If the request is not in the database, the tuple is stored and the node then performs an analysis of the request against its own set of locally available data. If the search criteria cannot be met, the message is re-broadcast. The REQ message also contains a route record. Before broadcasting, each node adds its own identifier to the record, thereby building up a record of the packet's network hops.

When a node determines that it does have the data being requested, it replies to the REQ message with a ROUTE-REPLY message. This message includes the requesting node identifier/sequence tuple, and is unicast to the requesting node using the route record. Nodes receiving ROUTE-REPLY messages either consume the message if they are the requesting node, or they forward the message according to the routing information included in the packet header.

Once a node receives a reply to its request, it has a fresh route to the node with which it

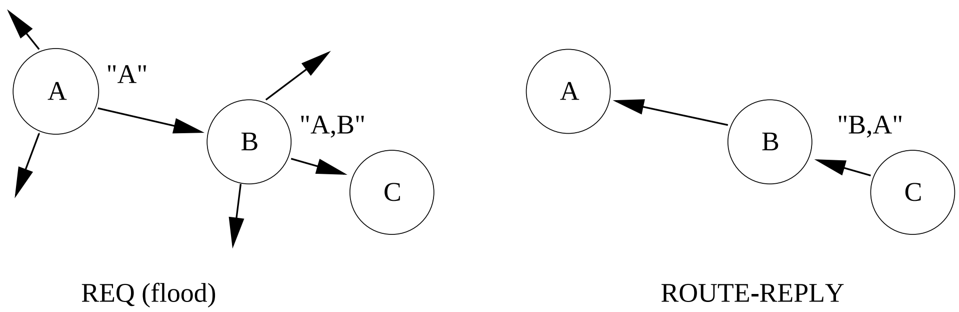


Figure 3-1: Basic SPIN-IT messaging: Node A requests data via a REQ message, and Node C replies with a ROUTE-REPLY message.

will perform further interaction. For example, the node might then send a *data request* to the source node to initiate the transfer of a large data set. It is possible that a requesting node will receive multiple ROUTE-REPLY messages, if multiple source nodes can satisfy the request. One way to handle this situation is to use a receive window. Upon receipt of the first ROUTE-REPLY message, the requesting node starts a timer. When the timer expires, the node selects the ROUTE-REPLY message received within the window that best meets its need, be it based on shortest route or some other measure. The size of the window can be adjusted based on network dynamics. An even more sophisticated technique related to this issue was described in [27] as diversity injection. In this approach, all received ROUTE-REPLY messages are cached temporarily, and the extra paths are "injected" into the follow-on data requests in order to return to the source node along multiple paths. The authors reported better performance in the presence of route failures, and even an improvement in the finding of shortest routes. Of course, this technique can add significant overhead. To simplify our analysis, in our modeling we assume a one-to-one relationship between each request and response, negating the need to implement receive windows or diversity injection.

The basics of SPIN-IT are illustrated in Figure 3-1, where Node A is searching for data, and Node C has the data Node A wants. Node A broadcasts a REQ message that eventually reaches Node C. Node C is then able to unicast a ROUTE-REPLY message using the routing information built up within the REQ message. Node A now learns that Node C has the data it seeks, Node A has a route to Node C, and Node A can initiate the appropriate action, be it a data request mechanism or the exercise of a service.

Route caching offers an interesting opportunity to enhance the performance of SPIN-IT. In DSR, cached routes are based on source-sink node identification pairs. Route caching in SPIN-IT is potentially much more complex in that in order to be meaningful, cached data would need to reflect the entire metadata structure. In some applications, the number of cached entries might need to be limited in order to minimize memory usage. Matching against the cache could be computationally non-trivial, especially if complex matching criteria are specified.

With route caching implemented, when a node receives a new REQ message, not only does the node check to see if the message's search criteria can be met locally, it also checks to see if any routes in its cache would match the search criteria. If so, the node does not re-broadcast the REQ message. The node updates the route record by combining the current route record with the cached route, creating a complete route from source to sink. The route is then returned to the requesting node as a ROUTE-REPLY message.

Route caching would clearly make sense for simplistic applications. As is always the case with route caching, it is important that one minimizes the occurrence of stale cached routes. In [22] the authors point out that bad caching is worse than no caching at all. They conclude that when looked at independently, "adaptive route expiry shows the most improvement in performance" of all proposed route caching optimizations. In our model, route expiry is established based on the dynamics of the simulation, but is static for the duration of the simulation. At the beginning of each run, we perform a simple calculation based on the maximum node velocity which establishes an expiry that provides a 90% confidence level that as the nodes along a route move, the nodes will stay within range of each other, and the route will remain valid.

As described in [24], a simple way to control SPIN-IT flooding overhead in a dense network would be to add a counter. When a node hears a REQ message, it sets a counter $c = 1$. Then it waits a short period. If no duplicate REQ tuples are heard during the probation period, the message is rebroadcast. If a second REQ message of the same REQ tuple is heard during the probation period, the message is not rebroadcast. This works on the assumption that if the network is that dense, and multiple nodes in range have already broadcast, a node that hears two REQ messages within a short period of time will probably not add value by re-broadcasting the message. The simulations reported in this thesis did not include this feature.

Even though we assume our nodes do not have location information, we should recognize that with the addition of location information such as GPS, techniques such as LAR [18] could also be used to limit the scope of the flooding and lower routing traffic overhead.

3.2 SPIN-DL (Sensor Protocols for Information via Negotiation, Directory Listing)

After spending a fair amount of time analyzing the performance and behavior of SPIN-IT, we began to recognize that it suffered a performance degradation when the rate of REQ messages grew. This was not a surprising artifact, given that SPIN-IT is based on flooding. After considering some techniques for optimizing SPIN-IT under these conditions, we concluded that we should explore a different approach as well.

The concept of a centralized look-up service is quite common in ad hoc discovery protocols, as mentioned in Chapter 1. We recognized that a centralized architecture operates more efficiently as the number of requests increases, as compared to the overhead of maintaining a look-up directory. Of course, in a highly dynamic environment, the cost of maintaining an accurate directory increases.

We decided a centralized approach held enough promise to create a second protocol, called SPIN-DL. SPIN-DL is a hybrid protocol, with some proactive and some reactive attributes. SPIN-DL uses a centralized directory as a look-up service, providing both data source and route information.

Once again in this model, we assume a node seeks to find routes to data of interest. Just like in SPIN-IT, data may reside on any node within the network, and the requesting node initiates an REQ message that identifies the definition of the data of interest. The REQ message is also tagged with a unique request sequence number, generated by the requesting node. Rather than being broadcast, however, the message is unicast to a special node on the network called the *central node*.

The central node may be a unique node within the network, such as a centrally located node connected to a power supply, or it could be a service offered by a node that is otherwise no different than any other node. The role of the central node could even be transferred from one node to another in scenarios where equitable energy consumption is an issue, as described in [13]. The central node's primary function is to maintain a directory that maps metadata to the nodes that possess the data.

Before describing the central node's processing of the REQ message, it will be helpful to understand two other elements of the SPIN-DL protocol. First, on a periodic basis known as the *beacon rate*, the central node broadcasts BEACON messages. These are short packets that contain nothing but a sequence number and a route record built up as the message floods the network. Flooding is controlled using the same logic as the REQ message in SPIN-IT. When a node first receives a BEACON, it records the route record as its *central node route*. The other element of interest is the UPDATE message. Nodes transmit a copy of their metadata cache to the central node using the central node route either asynchronously when new data is acquired, periodically at an *update rate*, or both. Upon receipt of UPDATE messages, the central node updates its directory listing. In addition to the metadata, the directory is also updated with the route record contained within the UPDATE message. In this way the central directory maintains a fresh route to nodes within the network. If periodic updates are used, the central node also has the option of timing out nodes that do not provide updates and removing them from the directory.

Upon receipt of a REQ message, the central node searches its directory to determine if any nodes within the network can satisfy the request. If a node does satisfy the request, a ROUTE-REPLY message is returned to the requesting node, reversing the route record built up within the REQ message. The ROUTE-REPLY message identifies the data source node and provides source routing information. If the search fails, a null ROUTE-REPLY message is returned.

In Figure 3-2, Node C is the central node, and the top drawing illustrates the flooded BEACON message. The bottom left drawing illustrates that UPDATE messages and REQ messages are both unicast from all non-central nodes to the central node using the route established via the BEACON message. In the bottom right drawing, Node C replies to Node A's REQ message using a unicast ROUTE-REPLY message, based on the routing information from the REQ message.

As was the case in SPIN-IT, a requesting node in SPIN-DL might receive several ROUTE-REPLY messages from the central node should multiple matches exist within the directory. Here again, a receive window technique could be applied. Alternately, the central node could package several potential source routes within a single ROUTE-REPLY message. Consistent with our SPIN-IT modeling simplification, we avoid this issue in our model by ensuring that our directory includes exactly one match for each request.

Realizing the flooding nature of the BEACON message introduces a significant amount of overhead, it would be important to define an algorithm to optimize the period of the BEACON messages. Such an algorithm would be based on network dynamics, and fortunately in SPIN-DL the central node can draw some conclusions about network dynamics based on the incoming UPDATE messages. For the purposes of our simulations we choose a fixed BEACON rate, leaving the implementation of this algorithm to future efforts.

If periodic UPDATE messages are to be employed, their period is another opportunity for optimization. Because UPDATE messages are unicast to the central node, they do not suffer

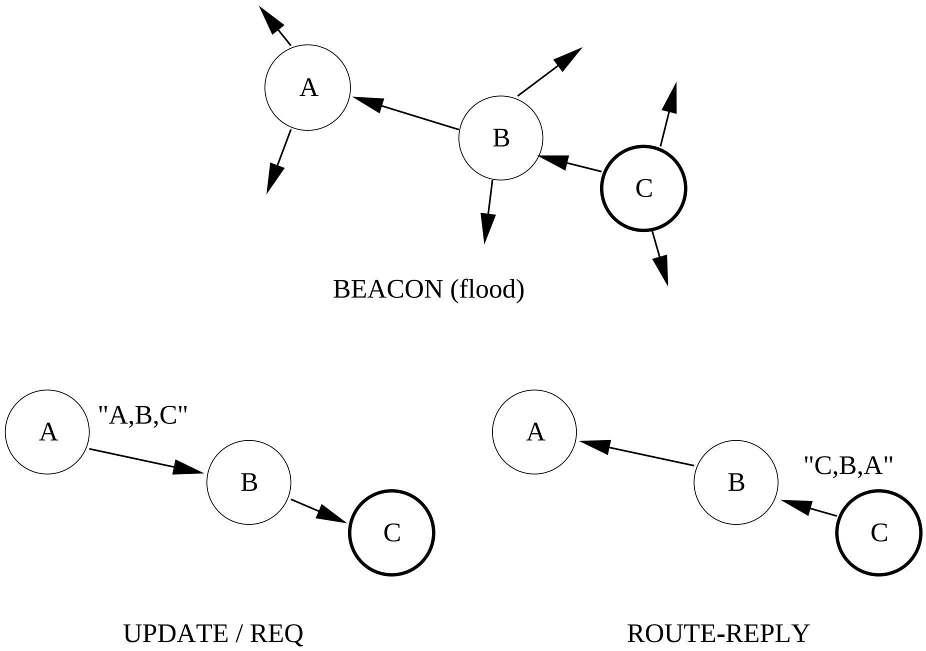


Figure 3-2: Basic SPIN-DL messaging: Node C is the central node, and it periodically floods the network with BEACON messages. Node A sends both UPDATE and REQ messages to Node C, and Node C replies to requests with a ROUTE-REPLY message.

flooding inefficiencies. However, in some applications the UPDATE message might be quite large. An algorithm could be developed where a node adjusts its UPDATE rate based on its mobility pattern. In [1] the authors defined an algorithm called Adapting to Route-demand and Mobility (ARM) that adjusts periodic routing messages based on the mobility pattern of its neighboring nodes. ARM might apply well to SPIN-DL's UPDATE message. Once again, we left the implementation of an adjustment algorithm for the UPDATE period to future efforts. Our simulation runs hold this rate constant.

In SPIN-IT, a node has a route to the data source upon receipt of the ROUTE-REPLY message. In SPIN-DL, the ROUTE-REPLY message identifies the source node, meaning the requesting node could use an algorithm like DSR to find a route to the source. But this would add a significant layer of overhead that would make SPIN-DL less efficient. We recognize that the central node has a significant cache of network knowledge that could be leveraged to make route discovery more efficient.

The ROUTE-REPLY message in SPIN-DL includes two routes, one from the requesting node to the central node, and one from the central node to the source node. The combination of these routes should represent a valid route from the requester to the source. Although it

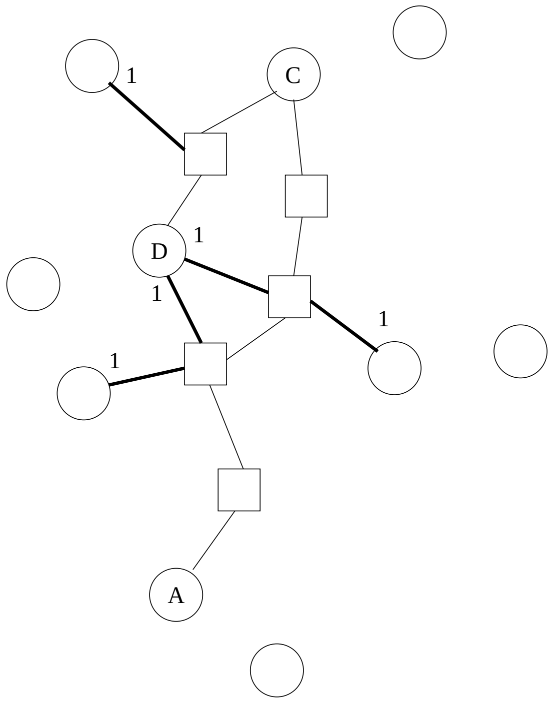


Figure 3-3: Illustration of query localization. In this case, Node A is the requesting node, Node C is the central node, and Node D is the data source node. The square nodes represent the two original routes to the central node, and the bold lines represent the extent of the query beyond the original routes. We see here that with $k = 1$, a much shorter route from Node A to Node D is established without flooding the entire network.

is not likely to be optimal, this route could be used without generating any additional route discovery overhead. In Figure 3-3, this route is represented by the square nodes from Node A, the requesting node, to Node C, the central node, then on to Node D, the source node.

In [7] the authors introduce the concept of spatial locality. This notion proposes that "a mobile node cannot move too far too soon," and therefore a shorter route is likely to exist within the *neighborhood* of this two-part cached route. This neighborhood is referred to in [7][18] as the *request zone*. As illustrated by the bold lines in Figure 3-3, the request zone is defined to include all nodes within at most k hops of the cached route, in this case with $k = 1$. A DSR-like route discovery protocol could then be limited to the request zone, minimizing the overhead impact. Failure to find a valid route would indicate the need to grow the request zone by increasing k.

The parameter k must be optimized dependent upon the topology and dynamics of the network, with highly mobile networks requiring more hops to identify shorter routes. The central node could also apply its knowledge of the network to dynamically adjust k. For example, the central node could compare the two routes, looking for common nodes. Common nodes would indicate spatial proximity, meaning that a higher value of k might find the shortest path with less overhead.

In addition, the requesting node should perform a loop analysis on the two routes returned from the central node. If the requesting node and the source node are in spatial proximity, there is a strong chance that their routes to the central node have points in common, leading to another opportunity to minimize the route. This analysis is also necessary to avoid routing loops.

3.3 Discussion

We have shown that the SPIN-IT protocol is designed to flood an ad hoc network with requests for a specified set of data or service. Given that minimum energy consumption is a primary objective, and realizing that even controlled flooding can be costly if done too often, we designed SPIN-DL as an alternate protocol which only uses a minimal amount of flooding. Instead, SPIN-DL employs a central directory that guides a requesting node to its desired source.

In the following section we report on experimental results of simulations of SPIN-IT and SPIN-DL exercised under a variety of network conditions.

Chapter 4

Performance Evaluation

4.1 Simulation Environment

In order to study the behavior of the protocols, and to aid in their development, we created simulations of SPIN-IT and SPIN-DL using the **ns-2** software package [25]. **ns-2** provides support for the realistic simulation of physical and link layers, the 802.11 wireless MAC layer, and node mobility. This framework provided stable, predictable lower network layers, allowing the research to focus on the upper level protocols. A number of recently published simulations were also based on **ns-2** [5][14][22].

Considering our example scenario, the personal imaging devices of interest might be devices one would want to use to connect to the Internet using an 802.11 WLAN access point. Given that, it would be cumbersome to change radios to perform the two types of communication. It is logical then to build our protocols on top of 802.11, thinking of our protocols as another radio "mode" in addition to the existing WLAN modes. In addition, the 802.11 PHY and MAC offer a convenient hardware platform for implementing these protocols in the lab in future research. The 802.11 MAC within **ns-2** uses the Distributed Coordination Function (DCF) as its basic access method, which includes carrier sense, RTS/CTS, and backoff mechanisms [15]. Although these are not necessarily appropriate for a MANET due to their inherent energy inefficiencies, the model offered a reliable simulation. As these issues will apply equally to our two protocols, we felt the inefficiencies could be ignored as they relate to our analysis.

Mobility in our simulation is patterned after the "random waypoint" model [5]. The starting position of a node is random. Nodes move towards a randomly selected destination within the defined topography, and stop at the destination for a pause time, before heading for a new

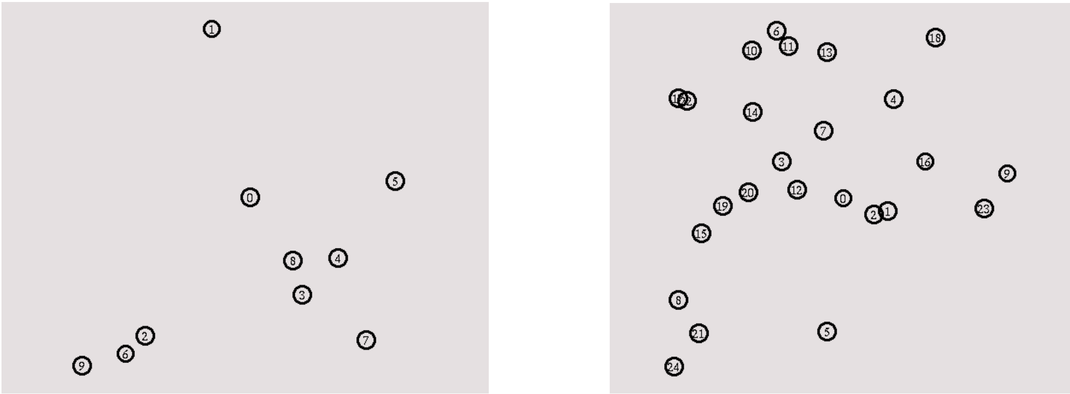


Figure 4-1: Example network topologies, 10 node and 25 node.

destination. The pattern repeats for the course of the simulation. Node velocity is randomly chosen between zero and a fixed maximum value, and a new value is selected after each pause.

Our network scenarios are characterized by varying the number of nodes, the node start positions, the maximum node velocity, and the pause time. We used node counts of 10, 25, 50, and 100, placed within a square region of 1000x1000 meters. Figure 4-1 shows example network topologies for our 10 node and 25 node scenarios. We retained the **ns-2** default 802.11 transmit range of 250 m, and we allowed the metadata size within REQ and UPDATE messages to vary randomly between 8 and 512 bytes. We used maximum node velocities of 0 m/s, 0.1 m/s, and 1 m/s, modeling people seated in a room (stationary), "low mobility" and walking people, similar to those used in [10]. We also ran some simulations using a maximum node velocity of 10 m/s to expose the impact of higher mobility. Our simulations are 900 seconds long, and our pause times were chosen to be comparable to a prior study [5]: 0, 30, 60, 120, 300, 600, and 900 seconds, where 0 results in the maximum node movement and 900 results in the least amount of node movement. This creates 88 different network situations, and we exercised each one 10 times to create a total of 880 simulation scenarios.

It is important to note that many sensor networks model network traffic using constant bit rate (CBR) traffic sources [21]. Our model, in contrast, is tied to the rates of the different messages. In SPIN-IT, traffic is specified by the request rate, that is, the rate at which nodes send REQ messages. We model this by setting a fixed rate that is then randomly distributed across all nodes in the network, meaning any given node's request rate is random within bounds set by the request rate and the number of nodes. In SPIN-DL, the traffic is a little more complex.

The central node sends out beacons at the beacon rate. The REQ messages are distributed as in SPIN-IT, except that the central node is not included. UPDATE messages are modeled similarly, with any given node's update rate being random, summed across all nodes (except the central node) to a fixed update rate.

We limit the messaging rates in such a way as to avoid operating the network in a state of congestion, which would result in a substantial number of dropped packets. Collisions are minimized during flooding by introducing small random delays when nodes forward packets.

Our simulations and measurements are limited to the point where a requesting node receives a ROUTE-REPLY message. We do not include the data request mechanism for obtaining the actual data, since it is assumed that it would be the same in both cases. Nor do we implement the spatial locality concept in SPIN-DL. We assume that the requesting node has a valid route within the ROUTE-REPLY message and leave the exercise of optimizing that route and analyzing the impact of this optimization to future work.

4.2 Performance Metrics

In order to evaluate the performance of our protocols, we look to the characteristics of concern in our architecture, namely energy utilization within a dynamic topology. The primary measure of performance will be the overhead cost of data discovery. For each protocol, the total number of bytes of routing traffic transmitted during the simulation is considered to be the overhead cost. This cost will be studied in the context of the size and mobile behavior of the network.

In addition, we will measure the latency of the protocols by measuring the elapsed time between when a node sends out a REQ message and when the ROUTE-REPLY is received. Latency may be an important requirement for some applications.

Other papers that study routing protocol performance track a metric referred to as *packet delivery ratio* in order to characterize the completeness and correctness of a protocol [5][9]. This is defined as the ratio of the number of packets originating at a source node to the number of packets received at the desired sink node. As our protocols are not node centric, we will look at a slightly different metric. We will measure the *request-response ratio*, which is defined as the number of request messages that are satisfied with a corresponding reply. We can also evaluate the number of reply messages with respect to the total amount of overhead, resulting in a measure of overhead per ROUTE-REPLY.

4.3 Verification of Simulation

To ensure the accuracy of our simulations, we first performed a series of experiments on a static, fully connected network. An analytical model of the overhead for the two protocols was developed, on the assumption that our simulation is accurate if our simplified experimental results mirrored the analytical result.

For both the SPIN-IT and SPIN-DL protocols, the total amount of overhead during the simulation time was recorded. Overhead is defined as the total number of bytes contained within routing packets. For the SPIN-IT model, overhead included the REQ packets flooded through the network in search of a data source and the ROUTE-REPLY messages from the data source to the requesting node. With route caching enabled, overhead also includes ROUTE-REPLY messages sent from nodes with matching cached routes. For the SPIN-DL model, overhead included the UPDATE messages to the central node and the REQ and ROUTE-REPLY message pairs between the requesting nodes and the central node. To simplify the analytical solution, the rates of REQ and UPDATE messages were held constant (no aperiodic messaging). In addition, since our verification model did not include mobility, we did not include the periodic BEACON messages. Instead, a single BEACON was broadcast at the start of the simulation to establish routes to the central node.

The simulation was run varying the ratio of the rate of UPDATES at each node, denoted by λ_1, and the rate of REQs generated at each node, denoted by λ_2. Intuitively, we would expect that for $\frac{\lambda_1}{\lambda_2} << 1$, a centralized approach would be more efficient in terms of overhead since there is less overhead associated with each REQ and there are more REQs than UPDATEs. However, for $\frac{\lambda_1}{\lambda_2} >> 1$, we would expect that a distributed approach is more efficient since each UPDATE message incurs overhead and there are more UPDATEs than REQs.

To determine analytically where this crossover point occurs, we assume that there are N nodes in the network, the path from a node to the central node is an average of L hops, and the average path length from one node to any other node in the network is k hops. Using these assumptions, without route caching, in SPIN-IT there are $\lambda_2 * (N-1)$ REQs/sec and $\lambda_2 * k$ ROUTE-REPLYs/sec sent throughout the network (assuming only one node has the requested data). Thus the total number of overhead messages transmitted per second in SPIN-IT is: $\lambda_2 * (N + k - 1)$. For SPIN-DL, there are $\lambda_1 * L$ UPDATEs/sec, $\lambda_2 * L$ REQs/sec, and $\lambda_2 * L$ ROUTE-REPLYs/sec sent throughout the network Thus the total number of overhead messages transmitted per second in SPIN-DL is: $\lambda_1 * L + \lambda_2 * 2L$. Comparing these two figures, we find

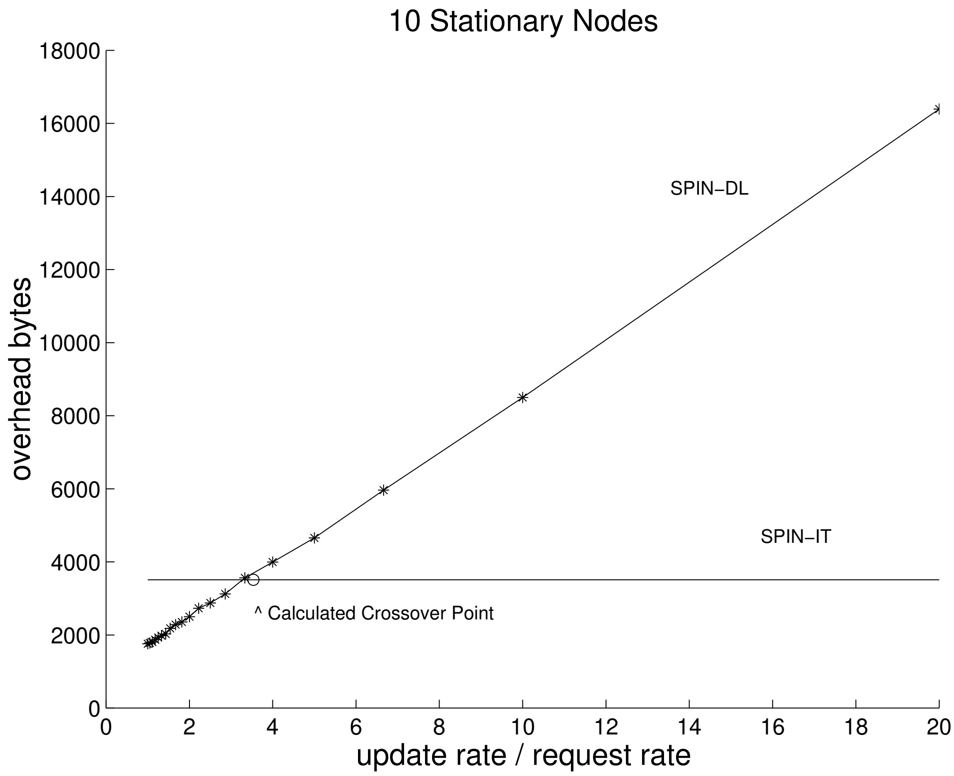


Figure 4-2: Simplified experimental results for 10 node network, showing point where SPIN-IT gains advantage over SPIN-DL, compared with expected results. The request rate is a constant 1 msg/sec, resulting in a constant overhead value for SPIN-IT.

that SPIN-IT has less overhead than SPIN-DL if:

$$\frac{\lambda_1}{\lambda_2} > \frac{N+k-1}{L} - 2 \tag{4.1}$$

To determine if these approximations hold, we simulated both SPIN-IT and SPIN-DL and measured the amount of overhead generated by each protocol as we varied the ratio $\frac{\lambda_1}{\lambda_2}$. There are no UPDATE messages in SPIN-IT, so the easiest way to vary $\frac{\lambda_1}{\lambda_2}$ was to hold λ_2 constant at 1 message/sec and vary λ_1.

We began with a 10-node network where $k = 1.4$, and $L = 1.9$, so we expect the crossover point to be $\frac{\lambda_1}{\lambda_2} = 3.5$. Figure 4-2 shows simulation results using this network, where we varied the update rate λ_1 from 1 to 20 messages per second. We see that the crossover point was very close to 3.5. Therefore, when the number of updates generated throughout the network is greater than 3.5 times the number of REQs generated throughout the network, SPIN-IT produces less overhead per REQ message, whereas when the number of updates is less than 3.5 times the

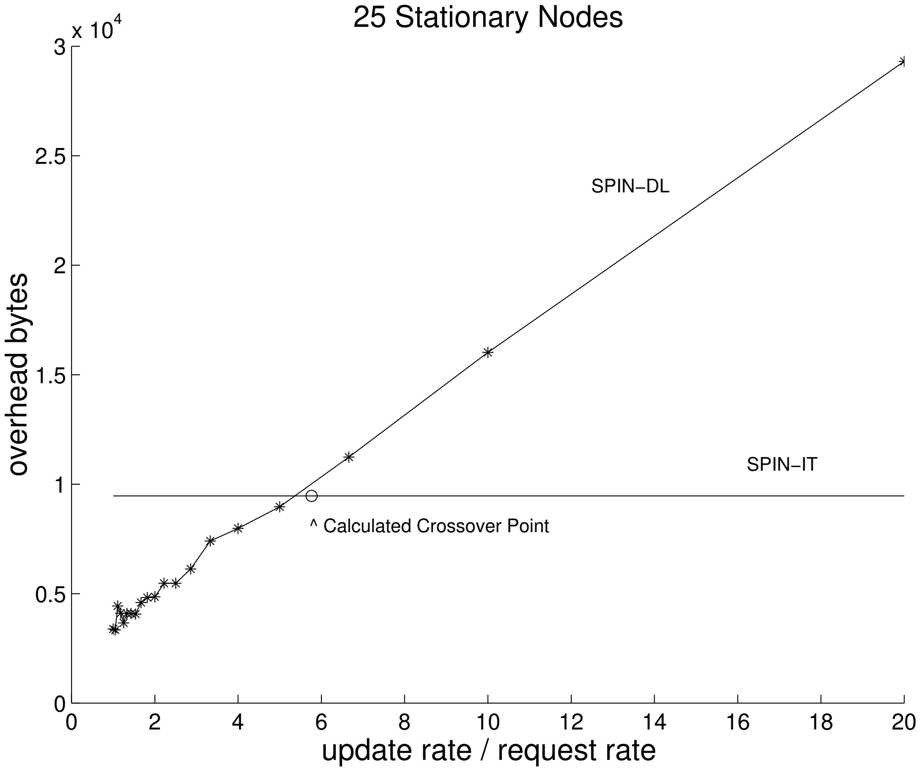


Figure 4-3: Simplified experimental results for 25 node network, showing point where SPIN-IT gains advantage over SPIN-DL, compared with expected results. The request rate is a constant 1 msg/sec.

number of REQs, SPIN-DL is more efficient.

We also measured the amount of time it takes between a node sending out an REQ and receiving the associated ROUTE-REPLY for SPIN-IT and SPIN-DL and found that for $\frac{\lambda_1}{\lambda_2} = 1$ the delay is, on average, 0.0335 ms for SPIN-IT and 0.0152 ms for SPIN-DL. Other values of $\frac{\lambda_1}{\lambda_2}$ showed comparable results. As expected, SPIN-DL exhibits better latency performance than SPIN-IT in this simplified example.

We also wanted to determine what effect the network topology has on the results described above, so we ran the same experiment for a 25 node network with $k = 3.5$ and $L = 3.2$. From Eq. 4.1 we expect the crossover point to be $\frac{\lambda_1}{\lambda_2} = 5.7$. The data from the simulations, shown in Figure 4-3, confirms that SPIN-IT becomes more efficient than SPIN-DL only when $\frac{\lambda_1}{\lambda_2}$ is greater than about 5.7.

These results lead to some early conclusions. In densely populated networks where there are many more updates than requests, as may occur in highly dynamic networks or networks where new data is continuously generated, SPIN-IT requires less overhead, allowing more bandwidth to be spent transmitting data and requiring less energy waste than SPIN-DL. However, if the network is fairly static, with few new data being generated and lots of requests for data, our preliminary experimental results showed that a centralized approach is more efficient.

4.4 Evaluating the Protocols

We start off our evaluations with some basic comparisons of protocol overhead performance. Since route caching is seen as a dynamic optimization technique, for most runs we keep the comparisons simple and use SPIN-IT with route caching turned off.

In many of our evaluations, we perform a series of experiments where we vary the scenario pause time. Varying the pause time from 0 to 900 varies the mobility characteristics of the scenario. All nodes begin the the experiment with a random position, a random destination, and a random speed greater than 0 and less than the maximum. With a pause time of 0, when nodes reach their destination, they immediately choose a new destination and head off in that direction at a new randomly chosen speed. With a pause time greater than 0, when a node reaches a destination, it waits there for the duration of pause time, then it chooses a new destination and speed. With a pause time of 900 seconds, since the experiment duration is 900 seconds, all nodes cease movement after reaching their first destination.

Random node positions and speeds have a significant influence on network dynamics within each pause time data set, even averaging over 10 different scenarios. This is especially true in the 10 node cases, as can be inferred from the distribution of nodes in Figure 4-1. In an attempt to minimize this effect, our scenario generator always placed the node used as the central node in SPIN-DL in the center of the topology with a speed of 0 m/s, emulating an ideal central node selection mechanism. Even so, comparisons between different pause times can only look for significant trends, as opposed to explanations for minor fluctuations. Comparisons between the two protocols at a given pause time are significant, since the scenarios used in these cases are exactly the same.

In our first set of results, shown in Figure 4-4, we illustrate in (a) the overhead cost of SPIN-IT and SPIN-DL for a 10 node network, where the maximum node speed is 1 m/s and the ratio of UPDATE messages to REQ messages is 10 (i.e., 10 UPDATEs for every REQ). We see immediately that SPIN-IT demonstrates significantly better overhead performance as expected, since $\frac{\lambda_1}{\lambda_2} >> 1$. In (b) we increase the number of nodes to 25 and we see that both protocols suffer an increase in overhead. It is interesting to note that in absolute cost, the margin remains

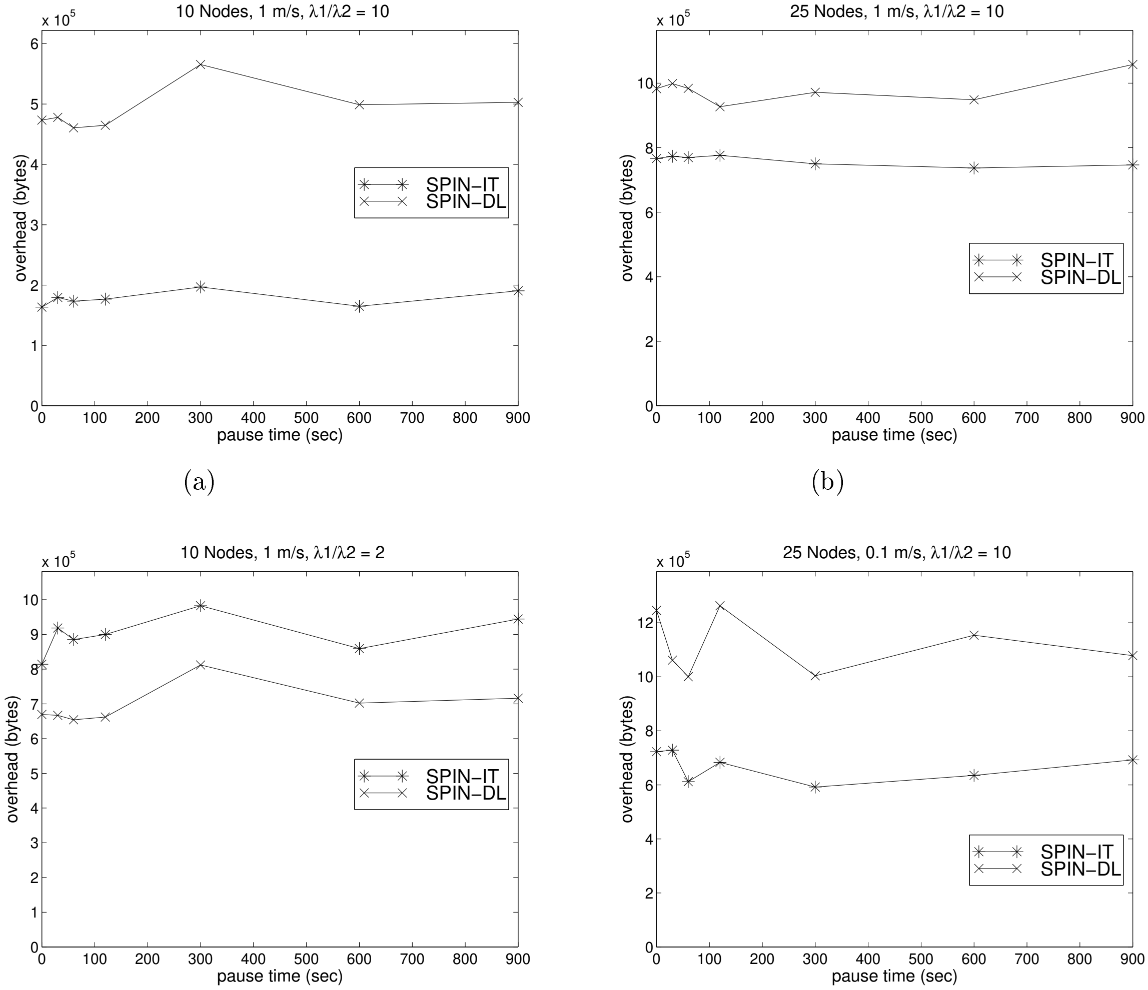
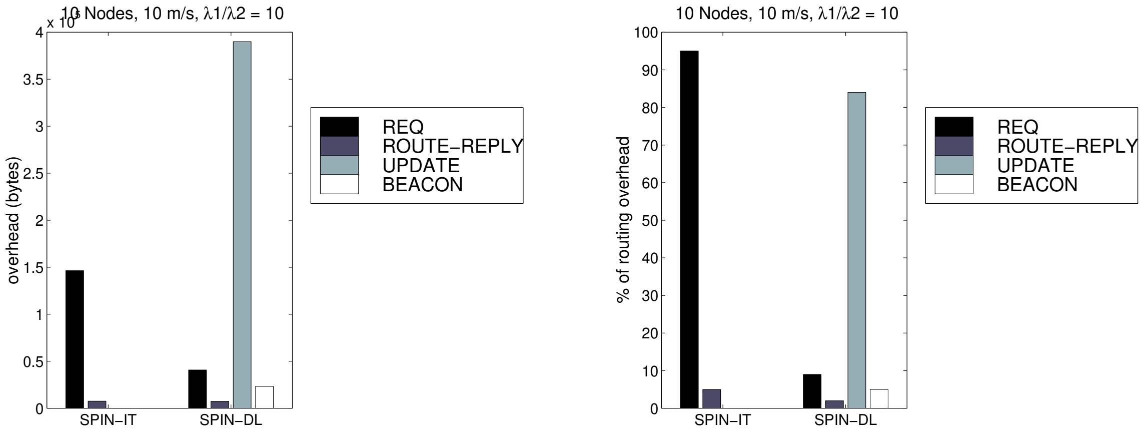


Figure 4-4: Total routing overhead for SPIN-IT and SPIN-DL as a function of pause time. In (a), we have 10 nodes with a maximum node velocity of 1 m/s. In (b) we illustrate the same conditions with 25 nodes. In (c) we compare with an increased rate of REQ messages, and in (d) we show 25 nodes with a maximum node velocity of 0.1 m/s. A pause time of 0 represents constantly moving nodes and a pause time of 900 implies that all nodes stop moving after having reached their first destination.



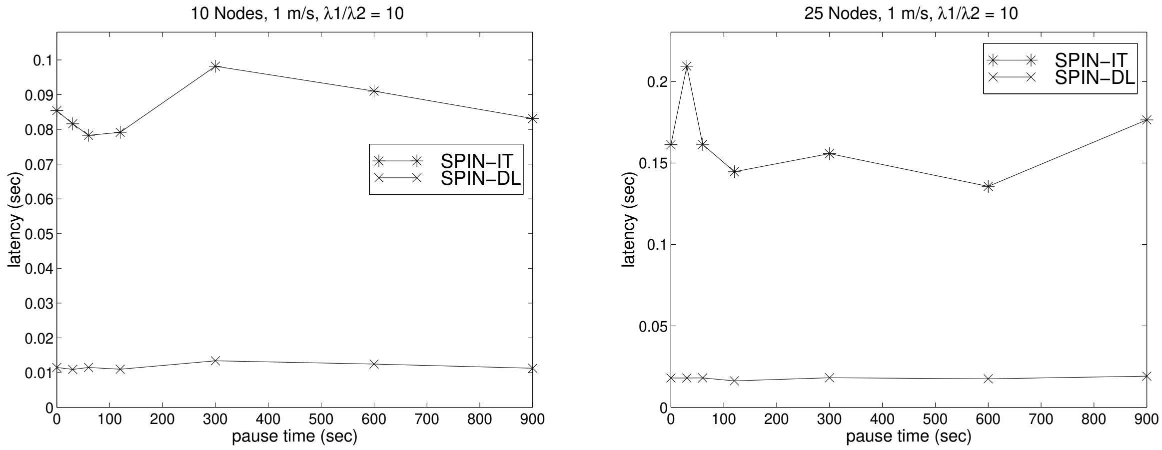
(a) (b)

Figure 4-5: Comparison of routing message sizes for SPIN-IT and SPIN-DL, with (a) being the total byte counts of the different message types, and (b) being the relative percentages.

somewhat constant between the two sets of scenarios. We go back to a 10 node network in plot (c), but now we decrease the rate of UPDATE messages by a factor of 5. In this case, SPIN-DL demonstrates superior performance, with SPIN-IT incurring higher overhead costs due to the magnitude of flooding in the network. And finally, in plot (d) we show a 25 node network with a decrease in maximum node speed to 0.1 m/s. The variation in the data increases over plot (b) because the lack of node movement increases the relevance of initial node position. Overall, the relative performance between the protocols remains about the same.

One needs to keep in mind the overhead costs in terms of the different messages that make up the two protocols. In Figure 4-5, we show in (a) the total byte counts for each protocol broken out by message type for a typical scenario. In (b) we show the percentages of each message type in terms of size. We note that SPIN-IT has a significantly larger number of bytes associated with REQ messages than in SPIN-DL, as would be expected in a flooding protocol. In SPIN-DL, a large portion of the overhead is attributed to the UPDATE messages. In this case, this results from the update rate being 1 message per second, 10 times greater than the request rate. The UPDATE message size is also relatively large, given that it includes a representation of the node's data contents. Both of these factors could be optimized to save overhead cost. For example, the UPDATE rate could be tailored to the dynamics of the data, or only asynchronous UPDATE messages could be used. Also, an algorithm could be established for only sending changed data in cases where large portions of the data set remain unchanged.

In Figure 4-6 we study the latency performance of the two protocols for the case of a 10



(a) (b)

Figure 4-6: Latency for SPIN-IT and SPIN-DL as a function of pause time, for 10 Nodes (a) and 25 nodes (b). A pause time of 0 represents constantly moving nodes.

node network in (a) and a 25 node network in (b). We observe that in both cases, SPIN-DL shows significantly better performance (lower latency) than SPIN-IT. This is to be expected. In SPIN-DL, nodes have a cached route to the central node and simply send a unicast REQ to it, and the central node replies via unicast along the same route. In SPIN-IT, flooding the network takes time, especially if the source node lies many hops away. These results may also be skewed by our decision to always centrally locate the central node. On average, the distance to the central node in SPIN-DL is less than the distance between source and sink in SPIN-IT. This could be justified if one implemented a central node selection algorithm that sought the geographic center of the network as a selection criterion.

We saw in plots (b) and (d) of Figure 4-4 that node speed did not show a significant impact on the relative performance of the two protocols. In Figure 4-7 we add two additional node speeds, 0 m/s and 10 m/s and compare performance in both 10 and 25 node networks. We see that although the data exhibit variation across the node speeds, the relative performance of the protocols remains the same, with SPIN-IT demonstrating superior energy utilization in all cases, as should be expected because $\frac{\lambda_1}{\lambda_2} = 10$.

Another measure of our protocol performance is the ability of the protocol to deliver a ROUTE-REPLY message for each REQ message with varying network topologies and dynamics. We plot the percentage of ROUTE-REPLY messages received relative to the number of REQ messages sent out in Figure 4-8. We see that SPIN-DL consistently delivers slightly more replies than SPIN-IT. Again, this is to be expected due to flooding, collisions, etc.. Our simulations

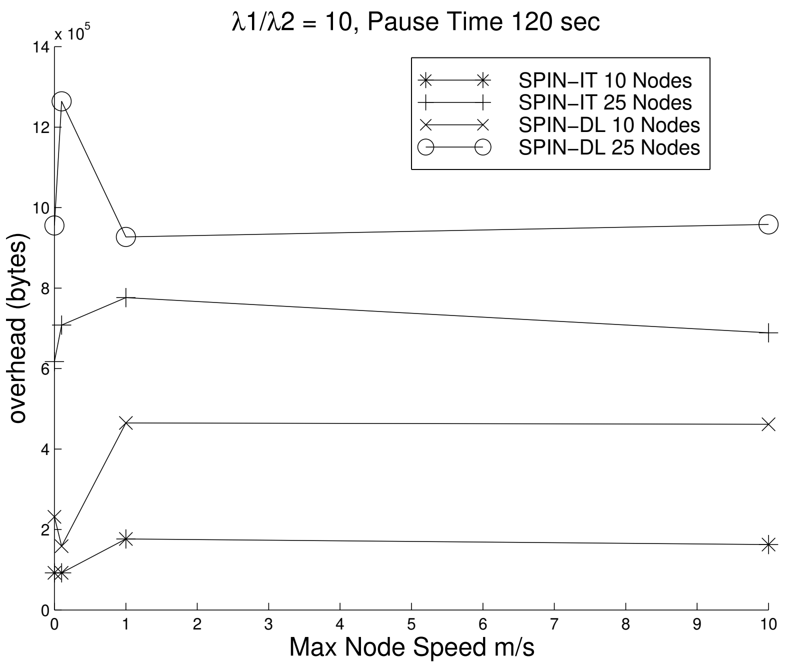


Figure 4-7: Routing overhead as a function of maximum node velocity.

are not free from dropped packets, despite our efforts to avoid network congestion. As will happen with random message timing, on occasion messages will be dropped. As a result, not all REQ messages translate to the successful receipt of a ROUTE-REPLY message, even when there exists a valid route between them.

As previously described, another variable in the performance of SPIN-IT and SPIN-DL was the different level of connectivity between the nodes. The ability to find the correct source node in SPIN-IT is tightly coupled to all node locations. In SPIN-DL, the ability for a node to communicate with the central node is dependent upon the existence of a valid central node route. These two dependencies are significantly different. Having tracked the number of ROUTE-REPLY messages received, we can plot the overhead cost per ROUTE-REPLY, thinking this a better measure of performance. The result is shown in Figure 4-9, for a network with 25 nodes, a maximum node speed of 1 m/s, and an update rate of 10. Plot (a) shows the total routing overhead, and plot (b) shows that the superiority of SPIN-IT in this situation is minimized if viewed in terms of cost per successful transaction.

An additional measure of performance for a routing protocol relates to the length of the route identified by the protocol for communication between two nodes. An ideal protocol would always

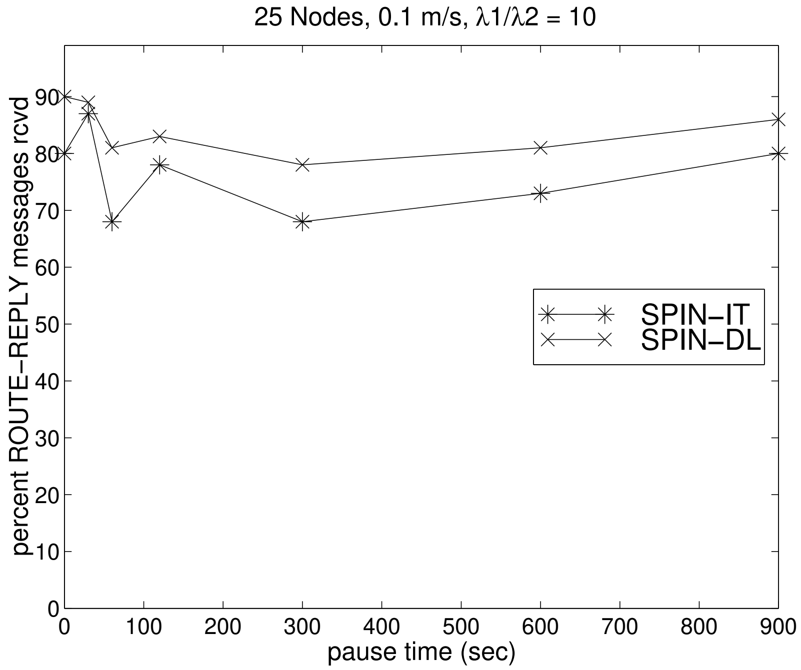
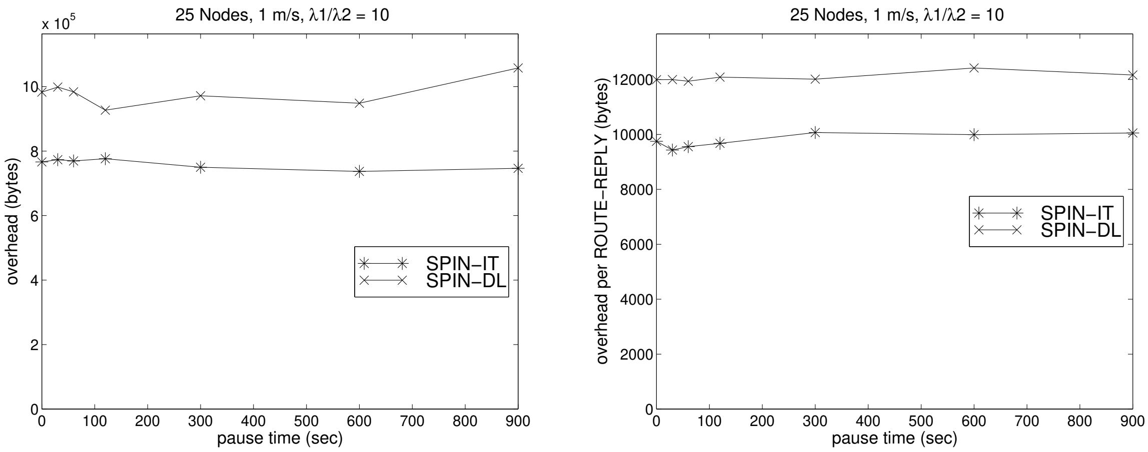


Figure 4-8: Percentage of ROUTE-REPLY messages successfully received at requesting nodes.



(a) (b)

Figure 4-9: Comparison of total routing overhead (a) and routing overhead per ROUTE-REPLY message received (b).

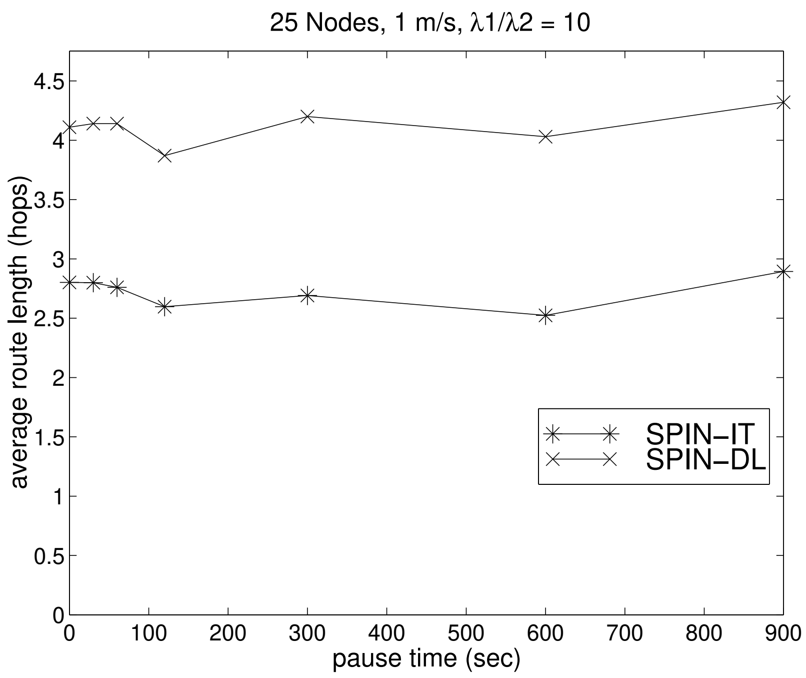


Figure 4-10: Average route lengths returned in ROUTE-REPLY messages in SPIN-IT and SPIN-DL, measured in hops.

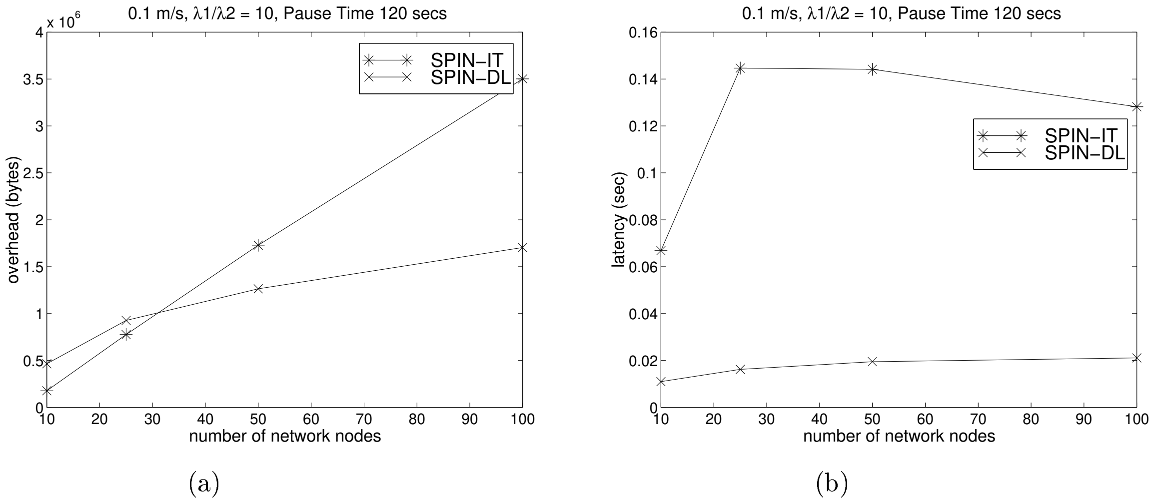


Figure 4-11: Here we study the impact of network size on the two protocols, where in (a) we show the routing overhead and in (b) we show latency.

find the shortest possible route. In our simulations, we did not keep track of the ideal routes. We did, however, monitor the lengths of the routes within the ROUTE-REPLY messages. This allows us to perform a relative analysis of the two protocols, as shown in Figure 4-10. As one would expect, since our implementation of SPIN-DL is restricted to the return of the inefficient route passing through the central node, the routes returned by SPIN-IT are consistently shorter by about 1 hop on average.

In Figure 4-11 we plot overhead cost and latency for the protocols as a function of the number of nodes in the network. In (a) we see that as the number of nodes grows, the overhead cost of SPIN-IT increases more rapidly than that of SPIN-DL, making SPIN-DL more efficient at higher node counts. In (b) we see that SPIN-DL always outperforms SPIN-IT in terms of latency.

Our final plot, Figure 4-12, measures the performance of route caching in SPIN-IT. In plots (a) and (b) we see that in a sparsely populated network with relatively little network traffic, caching has little benefit, in terms of overhead cost and latency. We increase the number of nodes and the request rate in plots (c) and (d), where we now see that route caching starts to improve the protocol's performance in cases with less mobility. In plots (e) and (f) we decrease mobility across all pause times by lowering the maximum node speed to 0.1 m/s, and we see consistently improved performance.

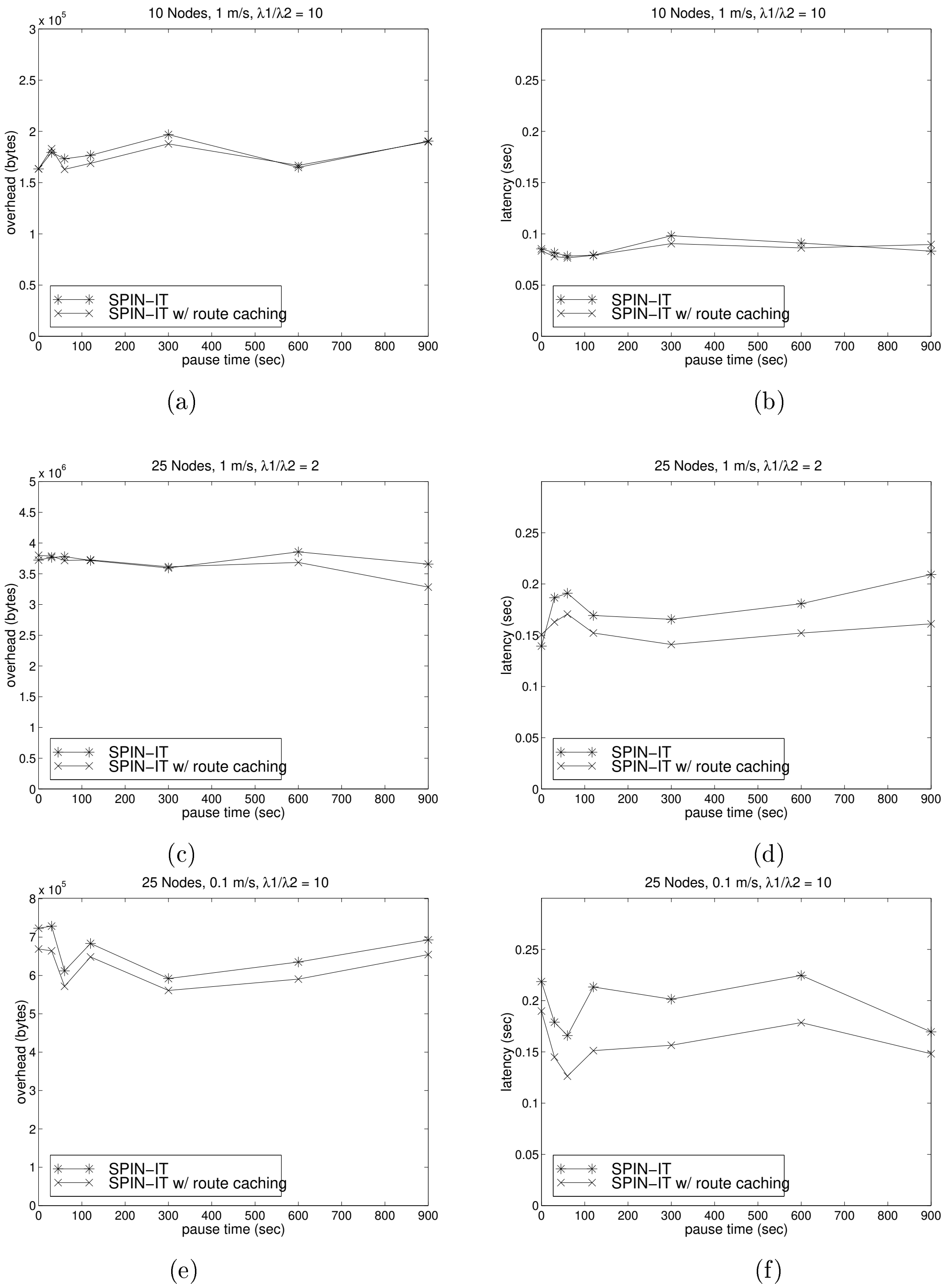


Figure 4-12: Effect of route caching on performance of SPIN-IT. Plots (a) and (b) show the minimal improvement in the case of a sparsely populated network with little traffic. Plots (c) and (d) show a performance improvement in the case of a more densely populated network with more traffic. In plots (e) and (f) the maximum node speed is 0.1 m/s, showing the clear benefit of caching in cases of less mobility.

Chapter 5

Related Work

Chapter 1 of this thesis referenced several published works that directly influenced our research. In this chapter we report additional work that relates to this thesis.

In [10] the authors conclude that most existing ad hoc routing protocols minimize the number of routing control packets at the expense of the information that is available for energy-based decisions. The result is that only a small fraction of the energy is spent on routing overhead, but the total energy is a large factor (5-50 times more that the theoretical minimum). In their paper they demonstrate how to modify DSR to use a variety of techniques, including adding energy information to ACK packets and transmit power information to route requests, to lower the overall energy use. Our protocols might be able to leverage these techniques.

The extension of proactive routing protocols to implement the *anycast* communication paradigm was presented in [26]. Their proposal demonstrates that anycast routing techniques can provide a one-to-any communication capability across a set of distributed servers or service points.

Building on the concept of using *backbone* nodes to comprise a centralized subnet for routing between nodes, in [12] the authors utilize location databases to form a *virtual backbone* for ad hoc mobility management. Whereas traditional backbone schemes impose high demands on channel bandwidth and node stability, the Randomized Database Groups scheme achieves minimal mobility management cost, adapting to changes in an ad hoc network environment.

Chord [32] is a distributed lookup protocol that provides the capability to locate a network node that stores a particular data item. This protocol maps keys to nodes through consistent hashing, and data location can be implemented on top of Chord by associating keys to data items.

In [6] the authors argue that many classes of modern applications, especially those targeted at mobile or wireless computing, demand the services of content-based publish/subscribe mechanisms. What these applications have in common is a new style of communication, whereby the flow of information from senders to receivers is determined by the specific interests of the receiver rather than by an explicit destination address assigned by the sender. The authors refer to this new communication infrastructure as *content-based networking*, and it is fundamentally similar to our definition of data-centric networking.

In addition to addressing the need for efficiency at the networking protocol level, other research has attacked the problem from a hardware perspective [29]. These efforts combine micro-sensor technology with low power sensor interface, signal processing, and RF communication circuits to create compact, low power nodes. The ideal systems level solution would combine the best hardware solution available with an efficient protocol like SPIN-IT or SPIN-DL.

Chapter 6

Conclusions

In this paper, we described two routing protocols based on the use of metadata to find data of interest. Unlike most routing protocols, SPIN-IT and SPIN-DL do not assume a node knows the identity of the node from which it seeks data. Our protocols are built upon a node's ability to describe what it wants, and the network's ability to respond to that request.

SPIN-IT uses a controlled flooding mechanism to propagate a request message to all nodes in a network. If a node has the data the requesting node seeks, it replies by reversing the path taken by the request message. In doing so, a route is established between the sink and source nodes, and data transfer can commence.

In the second protocol, SPIN-DL, source nodes supply a central node with fresh content information. The central node periodically floods the network with beacons to maintain valid routes. When a node seeks data, it sends a request message to the central node, and the central node replies, establishing a route from sink to source.

In this thesis we simulated the performance of these protocols under a variety of network configurations, and developed the following conclusions:

- The use of metadata to establish routes to data works well in lieu of prior knowledge of source node identity.
- A distributed approach was found to perform better than a centralized approach in terms of overhead cost when the rate of requests for data was significantly less than the rate of new data updates. A centralized model was better able to handle scenarios with higher request rates.
- Route caching, as an enhancement to our distributed protocol, showed value only in the

case of low network mobility.

- In general, neither protocol was dramatically impacted by node mobility.
- It is important to optimize the size and rate of update messages in a centralized model.

Although our results were promising, several opportunities for further research were identified. Both SPIN-IT and SPIN-DL can be optimized to further reduce overhead cost by adding algorithms to tailor messaging rates based on network dynamics. We would also like to study the behavior of these protocols under a broader set of conditions, by varying parameters like packet size in a controlled fashion. We could also experiment with the dynamics of a one-to-many relationship between the requesting node and the data sources. Path optimality would make an interesting metric to track in future studies.

Varying metadata size and the rate at which metadata changes is of particular interest, since these parameters impact the application space for our protocols.

Further investigation into latency performance is warranted. The design of our experiments may have unfairly favored the centralized approach.

In addition, we recognize that the central node in SPIN-DL represents a single point of failure, not to mention potential scaling problems with power consumption and computational burden. The "hub and spoke" nature of the SPIN-DL network could also make it more susceptible to collision problems. Further analysis into these issues is warranted, along with exploration into a mechanism for selecting the central node.

Finally, it seems a natural extension to this effort would be to combine SPIN-IT and SPIN-DL into a single protocol, taking advantage of each protocol's strengths, and downplaying their weaknesses. This hybrid protocol could either employ a mechanism for switching states based on network conditions, or a more sophisticated adaptive approach may be discovered.

Bibliography

[1] Sungjoon Ahn and A. Udaya Shankar. Adapting to route-demand and mobility (arm) for ad hoc network routing. 2001. *To appear: Proceedings IEEE ICNP.*

[2] J. Beck, A. Gefflaut, and N. Islam. Moca: A service framework for mobile computing devices. In *Proceedings of the ACM International Workshop on Data Engineering for Wireless and Mobile Access, August 20, 1999, Seattle, WA, USA*, pages 62–68. ACM, 1999.

[3] C. Bettstetter and C. Renner. A comparison of service discovery protocols and implementation of the service location protocol. In *Proceedings EUNICE 2000*, September 2000.

[4] J. Broch, D. Johnson, and D. Maltz. The dynamic source routing protocol for mobile ad hoc networks. Internet-Draft, draft-ietf-manet-dsr-06.txt. Work in progress. Earlier revisions published since March 1998., November 2001.

[5] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, October 1998.

[6] A. Carzaniga and A. Wolf. Content-based networking: A new communication infrastructure. In *NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, Scottsdale, Arizona, 2001.

[7] R. Castaneda and S. R. Das. Query localization techniques for on-demand routing protocols in mobile ad hoc networks. In *Proc. of the Fifth ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, 1999.

[8] S. Corson and J. Macker. Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations. IETF RFC 2501, January 1999.

[9] S. R. Das, R. Castaneda, and J. Yan. Simulation based performance evaluation of mobile, ad hoc network routing protocols. *ACM/Baltzer Mobile Networks and Applications (MONET)*, pages 179–189, July 2000.

[10] S. Doshi and T. Brown. Minimum energy routing schemes for a wireless ad hoc network. July 2001. *Submitted in July 2001 to IEEE INFOCOM 2002*.

[11] N. Gibbins and W. Hall. Scalability issues for query routing service discovery. In *Proceedings of the Second Workshop on Infrastructure for Agents, MAS and Scalable MAS*, May 2001.

[12] Z. Haas and B. Liang. Ad-hoc mobility management with randomized database groups. In *Proceedings IEEE ICC'99, Vancouver, BC, Canada*, 1999.

[13] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of HICSS 2000*, January 2000.

[14] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proc. 5th ACM/IEEE Mobicom Conference (MobiCom '99)*, August 1999.

[15] Ieee 802.11 wireless standards package. http://www.ieee.org, 1999.

[16] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of MOBICOM 2000*, pages 56–67, August 2000.

[17] Jini technology surrogate architecture specification and jini technology ip interconnect specification. http://www.jini.org/standards.

[18] Y.-B. Ko and N. H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. In *Proc. of the Fourth ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, 1998.

[19] Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. Modelling data-centric routing in wireless sensor networks. 2001. *Submitted in July 2001 to IEEE INFOCOM 2002*.

[20] X. Lin and I. Stojmenovic. Geographic distance routing in ad hoc wireless networks. Unpublished manuscript, December 1998.

[21] David A. Maltz, Josh Broch, Jorjeta G. Jetcheva, and David B. Johnson. The effects of on-demand behavior in routing protocols for multi-hop wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications Special Issue on Mobile and Wireless Networks*, August 1999.

[22] M. K. Marina and S. R. Das. Performance of route caching strategies in dynamic source routing. In *Proceedings of the 2nd Wireless Networking and Mobile Computing (WNMC)*, April 2001.

[23] J.C. Navas and T. Imielinski. Geocast -geographic addressing and routing. In *Proceedings of MOBICOM 1997*, pages 66–76, 1997.

[24] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of MOBICOM 1999*, pages 151–162, August 1999.

[25] Ucb/lbnl/vint network simulator. http://www-mash.cs.berkeley.edu /ns, 1998.

[26] V. Park and J. Macker. Anycast routing for mobile services. In *Proceedings Conference on Information Sciences and Systems (CISS)*, 1999.

[27] M. Pearlman and Z. Haas. Improving the performance of query-based routing protocols through diversity injection. In *IEEE Wireless Communications and Networking Conference 1999 (WCNC'99)*, pages 21–24, September 1999.

[28] C. E. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. Internet-Draft draft-ietf-manet-aodv-10.txt, work-in-progress, January 2002.

[29] G. Pottie and W. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[30] J. Rekesh. Upnp, jini and salutation - a look at some popular coordination framework for future network devices. Technical Report, California Software Lab, http://www.cswl.com/whiteppr/tech/upnp.html, 1999.

[31] L. Smith, C. Roe, and K. Knudsen. A jini lookup service for resource-constrained devices. In *4th IEEE International Workshop on Networked Appliances*, January 2002.

[32] I. Stoica, R. Morris, D. Kargerand M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings SIGCOMM, San Deigo, CA, August 2001*, pages 149–160. ACM, 2001.

[33] J. Wu and M. Zitterbart. Service awareness and its challenges in mobile ad hoc network. In *Workshop of Computer Science 2001: Mobile Communication of over Wireless LAN: Research and Applications, Vienna, Austria*, September 2001.