RICE UNIVERSITY

# Strata: A simple lightweight ad hoc communications infrastructure

by

**Ansley B. Post**

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

**MASTER OF SCIENCE**

Approved, Thesis Committee:

Peter Druschel , Chair
Professor of Computer Science

David B. Johnson
Associate Professor of Computer Science

Rudolf H. Riedi
Assistant Professor of Statistics

Houston, Texas

May, 2005

# ABSTRACT

## Strata: A simple lightweight ad hoc communications infrastructure

by

Ansley B. Post

This thesis presents Strata, a lightweight scalable implementation of the Safari architecture. Safari is an ad hoc network architecture, providing scalability to tens of thousands of nodes, integration of existing infrastructure, and self-organizing, decentralized network services. The Safari architecture is based on a self-organizing hierarchy that recursively partitions the network and assigns coordinates to nodes. Strata leverages the Safari structure to efficiently provide routing between two network hosts. As part of Strata's development, we developed a scalable, extensible network simulation environment that enables simulation of very large networks. The simulator's extensibility allowed us to explore the design space, revisiting many of the design decisions in the original Safari prototype. Early simulation results indicate that Strata can scale to several thousand fully mobile nodes with acceptable overhead.

# Acknowledgments

I would like to thank the members of my thesis committee for their helpful feedback. Both Dave Johnson and Rolf Riedi have provided valuable advice and discussion in all steps of the writing process. I would particularly like to thank my advisor, Peter Druschel, for his advice, guidance and patience. I would also like to thank Andreas Haeberlin for his insights and assistance. I would like to thank Alan Mislove for his help in proofreading. Last, but not least, I would like to thank my girlfriend Alexandria Kimsey and my family for their encouragement and moral support.

# Contents

# List of Figures

# Chapter 1
# Introduction

Traditional mobile ad hoc networking (MANET) protocols allow a group of nodes to form a network when no infrastructure exists. Protocols such as DSR [28, 29, 13] and AODV [44, 9] allow routing in such networks, even when all of the nodes may be moving. The protocols successfully find routes between nodes efficiently in networks that consist of up to several hundred nodes.

One scenario for ad hoc networking is providing communications infrastructure in a disaster, this scenario may have thousands of participating nodes, more nodes than traditional MANET protocols were designed for. Moreover, these protocols provide basic connectivity but do not provide higher level services required for coordinating the activities of the people involved. In this scenario, basic communication is not enough; self configuring services such as email or voice services are needed for users to effectively communicate and coordinate efforts.

The goal of the Safari project is to address these shortcomings and allow the construction of large networks with application support and without reliance on network infrastructure. It aims to build an architecture, "that is able to provide network connectivity and basic network services in a self-organizing fashion, using available infrastructure when and where it exists, without depending on its availability" [2].

The Safari project has defined an architecture, the Safari architecture, that defines several key components that are needed to realize the project's vision. The architecture defines: (1) a virtual coordinate structure, (2) host to host routing built upon the coordinate structure, and (3) a key based routing primitive, which can be used to build decentralized applications.

Strata is a lightweight and scalable implementation of the first two parts of the Safari architecture. Host to host routing is provided in a way that involves no complex reactive component and leverages the coordinate structure. Optimizations are added to this basic protocol as needed to ensure adequate performance characteristics.

This thesis describes the design and evaluation of Strata. The design of Strata went hand in hand with the design of a scalable extensible high level network simulator. At each step of the design of Strata, the simulator was used to explore the possible design space in order to arrive at our design.

The first component of Strata described is the structure formation and coordinate assignment algorithm, upon which everything else depends. The design settled upon is simple and, once optimized, quite efficient. The improved design is shown to scale to several thousand nodes.

Second, the routing layer, which provides connectivity within the network, is developed. The design for host to host routing is presented and evaluated. The

performance of host to host routing is evaluated in depth and is shown to scale up to several thousand nodes.

The remainder of this thesis is structured as follows. Chapter 2 presents related work. Chapter 3 describes the decentralized structure formation protocol and Chapter 4 presents the routing system. Chapter 5 presents evaluation of the basic system. Chapter 6 describes optimizations to both the routing and decentralized structure formation that were guided by the results of the initial evaluation. Chapter 7 evaluates the optimizations made and compares the results against the simple version of the protocol. Chapter 8 examines possible future work and Chapter 9 concludes.

.

# Chapter 2
# Related Work

The problem of wireless communication has been well studied in the past [5, 31, 10, 46, 30, 32] and many standards have been established [27, 25, 26, 40, 1]. The problem of transmitting bits and regulating medium access has been extensively researched, providing a base for higher level research. Routing along one-hop paths when infrastructure is in place is similarly a well studied problem and several solutions have been standardized [27, 25, 26, 40]. Recently the problem of how to route in a mobile ad hoc network has been a hot topic of research. This thesis continues along that line of research.

In this thesis, Strata, a system that provides routing in a large scale mobile ad hoc network is described. Strata is a scalable lightweight implementation of the Safari architecture. Strata is build on top of a virtual coordinate structure, and that structure is then used to provide routing capabilities. Strata is designed for a large scale ad hoc wireless deployment, where there are thousands of nodes, and many, if not all, are mobile. In the following sections we examine the different approaches that have been taken to solving the problem of routing in an ad hoc wireless setting.

## 2.1 Traditional Wireless Routing

The problem of how to route in a wireless ad hoc network has been studied extensively; many proposed protocols have been very successful in routing in networks where the number of nodes is on the order of several hundred or less. These traditional approaches all have a component that scales linearly with the number of nodes, thus fundamentally limiting their suitability for very large networks.

In Dynamic Source Routing (DSR), [28, 29, 13] routes are found to a destination on demand, or reactively. DSR finds a route to the destination for a packet by flooding a Route Request message. A node knowing how to reach the destination will return a Route Response. The path of this Route Response forms a source route that can be used to reach the destination. This mechanism does not scale well to large networks, as it requires, in the worst case, a flood of the entire network. There are several optimizations in the DSR protocol that reduce the frequency of these floods by using smart caching and range limited floods, but in the worst case a whole network flood will be necessary.

A competing reactive protocol is the Ad Hoc On-Demand Distance Vector (AODV) routing protocol [44, 9]. As with DSR, AODV discovers routes to the destination only as they are needed. It uses flooding to discover routes, which works well in small networks, but again, does not scale well as the network size increases.

On the opposite end of the spectrum is DSDV [43], which maintains the routing structure aggressively by exchanging distance vectors to all destinations among the participating nodes. Routes are proactively kept up to date by this exchange mechanism, so DSDV is classified as a proactive protocol. Exchanging distance vectors grows more difficult as the number of nodes in the network increases and the vectors must be kept up to date. DSDV, and proactive protocols in general, have the advantage of lower overhead if there are many routes, since all will be updated with the periodic maintenance. However, when the network is idle there is a constant overhead even when no packets are being sent.

SHARP [45] is a proposed routing protocol that dynamically changes between proactive and reactive modes depending on the workload and the desired overhead. This allows the protocol to avoid some of the deficiencies of proactive routing by scaling back periodic maintenance, if it is not needed. However as network size increases SHARP still suffers from the problems of both reactive and proactive protocols. Either the proactive maintenance traffic must still be propagated through the network, or a flood of the network is needed to discover a route reactively.

Another hybrid proactive/reactive protocol is the Zone Routing Protocol (ZRP) [23, 21, 20, 22], which partitions a node's view of the network into a local zone and a remote zone, which includes the rest of the network. Within the local zone routes are immediately available and maintained proactively. Information about destinations in

the remote zone is discovered proactively and shared with all nodes within the local zone. This partitioning aggregates information about destinations in the remote zone and limits the scope of proactive maintenance, but in the case where a remote destination must be discovered a whole network flood may still be necessary. This does not scale well to large networks where nodes must frequently discover nodes in the remote zone.

The Temporally-Ordered Routing Algorithm (TORA) [41] models the network as a directed acyclic graph. When a path to a destination is requested, the algorithm assigns heights to each node in the path to the destination. Packets flow "downhill" to the destination from nodes with greater heights to those with smaller height. The closer to the destination a nodes is the lower its height, thus packets will always travel towards the final destination. In the case of topology changes the heights are recomputed to maintain the downhill property. There may be multiple downhill paths to a destination in TORA, which may be used to alleviate congestion. As with other protocols in this category, TORA has a flood based route discovery process which limits its scalability.

## 2.2 Location Aided Wireless Routing

One approach to improve routing performance has been to provide additional information to the routing protocol. Systems such as [47, 33, 35] have leveraged geographic coordinate information provided by GPS receivers in order to improve routing

performance and scalability. Nodes in these systems advertise their geographic position and then the routing takes place by repeatedly forwarding through geographically closer nodes until the destination is reached. These systems are similar to Strata, since they also route in a coordinate space and require a way to map current location to network address. However all these systems require dedicated location hardware in order to function correctly, a requirement not necessary in implementations of the Safari architecture.

The first example of such a protocol is Location Aided Routing (LAR) [35]. LAR uses geographic information and a modified version of DSR to route. Route Requests are constrained to a geographic area for efficiency. In order to find the current location of a destination, LAR, in the worst case floods the network. As with the traditional routing approaches this does not scale as network sizes grow larger.

Another example of geographic routing protocol is the Distance Routing Effect Algorithm for Mobility (DREAM) [47]. DREAM operates similarly to LAR except that location changes are proactively flooded throughout the network. This approach is even less scalable then LAR, since all changes are flooded to the entire network.

Greedy Perimeter Stateless Routing (GPSR) [33] is a geographic routing protocol that is scalable to several hundred nodes. GPSR greedily routes toward the destination. If the packet becomes stuck due to an obstacle, GPSR routes around the perimeter to find a path to the destination. GPSR uses a distributed location

database that eliminates the need to flood in order to discover a node's current geographic position. A node's location is always stored by the node closest to a specific geographic location, so a simple routing operation instead of a flood is used to discover its location. GPSR does not specify all the details of the location service as part of the protocol; it instead recommends the use of the GRID location service [37].

## 2.3 Cluster Based Wireless Routing

The use of clusters in ad hoc routing has been proposed [8, 12, 24, 11, 3, 39, 38, 15, 14, 19] to reduce the maintenance and allow routing to clusters of nodes instead of individual nodes. This simplifies route maintenance and routing, but pushes complexity to the cluster formation and maintenance. Since clusters must actively be maintained, this approach is better suited to networks where the amount of mobility is low, otherwise cluster maintenance becomes prohibitively expensive.

Banerjee et al propose a system [8] for clustered routing that groups nodes together in order to limit the amount of state management required. Limits for minimum and maximum nodes per cluster can be specified, and the protocol will guarantee inter-cluster connectivity if the underlying network is connected. The system is mainly intended for use with sensor nodes where there is no mobility and cluster membership only changes when a node fails. Thus the deployment environment is decidedly different than that for Strata.

The Terminode project [12, 24, 11, 3] uses clustering to form a routing system. However, the research is ongoing and the full routing and clustering system has yet to be fully described. Much of the work is still focused on the lower levels of the protocol stack.

## 2.4 Landmark Based Wireless Routing

Strata uses a virtual coordinate system that attempts to approximate the coordinates used in geographic routing systems. However, since nodes in Strata are not assumed to have GPS receivers, this coordinate structure is built by using distances to a set of well known nodes in the network. These well known nodes are known as *landmarks*. This general approach is based on the concept of a landmark hierarchy for routing [48, 49].

The Landmark hierarchy [48, 49] has been proposed as a way for routers to organize a routing structure in a decentralized manner. This structure is built via a decentralized coordination algorithm that assigns levels to routers and each router is visible for a number of hops based on its level. Higher level landmarks are visible to a larger portion of the network. A node forms a virtual coordinate from the landmark it is closest to at each level. Routing is then accomplished hierarchically by routing to the closest shared landmark until the destination is reached. This technique has been shown to be scalable but was designed for a fixed network of routers in a wired network and thus has to be altered to be feasible in a mobile wireless network.

LANMAR [42] is an ad hoc routing protocol that leverages ideas from Landmark routing. LANMAR is designed for environments where nodes move together in groups, such as on a battlefield. Using this assumption, LANMAR is able to achieve good performance, however we do not make this assumption and thus Strata must solve a harder problem.

Beacon Vector Routing (BVR) [18] is a routing system for sensor networks. Like Strata it uses landmarks to define coordinates. These coordinates are not based upon a hierarchical set of landmarks, instead they are a vector of distances to every landmark in the network. The packets are routed using greedy forwarding to the node that matches these distances. Each landmark in BVR must flood the entire network to provide reference points to the other nodes. This is possible in a large network because it has to be done very infrequently, since the network is assumed to consist of immobile sensor nodes. The BVR approach is not directly transferable to the large scale mobile networks that Strata is designed for, as the overhead required to keep distance vectors to the landmarks up to date in a mobile network would be high.

### 2.4.1 L+

L+ [16] is a protocol that leverages the ideas from Landmark routing and applies them to mobile ad hoc wireless networks. L+ is similar to Strata in that it builds a hierarchy based on self organizing landmarks. It then uses this hierarchy for routing. L+ differs from Strata is several areas.

L+ uses distance vector routing to advertise route distances between nodes. These distance vectors are advertised during beacon floods of the landmarks. Strata relies only on the paths of the beacons and no further information. Advertising additional information adds complexity to the protocol and increases the size of beacon packets. Mobility in L+ may cause triggered updates upon topology changes, where as no such proactive mechanism is present in Strata.

In order to increase stability in L+ each node is assigned two hierarchical addresses. Each node chooses a parent and a backup parent at each level; a node forms its primary address from its parents and its secondary address from its backup parents. A node is routable via either address and the backup can be used if routing to the primary fails. Since each node has to have a backup parent, a landmark will only step down if every node in the area has both a parent and a backup. This can cause there to be extra landmarks as compared to Strata and thus extra overhead. To gain the stability that having backups achieves, Strata instead attempts to identify stable nodes to become landmarks.

### 2.4.2 Masai

The most similar work to Strata is Masai, a separate implementation of the Safari architecture. This implementation was described in Du's thesis [17]. Like Strata, Masai uses drums (landmarks) that emit beacon floods to define virtual coordinates used for routing. Masai is different in several key ways from Strata: in its basic routing

architecture, how it repairs stale information, and the implementation of differing sets of optimizations.

Masai uses a hybrid proactive/reactive protocol where the reverse path of beacons is used for routing to far away distances and DSR is used for routing close to the destination. This differs from Strata, where the routing algorithm is the same regardless of distance to the destination. Masai's use of DSR allows local routes to be discovered only when needed for routing. Strata has no reactive component such as DSR and thus the overhead is constant and independent of the traffic load.

Additionally Masai implements a sophisticated optimization for local route repair to patch paths that have broken. This mechanism, while effective, is somewhat complex. Strata foregoes this approach and instead explores other optimizations to route effectively when paths break due to mobility.

Strata implements an optimization for selecting stable nodes as drums, that is not used by the Masai implementation, however it is general enough that it could be applied to either implementation. Masai defines different values for the scope of beacon floods. Beacons will always travel the minimum TTL distance in Masai, where as in Strata they are confined to the nodes that share the same higher level drum. This can result in slightly higher overhead in Masai, but results in quicker discovery of a landmark's existence.

### 2.4.3 Analysis

An initial analysis of the drum protocol used in Masai was performed in Khan's thesis [34] in which he applied analytical models in order estimate the overhead of the structure formation system. The analysis shows that the per node overhead in Masai will grow logarithmically with the network size. This is complementary work to both the Masai and Strata implementations of Safari. The overhead results from both Strata and Masai match those expected from this work.

# Chapter 3
# Decentralized Structure Formation

In this chapter, we present a method for building a virtual coordinate structure on a network of mobile, wireless nodes. We do this with the motivation of later building on this structure to route between participants in the system.

## 3.1 Drum Designation

The term *drum* is used to designate a node with the specific role of providing a reference point to other nodes in the network. The coordinate structure that emerges in the network comes from the designation of particular nodes as drums. Each drum has a *level* and this level determines the scope of its visibility in the network. Higher level drums are visible to more nodes in the network than lower level drums. This naturally raises the question of how drums are designated, and how drum levels are determined.

When the network begins there are no drums in the system. Every node expects to see a drum of one level higher at a distance of $d_n$ hops, where n is the level. A node expects a level 1 drum within 0 hops, and thus every node becomes a level one drum. If there are no drums within $d_n$ hops, then a random node of level $n-1$ steps up to become a drum of level $n$. When beginning from the state where there are no

```
(1)  shouldStepup() {
(2)    if ( drumExists(level + 1)) {
(3)      if ( distanceTo(drum level + 1) > d_{level+1})
(4)        return true
(5)    } else if (nodeExists level){
(6)      return true
(7)    }
(8)    return false
(9)  }
```

**Figure 3.1** The pseudocode for determining if a node should step up.

drums, all nodes become level one drums and then the process repeats and higher level drums emerge.

A node only steps up if there exist other nodes in the system at the same drum level. This prevents unbounded stepping up by the highest level drum, since it never sees any higher level drums. There is only one drum of highest level in the network and the level of the network is said to be that of the highest level drum. For example, if the highest drum in the network is a level five drum, then the network is a level five network.

The above algorithm will allow nodes to be designated as drums, but it is possible that more drums emerge than should exist within a specified area. This situation, while technically correct, is undesirable. If two drums are closer than $1/2d_n$, one of the drums will step down to correct the situation.

The pseudocode in Figure 3.1 describes the step up process, but glosses over the details of how nodes determine if another node exists, and if so how far this node is

away. Section 3.3 explains the use of reference beacons to establish the distance a node is from a drum.

## 3.2 Virtual Coordinates

Once drums have established their level in the network, they then associate with other drums to determine their position in a virtual coordinate space. A drum of level $n$ associates with a drum of $n + 1$ for all but the top level. The one top level drum does not associate with any other node. The lower level drum considers the higher level drum it associates with to be its parent. Thus, the highest level drum has no parent.

A drum always chooses the closest drum of the next level as its parent. Ties are broken deterministically, by choosing the candidate with the lower nodeId. This association is all that is necessary for a node to determine its coordinate within the virtual coordinate space.

Once a parent has been selected, each node determines its current coordinate within the network. A coordinate is a $k$-digit string where $k$ is the level of the network. Each digit has a value between 0 and $m$, where $m$ is larger than the maximum number of drums with the same parent. A drum of level $n$ forms its coordinate by choosing a random digit for each of the digits from 1 to $n$. The digits from $n$ to $k$ are taken from the coordinate of the node's parent. The highest level drum in the network randomly

generates all of its digits, and level 1 drums only generate one digit and inherit the rest of their digits from their parent.

It is possible for nodes associating with the same parent to randomly pick the same value in the $n$th digit and thus have a collision in the identifier space. In order to resolve this conflict, when a node learns of another node with the same id, it selects a new random coordinate in the $n$th digit, if it has the lower nodeId. Eventually this process converges and all drums of level $n$ have a unique coordinate.

All decisions in choosing the coordinate are based on a node's local view. No part of the algorithm relies on agreement with other nodes in the network. A node updates its coordinate only periodically, and thus the process is insensitive to mobility that happens in between the periodic updates.

## 3.3 Reference Beacons

In the previous sections we have defined simple algorithms to define when a drum should step up and for finding a node's coordinates. In these algorithms we have left open how a node determines how close it is to others, and if a higher level exists. In this section we introduce reference beacons for this purpose.

A drum of level $n$ periodically emits a beacon every $t_n$ seconds, where $n$ is the drum level. These reference beacons are flooded periodically in order to inform nodes of the existence and locations of drums in the network. The beacon information is used by nodes to decide when to step up or down and which higher level node to

associate with. Each reference beacon includes a hop count field, which is incremented up from zero whenever it is forwarded by a node during flooding. By looking at this field, upon receipt of a beacon, a node can determine its distance in hops from the originating drum.

The beacon packets are flooded throughout the network up to a certain distance, measured in hop count. This distance must minimally be $d_n$ hops, so that a node can decide whether there are enough drums in the area or that it should step up. In practice this distance is larger and is required to be so in order to make routing work. The reasons for this will become clear when the routing algorithm is explained in Chapter 4. We endeavor to keep the scope of flooding to be as small as possible, because flooding a packet is quite expensive — especially in a bandwidth limited broadcast medium.

The actual scope of beacon packets is defined in the following way. A beacon from a drum of level $n$ is flooded to all nodes sharing the same digit in the $n+1$-th position in its coordinate. As we will show in Chapter 4, this expanded scope is sufficient to allow routing within the system.

In Figure 3.2 we see an example structure formed by Strata. The size of the squares is proportional to the level of each drum. The outlines indicate drums associating with the same parent at each level. This is generated from actual simulations, as described in Chapter 5, of a network running the described protocol.

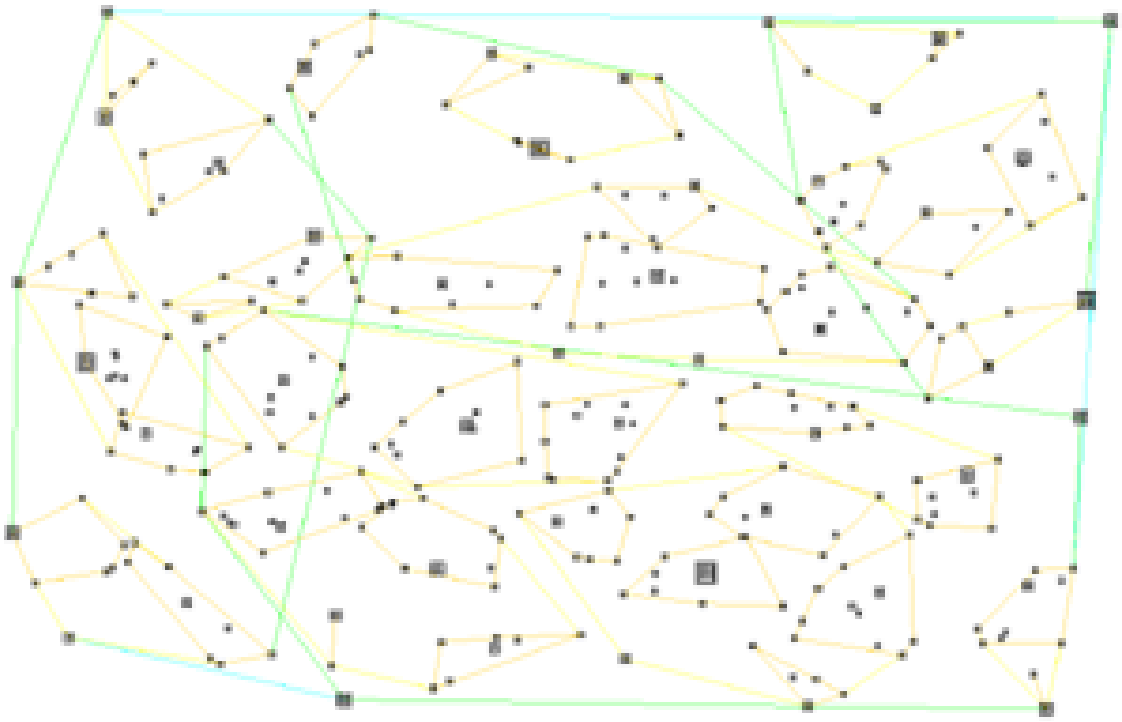


**Figure 3.2** A picture of the structure formation in a network of 300 nodes.

Figure 3.3 shows the pseudocode in Figure 3.1 expanded in greater detail to utilize the reference beacon information. The check to see if there have been any drums received at the current level is needed to prevent the highest level drum from continuously stepping up because it doesn't see any beacons from a higher level drum.

## 3.4 Detailed Design

Up to this point we have developed a very general scheme for building the coordinate structure without specifying parameters. In order to simulate Strata, we have chosen a specific set of parameters. We have chosen $d_1$ to be 0 and $d_2$ to be 1. All other distances have been defined to be $d_n = 2 * d_{n-1}$. In this way we have defined

```
(1)   shouldStepup(beaconRcvd) {
(2)      if (beaconRcvd[ currentLevel + 1] != null) {
(3)         if ( best(beaconRcvd[currentLevel +1]).hops >
(4)               minDistance(currentLevel +1) )
(5)            return true
(6)      } else if (beaconRcvd[currentLevel] != null){
(7)         return true
(8)      }
(9)      return false
(10)  }
```

**Figure 3.3** The pseudocode for determining whether a node should become a higher level drum.

every node to be a level 1 drum and all nodes are one hop away from a level 2 drum. Similarly we have chosen $t_1$ to be 1 and $t_n = 2 * t_{n+1}$. This choice of beacon period provides a balance between good routing performance, and low overhead from beacon floods.

# Chapter 4
# Routing

In this chapter, routing is described based upon the coordinate system developed in the previous chapter. Routing between two nodes in the system is the foundation for communication, coordination, and building network services and applications.

## 4.1 Addressing

In order to perform routing we assume that each node has a unique permanent identifier. This identifier is known as the *nodeId*. The nodeId can be generated from globally unique information such as the MAC address of the physical interface.

In addition to a nodeId, every node in the system keeps track of its current location within the coordinate space formed by the drums, as mentioned in the previous chapter. For the moment, we assume that we will always know the current coordinate of a node when we decide to route to it. In practice, this mapping is achieved through a higher level service that provides mappings from nodeId to current coordinate. Building this service is the subject of ongoing research, but a variant of an existing lookup service such as Grid [37] could be used.

## 4.2 Routing State

In order to route packets to their destinations, each node maintains a routing table providing next hop information for delivery of packets. The routing table maintains

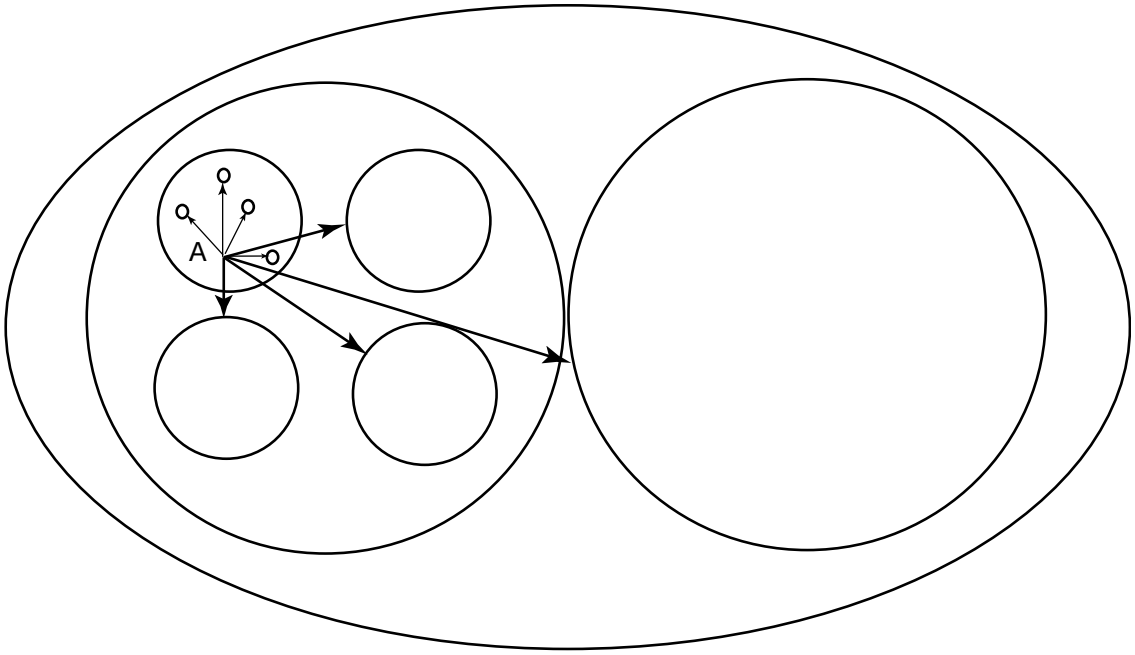



**Figure 4.1** The routing state that a node must keep for a 4 level network.

a path to every level $n$ drum with which it shares the same value in its level $n+1$-th coordinate, for all values of $n$. Since we have defined a node to be a level 1 drum, a node knows of every node that has the same level 2 parent. This state is maintained by the beacons emitted by drums as described in Section 3.3 .

The number of paths required to be kept is logarithmic in the size of the network, as there are logarithmically fewer drums at each level. As shown in Figure 4.1, a node must keep track of only a small number of paths to other nodes. In this four-level network, a node must only keep track of how to get to the other nodes that share its level 2 parent as well as at least one node that it shares a level 3 parent, and so on.

Every time a beacon arrives, a record of the node that forwarded it is stored in the routing table. This record is stored with an expiration time and hop count. The expiration time is set to be $t_n$, so that an entry expires if a beacon is not heard every beacon period. An expired entry is immediately removed from the routing table so that it will not be used to route.

Since all beacons are flooded, it is possible that a node will hear the same beacon broadcast repeatedly from different neighbors. When this happens, a node chooses the route with the smaller hop count so that the shortest known route is used to reach the destination.

## 4.3 Routing Algorithm

When a packet is routed the following procedure is repeated at each node. A node compares its coordinate to that of the destination, it then routes towards the drum that shares one more coordinate with the destination. For example, if a packet destined for coordinate 7.8.12.48 is received at a node with coordinate 7.16.44.17, the node will route toward the drum whose coordinate starts with 7.8. It then looks in the routing table to find the node it received the beacon from 7.8 from. It then uses this node as the next hop towards the destination. This process is repeated until the packet is delivered to the node whose coordinate matches the destination.

Since routing is iterative at each node, it is unlikely a drum will actually receive a packet routed towards it, as the path will most likely intercept a node with more

specific routing information. Nodes associating with the drum being routed towards always have more information about destinations around that parent drum. As soon as it reaches such a node, the packet be routed more directly the destination. Thus, routing responsibility is shared among all nodes in the system, independent of their drum level. Higher level drums just provide the reference point needed for routing.

In a static network with no coordinate changes, this routing algorithm will always route a packet to the correct destination. When mobility is introduced, some of the paths formed by beacons become stale. When this occurs, some destinations may be unreachable until the path information is refreshed.

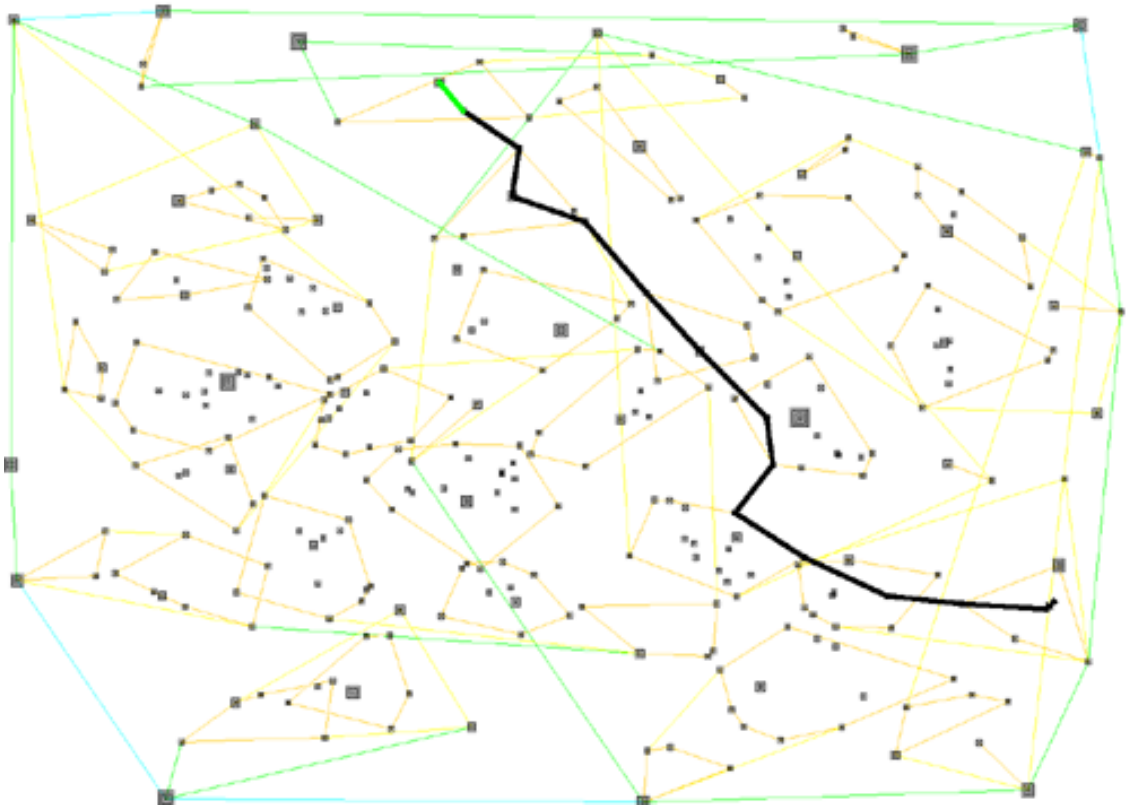


**Figure 4.2** An example of routing using Strata

# Chapter 5
# Evaluation

## 5.1 Simulator

### 5.1.1 Simulation Implementation

Strata was developed in conjunction with a scalable simulator called StrataSim. StrataSim was built because existing simulators such as ns-2 [4] or GloMoSim [50, 7, 6] could not provide the network scales we wanted to simulate. ns-2 in particular could only simulate scenarios up to 1000 nodes. In order to achieve additional scalability, the simulator we developed models the network without simulating low level characteristics such as radio propagation. StrataSim is extensible and uses a layered model similar to the OSI model [51]. It does not model details of the physical layer such as signal strength and radio propagation, it also uses an idealized infinite bandwidth channel. More detailed modeling of these features could easily be added later by switching to more detailed models of physical and MAC layers, but this would reduce the size of simulations capable of being run in a reasonable time.

### 5.1.2 Methodology

In order to perform the experiments below, the following procedures and parameters are used unless otherwise stated. For all experiments we use a node density of $50/km^2$ ; for experiments of more than 50 nodes we scale the dimensions appropriately. We use a constant radio range of 250m. In experiments where nodes are

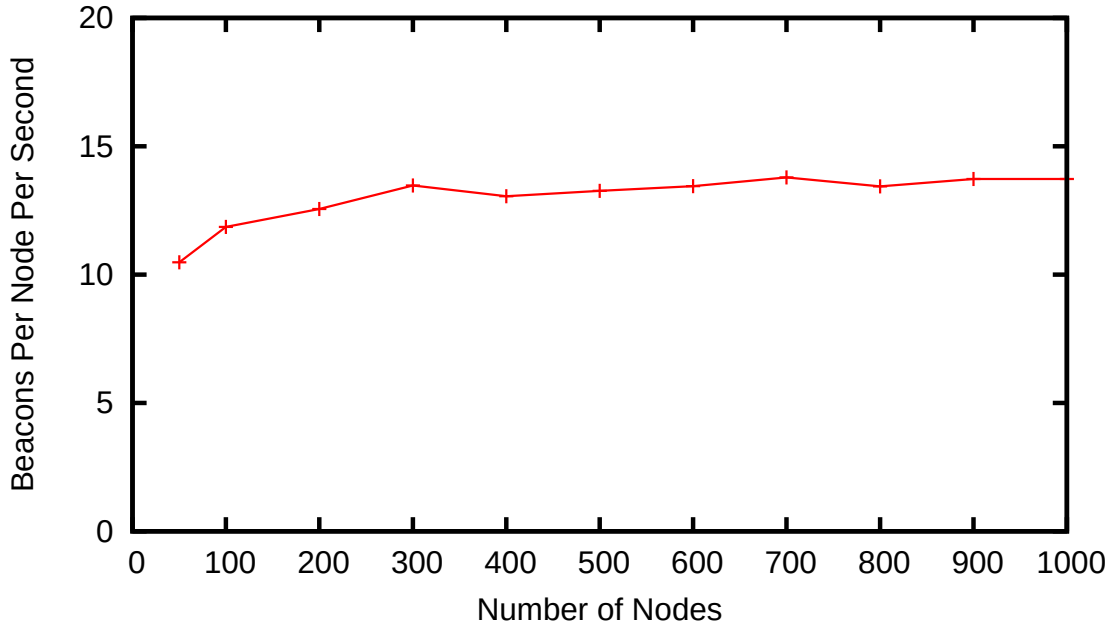



**Figure 5.1** Beacon overhead per node in unoptimized protocol

mobile, we use the random waypoint [13] model. The parameters used in the random waypoint model are a maximum node speed of $5m/s$ and 0 pause time. Each simulation is run for 1250 simulated seconds; the first 350 seconds are used to allow the network to stabilize and results are reported from the final 900 seconds. Each data point on a graph reflects the average of five runs.

## 5.2 Drum Protocol

In this section, we evaluate the cost of forming the coordinate structure. This is the predominant static cost of the system. An advantage of our protocol is that there is only fixed constant overhead regardless of the amount of traffic being sent. Unless otherwise stated, all nodes are mobile and are moving at a maximum speed of $5m/s$.

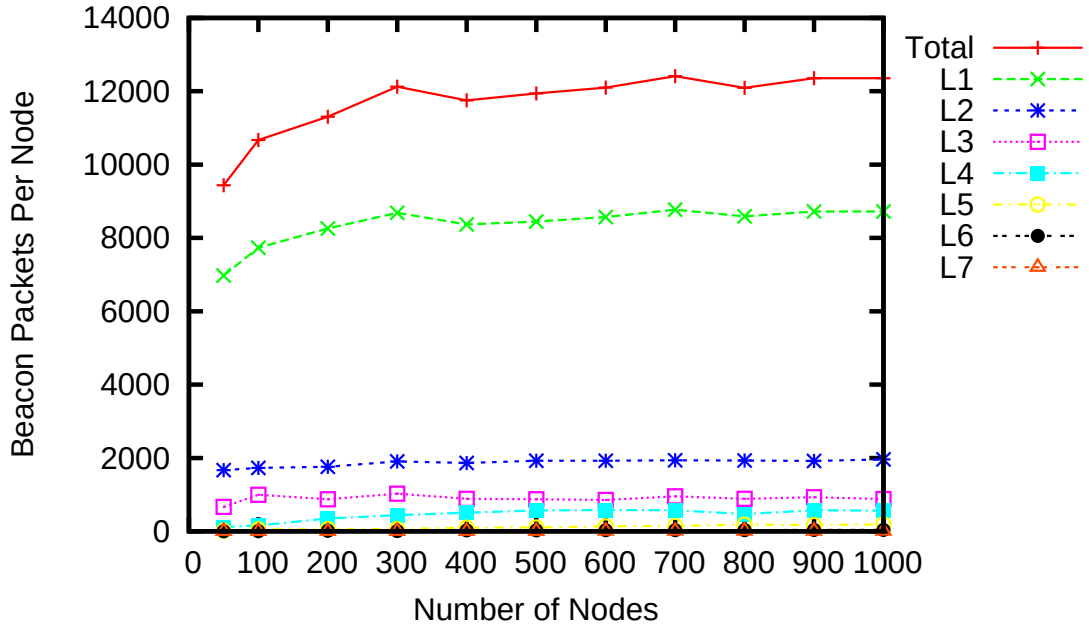



**Figure 5.2** Beacon Overhead by level as network grows

In Figure 5.1, we show the basic overhead per node participating in the system. Note that the overhead grows very slowly as the number of nodes in the system increases. In analysis it has been shown that overhead is expected to grow logarithmically with the size of the network [34]. This agrees with what is shown in Figure 5.1.

The drum overhead is broken down in Figure 5.2. The predominant cost is the beacons flooded by level 1 drums. This makes up more than half of all overhead and dwarfs the cost of transmitting beacons from higher level drums.

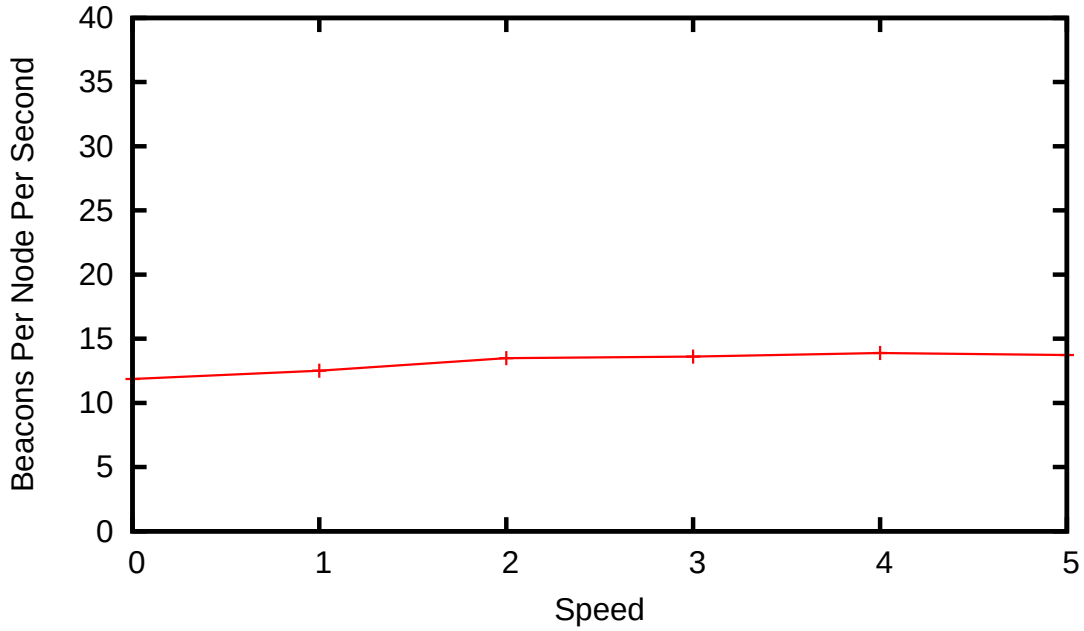



**Figure 5.3** The overhead due to beacons as node speed is altered

A desired property of the drum protocol is that the amount of overhead is not increased by mobility. As Figure 5.3 and Figure 5.4 show the overhead remains close to constant as the number of mobile nodes changes and the speed at which nodes move is altered. It increases slightly in both cases because mobility causes drums to step up and down in order to maintain the distance invariants. These situations can result temporarily in there being too many drums in an area, thus increasing the number of beacon packets transmitted.

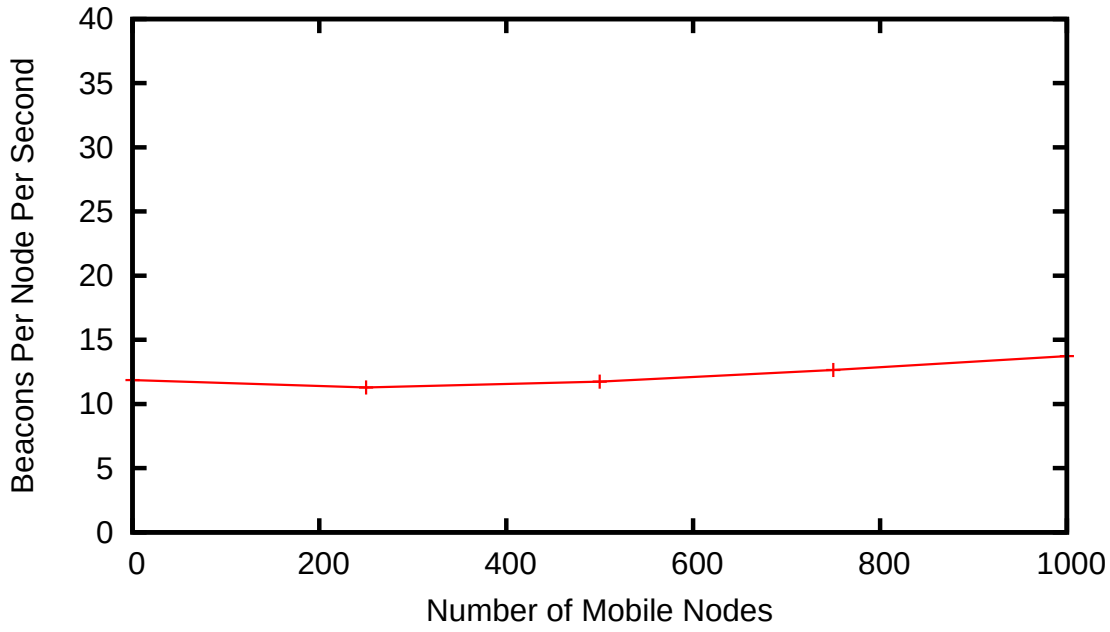



**Figure 5.4** The overhead due to beacons in a 1000 node network, as number of mobile nodes is varied

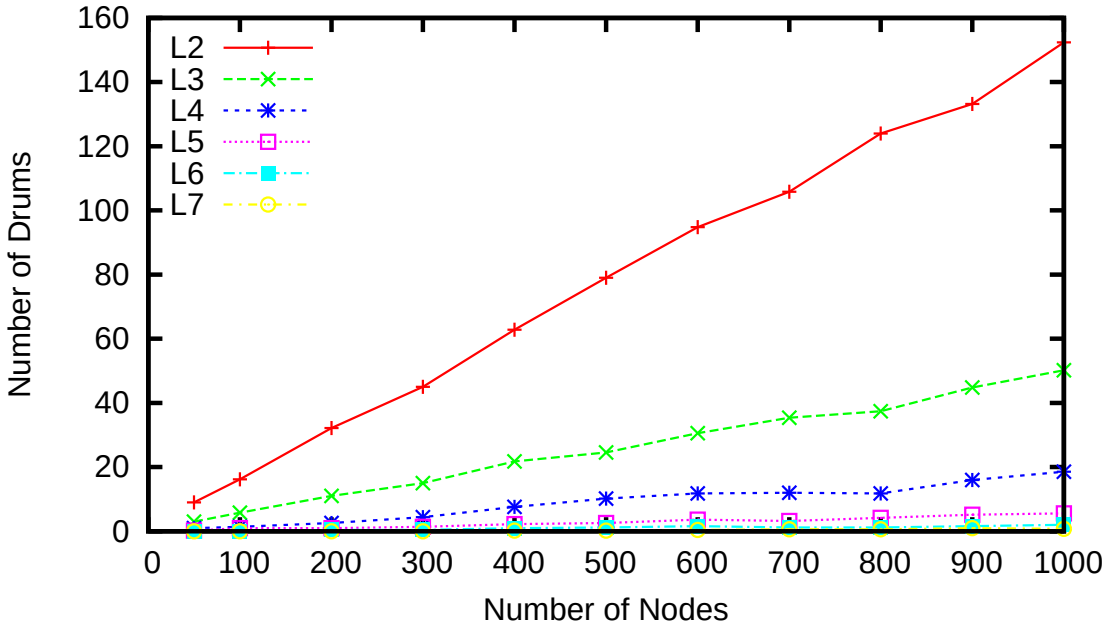



**Figure 5.5** The number of drum nodes, broken down by level

The drum protocol must be efficient in the number of nodes that are selected to become higher level drums. Figure 5.5 shows the number of nodes that are drums at each level. Figure 5.6 shows the information as a ratio. About 16% of nodes are chosen to be level two drums, and exponentially fewer are chosen to be higher level drums. In order to demonstrate this, the percentage of nodes that are drums for a particular level is shown in Figure 5.7 graphed on a log scale. Only a very small percentage of nodes are required to be high level drums.

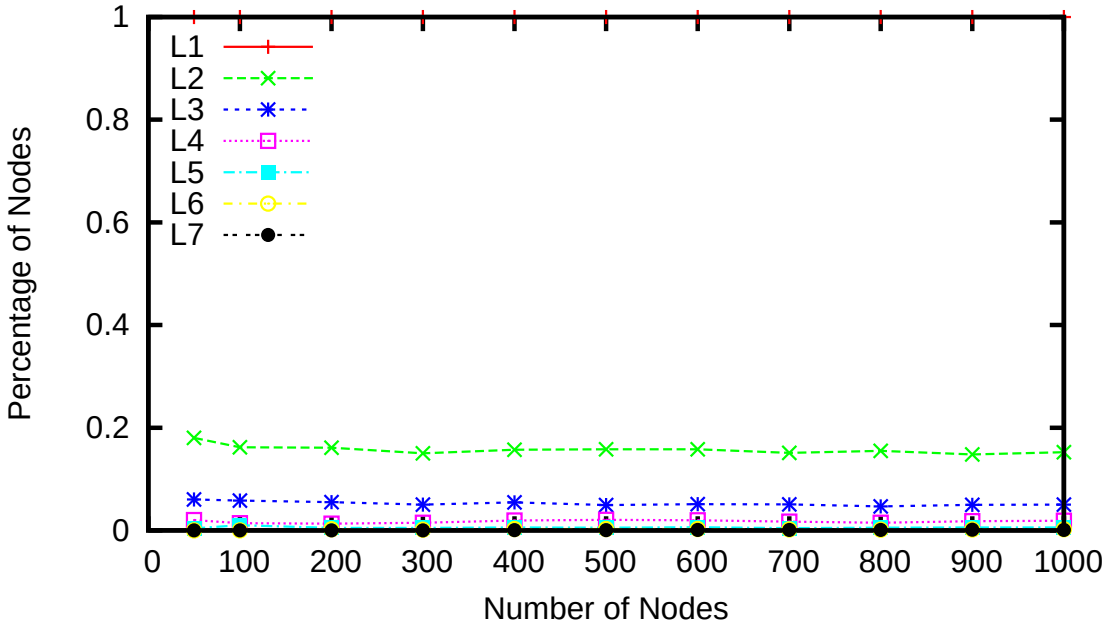



**Figure 5.6** The number of nodes that are drums as the network size is increased

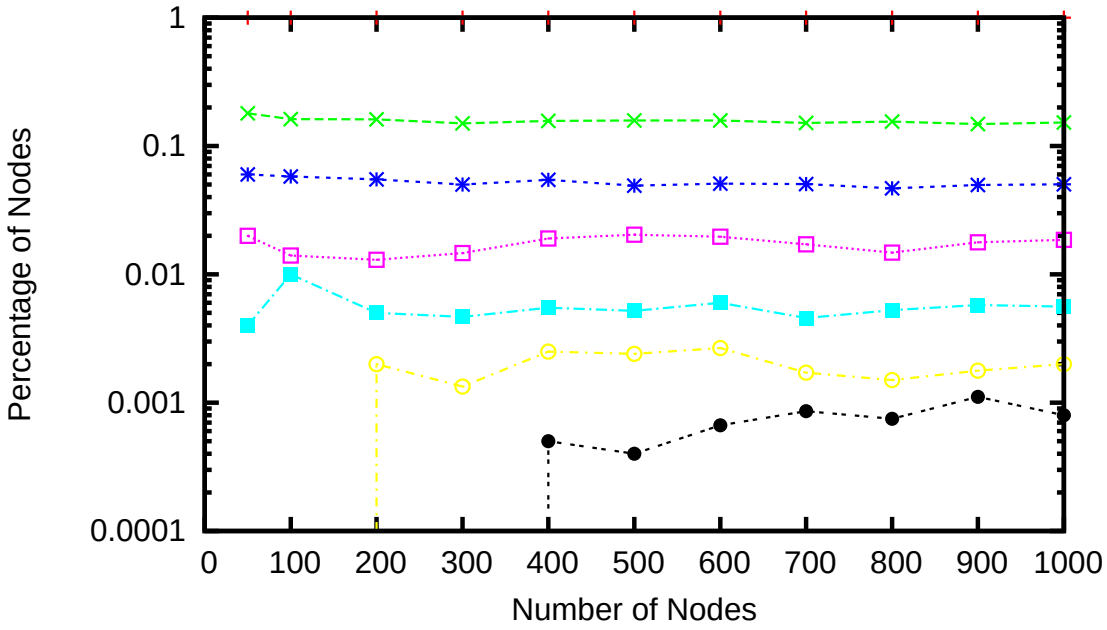



**Figure 5.7** The number of nodes that are drums as the network size is increased (shown in log scale)

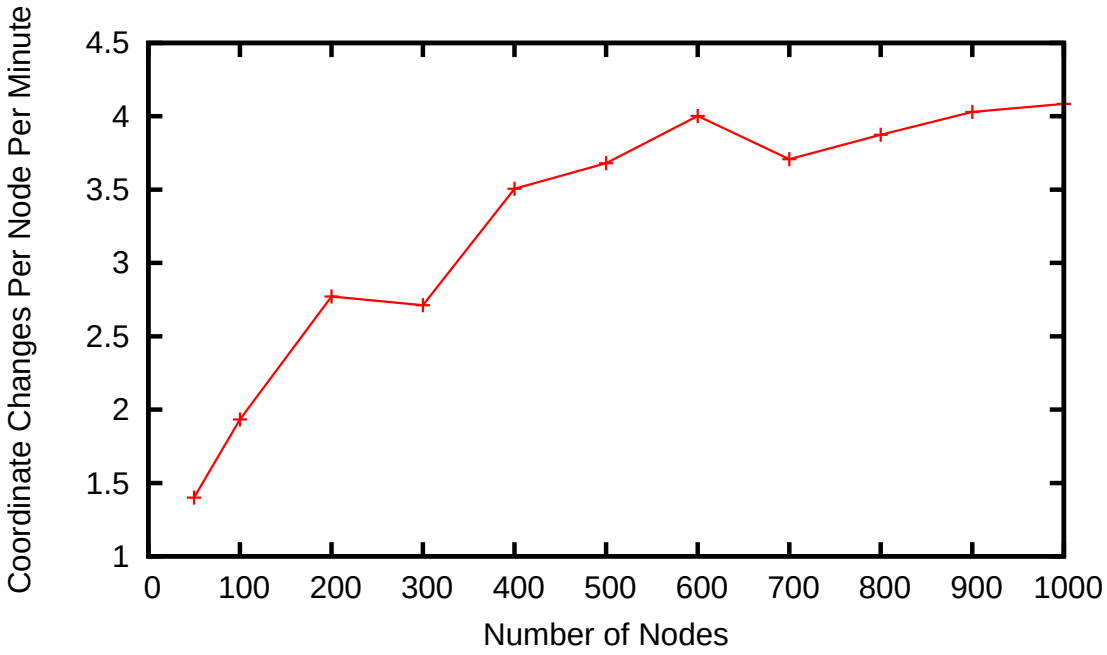



**Figure 5.8** The average number of coordinate changes as the network size increases

The coordinate structure formed by the drum protocol is used for routing. It is important that this structure is relatively stable in order to successfully locate nodes. When coordinates changes, it takes some time for updated routes to be inserted into the routing table. During this time period, a node may be unreachable. Figure 5.8 the number of coordinate changes per node is shown. Each node changes coordinates many times throughout the simulations. In a 1000 node network, a coordinate changes, on average, about once every 15 seconds.

Figure 5.9 shows that as the number of mobile nodes in the system increases the number of coordinate changes also increases. Figure 5.10 shows as the speed of the nodes in the system increases the number of coordinate changes goes up as well.

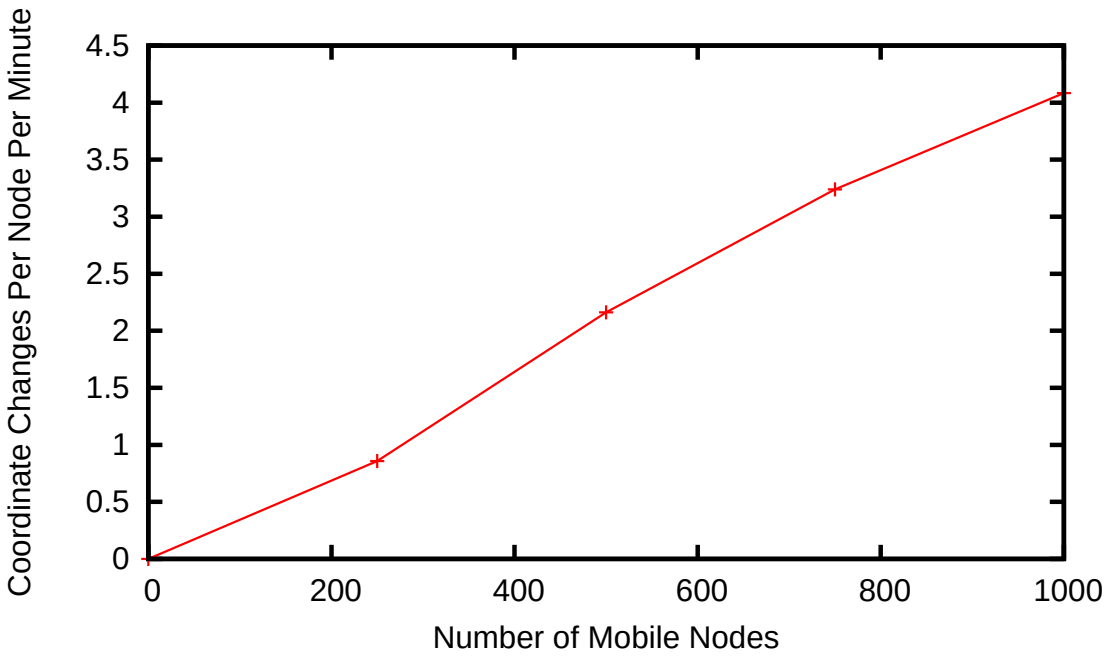



**Figure 5.9** The average number of coordinate changes in a 1000 node network as the number of mobile nodes is varied

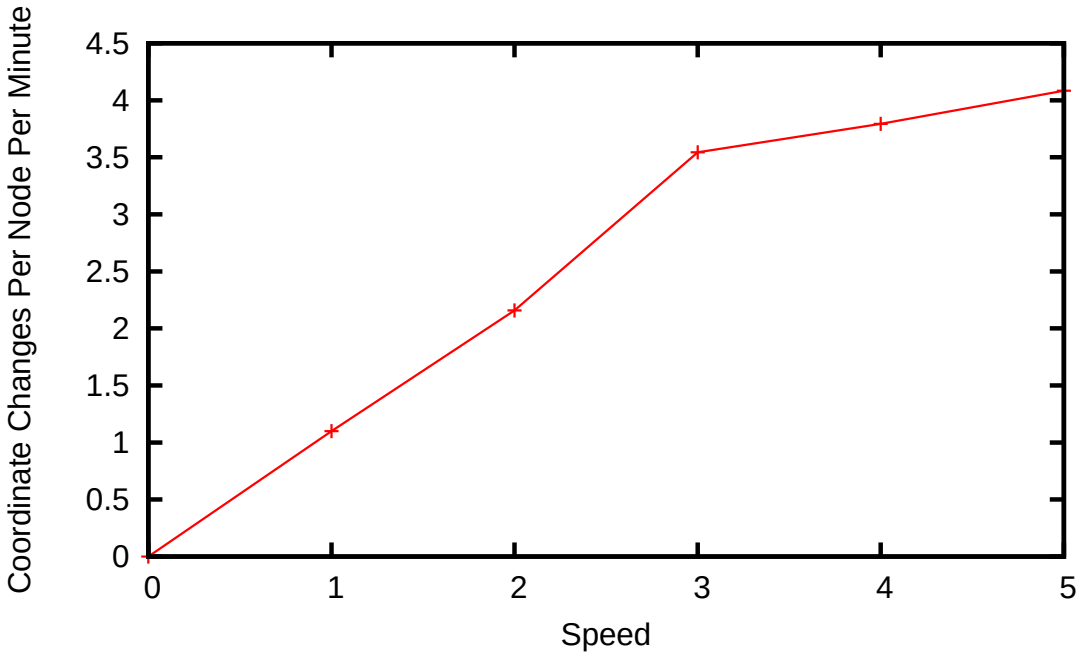



**Figure 5.10** The average number of coordinate changes as the maximum speed is varied

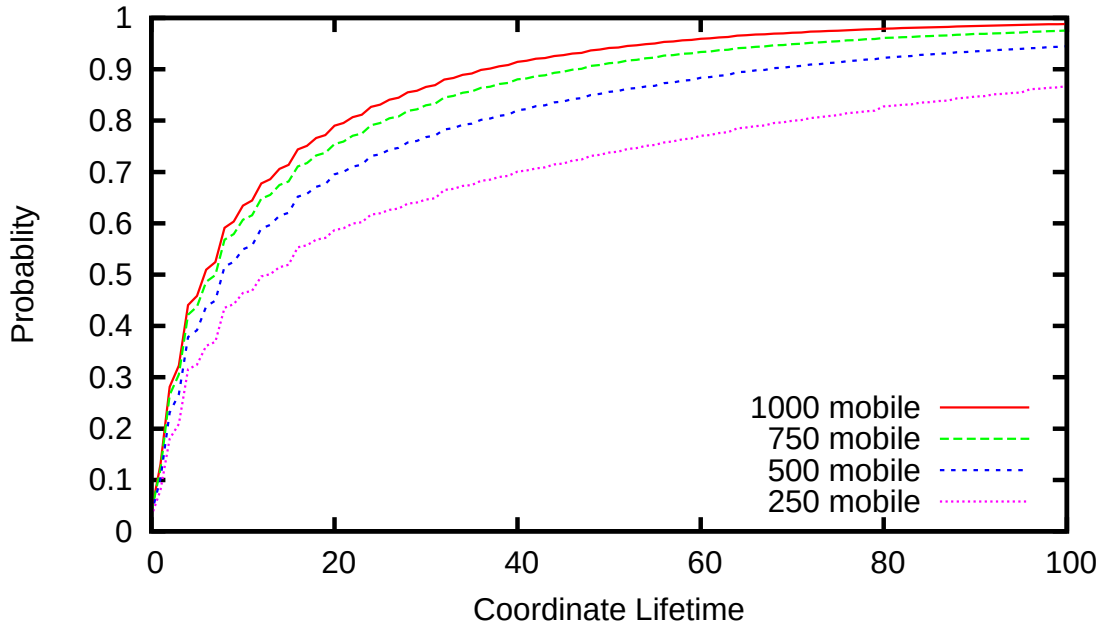



**Figure 5.11** CDF of of coordinate lifetime

To get a better understanding of how often a node's coordinate changes, we examined the results from a 1000 node simulation where all mobile nodes are moving at a maximum of 5 $m/s$. The CDF shown in Figure 5.11 shows the average coordinate lifetimes for a varying number of mobile nodes. When all nodes are mobile, the majority of coordinates are stable for 10 seconds or less while a few are stable for as long as 80 seconds.

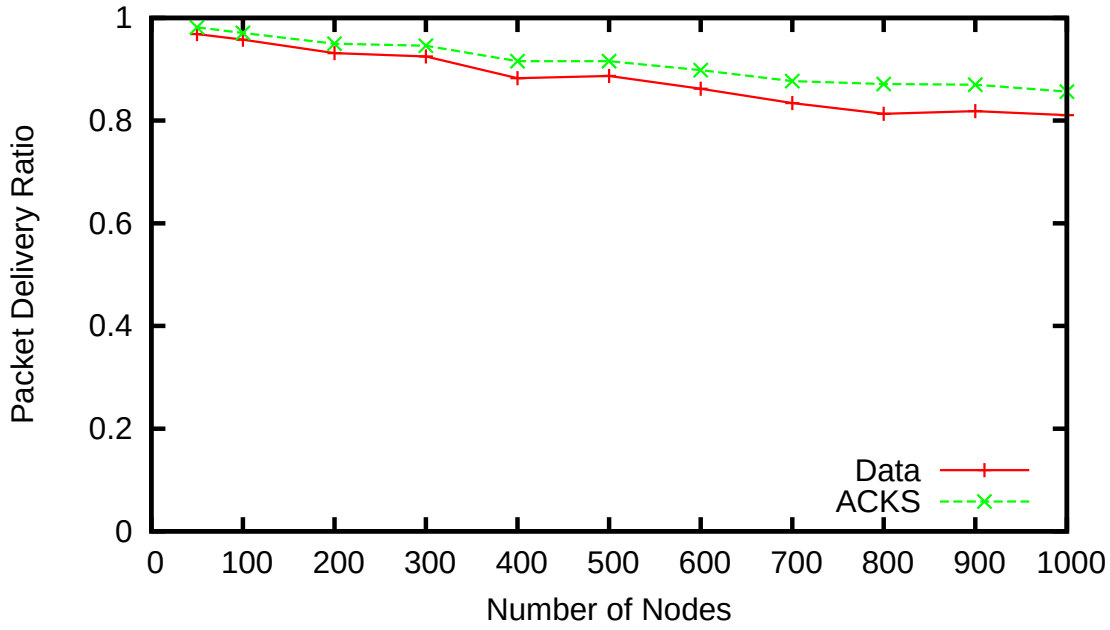



**Figure 5.12** The routing performance of the basic protocol

## 5.3 Routing

In this section, we evaluate the performance of routing in Strata. Unless otherwise stated, each experiment is run using the same parameters as the previous section. The routing scenario is as follows: 50 randomly selected nodes choose a random destination to send to, this random destination is changed every 50 seconds. A new packet is sent 4 times per second, and when a packet is successfully received an acknowledgment is returned. Routing begins at 350s when the network has stabilized. In this scenario all nodes are mobile and moving with a maximum speed of $5m/s$

Figure 5.12 shows the performance of the basic routing protocol as the size of the network increases. As expected, it drops as the number of nodes increases and the

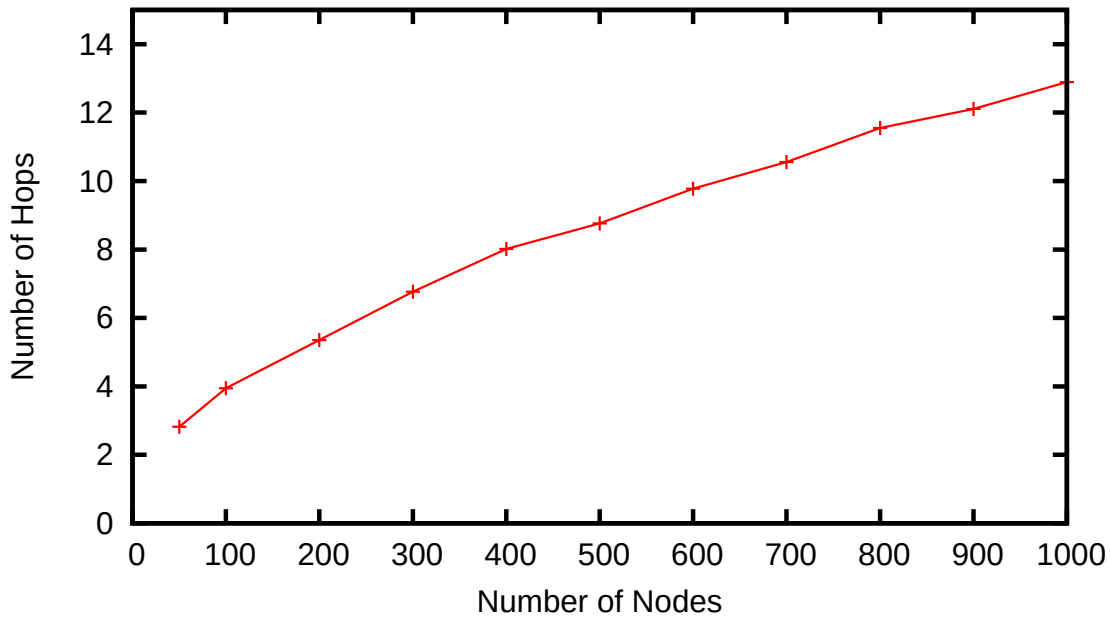



**Figure 5.13** The average number of hops to deliver a message in the basic protocol

length of the routes become greater. The line labeled "Data" indicates the performance of forward packets in the flow, while the line labeled "ACKS" measures the delivery rate of responses to the forward packets. The delivery ratio for acknowledgments is higher because they are triggered by a successfully packet delivery, thus a path between the nodes is more likely to exist.

In Figure 5.13 the number of hops to deliver a packet is shown. As expected, since the node density is kept constant and the distance between two randomly selected nodes will be greater, path lengths grow as the number of nodes increase.

In Figure 5.14 the distribution of path lengths is shown for a 1000-node network where all nodes are mobile. Most packets take between 10 and 15 hops, with a few

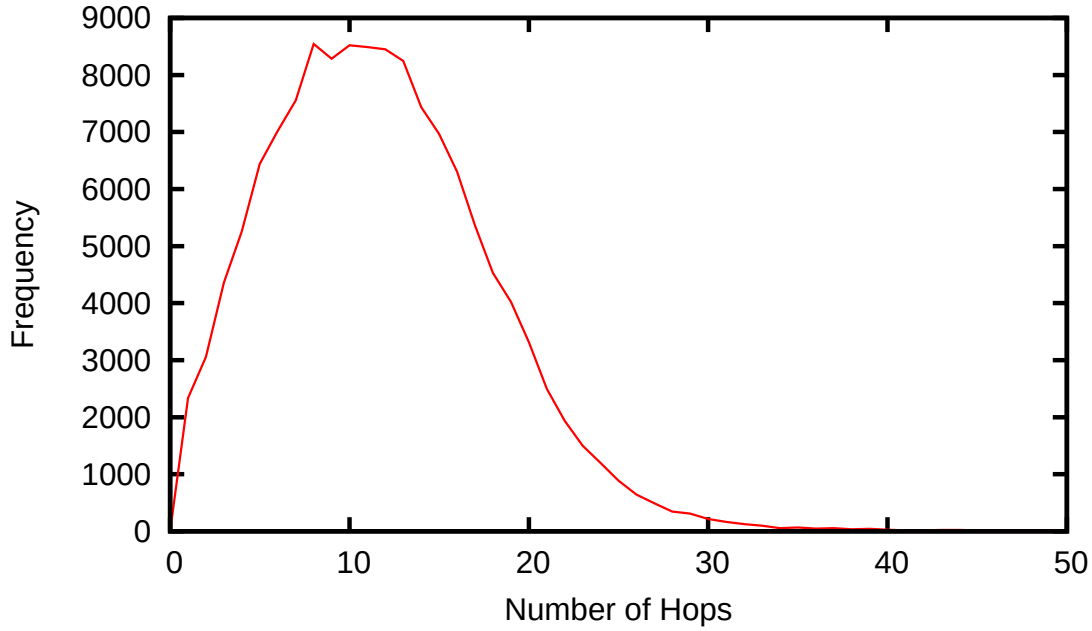



**Figure 5.14** The distribution of path lengths

taking up to 40 hops. This is the expected distribution, given that destinations are chosen randomly. Sometimes a close by node is chosen and it only take a few hops, while other times a very remote node is chosen and it will take many hops. Most times, an intermediate distant node is chosen and the hop count is close to the average.

# Chapter 6
# Optimizations

## 6.1 Drum Protocol

The basic Strata drum protocol has two main deficiencies: (1) it has a relatively high static overhead, and (2) it may result in unstable coordinates. The greater performance Strata can achieve in these areas the better a building block for services it will be. To improve these two characteristics of Strata we introduce two optimizations to the basic protocol described in Chapter 3.

### 6.1.1 Decreasing Overhead

In order to reduce overhead, a simple optimization is introduced. Since most overhead is caused by the beacons level 1 drums emitted, we simply turn off the emission of these drums. Enough routing information can be gleaned from overhearing neighbors forwarding beacons that the majority of packets can be delivered without level 1 drums broadcasting beacons. There are situations where a node has the same level 2 parent as a destination but has no a way to route to it. In these cases we can leverage that we have chosen our $d_2$ to be one hop, and just forward the packet to the level 2 drum. If the destination remains associated with the level 2 drum, it will then be correctly delivered. If not, then the cost incurred was just one additional hop. A

side effect of this optimization is that level 2 drums become responsible for routing slightly more packets, which sacrifices some load balance.

### 6.1.2 Increasing Coordinate Stability

The coordinate structure provides the means for routing between nodes. The longer a coordinate remains valid and routable the more likely a packet will reach its final destination. Since routing state is only updated periodically, changes to the coordinate structure will only be reflected upon beacon floods. In the time period between coordinate change and update some destinations may be unreachable. If coordinates change very frequently it makes routing very difficult, as the routing state at each node can not keep up with the dynamics of the network.

To improve coordinate stability, we have developed a heuristic to identify more stable nodes. In this environment, stable means not moving or moving slowly relative to the rest of the network. These nodes are then selected to be higher level drums in the system. Since these nodes are likely to remain in the same place the overall stability of the hierarchy is improved.

Intuitively, the optimization works by examining the neighbors a node sees over time. If the set of neighbors remains constant, then it is likely the node is stationary, or that it is moving with a group. If the neighbors are constantly changing then the node is likely to be moving. In our optimization, nodes that see the same neighbors gain weight, that is they consider themselves to be heavier or stationary. These nodes

will be good higher level drums, because the nodes around them stay the same. If the heavier node becomes a drum, the neighboring nodes will not have to change their coordinate, since they are not moving relative to the heavy node. A node that is moving will see churn it its neighbor set and will be considered light, this node is a bad candidate to become a high level drum, if it is chosen it is very likely that the nodes that associate with it will have to choose a new parent soon as the light weight drum continues moving on its path.

A local algorithm identifies heavy nodes. The algorithm counts the number of neighbors that remain constant over a small time period. For every neighbor, a count of the number of generations that it has remained a neighbor is kept up to a maximum limit. When a neighbor leaves, its count is divided by a reduction factor $\alpha$, for each time period that it is away, until it approaches zero and is removed from the list. A node keeps a value known as a weight, which is computed by combining the value for every neighbor. A node with a higher weight is considered more stable than one with a lower weight.

A node that has a small number of neighbors but is stationary will have a higher weight then one that is quickly moving through a dense region of the network. Since scores are decreased exponentially, the weight from the past is very quickly removed. The mobile node will only be able to gain a very low score from each node it is surrounded by. However, a stationary node can gain a lot of weight from each neighbor

it has. As long as the number of neighbors a mobile node has is not greater than the maximum per node score, the algorithm will favor stationary nodes in low density regions over mobile nodes in dense regions.

The weight value is used in the following two ways. The stepping up process is altered so that each node waits a small period after deciding to step up. The period is determined by the weight: the higher the weight, the shorter the waiting period. If the waiting period expires and no node has stepped up then the node that has decided to step up proceeds as normal. If the period expires and someone has stepped up then the node goes back to the normal step up algorithm. In this way the step up procedure is biased towards nodes with higher weight.

The second way the weight value is used is to determine which should step down when two drums of the same level get close together. In this case the node that has the lower weight will step down and the higher weighted one will remain. This results in a more stable system than if the drum to step down was chosen randomly.

## 6.2 Routing Protocol

The basic Strata routing presented in Chapter 4 performs adequately, but it leaves room for improvement. We augment our scheme to include several optimizations that improve routing performance under mobility. By including these optimizations, we were able to significantly increase the number of packets that can be successfully routed to their destination.

### 6.2.1 Aggressive State Updates

Maintaining a single entry for each destination in the routing table is fine in a stationary network; broken routing paths do not occur because nodes never change their location, and thus routes will remain valid forever. In order to handle mobility better, the basic scheme is altered. The first change is to store multiple entries for each routing table entry. This allows an alternate path to be used when a node's primary next hop fails. The multiple entries are kept by remembering beacons other than those with the shortest hop count. This increases the amount of state that must be stored and a policy must be used to decide which of the multiple next hops to use when routing.

Strata uses a scoring function that takes into account path length and the age of the entry. The scoring function adds the age of the entry to the number of hops to the destination, the lower the score the better the entry. The goal of doing this is to balance the fact that an old entry is more likely to be stale, and the shortest path is preferable.

To attempt to achieve the best possible routing performance, each node makes use of additional information available to it. In addition to the beacon broadcasts, a node also overhears data packets being routed through it or in the area within its reception range. These packets can be thought of as beacons emitted from the source of the data packet. They can be used to update the routing table to include nodes

that would not ordinarily be heard because their beacons have a limited range. Data packets from drums can also be used to update the routing table in the time period between beacon.

Data packets are treated in the exact same manner as beacons and are included in the routing table in the same way. They provide routing information to a specific node in the system, and allow routing state to be updated more frequently than relying on beacons alone. Often, data packets are sent repeatedly between the same two nodes, reinforcing the paths between them. This allows a path that is successfully delivering data to stay in the routing tables of intermediate nodes.

### 6.2.2 Routing Around Failures

In addition to keeping track of alternate paths, we must have a mechanism to update the routing table when one of the paths is found to be faulty. Once a faulty path is found, the goal is to route around the problem in order to prevent a packet from being dropped. In order to discover that a packet has not been successfully sent we assume the presence of, and access to, MAC layer acknowledgments. If we do not receive a MAC layer acknowledgment after a certain amount of time, we can assume that a packet has been lost.

If a packet is unable to be routed to the next hop, then the next hop is considered to be unreachable. All entries that depend on this next hop are removed from the routing table to avoid using a next hop that is unreachable. The node then chooses

another routing table entry for the destination (if it has one) and then uses it to continue routing. If the node does not have another entry for the destination, then the packet is dropped.

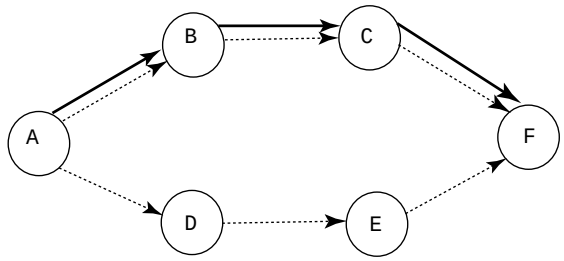
A further optimization, called packet push-back, is to not drop a packet if a node does not have any valid routing table entries. If a packet reaches such a node, then it is routed backwards one hop. The node that receives this packet updates it routing table by removing all entries toward the packet's destination that include the node that returned the packet. Then the node attempts to re-route the packet towards its destination as normal. This process is repeated until the packet returns to its source, reaches its final destination, or exceeds the maximum hop count.

The packet push-back optimization has two complementary effects on the routing: it allows a packet that would ordinarily be dropped to possibly be delivered, and it clears stale routing table entries from nodes along a path. While this may cause a packet to take a long path, it delivers some packets that otherwise would be dropped. In order to keep packets from taking arbitrarily long paths, each packet maintains a hop count that is incremented at each hop. Once a packet exceeds the maximum hop count, defined to be 200 in Strata, it is dropped.

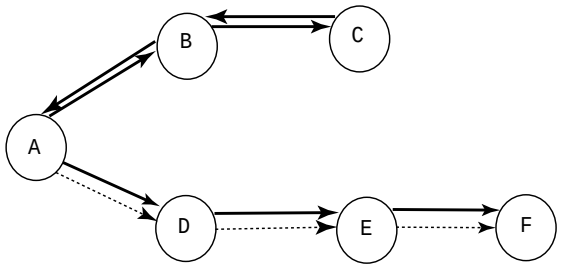
Furthermore, packet push-back removes routing table entries that are no longer valid in a proactive manner rather than simply allowing them to expire. This allows one packet to travel through the network and remove many stale entries instead

having a number of packets reaching a dead end until a single routing table entry expires. A packet taking a long path will have the benefit of clearing stale entries from many nodes.

Figure 6.1 shows an example of routing using packet push-back. In this figure, solid lines indicate the path that data travels, and dashed lines the routes each node has to node F. In Figure 6.1(a), node A is attempting to route to node F. Node A has two possible choices for routing to node F: a packet can either be sent to B or to D. Initially A chooses to route using B and is able to get the packet to the destination as shown by the solid lines. In Figure 6.1(b), F has moved so it is longer reachable by node C. A attempts to route a packet, it arrives at C and an error occurs. According to our protocol, the packet is sent back one hop and routed again with the route to C removed. B has no routing table entries and returns the packet to A. A has an alternate routing table entry using D and successfully delivers the packet to F. This shows how alternate routes can be used when an entry stored in the routing table becomes stale due to mobility.




(a)




(b)

**Figure 6.1** An example of routing around a failure

# Chapter 7
# Optimization Evaluation

## 7.1 Simulator

### 7.1.1 Simulation Implementation

The Strata simulator, StrataSim, is the same as that described in Chapter 5. In order to show that the optimized Strata protocol functions well even when network congestion is present, we modified the Strata simulator to be congestion aware. The details of these modification and the experiments using the congestion aware version of the simulator are presented in Section 7.5.

### 7.1.2 Methodology

In order to perform the experiments below, the following procedures and parameters are used unless otherwise stated. For all experiments we use a node density of $50/km^2$; for experiments of more than 50 nodes, we scale the dimensions appropriately. We use a constant radio range of 250m. In experiments where nodes are mobile, we use the random waypoint model. The parameters used in the random waypoint model [13] are a maximum node speed of $5m/s$ and 0 pause time. Each simulation is run for 1250 simulated seconds: the first 350 seconds are used to allow the network to boot up and results are reported from the final 900 seconds. Each data point on a graph is from an average of five runs.

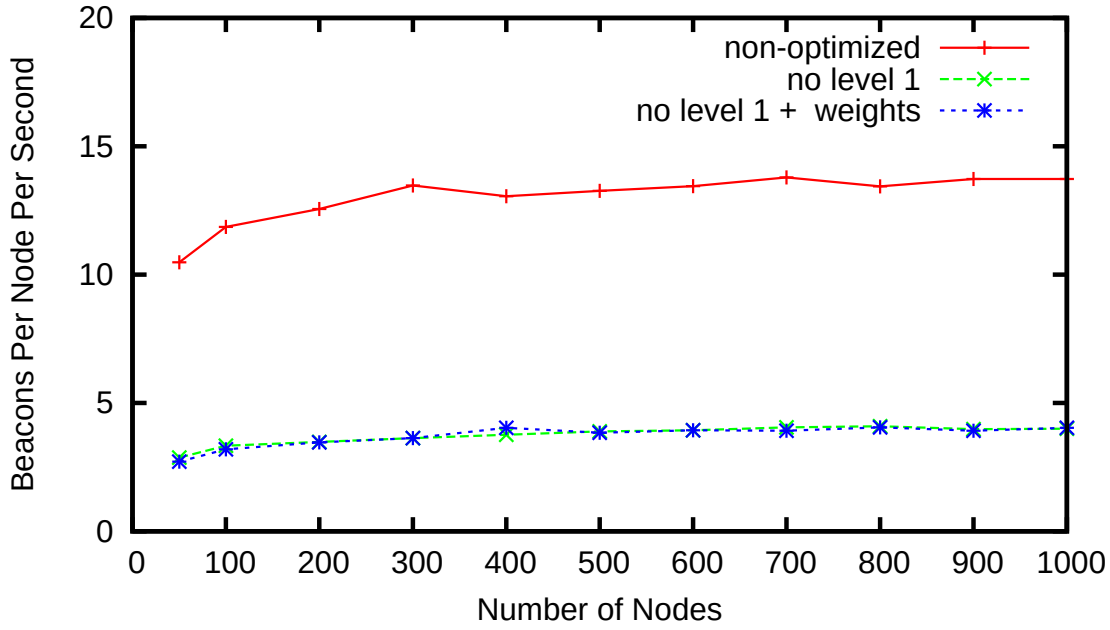



**Figure 7.1** Comparison of overhead in optimized protocol to basic protocol

## 7.2 Optimized Drum Protocol

In this section, we evaluate the benefits of the optimizations described in the previous chapter. We show that both provide significant improvements.

### 7.2.1 Overhead

Figure 7.1 shows that eliminating the level 1 drums significantly reduces the overhead of the beacons in the system. The number of beacons transmitted is reduced by about 75%.

The breakdown of beacons from other levels does not change when the level 1 beacons are removed. Figure 7.2 shows that the overhead from beacons at other levels remain the same.

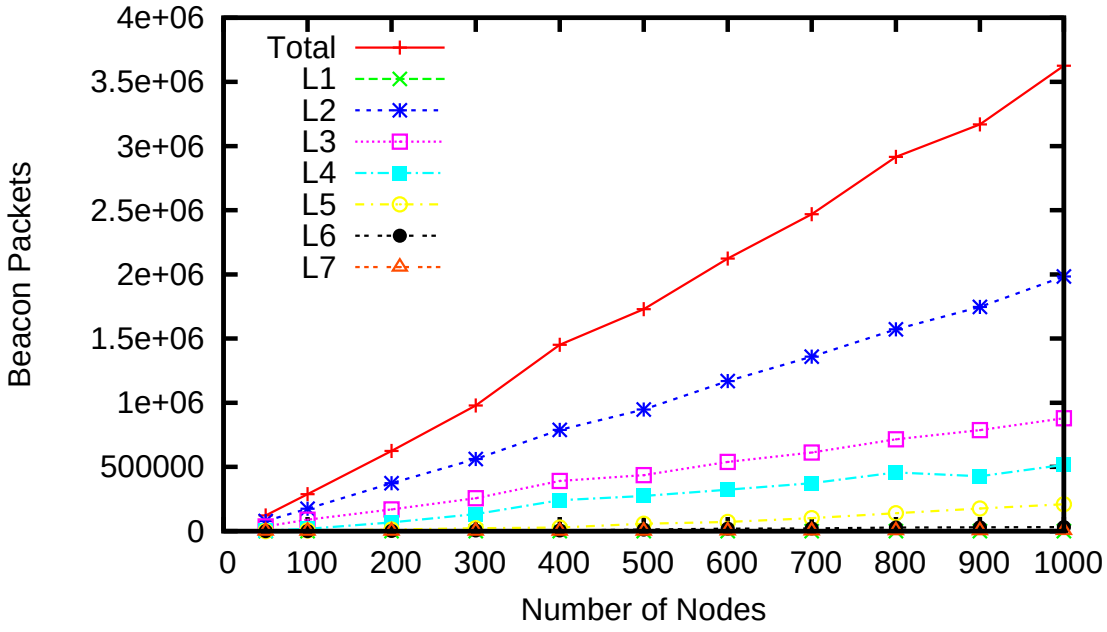



**Figure 7.2** Breakdown of drum overhead by level using no level 1 optimization

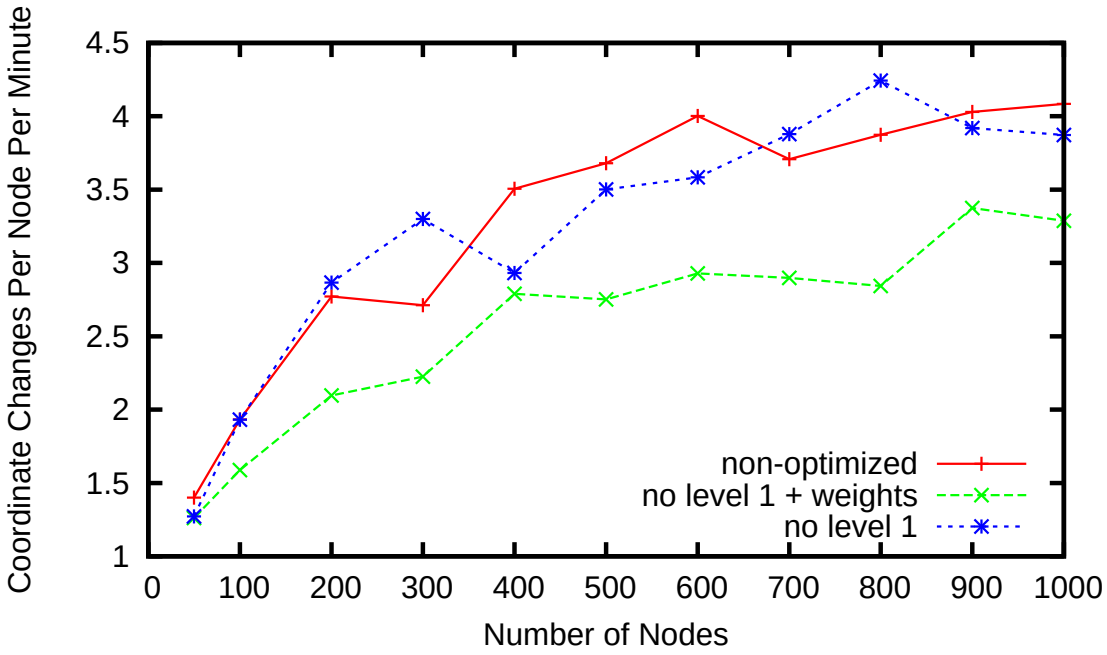



**Figure 7.3** Comparison of coordinate stability using different optimizations as the network size changes

### 7.2.2 Coordinate Stability

Figure 7.3 shows a comparison of the coordinate stability using the different optimizations. The removal of level 1 beacons does not affect the amount of coordinate changes, as the optimization was not targeted at reducing coordinate changes. The addition of the weight optimization strictly improved the number of coordinate changes in the network. All experiments in this section use 2 as the value for $\alpha$, the weight reduction parameter.

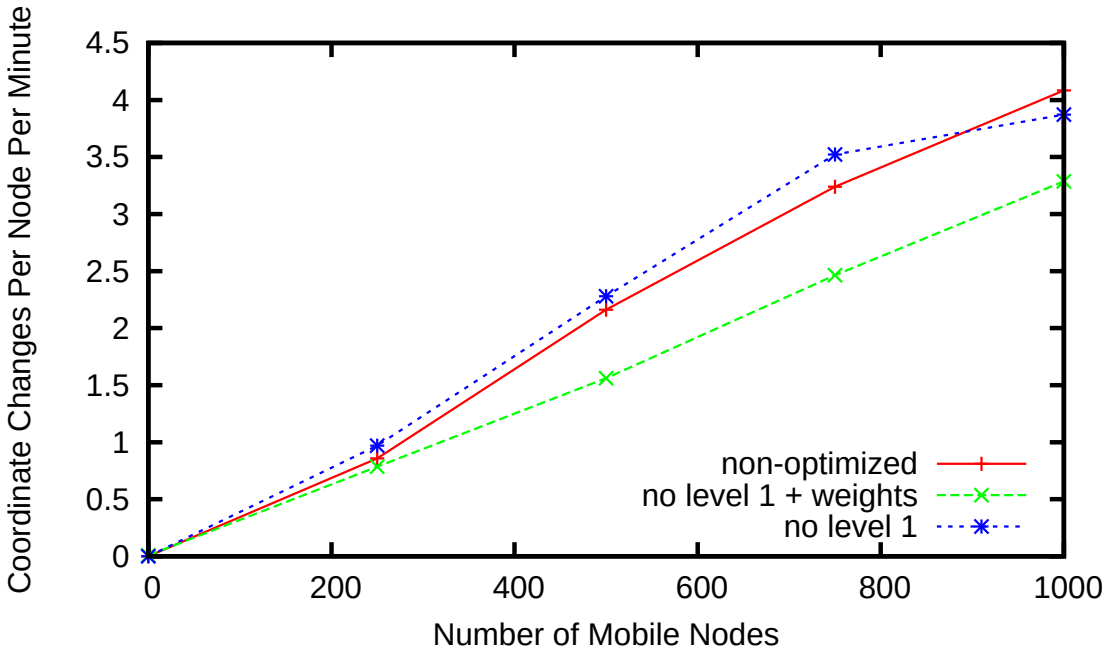



**Figure 7.4** Comparison of coordinate stability using different optimizations as the number of mobile nodes is varied

Again in Figure 7.4, the addition of the weights significantly improved the stability of the coordinates as the number of mobile nodes changed. There is only a modest improvement when there are 250 nodes, as the network is already fairly stable.

The weight optimization should increase the stability of the coordinates for each node in the system. Figure 7.5 shows that the average lifetime of coordinates with and without weights in a 1000 node network where the number of mobile nodes is varied.

The goal of using weights is to place the responsibility of being a drum on the more stable nodes in the network. To measure this, we calculated the percentage of drums that were stationary in a 1000 node network with 500 mobile nodes. Figure 7.6

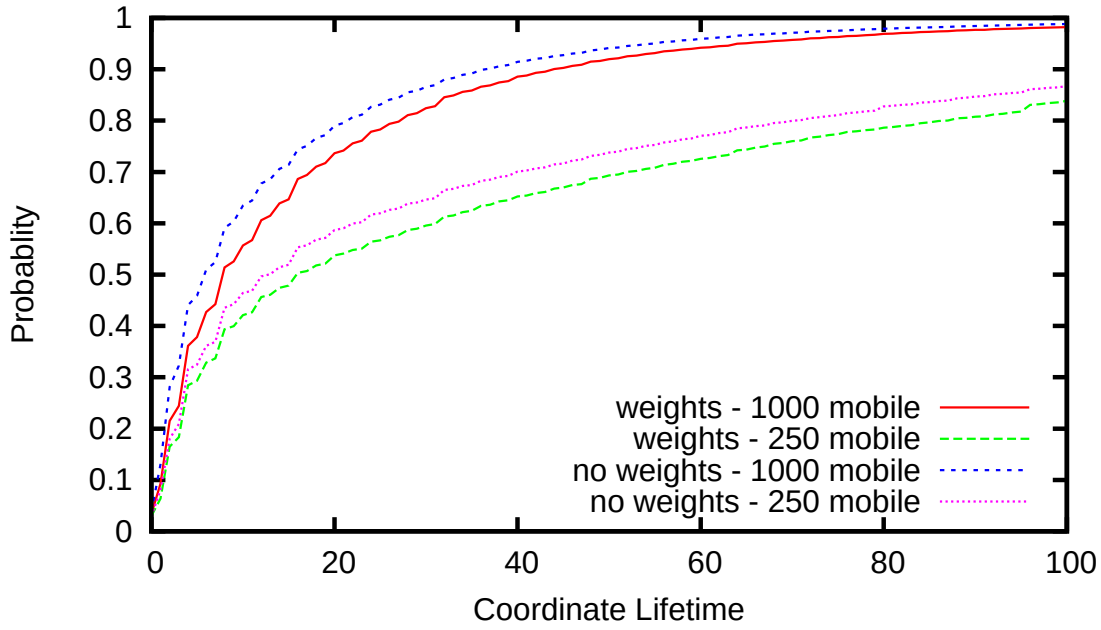



**Figure 7.5** CDF of coordinate lifetimes using weights

shows this measure as compared to the expected value of 0.5 which would result from randomly picked drums. A high number of stationary nodes are selected to be drums. At the highest level a stationary node was a drum 90% of the time.

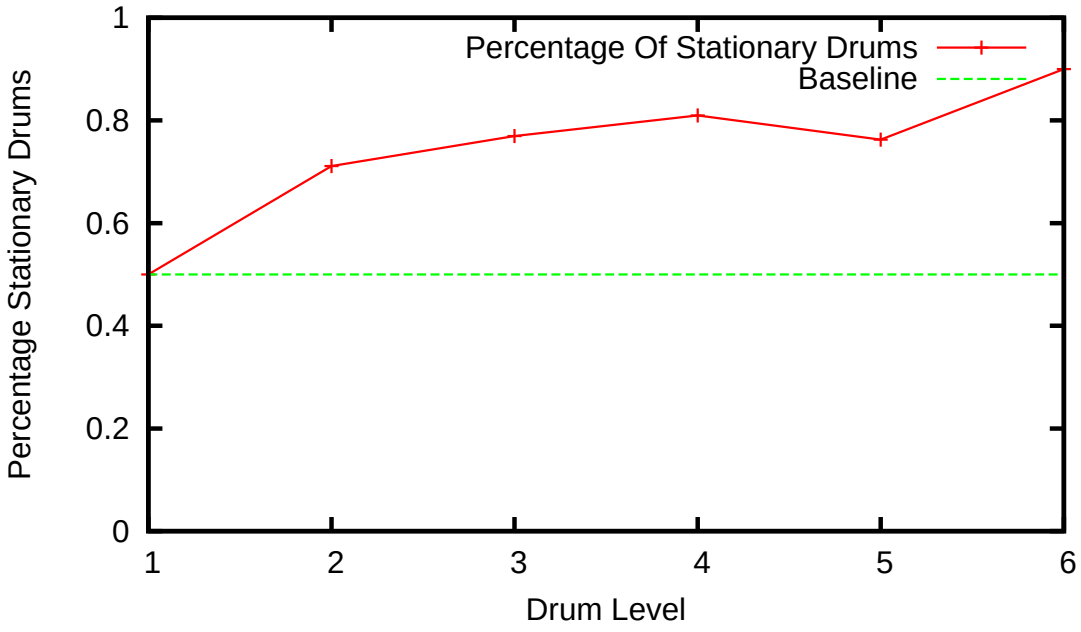



**Figure 7.6** The percentage of drums that are stationary in a 1000 node network

### Parameter Optimization

Above we show that the weight optimization is effective in making coordinates more stable. The main parameter in this optimization is $\alpha$, the rate at which the weight is decreased when a node is no longer seen as your neighbor.

The second design choice to consider is the function used to combine component scores into the weight value. In scenarios with an approximately uniform distribution of nodes, the difference between using summation and average to calculate weight is negligible. All experiments presented in this paper use a random placement of nodes, resulting in a uniform distribution over area, so summation and averaging produce identical results. In simulations where node distribution is highly non-uniform, experiments have shown that average outperforms summation. Since average is either equivalent or better, it is the preferred function for calculating weight.

Figure 7.7 shows the sensitivity of the optimization to different values of $\alpha$. The results are equivalent regardless of the speed in which the weight is decreased. Division by 2 is sufficient to achieve the desired results. In average is used to calculate the weight in this experiment.

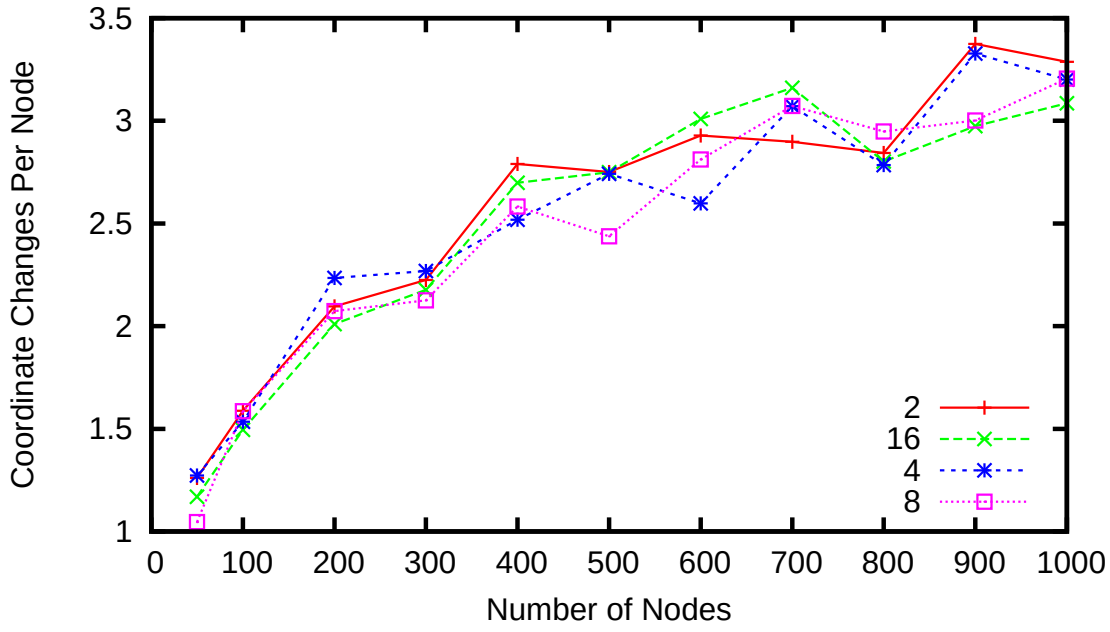



**Figure 7.7** Comparison of coordinate stability using different values of $\alpha$ and the average function as the network size changes

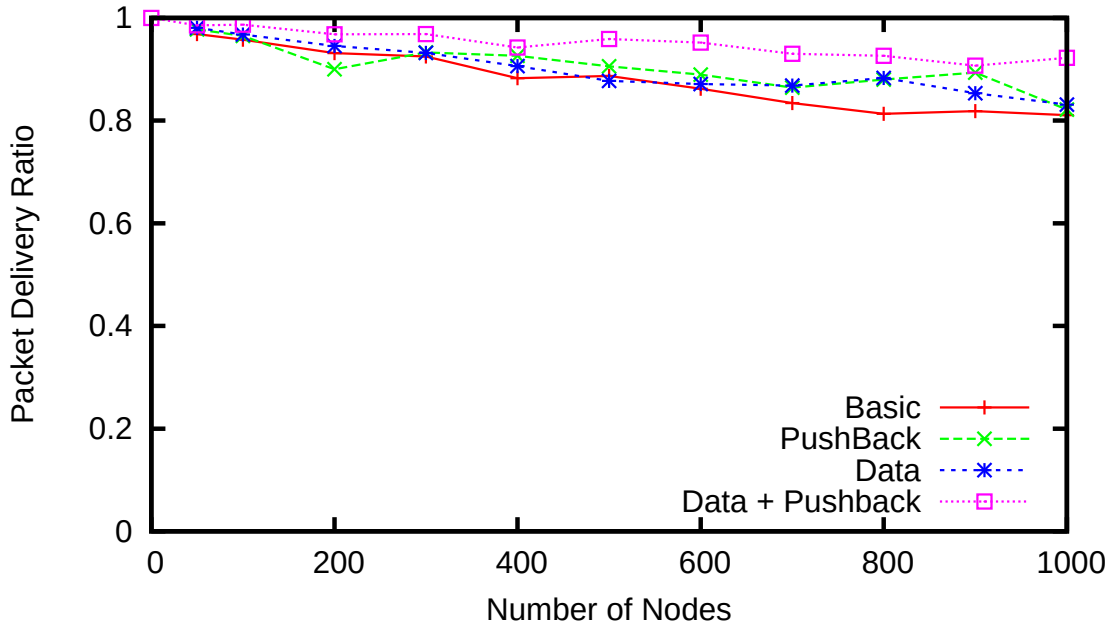



**Figure 7.8** The data delivery ratio achieved for different size networks comparing optimizations

## 7.3 Routing Optimizations

In this section, we evaluate the performance of routing using the optimizations introduced in Section 6.2. Unless otherwise stated each experiment is run using the same default parameters as described in the previous section. The routing scenario used is the following: 50 randomly selected nodes choose a random destination to send to, this random destination is changed every 50 seconds. A new packet is sent 4 times per second, and when a packet is successfully received, an acknowledgment is returned. Routing begins at 350s when the network has stabilized. As before, each data point on a graph reflects the average of five runs.

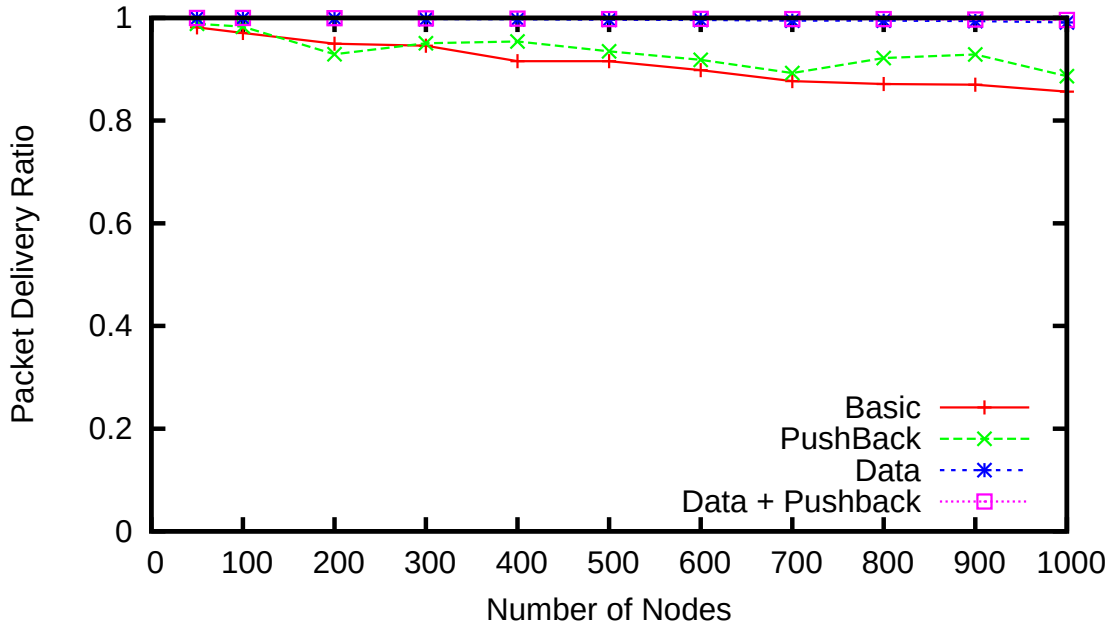



**Figure 7.9** The acknowledgment delivery ratio achieved for different size networks comparing optimizations

We evaluated the packet delivery ratio of the different optimized versions of the routing protocol. Figure 7.8 shows the routing performance of each optimization. With both optimizations enabled the routing performance is the highest.

In Figure 7.9, we show the delivery ratio for acknowledgment packets. Adding the data as drums optimization as described in Section 6.2.1 significantly improved the packet delivery ratio. When this optimization is enabled, the number of acknowledgment packets successfully routed approaches 100%. Because the route back to the originator of the data packet is cached, it is unlikely to break in the time it takes for the acknowledgment to return.

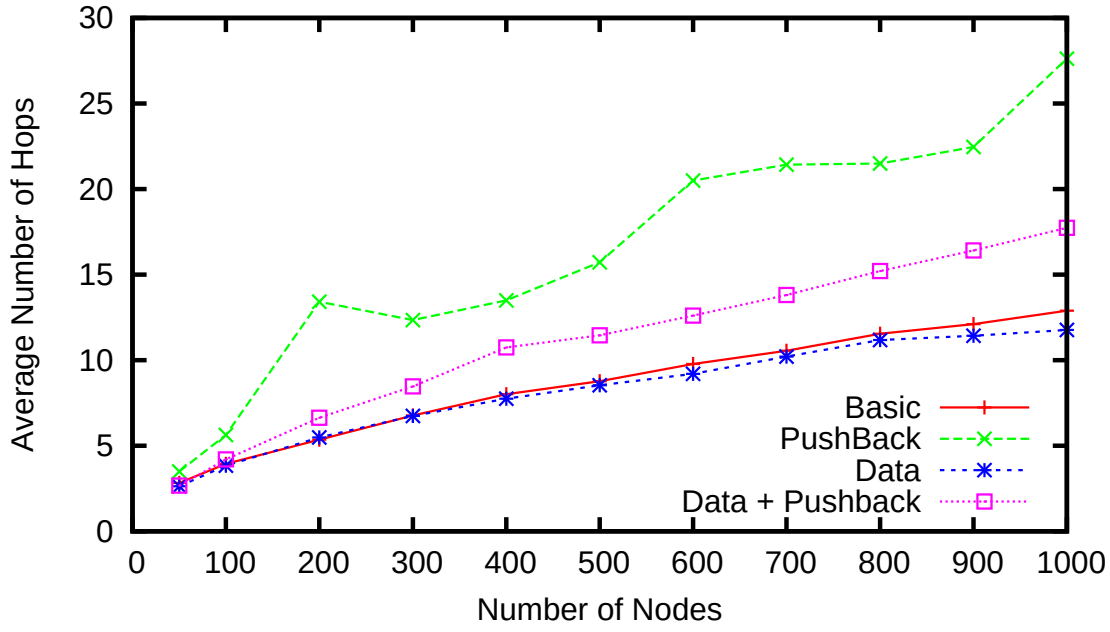



**Figure 7.10** The average number of hops to deliver a message comparing optimizations

Figure 7.10 shows the average path length for a packet to be delivered. The addition of the push-back optimization can increase the average hop count significantly. In combination with the data as beacons optimization, the effect is somewhat reduced, as there are more possible valid routes.

The distribution of hop counts for each version of the protocol is shown in Figure 7.11. They are all roughly normal distributions centered around 11. The packet push-back optimization causes some packets to travel very long routes attempting to find a route to a destination. These few packets that travel over 100 hops, increase the average hop count when the push-back optimization is enabled. This is caused

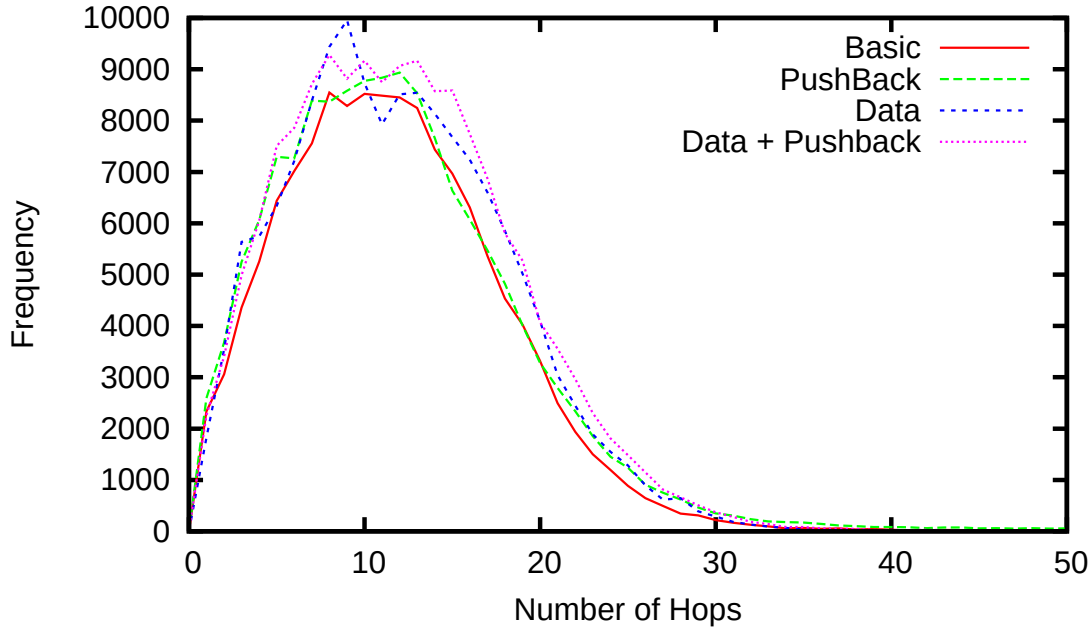



**Figure 7.11** The distribution of path lengths to deliver a message comparing optimizations

by some packets being pushed back many times and then redirected in a forward direction until finally the destination is reached.

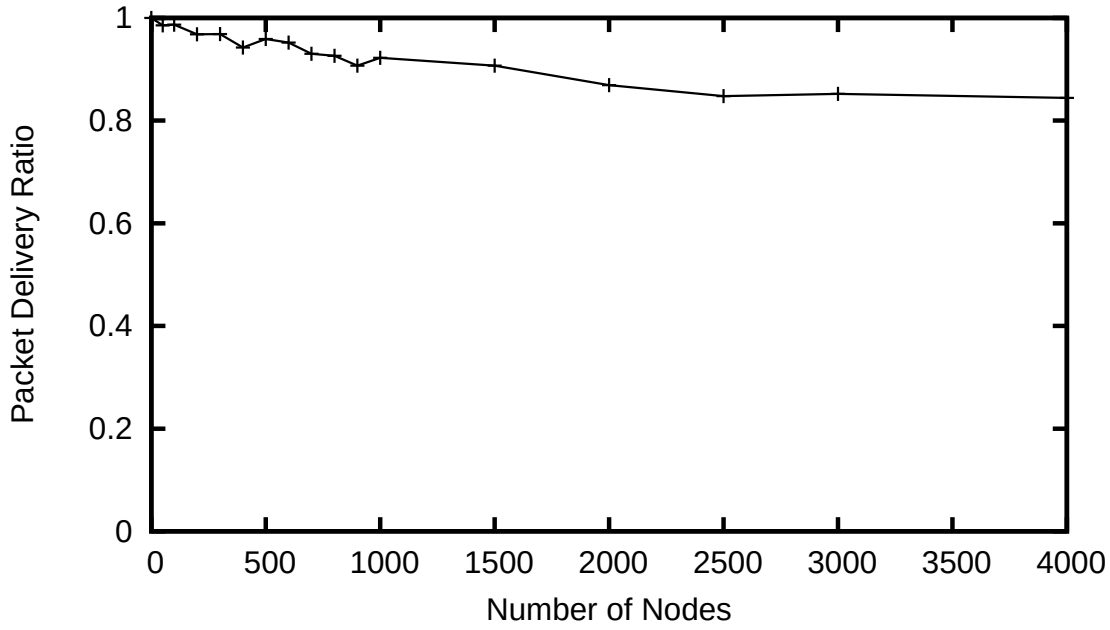



**Figure 7.12** The packet delivery ratio achieved for different size networks

## 7.4 Scalability Evaluation

In this section, we evaluate the scalability of the proposed routing protocol. The version of the routing protocol that includes all the optimizations is used for these experiments. The results show that our optimized current implementation provides scalability, which is sufficient for a large ad hoc scenario such as disaster relief. Our design degrades at a moderate pace as the number of nodes increases. The routing scenario is the same as in the previous section and all nodes are mobile and moving at a maximum speed of $5m/s$.

Figure 7.12 shows the packet delivery ratio as the number of nodes in the network is varied. It decreases as the number of nodes is increased. The decrease in packet

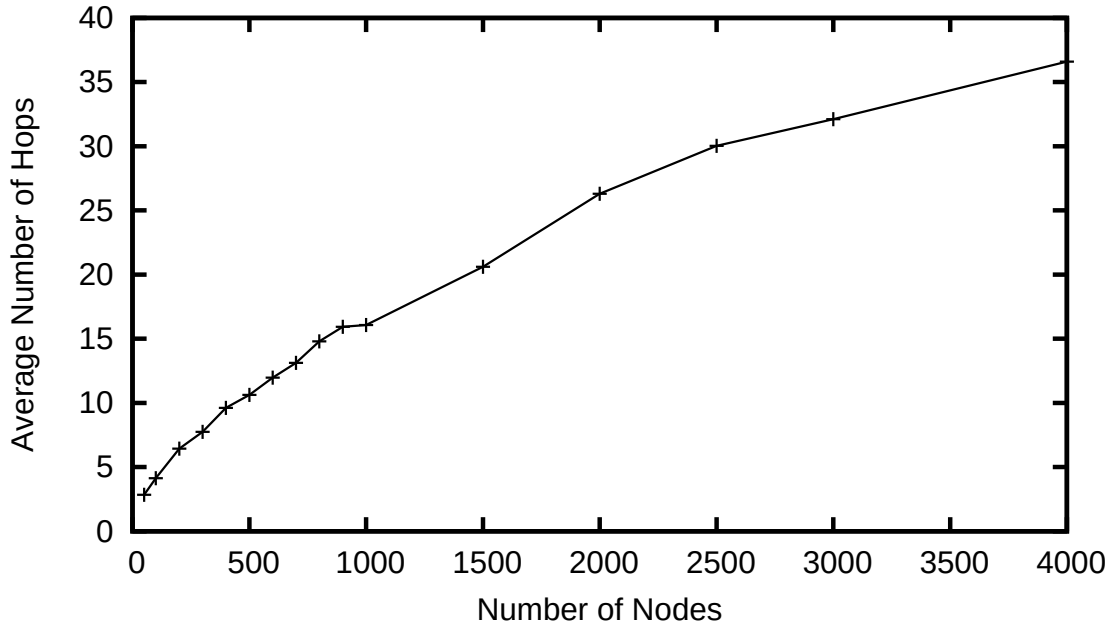



**Figure 7.13** The average number of hops to deliver a packet as network size grows

delivery ratio is expected, since packets must take longer paths and thus are more likely to be lost. Although there is a drop, the packet delivery ratio degrades gracefully, even in networks as large as four thousand nodes. This result shows that our routing protocol scales well as the number of nodes in the network increases.

The average number of hops a packet takes is shown in Figure 7.13. The number of hops required scales with the size of the network. It is a known result that the diameter of the network scales with square root of the number of nodes [36]. The results in Figure 7.13 agrees with this at the scales we have evaluated.

## 7.5 Congestion Evaluation

StrataSim was designed to be fast and simple. To achieve this, we abstracted away many of the details of the MAC and physical layers. Up until this point, congestion was not modeled at all. As the network becomes larger, it is more likely that congestion may become a problem. In order to determine if our experiments were causing network congestion, a MAC layer that simulated congestion was implemented in the Strata simulator. In order to simulate congestion, we chose a 1 Mbps bandwidth. We chose 1 Mbps instead of 11 Mbps because at 11 Mbps, congestion was rarely observed. A bandwidth of 1 Mbps allowed us to see some of the effects of congestion. If the protocol performs well under this constrained bandwidth, it will perform well given the more standard 11 Mbps bandwidth. The scenario used is the same as the experiments in Section 7.3 and uses the version of the protocol that includes all optimizations.

The simulator was altered to take into account the size of the data being sent in order to determine how long each node must transmit. If one node is transmitting then no node within radio range will transmit; instead they will back off and queue their messages. The queue size is infinite and we measure the maximum number of messages queued in order to determine of congestion is a problem. This is a simple and approximate modeling of congestion, but it will provide enough information to determine if congestion is a problem in Strata.

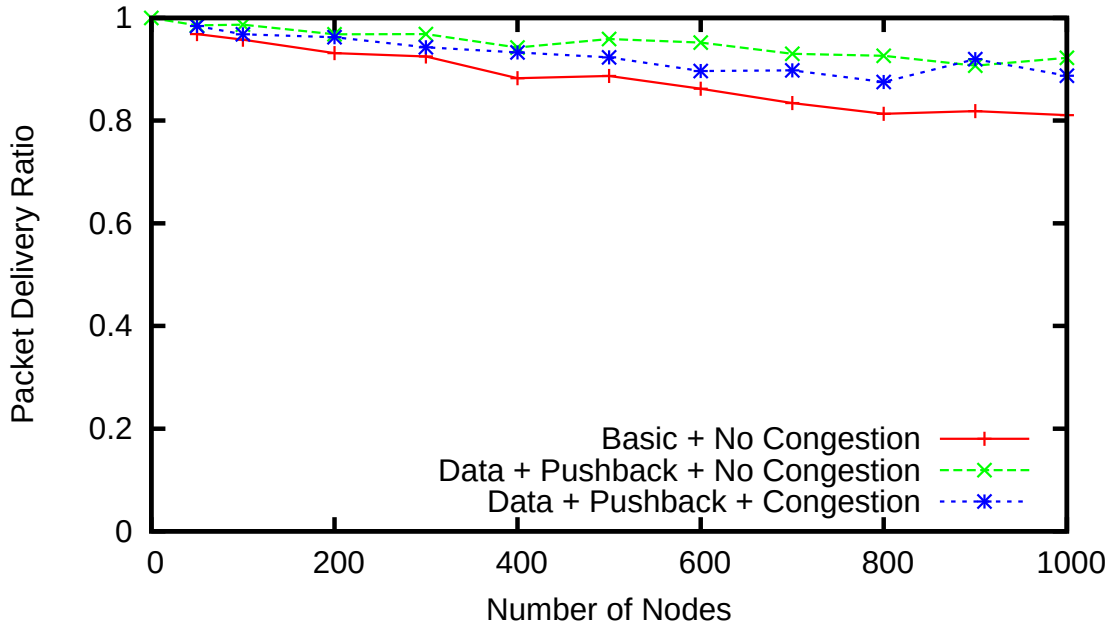



**Figure 7.14** PDR Comparison with and without congestion simulation

Figure 7.14 shows that the overall routing performance is similar whether congestion is simulated or not. There is a small difference at some sizes. This is caused by beacons being queued, while they are queued routing table entries will time out since they expect a beacon to be delivered on time. Packets relying on these entries are dropped if no alternate paths to the destination exist.

The maximum queue length observed in the network is shown in Figure 7.15. Although there is some queuing occurring, it was never permanent, indicating congestion was only temporary. Figure 7.16 shows the distribution of maximum queue lengths in a 1000 node network. Most nodes never had more than 5 packets enqueued, and

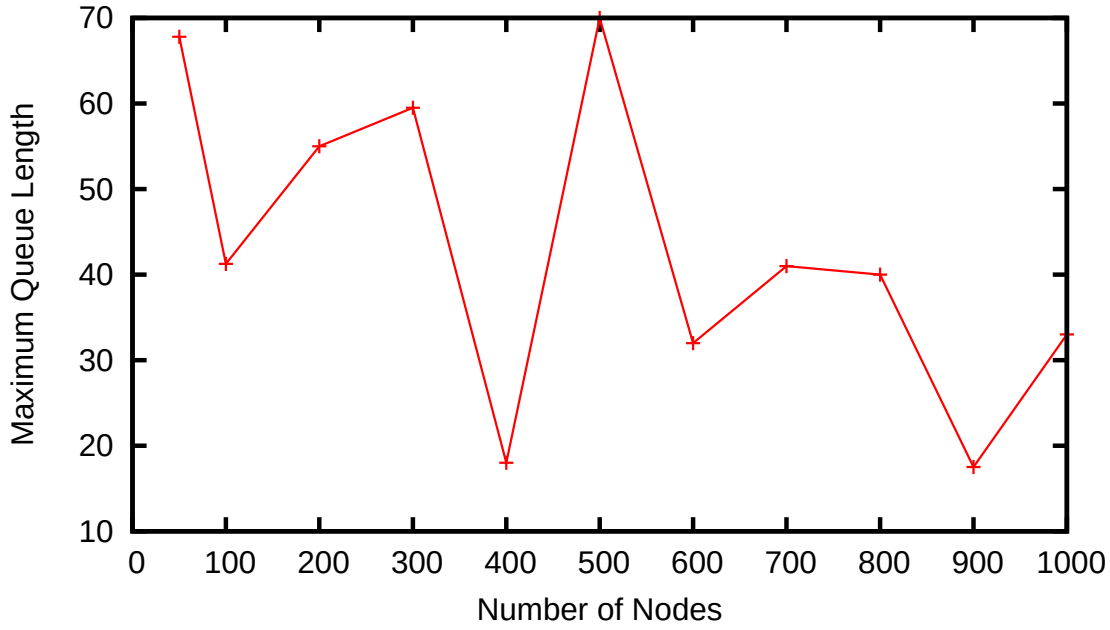



**Figure 7.15** Maximum Queue Length with congestion enabled

only two nodes filled their queue past 15 packets. There is one node that had a queue that contained 32 messages, corresponding to the maximum shown in Figure 7.15.

The final congestion experiment was to measure how often each node observed the broadcast channel around itself to be active. Figure 7.17 shows the results. As the network size increases, the average utilization decreases. The maximum also decreases as the size of the network increases.

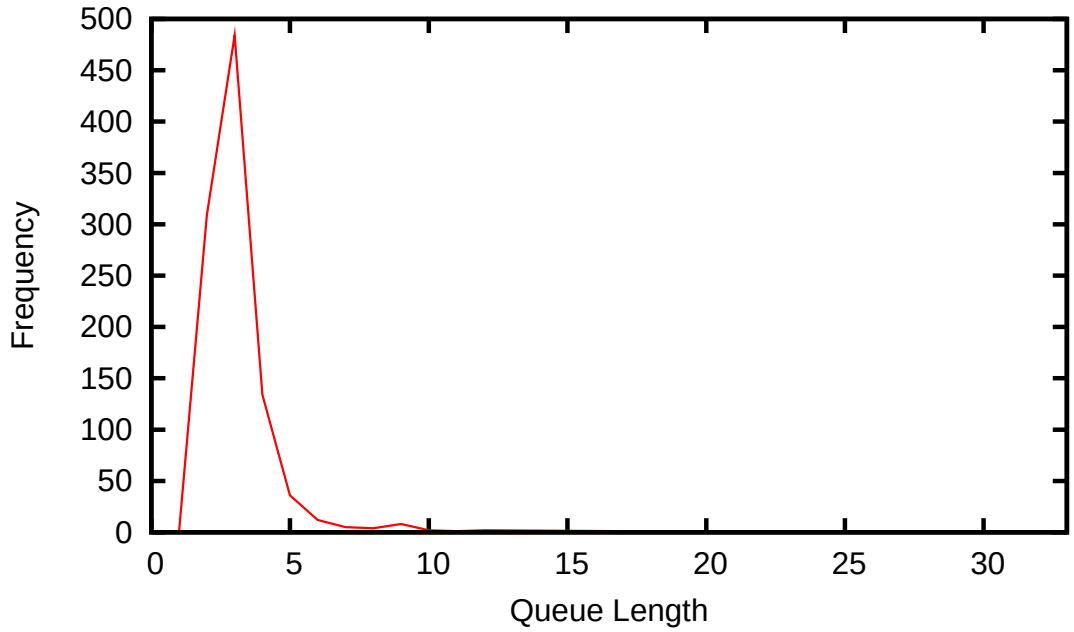



**Figure 7.16** Distribution of maximum queue length at every node in a 1000 node network

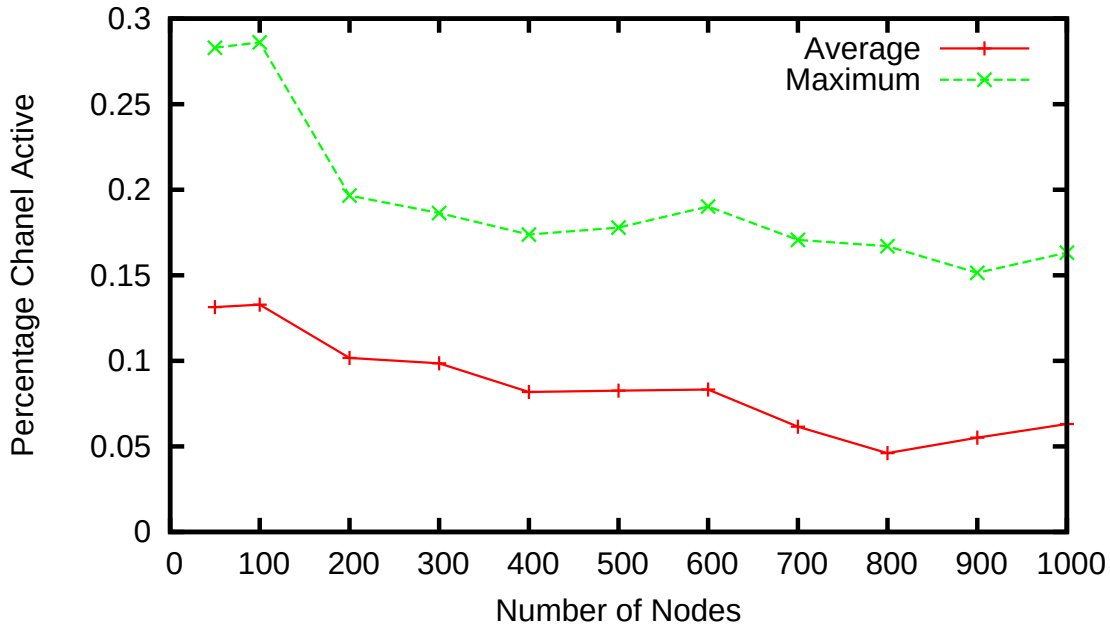



**Figure 7.17** The average observed channel activity by a node

# Chapter 8
# Future Work

Currently, the size of our simulations is limited to a size of 4000 nodes or less. This limitation is mainly due to the increasing memory footprint as the number of nodes increase. Given a machine with more addressable memory, this limitation could easily be removed. It is then expected that the simulation will scale to a larger number of nodes. Increasing processor efficiency, while not currently the bottleneck, would also prove helpful as it would decrease the amount of time individual simulations would take and allow greater efficiency in running large number of experiments.

A comparison between Strata, Masai, and L+ is planned. This will allow direct comparison of the protocols based on landmarks and identify the advantages and disadvantages of each. Strata will be implemented in ns-2, or Masai and L+ will be ported to StrataSim, to allow direct comparison of experiment results using the same simulation environment.

A key based routing primitive that allows routing to the node closest to a co-ordinate, is an abstraction that is included in the Safari architecture. Developing this primitive in Strata has been challenging, and providing consistent, low-overhead indirect routing primitive with high delivery ratio will be studied as future work.

Developing a large scale decentralized lookup service to map node identifiers to current coordinates is in progress. Solving this problem requires a functioning indirect routing primitive, which as described above, remains a challenge.

# Chapter 9
# Conclusions

Multi-hop ad hoc networking provides the potential for communication when traditional means of networking are unavailable. Conventional MANET protocols fail to scale to more than several hundred nodes and thus are unsuitable for some classic deployment scenarios, such as large scale disaster recovery. The Safari project aims to address this problem by scaling to several thousand nodes.

This thesis presented Strata, an implementation of the Safari architecture. The basic protocol is first developed with the aim of simplicity. It is evaluated and the design space is explored. Several key optimizations are identified which reduce the overhead and improve the delivery ratio. These optimizations are simulated and shown to significantly improve the performance of the protocol. Simulations show that the optimized version of Strata scales well to several thousand mobile nodes. Strata is suitable for the networking requirements of large scale deployments such as disaster recovery and battlefield scenarios.

# References


1. Official bluetooth wireless info site. http://www.bluetooth.com.

2. SAFARI home page. http://safari.rice.edu.

3. Terminode. http://www.terminodes.org.

4. *ns* notes and documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, Nov. 1997. Available from http://www-mash.cs.berkeley.edu/ns/.

5. N. Abramson. The aloha system another alternative for computer communications. In *Fall Joint Computer Conference, AFIPS Conference Proceedings*, volume 37, pages 281–285, 1970.

6. R. Bagrodia and M. Gerla. A modular and scalable simulation tool for large wireless networks. In *Proceedings of Performance Tools '98*, Palma de Mallorca, Spain, Sept. 1998.

7. L. Bajaj, M. Takai, R. Ahuja, R. Bagrodia, and M. Gerla. Glomosim: A scalable network simulation environment. Technical Report 990027, University of California, Los Angeles, 1999.

8. S. Banerjee and S. Khuller. A clustering scheme for hierarchical control in multihop wireless networks. In *Proceedings of IEEE INFOCOM 2001*, Anchorage, AK, Apr. 2001.

9. C. E. P. E. M. Belding-Royer and I. Chakeres. Ad Hoc On Demand Distance Vector (AODV) routing. IETF Internet Draft (Work in Progress), 2003 Oct.

10. V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A media access protocol for wireless LAN's. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 212–225, London, UK, Aug. 1994. ACM.

11. L. Blazevic, J.-Y. L. Boudec, and S. Giordano. A scalable routing scheme for self-organized terminode network. In *Communication Networks and Distributed systems modelling and Simulation conference (CNDS)*, San Antonio, TX, Jan. 2002.

12. L. Blazevic, S. Giordano, and J.-Y. L. Boudec. Anchored path discovery in terminode routing. In *Networking*, pages 141–153, 2002.

13. J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th International Conference on Mobile Computing and Networking (MobiCom'98)*, Dallas, TX, Oct 1998.

14. M. Chatterjee, S. Das, and D. Turgut. An on-demand weighted clustering algorithm (WCA) for ad hoc networks. In *Proceedings of IEEE Globecom*, San Francisco, CA, Nov. 2000.

15. M. Chatterjee, S. Das, and D. Turgut. WCA: A weighted clustering algorithm for mobile ad hoc networks. *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks)*, 5:193–204, Apr. 2002.

16. B. Chen and R. Morris. L+:scalable landmark routing and address lookup for multi-hop wireless network. Technical Report MIT-LCS-TR-837, Laboratory for Computer Science Massechusetts Institute for Technology, 2002.

17. S. Du. Routing in large-scale ad hoc networks based on a self-organizing coordinate system. Master's thesis, Rice University, 2004.

18. R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *Proceedings of 2nd Symposium on Networked Systems Design and Implementation*, Boston, MA, 2005.

19. J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. In *Symposium on Computational Geometry*, pages 188–196, Medford, MA, 2001.

20. Z. J. Haas. A routing protocol for the reconfigurable wireless network. In *IEEE 6th International Conference on Universal Person Communications Record*, volume 2, pages 562–566, Oct. 1997.

21. Z. J. Haas and M. R. Pearlman. The Zone Routing Protocol (ZRP) for ad hoc networks. Internet-draft, IETF MANET Working Group, November 1997.

22. Z. J. Haas and M. R. Pearlman. The performance of query control schemes for the zone routing protocol. In *Proceedings of the ACM SIGCOMM '98 conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 167–177, Vancouver, BC, Canada, Aug. 1998.

23. Z. J. Haas, M. R. Pearlman, and P. Samar. The Zone Routing Protocol (ZRP) for ad hoc networks. Internet-draft, IETF MANET Working Group, July 2002. Expiration: January, 2003.

24. J. Hubaux, J. L. Boudec, S. Giordano, and M. Hamdi. The terminode project: Toward mobile ad-hoc wans. In *In Proceedings of 6th IEEE International Workshop on Mobile Multimedia Communications*, San Diego, CA, Nov. 1999.

25. IEEE Computer Society LAN MAN Standards Committee. *Supplement to 802.11-1999,Wireless LAN MAC and PHY specifications: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band*. The Institute of Electrical and Electronics Engineers, New York, New York, 1997.

26. IEEE Computer Society LAN MAN Standards Committee. *Telecommunications and information exchange between systemsLocal and metropolitan area networksSpecific requirementsPart 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specificationsAmendment 4: Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band*. The Institute of Electrical and Electronics Engineers, New York, New York, 1997.

27. IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York, New York, 1997.

28. D. B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proceedings of the 1st IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, Santa Cruz, CA, Dec 1994.

29. D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996. Chapter 5.

30. J. Jubin and J. Tarnow. The darpa packet radio network protocols. In *Proceedings of the IEEE*, pages 21–32, Jan. 1987.

31. P. Karn. MACA — A new channel access method for packet radio. In *Proceedings of the Amateur Radio 9th Computer Networking Conference*, pages 134–140, London, ON, Canada, Sept. 1990.

32. P. Karn. The qualcomm cdma digital cellular system. In *Proceedings of 1993 USENIX Symposium on Mobile and Location-Independent Computing*, pages 35–40, Cambridge, MA, Aug. 1993.

33. B. Karp and H. Kung. GPSR: Greedy perimiter stateless routing for wireless networks. *Mobile Computing and Networking*, pages 243–254, 2000.

34. A. Khan. Analysis of SAFARI: An architecture for scalable ad hoc networking and services. Master's thesis, Rice University, 2004.

35. Y. B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in mobile ad hoc networks. In *Proceedings of the 4th International Conference on Mobile Computing and Networking (MobiCom'98)*, Dallas, TX, Oct. 1998.

36. J. Li, C. Blake, D. S. J. D. Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *Proceedings of the 7th International Conference on Mobile Computing and Networking (MobiCom'01)*, Rome, Italy, July 2001.

37. J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, Boston, MA, Aug. 2000.

38. C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 15(7):1265–1275, 1997.

39. A. McDonald and T. Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communication (JSAC)*, Aug. 1999.

40. M. Mouly and M.-B. Pautet. *The GSM System for Mobile Communications*. Telecom Publishing, 1992.

41. V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of IEEE INFOCOM 1997*, pages 1405–1413, Kobe, Japan, Apr. 1997.

42. G. Pei, M. Gerlan, and X. Hong. LANMAR: Landmark routing for large scale wireless ad hoc networks with group mobility. In *Proceedings of the 6th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2000)*, Stanford, CA, 2000.

43. C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, London, UK, Aug. 1994.

44. C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, Feb 1999.

45. V. Ramasubramanian, Z. J. Haas, and E. G. Sirer. SHARP: A hybrid adaptive routing protocol for mobile ad hoc networks. In *In Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing*, Annapolis, Maryland, Jun 2003.

46. T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, New Jersey, 1996.

47. S.Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proceedings of the 4th International Conference on Mobile Computing and Networking (MobiCom'98)*, Dallas, TX, Oct. 1998.

48. P. F. Tsuchiya. The landmark hierarchy: description and analysis. Technical Report Technical Report MTR-87W00152, MITRE corporation, June 1987.

49. P. F. Tsuchiya. Landmark routing: architecture, algorithms, and issues. Technical Report Technical Report MTR-87W00174, MITRE corporation, May 1988.

50. X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: A library for parallel simulation of large-scale wireless networks. In *Workshop on Parallel and Distributed Simulation*, pages 154–161, May 1998.

51. H. Zimmermann. OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, 28(4), Apr. 1980.