LOAD BALANCING IN MULTI-HOP WIRELESS AD HOC NETWORKS

Mohsen Eftekhari Hesari

A thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montréal, Québec, Canada

September 2010

Your file Votre référence
ISBN: 978-0-494-71082-1
Our file Notre référence
ISBN: 978-0-494-71082-1

Abstract

Load balancing in multi-hop wireless ad hoc networks

Mohsen Eftekhari Hesari

In this thesis we study the load distribution and load balancing problem in wireless ad hoc networks. Using a discrete unit disk graph model of the network, we analyze the distribution of load induced by greedy routing in the network with an all-to-all communication pattern between the nodes. We derive an estimate for average load of the nodes in the network. We also calculate the expected load of a node as a function of its geometric coordinates in the network. We express the actual load of a node in the network as a random variable and obtain the parameters of this random variable. Using this random variable we derive an estimate for the maximum load of the nodes in the network. Our result is more accurate than previous studies which were based on a continuous model of the network.

We analyze how different parameters of the network, i.e., number of nodes, transmission range, and different routing algorithms can affect the parameters of the load distribution. We give a technique to reduce the variance of the load distribution, and hence decrease the maximum load of the nodes in the network. Our technique can be combined with any location-based routing algorithm. We also introduce a class of algorithms that improve the maximum expected load of nodes in the network. Experimental results show that our algorithms outperform other existing algorithms in reducing the maximum load of the network.

Acknowledgments

I wish to express my gratitude to my supervisors, Prof. Narayanan and Prof. Opatrny. Their encouragement, guidance, and deep support during my study enabled me to develop an understanding of the subject. Their wide knowledge and detailed and constructive comments have been a great value for me.

I am grateful to all my student colleagues for their help and support. I would like to specially thank Mohammad Kiaei, Hossein Kassaei, Mona Mehrandish, Arash Azarfar, and Louisa Harutyunyan for their friendship and help.

I owe my best friend, Abolfazl Keighobadi and his wife Atefeh Adibi a lot for their continued moral support in the last two years.

Lastly, and most importantly, I wish to thank my parents and my sisters for their understanding, support and encouragement. They have lost a lot due to my studies abroad. To them I dedicate this thesis.

Contents

List of Figures

List of Tables

Table of Symbols and Notation

V	Set of nodes in the network
E	Set of links between nodes in the network
n	Total number of nodes in the network
R	Radius of the network
Tr	Transmission range of nodes
C_a	The circle with radius r and centered at the network origin
n_a	Average number of nodes in C_a
N_v	Set of neighbors of node v
src	Source node
$dest$	Destination node
ra	A routing algorithm
$load_{ra}(v)$	Load of node v with respect to the routing algorithm ra
$\|p(u,v)\|$	The hop length of the path $p(u,v)$
$\|\|u,v\|\|$	Euclidean distance between node u and node v
$\|\|\overline{u,v}\|\|$	Length of the projection of the line segment from u to v on the line connecting u to the destination node
$\overline{h}_{u,v}$	Length of the projection of the line segment from u to v on the line connecting u to the destination node
$\overline{h}$	Expected value of $\overline{h}_{u,v}$

X_i	Random variable corresponding to node v_i with respect to greedy routing
$\overline{\xi}_{d,\epsilon}$	Average load of nodes in an annulus with inner radius d and width ϵ centered at network origin
$\overline{\xi}_{d,0}$	Average load of nodes at distance d from the network origin
eml	Expected maximum load of the nodes in the network
mel	Maximum expected load of the nodes in the network

Chapter 1

Introduction

A communication network is an environment consisting of nodes and links that enables communication between applications running on the nodes. Every link in a communication network can be either wireless or wired (twisted pair wire, optical fiber, coaxial cable, etc.). A network might have basic components (i.e. switches, bridges, routers, access points, and etc.) to perform the communication between nodes. A router is a device that forwards the packets on behalf of other nodes to reach their intended destination(s) using the information provided in the packets and routing tables. An access point is a device that enables wireless nodes to connect to a wireless network (also called *managed* or *infrastructure* wireless network). In a managed wireless network, every wireless node connects to the network (and other nodes) through an access point. Communication protocols are a set of algorithms that enable communications between nodes in the network.

In contrast with managed wireless networks, a *wireless ad hoc network* consists of wireless nodes communicating in an *infrastructureless* environment. A network is called ad hoc because there is no preexisting infrastructure (e.g. routers in wired networks and access points in managed wireless networks) to perform the routing of packets among the nodes.

Applications of wireless ad hoc networks are diverse, for example environment sensing and monitoring using small wireless sensor nodes, providing communication between wireless devices or even Internet access in the absence of wired infrastructure, connecting computers (mainly ruggedized laptops) in a battle field, and fast deployment of communication in disaster fields where communication infrastructures are destroyed or were never built before.

Since the transmission range of nodes is limited, a node can only communicate directly with those nodes that are within its transmission range. Hence if one node decides to communicate with other nodes in the network which are not within its transmission range, it has to send its packet through other nodes (forwarding nodes). Wireless ad hoc networks have some inherent characteristics that distinguish them from wired and managed wireless networks:

- *Decentralized environment:* As mentioned earlier wireless ad hoc networks are infrastructure free. This is in contrast with the traditional wired networks, that use routers to perform the routing and management of the network or managed wireless networks in which each wireless node communicates with an access point and access points perform routing. In a wireless ad hoc network, each node has the capability to and might participate in the routing of other nodes' packets.

- *Frequent changes in the network topology:* All nodes in a wireless ad hoc networks are free to join or quit the network at any time, nodes or the links between nodes might fail and new links can be established, furthermore nodes might be mobile (in mobile ad hoc networks). The routing tables should be updated frequently to adapt to the changes of the network topology.

- *Limited energy or computation capability of nodes:* Nodes in wireless ad hoc

networks are usually powered with batteries. Also each node has limited memory and computational capability, especially in case of wireless sensor networks. This should be considered in the design of the communication protocols for ad hoc networks.

- *Heterogeneity of nodes:* Some nodes might have more capabilities than other nodes (energy, memory, computational capability, communication range, etc.) in the network. Protocols for wireless ad hoc networks need to consider this to design better algorithms that improve performance using the heterogeneity of nodes.

- *Large scale deployment:* A wireless ad hoc network might consist of few nodes (e.g. in a home network) to hundreds of nodes (e.g. in a large sensor network for environment monitoring). Therefore the designed algorithms should be scalable to be appropriate for different kinds of deployments.

These characteristics and constraints of wireless ad hoc networks make protocols designed for wired and managed wireless networks unsuited for them and create new challenges and research directions.

1.1 Routing

The routing problem in wireless ad hoc networks has received extensive research interest in recent years. The routing problem is to find intermediate nodes that forward a packet from its source node to its destination node in the network. These intermediate nodes form a path between the source node and the destination node. A routing algorithm defines a path for each pair of nodes in the network area. A path $p(s, d)$ between nodes s and d is a sequence of nodes $p(s, d) = (a_0, a_1, ..., a_m)$ in which $a_0 = s$ and $a_m = d$, called the source and destination of the path, respectively,

where consecutive nodes of the sequence are neighbors. The length of a path $p(s,d)$ denoted by $|p(s,d)|$ is defined as the number of hops (edges) in the path.

In terms of path discovery initiation, routing protocols can be divided into *proactive* (also known as table-driven) and *reactive* (also known as on-demand) routing. Routing protocols can also be classified by considering the use of geographical information in the routing algorithms. In this sense, routing protocols can be divided into location-aware routing and location-unaware routing protocols [29].

1.1.1 Proactive routing protocols

In proactive routing, each node maintains the information required for forwarding the packets in routing tables. Routing tables are updated to adapt to the changes in the network topology. The routing tables are updated periodically or when a change in the network topology is sensed (e.g. DSDV [40], CGSR [30], and WRP [34]).

1.1.2 Reactive routing protocols

In reactive routing, route discovery initiation is done by the source node (hence it is also called source-initiated on-demand routing). Routes are created only when they are demanded, and a routing maintenance process is responsible for maintaining the routes. Routes are deleted when they are not desired anymore or the destination is not accessible (examples are AODV [41], and DSR [3, 21, 22]).

Our work is orthogonal to this classification of the routing protocols and the results can be applied to both *reactive* and *proactive* routing algorithms.

1.1.3 Location-aware routing protocols

In location-aware routing algorithms it is assumed that every node (or most of the nodes) is aware of its own coordinates (via GPS, location servers or using localization

techniques). Moreover each node is assumed to know the coordinates of its neighbors and for each packet the coordinates of its destination node [15].

As in [15], location-aware routing algorithms can be divided into subgroups. Here we review two of the subgroups:

Distance-based algorithms:

Algorithms in this group decide the next hop only based on the geographical coordinates of the destination node and neighbor nodes. We give some examples below:

- *Random Progress Method* [35]: In this method, the next hop is selected randomly (with equal probability) among the neighbors that have a positive progress toward the destination node .

- *Most Forward within Radius* [50]: In this scheme, the next hop is selected as the neighbor with the greatest progress toward the destination node.

- *Greedy algorithm* [11]: In this method, the neighbor with the minimum Euclidean distance to the destination is selected as the next hop. In a variant of Greedy scheme called *GEDIR* [48], the message is dropped when the decision at the current node is to forward back the packet to the node that it came from.

- *Compass routing* [26]: In this method, the direction of nodes are calculated using their coordinates. The next hop is the neighbor that has the closest direction to the direction of the destination node.

From here on when we use the term *greedy routing*, we mean the *GEDIR* variant of the Greedy algorithm.

Partial flooding with location information

The methods in this group use the location information of nodes to perform an intelligent flooding of the packets to the nodes that are closer to the destination than the current node. Providing location information to the routing algorithms allows for algorithms with fewer control packets than are needed in location-unaware routing algorithms (examples are LAR [26], DREAM [1]). See [15, 39, 47] for more examples of location-aware routing algorithms.

1.1.4 Location-unaware routing protocols

All other routing protocols that do not use geographical location information are categorized in this group (e.g. AODV [41], DSR [3, 21, 22], and DSDV [40]). These algorithms use flooding of the control packets to get information about the network topology and perform the routing. For more information and examples of location-unaware protocols see surveys [4, 44, 45].

In this thesis we focus on location-aware routing algorithms.

1.2 Load Balancing

The load of a node is defined as the number of packets it transmits in a given period of time. Since we assume the transmission ranges of the nodes are fixed, there is a high correlation between the load of a node and its energy consumption. Many of the proposed routing algorithms (both reactive and proactive protocols) try to find the shortest path or its approximation from the source to the destination [21, 40, 41]. However using the greedy path, i.e. the path found by the greedy algorithm, may cause some nodes to participate in the forwarding of packets of many $(source, destination)$ pairs and hence lose their energy more quickly. Heavily loaded nodes are likely to fail

sooner than other nodes as observed in [46]. Load balancing, also known as congestion control, is the problem of modifying the routing paths to avoid the highly loaded nodes. Note that load balancing can also be done in other network layers (e.g. in transport layer). However, this is orthogonal to the load balancing in the routing layer and our discussion. The techniques for load balancing in routing algorithms can be divided as reactive and proactive load balancing.

1.2.1 Reactive load balancing

Algorithms in this category use the information from the neighbors to adjust the routing paths [46, 49]. As some of the network nodes are highly loaded and/or lose their energy, various parameters of the decision making process (e.g. lifetime and remaining energy of nodes) are updated, which results in selection of new paths according to newly obtained information. One subgroup of these algorithms categorized as power-aware routing protocols in [15] are concerned with energy consumption of nodes.

1.2.2 Proactive load balancing

In the algorithms in this category, routing paths are selected from the beginning (according to the node positions and communication patterns) so that nodes in the network are participating fairly in the routing of the packets. The idea is to avoid the highly loaded network areas (usually at the network center), so that the probability of load at each point of the network is equal [5, 19, 32, 43]. The algorithms from this category can be combined with reactive load balancing to achieve a better performance. In this thesis we are more focused on the proactive load balancing.

Another technique that has been introduced to balance the load in networks is

multi-path routing. In terms of the number of paths between a fixed *(source, destination)* pair, we can divide routing algorithms into single-path algorithms and multi-path algorithms. In single-path routing every packet addressed to a given destination will follow the same path, but in multi-path routing, different packets from a source to the same destination might go through different paths [12, 42].

1.3 Network Model

A computer network is usually represented as a graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes and $E \subseteq \{\{u, v\} | u, v \in V\}$ is the set of links. A link exists between two nodes where direct communication (without using an intermediate node) is possible between the two nodes. In general links can be directional where existence of the link from u to v is independent of the link from v to u. Here we assume only bidirectional links, thus the link from u to v exists if and only if a link from v to u also exists. We denote the set of neighbors of node v by N_v:

$$N_v = \{u \in V | \{v, u\} \in E\}$$

A common model used for homogeneous wireless networks is the *geometric unit disk graph* (UDG) [7]. In the UDG model, each node in the network corresponds to a point in the plane. The transmission ranges of all the nodes are equal (denoted by Tr), and two nodes can communicate directly if and only if their Euclidean distance is less than or equal to Tr. Hence the definition of E (set of links) is given as follows:

$$E = \{\{u, v\} | u, v \in V,\ \|u, v\| \leq Tr\}$$

where $\|u, v\|$ is the Euclidean distance between nodes u and v.

We assume that nodes are location-aware. This means each node knows its coordinates in the network (this can be achieved through GPS-enabled devices or localization techniques [17, 36]). Moreover each node knows the location of its neighbors and the destination coordinates for each packet.

We assume that every node has a limited energy resource. Since the transmission range of nodes is fixed the energy consumed to transmit a packet is independent of the euclidean distance between the node and its neighbor. We do not consider the interference between the nodes and consequent retransmission of the packets. With this simplification, every time a node transmits a packet it will consume a constant amount of energy.

Moreover we do not consider topology changes in the network (due to node mobility, failures of nodes or insertion of new nodes into the network).

1.4 Definitions and Problem Statement

Our network contains n wireless nodes that are located randomly in the network area. The transmission range of all nodes is equal to Tr and connectivity of nodes is determined using UDG model (hence all links are bidirectional). We assume an all-to-all communication pattern between nodes.

A routing algorithm ra is a function mapping a pair of nodes to a path as follows:

$$ra : V \times V \rightarrow P$$

where P is the set of all loop-free paths between pairs of nodes in V and $ra(s, d)$ is a path from node s to node d. The path between nodes s and d is a sequence of nodes $ra(s, d) = a_1, a_2, ..., a_m$ in which $a_1 = s$ and $a_m = d$, called the *source* and *destination* of the path, respectively, where consecutive nodes of the sequence are neighbors. A

routing algorithm defines a path for each pair of nodes in the network area.

For example, given a source *src* and a destination *dest*, the *greedy path* denoted $greedy(src, dest)$ is defined as follows:

$$greedy(src, dest) = a_1, a_2, ..., a_m \text{ where } a_1 = src,\ a_m = dest \text{ and,}$$

$$\|a_i, d\| = \min_{v_j \in N_{a_{i-1}}} \|v_j, d\|$$

where $\|v_i, v_j\|$ is the Euclidean distance between v_i and v_j. In a dense network, it is reasonable to expect that greedy routing creates a path that approximates the straight line connecting source and destination nodes.

Note that if the routing algorithm is multi-path the value of the function is a set of paths instead of one path. Here we focus on single-path routing only but one can extend the definitions for multi-path routing as well.

We define the load of a node w with respect to routing function ra denoted by $load_{ra}(w)$ as the number of packets it forwards during a given period of time.

We are interested in the load of nodes with respect to an all-to-all communication pattern, i.e. every node sends packets to every other node in the network with the same probability. The load of a node w with respect to the routing algorithm ra can then be calculated as follows:

$$load_{ra}(w) = \sum_{u \in V} \sum_{v \in V} Q_{\mathrm{ra}}(u, v, w)$$

where,

$$Q_{\mathrm{ra}}(u, v, w) = \begin{cases} 1 & \text{, if } w \in ra(u, v) \\ 0 & \text{, else} \end{cases}$$

Here we model the total number of packets forwarded by a node in a loop-free routing algorithm. However, note that by using other values of $Q_{\mathrm{ra}}(u, v, w)$, the notion of

load can be generalized to model other parameters, such as energy consumption.

The load balancing problem can be defined as a min-max problem. The goal is to find a routing algorithms ra' such that minimizes the maximum load of the nodes in the network:

$$\max_{v \in V} load_{ra}(v) = \min_{ra'} \max_{v \in V} load_{ra'}(v)$$

A similar problem definition can be found in [20].

1.5 Thesis Contributions

In this thesis we study the load distribution and load balancing problem in wireless ad hoc networks. We base our analysis on a discrete model (UDG model) of network. The contribution of our work is as follows:

- We start with the greedy routing algorithm and derive an estimate for the average and maximum load of the nodes in the network induced by greedy routing. We use a discrete model to calculate the expected load of a node as a function of its geometric coordinates. Also we express the actual load of a node in the network as a random variable, and we obtain the parameters of this random variable by simulations. Our estimate for the load is more accurate than those given by previous studies which were based on a continuous model of the network.

- We study the parameters of the load distribution of the nodes in the network and compare the different routing algorithms in this sense. We study the load distribution of the nodes in both the microscopic and macroscopic level and analyze how different parameters of the network (number of nodes, and transmission range) and different routing algorithms can affect the distribution.

- We introduce a new class of algorithms that moderate the maximum load in the network and compare our algorithms' performance with the existing ones. Experimental results show that our algorithms outperform other existing algorithms in reducing the maximum load of the network.

- We give some techniques to reduce the variance of load distribution among the nodes in the network that can be combined with any location-based routing algorithm to reduce the maximum load of the nodes.

1.6 Thesis Outline

The rest of this thesis is organized as follows. In Chapter 2 we review previous work that is related to the load balancing problem. In Chapter 3 we give our model to analyze the average and maximum load of the nodes in the network induced by greedy routing. We show how using a discrete model can help to achieve a better and more accurate view of the load distribution in the network in comparison with continuous models. In Chapter 4 we introduce some algorithms for reducing the maximum load in the network. We compare the performance of our algorithms using simulations with greedy routing and other load balancing algorithms [5, 9, 19, 43]. Finally in Chapter 5 we conclude our work and give suggestions for future work.

Chapter 2

Related Work

In this chapter we look at previous work on load balancing in wireless ad hoc networks. First we look at the work done on load balancing with multi-path routing in ad hoc networks. Then we look at the previous work on load balancing with single-path routing algorithms and explain them in more detail.

2.1 Load Balancing with Multi-Path Routing

In multi-path routing algorithms, instead of only one path being maintained for every *(source,destination)* pair of nodes as in single-path routing, a set of multiple paths is maintained and used [33]. Multi-path routing has been explored in traditional wired networks for many different reasons i.e. load balancing, throughput improvement, and fault tolerance. Some multi-hop routing algorithms also have been introduced for wireless ad hoc networks to improve the throughput, achieve higher reliability, minimize end-to-end delay, and to balance load (examples are *SMR* [28], *AOMDV* [31], and *AODVM* [51]; for more examples see [33]). Multi-path routing itself can be categorized into different groups depending on whether they have any node or link in

common or not [33]:

- *non-disjoint multi-paths:* where *multi-paths* (set of paths that are used for a given source and destination nodes) from a single source node to a single destination node might have common links or nodes.

- *link disjoint multi-paths*: where multi-paths from a single source node to a single destination node do not have any common link, but they might have common nodes.

- *node disjoint multi-paths*: this is the category where different multi-paths from a single source node to a single destination node have no node or link in common.

In [28] a routing algorithm based on *DSR* called *Split Multi-path Routing (SMR)* is introduced. The protocol tries to find the maximally disjoint multi-paths (multi-paths with minimum common nodes or links) between a given *(source,destination)* pair of nodes. It works similarly to *DSR* where the source node floods a route request packet to the entire network. Intermediate nodes do not send a reply back to the source even if they know a route to the destination node (in contrast with *DSR*). The destination node upon receiving the route request packet, selects the two maximally disjoint multi-paths (the first route is the shortest delay route and the second is the path that is maximally disjoint from the first route) and sends the route reply packets back to the source node through the two selected multi-paths.

In [31], an extension to *AODV* protocol called *AOMDV* is introduced to calculate link-disjoint multi-paths. In the routing tables, multiple entries are stored for a given destination (one for each multi-path) that store next hops and corresponding hop counts to the destination node. Each node stores a value called *advertised hop count* for every destination that represents the maximum hop distance to that destination from the node. A node accepts an alternate path to a destination if its hop count is

less than the advertised hop count for that destination to avoid loops in the paths.

In [51] another extension to *AODV* called *AODVM* is introduced. In *AODVM* intermediate nodes forward all the route request packets to the destination node (like in *AOMDV*). Also each intermediate node stores all the route request packets in a route request table. The destination node sends a route reply back for every received route request. On the way back each intermediate node sends the route reply back to the nodes in the route request table which is on the shortest path to the source node and deletes the route request entry of that node from the route request table. Also any node that overhears the route reply from another node, deletes the entry in the route request table corresponding to the transmitting node to ensure that no node is participating in multiple paths.

A quantitative comparison of the different multi-path routing algorithms is done in [37]. Algorithms were compared with different metrics: routing overhead, packet delivery ratio, average end-to-end delay, and load balancing. Their simulation results show that a 'very good' improvement in packet delivery ratio, and 'good' improvement in average end-to-end delay and routing overhead are achieved with different multi-path routing algorithms. However as their results show, in a network with high density and high load, the improvement is insignificant.

In [42] the performance of single path routing and multi-path routing is compared in terms of packet overhead and traffic distribution. An analytical model has been provided to study the overhead of single-path versus multi-path routing. They show that the overhead of the multi-path routing is not significant when the number of multi-paths created in the route discovery phase is not more than 3. They also show that the maximum increase in over head of multi-path routing compared to single path is 20% for a link failure rate of 50% and approximately 10% when the link failure rate is less than 10% (the failure rate is inverse of expected lifetime of that link). The

authors also analyzed single-path routing and multi-path routing in terms of network congestion and provide an upper bound on the average length of the multi-path routes that guarantees the decrease in the network congestion.

In [12] authors studied multi-path greedy routing versus single path greedy routing in wireless ad hoc networks in terms of load distribution. They modeled the k multi-paths greedy as parallel lines from source to destination node and showed that the multi-path routing will not improve the load balance unless the number of these multi-paths is very large. As they explained, when the density of the nodes in the network is high the greedy paths connecting source node and destination node are very close to the straight line from source to destination nodes. They also confirmed the result from [42] that increasing the number of multi-paths will lead to high overhead in the network and hence is not feasible. Therefore they showed feasible multi-path routing will behave similar to single-path routing in terms of load distribution. They suggested one should use paths that push the traffic load further from the network center (highly congested area) to gain load balance.

In [20] authors compared the single path and multi-path routing in massively dense wireless multi-hop networks from the load balancing perspective. They show that in contrast with fixed networks where multi-path routing can lead to lower congestion, in dense multi-hop wireless networks single path routing can lead to better solutions. In particular they showed that any set of paths can be combined into one single path with a better performance. They also showed how to transform a set of paths to one single path using the vector calculation over the given paths.

As shown in [12, 20, 42] in contrast with wired networks multi-path routing does not achieve a good performance in balancing the load in the network unless a very large number of multi-paths are used. Furthermore, the overhead of discovery and maintenance of such a large number of paths is high and is infeasible in wireless ad

hoc networks.

2.2 Load Balancing with Single-Path Routing

In this section we review some of the single-path routing algorithms for wireless ad hoc networks with respect to their performance in the load balancing.

2.2.1 Curveball routing

In [43], the crowded center effect in a circular network is studied. The authors provided an analytical study of the load probability (the probability that a node forwards a packet on the path of randomly chosen source and destination) distribution when routing on the shortest path. To find the optimal solution for the problem, a cost function is defined. The cost of a path (cost per unit length) at a node with distance r to the network center, is denoted $u(r)$. The values for $u(r)$ are approximated with a linear programming technique as follows. Starting with equal costs for every node the load of each node is calculated for point to point communication (the Dijkstra algorithm is used to implement shortest paths). The cost for each node is updated at each round ($u(r)$ is increased for highly loaded areas and decreased for lowly loaded areas). This is repeated until the values for $u(r)$ converge to a stable state. The routing problem is then to find the greedy path on the calculated costs per point which needs a path integral over the cost function. However the authors state that even using the numerical solutions to the integral is not practical. Then they introduced *curveball routing* as a practical solution that approaches the optimum solution.

In curveball routing each point in the network plane is mapped to a point on the surface of a virtual sphere. The mapping point is the intersection of the line connecting the top of the virtual sphere and the real point with the surface of the

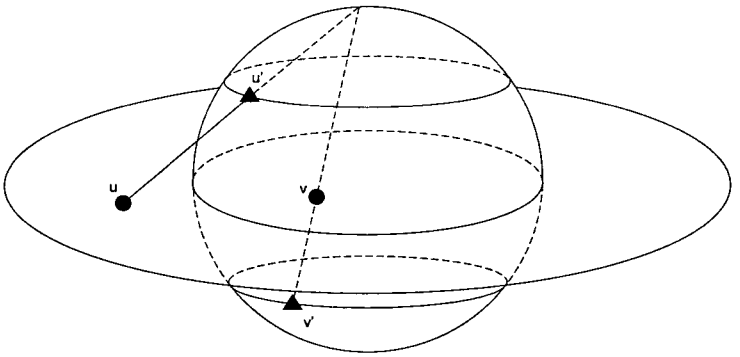


Figure 1: Mapping of nodes in curveball routing; Real nodes are shown with dark circles (u, v) and corresponding virtual points are shown with triangles (u', v').

sphere. Figure 1 shows two nodes in the network and their corresponding virtual points. The routing is simply to find the greedy path on new 3-D virtual coordinates of nodes (instead of real coordinates in greedy routing). An example of the path in curveball routing is shown in Figure 2. They evaluated their algorithm by high level simulation using ns-2[10]. They claim to achieve a decrease in the *maximum average load* (average number of packets forwarded by a node in an annulus) by $35 - 44\%$ and $25 - 40\%$ decrease in maximum load.

Since the decision at each hop is based on the new 3-D coordinates of the nodes, implementation of curveball routing can not be done without modification of the routing layer. Also if greedy routing on the 3-D coordinates fails, the algorithm must switch to greedy routing on 2-D coordinates. This further complicates the algorithm and increases the storage costs.

2.2.2 Radial-path routing

In [19] authors formulate the traffic load of the network by defining the notion of scalar packet flux similar to the concept of particle fluxes in physics. Scalar packet flux can be calculated at each point as the limit of the rate of packets that go through a small line segment perpendicular to the flow direction divided by the length of that line

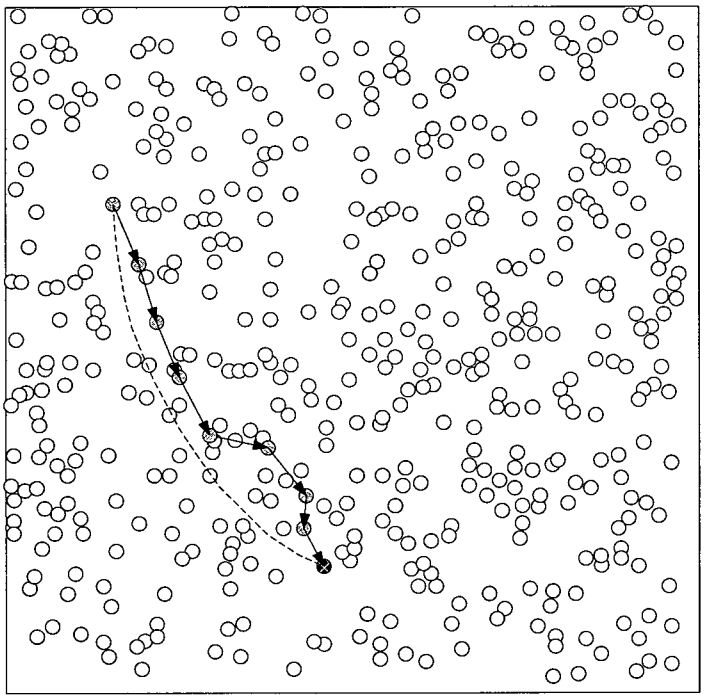
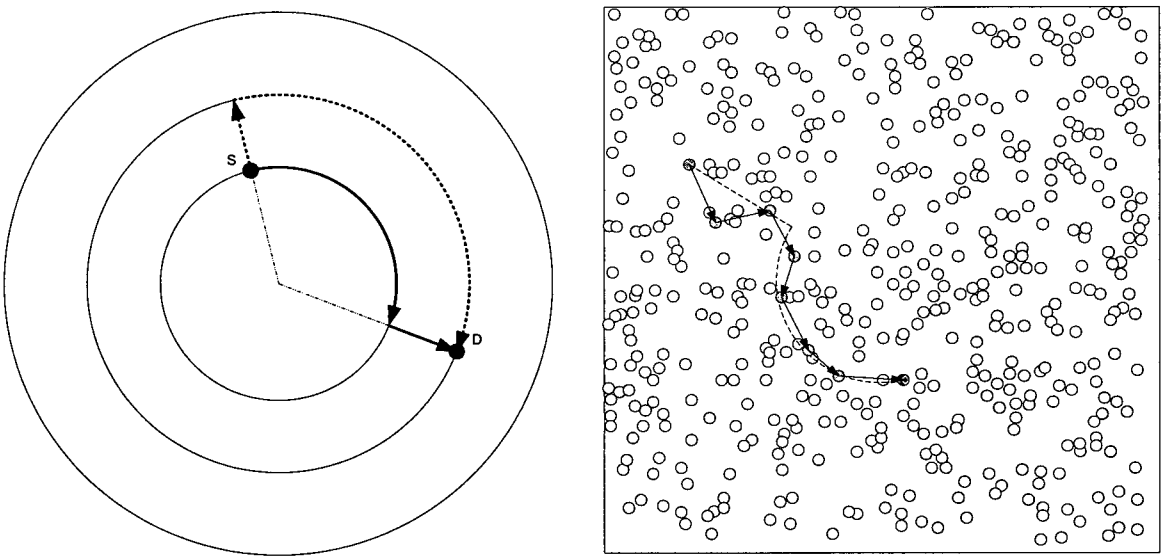


Figure 2: An example of the path in curveball routing. The network area is a square with side length 500, number of nodes in the simulation is 500, and the transmission range of nodes is 50. The dashed line is the path followed if the network area is very dense.

segment when the length goes to zero. So the load balancing problem is formulated as the problem of finding the set of paths that minimizes the maximum scalar packet flux in the network. They calculated the maximum packet flux of shortest path routing, and also gave lower bounds on the load balancing problem and showed how to calculate the scalar flux for any other set of curvilinear paths. They also gave three different heuristics (*Rin*, *Rout* and a combination of shortest-path and Rout) to decrease the maximum load of the network. The paths in these heuristics consist of two segments: radial segment and ring segment. The difference between Rin and Rout paths is in the order of radial and ring segment; In Rin the inner ring segment is used where in Rout path goes on the outer ring. Figure 3a illustrates how Rin and Rout paths are determined. Also an example of the path in Rin routing is shown in Figure 3b. The results are validated through the analytical model.



(a) (b)

Figure 3: Radial path routing: (a) Radial path routing from S to D. Rin and Rout paths are shown with solid and dashed lines respectively. (b) An example of the path in Rin routing. The network area is a square with side length 500, number of nodes in the simulation is 500, and the transmission range of nodes is 50. The dashed line is the path followed if the network area is very dense.

Similar to curveball routing, the implementation of this algorithm needs modifications of the routing layer protocols. This makes the algorithm unsuited to be used along with the existing routing algorithms for ad hoc wireless networks (e.g. DSR and AODV).

2.2.3 Oblivious routing

In [5] an oblivious load balancing algorithm is introduced. The term oblivious implies that each packet from a single source to a single destination node is routed without using the information from the previously routed packets. Hence two packets from a given flow might go through different randomly chosen routes. This randomization is intended to achieve a better load balancing, but also increases the computational overhead of the algorithm. For each packet the route is selected as follows:

Consider the line connecting the source node to destination node, denote the perpendicular bisector line of this line by $l^{\perp}$. An intermediate point is selected uniformly at random on the line segment of $l^{\perp}$ contained in the network area. Then the packet is sent to (a node close to) this intermediate point and after reaching the intermediate node, the packet is forwarded to the destination. The authors proved that on uniform unit disk graphs the stretch factor of the paths is $O(1)$ and the node congestion is $O(C^{*}_{node} \cdot \log n)$ where C^{*}_{node} is the optimal node congestion and n is the number of nodes. However they did not provide any high level simulation results of their algorithm.

2.2.4 One-turn rectilinear routing

In [9] studied different one-turn rectilinear paths to address the load balancing problem in wireless multi-hop ad hoc networks. One-turn rectilinear routing assigns to each *(source,destination)* pair one of the possible rectilinear paths (either first row

then column or first column then row combination) with at most one turn point. A heuristic named *diagonal rectilinear routing* is introduced (see Figure 4a for details). An example of the path in diagonal rectilinear routing is shown in Figure 4b.

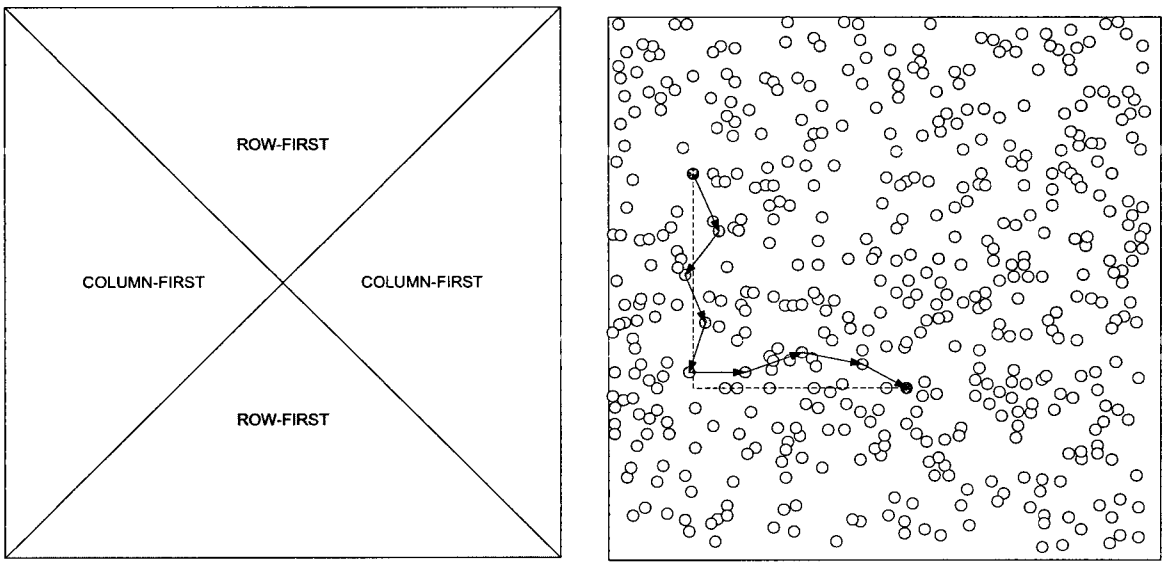


Figure 4: Diagonal rectilinear routing: (*a*) Illustration of diagonal rectilinear routing: if the source lies in shaded area row-first routing is selected and otherwise column-first routing is performed. (*b*) An example of the path in diagonal rectilinear routing. The network area is a square with side length 500, number of nodes in the simulation is 500, and the transmission range of nodes is 50. The dashed line is the path followed if the network area is very dense.

The authors formulated the load of the straight line routing and derived the expression for the maximum and average load. Furthermore they showed that for every one-turn rectilinear routing the average and maximum stretch factor is 1.274 and $\sqrt{2}$ respectively. They introduced several one-turn rectilinear policies and showed 33% decrease in the maximum load of diagonal rectilinear policy over straight line routing.

2.2.5 Outer space routing

In [32] authors introduced a new algorithm for balancing the load over the nodes in the network called *outer space routing*. The idea is to transform the node coordinates to new virtual coordinates and then use greedy path on the new coordinates. They define the network area to be symmetric if given two nodes v_1 and v_2 selected randomly from the nodes in the network, the probability that the greedy path from v_1 to v_2 goes through a specific node u is independent of its position. They showed that if a space is symmetric then an all-to-all communication pattern will lead to equal load distribution on the nodes. Furthermore they gave a candidate symmetric space (a torus) and provide a one to one mapping function from the nodes in a square to the torus. Providing the simulation results on their own event based simulator they showed that their algorithm (applied to geographical routing) gives 32% lower load in the network center and 34% increase in average path stretch and global energy consumption compared to the original algorithm.

Chapter 3

Load Modeling

3.1 Introduction

The routing problem in a wireless ad hoc network has received extensive attention in recent years and, as described in Chapter 2, many routing algorithms have been developed for these networks [2, 27, 38, 47]. Many of these routing algorithms (e.g. greedy routing) try to find an approximation of the shortest path between the source and destination nodes. These algorithms attempt to maximize the delivery rate of packets, and minimize the lengths of paths travelled by packets.

It is only recently that researchers started to consider the load induced by a routing algorithm on nodes of the network [5, 9, 13, 14, 19, 43]. It is easy to see that the routing algorithm that induces the *minimum average load* is the shortest path routing. However, in general, the *maximum load* induced by shortest path routing can be very high. In particular, nodes near the geographical center of the network are very highly loaded when using shortest path routing, while nodes near the periphery of the network are usually very lightly loaded.

To design load-balanced routing algorithms, it is necessary to have a good understanding of the average and maximum load induced by a routing algorithm. One common model that has been used to analyze the average and maximum load induced by a routing algorithm is to consider networks that are so dense that a node exists close enough to any point in the network. In other words, every point in the network area corresponds to a node in the network. In such a network, the shortest path between two nodes (points) is simply the straight line connecting them. This assumption is shared by all the researches described in Chapter 2. Thus continuous techniques were used to find the maximum and average load induced by shortest path routing in the network. In [19, 42, 43], these quantities were studied for a network contained in a circular area for shortest path routing in [19, 43], and radial ring paths in [19]. Shortest path routing and rectilinear routing in a square area were studied in [9].

We are interested in analyzing the maximum and the average load induced by *greedy routing* in a network modeled by a random *unit disk graph.* Since greedy routing in a dense network can be expected to return a path very close to a straight line between the source and destination, it is conceivable that the results of [9, 19, 43] can be used to determine maximum and average load induced by greedy routing in a discrete network. However, we argue that the continuous model has some limitations that make it unsuitable for this purpose:

- Analysis based on a continuous model does not give the actual values of the maximum and the average load. The values that have been calculated were only appropriate for comparison between performance of different algorithms.

- The techniques in the continuous model apply to straight line paths [9, 42], or smooth curves as in the analysis of radial ring paths [19]. While greedy routing can be approximated by a straight line from the source to the destination for

dense networks, in a real network, the path produced by greedy routing is not a straight line. The deviation of the path from the straight line varies according to the density of nodes and transmission radius. These factors are not considered in the previous analysis of the load of shortest path routing.

- The ratio of the maximum load to the average load of the network induced by greedy routing as calculated by simulations do not match the theoretical results for straight line routing. For example, the ratio of maximum load to average load induced by straight line routing in a circular area can be calculated to be 2.212 from the results of [19]. Fig. 5 shows the ratio of the maximum load to the average load of the network as obtained from simulations of greedy routing in a network of n nodes in a circular area. It can be seen that the ratio between the maximum load to average load varies depending on the number of nodes in the network area, and in all cases is much higher than the ratio predicted by the continuous model.

- In a random network of n nodes, the load of a node induced by greedy routing is a random variable, which in simulations exhibits high variance. However, in continuous models, the load of a node is not a random variable, and depends only on its position in the network. As shown in Fig. 6, simulations on a circular area show that the load of nodes that are situated very close to each other can vary widely. In contrast in the continuous model, in a circular area, the load of a node is a decreasing function of its distance from the center.

This shows that assuming a network to be continuous space (even if it is very dense) is not a realistic assumption.

In this chapter, we present a new approach to estimate the average and maximum load induced by greedy routing in a network that is modeled as a unit disk graph with

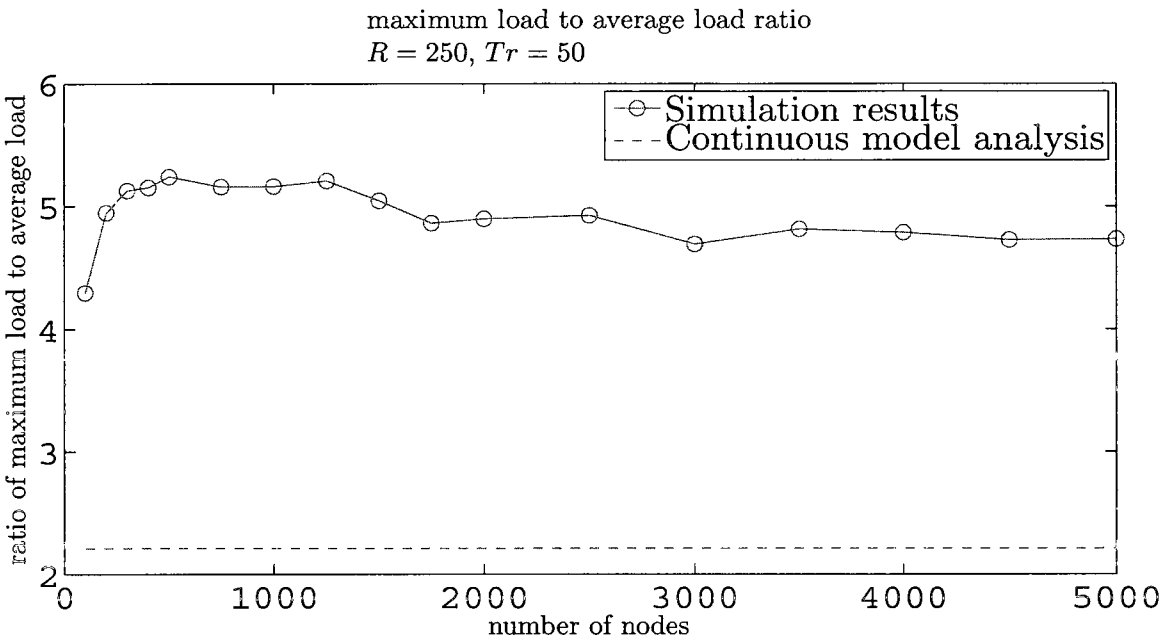


Figure 5: Simulation results versus analytical results for ratio of maximum load to average load of nodes in greedy routing. Network area is circular with radius R and transmission range of all the nodes is Tr.

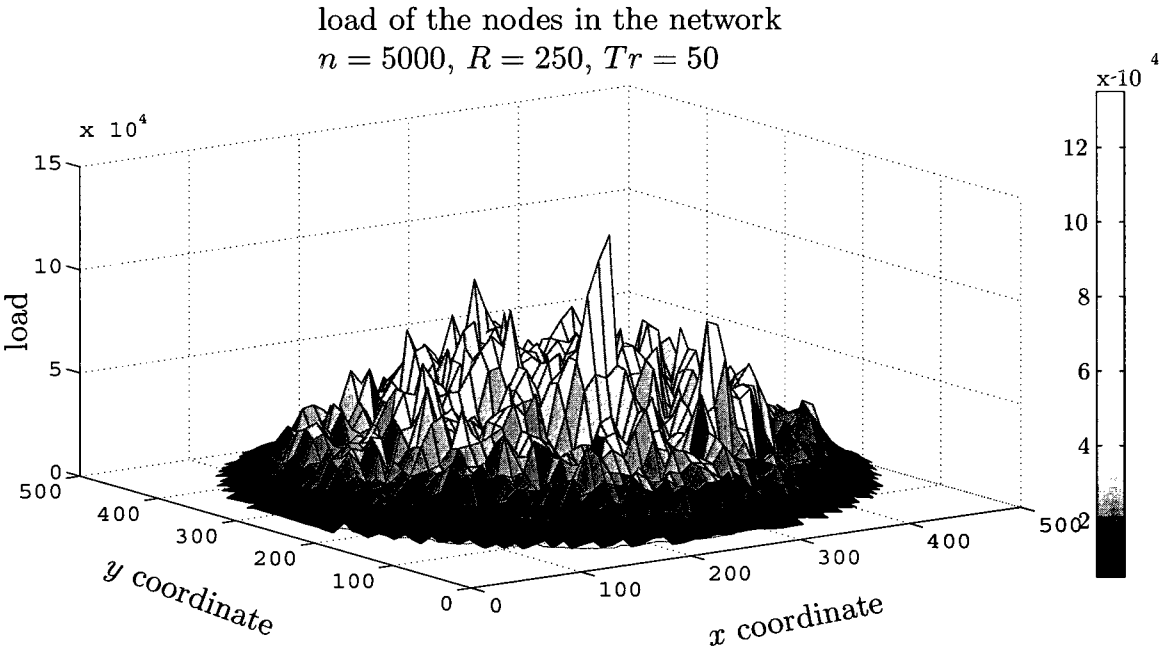


Figure 6: Simulation results of load of the nodes of greedy routing. Network area is circular with radius R, n is the total number of nodes and Tr is the transmission range of nodes.

nodes distributed uniformly at random in a circular area. This is the first study that gives discrete methods for the calculation of average and maximum load induced by greedy routing in a wireless network. Several variations of greedy routing have been introduced [18, 23, 25]. We consider only *least remaining distance* greedy routing (LRD) [8] with the condition that back forwarding is not allowed (when no positive progress is possible the packet is dropped). First we calculate the average progress made in a hop by greedy routing. Our techniques are somewhat different from and more precise than those used in [8, 18, 25, 50]. We then use this to calculate the average load of nodes in the network. Our estimate of average load is validated by simulation results. To calculate the maximum load over all nodes, we first find the average load of nodes contained in an annulus of radius d and width ϵ, denoted $\overline{\xi}_{d,\epsilon}$. Next we define a random variable X_i, which is a function of the load of node v_i. Using simulations, we demonstrate that the distribution of X_i depends only on the density of nodes in the network, and is independent of the position of the node v_i. We estimate the parameters of this random variable using simulations. Finally we can calculate the maximum load using X_i and $\overline{\xi}_{d,0}$. Our estimate is matched closely by simulation results.

The rest of this chapter is organized as follows. In Section 3.2, we define our notion of progress made in one hop, and show how to estimate the average progress in one hop. In Section 3.3 we estimate the average load of nodes of the network induced by greedy routing. In Section 3.4 we calculate the average load of nodes that are in an annulus with a variable inner radius and width. In Section 3.5,we give an estimate for the maximum load of the network. We discuss our conclusions and suggest directions for future work in Section 3.6.

3.2 Average progress in one hop

In this section we give an estimate for the average progress in one hop of the path produced by greedy routing. This problem has attracted the attention of several researchers [8, 18, 25, 50]. This is applicable in several areas of research such as average length of a path and transmission range control. In [25] the expected progress and path length in connection with optimal transmission range is evaluated assuming that moving away from the destination is allowed. In [50] the expected progress of a packet in one hop is calculated as a step of obtaining one-hop throughput and optimal transmission radii. In [18] some variation of greedy routing (e.g. MFR and MVR) are studied and average progress is obtained in relation to transmission range control and one-hop capacity. In [8] another variation of greedy routing (LRD) is studied and average progress in one hop is obtained which is used to derive bounds on the length of a path (in hop count) in greedy routing.

In this section we apply a similar approach to [25], with some modifications that make the result more precise, to calculate the expected progress toward the destination node in one hop (excluding the last hop).

Let hop length $h_{a_i,d} = \|a_i, a_{i+1}\|$ where a_{i+1} is the next hop in $p(a_i, d)$. We define $\|\overline{a_i, v}\|$ to be the length of the projection of the line segment (a_i, v) on the line connecting a_i and destination node (see Fig. 7).

Hence $\overline{h}_{a_i,d} = \|\overline{a_i, a_{i+1}}\|$ is the length of the projection of the hop length on the line connecting a_i to destination (the progress toward destination in i^{th} hop). Since the positions of the nodes are chosen uniformly at random, $\overline{h}_{a_i,d}$ is a random variable and its expected value can be calculated by:

$$E\left[\overline{h}_{a_i,d}\right] = \int_0^{Tr} 1 - Prob\left[\overline{h}_{a_i,d} \leq x\right] dx \qquad (1)$$

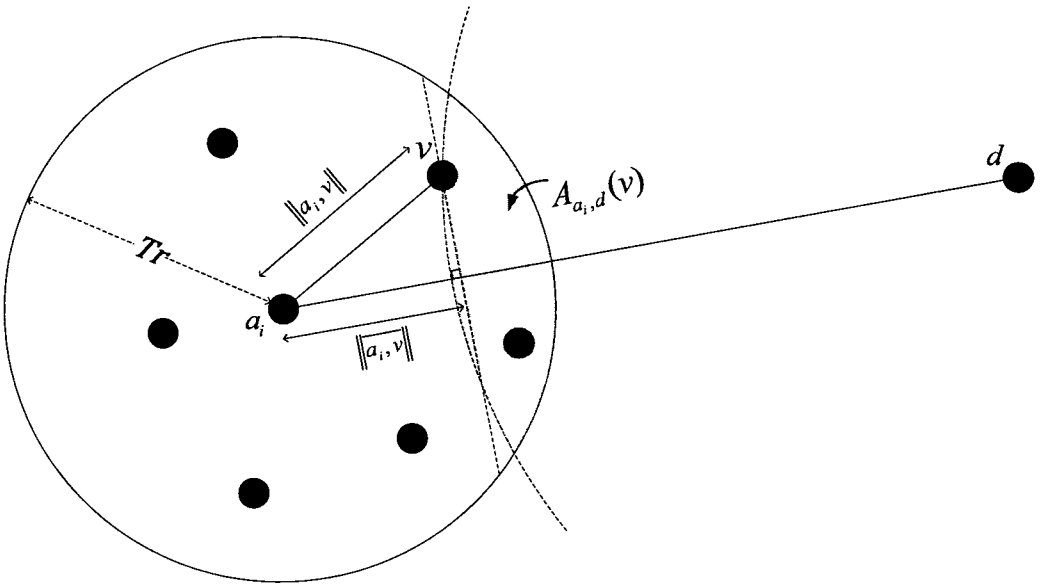


Figure 7: $A_{a_i,d}(v)$ is the set of all points in the transmission range of a_i that are closer to destination than v and $\|\overline{a_i, v}\|$ is the projection of $\|a_i, v\|$ on the line connecting a_i and destination.

Note that when no node among the neighbors is closer to the destination than the current node the packet is dropped and progress toward destination is zero thus we started the integral from 0 instead of $-Tr$ unlike in [25].

Let $A_{a_i,d}(v)$ be the intersection area of the circle centered at a_i with radius Tr and the circle centered at d that passes through v. From the definition of the routing algorithm, we know that a_{i+1} is the nearest neighbor of a_i to the destination. This implies that the region $A_{a_i,d}(a_{i+1})$ does not contain any nodes. To simplify the calculations, we assume that the distance of the current node to the destination node is much greater than the range of the nodes. Then the region $A_{a_i,d}(v)$ can be approximated by a circular segment $A'_{a_i,d}(\|\overline{a_i, v}\|)$ defined as follows (see Fig. 8):

$$v' \in A'_{a_i,d}(\|\overline{a_i, v}\|) \leftrightarrow v' \in N_{a_i} \text{ and } \|\overline{a_i, v}\| \leq \left\|\overline{a_i, v'}\right\|$$

It is an interesting property of this segment that it is only a function of Tr and $\|\overline{a_i, v}\|$

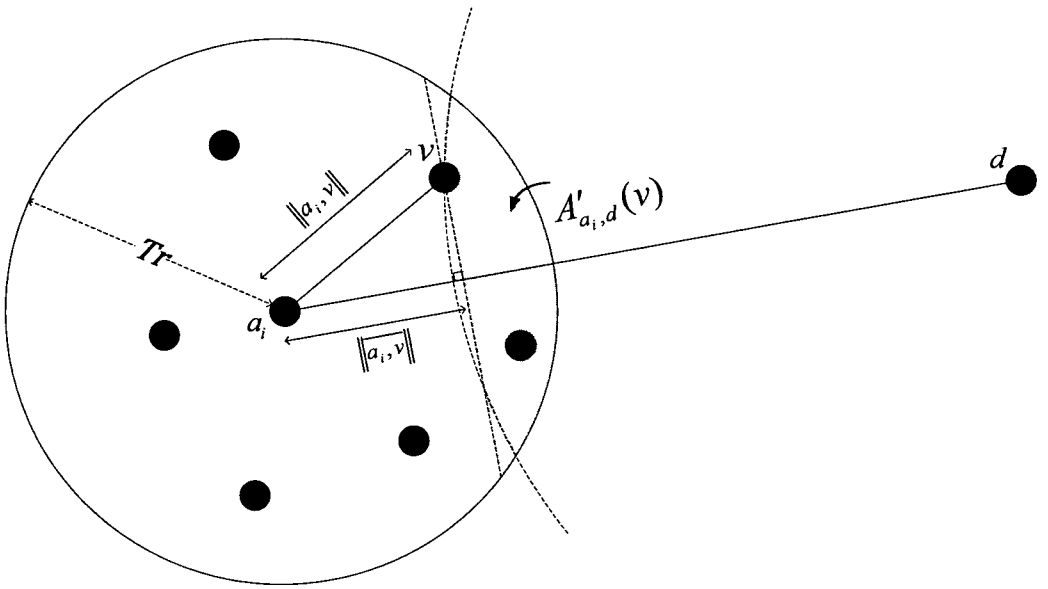


Figure 8: Approximation of $A_{a_i,d}(v)$ by $A'_{a_i,d}(\|\overline{a_i,v}\|)$

which is the length of the projection of the line segment (a_i, v) on the straight line connecting a_i and d. The area of $A'_{a_i,d}(\|\overline{a_i,v}\|)$ is calculated as follows:

$$\left|A'_{a_i,d}(\|\overline{a_i,v}\|)\right| = Tr^2 \arccos(\frac{\|\overline{a_i,v}\|}{Tr}) - \|\overline{a_i,v}\| \sqrt{Tr^2 - \|\overline{a_i,v}\|^2} \tag{2}$$

From the definition of $A'_{a_i,d}(\|\overline{a_i,v}\|)$ we know $Prob\left[\overline{h}_{a_i,d} \leq x\right] = Prob[A'_{a_i,d}(x)$ is empty$]$. Putting this in Eq.1 we will get:

$$\begin{aligned} E\left[\overline{h}_{a_i,d}\right] &\approx \int_0^{Tr} 1 - \left(1 - \frac{\left|A'_{a_i,d}(x)\right|}{\pi R^2}\right)^n dx \\ &= Tr - \int_0^{Tr} \left(1 - \frac{\left|A'_{a_i,d}(x)\right|}{\pi R^2}\right)^n dx \end{aligned} \tag{3}$$

where $n = |V|$ is the total number of nodes in the network. The expression in 3 is numerically computed in order to estimate average and maximum load of a network.

Our calculation of $E\left[\overline{h}_{a_i,d}\right]$ is based on the assumption that the distance between the current node and the destination is much greater than the transmission radius.

However, we use the calculated value of $E\left[\overline{h}_{a_i,d}\right]$ even when this assumption does not hold true. Moreover, we will use the calculated value of $E\left[\overline{h}_{a_i,d}\right]$ as an estimate of the average projection of the hop length on the line connecting the *source* node to the *destination* node (shown by $\overline{h'}_{a_i,d}$ in Fig. 9). It can be expected that the two quantities are quite close when the angle between the line connecting the current node to the destination and the line connecting the source node to the destination is small enough. Observe that this is true when the density of nodes is high or when the current node is far from the destination. In spite of the approximations made

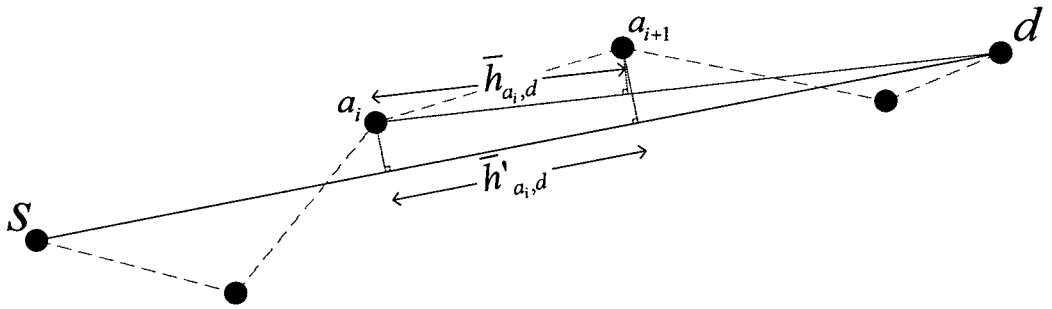


Figure 9: Difference between projection of $\|a_i, a_{i+1}\|$ on the line connecting current node to destination ($\overline{h}_{a_i,d}$) and on the line connecting source node to destination ($\overline{h'}_{a_i,d}$)

above, Fig. 10 shows that the average progress in one hop as measured by simulations matches the value of $E\left[\overline{h}_{a_i,d}\right]$ very closely. In the rest of this paper we denote $E\left[\overline{h}_{a_i,d}\right]$ by $\overline{h}$.

3.3 Average load of the network

The distance between any two nodes v_i and v_j can be expressed as the sum of a multiple of $\overline{h}$ and a remainder (which corresponds to an estimate for the last hop progress):

$$\|v_i, v_j\| = m_{ij} * \overline{h} + x_{ij}$$

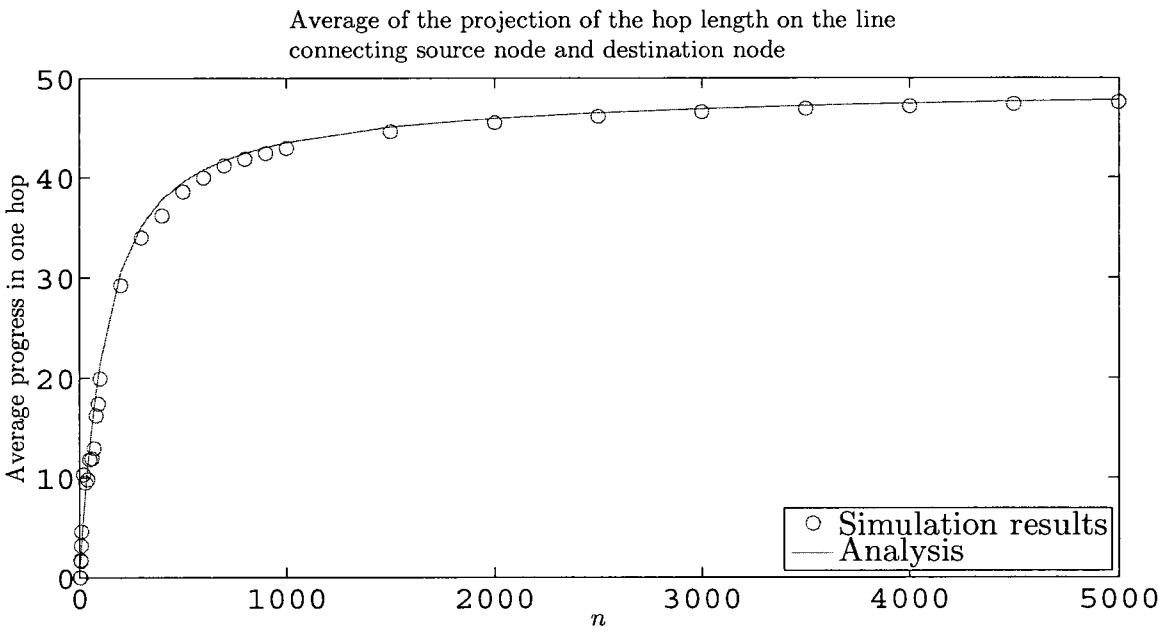


Figure 10: Simulation versus analytical results for the average of projection of the hop length on the line connecting source node to the destination node (progress in one hop).

where m_{ij} is an integer and x_{ij} is a real number in the range $\left(0, \overline{h}\right]$. We use $m_{ij}+1$

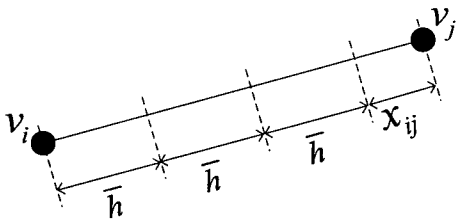


Figure 11: Approximation for the hop count of the path from v_i to v_j.

as an estimate for the hop count of the path between v_i and v_j (see Fig. 11). Using the equation for average forwarding index in a graph in [6] we calculate our estimate for average load as follows:

$$\begin{aligned}\overline{load}_{greedy} &= \frac{1}{n}\sum_{v_i\in V}\sum_{v_j\in V-\{v_i\}} \text{hop count of the path } v_i \text{ to } v_j \\ &\approx \frac{1}{n}\sum_{v_i\in V}\sum_{v_j\in V-\{v_i\}}\left(\frac{\|v_i,v_j\|-x_{ij}}{\overline{h}}+1\right)\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{n}\left(\sum_{v_i \in V}\sum_{v_j \in V-\{v_i\}} \frac{\|v_i, v_j\|}{\bar{h}} - \sum_{v_i \in V}\sum_{v_j \in V-\{v_i\}} \frac{x_{ij}}{\bar{h}} + \sum_{v_i \in V}\sum_{v_j \in V-\{v_i\}} 1\right) \\
&\approx \frac{(n-1)\bar{l}_R}{\bar{h}} - \frac{n-1}{2} + (n-1) \\
&= (n-1)\left(\frac{\bar{l}_R}{\bar{h}} + \frac{1}{2}\right) \qquad (4)
\end{aligned}
$$

where $\bar{l}_R$ is the average length of a line in a circle with radius R which is given by $\bar{l}_R = \frac{128}{45\pi}R$ [24]. Note that for all v_i and v_j we approximate $\frac{x_{ij}}{\bar{h}}$ as a random variable with uniform distribution in the range $(0, 1]$. As shown in Fig. 12, the estimated average load matches the simulation results very closely.

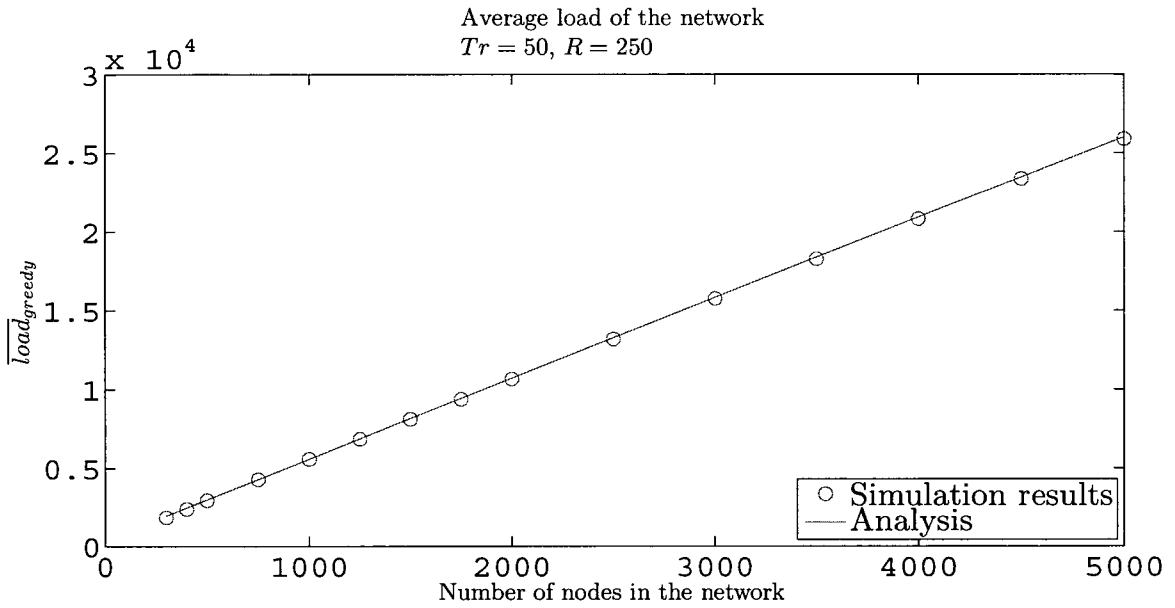


Figure 12: Simulation versus analytical results for the average load of nodes in the network.

3.4 Average load of nodes in an annulus

In the continuous models used in [9, 19], the maximum load of shortest path routing occurs at the center of the network. However in a discrete network when nodes are scattered randomly in the area this is not necessarily the case. Here we

first calculate the average load of nodes that are within a circle with radius $a \leq R$ centered at the origin. Then we will use this to give an estimate for the average of load of nodes that are in an annulus area with a given radius and width.

Define $\overline{load}_{greedy}(C_a)$ to be the average load of nodes in a circle of radius a centered at the origin:

$$\overline{load}_{greedy}(C_a) = \frac{1}{n_a} \sum_{v \in V_a} load_{greedy}(v) \tag{5}$$

where V_a is the set of the nodes inside the circle with radius a and $n_a = |V_a|$ is the number of nodes in V_a.

Here we use a similar approach to that we used to calculate the average load of the network. The packets forwarded by a node in C_a fall in the following categories, and we analyze the load contributed by them separately:

1. $\bar{l}_1(C_a)$: Average load contributed to nodes in C_a by packets where source and destination nodes are both inside the circle ($s, d \in V_a$).

2. $\bar{l}_2(C_a)$: Average load contributed to nodes in C_a by packets where source node is inside the circle and destination node is outside ($s \in V_a$ and $d \notin V_a$).

3. $\bar{l}_3(C_a)$: Average load contributed to nodes in C_a by packets where source node is outside the circle and destination node is inside ($s \notin V_a$ and $d \in V_a$).

4. $\bar{l}_4(C_a)$: Average load contributed to nodes in C_a by packets where source and destination nodes are both outside the circle ($s, d \notin V_a$).

Hence:

$$\overline{load}_{greedy}(C_a) = \bar{l}_1(C_a) + \bar{l}_2(C_a) + \bar{l}_3(C_a) + \bar{l}_4(C_a) \tag{6}$$

Using calculations similar to those for average load (using Eq.4) we can find $\bar{l}_1(C_a)$:

$$\bar{l}_1(C_a) \approx (n_a - 1)\left(\frac{\bar{l}_a}{\bar{h}} + \frac{1}{2}\right) \tag{7}$$

For any two nodes v_i, v_j, where v_i is inside C_a and v_j is outside C_a we define $d'(v_i, v_j)$ to be the length of the part of the straight line (v_i, v_j) that falls in C_a.

Let $d'(v_i, v_j) = m_{ij} * \overline{h} + x_{ij}$ where $m_{ij} \geq 0$ is an integer and $0 \leq x_{ij} < \overline{h}$ is a real number corresponding to the estimates of the number of complete hops that fall in C_a and the length of the last partial hop that falls in the C_a respectively (see Fig. 13). We estimate the number of nodes in C_a that forward packets from v_i destined to v_j by $m_{ij} + 1$ as follows:

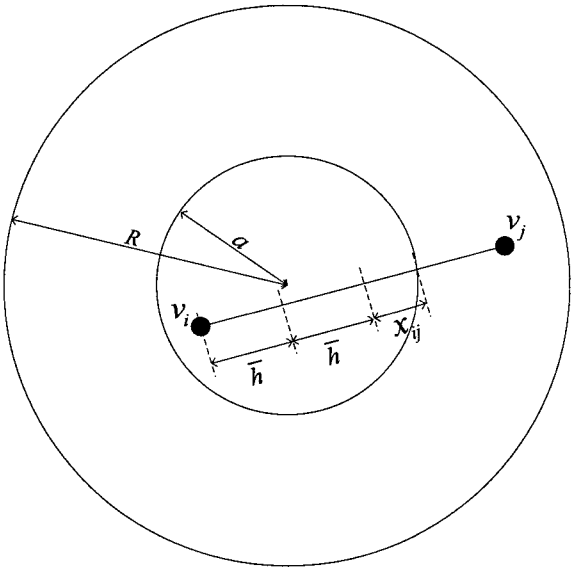


Figure 13: Estimate for hop count of a path that falls in C_a when source node (v_i) is inside C_a and destination node (v_j) is outside (case $\bar{l}_2(C_a)$).

$$\bar{l}_2(C_a) \approx \frac{1}{n_a} \sum_{v_i \in V_a} \sum_{v_j \in V - V_a} \left(\frac{d'(v_i, v_j) - x_{ij}}{\overline{h}} + 1 \right)$$

Consider the set of all line segments with one endpoint in C_a and the other endpoint in $C_R - C_a$. Let $\overline{l}'_{a,R}$ be the average length of the portion of these line segments that lies within C_a. Hence:

$$\bar{l}_2(C_a) \approx \frac{1}{n_a} \sum_{v_i \in V_a} \sum_{v_j \in V - V_a} \frac{d'(v_i, v_j)}{\overline{h}} - \frac{x_{ij}}{\overline{h}} + 1$$

$$
\begin{aligned}
&\approx \frac{1}{n_a}(n_a(n-n_a)\frac{\overline{l'}_{a,R}}{\overline{h}} - \frac{1}{2}n_a(n-n_a) + n_a(n-n_a)) \\
&= (n-n_a)\left(\frac{\overline{l'}_{a,R}}{\overline{h}} + \frac{1}{2}\right) \qquad (8)
\end{aligned}
$$

Similarly for $\bar{l}_3(C_a)$, let $d'(v_i, v_j) = m_{ij} * \overline{h} + x_{ij} + y_{ij}$ where $m_{ij} \geq 0$ is an integer and $0 \leq x_{ij}, y_{ij} < \overline{h}$ are real numbers corresponding to the number of complete hops that fall in C_a, length of the first partial hop of the path that falls in C_a and the length of the last hop respectively (see Fig. 14). We estimate the number of nodes in C_a that forward packets from v_i destined to v_j by $m_{ij} + 1$:

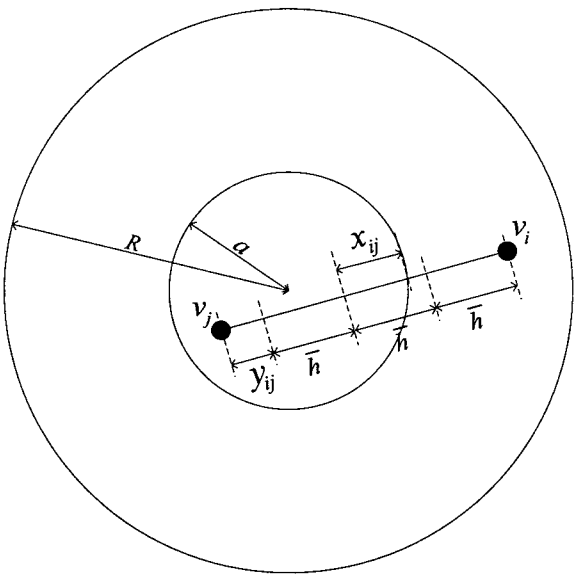


Figure 14: Estimate for hop count of a path that falls in C_a when source node (v_i) is outside C_a and destination node (v_j) is inside (case $\bar{l}_3(C_a)$).

$$
\begin{aligned}
\bar{l}_3(C_a) &\approx \frac{1}{n_a}\sum_{v_i \in V - V_a}\sum_{v_j \in V_a}\left(\frac{d'(v_i,v_j) - x_{ij} - y_{ij}}{\overline{h}} + 1\right) \\
&= \frac{1}{n_a}\sum_{v_i \in V - V_a}\sum_{v_j \in V_a}\frac{d'(v_i,v_j)}{\overline{h}} - \frac{x_{ij}}{\overline{h}} - \frac{y_{ij}}{\overline{h}} + 1 \\
&\approx \frac{1}{n_a}\left(n_a(n-n_a)\frac{\overline{l'}_{a,R}}{\overline{h}} - \frac{1}{2}n_a(n-n_a) - \frac{1}{2}n_a(n-n_a) + n_a(n-n_a)\right) \\
&= (n-n_a)\frac{\overline{l'}_{a,R}}{\overline{h}} \qquad (9)
\end{aligned}
$$

For v_i, v_j both outside C_a, we define $d''(v_i, v_j)$ to be the length of the part of the straight line (v_i, v_j) that falls in C_a. Let $d''(v_i, v_j) = m_{ij} * \overline{h} + x_{ij} + y_{ij}$ where $m_{ij} \geq 0$ is an integer and $0 \leq x_{ij}, y_{ij} < \overline{h}$ are real numbers corresponding to the number of complete hops that fall in C_a, length of the first partial hop of the path that falls in C_a and the length of the last partial hop that falls in C_a respectively (see Fig. 15). We use $m_{ij} + 1$ as our estimate for the number of nodes in C_a that forward packets originated at v_i and destined to v_j:

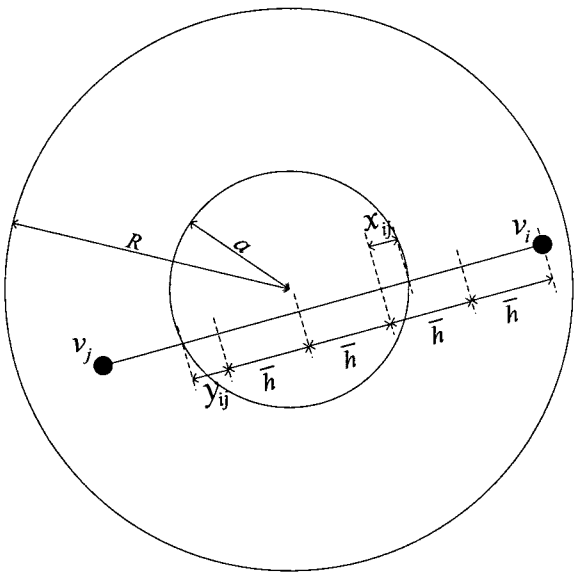


Figure 15: Estimate for hop count of a path that falls in C_a when both source(v_i) and destination(v_j) nodes are outside C_a (case $\bar{l}_4(C_a)$).

$$\bar{l}_4(C_a) \approx \frac{1}{n_a} \sum_{v_i \in V - V_a} \sum_{v_j \in V - V_a - v_i} \left(\frac{d''(v_i, v_j) - x_{ij} - y_{ij}}{\overline{h}} + 1 \right) \tag{10}$$

Consider the set of all line segments with both endpoints in $C_R - C_a$. Let $\overline{l''}_{a,R}$ be the average length of the portion of these line segments that lies within C_a. Hence:

$$\begin{aligned} \bar{l}_4(C_a) &\approx \frac{1}{n_a} \sum_{v_i \in V - V_a} \sum_{v_j \in V - V_a - v_i} \frac{d''(v_i, v_j)}{\overline{h}} - \frac{x_{ij}}{\overline{h}} - \frac{y_{ij}}{\overline{h}} + 1 \\ &\approx \frac{1}{n_a}((n - n_a)(n - n_a - 1)\frac{\overline{l''}_{a,R}}{\overline{h}} \end{aligned}$$

$$- (n - n_a)(n - n_a - 1) + (n - n_a)(n - n_a - 1))$$

$$= (n - n_a)(n - n_a - 1)\frac{\overline{l''}_{a,R}}{n_a \overline{h}} \quad (11)$$

Placing Eq. 7,8,9 and 11 into Eq.6 we can calculate the average load $\overline{load}_{greedy}(C_a)$ of nodes inside a circle centered at the origin. Fig. 16 shows the simulation and analytical results for $\overline{load}_{greedy}(C_a)$ ($\overline{l'}_{a,R}$ and $\overline{l''}_{a,R}$ are calculated numerically and n_a is estimated by $\frac{a^2 n}{R^2}$ in the calculations).

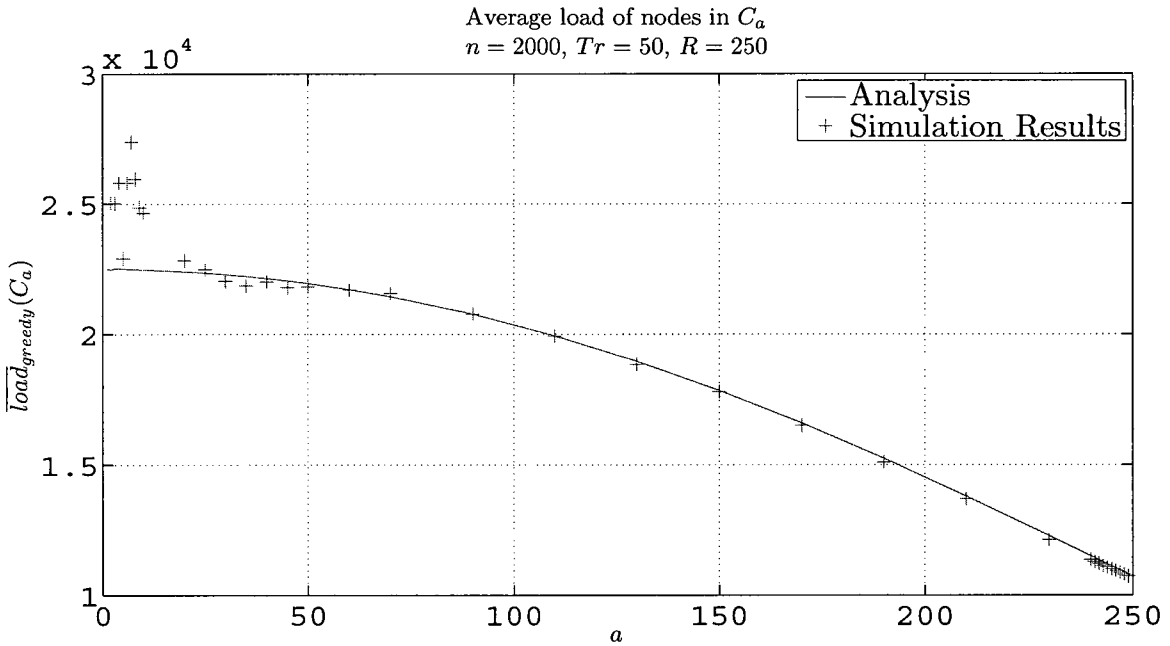


Figure 16: Simulation versus analytical results for the average load of nodes in C_a.

Using this result we can calculate the average load of nodes in an annulus with inner radius d and width ϵ centered at network origin denoted by $\overline{\xi}_{d,\epsilon}$ as follows:

$$\overline{\xi}_{d,\epsilon} = \frac{\overline{load}_{greedy}(C_{d+\epsilon})n_{d+\epsilon} - \overline{load}_{greedy}(C_d)n_d}{n_{d+\epsilon} - n_d} \quad (12)$$

3.5 Maximum load of nodes in the network

In the previous section we estimated the average load of the nodes in any annulus centered at origin with a given inner radius and width. In consistency with continuous models, the values for $\overline{\xi}_{d,0} = \lim_{\epsilon \to 0} \overline{\xi}_{d,\epsilon}$ can also be obtained for any $0 < d < R$ by a linear transform of the result given by a continuous model. We consider a general linear transform of the following form:

$$\overline{\xi}_{d,0} = \alpha \times \Phi_{sp}(\frac{d}{R}) + \beta$$

where $\Phi_{sp}(\frac{d}{R})$ is the *scalar flux* at distance $\frac{d}{R}$ of the center of a unit disk defined in [19] ($\Lambda = 1$).

Solving a system of two equations (maximum and minimum load of the network) we can calculate the parameters α and β as follows:

$$\alpha = \frac{\overline{\xi}_{0,0} - \overline{\xi}_{R,0}}{0.637}$$

and,

$$\beta = \overline{\xi}_{R,0}$$

where $\overline{\xi}_{0,0} = \lim_{d,\epsilon \to 0} \overline{\xi}_{d,\epsilon}$. Hence using analogy to the continuous model we can show:

$$\overline{\xi}_{d_1,0} \leq \overline{\xi}_{d_2,0} \leftrightarrow d_1 \geq d_2 \tag{13}$$

Here we define a random variable X_i as the ratio of $load_{greedy}(v_i)$ to $\overline{\xi}_{\|v_i,O\|,0}$:

$$X_i = \frac{load_{greedy}(v_i) - (n-1)}{\overline{\xi}_{\|v_i,O\|,0} - (n-1)}$$

Note that we subtracted $n-1$ which is the number of packets that one node generates

and sends to all other nodes in the network from both numerator and denominator because this part is deterministic and not random ($n-1$ for all nodes in the network).

Simulation results show that distribution of X_i is independent of the position of the nodes and the shape of the network area:

$$\forall v_i, v_j \in V \text{ and } \forall x > 0 \Rightarrow f_{X_i}(x) = f_{X_j}(x) = f_X(x)$$

where $f_{X_i}(x)$ is the probability density function of the random variable X_i. We can estimate X as a gamma random variable with parameters (k, θ). The *probability density function (pdf)* and *cumulative distribution function (cdf)* of X would be as follows:

$$f_X(x) = \frac{x^{k-1} e^{\frac{-x}{\theta}}}{\Gamma(k)\theta^k} \tag{14}$$

$$F_X(x) = \frac{\gamma(k, \frac{x}{\theta})}{\Gamma(k)} \tag{15}$$

where $\Gamma(k) = \int_0^\infty t^{k-1} e^{-t}\, dt$ and $\gamma(k, \frac{x}{\theta}) = \int_0^{\frac{x}{\theta}} t^{k-1}\, e^{-t}\, dt$ are *gamma* and *lower incomplete gamma* functions respectively. We can determine the parameters of this random variable by simulation. As our simulation shows parameters are a function of the average number of neighbors of every node which itself is a function of the density of nodes in the network and the transmission range of each node:

$$\theta \approx 0.883 x^{-0.309},\ k = \frac{1}{\theta}$$

where x in this formula is average number of neighbors of nodes in the network (Fig. 17).

We use this result to calculate the expected value for maximum load of the network

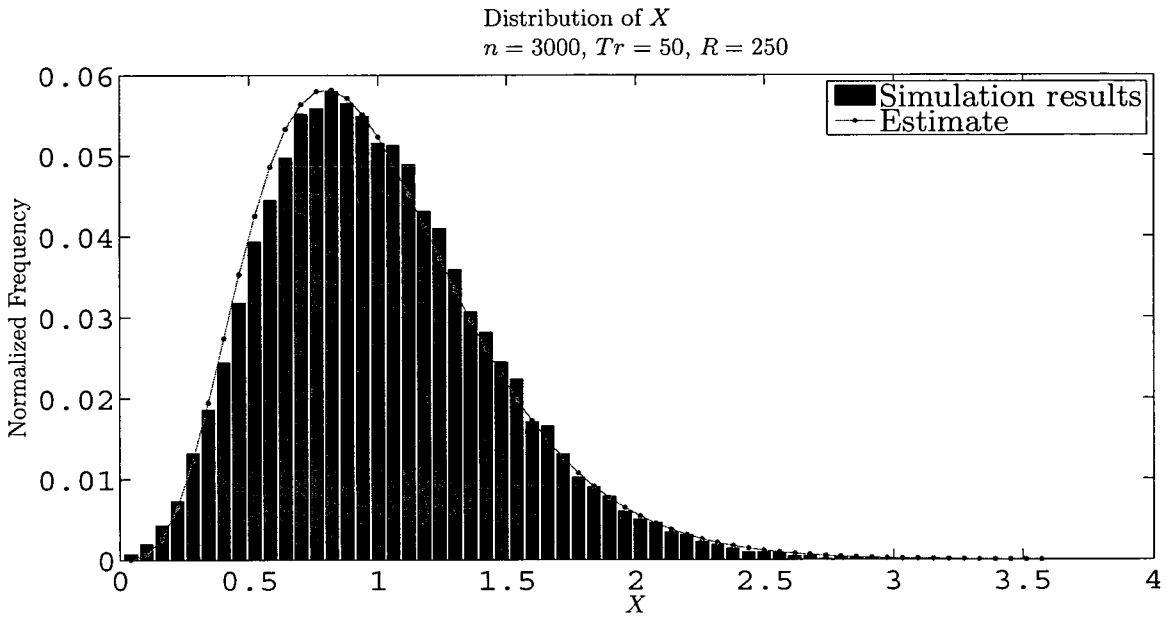


Figure 17: Simulation results for the distribution of X.

as follows:

$$
\begin{aligned}
E\left[\max_{v_i \in V}\left(load_{greedy}(v_i)\right)\right] = & \\
& E\left[\max_{v_i \in V}\left(\frac{load_{greedy}(v_i)-(n-1)}{\overline{\xi}_{\|v_i,O\|,0-(n-1)}}\left(\overline{\xi}_{\|v_i,O\|,0}-(n-1)\right)+n-1\right)\right] \\
= & \; E\left[\max_{v_i \in V}\left(X_i \times (\overline{\xi}_{\|v_i,O\|,0}-n+1)\right)\right]+n-1 \\
= & \max_{0\le a\le R}\left(E\left[\max_{v_i \in V_a}\left(X_i \times (\overline{\xi}_{\|v_i,O\|,0}-n+1)\right)\right]\right)+n-1 \qquad (16) \\
\ge & \max_{0\le a\le R}\left(E\left[\max_{v_i \in V_a}(X_i) \times \left(\min_{v_i \in V_a}(\overline{\xi}_{\|v_i,O\|,0})-n+1\right)\right]\right)+n-1 \\
\ge & \max_{0\le a\le R}\left(E\left[\max_{v_i \in V_a}(X_i) \times (\overline{\xi}_{a,0}-n+1)\right]\right)+n-1 \qquad (17) \\
= & \max_{0\le a\le R}\left(E\left[\max_{v_i \in V_a}(X_i)\right] \times (\overline{\xi}_{a,0}-n+1)\right)+n-1 \qquad (18)
\end{aligned}
$$

where the equality in (16) follows from the fact that $\forall 0 \le a \le R \rightarrow V_a \subseteq V$ and also the inequality in (17) follows from Eq. 13.

We have calculated $\overline{\xi}_{a,0}$ for $0 \le a \le R$ in the previous section (Eq.12). Here we will calculate $\max_{v_i \in V_a}(X_i)$. To simplify equations we assume that X_i, X_j are

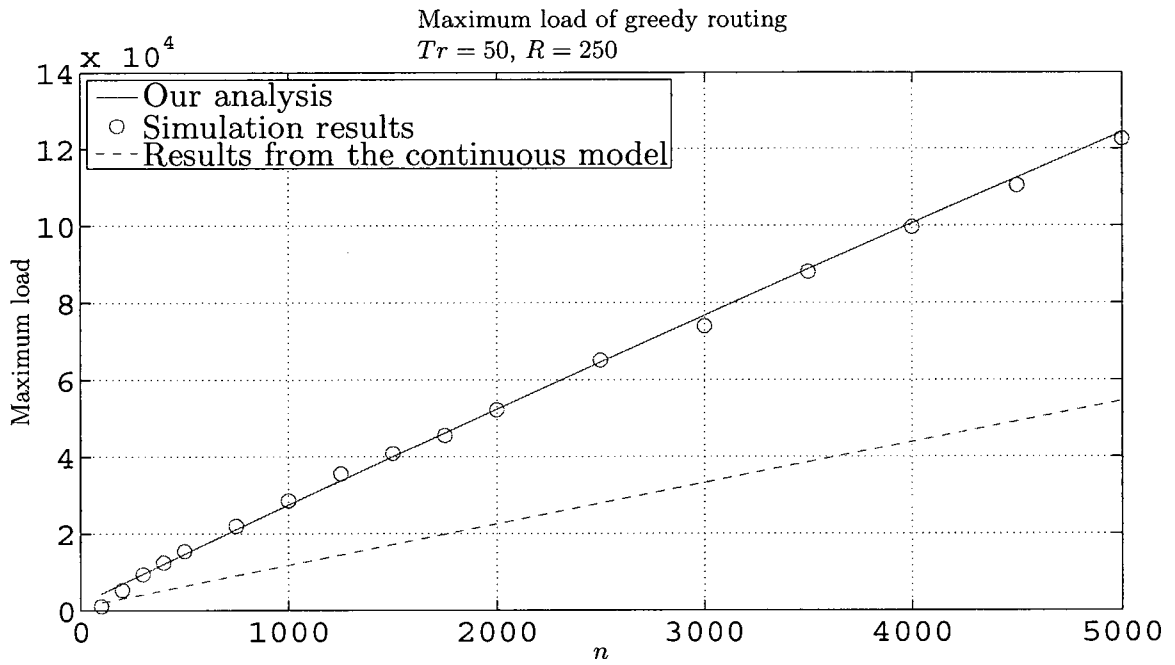


Figure 18: Simulation versus analytical results of the maximum load of greedy routing.

independent of each other for all pairs (v_i, v_j).

$$\begin{aligned} E\left[\max_{v_i \in V_a}(X_i)\right] &= \int_0^\infty 1 - Prob\left[\max_{v_i \in V_a}(X_i) \leq m\right] dm \\ &= \int_0^\infty 1 - Prob\left[\bigcap_{v_i \in V_a} X_i \leq m\right] dm \\ &= \int_0^\infty 1 - \prod_{v_i \in V_a} Prob\left[X_i \leq m\right] dm \\ &= \int_0^\infty 1 - Prob\left[X \leq m\right]^{n_a} dm \\ &= \int_0^\infty 1 - F_X(m)^{n_a} dm \end{aligned} \tag{19}$$

Substituting for $F_X(m)$ from Eq. 15 we will get our estimate of $E\left[\max_{v_i \in V_a}(X)\right]$ as follows:

$$E\left[\max_{v_i \in V_a}(X)\right] = \int_0^\infty 1 - \left[\frac{\gamma(k, \frac{m}{\theta})}{\Gamma(k)}\right]^{n_a} dm$$

Putting this result into Eq.18 we get our estimate for the maximum load of nodes in

the network:

$$E\left[\max_{v_i \in V}(load_{greedy}(v_i))\right] \geq \max_{0 \leq a \leq R}\left((\overline{\xi}_{a,0} - n + 1)\int_0^{\infty} 1 - \left[\frac{\gamma(k, \frac{x}{\theta})}{\Gamma(k)}\right]^{n_a} dx\right) + n - 1$$

Using a step value of 1 for a, $E\left[\max_{v_i \in V}(load_{greedy}(v_i))\right]$ is numerically calculated for different numbers of nodes. As shown in Fig. 18, this estimate matches the maximum load as obtained by simulation results. The estimate error is shown in Fig. 19.

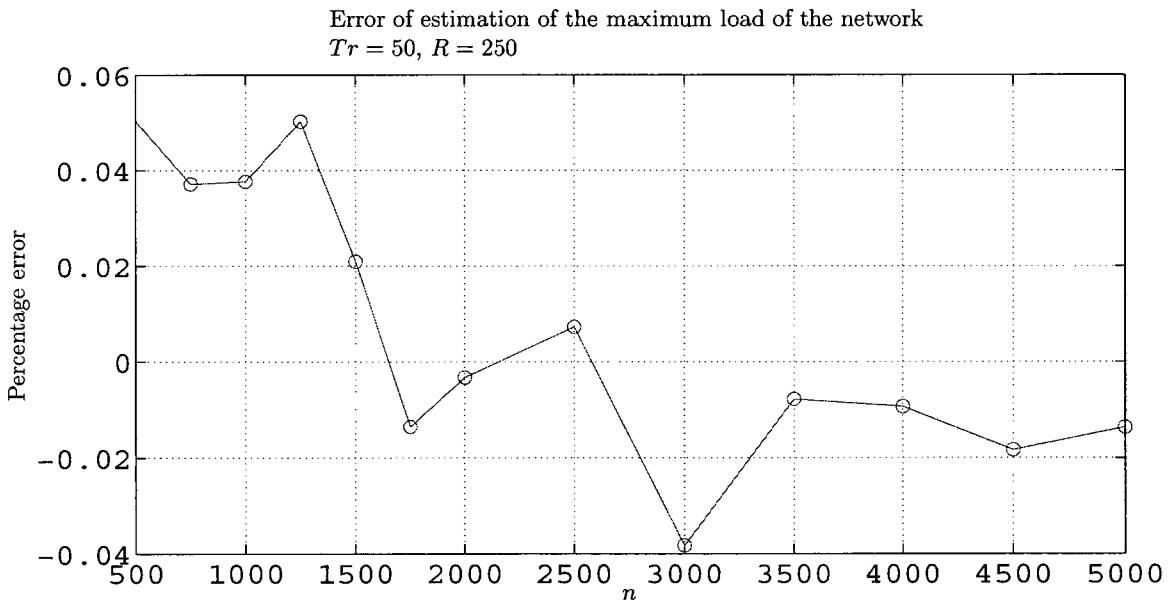


Figure 19: Error of estimate of the maximum load of greedy routing

Using similar techniques, we can calculate the maximum load induced by greedy routing in a square area. As shown in Fig. 20, our estimate matches the maximum load as obtained by simulation results.

3.6 Conclusion

In this chapter, we presented a new technique for calculating the average and the maximum load of the network when greedy routing is used. We calculated the average progress toward the destination in one hop. We provided an approach to

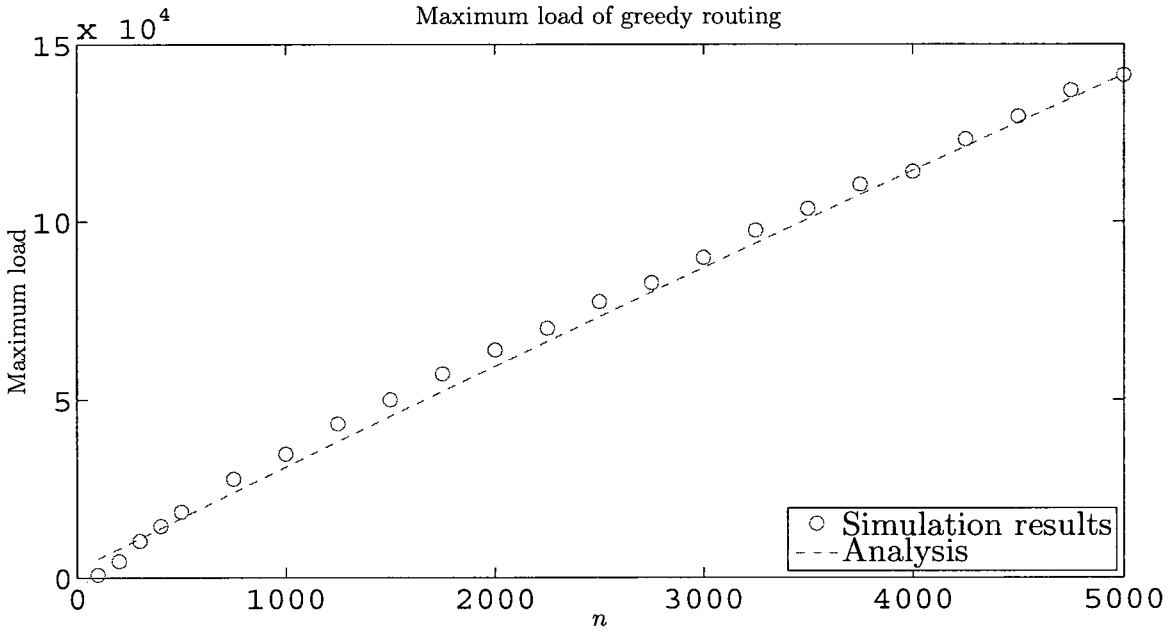


Figure 20: Simulation versus analytical results of the maximum load of the greedy routing (network area is a square with side length $S = 500$ and transmission radius $Tr = 50$).

calculate the average load of the network using the expected progress per hop. We obtained the average load of nodes in any annulus of the network with a given inner radius and width. Finally we introduced a new random variable that can express the characteristics of the load of nodes in the network. Using this random variable, we calculated the maximum load of the network. We also verified our results by simulation.

Chapter 4

Load Balancing Algorithms

4.1 Introduction

In this chapter we focus on the load distribution among the nodes in the network area. In Sections 4.2 and 4.3 We study both the *macroscopic* view and *microscopic* view of the load distribution. When we talk of the macroscopic view of the load distribution, we focus on the expected load of a node which is a function of the node's coordinates in the network. This is in contrast with the microscopic view, where we focus on the one-hop level and analyze the distribution of load among close nodes that have same expected load. In Section 4.4.1 we introduce a technique to reduce the variance of the load distribution and hence the maximum load of the nodes in the network. In Section 4.4.2 we give a class of algorithms called *elliptic routing* to decrease the maximum load of the nodes in the network. We also show how the performance of our algorithms can be improved by combining them with the introduced variance reducing technique.

4.2 Macroscopic View of Load Distribution

In the macroscopic view of the network, paths are lines connecting the source node to the destination node. The expected load of a node is calculated as the number of lines that go through that node (point in the Euclidean plane). In this view the nodes that are very close to each other have the same load in the network. The load of a node can therefore be seen as a function of its location in the network. Many previous studies that used continuous models of the network have used this view [9, 19, 42].

4.3 Microscopic View of Load Distribution

In the microscopic view of the network we focus on a small region of the network that contains nodes with approximately the same expected load. In the previous chapter we introduced a random variable denoted by X_i to explain the distribution of the load of node v_i induced by greedy routing:

$$X_i = \frac{load_{greedy}(v_i) - (n-1)}{\overline{\xi}_{\|v_i,O\|,0} - (n-1)}$$

As we saw the parameters of this random variable for any node are independent of the coordinates of that node. Therefore, as in the previous chapter, we use the random variable X to denote any of the X_i's. We also saw how these parameters affect the calculation of the maximum load of greedy routing in the network.

In this section we extend our study to other routing algorithms. We define a similar random variable X_{ra} with respect to routing algorithm ra (e.g. X_{greedy} and $X_{curveball}$ for greedy routing and curveball routing respectively). Intuitively one can see that the expected value of X_{ra} for any routing algorithm is equal to one. Therefore we compare the variance of X_{ra} for different parameters of the network (e.g. density

of nodes and transmission range for variance of X_{greedy}). Also we study the variance of X_{ra} for different routing algorithms and explain the difference.

4.3.1 The effect of network parameters on the variance of X_{greedy}

To see how the network parameters affect the variance of X_{greedy} we performed a series of simulations. In the first set of simulations we keep the network area and transmission range of nodes fixed and observe the effect of node density (total number of nodes in the network divided by the size of the network area) on the variance of X_{greedy}.

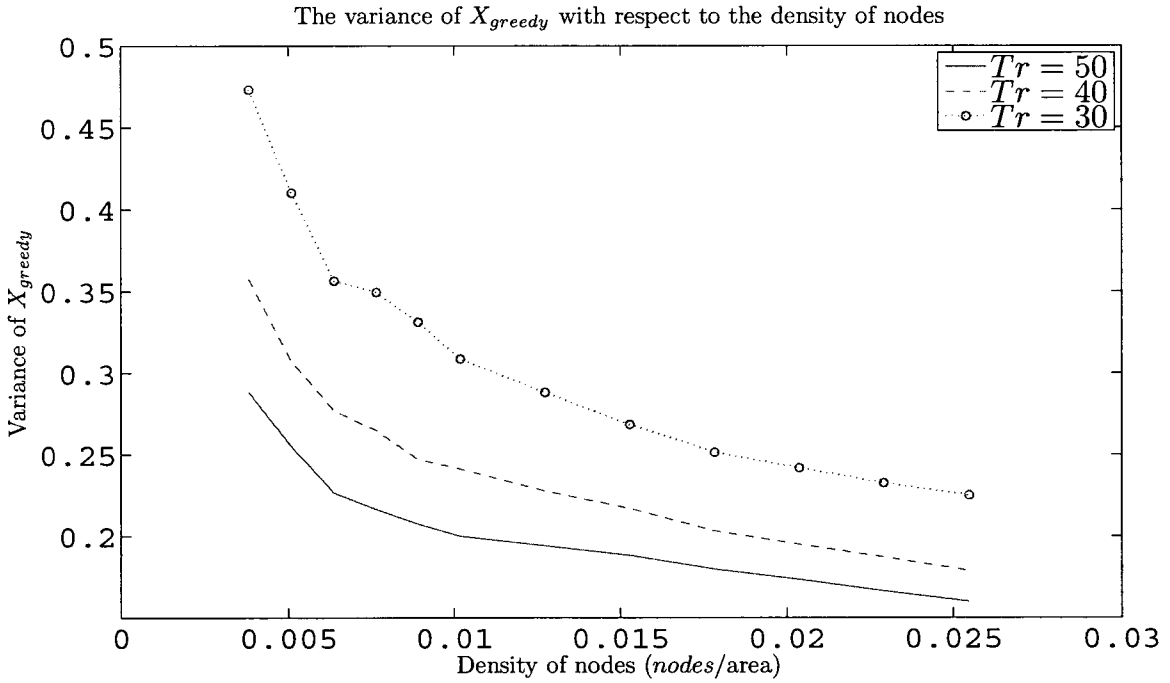


Figure 21: The variance of X_{greedy} with respect to the density of nodes in the network and different transmission ranges. The network area is circular with radius 250.

As the results in Figure 21 show, the variance of X_{greedy} decreases by the increase in the node density of the network. In the second set of simulations we fix the number of nodes and the network area and see how changes in transmission range of nodes

affect the random variable X_{greedy}. As can be seen from the figure, the variance decreases as the transmission range of nodes increases. We can combine the last two parameters in the *average number of neighbors* for every node in the network.

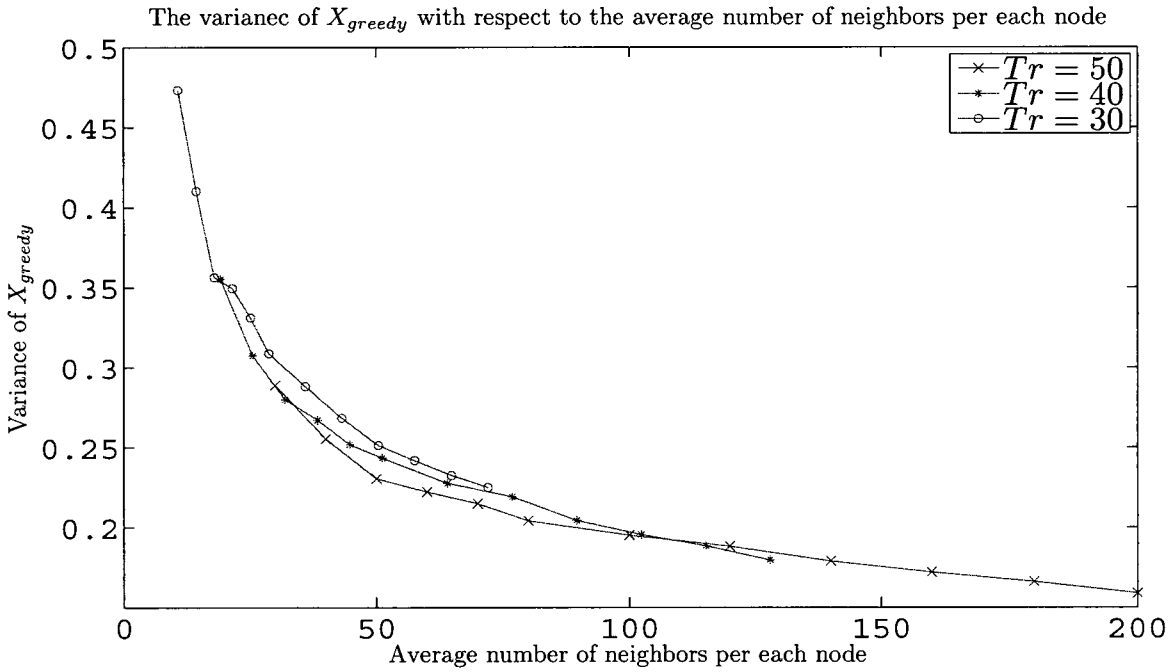


Figure 22: The variance of X_{greedy} with respect to the average number of neighbors per each node. The network area is circular with radius 250.

As shown in Figure 22, the variance of X_{greedy} decreases when the average number of neighbors per each node increases. One possible explanation for this effect is that as the number of nodes in the network increases, the probability that one node is left alone (that will lead to high load) will be lower. In the extreme case if the density of nodes is too high all nodes in a small region of the network will be in a similar condition and hence will have similar load which translates to very low (theoretically zero) variance of X_{greedy}.

4.3.2 The effect of routing algorithms on the variance of X

Different routing algorithms can also affect the variance of X. Figure 23 shows the difference in the variance of X for different routing algorithms and different density of nodes in the network. From the result we can see that the algorithms can be classified into two groups: Group 1 consists of greedy, and curveball routing, and Group 2 consists of one-turn rectilinear and radial paths routing. Clearly the algorithms in Group 1 have lower variance in X_{ra} compared to those in Group 2. In what follows, we hypothesise the reason for this difference.

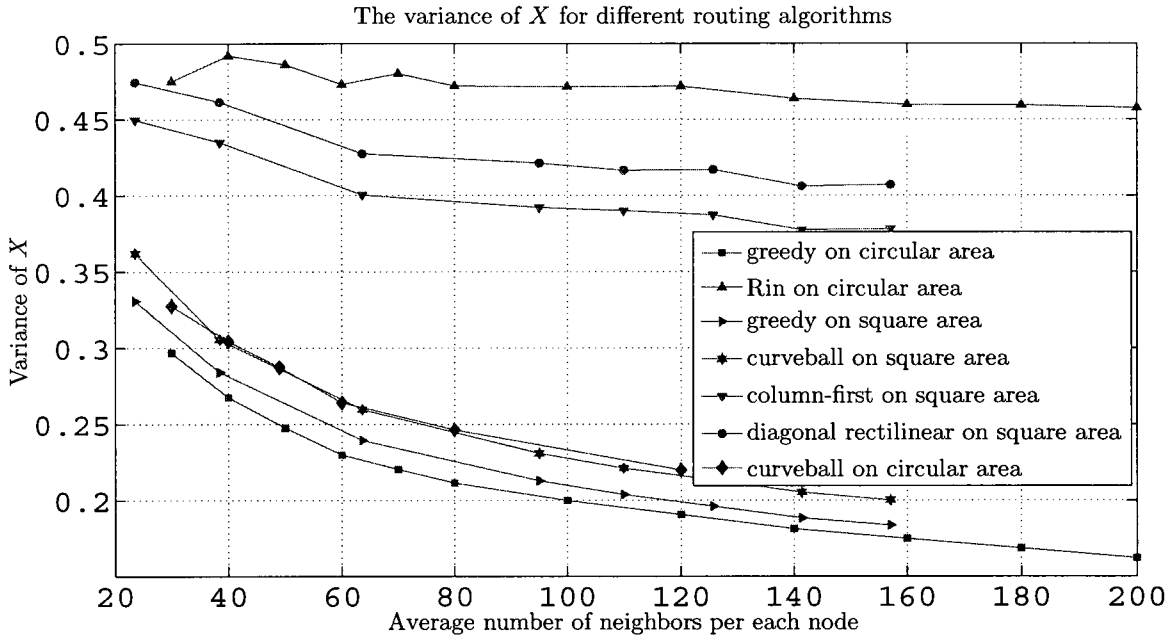


Figure 23: The variance of X_{ra} for different routing algorithms. The network area is circular with radius 250 or square with side length 500, and the transmission range of the nodes is 50.

In all the algorithms, at each hop the current node v forwards the packet to one of its neighbors u as determined by a criterion such as remaining distance to the destination as in greedy, or the best node along a given direction as in rectilinear or radial routing. We observe that in the algorithms in Group 2, the set of candidate nodes for the next node u is in general a strict subset of the neighbors of the node.

For example in one-turn rectilinear routing, for any node v (except for the border of the network) the *direction* of forwarding is either horizontal or vertical. In the case of radial-paths routing, for any node v, the directions are limited to either radial (toward or away from the origin) or on a ring that goes through the point (which is perpendicular to the radial direction).

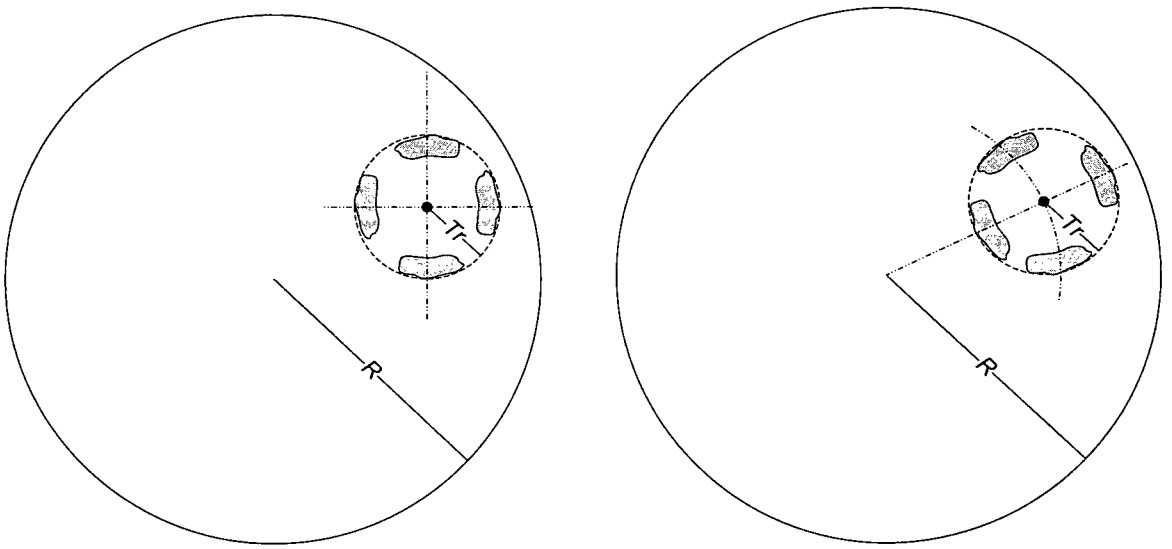


Figure 24: The area around a node which the next hop can be expected to lie (all possible directions for the destination node is considered). (a) one-turn rectilinear routing and (b) Radial routing.

Figure 24 shows the areas in which the next node u can be expected to lie for any destination for rectilinear and radial routing. In contrast, for the algorithms in Group 1, a node can forward a packet to any of its neighbors, depending on the destination, and is not limited to any set of directions.

Hence one can see that each node forwards packets to fewer neighbors for algorithms in Group 2 compared to the algorithms in Group 1. In other words, using the algorithms in Group 2 has a similar effect to reducing the number of neighbors or density for the greedy algorithm. In both cases, there is an increase in the variance of X.

4.3.3 Distribution of X on maximum load of the network

Since the values of the expected load of a node and its actual load are different, we can define different metrics to compare routing algorithms in terms of their load balancing performance. One possible metric is the *maximum of expected load* (*mel* for short) and the other is the *expected maximum load* (*eml* for short). These two metrics evaluate two different aspects of load balancing and could give different results when comparing algorithms.

The *mel* of a network with respect to routing algorithm *ra* is defined as follows:

$$mel_{ra}(V) = \max_{v \in V} E[load_{ra}(v)]$$

where *eml* for the same network is defined as:

$$eml_{ra}(V) = E\left[\max_{v \in V} load_{ra}(v)\right]$$

Theoretically the two metrics will be equal if and only if the variance of X_{ra} is equal to zero (load of every node is equal to its expected load). It is easy to see that $mel_{ra}(V) \leq eml_{ra}(V)$. We prove that in general $\max_{Y_i \in Y = \{Y_1, Y_2, \ldots, Y_n\}} E[Y_i] \leq E[\max Y_i \in Y]$ for any set of random variables Y.

$$\begin{aligned}
\forall Y_i \in Y : \cap_{Y_i \in Y}(Y_i < m) &\subseteq (Y_i < m) \\
Prob[\cap_{Y_i \in Y} Y_i < m] &\leq Prob[Y_i < m] \\
1 - F_{\max_{Y_i \in Y}}(m) &\geq 1 - F_{Y_i}(m) \\
\int_{-\infty}^{\infty} 1 - F_{\max_{Y_i \in Y}}(m)\, dm &\geq \int_{-\infty}^{\infty} 1 - F_{Y_i}(m)\, dm \\
E[\max Y_i \in Y] &\geq E[Y_i] \\
E[\max Y_i \in Y] &\geq \max_{Y_i \in Y} E[Y_i] \qquad (20)
\end{aligned}$$

4.4 Our Load Balancing Algorithms

In this section we introduce new techniques to balance the load over the nodes in the network area. First we give a technique to reduce the variance of the X_{ra} and hence the eml_{ra} value of the network. The effect of this technique on the expected load of nodes is negligible. Secondly, we introduce some routing algorithms that reduce the *mel* value of the network. The first technique can be combined with the new routing algorithms, or indeed any single-path routing algorithm to obtain a more balanced distribution of load.

4.4.1 Technique to reduce the variance of load distribution

As seen before, the variance of X_{ra} is a function of average number of neighbors that interact with a node. The technique in this section tries to increase the average number of neighbors that communicate with a node.

kBestNeighbor

In greedy routing, at each hop the neighbor that minimizes the Euclidean distance to destination is chosen. Given a node v and considering all possible destinations, we see that many neighbors of v are not used as the next hop for any of the destinations. To address this shortcoming, we propose a generalization of the greedy algorithm called *kBestGreedy* algorithm. This technique has been proposed previously to increase the delivery rate of greedy routing [16]. In this algorithm, at each node, a candidate set of the k neighbors nearest to the destination and closer to the destination than the current node is constructed. In case that only $k' < k$ neighbors are closer to the destination node than the current node, the candidate set consists of those k' neighbors only. Then the next node is chosen randomly from this set. Algorithm 1 shows *kBestGreedy* in more detail.

Algorithm 1 kBestGreedy

procedure kBestGreedy($curr, dest, k$)
  if dist($curr, dest$) $\leq Tr$ **then**
    return $dest$
  end if{else}
  $c\_list \leftarrow N_{curr}$
  for all $v \in c\_list$ **do**
    $dist[v] \leftarrow$ dist($v, dest$)
    if dist($curr, dest$) $\leq dist[v]$ **then**
      remove v from c_list
    end if
  end for
  for $i = 1$ to size(c_list) $- k$ **do**
    $u \leftarrow$ vertex in c_list with biggest $dist$
    remove u from c_list
  end for
  $index \leftarrow$ rand(size(c_list)) {a random number between 1 and size(c_list)}
  return $c\_list[index]$

On average this technique leads to more nodes being used at the one hop scale and therefore the variance of X_{ra} is expected to decrease. Simulation results for $var(X_{greedy})$ are shown in Figure 25 that confirms the predictions. As k increases the expected per hop progress goes down, which causes the expected load of nodes to increase. Figure 26 and 27 show the *eml* and the average load of the network with *kBestGreedy* for different values of k respectively. The best parameter can be chosen by simulations, but even $k = 2$ leads to $17 - 20\%$ decrease in *eml* and only $1 - 7\%$ increase in the average load.

4.4.2 Algorithms for balancing the expected load of nodes

In this section we introduce a set of algorithms called *elliptic routing*. We start with *one-level elliptic routing* which uses two parameters and then extend our algorithm to *k-level elliptic routing* that works with $2k$ parameters.

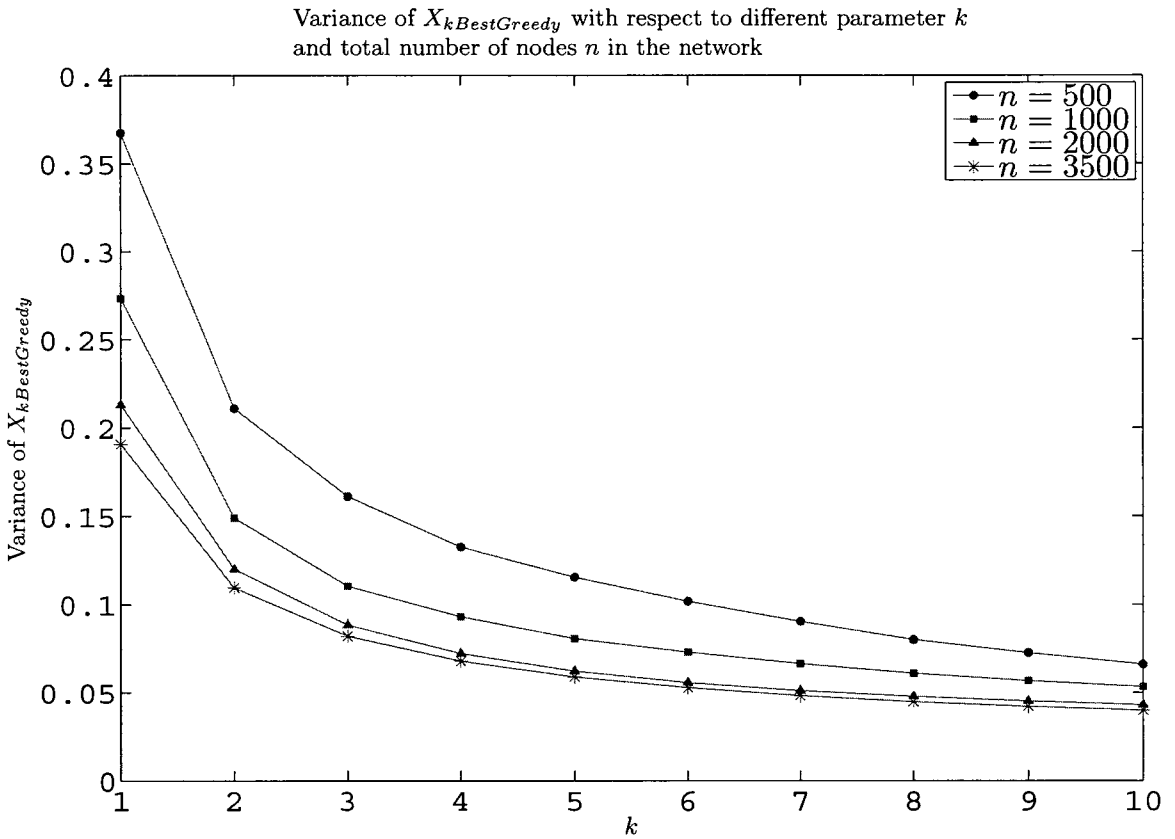


Figure 25: The variance of $X_{kBestGreedy}$ for different values of k. The number of nodes in the simulations is 1225, network area is circular with radius 250 and transmission range of nodes is 50.

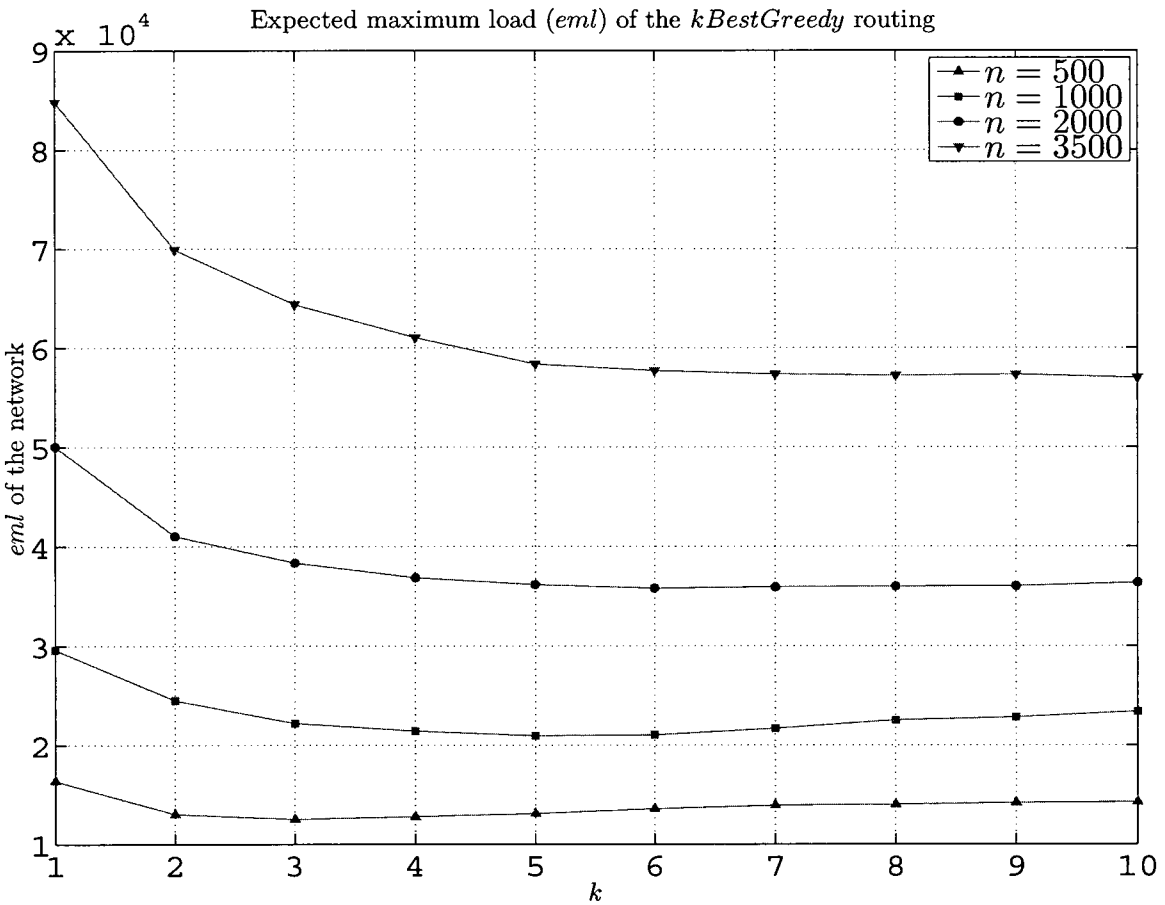


Figure 26: The *eml* of the network of *kBestGreedy* for different values of k and different number of nodes in the network. Network area is circular with radius 250 and transmission range of nodes is 50.

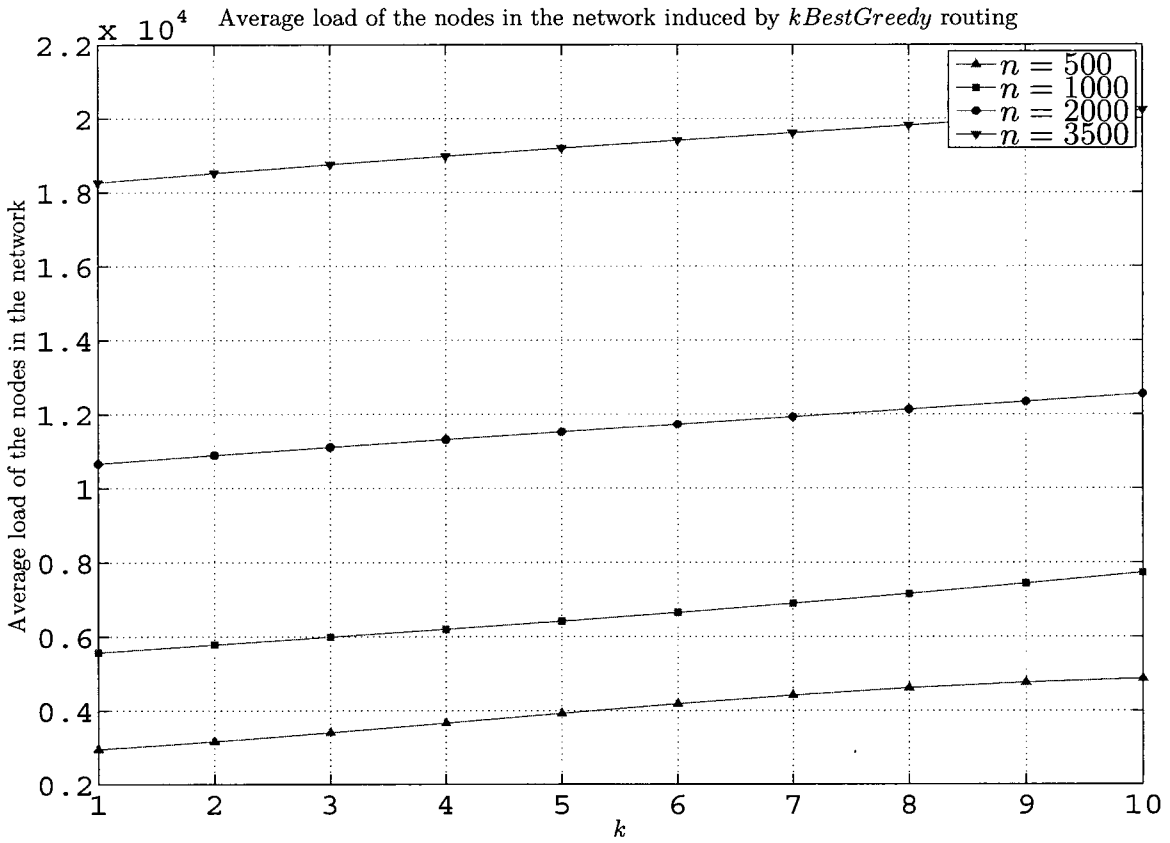


Figure 27: Average load of the network of *kBestGreedy* for different values of k and different number of nodes in the network. Network area is circular with radius 250 and transmission range of nodes is 50.

One-level elliptic routing

This algorithm takes as parameters a real value $0 \leq r < R$ where R is the network radius, and a function $f : R \times R \rightarrow R$. The network area is divided into two regions: the inner region within distance r from the network center and the outer region outside it. If both source and destination nodes are in the inner region, the routing path is the same as in greedy routing. The reason not to change the path is that when both nodes are in the highly loaded area (close to network center), increasing the path length increases the average of the load of nodes in this area which consequently increases the *mel* of the network (Figure 28a illustrates the path for this case). But when either the source node or the destination node (or both) are in the outer region, the route goes through an intermediate point to avoid the network center. Let l denote the line connecting the source node (*src*) and the destination node (*dest*). Also let l' denote the line that goes through the network *origin* and is perpendicular to l. If the intersection of l and l' does not lie between *src* and *dest* nodes on l, the path is again as in greedy routing (see Figure 28d). Otherwise we select an intermediate point *intPoint* on l' with distance $r_{intPoint}$ to the *origin*, defined by the parameter function f. Figure 28b and 28c illustrate how the intermediate point is selected. For the pseudocode of the algorithm, see Algorithms 2 and 3.

When the intersection point of l and l' in not between the source and the destination node we do not set an intermediate point. This includes many cases in which either no part of l lies in the highly loaded area or any deviation from the straight line path does not lead to a significant decrease in the length of the new path that falls in the highly loaded area (one example is shown in Figure 28d).

For our simulations we tried three different functions as candidates for f:

$$\min(d_1, d_2), \text{ mean}(d_1, d_2), \text{ and } \max(d_1, d_2)$$

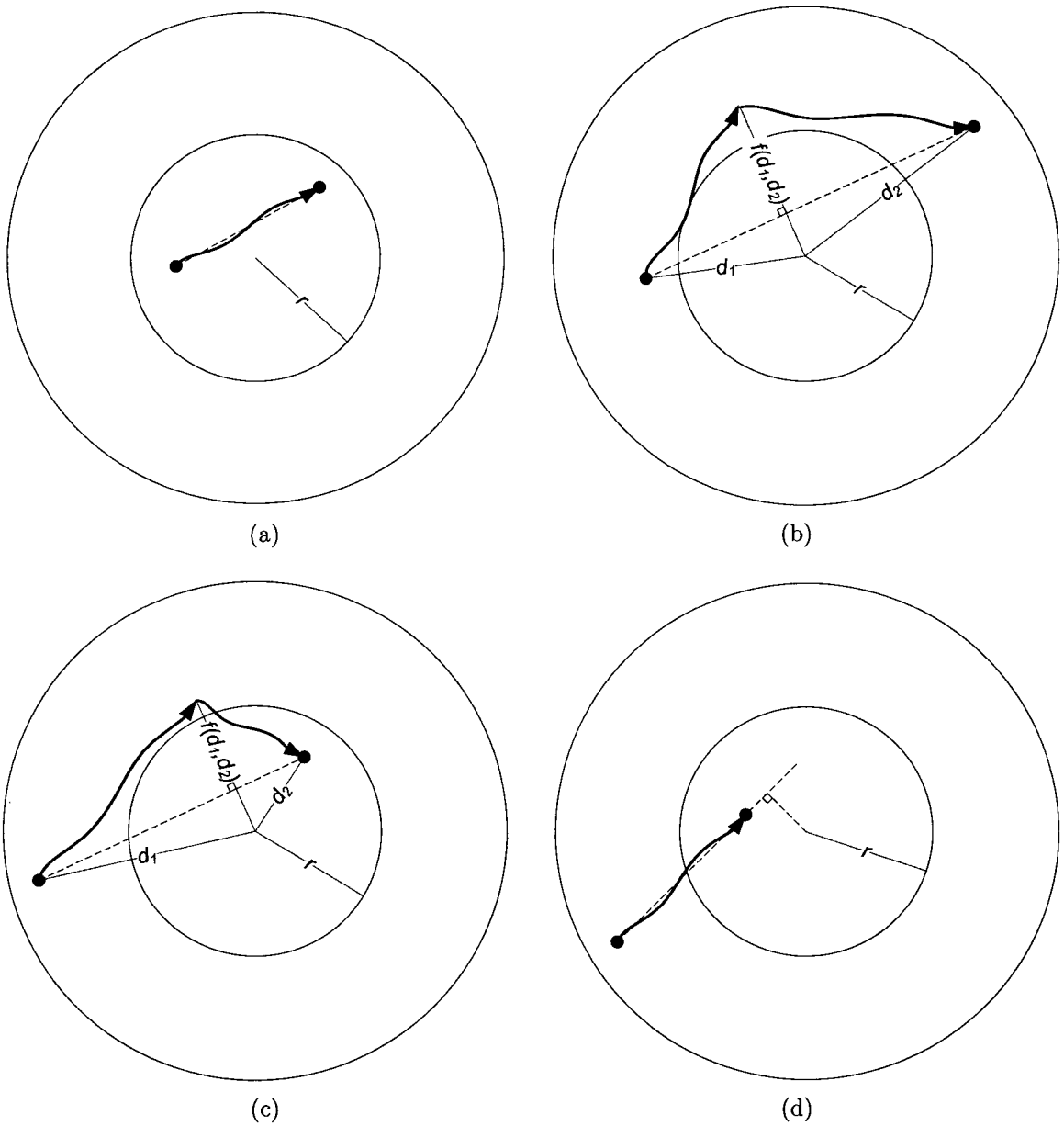


Figure 28: Path selection in *one-level elliptic routing:* (*a*) When both nodes are within distance r from the network center, (*b*) and (*c*) at least one of the source or destination nodes is at distance greater than r, and the intersection of l and l' is between the source and the destination nodes. An intermediate point is chosen as illustrated. (*d*) The intersection point is not between the source and destination and no intermediate point is assigned.

Algorithm 2 one-level elliptic routing

function 1levelelliptic(src, des, r, f)

```
path ← (src)
if dist(src, dest) ≤ Tr then
  path ← (path, dest)
  return path
end if
{else}
curr ← src
intPoint ← assignIntPoint(src, dest, r, f)
{use greedy path to a node within the transmission range of intPoint}
while dist(curr, intPoint) > Tr do
  curr ← greedy(curr, intPoint)
  path ← (path, curr)
end while
{use greedy path to a node within the transmission range of dest}
while dist(curr, dest) > Tr do
  curr ← greedy(curr, dest)
  path ← (path, curr)
end while
path ← (path, dest)
return path
```

function assignIntPoint($src, dest, r, f$)

```
if dist(src, origin) > r or dist(dest, origin) > r then
  calcIntPoint(src, dest, f, intPoint)
  return intPoint
else
  return dest
end if
```

Algorithm 3 calculates intermediate point

procedure calcIntPoint($p1, p2, f, intPoint$)
  $(x1, y1) \leftarrow$ cartesianCoordinates($p1$)
  $(x3, y3) \leftarrow$ cartesianCoordinates($p2$)
  $l \leftarrow$ line going through $(x1, y1)$ and $(x3, y3)$
  $l' \leftarrow$ line going through *origin* and perpendicular to l
  $(x2, y2) \leftarrow$ intersect(l, l')
  if $(x1 \leq x2 \leq x3$ and $y1 \leq y2 \leq y3)$ or $(x1 \geq x2 \geq x3$ and $y1 \geq y2 \geq y3)$ **then**
    $(r1, \theta1) \leftarrow$ polarCoordinates($x1, y1$)
    $(r2, \theta2) \leftarrow$ polarCoordinates($x2, y2$)
    $(r3, \theta3) \leftarrow$ polarCoordinates($x3, y3$)
    $r \leftarrow f(r1, r3)$
    $\theta \leftarrow \theta2$
    $intPoint \leftarrow$ polar2cartesian(r, θ)
  else
    $intPoint \leftarrow p2$
  end if

where d_1 =dist($src, origin$) and d_2 =dist($dest, origin$). The value of r that minimizes the metric *mel* is in general different from the value that minimizes the metric *eml*. Therefore for each combination of function f and metric the best values of r can be found by simulations. Table 1 shows these values for different combinations.

f	r that minimizes *eml*	(eml, mel) with r	r' that minimizes *mel*	(eml, mel) with r'
min	215	(27199, 9859)	210	(27377, 9610)
mean	225	(26708, 10377)	215	(26967, 9395)
max	225	(25999, 10353)	215	(28666, 9247)

Table 1: *One-level elliptic routing*: Best values of r and r' for different functions f that minimizes the *eml* and *mel* metrics respectively, as found by simulations. The number of nodes in the simulations is 1225, network area is circular with radius 250 and transmission range of nodes is 50.

Figure 29 shows the expected load of nodes as a function of their distance from the network center for *one-level elliptic routing* for a given function parameter f = max. One can see that as the parameter r of the algorithm decreases, more paths will go through intermediate points and hence the load in the nodes further from the

network center will increase. In contrast as the parameter r increases, it means more nodes participate in greedy routing, which results in an increase in the load of nodes close to the network center. The same pattern can be expected to occur for other functions $f = \{\text{min, mean}\}$ as well.

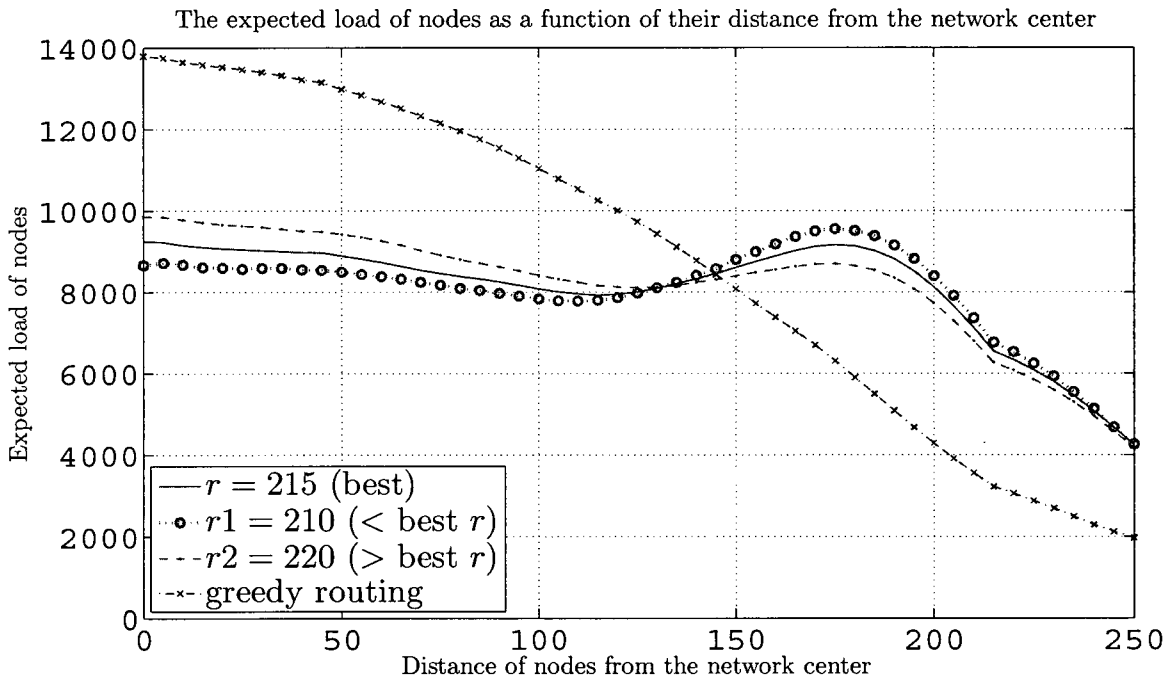


Figure 29: The expected load of nodes as a function of distance from the network center (f = max). The number of nodes in the simulations is 1225, network area is circular with radius 250 and transmission range of nodes is 50. To calculate the expected load average is taken over load of the nodes in an annulus with width 35.

Figure 30 shows the expected load of a node as a function of distance from the network center for different function parameters $f = \{\text{min, mean, max}\}$ and a fixed r parameter. For a fixed parameter r, using f = max pushes the load more outward the network center compared to f = mean and f = min. Hence the nodes close to the network center experience less load and in nodes further from the network center encounter more load. A similar intuition explains the difference in the load distribution of f = min and f = mean.

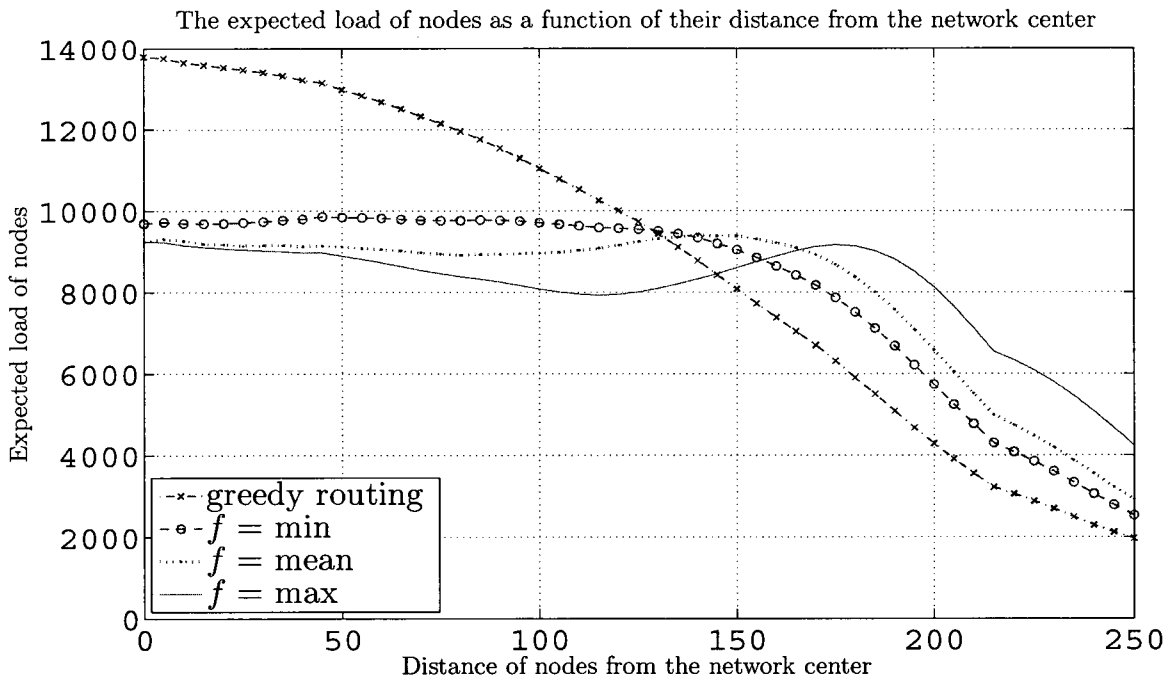


Figure 30: The expected load of nodes as a function of distance from the network center for different function parameters $f = \{\text{min, mean, max}\}$ (parameter $r = 215$ is fixed). The number of nodes in the simulations is 1225, network area is circular with radius 250 and transmission range of nodes is 50. To calculate the expected load average is taken over load of the nodes in an annulus with width 35.

Multi-level elliptic routing

In *one-level elliptic routing* by assigning only one intermediate point we try to push the path outward from the network center. However in some cases, a big portion of the path still lies inside the highly loaded area (the area close to the network center). Figure 31 shows an example where in spite of using an intermediate point, a major part of the path between the source and the destination nodes falls close to the network center. This shows that we can still improve on *one-level elliptic routing* by adding more intermediate points that push the path further out of the highly loaded area close to the network center.

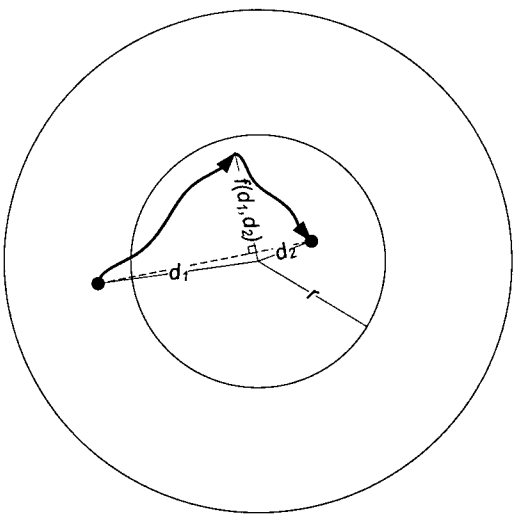


Figure 31: An example where, curving the path via an intermediate node increases the length of the path that falls in the highly loaded area.

To extend the algorithm for k-levels, we use two sets of parameters $r = [r_1, r_2, ..., r_k]$ and $f = [f_1, f_2, ..., f_k]$. The $0 \leq r_i < R$ are elements of a sorted array of distinct reals that are used to divide the network area into disjoint regions. The functions $f_i : R \times R \rightarrow R$ are used to assign the intermediate points. The decision of the function to use is made based on the regions where the source node and the destination node are located. In the first stage of the algorithm the number of intermediate

points is determined. We find the largest i such that at least one of the source or the destination nodes is at distance greater than r_i to the network center. The number of intermediate points is then calculated as $2^i - 1$. The coordinates of the intermediate points are calculated and stored in an array *intPoints* of size $2^i - 1$ using a recursive algorithm. First the middle element (i.e. $intPoints[2^{i-1}]$) is calculated as in Algorithm 6 using the given source and destination. Next the algorithm recursively fills in the first half of the *intPoints* array using the given source and $intPoints[2^{i-1}]$ as destination, and similarly recursively fills in the second half of the *intPoints* array using $intPoints[2^{i-1}]$ as source and the given destination. The pseudocode of *multi-level elliptic routing* is provided in Algorithms 4, 5, and 6.

Algorithm 4 Multi-level elliptic routing

```
function multilevelelliptic(src, des, r[], f[])
  path ← (src)
  if dist(src, dest) ≤ Tr then
    path ← (path, dest)
    return path
  end if
  {else}
  curr ← src
  intPoints ← assignIntPoints(src, dest, r, f)
  for i = 1 to size(intPoints) do
    {use greedy path to a node within the transmission range of intPoint[i]}
    while dist(curr, intPoints[i]) > Tr do
      curr ← greedy(curr, intPoints[i])
      path ← (path, curr)
    end while
  end for
  {use greedy path to a node within the transmission range of dest}
  while dist(curr, dest) > Tr do
    curr ← greedy(curr, dest)
    path ← (path, curr)
  end while
  path ← (path, dest)
  return path
```

One can see that *one-level elliptic routing* is a special case of *multi-level elliptic*

Algorithm 5 Assigns intermediate points

function assignIntPoints($src, dest, r[], f[]$)
  $k \leftarrow$ size(r)
  $level \leftarrow 0$
  $numberIntPoints \leftarrow 0$
  for $i = k$ downto 1 **do**
    if dist($src, origin$) $>$ $r[i]$ or dist($dest, origin$) $>$ $r[i]$ **then**
      $level \leftarrow i$
      $numberIntPoints \leftarrow 2^i - 1$
      break;
    end if
  end for
  if $level == 0$ **then**
    $intPoints \leftarrow$ an empty array of size 1
    $intPoints[1] \leftarrow dest$
  else
    $intPoints \leftarrow$ an empty array of size $numberIntPoints$
    calcIntPointsArray($src, dest, f[level], intPoints$)
  end if
  return $intPoints$

Algorithm 6 calculate intermediate points

procedure calcIntPointsArray($p1, p2, f, A[]$)
  $midIndex \leftarrow \lceil$size($subArray$)$/2 \rceil$
  calcIntPoint($p1, p2, f, A[midIndex]$)
  if size(A) > 1 **then**
    $p3 \leftarrow A[midIndex]$
    calcIntPointsArray($p1, p3, f, A[1..midIndex - 1]$)
    calcIntPointsArray($p3, p2, f, A[midIndex + 1..end]$)
  end if

routing where $size(f) = size(r) = 1$. In this thesis we looked at another special case of the general algorithm where $size(f) = size(r) = 2$. Figure 32 shows an illustration of *two-level elliptic routing*. If both nodes are within distance r_1 from the network center, the path is as in greedy routing (Figure 32a). Otherwise, if both nodes are within distance r_2 from the network center, similar to *one-level elliptic routing*, an intermediate point is selected and the path contains two segments: first to the intermediate point and second from the intermediate point to the destination node (as shown in Figure 32b). If the distance of either the source node or destination node from the network center is greater than r_2, up to three intermediate points may be used. First we find the middle intermediate point i_2 using the source node and the destination node with a technique similar to *one-level elliptic routing*. Then the other two intermediate points (i_1 and i_3) are found using i_2 once as the destination node and once as the source node (see Figure 32c and 32d). Depending on the positions of the source node and the destination node, the total number of the intermediate points could be zero, one, two or three. If i_2 does not exist (the perpendicular point falls outside the line segment between the source node and the destination node) we will have no intermediate point. If i_2 exists but neither i_1 or i_3 exist then we will have one intermediate point. If i_2 exists but either i_1 or i_3 do not exist we will have two intermediate point (see Figure 32d for an example), and finally if all of the intermediate points exist then the path will go through three intermediate points (as in Figure 32c).

We tried some of the possible combinations of different functions (as mentioned for *one-level elliptic routing*) for f_1 and f_2 to try to achieve better load balancing compared to *one-level elliptic routing*. Table 2 shows the simulation results for the best parameter sets of the algorithms. As with the simulation results for *one-level elliptic routing*, the best parameters for the *eml* and *mel* metrics are not necessarily

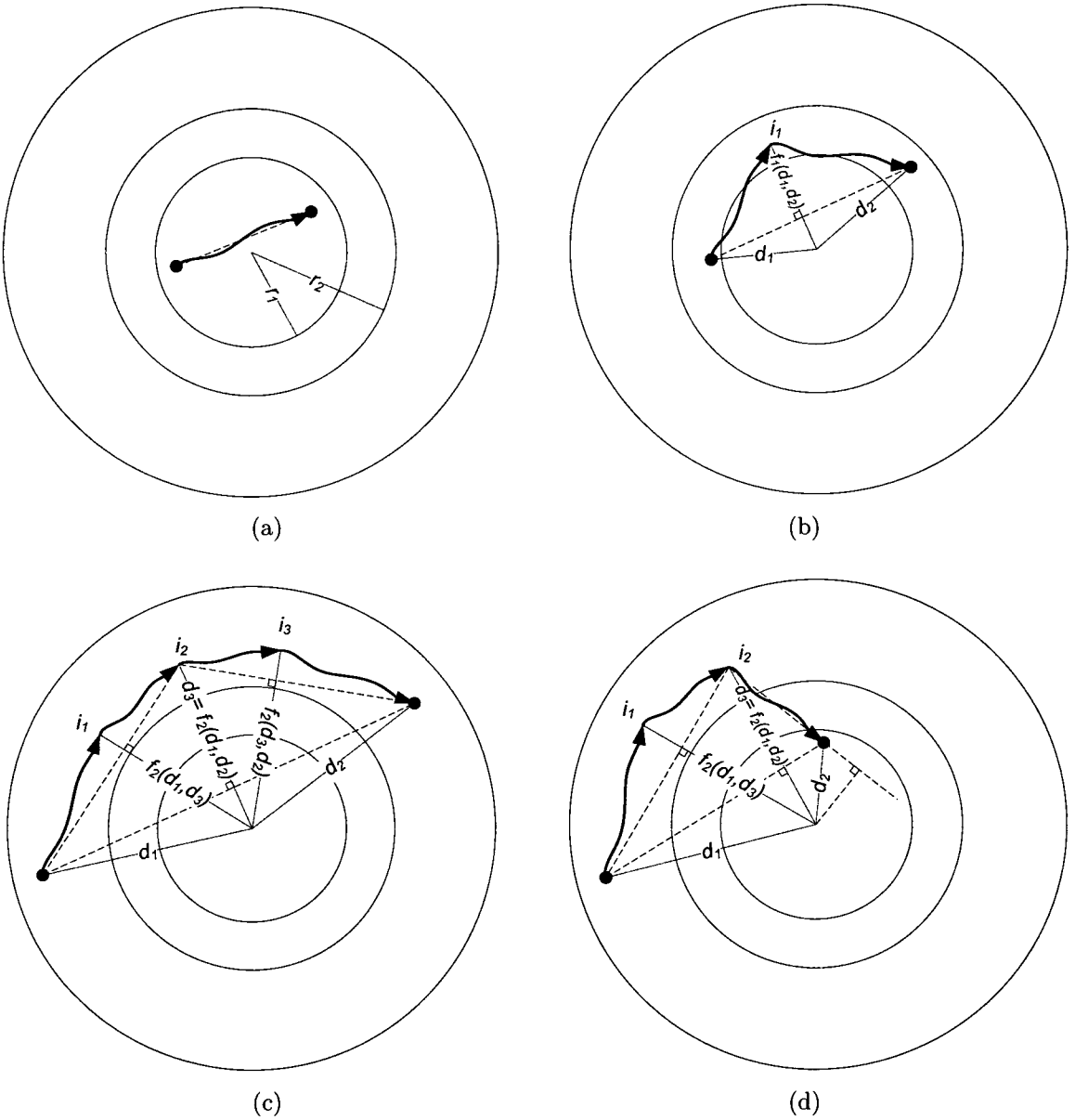


Figure 32: Path selection in *two-level elliptic routing.*

the same. Figure 33 shows the simulation results of the expected load of nodes as a function of their distance from the network center with different algorithms.

(f_1, f_2)	(r_1, r_2) that minimizes *eml*	(eml, mel) with (r_1, r_2)	(r'_1, r'_2) that minimizes *mel*	(eml, mel) with (r'_1, r'_2)
(min,mean)	$(215, 235)$	$(25047, 9497)$	$(205, 225)$	$(27179, 8458)$
(min,max)	$(210, 240)$	$(25110, 9194)$	$(195, 230)$	$(31570, 8235)$
(mean,max)	$(220, 240)$	$(25165, 9816)$	$(205, 230)$	$(32024, 8146)$

Table 2: *Multi-level elliptic routing*: Best values of (r_1, r_2) and (r'_1, r'_2) for different (f_1, f_2) pairs of functions that minimize the *eml* and *mel* metrics respectively, as found by simulations. The number of nodes in the simulations is 1225, network area is circular with radius 250 and transmission range of nodes is 50.

Figure 34 shows how changing the parameter r_1 of the *two-level elliptic routing* affects the expected load of the nodes in the network. Similar to *one-level elliptic routing*, as the parameter r_1 decreases, more paths will go through intermediate points and hence the load in the nodes further from the network center will increase. In contrast as the parameter r_1 increases, it means more nodes participate in greedy routing which results in an increase in the load of nodes close to the network center. Similar results are predicted to occur with changing the parameter r_2. Figure 35 shows the simulation results of changing r_2.

4.4.3 Combining the algorithms

Each of the algorithms in Section 4.4.2 can be combined with the technique in Section 4.4.1 to achieve an improvement in decreasing *eml* of the network. As also discussed in Section 4.4.1 the effect of the introduced technique on *mel* and average load of the nodes should be negligible. Each of the algorithms introduced in Section 4.4.2 uses simple greedy routing as a base to reach the destination or intermediate nodes. To combine the *kBestGreedy* technique with these algorithms, one only needs to replace *kBestGreedy* in the algorithm wherever *greedy* is used. Table 3 compares the

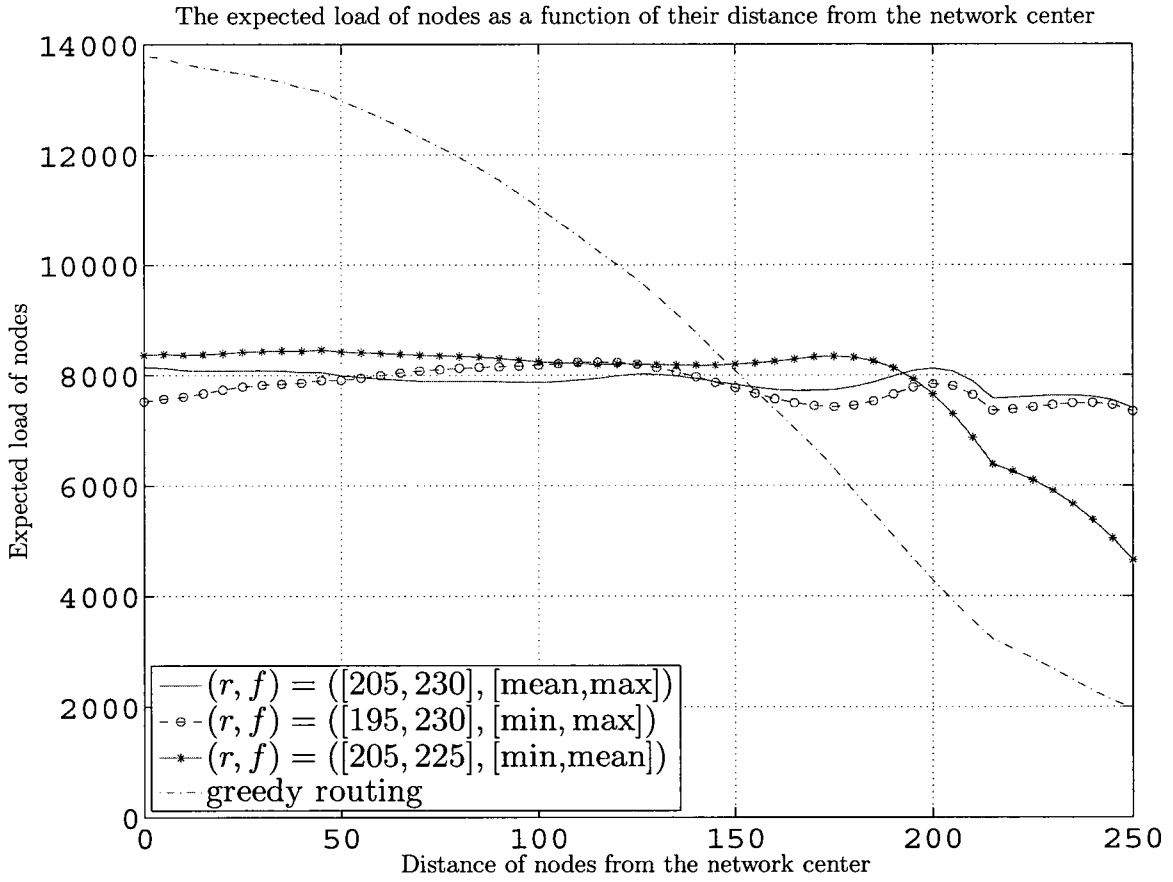


Figure 33: The expected load of nodes as a function of distance from the network center. For each pair of functions (f_1, f_2), the parameters (r_1, r_2) are selected to minimize the *mel* of the network. The number of nodes in the simulations is 1225, network area is circular with radius 250 and transmission range of nodes is 50. To calculate the expected load average is taken over load of the nodes in an annulus with width 35.

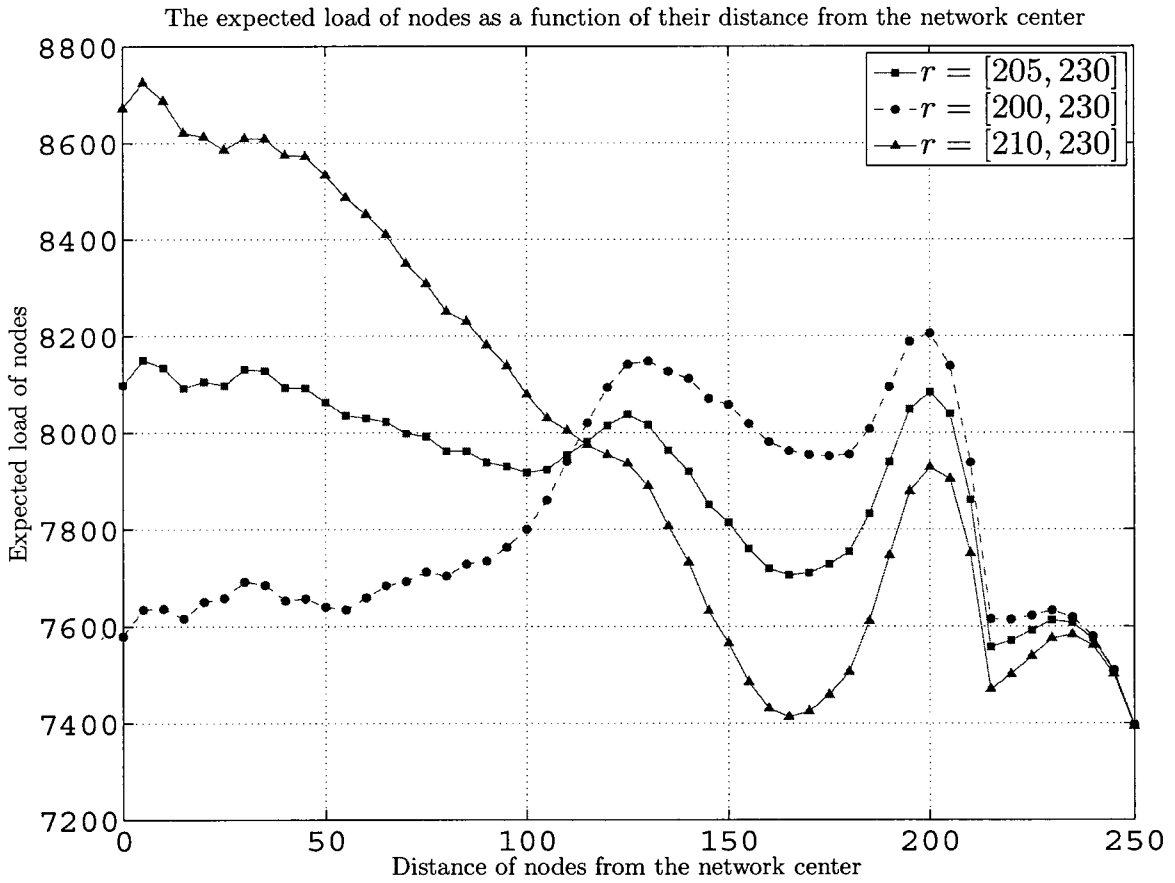


Figure 34: The effect of changing r_1 on the the expected load of nodes as a function of distance from the network center where (f_1, f_2) = (mean, max). The number of nodes in the simulations is 1225, network area is circular with radius 250 and transmission range of nodes is 50. To calculate the expected load average is taken over load of the nodes in an annulus with width 35.

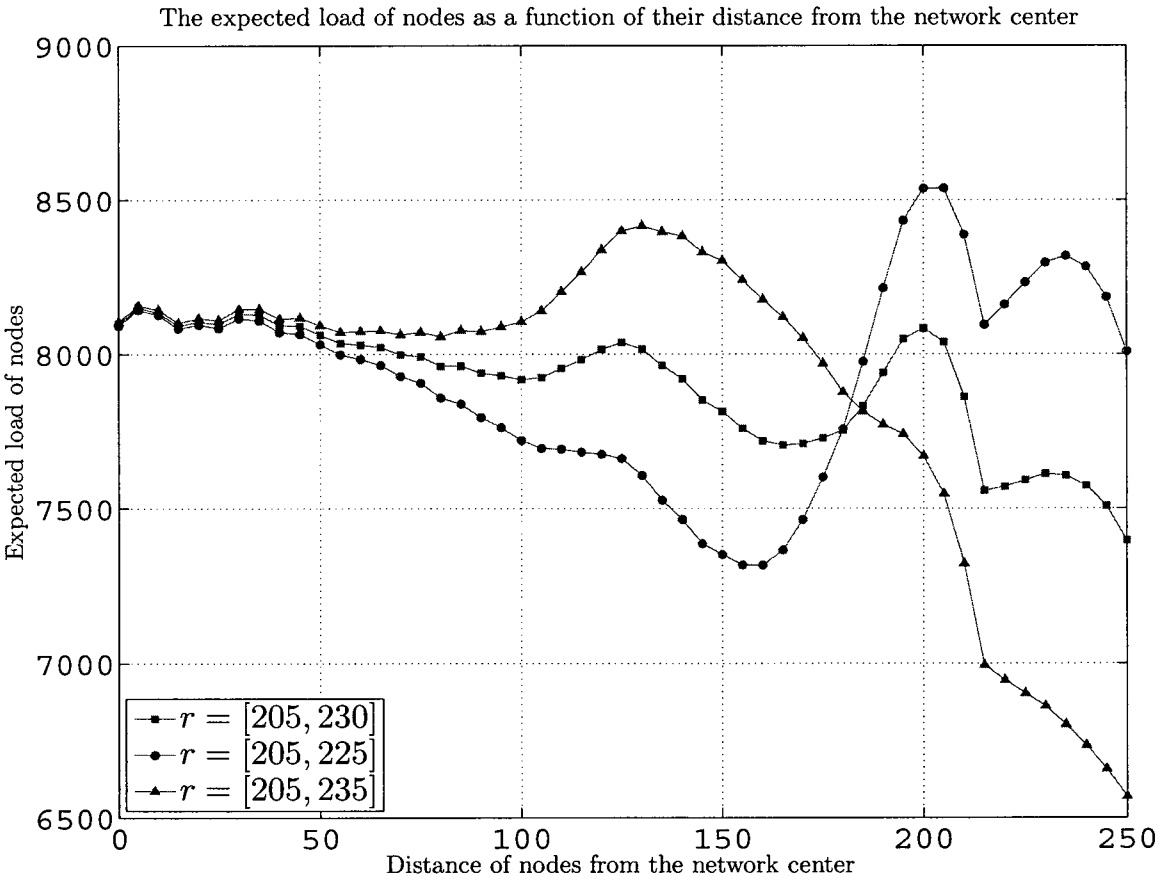


Figure 35: The effect of changing r_2 on the expected load of nodes as a function of distance from the network center where (f_1, f_2) = (mean, max). The number of nodes in the simulations is 1225, network area is circular with radius 250 and transmission range of nodes is 50. To calculate the expected load average is taken over load of the nodes in an annulus with width 35.

performance of *2Best one-level elliptic routing* and *2Best two-level elliptic routing* with the original *one-level* and *two-level elliptic routings*.

algorithm	*eml*	*mel*	*average load*
2Best one-level elliptic compared to *one-level elliptic*	23% ↓	3% ↑	4% ↑
2Best two-level elliptic compared to *two-level elliptic*	25% ↓	3% ↑	4% ↑

Table 3: Comparison of *2Best one-level elliptic routing* and *2Best two-level elliptic routing* with *one-level* and *two-level elliptic routings* in terms of *eml*, *mel*, and *average load* of the network. Parameters (r, f) for *one-level* and *two-level elliptic routings* are $(215, \max)$ and $([205, 230], [\text{mean,max}])$ respectively. The number of nodes in the simulations is 1225, network area is circular with radius 250 and transmission range of nodes is 50.

4.5 Comparisons with previous algorithms

In this section we compare our algorithms with previous algorithms. To compare the different algorithms we use three predefined metrics: *average load*, *eml*, and *mel* of the nodes in the network. We would like to find an algorithm that minimizes *eml* and *mel* of the network while not increasing *average load* of the network too much.

The algorithms that we compare in this section are: *curveball routing* (with input parameters 1.1 and 1.2 that are reported in [43] to have best *eml* and *mel* respectively), *Rin routing*, *greedy routing*, *one-level elliptic routing* with parameters $(r, f) = (225, \max)$ and $(r, f) = (215, \max)$, and *two-level elliptic routing* with parameters $(r, f) = ([215, 235], [\text{min,mean}])$ and $(r, f) = ([205, 230], [\text{mean,max}])$.

Figure 36 shows the expected load of the nodes as a function of their distance to the network center with respect to different algorithms. It can be observed from the graphs that *two-level elliptic routing* decreases *mel* of the network by 46% compared to greedy routing. Also a decrease of 36% in *mel* of greedy routing is obtained by *one-level elliptic routing*. Finally, *two-level elliptic routing* achieves the best *mel* though

only a slight improvement over curveball routing.

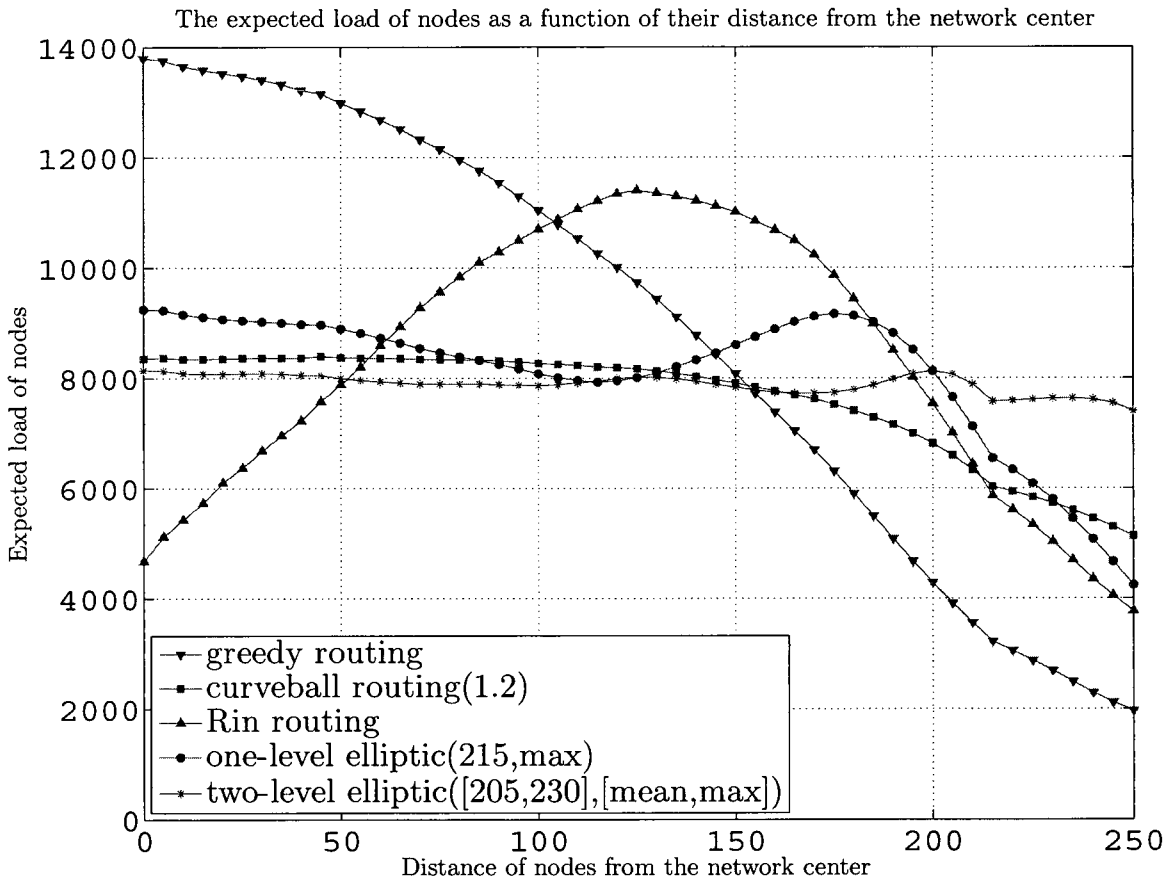


Figure 36: Expected load of nodes as a function of distance from the network center induced by different algorithms. The number of nodes in the simulations is 1225, network area is circular with radius 250 and transmission range of nodes is 50.

Figure 37 compares different algorithms with respect to *eml* of the network for different node densities. It is observed that *Rin* routing induces greater *eml* in the network compared with greedy routing. This is due to the fact that variance of X effects the *eml* of the network (in contrast with *mel*) and because X for these algorithms have higher variance therefore the *eml* of the network is higher for them. Compared to greedy routing, *one-level* and *two-level elliptic routing* can achieve up to 24% and 29% decrease in the *eml* of the network respectively. Finally, *two-level elliptic routing* achieves the best *eml* though only a slight improvement over curveball

routing.

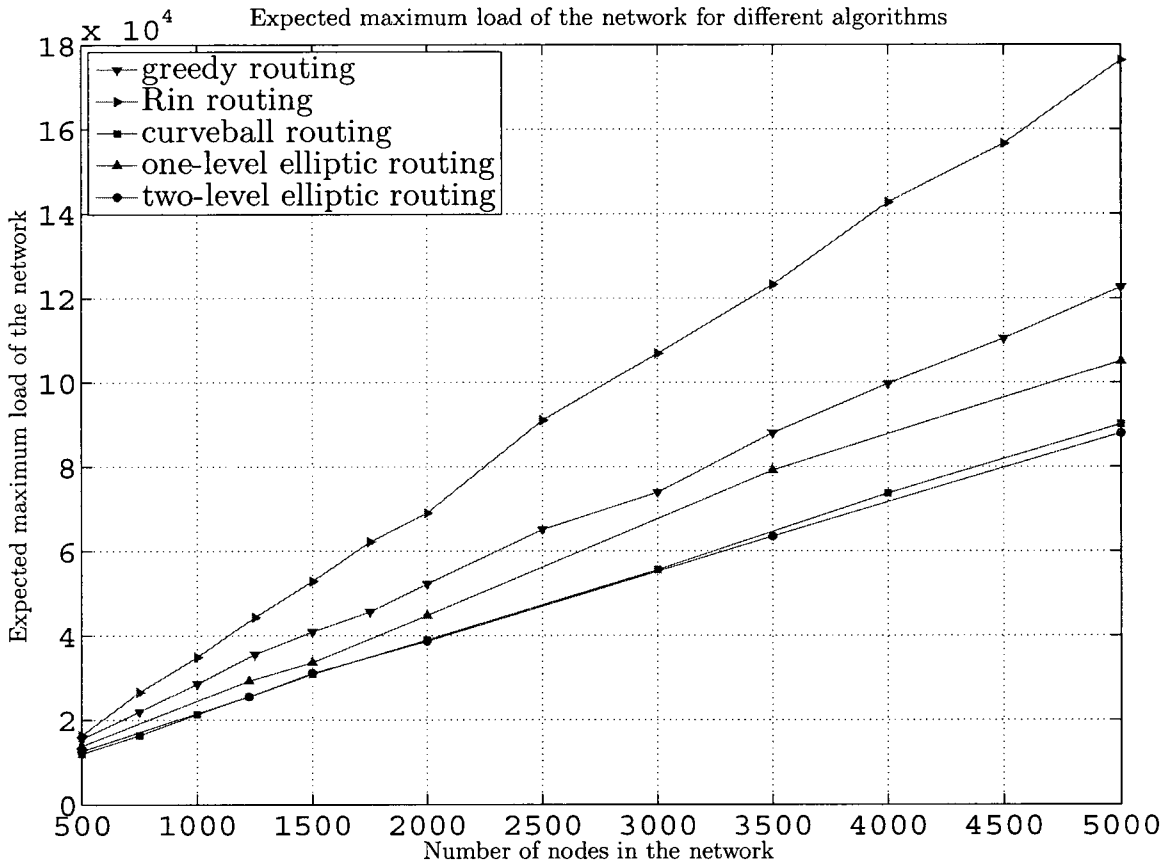


Figure 37: Expected maximum load of nodes in the network induced by different algorithms. Network area is circular with radius 250 and transmission range of nodes is 50.

Table 4 compares different algorithms with respect to *eml*, *mel*, and average load of the nodes in the network to greedy routing.

4.6 Conclusion

In this chapter, we defined two deferent metrics representing the maximum load of the network in two different ways: *expected maximum load (eml)* and *maximum expected load (mel)* of the network, and explained the difference between the two metrics. We also studied the random variable X introduced in the previous chapter for expressing

algorithm	*eml*	*mel*	*average load*
curveball routing (1.1)	$23-28\% \downarrow$	$36\% \downarrow$	$5\% \uparrow$
curveball routing (1.2)	$20-26\% \downarrow$	$43\% \downarrow$	$8\% \uparrow$
Rin routing	$23-46\% \uparrow$	$20\% \downarrow$	$29\% \uparrow$
one-level elliptic $(r, f) = (225, \max)$	$21-24\% \downarrow$	$27\% \downarrow$	$12\% \uparrow$
one-level elliptic $(r, f) = (215, \max)$	$13-18\% \downarrow$	$36\% \downarrow$	$16\% \uparrow$
two-level elliptic $(r, f) = ([215, 235], [\text{min,mean}])$	$24-29\% \downarrow$	$34\% \downarrow$	$9\% \uparrow$
two-level elliptic $(r, f) = ([205, 230], [\text{mean,max}])$	$1-3\% \downarrow$	$45\% \downarrow$	$19\% \uparrow$

Table 4: Comparison of different algorithms in terms of *eml*, *mel*, and *average load* of the network with greedy routing. Network area is circular with radius 250 and transmission range of nodes is 50.

the variance of the load of nodes in more detail and looked at the parameters of the network that affect the variance of X. Then we showed the importance of the distribution of X because of its effect on the *eml* of the network.

We introduced two categories of algorithms aimed to decrease each metric used to represent the maximum load of the network and we explained how one can combine the algorithms from the two categories to obtain a better performance. We compared our algorithms with greedy routing and other existing algorithms with respect to the different metrics. Our simulations show that our algorithms achieve a better performance than the previously proposed algorithms.

Also it is important to mention that our algorithm (elliptic routing), can be implemented on top of the existing location-aware routing algorithms, by using techniques to route to intermediate destinations. This gives the advantage of easier implementation and testing and backward compatibility to the previous algorithms.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis we studied the load distribution induced by routing algorithms and the problem of load balancing in wireless ad hoc networks with single-path routing. We showed how the continuous model which is used in many studies before is not a sufficient model to study load distribution of routing algorithms.

We based our analysis of the load distribution on a discrete model of the network. We calculated the *average progress per hop* of greedy routing as the expected Euclidean distance by which a packet gets closer to the destination node at each hop. Using this parameter and considering the last hop progress, we calculated the average load of the nodes in the network induced by greedy routing. To calculate the maximum load of the nodes induced by greedy routing, first we derived the expected load of any node as a function of its distance from the network center. Then we expressed the actual load of nodes as a random variable where parameters are calculated via simulations. The maximum load of the network (eml) is derived as

the expected maximum of the loads of the nodes in the network. We showed how our result for the expected load of the nodes is consistent with the previous studies which used continuous models for analysis. We also showed that our approach gives a more accurate view of the load distribution of the nodes in the network compared to studies based on continuous model.

We also showed how the new view of the network can affect the design of load balancing algorithms for ad hoc networks. We presented explanations for the observation of low performance of some techniques for load balancing. We distinguished two different metrics for the maximum load of the nodes in the network: *maximum expected load (mel)* and *expected maximum load (eml)* of the nodes. We introduced a technique called *kBestNeighbor* to decrease the *eml* and a group of algorithms called *elliptic routing* to reduce both the *mel* and *eml* of the nodes in the network. We evaluated the performance of our algorithms via simulations and compared our algorithms with other existing algorithms. Our simulations show that *two-level elliptic routing* with appropriately chosen parameters achieves the best *eml* and *mel* values, and a slight improvement over curveball routing.

5.2 Future Work

Although our analysis was based on a circular network area, it can be extended to a square area with small modifications. We expect that one can extend it to other shapes as well. Our algorithms can be combined with the existing location-based routing algorithms to achieve load balancing. The performance analysis of our algorithms combined with other algorithms can be left as a future work. Also our algorithms can be extended using different combinations of functions as parameter.

Bibliography

[1] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (dream). In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 76–84, 1998.

[2] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 48–55, 1999.

[3] J. Broch, D. Johnson, and D. Maltz. The dynamic source routing protocol for mobile ad hoc networks. Internet draft, draft-ietf-manet-dsr-09.txt, 1998.

[4] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97, 1998.

[5] C. Busch, M. Magdon-Ismail, and J. Xi. Oblivious routing on geometric networks. In *Proceedings of the 17th annual ACM symposium on Parallelism in algorithms and architectures*, pages 316–324, 2005.

[6] F. Chung, E. Coffman, M. Reiman, and B. Simon. The forwarding index of communication networks. *IEEE Transactions on Information Theory*, 33(2):224–232, 1987.

[7] B. N. Clark, Ch. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Math.*, 86(1-3):165–177, 1990.

[8] S. De, A. Caruso, T. Chaira, and S. Chessa. Bounds on hop distance in greedy routing approach in wireless ad hoc networks. *International Journal of Wireless and Mobile Computing*, 1(2):131–140, 2006.

[9] S. Durocher, E. Kranakis, D. Krizanc, and L. Narayanan. Balancing Traffic Load Using One-Turn Rectilinear Routing. In *Proceedings of the 5th International Conference on Theory and Applications of Models of Computation*, pages 467–478, 2008.

[10] K. Fall and K. Varadhan. ns notes and documentation. Available from http://www.isi.edu/nsnam/ns/, 2000.

[11] G. G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Research report, available from http://handle.dtic.mil/100.2/ADA180187, 1987.

[12] Y. Ganjali and A. Keshavarzian. Load balancing in ad hoc networks: single-path routing vs. multi-path routing. In *Proceedings of IEEE INFOCOM '04*, volume 2, pages 1120–1125, 2004.

[13] J. Gao and L. Zhang. Load balanced short path routing in wireless networks. In *Proceedings of IEEE INFOCOM '04*, pages 1099–1108, 2004.

[14] J. Gao and L. Zhang. Tradeoffs between stretch factor and load balancing ratio in routing on growth restricted graphs. In *Proceedings of the 23rd annual ACM symposium on Principles of distributed computing*, pages 189–196, 2004.

[15] S. Giordano, I. Stojmenovic, and L. Blazevic. Position based routing algorithms for ad hoc networks: A taxonomy. In *Ad Hoc Wireless Networking*, pages 103–136, 2001.

[16] I. T. Haque, T. Fevens, and L. Narayanan. Randomized routing algorithms in mobile ad hoc networks. In *Proceedings of Sixth IEEE International Conference on Mobile and Wireless Communication Networks*, pages 347–357, 2004.

[17] T. He, Ch. Huang, B. Blum, J. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 81–95, 2003.

[18] T. Hou and V. Li. Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications*, 34(4):38–44, 1986.

[19] E. Hyytiä and J. Virtamo. On load balancing in a dense wireless multihop network. In *Proceedings of the 2nd Conference on Next Generation Internet Design and Engineering*, pages 71–79, 2006.

[20] E. Hyytiä and J. Virtamo. On optimality of single-path routes in massively dense wireless multi-hop networks. In *Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems (MSWiM '07)*, pages 28–35, 2007.

[21] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In *Kluwer Academic Publishers*, pages 153–181, 1996.

[22] D. Johnson, D. Maltz, and J. Broch. *DSR: the dynamic source routing protocol for multihop wireless ad hoc networks*, pages 139–172. Addison-Wesley, 2000.

[23] B. Karp and H. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, 2000.

[24] M. Kendall and P. Moran. *Geometrical Probability.* Griffin, London, 1963.

[25] L. Klein and J. Silvester. Optimum transmission radii for packet radio networks or why six is a magic number. In *Proceedings of IEEE National Telecommunications Conference*, pages 4.3.1–4.3.5, 1978.

[26] Y.-B. Ko and N. H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321, 2000.

[27] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proceedings of the 11th Canadian Conference on Computational Geometry*, pages 51–54, 1999.

[28] S.-J. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Proceedings of ICC'01*, pages 3201–3205, 2001.

[29] W. H. Liao, J. P. Sheu, and Y. Ch. Tseng. Grid: A fully location-aware routing protocol for mobile ad hoc networks. *Telecommunication Systems*, 18(1–3):37–60, 2001.

[30] W. Liu, C. Chiang, H. Wu, and C. Gerla. Routing in clustered multihop mobile wireless networks with fading channel. In *Proceedings of IEEE SICON'97*, pages 197–211, 1997.

[31] M. K. Marina and S. R. Das. On-demand multipath distance vector routing in ad hoc networks. In *Proceedings of ICNP'01*, pages 14–23, 2001.

[32] A. Mei and J. Stefa. Routing in outer space: fair traffic load in multi-hop wireless networks. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 23–32, 2008.

[33] S. Mueller, R. Tsang, and D. Ghosal. Multipath routing in mobile ad hoc networks: Issues and challenges. In *Performance Tools and Applications to Networked Systems, volume 2965 of LNCS*, pages 209–234, 2004.

[34] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and App.*, 1(2):183–197, 1996.

[35] R. Nelson and L. Kleinrock. The spatial capacity of a slotted aloha multihop packet radio network with capture. *IEEE Transactions on Communications*, 32(6):684–694, 1984.

[36] D. Niculescu and B. Nath. Ad hoc positioning system (APS). In *Proceedings of GLOBECOM '01*, pages 2926–2931 vol.5, 2001.

[37] G. Parissidis, V. Lenders, M. May, and B. Plattner. *Multi-path Routing Protocols in Wireless Mobile Ad Hoc Networks: A Quantitative Comparison*, pages 313–326. Springer Berlin / Heidelberg, 2006.

[38] V. Park and M. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM '97: Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, pages 1405–1413, 1997.

[39] Ch. Perkins. *Ad Hoc Networking.* Addison-Wesley, 2001.

[40] Ch. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, 1994.

[41] Ch. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1997.

[42] P. Pham and S. Perreau. Performance analysis of reactive shortest path and multipath routing mechanism with load balance. In *Proceedings of INFOCOM '03*, pages 251–259, 2003.

[43] L. Popa, A. Rostamizadeh, R. Karp, C. Papadimitriou, and I. Stoica. Balancing traffic load in wireless networks with curveball routing. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 170–179, 2007.

[44] S. Ramanathan and M. Steenstrup. A survey of routing techniques for mobile communications networks. *Mobile Networks and Applications*, 1(2):89–104, 1996.

[45] E. Royer and Ch. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 6(2):46–55, 1999.

[46] S. Singh, M. Woo, and C. Raghavendra. Power-aware routing in mobile ad hoc networks. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 181–190, 1998.

[47] I. Stojmenovic. Position-based routing in ad hoc networks. 40(7):128–134, 2002.

[48] I. Stojmenovic and X. Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1023–1032, 2001.

[49] I. Stojmenovic and X. Lin. Power-aware localized routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1122–1133, 2001.

[50] H. Takagi and L. Kleinrock. Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984.

[51] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi. A framework for reliable routing in mobile ad hoc networks. In *Proceedings of INFOCOM'03*, pages 270–280, 2003.