Ant Routing, Searching and Topology Estimation algorithms for Ad Hoc Networks

Ant Routing, Searching and Topology Estimation algorithms for Ad Hoc Networks

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 2 september 2008 om 10.00 uur

door

Santpal Singh DHILLON

Master of Science Duke University, Durham, USA
geboren te Nathana, Punjab, India.

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. P.F.A. Van Mieghem

Samenstelling promotiecommissie:

Rector Magnificus,	Voorzitter
Prof.dr.ir. P.F.A. Van Mieghem,	Technische Universiteit Delft, promotor
Prof.dr.ir. I.G.M.M. Niemegeers,	Technische Universiteit Delft
Prof.dr.ir. S.M. Heemstra de Groot,	Technische Universiteit Delft
Prof.dr.ir. N.H.G. Baken,	Technische Universiteit Delft
Prof.dr. J.L. van den Berg,	University of Twente and TNO Netherlands
Prof.dr.ir. M.R. van Steen,	Vrije Universiteit, Amsterdam

This research was supported by the Dutch Ministry of Economic Affairs under the Innovation Oriented Research Program (IOP GenCom, QoS for Personal Networks @ Home).

ISBN 978-1-58603-901-1

Keywords: Ant routing, random walks, ad hoc networks

Published and distributed by IOS Press under the imprint Delft University Press

Publisher
IOS Press
Nieuwe Hemweg 6b
1013 BG Amsterdam
The Netherlands
tel: +31-20-688 3355
fax: +31-20-687 0019
email: info@iospress.nl
www.iospress.nl
www.dupress.nl

LEGAL NOTICE
The publisher is not responsible for the use which might be made of the following information.

PRINTED IN THE NETHERLANDS

to the silent winds, dark earth and monday morning rain.

Contents

Summary

Title : Ant Routing, Searching and Topology Estimation algorithms for Ad Hoc Networks.

The complexity of networks is increasing to cope with the network model of providing connectivity anywhere and anytime. The idea of universal connectivity has to lead the concept of *ad hoc* networks. The word *ad hoc* comes from Latin meaning "to this". Ad hoc networks are self-configuring, self-organizing networks that are formed on the fly. The dynamic and self-configuring behavior of ad hoc networks provides new challenges. Ad hoc networks have to deal with the inherent difficulties in the wireless medium as well as node mobility. Integration of multiple networks, architectures and technologies introduces further complexity for the paradigm of universal connectivity.

Developing novel algorithms and protocols and analyzing their performance is essential for the development of next generation networks. In this thesis, we aim to analyze the performance of dynamic routing and searching algorithms.

The main aims of this thesis are:

1. Studying the performance of a dynamic, self-adaptive routing paradigm known as ant routing.

2. Analyzing the behavior of searching and how it performs on graph topologies.

3. Understanding the topology of wireless ad hoc networks and its effects on performance of different algorithms in ad hoc networks.

4. Estimation of topology to build topology dependant algorithms.

This thesis is divided into four parts.

The first section is an introduction to the ad hoc networks. This section is divided into three chapters. Chapter 1 discusses different network technologies and architectures. In the second chapter, we describe how the communication networks can be modelled as graph. In this chapter, we also describe OSI layer architecture of Internet and different routing algorithms and protocols. The last chapter in this section presents a survey of routing protocols for ad hoc wireless networks.

The second part of thesis deals with ant routing. Ant routing is a probabilistic routing scheme inspired by real life ant colonies. Ant routing algorithms adapt to changes in network topology and traffic and aim to provide quality of service routing. In chapter 4.2, we study the performance of ant routing algorithms for wired networks. We study the convergence of ant routing algorithm to shortest path for static topology. We also analyze the effect of different parameters and network topology on the performance of ant routing algorithms.

Ant routing algorithms can handle limited dynamic behavior in networks. When the topology of networks changes quickly due to mobility in mobile ad hoc networks, the performance of ant routing algorithms needs to be analyzed. In chapter 6, we study ant routing in ad hoc wireless networks. We also compare the performance of ant routing algorithm with mobile ad hoc wireless routing protocols AODV and DSR.

Searching algorithms are building blocks for many different network algorithms, protocols and services. For example, web search engines and P2P networks need to search for webpages and data respectively. In chapter 8, we study the performance of searching with a single query based on random walk. We also define different searching techniques such as random walk with no repetition of steps, random walk with look-ahead etc. A number of results and conclusions about different searching techniques are presented.

Multiple random walk queries or flooding could be employed for searching. Currently, new versions of P2P networks such as Gnutella are using multiple random walk queries. However, the TTL for random walk queries and the number of queries is set heuristically. In chapter 9, we analyze the optimization of random walk queries based on the number of queries and the TTL for different graph topologies.

The last section of this thesis is divided into two chapters. The topology of ad hoc networks determines important parameters of the network such as the load on different nodes, performance of routing algorithms, overhead of searching algorithms and the lifetime of these networks. In chapter 10, we study the effect of different signal propagation models on the topology for ad hoc wireless networks. Chapter 11 studies the estimation of graph topology based on node degree information.

Part I

Introduction

Chapter 1

Networks and Technologies

Wireless networks and technologies have become ubiquitous in today's world. Cellular systems and wireless local area networks (WLANs) are typical examples of widely used wireless networks. We present a brief overview of different wireless networks and technologies in this chapter.

Communication over wireless channel is the basis for building wireless networks and technologies. Design of wireless networks is a challenging issue due to the nature of wireless channel. The wireless channel is unpredictable and a difficult communication medium. As a signal propagates through a wireless channel, it experiences random fluctuations in time. Thus, the characteristics of a channel appear to change randomly with time, which makes it difficult to design reliable systems with guaranteed performance. Moreover, the radio spectrum is a scarce resource that must be allocated to many different applications and systems. We explain the basics of wireless communication in more detail in section 1.1.

While most of the current wireless networks use infrastructure, it is increasingly common to see ad hoc networks. In infrastructure-based wireless networks each node, a processor with a radio transceiver (transmitter and receiver), communicates directly with a base station or a central station. On the other hand, in ad hoc wireless networks nodes communicate directly with each other without using any infrastructure. Section 1.2 describes various infrastructure based wireless networks and technologies. In section 1.3, we describe mobile ad hoc wireless networks.

1.1 Wireless Communication

The early wireless systems used analog signals. Today most wireless systems use digital signals composed of binary bits, where the bits are obtained directly from a data signal or digitizing an analog signal. Digital systems have higher capacity than analog systems since they can use more spectrally-efficient digital modulation and more efficient

techniques to share the spectrum.

Digital modulation and detection consist of transferring information in the form of bits over a communication channel. Digital modulation consist of mapping the information bits into an analog signal for transmission over the channel. Detection consists of determining the original bit sequence based on the signal received over the channel. There are two main categories of digital modulation: amplitude/phase modulation and frequency modulation. The amplitude and phase modulations embed the information bits into the amplitude and phase of the transmitted signal respectively.

Most of the current wireless systems also use spread spectrum. Spread spectrum is a modulation method applied to digitally modulated signals that increases the transmit signal bandwidth to a value larger than is needed to transmit the underlying information bits. The spread spectrum modulation is done using a spreading code that is independent of the data in the signal. Spread spectrum is typically implemented in one of two forms: direct sequence (DS) or frequency hopping (FH). In direct sequence spread spectrum (DSSS) modulation, the modulated data signal is multiplied by a wideband spreading signal.

In multiuser systems the system resources must be divided among multiple users. The signals of bandwidth B and time duration T occupy a signal space of dimension $2BT$. In order to support multiple users, the signal space dimensions of a multiuser system must be allocated to the different users. When dedicated channels are allocated to users, the system allocation is termed as multiple access. Applications with continuous transmission and delay constraints, such as voice or video, typically require dedicated channels for good performance to insure their transmission is not interrupted. Dedicated channels are obtained from the system signal space using a channelization method such as time-division, frequency-division, code-division, or a hybrid combination of these techniques. In frequency division multiple access (FDMA), the total system bandwidth is divided into orthogonal frequency channels. In time division multiple access (TDMA), time is divided orthogonally and each channel occupies the entire frequency band over its assigned timeslot. Because signaling dimensions can be allocate to different users in an infinite number of ways, multiuser channel capacity is defined by rate region rather than a single number. Allocation of signaling dimensions for users with bursty transmissions generally use a form of random channel allocation which does not guarantee channel access. Bandwidth sharing using random channel allocation is called random multiple access or random access.

In general, the choice of whether to use multiple access or random access, and which specific multiple technique to apply depends on the system applications, the traffic characteristics of the users in the system, the performance requirements, and the characteristics of the channel and other interfering systems operating in the same bandwidth.

Most wireless applications reside in the radio spectrum between 30 MHz and 30 GHz. The radio spectrum is controlled by regulatory bodies both regionally and globally. A

Table 1.1: Spectrum allocation for various wireless systems.

AM radio	536-1605 KHz
FM radio	88-108 MHz
Broadcast TV (UHF)	470-806 MHz
3G Broadband Wireless	746-764 MHz, 776-794 MHz
3G Broadband Wireless	1.7-1.85 MHz, 2.5-2.69 MHz
1G and 2G Digital Cellular Phones	806-902 MHz
Personal Communication Services (2G Cell Phones)	1.85-1.99 GHz
Wireless Communications Service	2.305-2.32 GHz, 2.345-2.36 GHz
Satellite Digital Radio	2.32-2.325 GHz
Digital Broadcast Satellite (Satellite TV)	12.2-12.7 GHz
Fixed Wireless Services	38.6-40 GHz

Table 1.2: Unlicensed Spectrum

ISM Band I (Cordless phones, IG WLANs)	902-928 MHz
ISM Band II (Bluetooth, IG WLANs)	2.4-2.4835 GHz
ISM Band III (Wireless PBX)	5.725-5.85 GHz
NII Band I (Indoor systems, 802.11a WLANs)	5.15-5.25 GHz
NII Band II (short outdoor and campus applications)	5.25-5.35 GHz
NII Band III (long outdoor and point-to-point links)	5.725-5.825 GHz

regional or global system operating in a given frequency band must obey the restrictions for that band set forth by the regulatory body. The spectrum is allocated in licensed bands, which regulatory bodies assign to specific operators, or in unlicensed bands, which can be used by any system subject to certain operational requirements. Table 1.1, taken from Goldsmith [53], shows the licensed spectrum allocated to major commercial wireless systems in the U.S.A. today.

Unlicensed spectrum is allocated by the governing body within a given country. In general, countries try to match their frequency allocation for unlicensed spectrum so that the technology developed for this spectrum is compatible world-wide. Table 1.2 shows the unlicensed spectrum allocations in the U.S.A. [53, 87].

1.2 Wireless Networks and Technologies

The design and development of mobile wireless networks poses significant challenges compared to traditional wired networks. In contrast to the stable link capacity of wired networks, wireless link capacity continually varies because of the impacts from transmission power, receiver sensitivity, noise, fading and interference. Additionally, wireless mobile networks have a high error rate, power restrictions and bandwidth limitations. In mobile networks, node mobility may cause frequent network topology changes which are rare in wired networks.

In this section, we describe wireless networks based on infrastructure. Wireless networks based on infrastructure are single-hop networks with direct communication between a node and base station. Most control issues in these networks such as mobility and scheduling are handled by the central base station or access point. The next subsections present an overview of cellular systems and WLANs.

1.2.1 Cellular Systems

The basic idea behind cellular systems is frequency reuse, which exploits the fact that the signal power falls off with distance so that the same frequency spectrum can be used at spatially separated locations. The coverage area of a cellular system is divided into nonoverlapping cells where some set of channels is assigned to each cell. Operation within a cell is controlled by a base station.

All base stations in a geographical area are connected via a high speed communication link to a mobile telephone switching office (MTSO). The MTSO acts as a central controller for the network, allocating channels within each cell, coordinating handoffs between cells when a node traverses a cell boundary, and routing calls to and from mobile users. The MTSO can route voice messages through the public switched telephone network or provide Internet access (Figure 1.1).

The first generation cellular systems were analog while the second and third generation cellular systems are digital. A prominent example of second generation digital system is Groupe Spéciale Mobile (GSM). The GSM system, used primarily in Europe, uses a combination of TDMA and slow frequency hopping with frequency-shift keying for the voice modulation. The third generation (3G) cellular systems are based on wideband code division multiple access (WCDMA) standard developed within the auspices of the International Telecommunication Union (ITU).

1.2.2 Wireless Local Area Networks

Wireless local area networks (WLANs) provide high speed data within a small region as users move from place to place. Wireless devices that access these LANs are typically stationary or moving at pedestrian speeds. All wireless LAN standards in USA operate

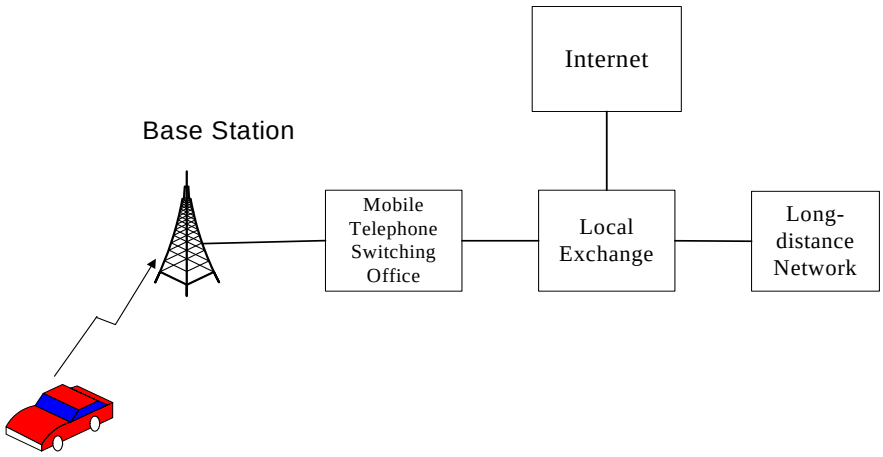


Figure 1.1: Architecture of a Cellular Network

in the unlicensed frequency bands. The primary unlicensed bands are the ISM bands at 900 megahertz (MHz), 2.4 gigahertz (GHz) and 5.8 GHz, and the unlicensed national information infrastructure (U-NII) band at 5 GHz. The wireless LAN standard IEEE 802.11b operates with 80 MHz of spectrum in the 2.4 GHz ISM band[1]. The standard specifies DSSS with data rates of around 1.6 megabit per second (Mbps) and a range of approximately 150 metres (m).

The IEEE 802.11a wireless LAN standard operates with 300 MHz spectrum in the 5 GHz U-NII band. The 802.11a standard is based on multicarrier modulation and provides 20-70 Mbps data rates. Another standard 802.11g uses multicarrier modulation and can be used in either the 2.4 GHz and 5 GHz bands with speeds of up to 54 Mbps. In Europe wireless LAN standard HIPERLAN (high performance radio LAN) standard has been developed. The HIPERLAN Type 1 has data rate of 20 Mbps at a range of 50 m.

1.2.3 Broadband Wireless Access Technology

Broadband wireless access provides high-rate wireless communication between a fixed access point and multiple terminals. Worldwide Interoperability for Microwave Access (WiMAX) is a broadband wireless technology based on IEEE 802.16 standard[2]. The 802.16 specification is a standard for broadband wireless access systems operating at radio frequencies between 10 GHz and 66 GHz. WiMAX standard supports data rates of 40 Mbps for fixed users and 15 Mbps for mobile users with a range of several kilometers.

[1] *http://www.ieee802.org/11/*
[2] *http://www.wimaxforum.org*

WiMAX competes with wireless technologies such as WLANs, 3G cellular services, and wired technologies such as cable.

1.3 Mobile Ad hoc Wireless Networks

Mobile ad hoc networks have attracted significant amount of interest in recent years because of their flexibility, robustness and reduced costs. Mobile ad hoc networks are multihop wireless networks, where each node not only generates its own data but also forwards the data of other nodes. Some of the ad hoc wireless networks such as personal networks, sensor networks and mesh networks are described in next subsections.

The lack of infrastructure adds additional complexity in mobile ad hoc networks. In these networks, all processing and control must be done by the network nodes in a distributed fashion. An important aspect of lack of infrastructure in ad hoc networks is that network topology changes have also to be handled by nodes.

1.3.1 Low Cost and Low Power Radio technologies

Bluetooth[3] and Zigbee[4] are examples of radio technologies, which due to their low cost and power consumption can be embedded in a variety of devices to create ad hoc networks (smart homes, sensor networks) etc. Bluetooth's range of operation is 10 m (at 1mW transmit power) and this range can be increased to 100 m by increasing the transmission power to 100 mW. The system operates in the unlicensed 2.4 GHz frequency band and provides 1 asynchronous data channel at 723.2 Kbps. Bluetooth uses frequency-hopping for multiple access with a carrier spacing of 1 MHz. Typically, up to 80 different frequencies are used for total bandwidth of 80 MHz.

The Zigbee radio specification is designed for lower cost and power consumption than Bluetooth. The specification is based on the IEEE 802.15.4 standard and radio is capable of connecting 255 devices per network. The specification supports data rates of 255 Kbps at a range of about 30 m. Zigbee is designed to provide radio operation for months or years without recharging.

1.3.2 Personal Networks

Personal Network (PN) is a concept proposed by Niemegeers and Heemstra de Groot [78] related to the field of pervasive computing that extends the concept of a Personal Area Network (PAN). The latter refers to a space of small coverage (less than 10 m) around a person where ad hoc communication occurs, typically between portable and mobile computing devices such as laptops, personal digital assistants (PDAs), cell phones,

[3] *http://www.bluetooth.com*
[4] *http://www.zigbee.org*

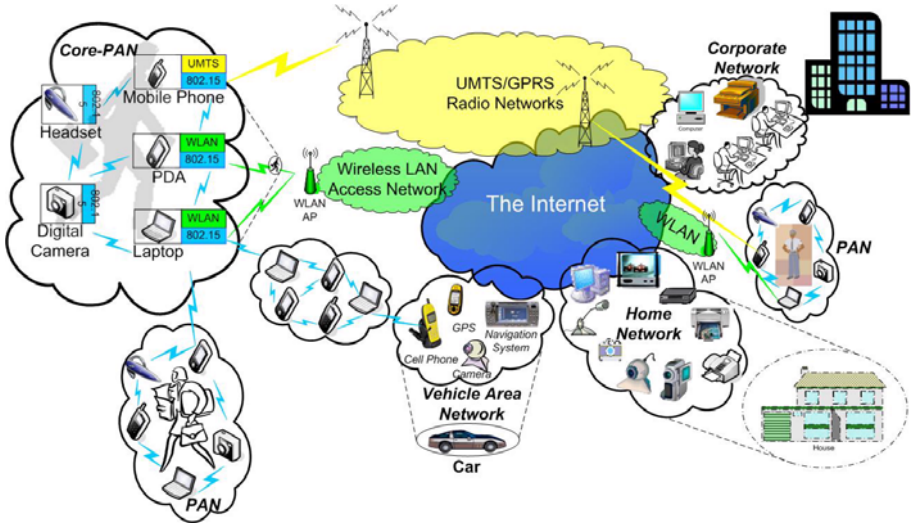


Figure 1.2: Personal Network

headsets and digital gadgets. A PN has a core consisting of a PAN, which is extended on-demand and in an ad hoc fashion with personal resources or resources belonging to others. This extension is made physically via infrastructure-based networks, e.g., the Internet, an organization's intranet, or a PN belonging to another person, a vehicle area network, or a home network. The resources, which can become part of a PN, are very diverse. These resources can be private or may have to be shared with other people. Figure 1.2 shows an example of a PN.

1.3.3 Sensor Networks

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, pressure or motion [60]. In addition to one or more sensors, each node in a sensor network is typically equipped with a radio transceiver, a small microcontroller, and an energy source usually a battery. The individual devices in WSN are inherently resource constrained. They have limited processing speed, storage capacity, and communication bandwidth. These devices have substantial processing capability in the aggregate, but not individually.

Area monitoring is a typical application of WSNs [32]. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. As an example, a large quantity of sensor nodes could be deployed over a battlefield to detect enemy intrusion instead of using landmines. When the sensors detect the event being monitored, the event is reported to one of the base stations, which takes appropriate

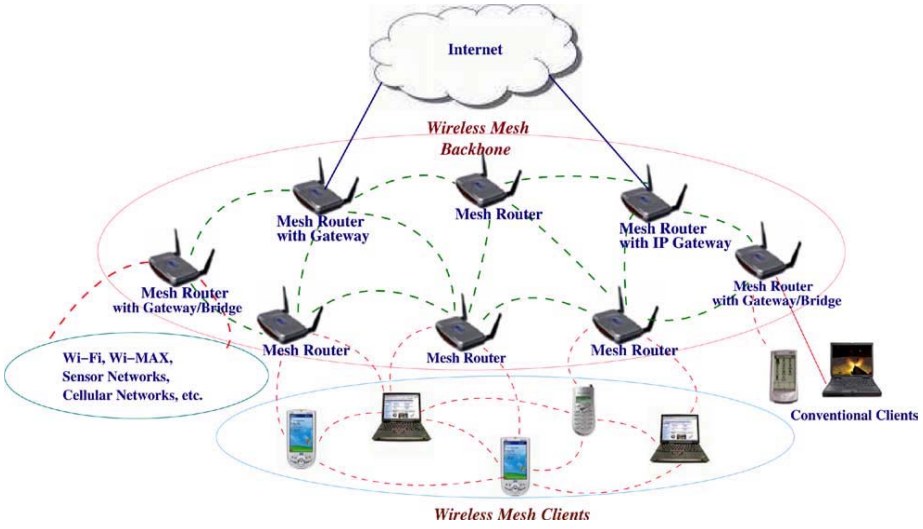


Figure 1.3: Hybrid Mesh Network (This figure is taken from Akyildiz [3])

action (e.g., send a message on the Internet). Communication in sensor networks could be both single hop and multiple hop. Depending on the need of applications such as real-time response, redundancy of the data, security etc., different data propagation strategies could be employed.

1.3.4 Mesh Networks

Wireless mesh networks (WMNs) consist of mesh routers and mesh clients, where mesh routers have minimal mobility and form the backbone of WMNs. They provide network access for both mesh and conventional clients. Both the mesh clients and routers have the capability to forward data packets from other nodes, thus acting as hosts and routers. Mesh clients can access the network through mesh routers as well as directly meshing with other mesh clients. While the infrastructure provides connectivity to other networks such as the Internet, Wi-Fi, WiMAX, cellular, and sensor networks; the routing capabilities of clients provides improved connectivity and coverage inside the WMN.

WMN is a promising wireless technology for numerous applications [3], e.g., broadband home networking, community and neighborhood networks, enterprise networking, building automation, etc.

The architecture of WMNs can be classified into three main groups based on the functionality of the nodes: infrastructure/backbone WMNs, client WMNs and hybrid WMNs. Figure 1.3 gives an example of a hybrid WMN [3].

1.4 Peer-to-peer Networks

Peer-to-peer (P2P) networks can be considered an example of ad hoc networks because of the dynamic topology. P2P networks are overlay networks that can be formed over wired or wireless network. A pure P2P network does not have the notion of clients or servers, but only equal peer nodes that simultaneously function as both clients and servers to the other nodes on the network. This model of network arrangement differs from the client-server model where communication is usually to and from a central server. A typical example for a non peer-to-peer file transfer is a file transfer protocol (FTP) server where the client and server programs are quite distinct, and the clients initiate the download/uploads and the servers react to and satisfy these requests.

Peer-to-peer networks are typically used for connecting nodes via largely ad hoc connections. Such networks are useful for many purposes. Sharing content files containing audio, video, data or anything in digital format is very common, and real-time data, such as telephony traffic, is also passed using P2P technology.

The P2P overlay network consists of all the participating peers as network nodes. Based on how the nodes in the overlay network are linked to each other, we can classify the P2P networks as unstructured or structured [30].

An unstructured P2P network is formed when the overlay links are established arbitrarily. Such networks can be easily constructed as a new peer that wants to join the network can copy existing links of another node and then form its own links over time. In an unstructured P2P network, if a peer wants to find a desired piece of data in the network, the query has to be flooded through the network to find as many peers as possible that share the data. The main disadvantage with such networks is that the queries may not always be resolved. Most of the popular P2P networks such as Gnutella[5] are unstructured.

Structured P2P networks employ a globally consistent protocol to ensure that any node can efficiently route a search to some peer that has the desired file, even if the file is extremely rare [50]. Such a guarantee necessitates a more structured pattern of overlay links. By far the most common type of structured P2P network is the distributed hash table (DHT), in which a variant of consistent hashing is used to assign ownership of each file to a particular peer, in a way analogous to a traditional hash table's assignment of each key to a particular array slot. Some well known DHTs are Chord [97], Pastry [90] and CAN [88].

[5] *http://www.gnutella.com*

Chapter 2

Network Modelling

A communication network can be modeled as a graph. A network of N nodes and L links can be represented as a graph with N vertices and L edges. Section 2.1 gives basic graph definitions. Different graph models such as ER random graphs and power law graphs are used for analysis of networks. Section 2.2 explains different graph models and their unique features. The network architecture is described in section 2.3. This section also explains the routing algorithms and protocols. The last section of this chapter also gives an introduction to routing in wireless networks.

2.1 Graph Definitions

A graph is a data structure consisting of N vertices (nodes) joined together by L edges (links). A link $(u, v) \in L$ is said to be incident to nodes u and v, and vice versa. If $(u, v) \in L$, then nodes u and v are said to be adjacent. The adjacency matrix $A[G] = a_{uv}$ corresponding to an undirected graph G is defined as:

$$\begin{aligned} a_{uv} &= 1, \text{ if } (u, v) \in L \\ &= 0, \text{ otherwise.} \end{aligned} \tag{2.1}$$

The adjacencies defining the graph can also be represented by an adjacency-list. The adjacency-list contains for each node $u \in V$ a list $Adj[u]$ with pointers to all nodes that are adjacent to u. Each link $(u, v) \in L$ can also be assigned a weight $w(u, v)$ and the resulting graph is known as weighted graph. In network terminology, the weight of a link is also termed as the cost of the link.

Definition 1 *Walk: A walk from node u to node v is an alternating finite sequence $v_0, l_1, v_1, ..., l_k, v_k$ of nodes v_i and links l_i, where l_i is a link connecting v_{i-1} and $v_i, v_0 = u$ and $v_k = v$.*

Definition 2 *Path: A path is a walk in which all nodes v_0 to v_k are distinct ($v_i \neq v_j$ for every index $i \neq j$).*

Definition 3 *Connected: A graph is connected if there exists a path between each pair of nodes in the graph.*

Definition 4 *Cycle: A cycle is a walk for which all nodes except the first and last are distinct. If the graph contains no cycles it is called acyclic.*

Definition 5 *Degree of Connected Graph: In a connected graph with no parallel links and self loops, a degree $\deg_u$ of a node u represents the number of nodes adjacent to node u,$1 \leq \deg_u \leq N-1$.*

The degrees of the nodes in the graph must satisfy the following equality [105]:

$$\sum_{i=1}^{N} \deg_i = 2L \tag{2.2}$$

2.2 Graph Models

One of the simplest graph models used in network modelling is a full mesh or complete graph. A complete graph K_N consists of N nodes and each node is connected to every other node in the graph. Thus, the number of links in a complete graph K_N are given by $L = L_{\max} = \frac{N(N-1)}{2}$ links.

An example of an extremely regular topology is the class of lattice topologies. We only consider a subclass of lattices, namely rectangular two-dimensional lattices with size l_1 and l_2 and $N = (l_1+1)(l_2+1)$. The shortest hop path between two diagonal corner points in the rectangular two-dimensional lattice has $l_1 + l_2$ hops. Figure 2.1 gives an example of square lattice with $N = 49$ nodes.

The random graphs, first defined by Erdös and Rényi in 1959, is another commonly used graph topology for network modelling [48]. Random graphs are also referred to as Erdös-Rényi (ER) random graphs to distinguish them from power law random graphs. The two most frequently occurring models for random graphs are $G_p(N)$ and $G(N, L)$. The class of graphs denoted by $G_p(N)$ consists of graphs with N nodes in which each possible edge exists with probability p. Thus, the average number of links are given by $pL_{\max}$.

The degree distribution of node u in $G_p(N)$ is Binomial [14]

$$\Pr[\deg_u = j] = \binom{N-1}{j} p^j (1-p)^j \tag{2.3}$$

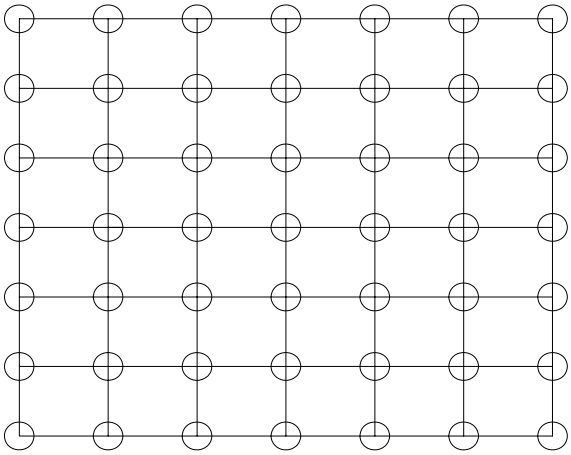


Figure 2.1: An example of square lattice with $N = 49$ nodes.

and the average degree is $p\,(N-1)$. Erdös and Rényi [48] identified a phase transition in random graphs. The probability that almost every graph $G_p\,(N)$ is connected is restricted from below by the critical threshold $p_c \sim \frac{\ln N}{N}$ for N large [14, 48]. Thus, if $p > p_c$ then almost all graphs $G_p\,(N)$ are connected, else almost all graphs are disconnected.

The class of Waxman graphs belongs to the class of random graphs, where the probability of existence of a link between two nodes decays exponentially with the geographic distance between those two nodes [65, 106]. More formally, the Waxman graphs belong to the class $G_{p_{uv}}\,(N)$ with $p_{uv} = f\,(\overrightarrow{r_u} - \overrightarrow{r_v})$, where the vector $\overrightarrow{r_u}$ represents the position of node u and all nodes are uniformly distributed in a hyper cube in the $\omega-$dimension space. The distance function $f\,(\overrightarrow{r}) = e^{-\alpha|\overrightarrow{r}|}$, where $\overrightarrow{r}$ is a norm denoting the distance from the origin.

Random geometric graphs (RGG) have gained new relevance with the advent of ad-hoc and sensor networks as they are used to model these networks [11, 57, 58]. A random geometric graph is a graph $G(N, r)$ resulting from placing N points uniformly at random on the unit square 1 and connecting two points iff their Euclidean distance is at most r. Consider that wireless nodes forming a network are uniformly distributed with certain density δ. The number of nodes within a circular area of radius r follows a Poisson distribution with mean number of neighbors $\delta\pi r^2$ [11, 58]. However, the degree distribution of ad hoc wireless networks also depends on shadowing, fading and interference [41, 57]. The modelling of ad hoc wireless networks is discussed in detail in chapter 10 of this thesis.

The degree distribution in the Internet and peer-to-peer networks follows a power law [49, 52]. Albert and Barabási [4] demonstrated via empirical results that the degree distribution for many other networks such as World Wide Web (WWW), metabolic

networks, phone call graphs, movie actor collaboration networks also follow power laws. These networks can be modeled as power law graphs. In power law graphs, the degree distribution of node u is

$$\Pr\left[\deg_u = j\right] = cj^{-\alpha} \tag{2.4}$$

where α is the power law exponent and c is the normalization constant such that $\sum_{j=1}^{N-1} \Pr\left[\deg_u = j\right] = 1$. Measurements in the Internet [49] suggest that $\alpha \approx 2.4$.

Power law graphs can be generated using a variety of methods. The Barabási-Albert (BA) model for generating power law graphs is defined in two steps [6]. Starting with a small number (v_0) of nodes, at every timestep, a new node is added with l ($\leq v_0$) links. A new node connects to nodes already in the graph with probability $y = \frac{d_u}{\sum_{v \in Z} d_v}$, where d_u is the degree of node u and Z is the number of nodes in the graph at a particular timestep.

After t timesteps the model leads to a random network with $N = t + v_0$ nodes and lt links. It has been shown in [6] that $\Pr[\deg \leq j] = 1 - \frac{l^2 t}{j^2 N}$. Thus, the probability that a node s has degree j in this model follows a power law [6],

$$\Pr[\deg_s = j] = \frac{2l^2 t}{N} \frac{1}{j^3} \; = \mathfrak{c} j^{-\upsilon} \tag{2.5}$$

where the scaling exponent $\upsilon = 3$ is independent of l.

The number of nodes with degree less than $\log N$ in the BA model is $N \cdot \Pr[\deg \leq \log N] = N\left(1 - \frac{4}{(\log N)^2}\right)$ and the number of nodes with a large degree is small. On the other hand, in *almost surely* (a.s.) connected ER random graph where $p \geq \frac{\log N}{N}$, the average node degree is close to or greater than $\log N$. Figure 2.2(a) shows the degree distribution for ER random graphs. Figure 2.2(b) shows the degree distribution of power law random graph generated using BA model on a log-log scale.

Many large-scale systems in communications, biology and sociology such as WWW, the Internet, metabolic networks, phone call graphs, movie actor collaboration networks are classified as complex networks [4, 100]. To understand complex networks, it is essential to know clustering coefficient [77]. The clustering coefficient of a vertex in a graph quantifies how close the vertex and its neighbors are to being a clique (complete graph). For instance, sparse random graphs have a vanishingly small clustering coefficient while real world networks often have a coefficient significantly larger. Complex networks are characterized by power law degree distribution, a high clustering coefficient, assortativity[1] or disassortativity among vertices, and evidence of a hierarchical structure. This

[1] Assortativity refers to a preference for a network's nodes to attach to others that are similar or different in some way

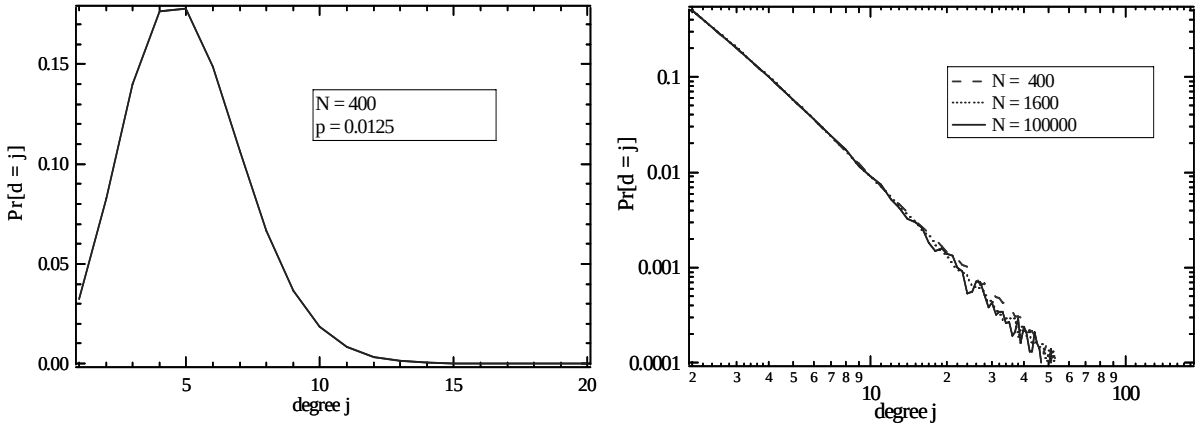


Figure 2.2: Degree distribution for ER random graph and power law random graph.

is in contrast to lattice or random graphs which exhibit a high similarity.

2.3 Routing Algorithms and Protocols

We study the architecture of the Internet to understand the concepts of routing algorithms and protocols and medium access layer protocols. The Internet is a collection of interconnected networks that use packet switching [83]. The design of complex networks such as the Internet lead to the concept of layering. Figure 2.3 shows the Open Systems Interconnection (OSI) layered architecture for the Internet. Though the OSI architecture defines 7 layers, our focus in this thesis is on physical, data link, network and application layers. Figure 2.3 also shows some of the common technologies used at each layer.

The physical layer represents the physical medium through which the data is transmitted. In case of wired networks, there are technologies such as coaxial cable, twisted pair etc. that are use to make the physical connection. The data link layer (layer 2) is the layer which transfers data between adjacent network nodes in a wide area network or between nodes on the same local area network (LAN) segment. At this layer, the frames are forwarded by nodes based on spanning tree algorithm. We do not into details of the algorithm, however, the nodes using this algorithm do not need to know the topology of the whole network. The layer 2 nodes forwarding packets are referred to as bridges. In some networks, such as IEEE 802 LANs, the data link layer is split into Medium Access Control (MAC) and Logical Link Control (LLC) sublayers. The MAC layer specifies number of protocols (e.g. CSMA/CA, CSMA/CD) which deal with

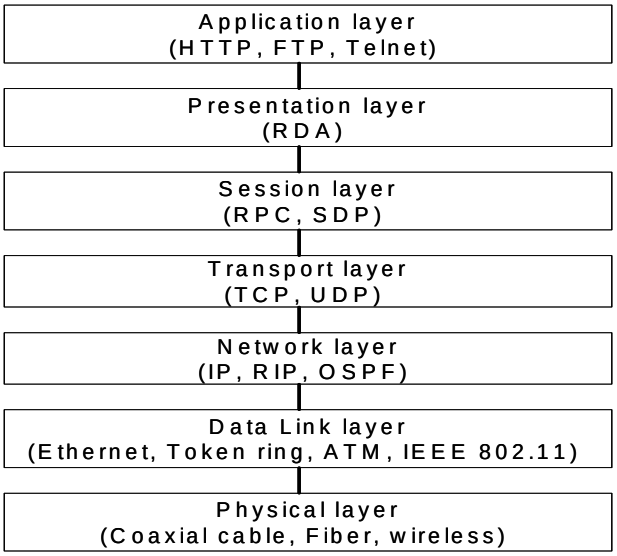


Figure 2.3: OSI model

channel access control mechanisms.

The layer 3 involves interconnection of networks. At layer 3, the nodes forwarding data packets termed routers, use routing protocols to learn the topology of whole network. The routing table stores the routes and in some cases, metrics associated with those routes, to particular network destinations. Once the routers have topology information, a routing algorithm is used to compute the path to the destination [107].

There are two types of routing protocols defined at layer 3 - Link State and Distance Vector [83]. In distance vector routing protocols, each node constructs a one-dimensional array or vector containing the costs to all other nodes and distributes that vector to its immediate neighbors. The routing information protocol (RIP) is an example of distance vector routing protocol. The Bellman-Ford shortest path algorithm is used in RIP to compute the shortest paths.

In link state routing, each node knows the link weight of its directly connected neighbors. This information is flooded by the nodes through the entire network. Open Shortest Path First (OSPF) is the most widely used link state routing protocol in the Internet today. In OSPF, once a node knows the full network topology and link weight information, it uses Dijkstra's shortest path algorithm to compute paths to different destinations. In the next section, we describe the Dijkstra's algorithm in detail.

```
INITIALIZE_SINGLE_SOURCE (G, s, F, π)
  for each vertex v ∈ N
      do F [v] ← ∞
         π [v] ← NIL
  F [s] ← 0
```

Figure 2.4: Initialization

```
RELAX(u, v, w, F, π)
  if F [v] > F [u] + w (u, v)
     then F [v] ← F [u] + w (u, v)
          π [v] ← u
```

Figure 2.5: Relaxation

2.3.1 Dijkstra's algorithm

We first describe an important property of one-dimensional shortest paths and describe the technique of relaxation used by Dijkstra's shortest path algorithm

Property 6 *Subpaths of shortest paths in one dimension are also shortest paths.*

Property 1 is used in the technique of relaxation to obtain shortest path length in a monotonically decreasing fashion. Each node $u \in V$ maintains an estimate $F[u]$ of the shortest path distance from the source node s to node u. Based on the above property we know that subpaths of shortest paths must also be shortest. Therefore, if $F[v] > F[u] + w(u,v)$ we can improve the "shortest" path to v found so far by going via the node u to node v, using link (u,v). This process of checking whether we can improve the distance estimate of a path to a node v by going via a different path to a neighboring node u and taking the link (u,v), is called relaxing the link (u,v). Initially, all estimates $F[u] \forall u \in V$ are set to infinity. In Figures 2.4 and 2.5 the meta code for the initialization and the relaxation are given. Lines 1-3 of the INITIALIZE routine in Figure 2.4 set for all the nodes the estimates to infinity and the predecessors to NIL. Only the estimates $F[s]$ of the source node is set to 0 in line 4, since the search started from source itself. Line 1 of the procedure RELAX checks whether the distance $F[v]$ can be improved by going via the node u and link (u,v) to node v. If this is the case then the estimate and predecessor of node v are updated in lines 2 and 3.

Dijkstra's algorithm is used to compute the shortest path from a source to node on a weighted directed graph $G(N,L)$ for the case where all link weights are nonnegative (Figure 2.6). Dijkstra's algorithm maintains a set S of vertices whose final shortest-path weights from the source s have already been determined. The algorithm repeatedly selects the vertex $u \in N - S$ with the minimum shortest-path estimate, adds u to S,

```
DIJKSTRA(G, w, s)
  INITIALIZE_SINGLE_SOURCE (G, s, F, π)
  S ← ∅
  Q ← N[G]
  while Q ≠ ∅
     do u ←EXTRACT_MIN(Q)
        S ← S ∪ {u}
        for each vertex v ∈ Adj[u]
           do RELAX(u, v, w, F, π)
```

Figure 2.6: The Dijkstra algorithm

and relaxes all links leaving u. Line 1 performs an initialization of all nodes, and line 2 initializes the set S to the empty set. Line 3 initializes the min-priority queue $\mathcal{Q}$ to contain all vertices in V. Each time through the while loop of lines 4-8, a vertex u is extracted from $\mathcal{Q} = V - S$ and added to set S. Vertex u, therefore, has the smallest shortest-path estimate of any vertex in $V - S$. Then, lines 7-8 relax each link (u, v) leaving u, thus updating the estimate $F[v]$ and the predecessor $\pi[v]$ if the shortest path to v can be improved by going through u.

The proof of correctness of Dijkstra's algorithm is given in Cormen *et al.* [29]. The worst-case complexity of the Dijkstra's algorithm when the min-priority queue $\mathcal{Q}$ is implemented as Fibonacci heap is $O(N \log N + L)$ [29].

2.3.2 QoS Routing Protocols and Algorithms

Quality of service, abbreviated as QoS, refers to resource reservation control mechanisms. QoS can provide different priority to different users or data flows, or guarantee a certain level of performance to a data flow in accordance with requests from the application program. In soft-QoS, some traffic is given preference over other, however, no guarantee is provided. On the other hand, hard QoS or guaranteed services involves absolute reservation of networks resources for specific traffic.

To understand QoS algorithms, we use the formulation of Kuipers [65]. Each link $(u, v) \in L$ is characterized by an $\omega-$dimensional link weight vector $\overrightarrow{w}(u, v) = [w_1(u, v), w_2(u, v), ..., w_\omega(u, v)]$, where $w_i(u, v) > 0 \forall (u, v) \in L$ and the ω components referred to as QoS measures such as delay, jitter, available bandwidth etc. Given ω constraints U_i where $1 \leq i \leq \omega$, the multi-constrained problem is to find a path P from a source node s to a destination node d such that

$$\sum_{(u,v)\in P} w_i(u, v) \leq U_i \forall 1 \leq i \leq \omega \tag{2.6}$$

Note that the above formulation is valid only for additive link weights. Self-Adaptive

Multiple Constraints routing algorithm (SAMCRA) is an example of exact QoS algorithm that finds the minimum cost path satisfying the required constraints [108].

2.3.3 Routing in Wireless Networks

As explained in the last section, routing in communication networks can be classified into routing protocol and routing algorithm. The function of a routing protocol is to spread the routing information while the routing algorithm computes the paths. The boundary between routing protocols and algorithms is blurred in ad-hoc networks where the nodes may not have enough topology information and paths are set up on demand when the nodes have data to send [39]. Moreover, single hop networks such as infrastructure based wireless networks do not need routing protocols and algorithms.

Layer 2 in wireless networks defines different MAC protocols to access the medium. MAC protocols are needed to regulate communication between nodes through a shared medium. A variety of MAC protocols have been developed for communication in wired and wireless networks. For example, IEEE 802.3 (based on carrier sense multiple access with collision detection (CSMA/CD) for wired Ethernet) and IEEE 802.11 for WLAN. The IEEE 802.11 standard uses CSMA with collision avoidance.

Since each node in ad hoc wireless networks forwards packets of other nodes, it needs to have functionality till layer 3 in the OSI model. Indeed, the first generation of routing protocols such as dynamic source routing (DSR) can be considered purely to belong to the network layer. Such routing protocols are designed independent of the lower level layers i.e., physical layer and MAC layer. Since MAC protocols affect interference levels and capacity in wireless networks, increasingly it has been observed that a cross layer strategy may provide a better solution to routing in ad hoc networks. A detailed survey of various routing protocols and cross-layer design for ad hoc networks is presented in the next chapter.

Chapter 3

Survey of Ad hoc Routing Protocols

Most of the routing protocols for ad-hoc networks can be classified into three major categories (Figure 3.1) based on the routing information stored at the nodes, namely, proactive or table driven, reactive or on-demand and hybrid. Other classification schemes categorize the protocols based on whether they are hierarchical, support multicast, or are power aware [26].

The first section in this chapter shows how routing protocols are categorized. In the remaining sections, we explain 3 ad hoc wireless routing protocols namely DSDV, AODV and DSR in detail. We compare the performance of these routing protocols with ant routing in chapter 6.

3.1 Classification of Ad hoc Routing Protocols

Table-driven routing protocols (e.g. DSDV [82], CGSR [21], OLSR [25]) attempt to maintain consistent, up-to-date routing information from each node to every other node in the network. These protocols require each node to maintain one or more tables to store routing information, and they respond to changes in network topology by propagating updates throughout the network in order to maintain a consistent network view. The different table-driven protocols differ with respect to each other in terms of number of routing-related tables and the methods by which changes in the network structure are broadcast.

In reactive or on-demand protocols (e.g. AODV [81], DSR [62], ABR [102]), routes are created only when desired by the source node. When any node requires a route to the destination, it initiates a route discovery process within the network. This process is completed once a route is formed or all possible route permutations have been examined. Once a route has been established, it is maintained by a route maintenance procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired. The routes in reactive protocols could be established

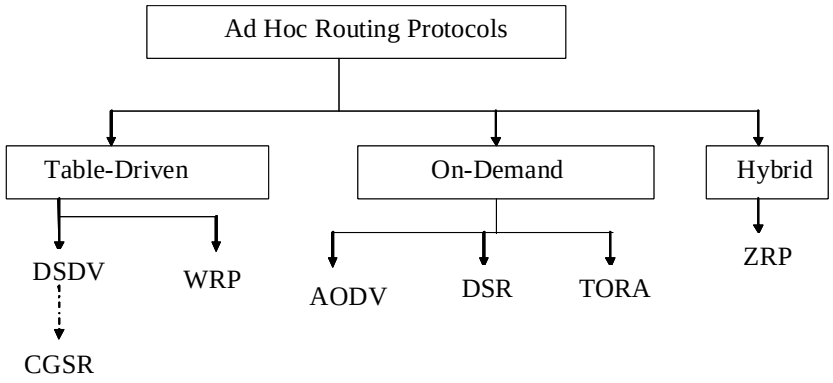


Figure 3.1: Classification of Routing protocols for ad hoc networks

based on different parameters such as signal stability, battery power available to nodes etc.

The advantage of the proactive schemes is that once a route is requested, there is little delay until a route is determined. The disadvantage is that table-driven protocols need to maintain up-to-date routing information at the nodes. Since in ad hoc network nodes might move very fast, the changes in routing information may be more frequent than the routing requests, leading to waste of the network capacity as much of the routing information is never used.

Reactive protocols have significant advantages over table-driven protocols in terms of reducing the overhead of routing protocols and ability of the routing protocols to react quickly to topology changes in the network. They are better scalable than table driven protocols in terms of memory overhead and topology changes but have a major shortcoming as the delay in path discovery increases with the increase in the number of the nodes in the network.

In order to overcome the disadvantages of pure table-driven or reactive protocols, hybrid protocols such as the zone routing protocol (ZRP) have been proposed [55]. Hybrid protocols exhibit behavior that is a combination of proactive and reactive routing schemes and aim to minimize the delay for route determination and optimize routing table updates for better utilization of network capacity.

Routing protocols can also be classified based on whether the underlying architecture is flat-routed or hierarchical. In flat-routed networks, all the nodes are alike and the routing is done based on peer-to-peer connections, restricted only by the propagation conditions. In hierarchical networks, there at least two layers. On the lower layer, routing in geographical proximity is done based on peer-to-peer connections and at least one of the nodes is designated as gateway to the higher layer. These gateway

nodes create the higher layer network. Thus, routing between nodes that belong to the same lower-layer network is based on peer-to-peer routing and routing between nodes that belong to different lower-layer networks is through the gateway nodes. The major advantage of hierarchical protocols is scalability. Cluster head formation schemes provide a very efficient and distributed solution for routing and thus scale well even for large networks. On the other hand creation and maintenance of clusters leads to extra overhead for routing protocol. Clusterhead Gateway Switch Routing (CGSR) is an example of hierarchical routing [21].

The routing protocols for ad-hoc wireless networks can also be classified according to distinct features that they implement. Location aided protocols use the location information of intermediate nodes and destination node to make routing decisions. Since nodes in ad hoc networks may move at any time, location aided protocols try to efficiently use the location information and node movement information for routing packets. Routing protocols can also be classified based on whether they support routing of unicast or multicast packets. A number of on-demand and table-driven multicasting routing protocols such as ODMRP [67], MCEDAR [95] etc. have been proposed for ad hoc networks. Most of the on-demand multicast routing protocols rely on significant periodic (non-on-demand) behavior within portions of the protocol. Routing protocols which support multicast provide an efficient means of supporting group-oriented applications.

3.1.1 Power-saving routing protocols

Power saving or energy efficient communication techniques are important in ad hoc wireless networks as devices may be battery operated and hence power constrained. The most common technique proposed is the power control scheme, in which a node transmits data packets to its neighbor at minimum power level. Recent studies (e.g. LAPAR [109], PAMAS [94]) have stressed the need for designing protocols both at MAC and network layers to ensure longer battery life.

In wireless networks, the power of transmitted signal is attenuated at the rate of $r^{-\alpha}$, where r is the distance between the sender and the receiver and α is the path loss exponent between 2 and 6. Consequently, transmitting data packets directly to the node may consume more energy that going through some intermediate nodes. Based on this observation, most of the proposed energy-efficient routing protocols try to find a path that has many short-range hops in order to consume the least amount of total energy. These protocols can be classified into three main categories

- Minimum total transmission power protocols: These protocols set the link cost to the transmission power and use a shortest path algorithm to search for the minimum energy path.

- Minimum total transceiving power protocols: As the intermediate nodes consume energy not only when forwarding packets but also when receiving packets, these protocols assign the transmission power and receiving power to the link cost metric.

- Minimum total reliable transmission power protocols: In these protocols, link cost is a function of both the energy required for a single transmission attempt across the link and link error rate, which determines the number of retransmission attempts needed for successful transmission.

Power aware multi-access protocol with signaling (PAMAS) is a multi-access MAC layer protocol which is based on the original MACA protocol with addition of a separate signaling channel [94]. PAMAS conserves battery power at nodes by intelligently powering off nodes that are not actively transmitting or receiving packets. The manner in which nodes power themselves off does not influence the delay or throughput characteristics of the PAMAS protocol. PAMAS searches for the minimum energy path by using Dijkstra's shortest path algorithm. Location-aided power-aware routing (LAPAR) protocol [109] is a location-aided power-aware routing protocol that dynamically makes local routing decisions so that a near-optimal power-efficient end-to-end route is formed for forwarding data packets.

3.1.2 Cross-Layer Design

Cross-layer design is a joint design optimization across several layers (e.g. physical, MAC and routing layers) under given resource constraints to improve network performance [31, 72]. For example, in sensor networks, the average transmission distance is in the order of a few meters. As a result, the circuit processing power becomes comparable to the transmission power. Therefore, for energy efficient network design, the transmission power and the circuit processing power need to be jointly considered in a cross-layer optimization problem. Cross-layer design involves information exchange between different layers, adaptivity at each layer to this information, and diversity built into each layer to insure robustness [53]. For example, the physical layer can deploy adaptive modulation and coding to compensate for time-varying wireless channel. This adaptivity could be used by higher layers to achieve better performance. The MAC layer can assign a longer channel usage time to links with low-rate modulation schemes to meet the throughput or energy constraints and the network or routing layer can reroute traffic to links supporting high-rate modulation schemes to minimize congestion. Though cross-layer optimization is beyond the scope of this thesis, we do consider the effect of an adaptive power strategy on the lifetime of ad hoc wireless networks in chapter 10.

3.2 Destination-Sequenced Distance-Vector Routing

The destination-sequenced distance-vector (DSDV) is a table-driven algorithm proposed by Perkins and Bhagwat and is based on the classical Bellman-Ford routing mechanism [82]. Every mobile node in the network implementing DSDV maintains a routing table. In routing tables of DSDV, an entry contains the next hop towards a destination, the cost metric for the routing path to the destination and a destination sequence number that is created by the destination. Sequence numbers are used in DSDV to distinguish stale routes from fresh ones and avoid formation of route loops.

The route updates of DSDV can be either time-driven or event-driven. Every node periodically transmits updates including its routing information to its immediate neighbors. While a significant change occurs from the last update, a node can transmit its changed routing table in an event-triggered style. Moreover, the DSDV has two ways when sending routing table updates. One is *full dump* update type and the full routing table is included inside the update. A *full dump* update could span many packets. An incremental update contains only those entries that with metric have been changed since the last update is sent. Additionally, the incremental update fits in one packet.

Broch *et al.* [16] have shown that DSDV performs well when node mobility rate and node movement speed are low, but has convergence problems when the node mobility increases. Furthermore, the routing protocol overhead increases as the network diameter and node mobility increase.

3.3 Dynamic Source Routing

The Dynamic Source Routing (DSR) is an on-demand routing protocol based on the concept of source routing [62]. Source routing is a routing technique in which the sender of a packet determines the complete sequence of nodes through which to forward the packet. The key advantage of source routing is that intermediate nodes do not need to maintain up-to-date routing information in order to route the packets they forward, since the packets themselves already contain all the routing decisions. Mobile nodes using DSR maintain route caches that contain source routes of which the node is aware.

There are two major phases in DSR, the route discovery phase and the route maintenance phase. When a source node wants to send a packet, it first consults its route cache. If the required route is available, the source node includes the routing information inside the data packet before sending it. Otherwise, the source node initiates a route discovery operation by broadcasting route request (RT_REQ) packets. A RT_REQ packet contains addresses of both the source and the destination and a unique number to identify the request. Receiving a RT_REQ packet, a node checks its route cache. If the node doesn't have routing information for the requested destination, it appends its own address to the route record field of the RT_REQ packet. Then, the

RT_REQ packet is forwarded to its neighbors. To limit the communication overhead of RT_REQ packets, a node processes RT_REQ packets that it has not seen before and its address is not present in the route record field. If the RT_REQ packet reaches the destination or an intermediate node has routing information to the destination, a route reply (RT_REP) packet is generated. When the RT_REP packet is generated by the destination, it comprises addresses of nodes that have been traversed by the route request packet. Otherwise, the RT_REP packet comprises the addresses of nodes the RT_REQ packet has traversed concatenated with the route in the intermediate node's route cache.

After being created, either by the destination or an intermediate node, a RT_REP packet needs a route back to the source. There are three possibilities to get a backward route. The first one is that the node already has a route to the source. The second possibility is that the network has symmetric (bidirectional) links. The route reply packet is sent using the collected routing information in the route record field, but in a reverse order as shown in Figure 3.2. In the last case, there are asymmetric (unidirectional) links and a new route discovery procedure is initiated to the source. The discovered route is piggybacked in the route request packet.

In DSR, when the data link layer detects a link disconnection, a route error (RT_ERR) packet is sent backward to the source. After receiving the RT_ERR packet, the source node initiates another route discovery operation. Additionally, all routes containing the broken link should be removed from the route caches of the immediate nodes when the RT_ERR packet is transmitted to the source.

Source routing protocols such as DSR face a scaling challenge as network diameter in hops and mobility increase because the product of these two factors determines the rate that end-to-end paths change. DSR must query longer routes as the network diameter increases, and must do so more often as the mobility increases, and caching becomes less effective.

3.4 Ad Hoc On-Demand Distance Vector Routing

AODV is an on-demand routing protocol proposed by *Perkins et al.* [81]. It is a combination of DSR and DSDV protocols. It borrows the basic on-demand mechanism of route discovery and route maintenance from DSR, plus the use of hop-by-hop routing, sequence numbers and periodic beacons from DSDV. AODV is an improvement over DSDV because it typically minimizes the number of required broadcasts by creating routes on a demand basis, rather than maintaining the complete list of routes as in the DSDV algorithm. As a reactive routing protocol, AODV only needs to maintain the routing information about the active paths. In AODV, each mobile node keeps a next-hop routing table, which contains the destinations to which it currently has a route. A routing table entry expires if it has not been used or reactivated for a pre-specified

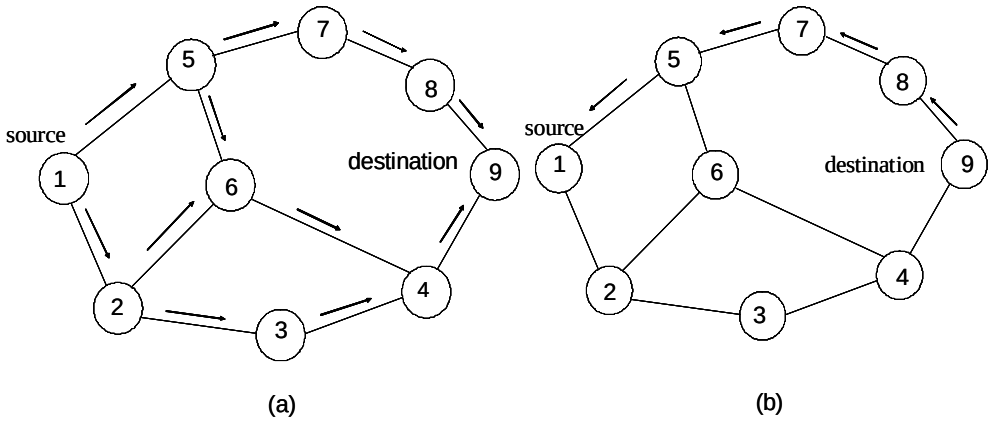


Figure 3.2: Route discovery in DSR (a) Route request (RT_REQ) packets (b) Route reply (RT_REP) packet

expiration time. AODV differs from DSR since it does not use source routing, rather relies on dynamically establishing route table entries at intermediate nodes.

In AODV, when a source node wants to send packets to the destination but no route is available, it initiates a route discovery operation. In the route discovery operation, the source broadcasts route request (RREQ) packets. A RREQ includes addresses of the source and the destination, the broadcast identity, which is used as its identifier, the last seen sequence number of the destination as well as the source node's sequence number. Sequence numbers are important to ensure loop-free and up-to-date routes. To reduce the flooding overhead, a node discards RREQs that it has seen before and the expanding ring search algorithm is used in route discovery operation. The RREQ starts with a small time-to-live (TTL) value. If the destination is not found, the TTL is increased in following RREQs.

Each node maintains a cache to keep track of RREQs it has received. The cache also stores the path back to each RREQ originator. When the destination or a node that has a route to the destination receives the RREQ, it checks the destination sequence numbers it currently knows and the one specified in the RREQ. To guarantee the freshness of the routing information, a route reply (RREP) packet is created and forwarded back to the source only if the destination sequence number is equal to or greater than the one specified in RREQ. AODV uses only symmetric links and a RREP follows the reverse path of the respective RREP. Upon receiving the RREP packet, each intermediate node along the route updates its next-hop table entries with respect to the destination node. The redundant RREP packets or RREP packets with lower destination sequence number will be dropped.

A node uses hello messages to notify its existence to its neighbors. Therefore, the

link status to the next hop in an active route can be monitored. When a node discovers a link disconnection, it broadcasts a route error (RERR) packet to its neighbors, which in turn propagates the RERR packet towards nodes whose routes may be affected by the disconnected link. Then, the affected source can re-initiate a route discovery operation if the route is still needed.

3.5 Summary

The above ad hoc routing approaches have introduced several paradigms such as exploiting user demand, and the use of location, power, and the association parameters. Both table-driven and on-demand routing protocols have their advantages and disadvantages and cannot be universally applied equally well to all networks. A flexible routing approach could be the solution. A flexible routing protocol could invoke table-driven and/or on-demand approaches based on situations and communication requirements. Coexistence of both approaches may also exist in spatially clustered ad hoc groups, with intracluster employing the table driven approach and intercluster employing demand driven approach or vice versa.

Part II

Ant Routing

Chapter 4

Introduction to ant routing

Stigmergy (from the Greek *stigma*: sting and *ergon*: work) is a concept first introduced by Paul Grasse in 1959 to explain the coordination and regulation of collective behavior in insect colonies [101]. A formal definition used in Biology is: stigmergy is a class of mechanisms that mediate animal-animal interactions. Most of the observations have been about collective behavior of insects though there has been some recent work on social interactions among higher species [101].

The basic principle of stigmergy is simple. Traces of chemicals left in the environment or modifications made by individuals in their environment are used as feedback. The insect colony records its activity in a physical environment and uses it to organize collective behavior. Various forms of storage include: gradients of chemical substance known as *pheromones*, material structures, or spatial distribution of insect colony. Such structures materialize the dynamics of the colony's collective behavior and constrain the behavior of individuals through a feedback loop. A simpler definition of stigmergy is sufficient for this thesis: stigmergy is a form of indirect communication mediated by modifications of the environment [101].

Stigmergy mechanisms have been classified into two categories. With quantitative stigmergy, the stimulus-response sequence comprises stimuli that does not differ qualitatively, and only modify the probability of response of the individuals to this stimuli. An example of quantitative stigmergy is the construction of pillars in termites studied by Grasse [101]. Qualitative stigmergy is based on discrete set of stimulus types: For example, an insect responds to type-1 stimulus with action A and responds to type-2 stimulus by action B. An example of the qualitative stigmergy is *nest building* in wasp *Polistes*.

The foraging behavior of ant colonies is an example of quantitative stigmergy. While walking from food sources to nest and vise versa, ants deposit pheromone on the ground, forming in his way a pheromone trail. Ants can smell pheromone and the probability of choosing paths marked by strong pheromone concentration increases. It has been shown experimentally that the pheromone trail following behavior of a colony of ants

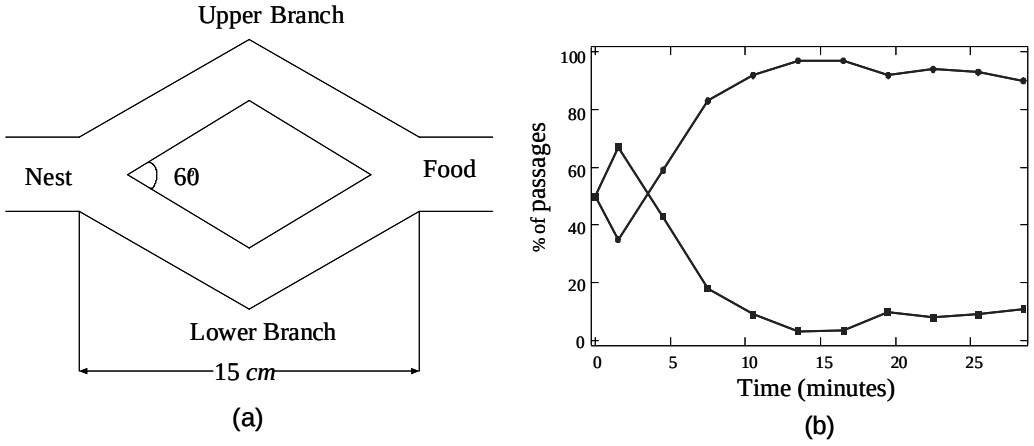


Figure 4.1: Foraging behavior of Argentine ant *Iridomyrmex humilis*. (a) Double bridge experiment setup (b) Percentage of ants per 3-min period passing on the two branches of the bridge. Colony of 1000 workers.

leads to the emergence of shortest paths [10, 36]. Deneubourg *et al.* [36] studied the foraging behavior of a colony of the Argentine ant called *Iridomyrmex humilis* under controlled conditions. The nest of the colony of ants was separated from the food source by a double bridge in which each bridge has the same length. Ants were left to move between the nest and food source. Initially both branches are chosen equally. However, in marking, each ant that passes modifies the following ant's probability of choosing left or right, a positive-feedback system that, after initial fluctuation, rapidly leads to one of the two bridges becoming preferred to the other. Thus, over a period of time the ants tend to converge on one of the two paths (Figure 4.1).

The observation that real ants are able to find the shortest paths lead to the development of ant routing algorithms [42, 92]. The idea behind ant routing algorithms is to use a form of stigmergy to coordinate societies of artificial agents. The artificial agents (mobile agents) or ants move through the network and are used to update the routing tables. The mobile agents update the routing tables in an asynchronous manner and independently of other mobile agents (Figure 4.2). Thus, ant routing combines the routing algorithm and protocol into a single entity and the nodes do not need to directly exchange routing information for updating the routing tables. This is in contrast to the traditional approach to routing where the routing tables are updated by exchanging routing information between the routers. For example, in OSPF the routers exchange link-state information by flooding.

In this thesis, we collectively refer to various algorithms that use mobile agents as ant routing algorithms (ANTRALs). The principles of ant colony and stigmergy have been applied to numerous other optimization problems besides routing and have been

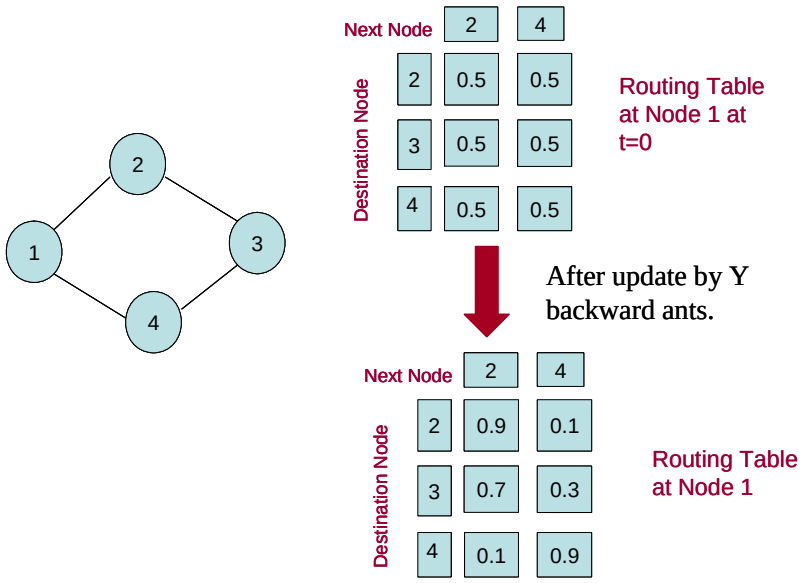


Figure 4.2: Routing table update and maintainance in ant routing.

referred to as ant colony optimization in the literature [45].

There are a number of differences between the traditional routing algorithms and ANTRALs. ANTRAL is a hop-by-hop routing approach. In an ANTRAL, traffic across the network is continuously monitored by the mobile agents. Due to the probabilistic routing in ANTRALs, the paths are not loop-free and oscillations might occur. The hop-by-hop nature of routing in ANTRALs cannot reserve paths that satisfy QoS constraints [39]. However, in ANTRALs, different paths are continuously monitored and quality of paths is reflected in the routing table values. In OSPF, traffic fluctuations and the end-to-end delay are not measured. The main advantage of ANTRALs over traditional routing protocols is that ANTRALs perform load balancing, i.e. distribute traffic along multiple paths, and automatic adaptation to node or link failure [92, 42, 37].

4.1 Overview of ANTRAL implementations

A number of routing algorithms inspired by the ant-colony metaphor and using mobile agents have been proposed for both wired and wireless networks. We summarize different ant routing algorithms in this section.

AntNet proposed by Di Caro and Dorigo [42] is a routing algorithm proposed for wired datagram networks based on the principle of ant colony optimization. In AntNet, each node maintains a routing table and an additional table containing statistics about the traffic distribution over the network. The routing table maintains for each destina-

tion and for each next hop a measure of the goodness of using the next hop to forward data packets to the destination. These goodness measures, called pheromone variables, are normalized to one in order to be used by a stochastic routing policy. AntNet uses two sets of homogeneous mobile agents called forward ants and backward ants to update the routing tables. The forward ants use heuristics based on the routing table to move between a given pair of nodes and are used to collect information about the traffic distribution over the network. The backward ants retrace the paths of forward ants in the opposite direction. At each node, the backward ants update the routing table and the additional table containing statistics about the traffic distribution over the network.

Ant-Based Control (ABC) is an algorithm proposed by Schoonderwoerd *et al.* [92] for load balancing in circuit-switched networks. In ABC, the calls are routed using probabilistic routing tables that consist of next hop probabilities for each destination. The link costs are assumed to be symmetric and hence, only one-directional mobile agents are used for updating and maintaining the routing tables. The mobile agents use heuristics based on the routing tables to move across the network between arbitrary pairs of nodes. At each node along the path, the mobile agents update the routing tables based on their distance from the source node and the current state of the routing table.

Another ANTRAL for wired networks has been proposed by Kuntz *et al.* [59]. The proposed algorithm differs from AntNet in terms of different loop detection behavior, simpler backward mobile agent and different routing table update procedure. The authors also propose another routing protocol called Co-operative Asymmetric Forward (CAF) routing. CAF is similar to ABC but it works for asymmetric networks where the link costs are not identical in opposite directions. CAF has been shown to perform as well as AntNet and is able to cope with changing bandwidth and network topology [59].

AntHocNet is a hybrid routing protocol proposed by Ducatelle *et al.* [47] for mobile ad hoc networks. AntHocNet consists of both the reactive and proactive components. In AntHocNet, nodes do not maintain routes to all possible destinations at all the times, rather the nodes generate mobile agents only at the beginning of a data session. The mobile agents search for multiple paths to the destination and these paths are set up in the form of pheromone tables indicating their respective quality. During the course of the data session, the paths are continuously monitored and improved in a proactive manner. AntHocNet has been shown to outperform AODV in terms of end-to-end delay and delivery ratio [47].

Ad hoc Networking with Swarm Intelligence (ANSI) is a reactive routing protocol proposed by Rajagopalan and Shen [86] for mobile ad hoc networks. ANSI protocol uses two sets of mobile agents called forward reactive ants and backward reactive ants. The routing tables in ANSI contains an entry for each reachable node and next best hop while the ant decision table stores the pheromone values. In ANSI, the forward reactive ants are generated on demand i.e., the forward reactive ants are generated only when a node needs to transmit data to another node. The forward reactive ants

are broadcast while the backward reactive ant retrace the path of forward reactive ant and update the pheromone values at the nodes. The data packets choose the next hop deterministically i.e., the hop which contains the largest pheromone value is chosen as the next hop. ANSI has been shown to perform either better or comparable with AODV with respect to packet delivery, end-to-end delay and delay jitter [86].

A number of routing protocols in which the mobile agents do not update the routing tables directly have also been proposed (e.g. Termite [89], Global Positioning System/Ant-Like Routing Algorithm (GPSAL) [18]). Termite, a routing protocol proposed by Roth and Wicker [89] for mobile ad hoc networks, is one such example. In Termite, the size of the routing table at each node varies and depends on the number of nodes that have been discovered by the particular node and the number of neighbors. The routing table entries in Termite contain pheromone values for choosing a neighbor as the next hop for each destination. The pheromone values decay exponentially with time and the corresponding entries are removed from the routing tables, if all the pheromone for a particular destination decays. Thus, the routing tables in Termite maintain entries for only the destinations from which packets have been received during recent times. Termite does not use mobile agents for updating the routing tables instead node discovery and route discovery and maintenance are performed by a set of four control packets: route request packets, route reply packets, hello packets and seed packets. Both data packets and control packets (except route request packets) are used for updating the routing tables.

Ant-Colony-Based Routing Algorithm (ARA) is a routing protocol proposed by Güneş and Spaniol [54] for mobile ad hoc networks. The routing table entries in ARA contain pheromone values for choosing a neighbor as the next hop for each destination. In ARA, the pheromone values in the routing tables decay with time and the nodes enter a sleep mode if the pheromone in the routing table has reached a lower threshold. Route discovery in ARA is performed on demand and by a set of two mobile agents - forward ants and backward ants. During route discovery, the forward and backward ant packets having unique sequence numbers, to prevent duplicate packets, are flooded through the network by the source and destination nodes respectively. The forward and backward ants update the pheromone tables at the nodes along the path for the source and destination nodes respectively. Once the route discovery for a particular destination has been performed, the source node does not generate new mobile agents for the destination instead the route maintenance is performed by the data packets.

Uniform ant routing algorithms are a class of ANTRALs in which the mobile agents choose the next node uniformly among the neighbors of the node [98]. Thus, in uniform ANTRALs, the mobile agents move independently of the routing tables and perform a random walk with memory on the network graph while searching for the destination. This feature of uniform ANTRALs reduces the complexity of the ant routing algorithm and leads to exploration of all the paths with equal probabilities.

4.2 Performance of Ant Routing Algorithms

There are two critical components that determine the performance of an ANTRAL. First, the performance depends on how the mobile agents search for the shortest path, which is referred to as *network exploration.* The mobile agents could either use heuristics based on the routing tables or the routing table values without any modifications to move to the next node. The mobile agents could also search independently of the routing tables. We introduce an example of such algorithm in chapter 5.

The second critical step determining the performance of ANTRALs is the routing table update. The mobile agents have to update the routing tables based on the paths that have been searched.

There has been very little work in studying the performance of ANTRALs. To our knowledge, the only work is by Bean and Costa [9], Yoo *et al.* [110] and recently by Purkayastha and Baras [85]. Bean and Costa [9, 30] have conducted analytic modelling of ANTRALs and have shown that since ANTRALs employ the same policy for network exploration and decision-making, it leads to sub-optimal routing decisions. Indeed, they have conjectured that a majority of the ants choose shortest paths while some of the ants choose sub-optimal paths. Furthermore, they propose algorithms for de-coupling of network exploration and routing table updates when there is no data traffic and for varying traffic.

The exact convergence of ANTRAL to shortest path routing for static topology and link weights has been studied analytically by Yoo *et al.* [110]. However, the results are only for a two node network with multiple links. Recently, Purkayastha and Baras [85] have used ordinary differential equations (ODE) to analyze the performance of ANTRALs.

We study the performance of ANTRALs in the next couple of chapters. In chapter 5, we analyze the performance of AntNet, an ANTRAL proposed by Di Caro and Dorigo [42]. In chapter 6, we study the performance of ANTRALs for ad hoc wireless networks. We compare the performance of W_AntNet, a modification of the AntNet algorithm, with ad hoc routing protocols AODV and DSR.

Chapter 5

Ant Routing in Wired Networks

In this chapter, we study the convergence of AntNet to shortest path for static link weights and fixed topologies. We compare the performance of AntNet with Dijkstra's shortest path algorithm. Although AntNet has been shown by Di Caro and Dorigo to perform load balancing[1], a study of whether AntNet is able to find paths close to the shortest path is missing. Such an analysis is essential for widespread deployment of ANTRALs.

In all the ANTRALs proposed so far, the algorithm parameters have been chosen heuristically. There has been little work on studying the robustness of ANTRALs to the variations in different parameter values. Furthermore, the performance analysis of ANTRALs has been limited to a few network topologies. We study the effect of different parameters used for network exploration and routing table updates on the performance of AntNet algorithm. We also study the impact of topology on the performance of AntNet algorithm.

The rest of this chapter is organized as follows. The next section presents the network model and section 5.2 presents the details of AntNet algorithm. Section 5.3 presents the simulation results. Finally, section 5.4 presents the conclusions.

5.1 Network Model

We model the network as a graph $G(N,L)$ consisting of N nodes and L links. All the links in the network are considered bidirectional and specified by a transmission capacity and a transmission delay. Each node is considered a communication end-point (host) and a forwarding unit (router).

As illustrated in Figure 5.1, every node in the network maintains an input buffer composed of a single queue and an output buffer composed of a high priority queue and a low priority queue for each neighbor or outgoing link. The high priority queue is

[1] Schoonderwoerd [92] showed load balancing for another ANTRAL.

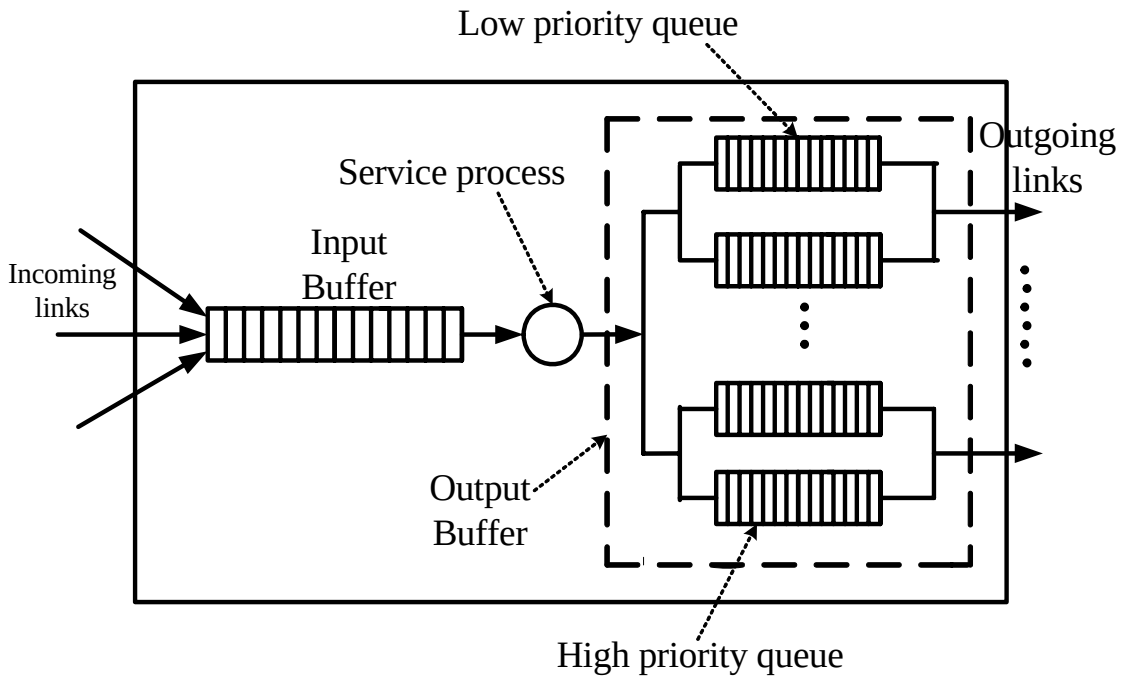


Figure 5.1: Buffers at a Node. The input buffer consists of a single queue and the output buffer consists of a low priority queue and a high priority queue for each outgoing link.

served before the low priority queue. All the packets within the network can be divided into two different classes:

- Data packets: represent the information that the end-users exchange with each other. In ant-routing, data packets do not maintain any routing information but use the information stored at routing tables for travelling from the source to the destination node.

- Mobile agents (Forward ants and Backward ants): are used to update the routing tables and distribute information about the traffic load in the network.

Backward ant packets have a higher priority than the data and forward ant packets and are thus stored in the high priority queue, while data and forward ant packets are stored in the low priority queue. We assume that all the packets in the low priority queue and the high priority queue in the output buffer are served in FIFO order. Further, the maximum number of packets stored in the input buffer or output buffer is limited by the size of the buffer. However, we have assumed that the buffer size is sufficiently large to neglect buffer overflow.

When a node receives a packet from a neighbor, the packet is first stored in the input buffer. The packet in the input buffer is served in FIFO order. After the packet has been served, the packet is sent to the output buffer. Within the output buffer, the packet goes to a particular queue for a particular outgoing link based on the type of the packet and the next node.

5.1.1 Data Structures at Nodes

Mobile agents communicate in an indirect way, through the information they concurrently read and write in two data structures stored at each network node k:

1. A routing table R_k, organized as a matrix with probabilistic entries as shown in Figure 5.2. Each row in the routing table corresponds to one destination in the network and each column corresponds to a neighbor of the current node. The routing table R_k defines the probabilistic routing policy currently adopted at node k: for each possible destination d and for each neighbor node n, R_k stores a probability value p_{nd} expressing the probability of choosing n as the next node when the destination is d such that:

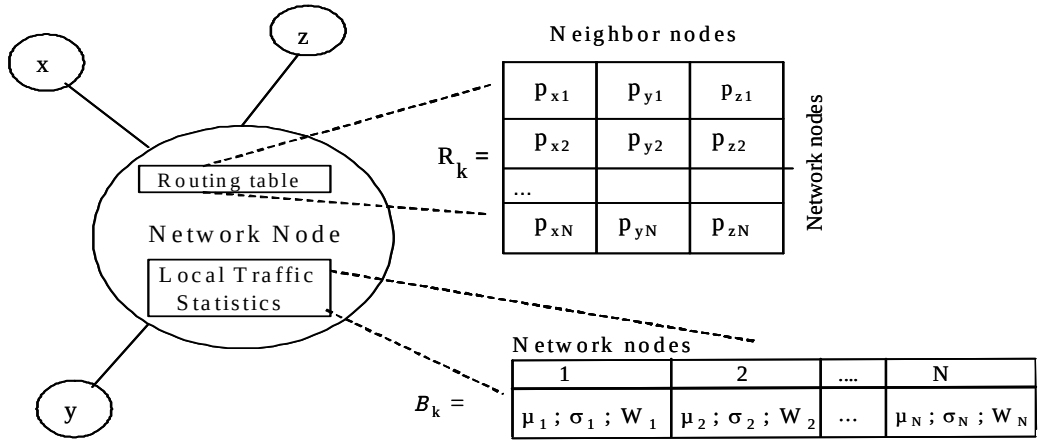
$$\sum_{n \in N_k} p_{nd} = 1$$

   where $d \in [1, N]$ and $N_k = \{neighbors\,(k)\}$

2. A table $\mathcal{B}_k(\mu_d, \sigma_d^2, W_d)$ containing statistics about the network topology and the traffic distribution over the network as seen by the local node k. For each destination d in the network, the table $\mathcal{B}_k$ contains a moving observation window W_d, an estimated mean μ_d and an estimated variance σ_d^2. The moving observation window W_d, of size $W_{\max}$, represents an array containing the trip times of the last $W_{\max}$ forward ants that travel from node k to destination d. The moving observation window W_d is used to compute the best trip time t_{best_d} i.e., the best trip time experienced by a forward ant travelling from node k to destination d among the last $W_{\max}$ forward ants that travel from node k to destination d. The mean μ_d and variance σ_d^2 represent the mean and variance of the trip times experienced by the forward ants to move from node k to destination node d and are calculated using the exponential model:

$$\mu_d \longleftarrow \mu_d + \eta\,(t_{k \longrightarrow d} - \mu_d) \tag{5.1}$$

$$\sigma_d^2 \longleftarrow \sigma_d^2 + \eta\left((t_{k \longrightarrow d} - \mu_d)^2 - \sigma_d^2\right) \tag{5.2}$$



(a) Routing table: For any destination node i, $p_{xi} + p_{yi} + p_{zi} = 1$ (b) Statistic Table: The mean μ_i and variance s_i represent the estimated mean and variance of the end-to-end delay to the node i. The moving observation window W_i is an array containing the end-to-end delay for node i.

Figure 5.2: The data structures for a node with neighbors x, y and z and a network with N nodes: Routing table (R_k) and Statistics Table ($\mathcal{B}_k$).

In (5.1) and (5.2), $t_{k \longrightarrow d}$ represents the newly observed forward ant's trip time to travel from node k to destination node d and $\eta \in (0, 1]$ is a factor that weighs the number of recent samples that will affect the mean μ_d and the variance σ_d^2. Di Caro & Dorigo [42] relate η to the maximum size of the observation window $W_{\max}$ by

$$W_{\max} = \frac{5c}{\eta} \text{ where } c < 1 \tag{5.3}$$

Di Caro & Dorigo [42] calculated that $\frac{5}{\eta}$ samples affect the mean so (5.3) has been used to ensure that the mean and the best trip time are calculated over the same moving observation window. We choose the maximum size of the moving observation window $W_{\max}$ to be sufficiently large i.e., $W_{\max} > \frac{5}{\eta}$ but independently of the parameter η.

5.2 AntNet Algorithm

5.2.1 Description of the AntNet algorithm

The AntNet algorithm [42] can be described as follows:

1. At regular intervals, from every network node s, a forward ant $F_{s \longrightarrow d}$ is launched

with a randomly selected destination node d. Destinations are chosen to match the current traffic patterns i.e., if f_{sd} is a measure (in bits or in the number of packets) of the data flow $s \longrightarrow d$, then the probability y_d of creating at node s a forward ant with node d as destination is:

$$y_d = \frac{f_{sd}}{\sum_{d'=1}^{N} f_{sd'}} \tag{5.4}$$

2. While travelling towards their destination nodes, the forward ants store their paths and the traffic conditions. The identifier of every visited node k and the time elapsed since the launching time of the forward ant to arrive at this k-th node are pushed onto a memory stack $\mathcal{S}_{s \longrightarrow d}$ stored in the data field of the forward ant[2]. Forward ants share the same queues as data packets, so they experience the same traffic delays as data packets.

3. At each node k, each forward ant chooses the next node as follows:

   - If not all the neighboring nodes have been visited, then the next neighbor is chosen among the nodes that have not yet been visited as:

   $$p'_{nd} = \frac{p_{nd} + \ \alpha b_n}{1 + \ \alpha \ (|N_k| - 1)} \tag{5.5}$$

   In (5.5), N_k represents the set of neighbors of node k and $|N_k|$ the cardinality of that set, i.e., the number of neighbors while the heuristic correction b_n is a normalized value $[0, 1]$ such that $1 - b_n$ is proportional to the length q_n of the queue of the link connecting the node k with its neighbor n:

   $$b_n = 1 - \frac{q_n}{\sum_{n'=1}^{|N_k|} q_{n'}} \tag{5.6}$$

   The value of α in (5.5) weighs the importance of the instantaneous state of the node's queue with respect to the probability values stored in the routing table.

   - If all the neighboring nodes have been visited previously, then the next node is chosen uniformly among all the neighbors. In this case, since all the neighbors have been visited previously the forward ant is forced to return to a previously visited node. Thus, irrespective of which neighbor is chosen as the next node, the forward ant is in a loop (cycle).

[2]The accuracy of time stored by forward ant depends on clock synchronization in the network. We assume that network is synchronized with accuracy of few ms.

- With a small probability ε, the next node may be chosen uniformly among all the neighboring nodes. The parameter ε is deliberately incorporated in the ANTRAL to overcome the problem where one of the entries in the routing table is almost unity, while the other are vanishingly small. In such a situation, the forward ants always choose the same link and thus stop exploring the network for other routes. The parameter ε ensures that the network is being constantly explored, though it introduces an element of inefficiency in the algorithm. The use of parameter ε to introduce randomness in the algorithm is referred to as ε-*greedy* policy in reinforcement learning literature [99]. Thus, due to the ε-*greedy* policy there are no restrictions on the routing table values and some of the routing table entries may be zero. In AntNet implementation [42, 43], the value of parameter ε is chosen as zero. In the special case, when $\varepsilon = 1$, the AntNet algorithm is a uniform ANTRAL (unfNet).

4. If a cycle is detected, that is, if the ant is forced to return to an already visited node, the cycle's nodes are popped from the ant's stack and all memory about the cycle is destroyed. If the cycle lasted longer than the lifetime of the forward ant before entering the cycle, the ant is destroyed (Figure 5.3). The lifetime of a forward ant is defined as the total time since the forward ant was generated.

5. When the destination node d is reached, the forward ant $F_{s \longrightarrow d}$ generates a backward ant $B_{d \longrightarrow s}$. The forward ant transfers all the memory contained in the stack $\mathcal{S}_{s \longrightarrow d}$ to the backward ant, and dies.

6. The backward ant takes the same path as the corresponding forward ant, but in the opposite direction. At each node k, the backward ant pops the stack $\mathcal{S}_{s \longrightarrow d}$ to move to the next node. Backward ants do not share the same queues as data packets and forward ants; they use high priority queues to quickly propagate to the routing tables the information collected by the forward ants.

7. Arriving at a node k coming from a neighbor node s', the backward ant updates the two main data structures of the node, the local model of the traffic $\mathcal{B}_k$ and the routing table R_k, for all the entries corresponding to the destination node d. The update of routing tables at each node along the path as the backward ant travels from the destination to the source node is known as sub-path update method[3].

   - The mean μ_d and variance σ_d^2 entries in the local model of traffic $\mathcal{B}_k$ are modified using (5.1) and (5.2). The best value t_{best_d} of the forward ants

[3] If the cycles are not removed from the forward ant's path, then the sub-path update will lead to statistical bias [30]. Thus, the sub-path update should be used only if the cycles are removed from the forward ant's path.

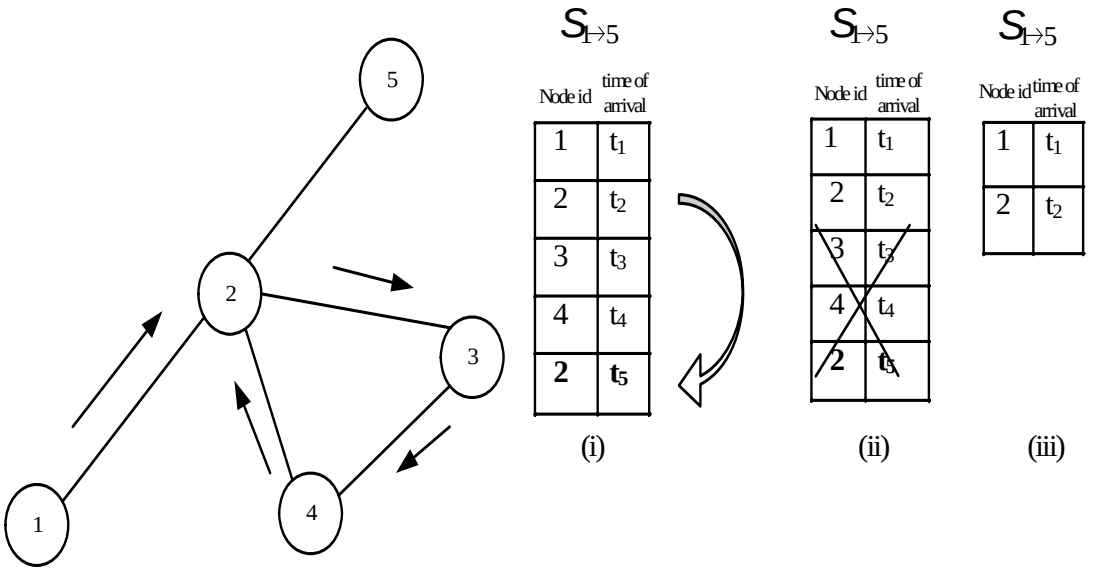


Figure 5.3: (i) A cycle $(2 \rightarrow 3 \rightarrow 4 \rightarrow 2)$ in the forward ant's $F_{1 \longrightarrow 5}$ path is detected. (ii) All the memory about the cycle $(2 \rightarrow 3 \rightarrow 4 \rightarrow 2)$ is destroyed. (iii) The stack $S_{1 \longrightarrow 5}$ maintained by forward ant $F_{1 \longrightarrow 5}$ after the removal of cycle. Further if the time spent in cycle $(t_5 - t_2)$ is greater than the lifetime of the ant before the cycle $(t_2 - t_1)$, the forward ant $F_{1 \longrightarrow 5}$ is destroyed.

trip time from node k to the destination d stored in the moving observation window W_d is also updated by the backward ant. If the newly observed forward ant's trip time $t_{k \longrightarrow d}$ from node k to destination d is less then t_{best_d}, then t_{best_d} is replaced by $t_{k \longrightarrow d}$.

- The routing table R_k is changed by incrementing the probability $p_{s'd'}$ (i.e., the probability of choosing neighbor s' when the destination is d') and decrementing, by normalization, the other probabilities $p_{nd'}$.The probability $p_{s'd'}$ is increased by the reinforcement value r as:

$$p_{s'd'} \longleftarrow p_{s'd'} + \varpi \, (1 - p_{s'd'}) \tag{5.7}$$

The probabilities $p_{nd'}$ of the other neighboring nodes n for destination d' are decreased by the negative reinforcement as:

$$p_{nd'} \longleftarrow p_{nd'} - \varpi p_{nd'}, \; \forall \; n \neq s', \; n \in N_k \tag{5.8}$$

Thus, in AntNet, every path found by the forward ants receives a positive reinforcement.

- The reinforcement value ϖ used in (5.7) and (5.8) is a dimensionless constant $(0, 1]$ and is calculated as:

$$\varpi = c_1 \frac{t_{best_d}}{t_{k \longrightarrow d}} + c_2 \frac{t_{sup} - t_{best_d}}{(t_{sup} - t_{best_d}) \; + \; (t_{k \longrightarrow d} - t_{best_d})} \tag{5.9}$$

In (5.9), $t_{k \longrightarrow d}$ is the newly observed forward ant's trip time from node k to the destination d and t_{best_d} is the best trip time experienced by the forward ants traveling towards the destination d over the observation window W_d. The value of t_{sup} is calculated as:

$$t_{sup} = \mu_d + \frac{\sigma_d}{\sqrt{1 - \gamma}\sqrt{|W_{max}|}} \tag{5.10}$$

where γ is the confidence level[4]. Equation (5.10) represents the upper limit of the confidence interval for the mean μ_d, assuming that the mean μ_d and the variance σ_d^2 are estimated over W_{max} samples [68]. There is some level of arbitrariness in choosing the confidence interval in (5.10) since the confidence interval is asymmetric and the mean μ_d and the variance σ_d are not arithmetic estimates [45]. The first term in (5.9) evaluates the ratio between

[4]Consider a sample of observations $\mathbf{X} = (X_1, ..., X_n)$. For any parameter δ defined for $\mathbf{X}$, find an interval $[l(\mathbf{X}), u(\mathbf{X})]$ such that $P[l(\mathbf{X}) \leq \delta \leq u(\mathbf{X})] = 1 - \phi$ i.e., the interval contains the true value of the parameter δ with probability $1 - \phi$. In such a case the interval $[l(\mathbf{X}), u(\mathbf{X})]$ is a $(1 - \phi) \times 100\%$ confidence interval and $1 - \phi$ is the confidence level [68].

the current trip time and the best trip time observed over the moving observation window. The second term is a correction factor and indicates how far the value of $t_{k \longrightarrow d}$ is from t_{best_d} in relation to the extension of the confidence interval [45]. The values of $c_1$and c_2 indicate the relative importance of each term. It is logical to assume that the first term in (5.9) is more important than the second term. Hence, the value of c_1 should be chosen larger than the value of c_2.

The value ϖ calculated in (5.9) is finally transformed by means of a squash function $s(x)$ defined by:

$$s(x) = \frac{1}{1 + \exp\left(\frac{a}{x|N_k|}\right)} \ , \text{ where } x \in (0, 1], \ a \in \mathbb{R}^+ \tag{5.11}$$

$$\varpi \longleftarrow \frac{s(\varpi)}{s(1)} \tag{5.12}$$

The squash function $s(x)$ is introduced in the AntNet algorithm so that small values of ϖ would have negligible effect in updating the routing tables [42]. Due to the squash function $s(x)$, the low values of ϖ are reduced further, and therefore do not contribute in the update of routing tables[5]. The coefficient $\frac{a}{N_k}$ determines the dependence of squash function $s(x)$ on the number of neighbors N_k of the node k. Figure 5.4 shows the effect of coefficient $\frac{a}{N_k}$ on the squash function $s(x)$. Figure 5.4 shows that if the value of coefficient $\frac{a}{N_k}$ is less than 1, then even low values of ϖ get incremented due to the squash function $s(x)$. Thus, the value of parameter a should be chosen such that the coefficient $\frac{a}{N_k}$ is greater than 1.

Data packets use different routing tables than the forward ants for travelling from the source node to the destination node. The routing table values for data packets are obtained by re-mapping the routing table entries used by forward ants by means of a power function $g(v)$ and re-normalizing these entries.

$$g(v) = v^\beta, \beta > 1 \tag{5.13}$$

The power function $g(v)$ emphasizes the high probability values and reduces lower ones, and thus prevents the data packets from choosing links with very low probability. The data packets have a fixed TTL; if the data packets do not arrive at the destination within the TTL, they are dropped.

[5]Low values of r indicate sub-optimal paths.

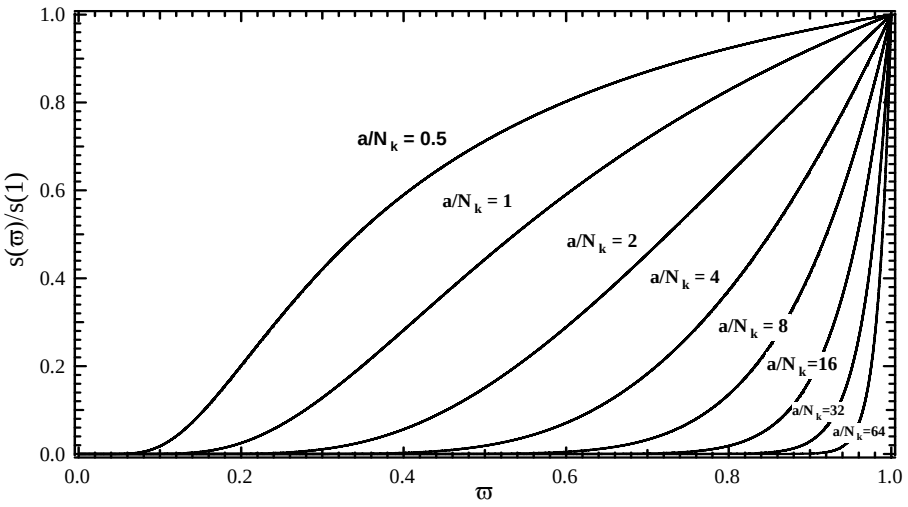


Figure 5.4: The squash function $s(x)$ for different values of the coefficient $\frac{a}{N_k}$

5.2.2 Complexity Analysis of the AntNet algorithm

We first calculate the complexity, defined as number of elementary operations, of a single forward ant to travel between a given source node and a given destination node in the AntNet algorithm. At every node along the path between a given source and destination node, the forward ant needs to search through the stack it maintains in the memory to find whether to use (5.5) for choosing the next node or to choose the next node uniformly among the neighbors. The worst-case complexity of searching through the stack is $O(1)$, if the stack is implemented as a combination of linked list and an additional array or an hash table. Further, the complexity of (5.5) is $O(N_k)$ since the probability values have to be calculated for each of the N_k neighbors. In unfNet, the forward ants choose the next node uniformly among the neighbors of the node, and therefore the above operations are not required. There are other computations that a forward ant performs at each node: the forward ant needs to push the identifier of the current node and the time at which it arrived onto the stack which is $O(1)$, the forward ant goes to the queue for one of the outgoing links which is $O(1)$. Let the maximum hopcount of the forward ant be h. In the worst case, the forward ant has to do all the computations at each of the h nodes. Thus, the worst-case complexity for a single forward ant to travel between a given source node and a destination node in AntNet is $O(hN_k)$, while in unfNet it is $O(h)$.

The worst-case complexity for a single backward ant to travel between a given source node and a destination node is also $O(hN_k)$. This can be calculated as follows. The backward ant performs three operations at each node along the path it travels between

a given source node and a given destination node. First, the backward ant needs to pop the stack to find the next node to travel to. The pop operation in the stack is $O(1)$. Second, the backward ant goes to the queue for one of the outgoing links which is $O(1)$. The backward ant updates the routing table for each of the neighbors N_k for the destination d which is $O(N_k)$. Furthermore, in the worst case the backward ant has to do all the computations at each of the h nodes.

We calculate the worst-case complexity of AntNet when the total number of forward or backward ants generated is given by $\rho > 1$. Since the worst-case complexity of a single forward or backward ant to travel between a given pair of nodes is $O(hN_k)$, the worst-case complexity of AntNet when ρ forward or backward ants are generated is $O(\rho hN_k)$. Furthermore, we know that the worst-case complexity for Dijkstra's shortest path algorithm using a Fibonacci heap is $O(N \log N + L)$. This analysis shows that the complexity of AntNet algorithm is comparable to Dijkstra's' shortest path algorithm when h and N_k are small as compared to N.

We now calculate the worst-case complexity of AntNet for finding the shortest path. Let us assume that one forward ant searches for one distinct path and one forward/backward ant pair updates only the routing tables at the source node for the given destination node. In the worst case, at any given node there is an equal probability of creating a forward ant with any one of the $N-1$ nodes as the destination. Thus, only one out of $N-1$ forward ants is used to search for the shortest path between a given source and destination. Let us denote the number of paths between a source and a destination in $G(N,L)$ by $\mathfrak{p}$. Adding the contributions yields a worst-case complexity to search for the shortest path in AntNet with $\mathfrak{p} = \mathfrak{p}_{\max}$ of $O(\mathfrak{s}hN_kN)$, where $\mathfrak{p}_{\max}$ is the upper bound on the maximum number of paths between a source and destination node in $G(N,L)$. The upper bound $\mathfrak{p}_{\max}$ on the total number of paths between a source and destination node in $G(N,L)$ is [105],

$$\mathfrak{p}_{\max} = [e(N-2)!] \tag{5.14}$$

In case of a random graph $G_p(N)$, with a fixed link density p and N nodes, the following upper-bound for the average number of paths applies [106],

$$\mathfrak{p}_{max} = \left[p^{N-1}e^{\frac{1}{p}}(N-2)!\right] \tag{5.15}$$

Equation (5.15) shows that the number of paths between a given pair of nodes decreases as the link density p in the random graph $G_p(N)$ is decreased.

5.2.3 AntNet Implementation

The time $T_{u \longrightarrow v}$ for a data packet or ant packet (forward ant or backward ant) to travel from a node u to a node v is calculated as:

$$T_{u \longrightarrow v} = \frac{q_v + size_{Packet}}{C_{u \longrightarrow v}} + D_{u \longrightarrow v} \tag{5.16}$$

where q_v is the length of the low priority queue at node u for the link $u \longrightarrow v$, $size_{Packet}$ is the size of the packet, $C_{u \longrightarrow v}$ is the available capacity of the link between node u and node v and $D_{u \longrightarrow v}$ is the propagation delay of the link $u \longrightarrow v$. The term $\frac{size_{Packet}}{C_{u \longrightarrow v}}$ represents the transmission delay for the packet and the term $\frac{q_v}{C_{u \longrightarrow v}}$ represents the queuing delay experienced by the packet while waiting at node u for the link $u \longrightarrow v$. The term propagation delay and link weight have been used interchangeably in the text.

We have simplified the model shown in Figure 5.1 for our implementation of the AntNet algorithm. We assume that the packets go directly to the outgoing buffer and the outgoing buffer consists of a low priority and a high priority queue for all the outgoing links. But the queuing delay in (5.16) and parameter b_n in (5.5) are still calculated using the number of packets waiting in the low priority queue for a particular link[6]. Further, we assume that each node is able to remove one packet from its high priority queue and low priority queue in the output buffer (for any outgoing link) at a rate of 0.01 milliseconds (ms). This assumption makes the queueing delays negligible in most of our simulations.

We implement the AntNet algorithm under static and dynamic conditions. In the static implementation of AntNet, we assume that the forward ants use only the propagation delays for network exploration and routing table updates. Thus, no queueing or transmission delays that depend on the packet size and capacity of the links are taken into account during the static implementation of AntNet. In this case, the equation (5.5) reduces to $p'_{nd} = p_{nd}$ and equation (5.16) for forward ant packets reduces to $T_{u \longrightarrow v} = D_{u \longrightarrow v}$. In the dynamic implementation of AntNet, both the queueing and transmission delays as well as the propagation delays are used for choosing the next node and updating the routing tables.

We compare the results of AntNet with Dijkstra's shortest path algorithm. However, under dynamic implementation, there is an important distinction for comparison of AntNet to Dijkstra's shortest path algorithm. In Dijkstra's shortest path algorithm, the cost of the shortest path is the sum of individual link weights (propagation delays) along the path. However in AntNet, the cost of the shortest path used for updating the routing tables is assumed to be the sum of link weights along the path and the transmission and queuing delays. The delay in the queues is included in the AntNet algorithm to account for the traffic conditions in the network. Thus, there is no static shortest path in the AntNet algorithm. However, for the comparison of AntNet to Dijkstra's shortest path algorithm, we plot the weight of the paths in the AntNet algorithm excluding the transmission and queuing delays.

To validate our simulations, in addition to plotting the pdf of the hopcount for the

[6]The packets in the high priority queue and low priority queue are served in FIFO order.

shortest path and the paths in the AntNet algorithm, we also plot the values for pdf of hopcount of the shortest path obtained by using theory. It has been demonstrated by van der Hofstad *et al.* [104] that for a fixed link density p and sufficiently large N, the shortest path tree in a random graph $G_p(N)$ with uniformly distributed link weights is a uniform recursive tree (URT). The probability density function of the hopcount in the URT with N nodes is:

$$\Pr[H_N = j] = \frac{(-1)^{N-(j+1)} S_N^{(j+1)}}{(N-1)\,(N-1)!} \tag{5.17}$$

where $S_N^{(j)}$ is the Stirling number of the first kind [1].

5.3 Results

5.3.1 Simulation Parameters

The simulations are performed on random graphs of the class $G_p(N)$ consisting of N nodes and independently chosen links with probability p. The link weights reflecting the propagation delays are uniformly distributed in $(0, 1]$ ms and the capacity of each link is 8.192 Mbit/s for all our simulations. For each random graph of the type $G_p(N)$, we used Dijkstra's algorithm to compute the shortest path between the source node and the destination node based on the link weights. On the other hand, in AntNet, the data packets travel between a given pair of nodes using the probabilistic routing tables. Indeed, there could be a number of paths that the data packets choose which might be the shortest path, the second shortest path etc. For AntNet, we find the average link weight (or end-to-end delay) and the average hopcount of the paths that the data packets use to travel between the source and the destination node over the entire simulation period. Furthermore, each simulation consists of a large number (10^5 or 10^4) of random graphs of the type $G_p(N)$. The source (node 1) and destination (node N) are chosen to be fixed in our simulations.

In our implementation of the AntNet algorithm, the simulation period (S.P.) consists of the training period (T.P.) and the test period (TE.P.). During T.P., only ant packets are generated while during TE.P., both the data and ant packets are generated. We have chosen the simulation period (S.P.) to be 10^4 ms. The training period (T.P.) has been chosen to be 10^3 ms. Each node generates data packets according to a Poisson process with mean interarrival time of 12.5 ms. The destination for data packets is chosen randomly. The size of data packets generated follows a negative exponential distribution with mean of 4096 bits. The size of forward ant packet is assumed to be 192 bits at the time the forward ant packet is generated and the size increases by 64 bits for each hop the forward ant travels. The size of backward ant is assumed to be 500

bits. The average number of neighbors of a node in a random graph $G_p(N)$ is $p(N-1)$, the default value of parameter a is chosen as $p * N$ in our simulations.

Table 5.1: Various parameters and their default values used in the simulations.

Name, Symbol	Value
Link Capacity $C_{i \longrightarrow j}$	8.192 Mbit/s
Link Weight (Propagation delay)	uniformly distributed (0,1] ms
Simulation period (S.P.)	10^4 ms
Test period (TE.P.)	9.10^3 ms
Training period (T.P.)	10^3 ms
Backward ant size	500 bits
Initial forward ant size	192 bits
α used in (5)	0.2
ε	0.1
c_1 used in (9)	0.7
c_2 used in (9)	0.3
a used in (11)	$p * N$
β used in (13)	3
Confidence interval (γ) used in (10)	0.95
Output buffer size	10^9 bits
Mean interarrival time for data packets	12.5 ms
Data packets TTL	10 ms
Mean data packet size	4096 bits

The maximum size of the output buffer, which is assumed to be the sum of the sizes of all low priority queues in the output buffer, is 10^9 bits. Table 5.1 lists the various parameters and their default values that we have chosen for our simulations. The values of parameters α, c_1, c_2, a, confidence interval (γ) and the size of forward ant packet have been replicated from [45]. The values of the parameter η, the maximum size of the observation window $W_{\max}$ and the ant generation rate have been varied extensively and therefore are not listed in Table 5.1.

5.3.2 Static Implementation of the AntNet algorithm

In this section, we study the AntNet algorithm under static conditions i.e., the forward ants use only the propagation delays for network exploration and routing table updates. Under these conditions, a direct comparison between the static shortest path calculated by using Dijkstra's algorithm and the AntNet algorithm paths can be made. The ant

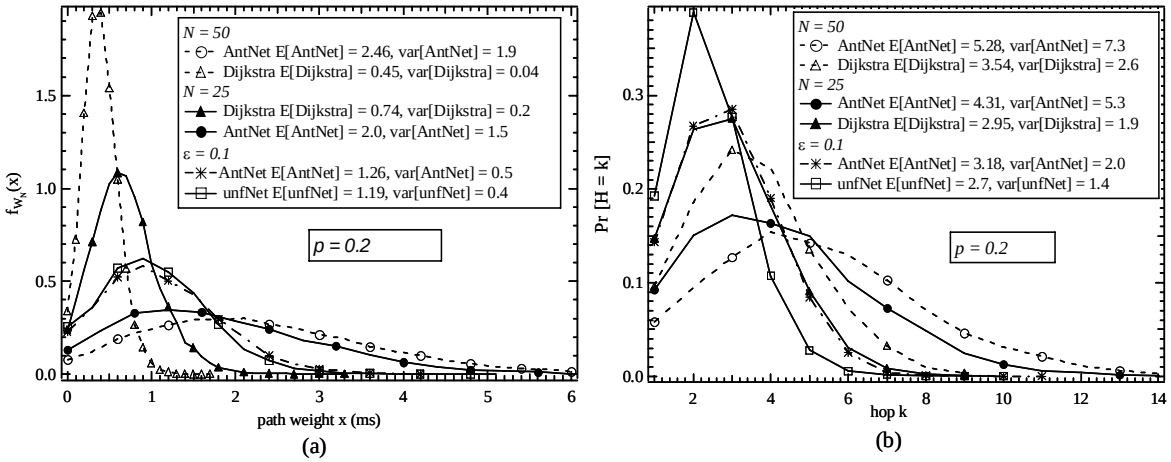


Figure 5.5: The pdf of the (a) weight and the (b) hopcount of the Dijkstra's shortest path, unfNet algorithm paths and the AntNet algorithm paths for $N = 25$ and $N = 50$ for $p = 0.2$. The ant generation interval during the T.P. is 4 ms. Each simulation consists of 10^4 iterations. ($\eta = 0.1$, $W_{\max} = 50$)

generation interval is 4 ms during T.P. and 40 ms during TE.P. The *TTL* of data packets is assumed to be 20.0 ms. The value of parameters $\eta = 0.1$ and $W_{\max} = 50$. Figure 5.5 shows the simulation results comparing the pdf of the hopcount and the weight for the shortest path and the AntNet algorithm paths[7] for $N = 25, 50$ and $p = 0.2$ (The default value of parameter ε is chosen as 0.)

Figure 5.5 shows that the AntNet algorithm converges to a good solution. Furthermore, the performance of AntNet improves when the value of parameter ε is chosen greater than 0. Since no queuing delays are considered, the network exploration in AntNet is restricted by the routing tables and many paths are not explored (The value of p'_{nd} in equation (5.5) is equal to p_{nd}). With $\varepsilon = 0.1$, the probability of exploring different paths increases leading to an improvement in the performance of AntNet. Indeed for static implementation, unfNet performs better than AntNet using $\varepsilon < 1$ since all the paths are explored with equal probabilities in unfNet.

[7]The probability density function (pdf) of the hopcount and the weight for the AntNet algorithm paths and the shortest path are computed over (10^4) random graphs of the type $G_p(N)$. In AntNet, for each random graph of the type $G_p(N)$, the path weight and hopcount represent the "average path weight" and "average hopcount" for all the packets to travel between the source and destination node during the entire simulation period.

5.3.3 Dynamic Implementation of the AntNet algorithm

In this section, the AntNet algorithm implementation is dynamic but we compare the results with Dijkstra's shortest path algorithm. We also study the effect of different parameters, such as ant generation rate, squash function $s(x)$, parameter β etc., on the performance of AntNet. The performance comparison between AntNet and Dijkstra's algorithm and the optimization of various parameters for AntNet is still valid since the transmission delays are very small for forward ant packets. Furthermore, the queueing delays are also small since packets are removed from nodes at a very fast rate as compared to the rate at which packets are generated. Thus, the total delay experienced by forward ant packets is still dominated by the propagation delays. The *TTL* of data packets is assumed to be 10.0 ms and the value of parameter ε is 0.1 in all the simulations in this section[8].

The effect of the ant-generation rate and the link density p

We first compare the performance of AntNet with Dijkstra's shortest path algorithm for different ant-generation rates and different values of the link density p for a 25 node network.

The values of T.P. and TE.P. are assumed to be constant in our simulations as shown in Table 1. To study the effect of the ant generation rate on the AntNet algorithm, we assume a fixed ant generation rate during the TE.P., varying the ant generation rates only during the T.P. We consider three different cases of the ant generation during the T.P., namely 40, 4, 0.4 ms[9]. The corresponding results for the pdf of the hopcount and the weight for $N = 25$ and different values of the link density p are shown in Figures 5.6-5.8.

Figures 5.6, 5.7 and 5.8 show that the AntNet algorithm gives a near optimal solution for a 25 node network at low values of the link density p ($p = 0.2$ and $p = 0.1$). The performance of AntNet algorithm decreases as the value of the link density p in the random graph $G_p(N)$ is increased. Additional simulations for $N = 50$ show similar trends as the link density p is varied. Moreover, the comparison of AntNet for $N = 25$ and $N = 50$ shows that the performance of AntNet degrades as the network size N is increased. Figure 5.7 shows the validity of our comparison between AntNet and the Dijkstra's algorithm since the results for AntNet are similar to the results of AntNet under static implementation (Figure 5.5).

Comparison of Figure 5.6 with Figures 5.7 and 5.8 show that the AntNet algorithm converges to a good solution even at low ant generation rates. In Figure 5.6, the ant generation interval is 40 ms during both the T.P. and the TE.P. Thus, only 25 forward

[8] In dynamic implementation of AntNet, there is negligible difference between the performance of AntNet for $\varepsilon = 0$ and $\varepsilon = 0.1$.

[9] $\eta = 0.1, W_{max} = 50$ for all the simulations in this sub-section.

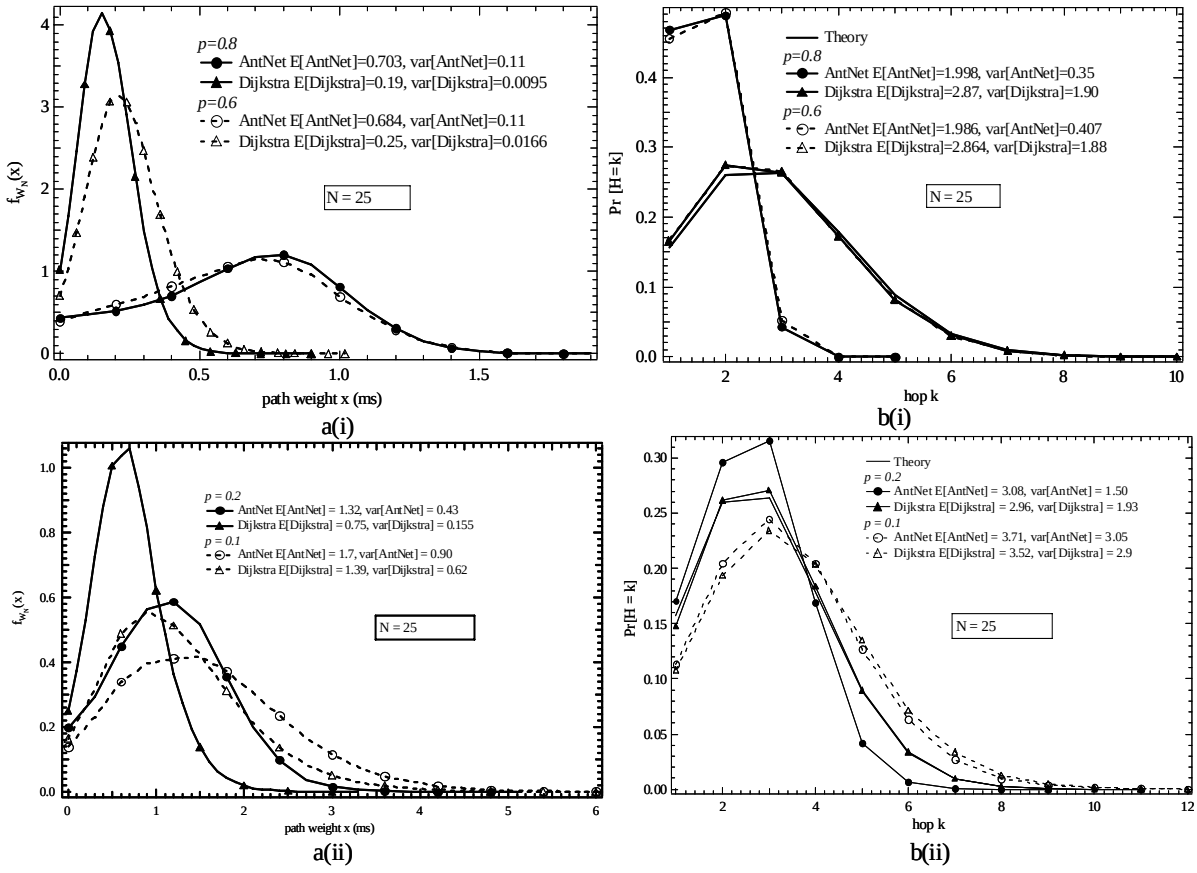


Figure 5.6: (a) The pdf of the weight of the Dijkstra's shortest path and the AntNet algorithm paths (i) $p = 0.8$ and $p = 0.6$ (ii) $p = 0.2$ and $p = 0.1$ (b) The pdf of the hopcount of the Dijsktra's shortest path, AntNet algorithm paths and obtained by theory (i) $p = 0.8$ and $p = 0.6$ (ii) $p = 0.2$ and $p = 0.1$. The ant generation interval during T.P. is 40 ms. (Each simulation consists of 10^5 iterations.)

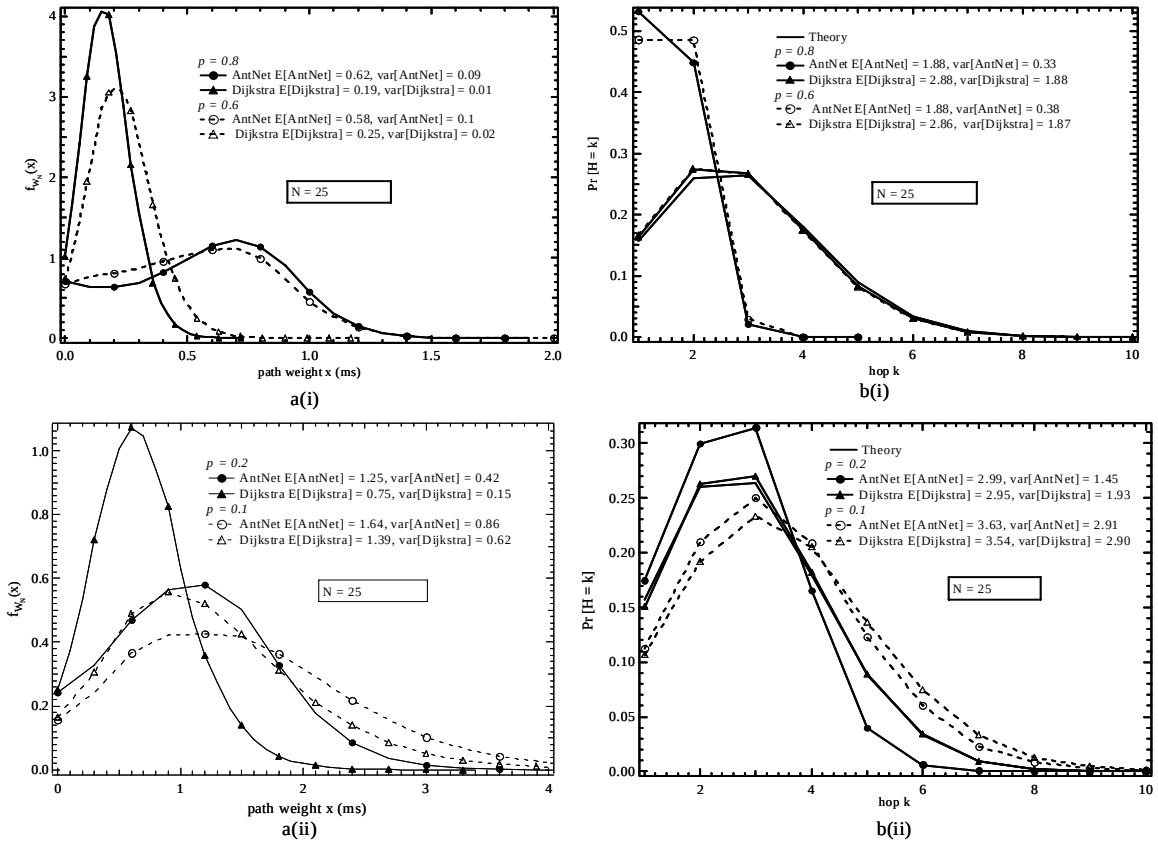


Figure 5.7: (a) The pdf of the weight of the Dijkstra's shortest path and the AntNet algorithm paths (i) $p = 0.8$ and $p = 0.6$ (ii) $p = 0.2$ and $p = 0.1$ (b) The pdf of the hopcount of the Dijsktra's shortest path, AntNet algorithm paths and obtained by theory (i) $p = 0.8$ and $p = 0.6$ (ii) $p = 0.2$ and $p = 0.1$. Same scenario as in Fig. 5 but with an ant generation interval of 4 ms during the T.P.

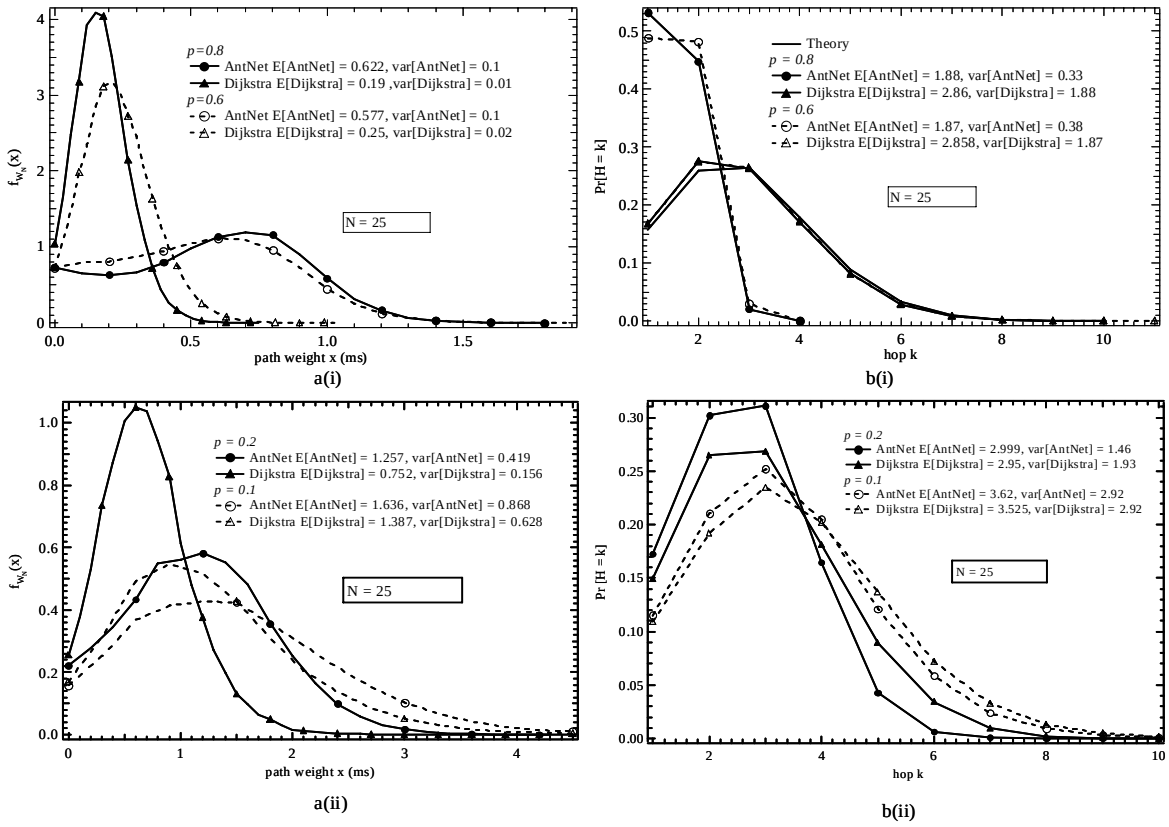


Figure 5.8: (a) The pdf of the weight of the Dijkstra's shortest path and the AntNet algorithm paths (i) $p = 0.8$ and $p = 0.6$ (ii) $p = 0.2$ and $p = 0.1$ (b) The pdf of the hopcount of the Dijsktra's shortest path, AntNet algorithm paths and obtained by theory (i) $p = 0.8$ and $p = 0.6$ (ii) $p = 0.2$ and $p = 0.1$. Same scenario as in Fig. 5 but with an ant generation interval of 0.40 ms during the T.P.

Table 5.2: The mean hopcount and weight for the unfNet algorithm for $N = 25$. ($p = 0.2$ and 0.8. $\eta = 0.1$ and $W_{max} = 50$)

Link density p	Ant generation interval during T.P.	E[Path weight]	E[Hopcount]
0.8	40 ms	0.75 ms	1.82
0.8	4 ms	0.63 ms	1.62
0.8	0.4 ms	0.59 ms	1.55
0.2	40 ms	1.38 ms	2.96
0.2	4 ms	1.2 ms	2.67
0.2	0.4 ms	1.16 ms	2.61

ants are generated by each node during the T.P. to search for the shortest paths to all other nodes in the network. The number of forward ants searching for the shortest path increases as the ant generation interval is decreased from 40 ms to 4 ms during the T.P.. This leads to an improvement in the performance of AntNet. As the ant generation interval is decreased from 4 ms to 0.4 ms, the performance of AntNet remains the same. This can be attributed to the fact that the variations in the ant generation rate are related to the size of the moving observation window $W_{\max}$ and the parameter η. The size of the moving observation window $W_{\max}$ in the above simulations is large enough to distinguish between ant generation intervals of 40 ms and 4 ms during T.P. But since the size of the moving observation window is small, it can store only the times of the last 50 forward ants generated from a given source to a given destination. This indicates that even if the ant generation rate is increased with a small window size, the performance of AntNet remains the same or may even go down. Additional simulations show that the AntNet algorithm performs better at $\eta = 0.02$ and $W_{\max} = 200$ than at $\eta = 0.1$ and $W_{\max} = 50$ for different ant generation rates and different values of the link density p.

We also performed simulations for the unfNet algorithm under identical conditions, as above. Table 5.2 lists the simulation results for the unfNet algorithm for $N = 25$ and different values of the link density p ($p = 0.2$ and 0.8). The results show that unfNet performs worse than AntNet for low ant generation rates (or large values of the ant generation interval). In unfNet, all the paths are continuously searched independently of the routing tables. While in AntNet, the probability that future forward ants choose paths with large delays is decreased by each forward ant. Therefore, in AntNet, mainly low delay paths contribute to routing table updates and the AntNet algorithm performs well even with a low ant generation rate[10]. When the ant generation rate is high, the

[10] All paths searched by the forward ants lead to positive reinforcement.

performance of unfNet is comparable to the AntNet algorithm. This can be attributed to the fact that the data traffic is small and the variations in delays along different paths is negligible.

Lattice Topologies

In this section, we investigate the performance of AntNet for Lattice topologies. We compare the performance of AntNet with Dijkstra's shortest path algorithm for the lattice topology shown in Figure 5.9. The source and destination nodes D1 and D2 are assumed to be fixed. The ant generation interval is 4 ms during T.P. and 40 ms during TE.P. ($\eta = 0.1, W_{\max} = 50$. The value of parameter a is chosen as 5.) Figure 5.10 shows the simulation results comparing the pdf of the hopcount and the weight for the shortest path and the AntNet algorithm paths. Figure 5.10 shows that the AntNet algorithm performs well for Lattice topologies. Also, the performance of AntNet degrades as the number of nodes in the network is increased.

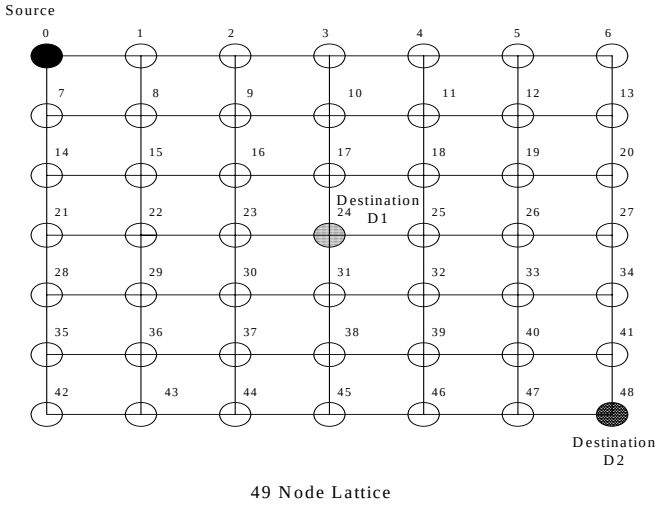
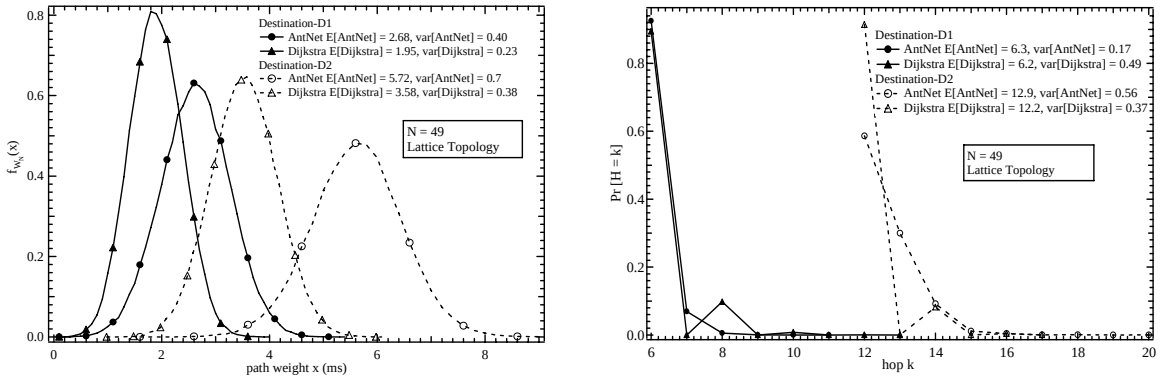


Figure 5.9: The 49-node lattice topology used for our simulations. The source node and the destination nodes D1 and D2 are assumed to be fixed as indicated

The effect of the moving observation window size $W_{\max}$ and the parameter η

In AntNet, the moving observation window is used to store the cost of the shortest path from a given source node to a given destination node. Furthermore, the moving observation window is also used to improve the accuracy of the mean μ_d as shown in (5.10). The optimal size of the moving observation window is hard to determine



(i) (ii)

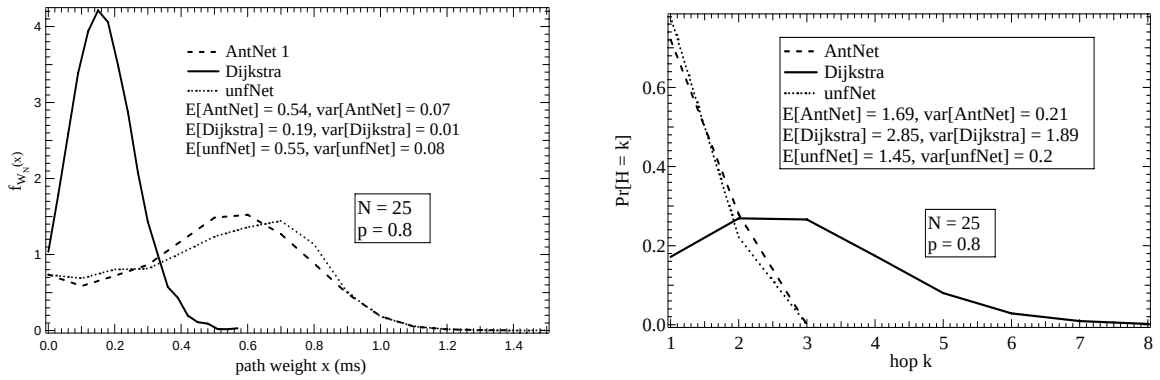
Figure 5.10: The pdf of the (i) weight and the (ii) hopcount of the Dijkstra's shortest path and the AntNet algorithm paths for a 49 node lattice topology. The source node and destination nodes D1 and D2 are fixed as shown in Figure 5.9. Each simulation consists of 10^5 iterations.

because it is linked to the ant generation rate and the parameter η. For determining the optimal value of $W_{\max}$ it is not sufficient to calculate the number of forward/backward ants alone. This is due to two reasons. First, each forward ant does not search for a distinct path. Second, the backward ants update the routing tables at each of the nodes along the path for the given destination as they travel from the destination node to the source node. Under ideal conditions, if each ant searches for one distinct path, the size of the moving observation window should be equal to or greater than the number of paths between any pair of nodes in the network. Thus, at any instant of time the moving observation window would contain the shortest path. Indeed, the size of the moving observation window can be effectively chosen in small networks where the number of paths between a pair of nodes in the network is small. Our simulations show that increasing the value of $W_{\max}$ without changing other parameters such as the ant generation rate and η leads to a small improvement in the performance of AntNet.

The parameter η is used to estimate the mean μ_d and the variance σ_d^2 by using the exponential model as shown in (5.1) and (5.2). The parameter η represents how many of the previous forward ant's trip times affect the mean or average value. In the exponential model, the weight of the current forward ant's trip time to a destination t_z after m forward ants for the particular destination have been received is $\eta(1-\eta)^{m-z}$. The smaller values of η indicate that the mean value is calculated over a large number of forward ant trip time samples. Our simulations show that changing η from 0.1 to 0.02 does not effect the performance of AntNet significantly. This indicates that at light traffic loads, the paths between pair of nodes in the AntNet algorithm quickly

converge to the mean value for that path making the second term in (5.1) redundant. The variations in parameter η might effect the performance of AntNet, if the traffic conditions or topology of the network varies. Under such conditions, the path between a pair of nodes will deviate significantly from the mean value. Thus, for a static topology with low traffic loads, the AntNet algorithm is very robust to changes in the parameter η.

Finally, to study the combined effect of the parameters η, $W_{\max}$ and the ant generation rates on the performance of AntNet, we set the ant generation to a very high value i.e., the ant generation interval is 0.4 ms during the T.P. and 4 ms during the TE.P. We also choose a very large size of the moving observation window i.e., $W_{\max} = 16000$. The value of $W_{\max}$ is sufficient to store the trip times of all forward ants generated by the given node as well as the trip times of forward ants received due to the sub-update method. Furthermore, the value of parameter η is chosen as very small i.e., $\eta = 0.001$ so that large number of sample trip times of forward ants are used to calculate the mean and variance in (5.1) and (5.2). Figure 5.11 presents the simulation results comparing the pdf of the hopcount and the weight for the shortest path, and the unfNet and AntNet algorithm paths for different values of $W_{\max}$ for $N = 25$ and $p = 0.8$.



(a) (b)

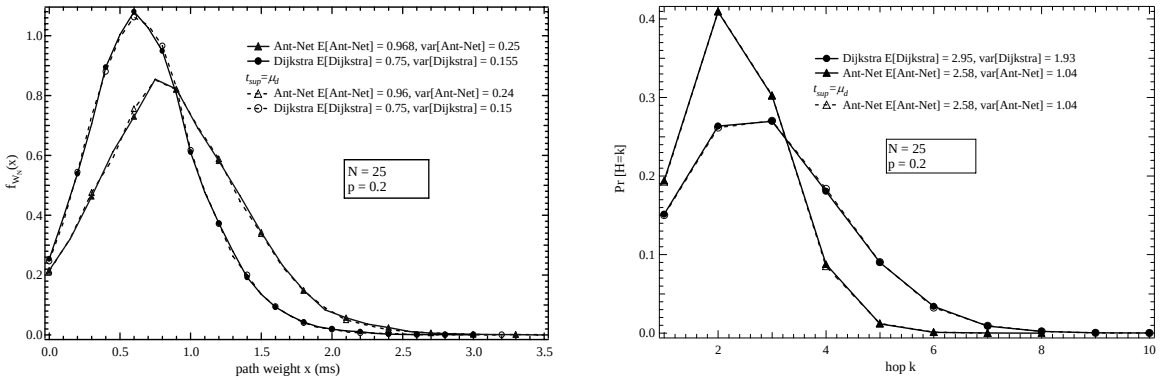
Figure 5.11: The pdf of the (a) weight and the (b) hopcount of the Dijkstra's shortest path, unfNet algorithm paths and the AntNet algorithm paths for $N = 25$ and $p = 0.8$. ($\eta = 0.001$, $W_{max} = 16000$). Each simulation consists of 10^4 iterations.

Figure 5.11 shows that the performance of AntNet can be improved by varying the parameters η, $W_{\max}$ and the ant generation rates in conjunction with each other. Thus, the inherent coupling between ant generation rate and the parameters η and $W_{\max}$ contributes to the complexity and robustness of the AntNet algorithm. Figure 5.11 also shows that the performance of unfNet is worse than AntNet, even though the data traffic is small. In unfNet, the forward ants move independently of the routing tables

and all the paths are searched with equal probabilities. In AntNet, if the path is found to incur a large delay, the probability of choosing the particular path by the future forward ants becomes less. But since every path found receives a positive reinforcement (in both unfNet and AntNet), the number of non-optimal paths updating the routing tables in unfNet is more.

The effect of the confidence interval, squash function $s(x)$ and the parameter β

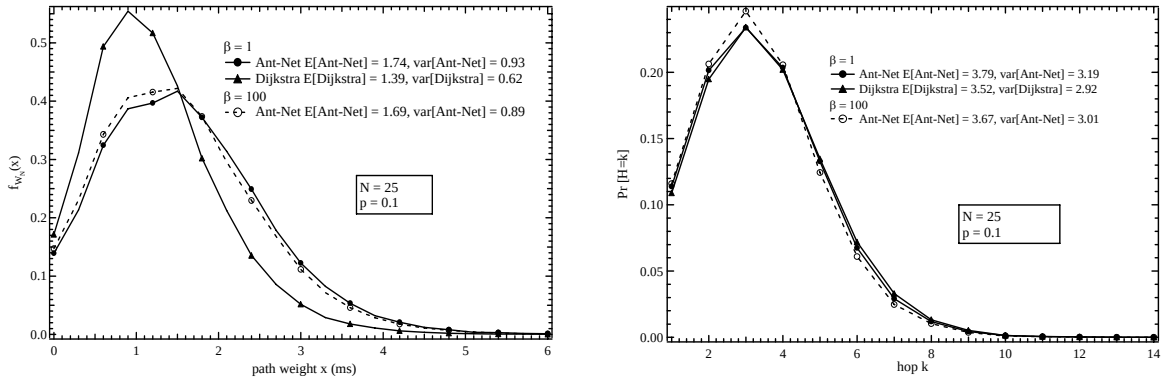
We first study the effect of the confidence interval on the performance of AntNet. The second term in (5.10) i.e., $\frac{\sigma_d}{\sqrt{1-\gamma}\sqrt{|W_{\max}|}}$ is used to improve the accuracy in estimating the mean μ_d but introduces additional complexity in the AntNet algorithm. Figure 5.12 shows the simulation results comparing the pdf of the hopcount and the weight for Dijkstra's shortest path and the AntNet algorithm paths, when the term $\frac{\sigma_d}{\sqrt{1-\gamma}\sqrt{|W_{\max}|}}$ is not used in the estimation of t_{sup}. This case is shown as $t_{sup} = \mu_d$ in Figure 5.12. The simulations are performed for $N = 25$ and $p = 0.2$. Also, the value of parameter a is chosen as 20 such that the average value of the coefficient $\frac{a}{N_k}$ is 4. (The ant generation interval is 40 ms during T.P. and TE.P. The value of parameters $\eta = 0.1$ and $W_{\max} = 50$.)



(a) (b)

Figure 5.12: The pdf of the (a) weight and the (b) hopcount of the Dijkstra's shortest path and the AntNet algorithm paths for $N = 25$ and $p = 0.2$ for $t_{sup} = \mu_d$ and t_{sup} calculated using (11) ($\eta = 0.1$, $W_{max} = 50$, $a = 20$). The ant generation interval is 40 ms during the T.P. and TE.P. Each simulation consists of 10^5 iterations.

Comparison of Figure 5.12 and Figure 5.6 shows that choosing a large value of parameter a improves the performance of AntNet. Furthermore, under the simulation



(a) (b)

Figure 5.13: The pdf of the (a) weight and the (b) hopcount of the Dijkstra's shortest path and the AntNet algorithm paths for $\beta = 1$ and $\beta = 100$ for $N = 25$ and $p = 0.1$. The ant generation interval is 40 ms during the T.P. and TE.P. Each simulation consists of 10^5 iterations. ($\eta = 0.1$ and $W_{max} = 50$.)

parameters considered, the removal of the term $\frac{\sigma_d}{\sqrt{1-\gamma}\sqrt{|W_{\max}|}}$ does not change the performance of AntNet. Thus, equation (5.10) could be simplified to reduce the complexity of AntNet.

The parameter β determines whether single or multi-path routing is followed by the data packets. The value of parameter β needs to be greater than 1 to prevent the data packets from choosing links with very low probabilities. A large value of β ($\beta >> 1$) indicates that the data packets follow only single-path routing. On the other hand $\beta = 1$ indicates that the data packets follow the routing tables and may even choose links with very low probabilities. Thus, $\beta = 1$ corresponds to multi-path routing. We compare the performance of AntNet algorithm for $\beta = 1$ and $\beta = 100$. When $\beta = 100$, the data packets are effectively following a single-path routing. Figure 5.13 shows the simulation results comparing the pdf of the hopcount and the weight for the shortest path and the AntNet algorithm paths for $p = 0.1$ for $\beta = 1$ and $\beta = 100$ for a 25 node network[11].

Figure 5.13 shows that the performance of AntNet improves as β is increased for $p = 0.1$. This shows that there is a greater probability that the shortest path has been correctly identified in the AntNet algorithm at lower values of p. Thus, making $\beta = 1$ prevents the data packets from choosing the path with the highest probability and leads to data packets following sub-optimal paths. On the other hand, $\beta = 100$ improves the performance of AntNet since the data packets follow only a single-path routing which

[11] The ant generation interval is 40 ms during the T.P. and the TE.P. The value of parameter $\eta = 0.1$ and $W_{max} = 50$.

has a greater probability of being the shortest path also.

5.3.4 Traffic Measurements

In this section, we compare the end-to-end delays for AntNet and Dijkstra's shortest path algorithm. A set of randomly chosen nodes are congested during each iteration and have an additional queueing delay of 10 ms for every packet. To make a fair comparison between the algorithms, we assume that the size of all data packets is same i.e., 4096 bits and the *TTL* for data packets is set to infinity. In addition to the normal data, we send a small number of data packets along the shortest path from the source to the destination node[12] (source routing or src_rt). Figure 5.14 shows the simulation results comparing the pdf of the hopcount and the end-to-end delay (weight) for the Dijkstra's shortest path and the AntNet algorithm paths for $N = 50$ and $p = 0.1$ (The ant generation interval is 4 ms during T.P. and 40 ms during TE.P. The value of parameter $\eta = 0.1$ and $W_{\max} = 50$.)

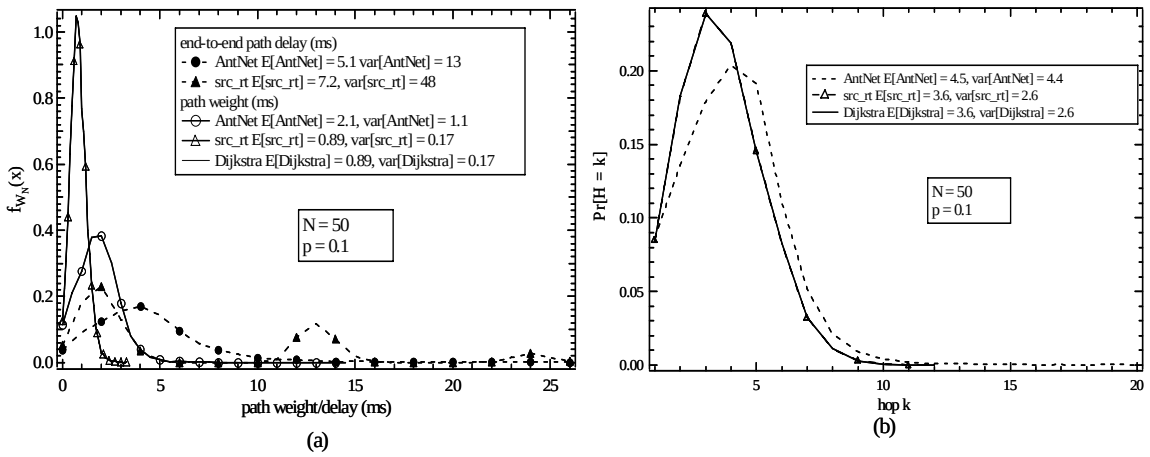


Figure 5.14: The pdf of the (a) end-to-delay/weight and the (b) hopcount of the Dijkstra's shortest path, AntNet algorithm paths and the source routing path for $N = 50$ and $p = 0.1$. The ant generation interval is 4 ms during the T.P. and 40 ms during the TE.P. ($\eta = 0.1$, $W_{max} = 50$.) Each simulation consists of 10^4 iterations.

Figure 5.14 shows that the AntNet algorithm performs better than single shortest path routing in terms of end-to-end delay. In src_rt, since Dijkstra's algorithm is used to compute the shortest path from the source to the destination, the queueing delays

[12]Node 1 is the source node and node 25 is the destination node. In AntNet, the data packets are generated at a uniform interval of 5 ms, while in src_rt, the data packets are generated at a uniform interval of 100 ms.

are not considered. During the iterations when there are no congested nodes along the shortest path, the end-to-end delays for the data packets using src_rt and AntNet are comparable. However, when there are one or more congested nodes along the shortest path, the data packets using src_rt incur queueing delays along the congested nodes. On the other hand, the AntNet algorithm performs load balancing and reduces the probability of data packets choosing the paths with congested nodes.

Table 5.3 lists the simulation results for the AntNet, unfNet and the src_rt algorithms for $N = 50$ and $N = 25$ ($p = 0.2, 0.1$). We consider different number of congested nodes and ant generation rates. Table 5.3 shows that unfNet generally performs worse than AntNet under varying traffic loads. The performance of unfNet becomes worse as compared to the AntNet algorithm, when the ant generation rate is low and the size of the network is increased. In AntNet, the network exploration is restricted to paths that incur low delays and only these paths are used to update the routing tables. In unfNet, the forward ants move independently of the routing tables. Thus, even the large delay paths are used to update the routing tables leading to a poor performance of the unfNet algorithm. This shows that the unfNet algorithm reduces the complexity but, in general, leads to a decrease in the performance.

Table 5.3: The expected hopcount/weight for the unfNet, AntNet, and the src_rt algorithms for $N = 25, 50$ ($p = 0.2, 0.1$) with different number of congested nodes and different ant generation rates ($\eta = 0.1$ and $W_{max} = 50$).

Routing Protocol	Network size (N)	Link density p	Ant generation T.P.	interval TE.P.	Congested Nodes	E[path delay]	E[Hopcount]
unfNet	50	0.1	4 ms	40 ms	8	5.6 ms	3.8
AntNet	50	0.1	40 ms	40 ms	8	5.8 ms	4.7
unfNet	50	0.1	40 ms	40 ms	8	12.7 ms	6.2
AntNet	25	0.2	4 ms	40 ms	4	3.4 ms	3.2
src_rt	25	0.2	4 ms	40 ms	4	5.8 ms	2.9
unfNet	25	0.2	4 ms	40 ms	4	3.5 ms	2.8
AntNet	25	0.2	40 ms	40 ms	4	3.7 ms	3.4
unfNet	25	0.2	40 ms	40 ms	4	5.5 ms	3.3
AntNet	25	0.2	4 ms	4 ms	4	3.36 ms	3.3
unfNet	25	0.2	4 ms	4 ms	4	3.1 ms	2.8
AntNet	50	0.2	4 ms	40 ms	4	3.1 ms	3.3
src_rt	50	0.2	4 ms	40 ms	4	4.3 ms	3.5

To further study the load balancing capabilities of AntNet, we assume that during

each iteration, a set of five randomly chosen nodes are congested after 5000 ms of the S.P. For each iteration, we compute the packet delay and the hopcount averaged over 250 ms moving windows. Figure 5.15 shows the simulation results for the packet delay and hopcount for $N = 50$ and $p = 0.1$ averaged over 10^4 iterations (The ant generation interval is 4 ms during T.P. and TE.P. The value of parameters $\eta = 0.02$ and $W_{\max} = 200$). Figure 5.15 shows how AntNet adapts to the introduction of congested nodes. Before 5000 ms there is small data traffic, and the average end-to-end delay and hopcount for the AntNet algorithm are constant. After the nodes become congested, there is a sudden increase in the end-to-end delay. However, the AntNet algorithm adjusts to the changing traffic leading to a decrease in the end-to-to-end delay. On the other hand in src_rt, a single shortest path is used for routing throughout the S.P. As a result when the nodes become congested, the average end-to-end delay increases.

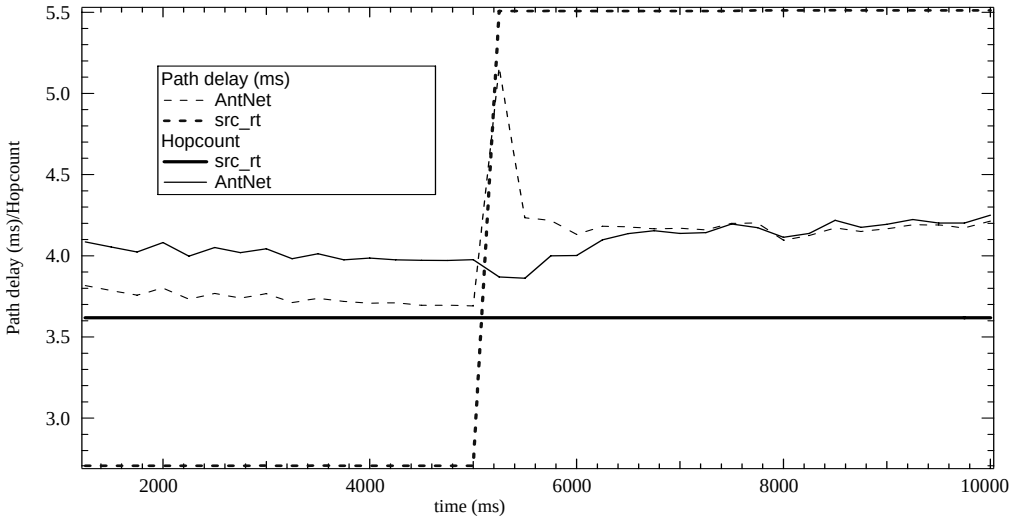


Figure 5.15: Transient analysis of AntNet for $N = 50$ and $p = 0.1$. The path weight and delay are averaged over 250 ms moving windows and 10^4 iterations. The ant generation interval is 4 ms during the T.P. and TE.P. ($\eta = 0.02$, $W_{max} = 200$.).

5.4 Conclusions

The AntNet algorithm performs well for random graphs and lattice topologies. The AntNet algorithm gives a near optimal solution at $p = 0.2$ and $p = 0.1$ for a 25 node network. This can be attributed to the fact that the number of paths between any given pair of nodes in the network at $p = 0.2$ and $p = 0.1$ is small. Since the AntNet algorithm starts searching for the shortest path in a random fashion, the smaller the

number of paths between a pair of nodes the greater is the probability that the algorithm converges to a near optimal solution. The performance of AntNet algorithm degrades as the network size or the link density p is increased.

The AntNet algorithm is robust to changes in the training of the network and converges to a good solution even at low ant generation rates. Further increasing the ant generation rates leads to an improvement in the performance but this is related to choice of other parameters such as $W_{\max}$ and η. Indeed, the coupling of various parameters is the inherent cause of complexity in the AntNet algorithm and changing one parameter favorably may not lead to an improvement in performance until the other parameters are also changed. The performance of AntNet can be improved by using a large value of the parameter a in the squash function $s(x)$. In this case, only the near optimal paths update the routing tables. The complexity of AntNet can be reduced by a number of methods, such as choosing parameter $\beta = 1$ and simplifying the calculation of parameter ϖ, but this generally comes at the expense of performance.

Due to inherent load balancing, AntNet performs better than shortest path routing under varying traffic loads. For small networks, when the traffic loads are small and the ant generation is sufficiently high, a modified version of the AntNet algorithm (unfNet) can be used. This reduces the complexity of the AntNet algorithm. The robustness and near optimal performance of the AntNet algorithm makes it an attractive solution for routing in wired networks.

Chapter 6

Ant Routing in Mobile Ad hoc Networks

A variety of ANTRALs have been proposed for ad hoc and mesh networks. Among the traditional routing protocols for ad hoc wireless networks, on-demand protocols perform better than table-driven protocols [16].

The performance of ANTRAL for mobile ad hoc networks is an open issue. ANTRAL algorithms do not take into account the mobility of the nodes. Moreover, load balancing for frequently changing topologies is a challenging issue. Load balancing involves distributing traffic along multiple paths depending on the traffic conditions such that the congested nodes or the unavailable links are by-passed. With node mobility, the paths are not stable making load balancing difficult. Furthermore, characteristics of traffic in ad hoc networks is not known. In this chapter, our aim is to study whether ANTRAL algorithms can be applied for routing in mobile ad hoc networks.

ANTRAL algorithms for ad hoc networks, such as AntHocNet [47], Ad hoc Networking with Swarm Intelligence (ANSI) [86], Ant-Colony-Based Routing Algorithm (ARA) [54], Ant-AODV [74] and Termite [89], use on-demand mobile agents for discovering routes. The on-demand flooding of mobile agents is similar to flooding of route request (RREQ) packets in on-demand protocols AODV and DSR. In ANSI, ARA and Termite, the pheromone decays and as a result after a certain period the routing tables are empty. The decay of routing table values is similar to AODV where the paths are valid for a certain duration of time. Thus, the paths obtained by using AODV or ANSI, ARA and Termite are identical. In addition, if a link breaks in ANSI, AntHocNet, Ant-AODV and ARA routing protocols inject route error packets similar to on-demand protocols. Indeed, the performance of ANSI, ARA and Ant-AODV is similar to AODV and DSR [54, 74, 86]. In our view, these algorithms are a variation of the on-demand protocols and deviate from the original idea of ANTRAL algorithms in which routing tables are sufficient for routing data packets. We do not consider the performance of ANTRAL algorithms that use on-demand flooding of mobile agents.

6.1 W_AntNet algorithm

The W_AntNet algorithm is based on AntNet but uses neighbor discovery and added functionality to deal with node mobility [39].

- Neighbor Discovery Protocol

At regular intervals, *hello* messages are exchanged between neighboring nodes. When the neighbors are lost or new neighbors are added, the routing tables are updated and the probability values are re-normalized. The forward ant packets and data packets waiting in the buffer are rerouted if the next hop neighbor is lost. However, the backward ant packets are dropped if the next hop neighbor is lost. This ensures that only paths that are stable during a sufficiently long time interval, i.e. the time between the creation of forward ant and the receipt of the backward ant appear in the routing tables.

6.2 Performance Analysis of W_AntNet

The routing overhead for obtaining a path to any destination is $O(N)$ in AODV and DSR since both these protocols use controlled flooding or sequence numbers. To compute the complexity of W_AntNet, we use the fact that ad-hoc wireless networks can be modeled as a geometric random graph (or random graph) [58]. In W_AntNet, the forward ants perform random walks to search for the destination. Therefore, the worst-case routing overhead for a single update of the routing tables [5] is $O(N \log N)$. Moreover, the number of updates required for routing table convergence, i.e. probability for one of the entries in routing tables to be one, depends on the quality of different paths, network topology and the routing table update function. We show that multiple backward ants are needed for the routing tables to converge.

Consider a node k with N_k neighboring nodes. At $t = 0$, the probability of choosing neighbor n as the next hop for destination d, $p_{nd}(0) = \frac{1}{N_k}$. We assume that only the routing table values for neighbor n i.e., p_{nd} receive positive reinforcement. Under these assumptions, the probability $p_{nd}(t+1)$ is,

$$p_{nd}(t+1) = p_{nd}(t) + cp_{nd}(t) \tag{6.1}$$

where c is a constant. After $\mathcal{Y}$ updates,

$$p_{nd}(\mathcal{Y}) = p_{nd}(0)(1+c)^{\mathcal{Y}}$$

Since the routing table converges when $p_{nd}(\mathcal{Y}) \longrightarrow 1$, the value of $\mathcal{Y}$ is :

$$\mathcal{Y} = \frac{\log(N_k)}{\log(1+c)} \tag{6.2}$$

where N_k is the number of neighbors of node k. Thus, the overhead for routing tables to converge in W_AntNet under the given assumptions is $O(N \log N_k \log N)$. The value of N_k varies from N for a complete mesh to $p\,(N-1)$ for a random graph or geometric random graph [58]. The geometric random graph is almost surely connected [58] if the link density $p > \frac{\log N}{N}$ when N is large and therefore, the minimum value of $N_k = \log N$.

We performed extensive simulations of W_AntNet using our simulator written in C and NS-2 simulator[1]. NS-2 is a discrete event simulator written in C++ and uses OTcl as a command and configuration interface. Using NS-2, we compare the performance of W_AntNet with AODV and DSR.

The simulations are performed for the *benchmark scenario* [16]: 50 nodes moving over an area of $1500m \times 300m$ for 900 s. In W_AntNet, there is training period (T.P.) at beginning of simulations during which data packets are not generated. The values of various parameters in W_AntNet are taken from [42, 37]. The mobility model in the simulations is random waypoint [16]. A pause time of 0s corresponds to continuous motion while a pause time of 900s corresponds to the static scenario. We assume a transmission range of 250m for each node in our simulator. Using these parameters, the average node degree is $\frac{N}{A} * \pi r^2 = \frac{50*\pi*(250)^2}{1500*300} \approx 22$ and the network diameter or worst case hopcount is $\frac{\sqrt{1500^2+300^2}}{250} \approx 6$.

We first compare the performance of W_AntNet for static and dynamic topologies using our simulator. The MAC layer is ignored in these simulations. Moreover, we assume that each node is able to remove 1 packet/ms from the queue and no TTL is set for data packets. The routing tables are updated based on the hopcount between the source and destination nodes so that the performance is independent of the packet size, capacity of the links and the queueing delays. Thus, instead of (5.5), $p'_{nd} = p_{nd} + \varepsilon$ is used for choosing the next hop. Figure 6.1 shows the probability distribution of hopcount for W_AntNet with different pause times. To verify our analysis, we also plot the number of connectivity changes in our simulation model and the model used by Broch *et al.* [16] (insert in Figure 6.1).

The legend in Figure 6.1 shows expected hopcount for W_AntNet for different pause times. Figure 6.1 shows that W_AntNet performs similar to Dijkstra's algorithm in a static topology. However, when mobility is introduced (pause time is less than 900s) the expected hopcount increases which points to a significant amount of packets in loops. Figure 6.2 shows the percentage of data packets with loops as a function of the pause time. Figure 6.2 also shows results for W_AntNet with look-ahead[2]. The increase of data packets in loops with increased mobility in W_AntNet is in sharp contrast to AODV and DSR that are shown to be loop-free [62, 81]. Sequence numbers in AODV excludes loops at all times while DSR uses source routing that is inherently loop-free.

[1] NS-2 Network Simulator, *http://www.isi.edu/nsnam/ns*, 2005.

[2] Look-ahead means that if the destination is among the neighbors, then the destination is chosen as the next hop

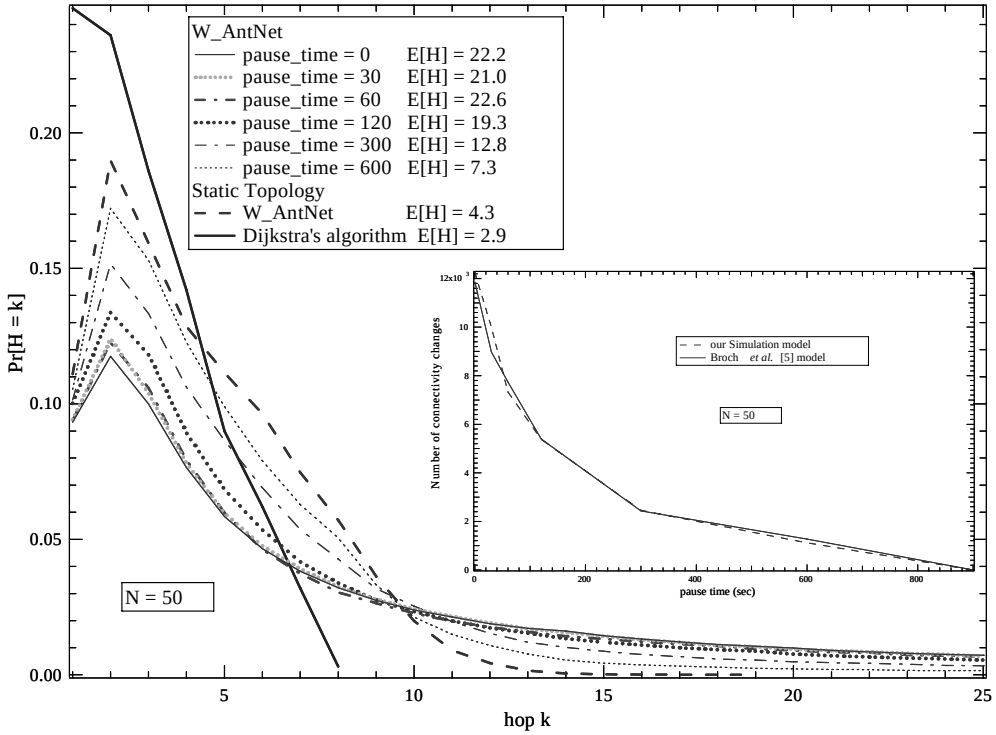


Figure 6.1: The probability distribution of hopcount for W_AntNet algorithm as a function of pause time. The legend shows the expected hopcount for different values of pause times.

Figure 6.2 also shows that look-ahead reduces the percentage of data packets in loops at increased mobility.

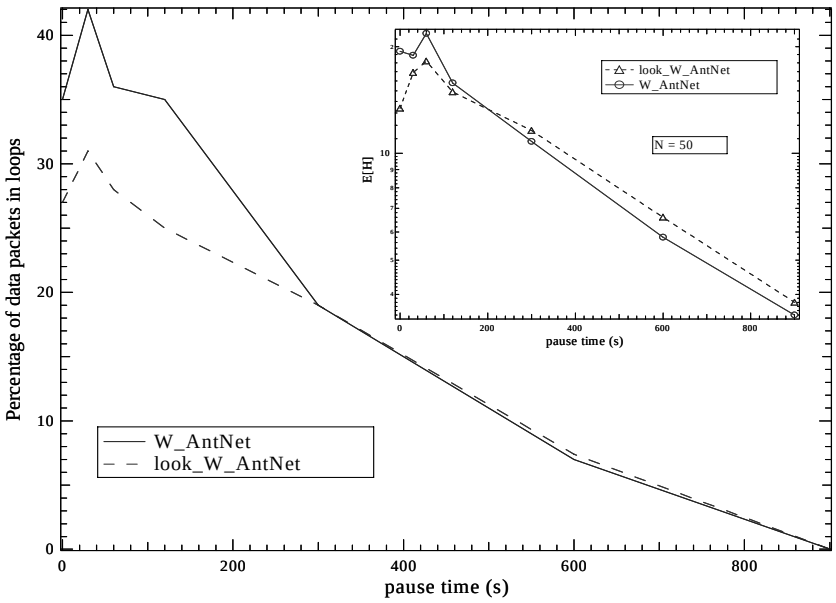


Figure 6.2: Percentage of data packets with a loop in their path as a function of pause time for W_AntNet (look_W_AntNet). The insert shows the expected hopcount in W_AntNet for different pause times.

6.2.1 NS-2 simulations

The default settings for all the experiments and the simulation code and parameters for AODV, DSR are taken from the CMU/Monarch extensions for NS-2 [16]. The MAC layer is 802.11 and the interface queue size is assumed to be 100 packets. The number of CBR sources is 10 and the data rate is 4 packets/sec. The capacity of links is 2 Mbps. In W_AntNet, we assume that each node can store 50 packets in the low and high priority queues.

Case Study 1: Static Network

We compare the packet delivery ratio (PDR) and the end-to-end delay for AODV, DSR and W_AntNet for a static scenario. The size of data packets is varied from 64 bytes to1024 bytes. We also show results for W_AntNet when the forward ants are generated only during the T.P. (W_AntNet_antsTP). This reduces the number of control packets

in the network. Since the topology is static and the amount of data traffic is small, the generation of forward ants only during T.P. is sufficient for routing under these conditions.

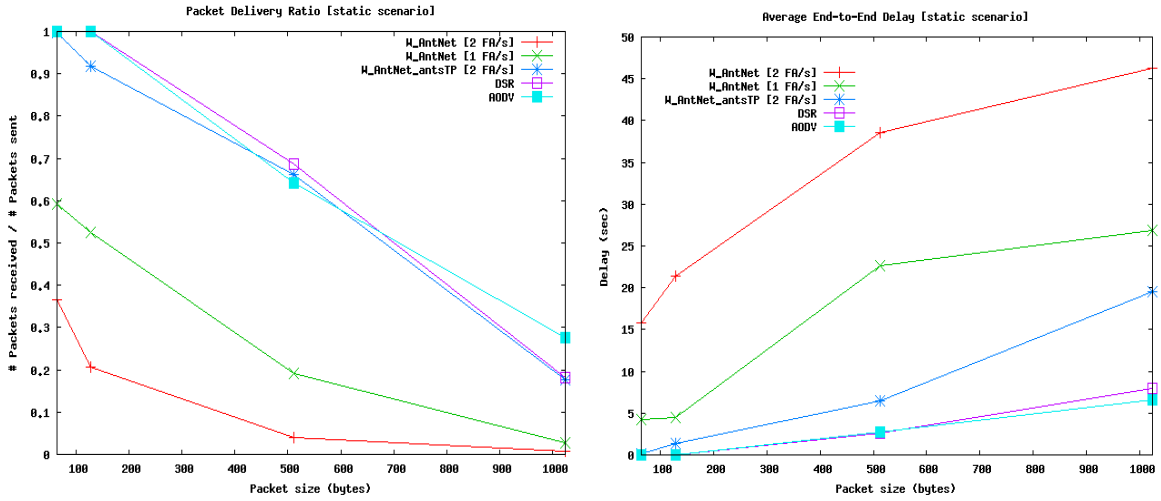


Figure 6.3: The PDR and end-to-end delay for W_AntNet, AODV and DSR for static topology.

The PDR and end-to-end delay in Figure 6.3 show that W_AntNet performs similar to AODV and DSR when forward ants are generated only during the T.P. However, as the forward ant generation rate is increased from 0 (W_AntNet_onlyTP) to 2 forward ants/s, the routing overhead causes congestion in the queues. Thus, W_AntNet causes a dual problem in ad hoc networks. A large number of forward ants cause congestion in the network. However, with increasing mobility, more forward ants need to be generated to account for frequent changes in the topology.

Case Study 2: Mobile Scenario

We study the performance of different routing protocols with mobility. The packet size is assumed to be 64 bytes. Figure 6.4 shows the PDR, end-to-end delay and the routing overhead (measured in number of bytes since the size of control packets varies in W_AntNet) for AODV, DSR, W_AntNet and W_AntNet with look-ahead (W_AntNet_look). We also reduce the routing overhead in W_AntNet by generating forward ants only from source-destination pairs that have data packets to send (W_AntNet_onlysrc). Thus, the remaining nodes in the network do not maintain up-to-date routing tables.

Figure 6.4 shows that with high node degree, AODV and DSR perform well in terms of PDR and end-to-end delay. Figure 6.4 shows that using optimizations such as look-

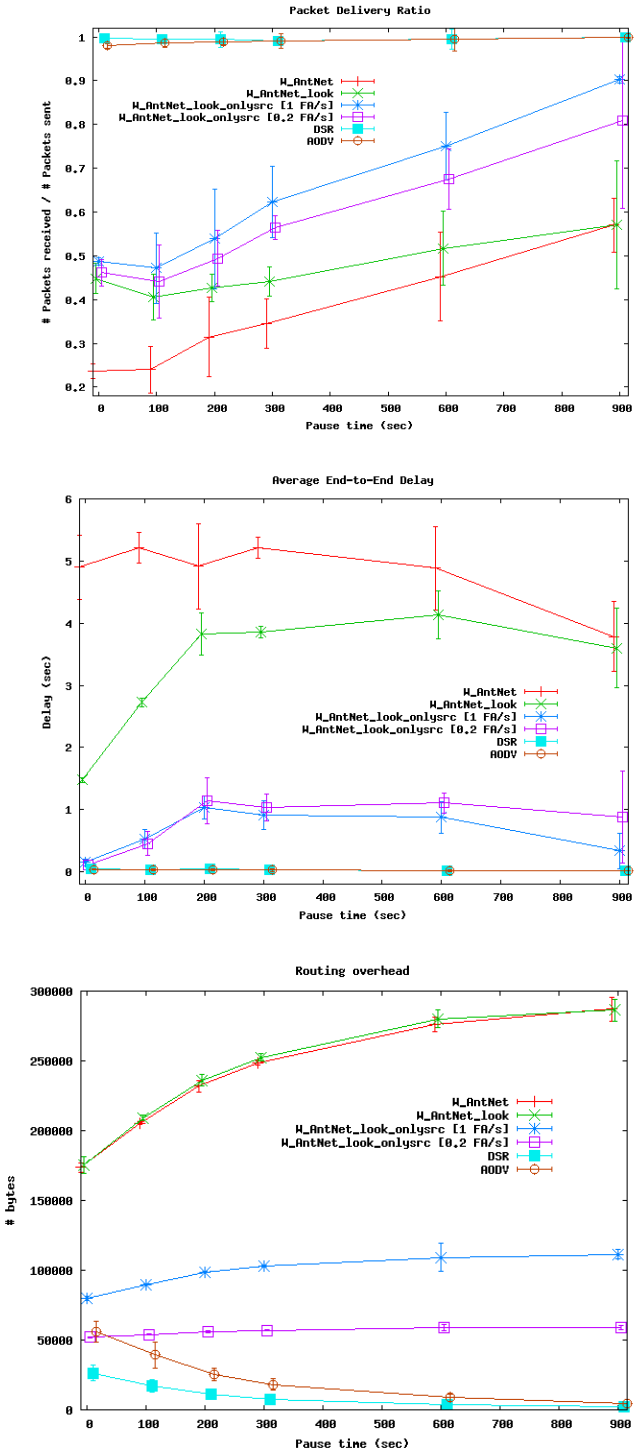


Figure 6.4: The PDR, end-to-end delay and routing overhead for W_AntNet, AODV and DSR for mobile network of 50 nodes moving in an area $1500 \times 300 m^2$.

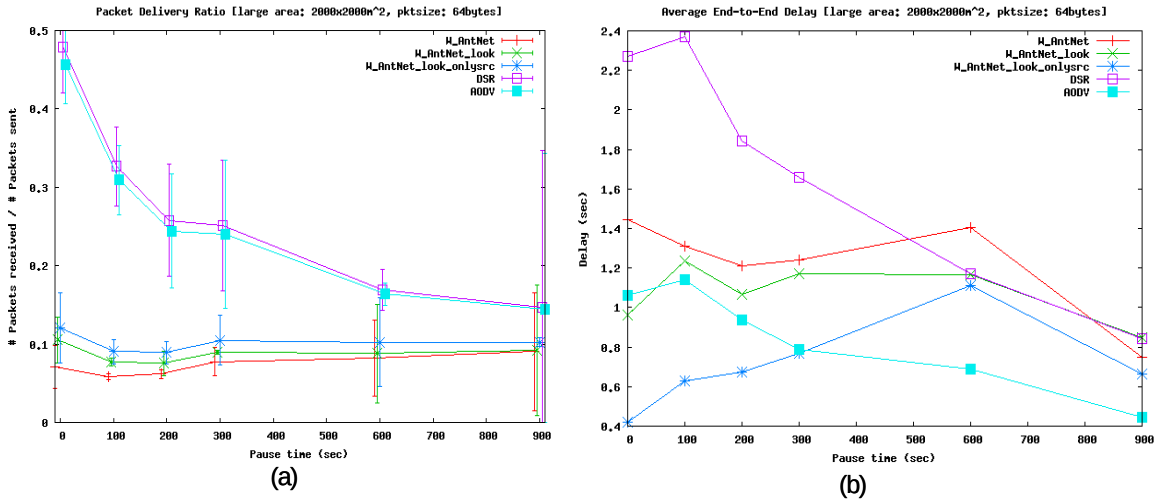


Figure 6.5: The (a) PDR and (b) end-to-end delay for W_AntNet, AODV and DSR for different values of the pause time. Nodes are moving in an area 2000m×2000m.

ahead and generating ants only from source-destination pairs reduces the overhead in W_AntNet. However, the routing overhead is still more than in AODV and DSR. As a result, the PDR for W_AntNet_look and W_AntNet_onlysrc is still lower than AODV and DSR.

Case Study 3: Large Area Network (Sparse Graph Topology)

In this case, we compare the performance of different protocols in a larger area with the same number of nodes ($N = 50$). The area over which the nodes move is $2000 \times 2000m^2$. Under these conditions, the average node degree is 2.5 and the worst case hopcount is given by 11. The number of CBR sources is also increased to 20. Figure 6.5 shows the PDR and end-to-end delay for AODV, DSR and W_AntNet. The packet size is assumed to be 64 bytes.

This scenario leads to a sparse graph topology and an increase in network diameter. As a result, the performance of all three protocols degrades considerably. The simulations show that all three protocols have scalability problems. To improve the scalability of routing protocols, additional schemes such as clustering need to be implemented [107]. Figure 6.5 also shows that the use of look-ahead does not improve the performance of W_AntNet since the average node degree is small.

6.3 Conclusions

The performance of W_AntNet is comparable to the shortest path algorithm for static topology but is dependent on the buffer size at the nodes. Since forward ants share the same queue as data packets in W_AntNet, a high ant generation rate leads to congestion in the network. This causes W_AntNet to perform poorly compared to AODV and DSR when the size of the buffer is small. In a dynamic topology, a significant amount of packets in W_AntNet have loops. This can be attributed to the non-convergence of routing tables in W_AntNet. Therefore, with mobility, the performance of W_AntNet deteriorates in comparison to AODV and DSR that are loop-free.

Part III

Searching

Chapter 7

Introduction

Searching for resources or services efficiently is an important issue in various networks. There are two common algorithms employed for searching - flooding and random walks (RW). In flooding, each node forwards the packet to all its neighbors. In RW, the packet is forwarded to a single node, chosen uniformly among the neighbors of a node. Variations of RW that could be employed for searching include RW strategy where the next hop is chosen as the node with maximum degree, RW with no retracing of steps etc. We use the term *RW using highest degree* even though the next node is chosen deterministically and not according to uniform distribution. The term local search algorithm or path finding strategies is also used for such algorithms [64, 100].

There are numerous applications of searching algorithms in different networks. OSPF uses flooding to distributed and synchronize the link-state routing tables between nodes. In wireless ad-hoc networks, reactive protocols such as AODV and DSR use flooding to locate the destination [34]. In web-graphs, search engines use breadth first search to perform a complete search of the web. However, to reduce the overhead of searching, agents or spiders based on RW, or variations of the RW such as the RW strategy where the next hop is chosen as the node with maximum degree, and the RW strategy where the probability of choosing the next node is proportional to the nodal degree are widely used [27].

In the context of routing, RWs can be considered an extreme case of routing algorithms that are used when the topology and the link weight structure of the graph are not known. Variations of RW can be considered routing algorithms that use partial topology information. Dijkstra's shortest path algorithm can be used when both the topology and the link weight structure of the entire graph are known. However, the dynamic nature of ad hoc and overlay networks does not always allow to collect information concerning the current topology of the network. An attempt to collect such information will often result in outdated information. Under the conditions that neither the graph topology nor the link weight structure is known, the only possible routing algorithms are flooding or RWs [38, 40].

Hot-potato routing, also called *deflection* routing, is a term first introduced by Baran [7]. In hot-potato routing, the nodes in the network have no buffers to store packets and the packets are forwarded to another node immediately [7, 17]. *Hot-potato* routing algorithms are well suited for optical networks since it is difficult to buffer optical messages. Hot potato routing algorithms have also been used in parallel machines such as the HEP multiprocessor, the Connection machine, and the Caltech Mosaic C, as well as in high speed communication networks [17]. The term *hot-potato* routing used in this context is exactly the same as a RW.

In mobile agent based routing, the mobile agents perform a RW or a variant of RW while searching for the destination. In Ant-Net, loop-erased RWs are used by the mobile agents. Mobile agents using RW have been proposed for providing membership services for ad-hoc networks by Dolev *et al.* [44]. As a sampling technique, RWs have been used for providing membership services in ad hoc networks [44, 8] that provide the nodes in the network with a view of the other nodes in network and that are used by various applications such as location services, peer sampling services and random overlay constructions [8]. In [8], Bar-Yossef *et al.* develop a membership service for ad hoc networks based on RW using highest degree. They show that the performance of such membership service is superior to other existing membership services based on gossiping or flooding [8].

Unstructured overlay networks such as Gia proposed by Chawathe *et al.* [20] and Gnutella build a random graph and use flooding or RWs to discover data stored at different nodes. RWs have been shown to induce lower overhead than constrained flooding used by the current versions of Gnutella [71, 51]. In the original Gia [20], the RWs were biased to prefer nodes with higher capacity but Castro *et al.* [19] have shown that preferring nodes with higher degree leads to a higher success rate and a lower delay. Thus, further improvements have been proposed to Gia in which RWs are biased towards the higher degree nodes [19]. Also, variations of RWs have been proposed in which there are no loops [19].

Multiple RWs have been proposed for searching on unstructured peer-to-peer networks by Lv *et al.* [71]. However, the optimization of multiple RWs has not been analyzed. Adaptive techniques based on RWs have been proposed for searching by Bisnik and Abouzeid [13]. In the searching technique proposed in [13], the number of RW queries used for searching are varied depending on the previous performance of searching. Our work differs from this approach since we study the optimization between the number of queries and the TTL of queries for different graph topologies.

The analysis of RWs has been an active topic of research [28, 63, 70]. Lovász [70] presents a detailed survey of RWs. An exact analysis of RWs on graph topologies such as lattice and torus has been studied in [15]. A detailed mathematical analysis of RWs is given in Chung [24]. Different search algorithms for scale-free and power law graphs have been analyzed in [2, 64, 100].

7.1 Overview

While most RW strategies have been studied in specific scenarios, a general analysis of performance of searching with RW strategies is missing. In this section, we analyze the performance of searching with different RW strategies on ER random graphs and power law graphs generated using preferential attachment. Both these graph topologies are of fundamental important since ad-hoc wireless networks can be modeled as ER random graphs while the web graphs and peer-to-peer networks can be modeled as power law random graphs [51, 58]. We study both a single query and multiple queries to search for the destination. We do not consider dynamic topologies in our analysis.

In [64] and [100], RW strategies where the next hop is chosen as the node with the highest degree and with no retracing of steps have been analyzed in terms of expected hopcount and weight. We show that strategies such as RW using highest degree and RW with no retracing of steps can lead to infinite loops. Therefore, the comparison of different RW strategies should also include an analysis of infinite loops, in addition, to the expected hopcount comparison.

In case of multiple RW queries, we study the optimization of number of queries and the TTL of queries for ER random graphs and preferential attachment power law graphs. We also show an efficient way of searching graphs using RWs with no repetition of steps (memory). Our analysis shows that searching with RW with memory performs better than RW. In addition, the optimized value of memory M depends on the topology of the network.

7.2 Definitions and Random Walk properties

A random walk is a finite Markov chain that is time-reversible. The access or hitting time $\mathcal{H}_{uv}$ is the expected number of hops before node v is visited, starting from node u. The sum $\mathcal{J}(u,v) = \mathcal{H}_{uv} + \mathcal{H}_{vu}$ is called the commute time. The cover time (starting from a given distribution) is the expected number of hops to reach every node in the graph. If no starting node starting distribution is specified we mean the worst case, i.e., the node from which the cover time is maximum.

A RW can be described as a finite Markov chain that is time-reversible [70]. The stochastic matrix $P = \Delta^{-1}A$, where $\Delta = diag(\deg_1, \deg_2, ..., \deg_N)$ is the degree matrix and A is the adjacency matrix, represents the transition matrix of the RW. It is known [70] that the RW has a unique stationary distribution κ, such that $\kappa P = \kappa$, with $\kappa_i = \frac{\deg_i}{2L}$. Let the RW start at node v_0. The node v_0 could be drawn from some initial distribution Y_0. Denote the sequence of random nodes by v_t $(t = 0, 1...)$. If we denote by Y_t the distribution of v_t i.e., $Y_t(i) = \Pr[v_t = i]$, the RW can be expressed as $Y_{t+1} = P^T Y_t$ and hence, $Y_t = \left(P^T\right)^t Y_0$. Thus, the probability that RW starting at u reaches node v in t steps is given by (u,v) entry of the matrix P^t [70].

It is known that a symmetric P matrix has n real eigenvectors with corresponding eigenvalues $1 = \mathfrak{y}_1 \geq \mathfrak{y}_2 \geq ... \geq \mathfrak{y}_n \geq -1$. Moreover, if we exclude bipartite graphs or reducible Markov chains, then $\mid \mathfrak{y}_i \mid < 1$, for $i > 1$. Cover time is the expected number of steps for RW to visit all nodes at least once. Let $\mathcal{C}_N$ be the cover time for the RW, then the cover time is [8],

$$\mathcal{C}_N = O\left(\frac{\kappa_{\min}^{-1} \log N}{1 - \mathfrak{y}_2}\right) = O\left(\frac{N \log N}{1 - \mathfrak{y}_2}\right), \text{ where } \kappa_{\min} = \frac{\deg_{\min}}{2L} \tag{7.1}$$

Jonasson [63] has shown that when $p \geq \frac{\log N}{N}$, w.h.p a random graph has the same cover time as the complete graph K_N i.e., the cover time is $N \log N$. Similarly, for power law graphs generated using BA model, it has been shown by Cooper and Frieze [28] that if $l \geq 2$, then w.h.p. the cover time $\mathcal{C}_N \leq \lceil \frac{2l}{l-1} N \log N \rceil + o\left(\lceil \frac{2l}{l-1} N \log N \rceil\right)$.

Chapter 8

Searching with single query

In this chapter, we investigate the performance of searching with a single query based on RW or variation of RW. We investigate the expected hopcount obtained by different RW strategies and the probability of finding the short paths. A general expression for the expected hopcount of a RW is computed. The performance of different RW strategies is compared on different graph topologies such as a complete graph, ER random graphs $G_p(N)$ and power law graphs generated using the BA model.

The rest of this chapter is organized as follows. Section 8.1 gives a description of different RW strategies. Sections 8.2 and 8.3 give an analysis and simulation results for searching with different RW strategies. The last section presents the conclusions.

8.1 Random Walks

The topology and link weight structure are essential for characterizing the network. Therefore, we study the RW strategies under two distinct regimes. In the first case, we assume that all the link weights are 1. Thus, the RW strategies use only topological information such as degree for choosing one of the neighbors as the next hop.

In the second case, we investigate different RW strategies on weighted random graphs. The weighted graphs offer a more realistic view of the network. For example, Internet traffic [100] and link capacities in overlay networks [19, 71] can be represented as weighted edges. In weighted graphs, RW strategies can use link weights, in addition to any topological information, for choosing the next hop.

8.1.1 Random Walk with memory M

A first-in first-out (FIFO) list, called the memory list $\mathcal{M}$, is maintained. The memory list $\mathcal{M}$ contains the node identifiers n_j of the last M nodes visited during the RW, i.e. $\mathcal{M} = \{n_1, n_2, .., n_M\}$, where $M =\mid \mathcal{M} \mid$ represents the number of elements in the

memory list $\mathcal{M}$. The next hop is chosen uniformly among the neighbors of the node that are not in the memory list $\mathcal{M}$. In our implementation of the RW strategy with memory M ($\mathrm{RW_M}$), the node identifier of the current node is not stored in the memory list $\mathcal{M}$ and no self-loops are allowed (The next hop cannot be chosen as the node itself.) The idea of using $\mathrm{RW_M}$ on a graph is similar to the RW with unbounded memory in a continuous 3-dimensional space which has been referred to as a self-avoiding RW [66, 93]. The $\mathrm{RW_M}$ strategy is equivalent to the search queries proposed for overlay networks where the structure is used to ensure that nodes are visited only once during a query and to control the number of nodes that are visited accurately [19].

From the routing point of view, the major shortcoming of RWs on graphs is the existence of loops in the path while travelling from the source to the destination node. To prevent loops, the simplest method is to introduce memory in the RWs. The one hop loops can be prevented by using $M = 1$, both the two hop and one hop loops can be prevented by using $M = 2$ and so on. Thus, a complete memory $M = N - 1$ totally eliminates loops in the RWs. But the introduction of memory ($M \geq 1$) in RWs leads to a *deadlock*. Figure 8.1 explains the concept of deadlock in a RW with memory. Let us assume that the RW has a memory $M = 3$ represented as $\mathcal{M} = \{n_1\ n_2\ n_3\}$ where n_3 represents the last node visited. Consider the situation where the RW enters the cluster **B** at node 1. Suppose that the RW leads to a path $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4$ in the cluster **B**. At node 2, the memory list $\mathcal{M}$ contains $\{1\}$ and at node 3 the memory list $\mathcal{M}$ contains $\{1, 2\}$. At node 4, the memory list $\mathcal{M}$ contains $\{1, 2, 3\}$ and since all the neighbors of node 4 are in the memory list $\mathcal{M}$ and no self loops are allowed, the RW cannot move forward nor backward. Therefore, a deadlock prevents the RW from ever reaching the destination. Keeping the memory M equal to the degree of nodes is not sufficient to prevent loops. Consider the same situation as above but with memory equal to the degree of node 4, i.e. $M = 2$. At node 4, the memory list $\mathcal{M}$ contains $\{2, 3\}$ and the next hop can be chosen as node 1. Thus, introducing memory removes the loops in the RW but induces deadlocks. In the implementation of Gia, Castro *et al.* [19] have used a query in RWs which consists of all the previously visited hops. This is similar to using complete memory in our analysis. The above analysis shows that there are two distinct regimes possible for RWs. Without memory, i.e. $M = 0$, the RWs can have loops but no deadlocks. For complete memory, the RWs can only have deadlocks and no loops. When the value of the memory M is such that $0 < M < N - 1$, the RWs can have loops and deadlocks.

8.1.2 Random Walk with look-ahead

The RW with look-ahead ($\mathrm{RW_{LA}}$) is a RW that uses information about the neighbors [52, 73]. In $\mathrm{RW_{LA\,=\,1}}$, the destination is chosen as the next node if it is among the neighbors of the node, else the next hop is chosen uniformly among all the neighbors of the node. The use of look-ahead 1 has been assumed in recent work [2, 64] in searching

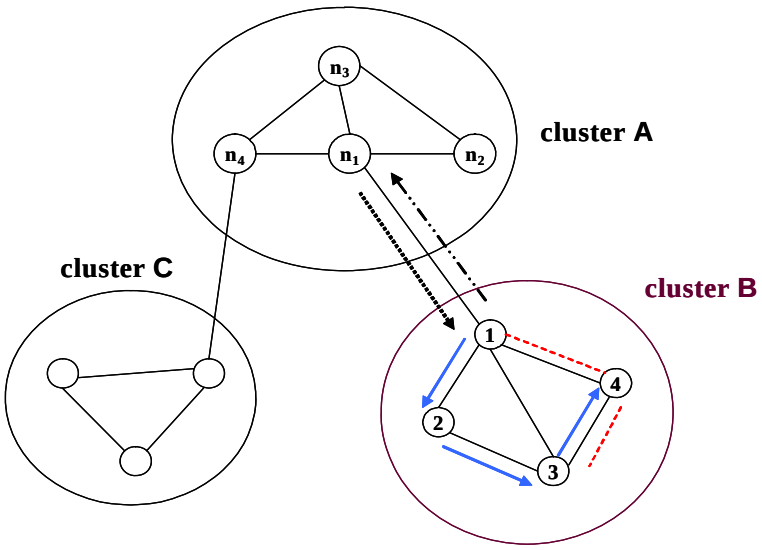


Figure 8.1: Explanation of deadlock in (a) RW with memory: The path $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ is shown by arrows. At node 4, the dashed lines indicate the links that cannot be used with memory $M \geq 3$. (b) RW using highest degree: Node n_1 chooses node 1 as next hop and node 1 chooses node n_1 as next hop leading to an infinite loop.

on graphs. In $\mathrm{RW}_{\mathrm{LA}=1}$, there are no deadlocks. The $\mathrm{RW}_{\mathrm{LA}=1}$ can also be used in combination with memory M (see Table 1).

8.1.3 Random Walk using highest degree

In RW using highest degree ($\mathrm{RW}_{\mathrm{HD}}$), the next hop is chosen as the node with highest degree until the destination is reached. Thus, at node u the next node v is chosen such that $\deg_v = \max_{s \epsilon N_u} \deg_s$, where N_u is the set of neighbors of u. The $\mathrm{RW}_{\mathrm{HD}}$ strategy can end in a deadlock. Deadlocks occurring in RW_{M} and $\mathrm{RW}_{\mathrm{HD}}$ have different characteristics. In deadlocks in RW_{M}, the node can terminate the RW because it cannot choose any of the neighbors as the next hop. In deadlocks in $\mathrm{RW}_{\mathrm{HD}}$, the node can always choose a neighboring node with highest degree as the next hop but this process may repeat. Figure 8.1 illustrates the concept of deadlock in $\mathrm{RW}_{\mathrm{HD}}$. At node n_1, node 1 is chosen as the next node since it has the highest degree among the neighbors of the node n_1. At node 1, node n_1 is chosen as the next hop since node n_1 has the highest degree among the neighbors of the node 1. Thus, the $\mathrm{RW}_{\mathrm{HD}}$ ends in an infinite loop or a deadlock. The $\mathrm{RW}_{\mathrm{HD}}$ can also be used in combination with different values of memory M and look-ahead (see Table 1).

8.1.4 Random Walk proportional to the degree

We consider a general case of the RW proportional to the degree ($\mathrm{RW_{PD}}$) with a parameter ζ. The probability of choosing the next node is proportional to the degree of the node with exponent ζ: At node u, $q_{uv} \propto \deg_v^\zeta$ where q_{uv} is the probability of choosing the neighbor v. By normalization, we obtain $q_{uv} = \frac{\deg_v^\zeta}{\sum_{v \epsilon N_u} \deg_v^\zeta}$. Using $\zeta = 1$, we get the RW strategy in which the probability of choosing the next node is proportional to the degree of the node. There are no deadlocks in $\mathrm{RW_{PD}}$ with parameter $\zeta = 1$. On the other hand, the $\mathrm{RW_{PD}}$ strategy with $\zeta = 0$ is same as RW.

8.1.5 Random Walk using minimum link weight

In RW using minimum link weight ($\mathrm{RW_W}$), the next hop is chosen as the link with minimum weight. Thus, at node u the next node v is chosen such that $w_{uv} = \min_{s \epsilon N_u} w_{us}$, where w_{us} is the weight of link between node u and node s. Deadlocks in RW using minimum link weight are similar to the deadlocks occurring in RW using highest degree. Indeed, the degree can be considered as a special kind of link weight.

8.1.6 Random walk proportional to the link weight

The probability of choosing the next node is proportional to the ζ-th power of the link weight: At node u, $y_{uv} \propto w_{uv}^\zeta$ where y_{uv} is the probability of choosing the neighbor v. By normalization, we obtain $y_{uv} = \frac{w_{uv}^\zeta}{\sum_{v \epsilon N_u} w_{uv}^\zeta}$. Using $\zeta = -1$, we get the RW strategy in which the probability of choosing the next node is inversely proportional to the link weight.

Table 1 lists the various RW strategies and explanation of how the next hop is chosen in different strategies.

8.2 Analysis of Searching with single query

The probability distribution of hopcount for a RW on a complete graph K_N is a geometric distribution [70, 38],

$$\Pr[H = j] = \frac{1}{N-1}\left(1 - \frac{1}{N-1}\right)^{j-1} \tag{8.1}$$

For multiple items $m = q\,(N-1)$ uniformly located over K_N, the hopcount distribution is

$$\Pr[H = j] = \frac{m}{N-1}\left(1 - \frac{m}{N-1}\right)^{j-1} \tag{8.2}$$

Table 8.1: Explanation of different RW strategies.

RW strategy	Next hop
RW	uniformly among the neighbors.
RW_{M}	uniformly among the neighbors that are not in the memory list $\mathcal{M}$.
$RW_{LA = 1}$	destination, if it is a direct neighbor, else uniformly among the neighbors.
$RW_{LA = 1; M}$	destination, if it is a direct neighbor, else uniformly among the neighbors that are not in the memory list $\mathcal{M}$.
RW_{HD}	neighbor with the highest degree.
$RW_{HD; M}$	highest degree neighbor that is not in the memory list $\mathcal{M}$.
$RW_{HD; LA = 1}$	destination, if the destination is among the neighbors, else highest degree neighbor.
$RW_{HD; LA = 1; M}$	destination, if the destination is among the neighbors, else highest degree neighbor not in the memory list $\mathcal{M}$.
RW_{PD}	probability of choosing the neighbor is proportional to the degree of neighbor.
$RW_{PD; M}$	probability of choosing the node is proportional to the degree of the node but nodes in the memory list $\mathcal{M}$ are not considered.
$RW_{PD; LA = 1}$	destination, if the destination is among the neighbors, else probability of choosing the node is proportional to the degree of the node.
$RW_{PD; LA = 1; M}$	destination, if the destination is among the neighbors, else probability of choosing the node is proportional to the degree of the node but nodes in the memory list $\mathcal{M}$ are not considered.
RW_{W}	link with minimum weight.
$RW_{W; LA = 1; M}$	destination, if it is among the neighbors, else neighbor with minimum weight among the neighbors that are not in the memory list $\mathcal{M}$.
RW_{PW}	probability of choosing the link is proportional to the link weight.
$RW_{PW; LA = 1; M}$	destination, if the destination is among the neighbors, else probability of choosing the link is proportional to the link weight but nodes in the memory list $\mathcal{M}$ are not considered.

and the expected hopcount is $\frac{N-1}{m}$.

With memory M for $\mathrm{RW_M}$ on a complete graph K_N,

$$\Pr\left[\hat{H}=j\right]=\frac{1}{N-1} \tag{8.3}$$

which means that the hopcount for $\mathrm{RW_M}$ is a uniform random variable on $[1, N-1]$ when $1 \leq j \leq M$. The probability distribution of RW with memory M for $j \geq M+1$ is [38],

$$\Pr\left[H_M=j\right]=\frac{1}{N-1}\left(\frac{N-(M+2)}{N-(M+1)}\right)^{j-M-1} \tag{8.4}$$

Thus, the probability distribution for a $\mathrm{RW_M}$ on the complete graph K_N is a uniform random variable on $[1, M]$ and geometric variable on $[M+1, N-1]$. With multiple data items uniformly located over K_N, $\frac{1}{N-1}$ is replaced by q in (8.3) and (8.4).

We now compute the probability of deadlocks and the hopcount distribution for $\mathrm{RW_M}$. Let the $\mathrm{RW_M}$ be at node n_i after $j-1$ hops. We have used n_i instead of n_j since there may be loops in $\mathrm{RW_M}$ and this node might have been visited before. In the next step, the $\mathrm{RW_M}$ has to choose one of the neighbors as the next hop (hop j). Let $X_{j;n_i}$ represent the event that all neighbors of node n_i have been visited before hop j. Clearly, $\Pr\left[X_{1;n_i}\right]=0$ for all $n_i \in N$. Let $\deg_{n_i}$ represent the degree of node n_i. Let Φ_j be the probability that there is a deadlock before making hop j and Φ denote the probability that there is a deadlock in $\mathrm{RW_M}$. A deadlock can happen while choosing hop j iff $\deg_{n_i} \leq M$ and all the neighbors of node n_i have been visited before[1]. In a complete graph since degree of any node n_i is always greater than any M i.e., $\Pr\left[\deg_{n_i} > M\right]=1$, the probability of deadlock before making hop j is always zero. In general,

$$\Pr\left[\Phi_j\right]=\Pr\left[X_{j;n_i} \mid \deg_{n_i} \leq M\right] \cdot \Pr\left[\deg_{n_i} \leq M\right] \tag{8.5}$$

The probability of deadlock in $\mathrm{RW_M}$ is

$$\begin{aligned}\Pr\left[\Phi\right] &= \Pr\left[\text{deadlock}_{hop1}\right]+\Pr\left[\text{deadlock}_{hop2}\right]+\Pr\left[\text{deadlock}_{hop3}\right]+... \\ &\leq \sum_{j=1}^{\infty}\Pr\left[\Phi_j\right]=\sum_{j=1}^{\infty}\Pr\left[X_{j;n_i} \mid \deg_{n_i} \leq M\right] \cdot \Pr\left[\deg_{n_i} \leq M\right]\end{aligned} \tag{8.6}$$

The probability of deadlock ($\Pr\left[\Phi\right]$) is less than the sum of probabilities of deadlock at each hop ($\sum \Pr\left[\Phi_j\right]$) since we did not consider the probability of finding the destination at any hop while computing $\Pr\left[\Phi_j\right]$.

In ER random graphs, the probability of deadlock in $\mathrm{RW_M}$ tends to 0 when $M < (1-\delta)\cdot p(N-1)$. For any fixed $\alpha, 0<\alpha<1$, let $\delta=\sqrt{\frac{9}{p(N-1)}\log\frac{N}{\alpha}}$. By Chernoff

[1] If memory list $\mathcal{M}$ is not full, then M represents the number of elements in the memory list.

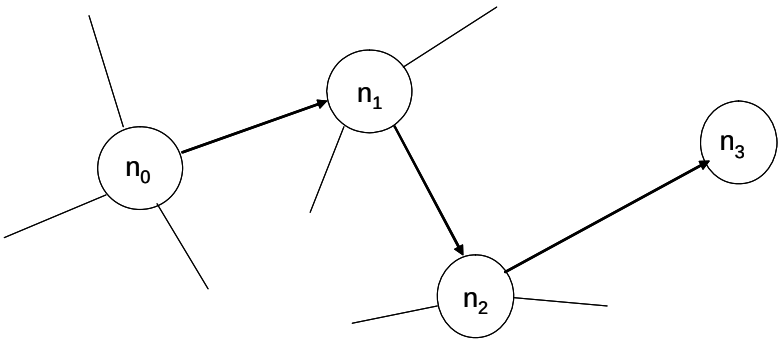


Figure 8.2: Diagram showing the progression of RW and $\mathrm{RW_M}$.

bounds [105],

$$\Pr\left[\left|\deg_{n_i} - p\left(N-1\right)\right| > \delta \cdot p(N-1)\right] \leq 2 \cdot \exp\left(-\frac{\delta^2 p\left(N-1\right)}{3}\right) \tag{8.7}$$

Substituting the above value of δ

$$\Pr\left[\left|\deg_{n_i} - p\left(N-1\right)\right| > \delta \cdot p(N-1)\right] \leq 2 \cdot \exp\left(-2\log\frac{N}{\alpha}\right) \leq \left(\frac{\alpha}{N}\right)^2 \tag{8.8}$$

Therefore, for $M < (1-\delta)\cdot p(N-1)$, $\Pr\left[\deg_{n_i} \leq M\right] \leq \left(\frac{\alpha}{N}\right)^2$. Hence using (8.5), the probability of deadlock before making hop j, $\Pr\left[\Phi_j\right] \leq \left(\frac{\alpha}{N}\right)^2$. The general expression for probability of deadlocks and the probability distribution of hopcount for $\mathrm{RW_{M=N\text{-}1}}$on $G\left(N, L\right)$ is computed below. Let n_0 be the source node (Figure 8.2). Since $\Pr\left[\Phi_1\right] = 0$, the probability of deadlock before making first hop

$$\Pr\left[\text{deadlock}_{hop1}\right] = \Pr\left[\mathfrak{E}_0\right] \cdot \Pr\left[n_0 \neq v\right] \cdot \Pr\left[\Phi_1\right] = 0$$

Let $\mathfrak{E}_1$ denote the event that $\mathrm{RW_{M=N\text{-}1}}$ has made 1 hop already and that node n_1 is the current node.

$$\Pr\left[H=1\right] = \Pr\left[\mathfrak{E}_1\right] \cdot \Pr\left[n_1 = v\right] = \frac{1}{N-1}$$

$$\begin{aligned}\Pr\left[\text{deadlock}_{hop2}\right] &= \Pr\left[\mathfrak{E}_1\right] \cdot \Pr\left[n_1 \neq v\right] \cdot \Pr\left[\Phi_2\right] \\ &= \left(1 - \frac{1}{N-1}\right) \cdot \Pr\left[\Phi_2\right]\end{aligned}$$

Now a transition is made from node n_1 to node n_2. Let $\mathfrak{E}_2$ denote the event that $\mathrm{RW_{M=N\text{-}1}}$ has made a transition to node n_2. Therefore, $\Pr\left[\mathfrak{E}_2\right] = \Pr\left[\mathfrak{E}_1\right] \cdot \Pr\left[n_1 \neq v\right] -$

$\Pr[\text{deadlock}_{hop2}]$

$$\begin{aligned}\Pr[H=2] &= \Pr[\mathfrak{E}_2]\cdot\Pr[n_2=v] = \frac{1}{N-2}\cdot\left(1-\frac{1}{N-1}\right)\cdot(1-\Pr[\Phi_2]) \\ &= \frac{1}{N-1}\cdot(1-\Pr[\Phi_2])\end{aligned}$$

$$\begin{aligned}\Pr[\text{deadlock}_{hop3}] &= \Pr[\mathfrak{E}_2]\cdot\Pr[n_2\neq v]\cdot\Pr[\Phi_3] \\ &= \left(1-\frac{1}{N-1}\right)\left(1-\frac{1}{N-2}\right)(1-\Pr[\Phi_2])\Pr[\Phi_3]\end{aligned}$$

Thus, the probability of deadlock and hopcount distribution for $\text{RW}_{\text{M=N-1}}$ on graph $G(N,L)$ is

$$\Pr[\text{deadlock}_{hopj}] = \left(\frac{N-j}{N-1}\right)(1-\Pr[\Phi_2])(1-\Pr[\Phi_3])...(1-\Pr[\Phi_{j-1}])\Pr[\Phi_j] \quad (8.9)$$

$$\Pr[H=j] = \frac{1}{N-1}(1-\Pr[\Phi_2])(1-\Pr[\Phi_3])...(1-\Pr[\Phi_{j-1}])(1-\Pr[\Phi_j]) \quad (8.10)$$

With multiple data items $m = q(N-1)$ uniformly located over the network, the hopcount distribution is

$$\Pr[H=j] = \frac{m}{N-j}\left(1-\frac{m}{N-1}\right)...\left(1-\frac{m}{N-j-1}\right)(1-\Pr[\Phi_2])...(1-\Pr[\Phi_j]) \quad (8.11)$$

Since we know that in ER random graphs, when $M \leq (1-\delta)p(N-1)$, $\Pr\left[\deg_{n_i} \leq M\right]$ is small and when $M > (1+\delta)p(N-1)$, $\Pr\left[\deg_{n_i} < M\right]$ is close to 1, the probability of deadlock in $\text{RW}_{\text{M=N-1}}$ can be approximated as

$$\Pr[\Phi] \leq \sum_{j=1}^{N}\Pr\left[X_{j;n_i} \mid \deg_{n_i} \leq M\right]\cdot\Pr\left[\deg_{n_i} \leq M\right] \approx \sum_{j=(1+\delta)p(N-1)}^{N}\Pr[X_{j;n_i}]$$

When the link density p is small, the probability of deadlocks is large. However, if p is sufficiently large, then $\Pr[\Phi]$ is close to 0. In power law graphs, since there are large number of nodes with very small degree, the probability of deadlocks becomes large even for small values of the memory M.

It is difficult to compute the exact expression for $\text{RW}_{\text{LA=1}}$ because of the loops. We compute the expression for probability of deadlocks and the probability distribution of hopcount for $\text{RW}_{\text{LA=1;M=N-1}}$ on ER random graphs. Let S_{n_i} represent the set of neighbors of node n_i that have not been seen before. Then,

$$\begin{aligned}\Pr[H=1] &= \frac{\sum_{j=0}^{N-1} j\Pr[\deg=j]}{N-1} \\ &= p\end{aligned}$$

If the destination is not found at hop 1, then $\mathrm{RW}_{\mathrm{LA=1;M}}$ reaches node n_1. Since node n_1 is connected to node n_0 (n_0, n_1), the probability that node n_1 has degree j is the probability that node n_1 has degree $(j-1)$ from the remaining $N-2$ nodes.

$$\begin{aligned}\Pr\left[\deg_{n_1} = j \mid (n_0, n_1)\right] &= \Pr\left[\deg_{n_1} = j-1 \mid (N-2) \text{ nodes}\right] \\ &= \binom{N-2}{j-1} p^{j-1} (1-p)^{N-1-j}\end{aligned}$$

Each of neighbors of node n_1 is connected to node n_0 with probability p, thus

$$\begin{aligned}\Pr[H=2] &= \frac{\sum_{j=0}^{N-2} (j+1) \binom{N-2}{j} p^j (1-p)^{N-2-j} - 1 - p \sum_{j=0}^{N-2} j \binom{N-2}{j} p^j (1-p)^{N-2-j}}{N-1} \\ &= p(1-p)\frac{N-2}{N-1}\end{aligned}$$

$$\Pr[\text{deadlock}_{hop2}] = \Pr[\mathfrak{E}_1] \cdot \Pr[v \notin S_{n_1}] \cdot \Pr[\Phi_2] = (1-p)\left(1 - p\frac{N-2}{N-1}\right) \cdot \Pr[\Phi_2]$$

Now a transition is made from node n_1 to node n_2. If node n_2 is connected to node n_1 and not n_0 (n_2, n_1), then the probability that node n_2 has degree j is

$$\begin{aligned}\Pr\left[\deg_{n_2} = j \mid (n_2, n_1)\right] &= \Pr\left[\deg_{n_2} = j-1 \mid (N-3) \text{ nodes}\right] \\ &= \binom{N-3}{j-1} p^{j-1} (1-p)^{N-2-j}\end{aligned}$$

If node n_2 is connected to node n_1 and n_0 (n_2, n_1, n_0), then the probability that node n_2 has degree j is

$$\begin{aligned}\Pr\left[\deg_{n_2} = j \mid (n_2, n_1, n_0)\right] &= \Pr\left[\deg_{n_2} = j-2 \mid (N-3) \text{ nodes}\right] \\ &= \binom{N-3}{j-2} p^{j-2} (1-p)^{N-1-j}\end{aligned}$$

Using the above expressions, the probability that hopcount is 3 can be computed as

$$\Pr[H=3] = p(1-p)^2 \frac{N-3}{N-1}(1 - \Pr[\Phi_2])$$

Thus, the probability distribution of hopcount for $\mathrm{RW}_{\mathrm{LA=1;M=N\text{-}1}}$ in ER random graph is

$$\Pr[H=j] = p(1-p)^{j-1} \frac{N-j}{N-1}(1 - \Pr[\Phi_2]) \ldots (1 - \Pr[\Phi_{j-1}]) \tag{8.12}$$

With multiple items $m = q(N-1)$ uniformly distributed over the network, the probability distribution of hopcount (neglecting the term $\frac{N-j}{N-1}$) can be approximated by

$$\Pr[H=j] = \frac{p}{q}\left(1 - \frac{p}{q}\right)^{j-1} (1 - \Pr[\Phi_2]) \ldots (1 - \Pr[\Phi_{j-1}]) \tag{8.13}$$

8.3 Simulation Results

This section is divided into three parts. The first part shows the simulation results for RW strategy with lookahead j ($j \geq 1$), which is generalization of $\mathrm{RW_{LA}}$ strategy. In the second part, simulation results comparing different RW strategies for ER random graphs and BA power law graphs are presented. In each simulation, 10^6 random graphs of the class $G_p(N)$ are constructed. The source and destination are chosen uniformly.

8.3.1 RW with lookahead j

The RW with lookahead j ($\mathrm{RW_{LA\,=\,j}}$) is a RW strategy in which each node has topology information upto j hops, i.e. each node maintains a level set[2] L_N upto level j. If the destination is located within the level set L_N, a shortest path is chosen. However, if the destination is not located within the level set L_N, then the next node is chosen uniformly among the neighbors of the node. In the extreme case, when each node maintains topology information about the entire network, the $\mathrm{RW_{LA\,=\,j}}$ strategy leads to deterministic routing [103]. Indeed, it has been shown that there is a phase transition in the performance of $\mathrm{RW_{LA\,=\,j}}$, when j is equal to the average path length [103]. Moreover, with look-ahead greater than the average path length, the performance of $\mathrm{RW_{LA\,=\,j}}$ is similar to the shortest path routing [103].

When look-ahead ≥ 2, the probabilities that the hopcount is one and two can be calculated exactly for ER random graphs [105]:

$$\Pr\left[H_N = 1\right] = p$$

$$\Pr\left[H_N = 2\right] = (1-p)\left(1 - \left(1-p^2\right)^{N-2}\right) \tag{10}$$

As the link density p and the network size N are increased, $\Pr[H_N > 2] = (1-p)\left[1-p^2\right]^{N-2}$ becomes negligible. Therefore, for large values of link density p, the performance of $\mathrm{RW_{LA\,=\,2}}$ is comparable to Dijkstra's shortest path algorithm. Figure 8.3 shows the comparison for $\mathrm{RW_{LA\,=\,2}}$ and Dijkstra's shortest path algorithm for $N = 100$ and different values of the link density p. The insert in Figure 8.3 shows the expected hopcount for different values of look-ahead for $N = 800$ and $p = 0.0084$. As shown in [105], the expected hopcount between any two nodes in ER random graphs can be approximated by $\frac{\log N}{\log p(N-1)}$. Therefore, for look-ahead greater than $\frac{\log N}{\log p(N-1)} = 3.5$, the expected hopcount for $\mathrm{RW_{LA\,=\,j}}$ is constant and equal to the expected hopcount obtained by using Dijkstra's algorithm (see insert Figure 8.3).

[2]Level set at level 1 is defined [105] as the set of nodes 1 hop away from the source node, level set at level 2 is defined as the set of nodes 2 hops away from the source node and so on.

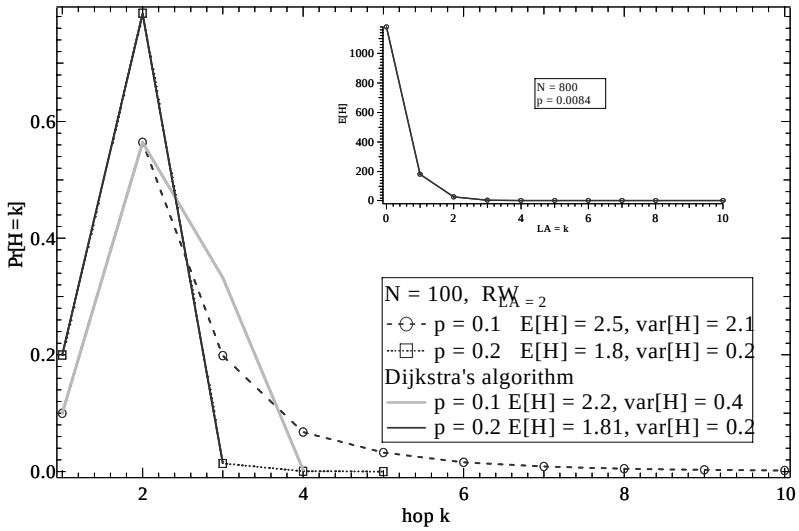


Figure 8.3: Comparison of the $\mathrm{RW}_{\mathrm{LA}\,=\,2}$ and the Dijkstra's shortest path algorithm for $N = 100$ and different values of the link density p. The insert shows expected hopcount for different values of look-ahead for $N = 800$ and $p = 0.0084$.

8.3.2 Comparison of RW strategies

In ER random graphs, at large values of the link density p, $E[H_{\mathrm{RW}}] \sim N$ and therefore $E[H_{\mathrm{RW}_{\mathrm{LA}\,=\,1}}] \sim \frac{1}{p}$. Moreover, $\Pr[\Phi] \longrightarrow 0$ for any M for large values of the link density p. Also, $p\,(1-p)^{j-1}$ is a good approximation when p is large for $\mathrm{RW}_{\mathrm{LA=1;M=N\text{-}1}}$ (Figure 8.4).Thus, when the link density p is large, the results for different RW strategies on a random graph $G_p(N)$ are similar to a complete graph K_N.

We compare the performance of RW strategies in sparse graphs i.e., random graphs with small link density p. The sparse graphs are particularly interesting since the average degree for most real networks like Gnutella [71] and Internet [2] is known to be small.

Figure 8.5 shows the simulation results for a sparse ER random graph. The Pr[success] represents the probability that destination is found by the query. The simulations indicate that the mean hopcount is linear in N for these strategies and the mean hopcount for the $\mathrm{RW}_{\mathrm{LA}\,=\,1}$ and the RW strategies are related by the expression $E[H_{\mathrm{RW}_{\mathrm{LA}\,=\,1}}] \sim \frac{E[H_{\mathrm{RW}}]}{pN}$. In $\mathrm{RW}_{\mathrm{M=N\text{-}1}}$, the probability of deadlock $\Pr[\Phi] \longrightarrow 1$ as N is increased.

Figures 8.6 shows the simulation results for preferential attachment graphs generated using the BA model. $\mathrm{RW}_{\mathrm{LA=1}}$ and $\mathrm{RW}_{\mathrm{HD}}$ are not effective as might be expected since there are large number of nodes with small degree in this model. It has been shown by Mihail *et al.* [75] that in power law graphs, the expected number of hops for $\mathrm{RW}_{\mathrm{LA=1}}$

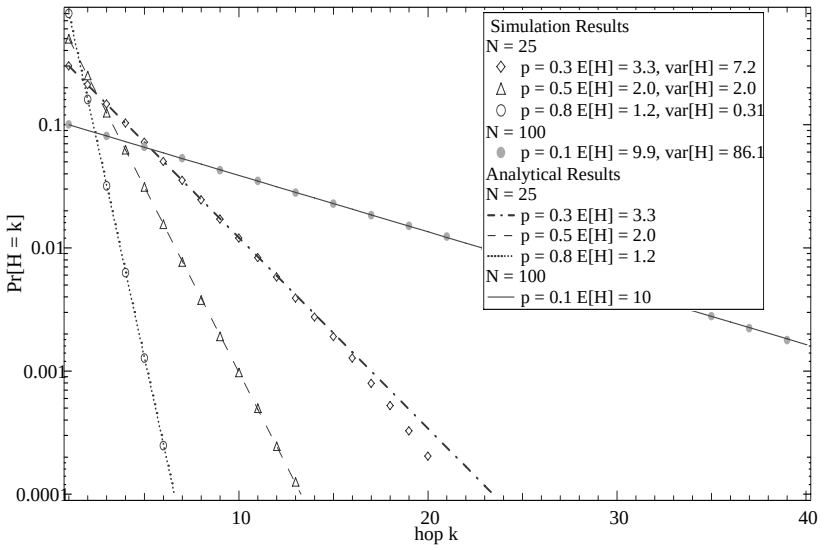


Figure 8.4: Comparison of the expression $\Pr[H = j] \cong p(1-p)^{j-1}$ and simulation results for $\mathrm{RW}_{\mathrm{LA\ =\ 1;\ M\ =\ N\text{-}1}}$ on a random graph for $N = 25, 100$ and different values of the link density p.

to discover $\Omega\left(N^{1-\epsilon\left(\frac{1}{2}-\varepsilon\right)}\right)$ nodes is $O\left(N^{\frac{1}{2}+\varepsilon}\log N\right)$ where $0 < \varepsilon < \frac{1}{2}$. This shows that look-ahead is not effective for power law graphs. Using $\mathrm{RW}_{\mathrm{M=N\text{-}1}}$, the probability that there is a deadlock is close to 1 when N is large.

The $\mathrm{RW}_{\mathrm{HD;\ LA=1;\ M=N\text{-}1}}$ strategy performs best in both graph topologies. The expected hopcount for certain RW strategies such as $\mathrm{RW}_{\mathrm{HD}}$ is similar for both the graph topologies. However, the average hopcount does not accurately reflect the performance of different search strategies since these strategies generally lead to infinite loops. Look-ahead and memory both improve performance in power law graph and ER random graphs. However, the performance with $M = N - 1$ and $LA = 1$ improves more in ER graphs then the BA model. Moreover, in ER random graphs and power law graphs generated using the BA model, the performance of $\mathrm{RW}_{\mathrm{LA}}$ is close to shortest path routing with small values of look-ahead. This is due to the small diameter ($O(\log N)$) of these graph topologies [22, 23].

8.4 Conclusion

The hopcount distribution for RW strategies based on memory and lookahead is close to a geometric variable for random graphs. We also investigated the occurrence of deadlocks in various RW strategies. Deadlocks have been neglected in most of the previous

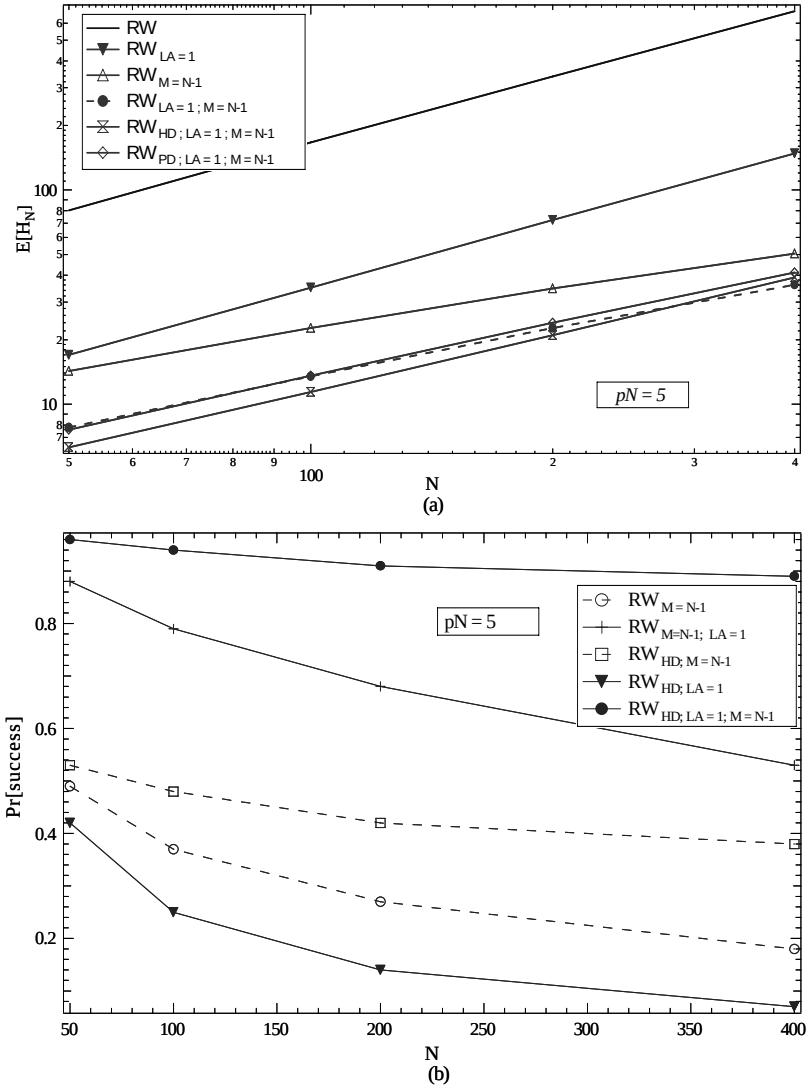


Figure 8.5: (a) The mean hopcount for different RW strategies as a function of N with constant number of neighbors ($pN = 5$) for ER graphs. (b) The Pr[success] for different RW strategies for $N = 200$ and link density $p = 0.025$.

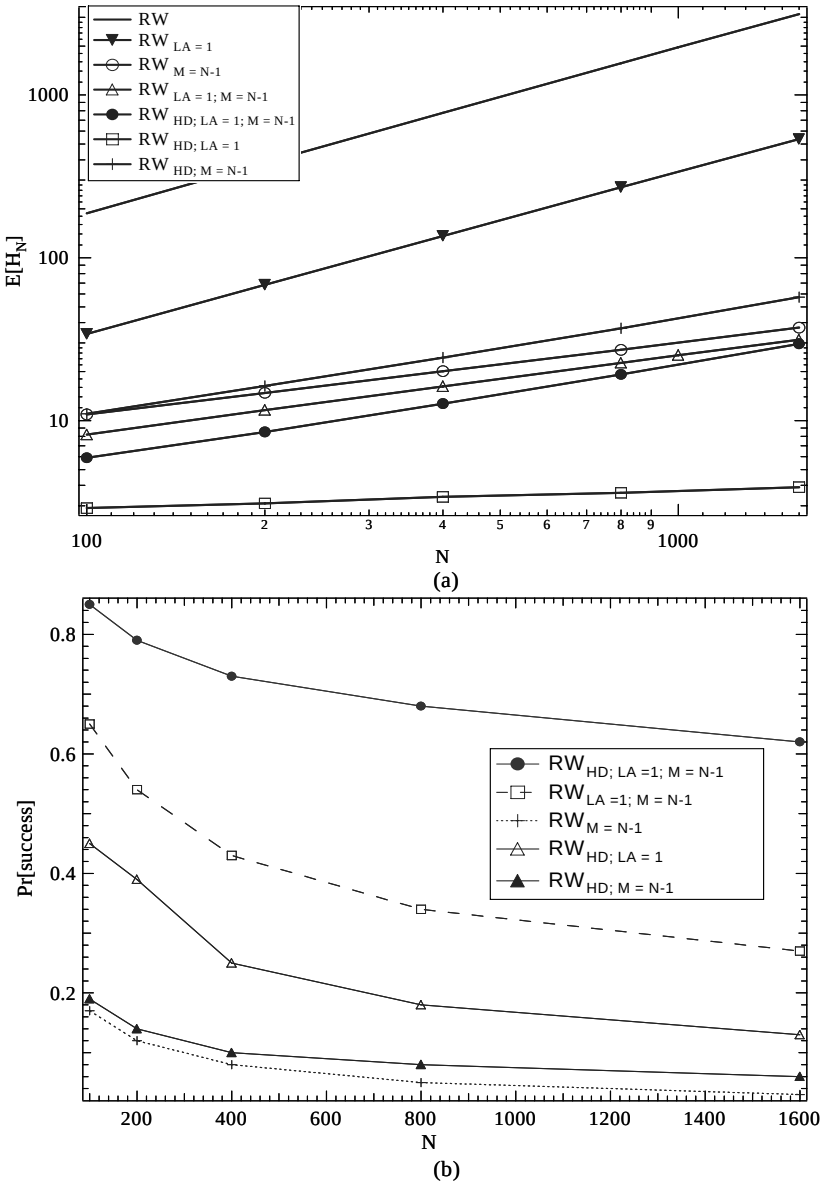


Figure 8.6: (a) The mean hopcount for different RW strategies as a function of N for power law graph generated using BA model (b) The Pr[success] for different RW strategies.

search strategy comparisons. A mere comparison of the expected hopcount or weight of different RW strategies gives little insight into the performance of these strategies. The simulations showed that for RW strategies using highest degree or minimum link weight, the expected hopcount or weight is small but these strategies generally lead to deadlocks. In conclusion, RW strategies using highest degree or minimum link weight perform well when used in combination with memory or look-ahead. Also, for different RW strategies, if the probability of finding the destination is large, the expected hopcount for the particular strategy is also large and vice versa.

Chapter 9

Searching with multiple queries

Searching overhead or packet overhead is defined as the product of number of hops for each query (packet) and number of queries (packets) i.e., the total number of packets exchanged. Our work in this chapter shows that searching using multiple RW queries reduces the overhead for searching as compared to flooding with sequence numbers.

It is known that the number of hops required for uniform sampling by RW can be as low as the number of samples in independent uniform sampling [13, 51]. Thus, if the RW starts at any node and makes h hops, and using each visited node as a sample point, approximately the same statistical properties can be achieved as h independent uniform samples. However, our simulations show that there are large differences in the behavior of RW as compared to independent uniform sampling particularly when the TTL is small.

A single RW query has an overhead and time to discover of $O(N \log N)$ for ER random graphs and BA power law graphs [28, 63]. Since the time to discover is large in RW, we split a single RW query into Q multiple queries each with a given TTL. Each RW query stops when the TTL is reached or the destination is located (In $\mathrm{RW_M}$, the query also stops if there is a deadlock). Since each RW query makes at most TTL hops, the worst-case searching overhead is $Q \times TTL$. The expected number of hops are less since any query stops once the destination is found. However, there is a probability that a destination is never located by the multiple queries. Thus, with multiple queries we define the probability of success as the probability that the destination is located by at least one query out of the Q queries generated. In flooding with sequence numbers, the probability of success is defined as the probability that a destination is located with a given TTL. We want to minimize TTL and Q and maximize Pr [success].

9.1 Analysis of searching with multiple RW queries

To analyze the performance of multiple RW queries, we define the efficiency as the inverse of expected packet overhead needed to discover the destination node. The efficiency is normalized by multiplying by N. The gain of searching in one scheme over another is defined as the ratio of efficiency for the corresponding schemes.

$$\mathfrak{n} = \frac{\text{Number of iterations}\,(\Pr\,[\text{success}])}{\text{Total number of packets exchanged}} \times N \tag{9.1}$$

$$ga\left(\frac{\text{RW}_\text{M}}{\text{RW}}\right) = \frac{\mathfrak{n}\,(\text{RW}_\text{M})}{\mathfrak{n}\,(\text{RW})} \tag{9.2}$$

Consider a single RW and RW_M on a complete graph K_N. The searching overhead can be approximated by the product of expected hopcount to discover the destination and the number of iterations. Since in RW and RW_M, the expected hopcount is $N-1$ and $(N-1)/2$ respectively, the gain $ga\left(\frac{\text{RW}_\text{M}}{\text{RW}}\right) = 2$ for a complete graph K_N. In flooding with $TTL = 1$, $\mathfrak{n}$ is $N/(N-1)$ and with $TTL = 2$, $\mathfrak{n}$ is $N/(N-1)^2$. Thus, the efficiency of flooding depends on the value of TTL and the efficiency decreases with TTL.

Table 9.1 and 9.2 show the efficiency for a single RW and RW_M query and flooding with sequence numbers for different values of N and p. The results for flooding in Tables 9.1 and 9.2 are for optimized values of TTL such that $\Pr\,[\text{success}]$ is close to 1. The gain obtained by using RW over flooding is significant, particularly, when the link density p is large. In addition, $\text{RW}_{\text{M=N-1}}$ is a more efficient way of searching than RW since the expected number of hops required to find the destination or to reach a deadlock is less. However, the probability of deadlocks is high when the link density p is small. As the link density p is decreased, efficiency for both RW and RW_M decreases. This is in contrast to flooding where the efficiency increases as the link density is decreased.

Table 9.1: Efficiency of searching by using flooding, and a single RW or RW_M query for dense ER random graph ($pN = 80$) for different values of N

$pN = 80$					
N	$\mathfrak{n}\,(\text{RW})$	$\mathfrak{n}\,(\text{RW}_\text{M})$	$\mathfrak{n}\,(\text{flooding})$	$g\left(\frac{\text{RW}}{\text{flooding}}\right)$	$g\left(\frac{\text{RW}_\text{M}}{\text{RW}}\right)$
100	1.17	1.82	0.015	78	1.55
200	1.08	2.16	0.03	36	2
400	1.05	1.997	0.06	17.5	1.90
800	1.017	2.0	0.12	8.5	1.98

Table 9.2: Efficiency of searching by using flooding, and a single RW and $\mathrm{RW_M}$ query for sparse ER random graph ($pN = 10$) for different values of N

$pN = 10$						
N	$\mathfrak{n}$ (RW)	$\mathfrak{n}(\mathrm{RW_M})$	Pr[success]	(flooding)	$g\left(\frac{\mathrm{RW}}{\mathrm{flooding}}\right)$	$g\left(\frac{\mathrm{RW_M}}{\mathrm{RW}}\right)$
100	0.84	1.6	0.73	0.14	6	1.89
200	0.84	1.54	0.66	0.21	4	1.83
400	0.82	1.47	0.59	0.11	7.5	1.79
800	0.8	1.4	0.52	0.14	5.7	1.76

A single RW leads to lower overhead to locate a destination than flooding with sequence numbers. However, the time to search for the destination is much larger than in flooding. Since we want to maximize Pr [success] and minimize TTL, we split the single RW or $\mathrm{RW_M}$ into multiple queries with fixed TTL such that $Q \times TTL = N \log N$. Table 9.3 and 9.4 show the results for multiple RW and $\mathrm{RW_M}$ queries for ER random graph with $N = 400$ and link density $p = 0.015$ and 0.2 respectively. In Table 9.4, since the link density p is large, Pr [success] $\simeq 1$ and is not shown.

The probability of success is very low in $\mathrm{RW_M}$ with only a single query when the link density p is small. However, when split into multiple queries, Pr [success] increases. As the TTL is decreased and Q is increased, the efficiency decreases for both RW and $\mathrm{RW_M}$. The decrease in efficiency occurs with small TTL since most of the queries search only the neighboring nodes which have been visited already by other queries. There is also a decrease in efficiency because the number of queries and the TTL is fixed. Thus, multiple queries might locate the destination. The terminating conditions can be included which improve the efficiency but increases the complexity of searching algorithms. For example, a scheme is proposed in [71], where the query checks with the source node whether the destination is located. As shown by tables 9.3 and 9.4, the efficiency decreases by a factor of 3 as the TTL is decreased from 2400 to 120. Moreover, when the TTL is small, the gain obtained by using $\mathrm{RW_M}$ over RW is small. Therefore, only for large values of link density p and TTL, $\mathrm{RW_M}$ is a more efficient way of searching than RW. If we use $Q > 1$ with $TTL = 2400$, the efficiency decreases.

Figure 9.1 shows Pr [success] versus the searching overhead (number of packets exchanged) and the worst time to discover the destination. Since the size of network is not known *a priori*, we also show simulations for query split into multiple queries with a different TTL. We use a linearly increasing TTL and the maximum time to discover is given by maximum $TTL_{\max}$.

$$TTL(l) = TTL_{\min} + (l-1) * add\_TTL \tag{9.3}$$

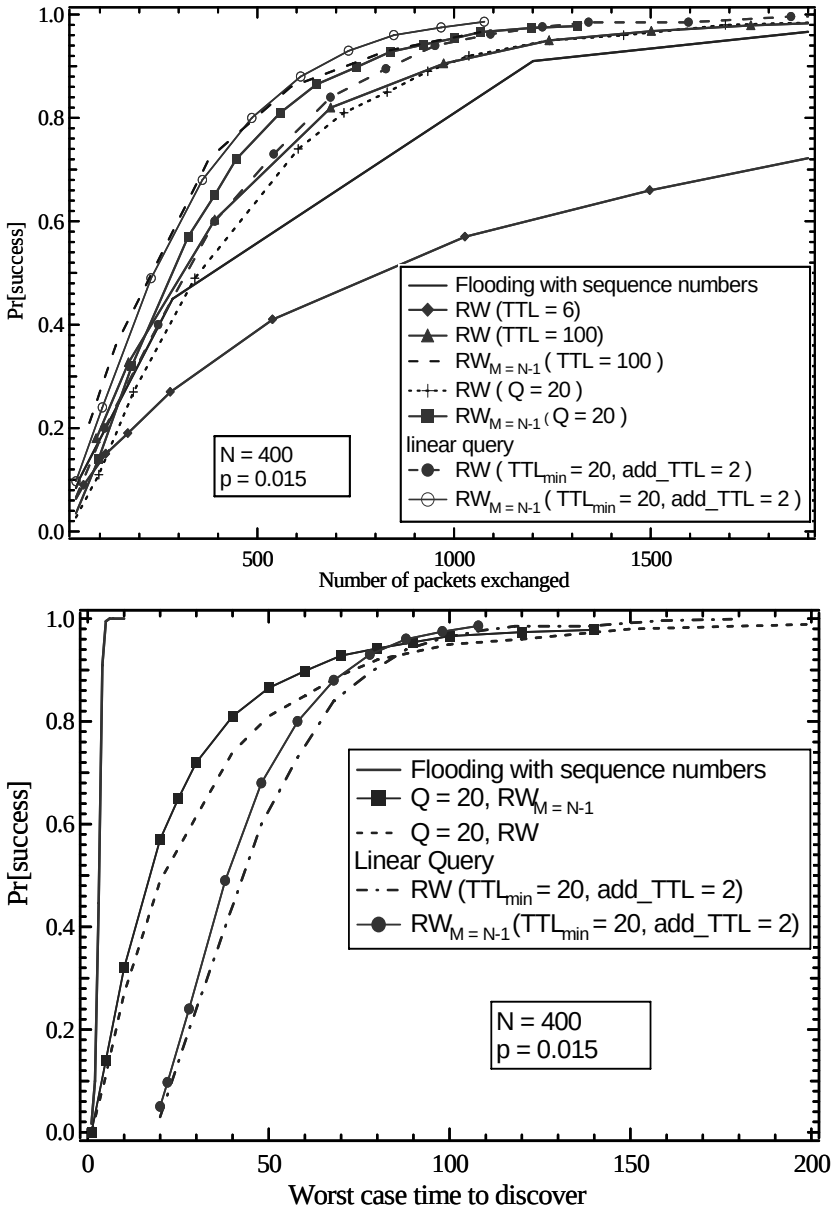


Figure 9.1: Performance of searching by using multiple queries in ER random graph for $N = 400$ and $p = 0.015$

Table 9.3: Efficiency of searching by multiple RW and $\mathrm{RW_M}$ queries for $p = 0.015$ and $N = 400$.

$pN = 6$	RW		$\mathrm{RW_M}$		$g(\frac{\mathrm{RW_M}}{\mathrm{RW}})$
$N = 400$	$\mathfrak{n}$	Pr[succ.]	$\mathfrak{n}$	Pr[succ.]	
$Q = 1, TTL = 2400$	0.7	0.996	1.23	0.28	1.8
$Q = 20, TTL = 120$	0.27	0.96	0.33	0.97	1.2
$Q = 120, TTL = 20$	0.24	0.95	0.26	0.97	1.1
$Q = 400, TTL = 6$	0.15	0.73	0.18	0.82	1.2

Table 9.4: Efficiency of searching using multiple RW and $\mathrm{RW_M}$ queries for $p = 0.2$ and $N = 400$.

	$pN = 80$		
$N = 400$	$\mathfrak{n}\,(\mathrm{RW})$	$\mathfrak{n}\,(\mathrm{RW_M})$	$g\left(\frac{\mathrm{RW_M}}{\mathrm{RW}}\right)$
$Q = 1, TTL = 2400$	1.05	1.997	2
$Q = 20, TTL = 120$	0.30	0.298	1
$Q = 120, TTL = 20$	0.285	0.285	1
$Q = 400, TTL = 6$	0.26	0.262	1

Using (9.3), the add_TTL parameter can be expressed as $\frac{TTL_{\max} - TTL_{\min}}{Q-1}$. Figure 9.1 shows that efficiency of searching by RW and $\mathrm{RW_M}$ decreases with TTL. This is in contrast to the assumptions made in the analysis in Bisnik and Abouzeid [13], where searching with multiple queries and independent uniform sampling are assumed to be equivalent. Also, the linear query performs as good as sending multiple queries with a large $TTL(100)$. Thus, the simulations show that searching by using a single RW query with TTL is more efficient than sending Q RW queries with $TTL'(TTL = Q \times TTL')$ for ER random graphs.

Figure 9.2 shows the results for searching with multiple queries in a BA power law graph. In these graphs, $\mathrm{RW_{M=2}}$ gives the best performance in terms of reducing search overhead. $\mathrm{RW_{M=N\text{-}1}}$ performs worse than $\mathrm{RW_{M=2}}$ since many of the queries end in a deadlock. This reduces the efficiency of the $\mathrm{RW_{M=N\text{-}1}}$ strategy and queries with larger TTL need to be sent to achieve the same probability of success as $\mathrm{RW_{M=2}}$. Moreover, the improvement in performance of different RW strategies compared to flooding is limited. Even with a large $TTL = 1000$, the RW does not perform better than flooding.

Figure 9.3 compares the performance of searching with RW and $\mathrm{RW_M}$ in an ER random graph and a BA power law graph using the same $TTL = 400$. The results are for $N = 10000$ and for an ER random graph with link density $p = 0.001$. Figure 9.3 also

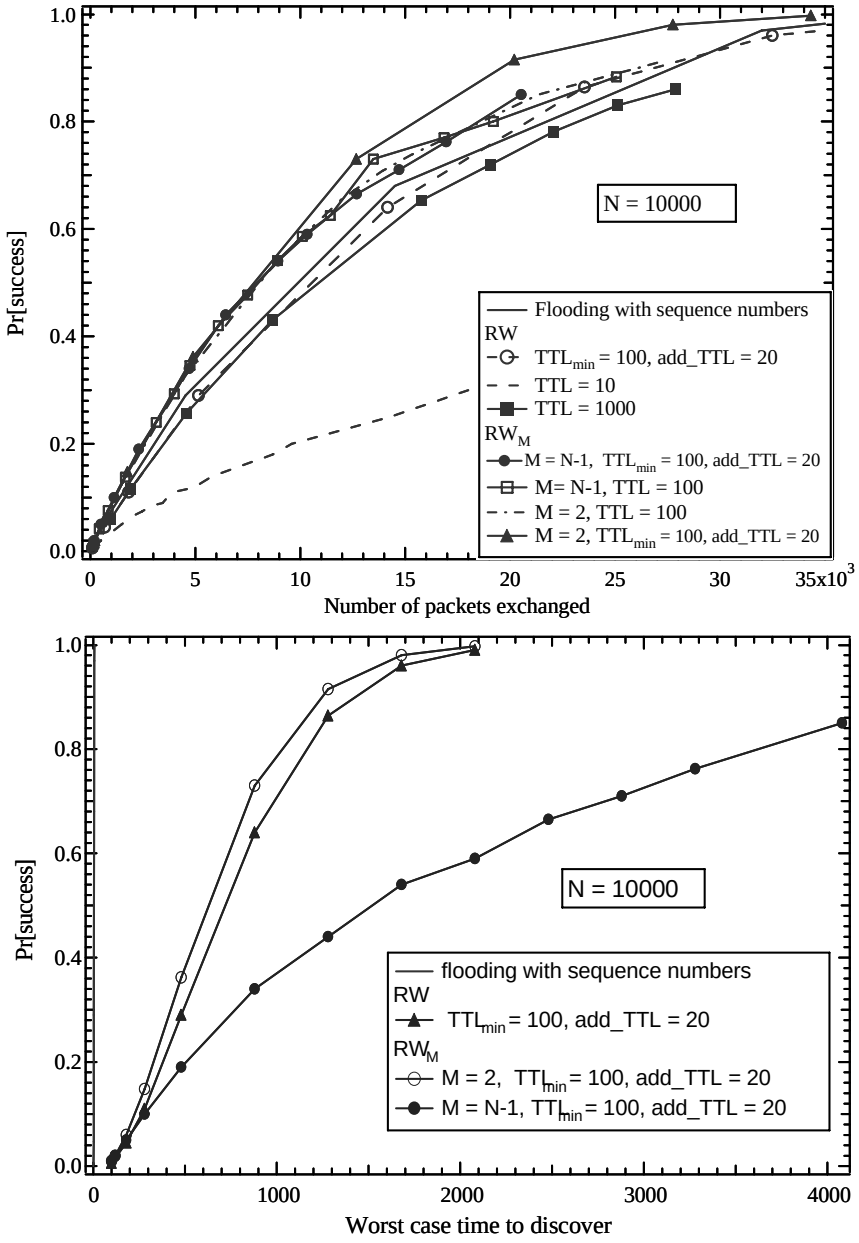


Figure 9.2: Performance of searching using RW and $\mathrm{RW_M}$ with multiple queries for BA power law graph ($N = 10000$).

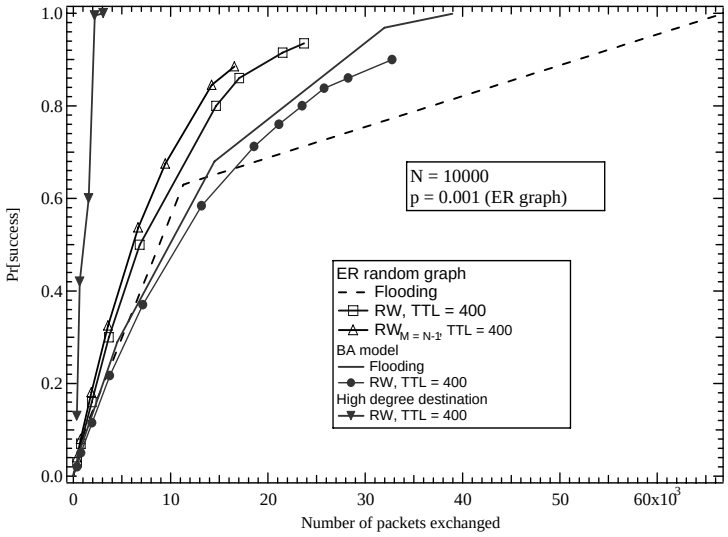


Figure 9.3: Comparison of searching by multiple RW and $\mathrm{RW_M}$ queries on ER random graph and BA power law graph. ($N = 10000$ and for ER random graph $p = 0.001$)

shows the results for searching for a high-degree destination node in a BA power law graph (the average degree of the destination is 72). Searching in ER random graphs performs better than in a BA power law graph. This can be attributed to the fact that most of the nodes in an ER random graph have a larger degree than the degree of nodes in a BA power law graph (section 2.2). Thus, in a BA power law graph, the RW makes a large number of hops among the low degree nodes while searching for the destination node. Moreover, in a BA power law graph, since the degree of uniformly chosen destination and source nodes is small, performance of searching for a uniformly chosen node is much worse than searching for a high degree node.

9.2 Conclusions

We have analyzed the performance of searching with multiple RW strategies in two types of graphs. The topology of graphs plays an important role in determining the performance of different search strategies. Searching with multiple RW queries performs better than flooding with sequence numbers in terms of overhead. The overhead can be reduced further by using multiple $\mathrm{RW_M}$ queries. However, the searching efficiency decreases with TTL for both RW and $\mathrm{RW_M}$. The performance of searching with multiple RW and $\mathrm{RW_M}$ queries is better in ER random graphs than in BA power law graphs. Moreover, in a BA power law graph, only small values of memory M improve performance of searching using multiple RW queries.

Part IV

Topology Analysis

Chapter 10

Topology of Ad hoc wireless networks

The topology of ad hoc networks is critical in determining the performance of routing algorithms, capacity and lifetime of ad hoc networks. In this chapter, we study the impact of different signal propagation models on the topology of ad hoc networks. We also analyze the impact of topology on shortest path routing based on distance and hopcount metrics.

We characterize the variation in received signal power over distance due to path loss and shadowing. Path loss is caused by dissipation of the power radiated by the transmitter as well as effects of the propagation channel. Path loss models generally assume that path loss is the same at a given transmitter-receiver distance [53]. Shadowing is caused by obstacles between transmitter and receiver that attenuate signal power through absorption, reflection, scattering and diffraction. Variation due to path loss occurs over very large distances (100-1000 meters), whereas variation due to shadowing occurs over distances proportional to the length of obstructing object (10-100 meters in outdoor environments and less in indoor environments). Since variations due to path loss and shadowing occur over relatively large distances, this variation is referred to as large-scale propagation effects. Variations in received signal power also occur due to the constructive and destructive addition of multipath signal components. Variations due to multipath occurs over very short distances, on the order of signal wavelength, so these variations are referred to as small-scale propagation effects.

The next section explains different signal propagation models. In section 10.2, we compute the average node degree with these models. Section 10.3 analyzes the effect of shadowing on traffic load. Finally, section 10.4 studies the lifetime of ad hoc wireless networks.

Table 10.1: Path loss exponent in different environments [53, 87].

Environment	α range
Urban macrocells	3.7-6.5
Urban microcells	2.7-3.5
Office Building (same floor)	1.6-3.5
Office Building (multiple floors)	2-6
Store	1.8-2.2
Factory	1.6-3.3
Home	3

10.1 Signal Propagation Models Topology Modeling of Wireless Ad hoc Networks

The complex nature of signal propagation prevents us to use a single model that characterizes path loss accurately across a range of different environments. However, a simplified path loss model that is generally used is:

$$P_{rx} = P_{tx} K \left(\frac{r}{r_0} \right)^{-\alpha} \tag{10.1}$$

where P_{tx} and P_{rx} are the transmitted and received power respectively, K is a constant, r_0 is a reference for distance for the antenna far-field, r is the distance between the transmitter and the receiver, and α is the path loss exponent.

A table summarizing α values for different indoor and outdoor environments and antenna heights at 900 MHz and 1.9 GHz is given in Table 10.1.

The models for path loss and shadowing can be combined to capture power falloff versus distance along with random attenuation about this path loss from shadowing. For the combined model, the ratio of received to transmitted power in dB is given by [53]:

$$10 \log_{10} P_{rx} = 10 \log_{10} P_{tx} + 10 \log_{10} K - 10\alpha \log_{10} \left(\frac{r}{r_0} \right) - \psi'_{dB} \tag{10.2}$$

where ψ'_{dB} is a Gaussian random variable with mean zero and variance $\sigma^2_{\psi_{dB}}$. Thus, the average power at a distance r from the transmitter is the same for both the path loss model and the path loss and shadowing model together.

The multipath effects are captured by ray-tracing models for deterministic channels. However, since deterministic channels are rarely available, the multipath channels are characterized statistically. The power distribution with Rayleigh fading is given by

[53, 76]:

$$p(x) = \frac{1}{P_{rx}} e^{-x/P_{rx}} \tag{10.3}$$

where P_{rx} is the average received signal power i.e., the received power based on path loss and shadowing alone.

The number of neighbors and the topology of an ad-hoc network becomes more complex if the interference is considered from other nodes. In this case, the signal to interference ratio (SINR) at the receiver is [46]:

$$SINR = \frac{P_{tx} f(l)}{G + \gamma \sum_{i=1}^{m} P_i f(l_i)} \tag{10.4}$$

where m is the number of interfering nodes, P_{tx} and P_i are the transmission powers of the transmitting node and the i^{th} interfering node, G is the additive white Gaussian noise. $f(l)$ denotes the path loss component based on the considered signal propagation model.

10.2 Average node degree in ad hoc wireless networks

A node is defined as a neighbor if the received power P_{rx} or SINR $\geq P_{\min}$. We first study the degree distribution for the path loss model. In the path loss model, any node within a circular area with radius ρ is a neighbor i.e., $P_{tx} K \rho^{-\alpha} = P_{\min}$. The following derivation for the probability distribution of degree of a node in an ad hoc network with arbitrary node distribution is given by Bettstetter [11]. Assume that a node is at a given location $\mathbf{x}$, where $\mathbf{x} = (x, y)$ in two dimensional Cartesian coordinates. A second node is randomly placed according to some arbitrary pdf $f_{\mathbf{X}}(\mathbf{x}')$. The two nodes have a link if the second node is placed within a circle of radius ρ around $\mathbf{x}$. The probability of the link is given by

$$\begin{aligned} p_0(\mathbf{x}) &= \iint_{A_0(\mathbf{x})} f_{\mathbf{X}}(\mathbf{x}')\, d\mathbf{x}' \\ &= \int_{y-\rho}^{y+\rho} \int_{x-\sqrt{\rho^2-(y'-y)^2}}^{x+\sqrt{\rho^2-(y'-y)^2}} f_{XY}(x', y')\, dx' dy' \end{aligned}$$

In the network with N nodes, the probability that a node has degree j is given by the binomial distribution

$$\Pr[\deg = j] = \binom{N-1}{j} (p_0(\mathbf{x}))^j (1 - p_0(\mathbf{x}))^{N-1-j}$$

with an expected value $(N-1)\,p_0(\mathbf{x})$. For small $p_0(\mathbf{x})$ and large N, the binomial distribution can be approximated by a Poisson distribution with mean $(N-1)\,p_0(\mathbf{x})$:

$$\Pr[\deg = j] \cong \frac{\lambda^j e^{-\lambda}}{j!}$$

Let us assume a uniform distribution of nodes with density δ on an infinitely large system plane (neglecting border effects). The limiting case of a uniform distribution is called a homogeneous Poisson point process of density δ. This process is defined by the following two properties [33]: (1) the number of nodes in each finite subarea follows a Poisson distribution, (2) the number of nodes in non-overlapping subareas are independent random variables. Thus, the degree distribution with uniform node density and path loss model is Poisson [57, 76] with mean number of neighbors

$$E[N_{PL}] = \delta\pi\left(\frac{P_{tx}K}{P_{\min}}\right)^{\frac{2}{\alpha}} = \delta\pi\rho^2 \tag{10.5}$$

where ρ is the transmission radius such that $P_{tx}K\rho^{-\alpha} = P_{\min}$ i.e., $\rho = \left(\frac{P_{tx}K}{P_{\min}}\right)^{\frac{1}{\alpha}}$. If white noise of power G is assumed to be present at the receiver, the transmission radius $\rho = \left(\frac{P_{tx}K}{GP_{\min}}\right)^{\frac{1}{\alpha}}$.

The topology and average node degree with path loss and shadowing has been studied extensively by Hekmat [57]. Hekmat and Van Mieghem [58] have shown that degree distribution with path loss and shadowing is Binomial with the probability of link between two nodes at normalized distance $\widehat{r} \triangleq \frac{r}{\rho}$:

$$p(\widehat{r}) = \frac{1}{2}\left[1 - \operatorname{erf}\left(3.07\frac{\log(\widehat{r})}{\xi}\right)\right], \quad \xi \triangleq \sigma_{\psi'_{dB}}/\alpha$$

where ξ is defined as the ratio between the standard deviation of radio signal power fluctuations $\sigma_{\psi'_{dB}}$ and the pathloss exponent α.

We compute the expected number of neighbors with path loss and shadowing. The path loss model with shadowing is given by [76, 112]:

$$f(\psi) = \frac{1}{\sqrt{2\pi}\sigma\psi}e^{-\frac{1}{2}\left(\frac{\log\psi + \log Kr^{-\alpha}}{\sigma}\right)^2} \tag{10.6}$$

The average number of neighbors for a node with path loss and shadowing as shown in Appendix A are:

$$E[N_{PL;S}] = \delta\pi\rho^2 e^{\left(\frac{\sqrt{2}\sigma}{\alpha}\right)^2} \tag{10.7}$$

Thus, the increase in average degree with shadowing depends on the ratio $\frac{\sigma}{\alpha}$. The increase in number of neighbors by a factor of $e^{\left(\frac{\sqrt{2}\sigma}{\alpha}\right)^2}$ with shadowing has also been

proved by Orriss and Barton [80]. We now calculate the number of neighbors with path loss and shadowing within the transmission range $\rho = \left(\frac{KP_t}{GP_{\min}}\right)^{\frac{1}{\alpha}}$ due to path loss alone. The average number of neighbors within the transmission radius ρ due to path loss alone is (shown in the Appendix):

$$E\left[N_{PL;S}\left(\rho\right)\right] = \frac{\pi\delta\rho^2}{2} + \pi\rho^2\delta\frac{1}{2}e^{\left(\frac{\sqrt{2}\sigma}{\alpha}\right)^2}\operatorname{erfc}\left(\frac{\sqrt{2}\sigma}{\alpha}\right) \tag{10.8}$$

The expected number of neighbors within radius r has been computed in [53],

$$E\left[N_{PL;S}\left(r\right)\right] = \delta\pi r^2\left(f\left(\mathfrak{a}\right) + e^{\left(\frac{2-2ab}{b^2}\right)}f\left(\frac{2-2\mathfrak{a}\mathfrak{b}}{\mathfrak{b}}\right)\right) \tag{10.9}$$

where $\mathfrak{a} = \frac{P_{\min}-P_{rx}(\rho)}{\sigma_{\psi'_{dB}}}$, $\mathfrak{b} = \frac{10\alpha\log_{10}e}{\sigma_{\psi'_{dB}}}$ and $P_{rx}\left(r\right) = P_{tx} + 10\log_{10}K - 10\alpha\log_{10}\left(\frac{r}{r_0}\right)$. $f\left(x\right)$ can be related to the complementary error function as $f\left(x\right) = \frac{1}{2}\operatorname{erfc}\left(\frac{x}{\sqrt{2}}\right)$. Note that (10.8) and (10.9) are equivalent when $\mathfrak{a} = 0$. This shows that the derivation of (10.7) in recent literature is related to the well known formula for coverage in cellular networks.

The expected number of neighbors with path loss, shadowing and Rayleigh fading is [76]:

$$E\left[N_{PL;S;F}\right] = \delta\pi\left(\frac{2}{\alpha}\right)\Gamma\left(\frac{2}{\alpha}\right)\rho^2 e^{\left(\frac{\sqrt{2}\sigma}{\alpha}\right)^2} \tag{10.10}$$

where $\Gamma\left(\cdot\right)$ represents the gamma function. Since $\alpha \geq 2$, fading reduces the average number of neighbors by a factor of $\left(\frac{2}{\alpha}\right)\Gamma\left(\frac{2}{\alpha}\right)$.

10.3 Shortest Path Routing and Load Balancing

The number of shortest paths passing through a node (traffic load) can be calculated by making the following assumptions for calculating the traffic load on nodes. Let us assume that nodes are uniformly distributed with density δ in a circular area of radius $\mathcal{R}$ (δ is large). Moreover, we assume that the nodes communicate with uniform rate λ. In shortest path routing based on distance, an approximation to the load on a node at a distance $\mathfrak{d}$ from the center is [84]:

$$Lo\left(\mathfrak{d}\right) = \left(\pi\mathcal{R}^2\delta - 1\right)\lambda + \frac{\pi\left(\mathcal{R}^2 - \mathfrak{d}^2\right)^2\delta^2\lambda\sigma}{2} \tag{10.11}$$

Note that in 10.11 the traffic load on a node is independent of transmission range of nodes and only depends on distance of node from the center. When the node density

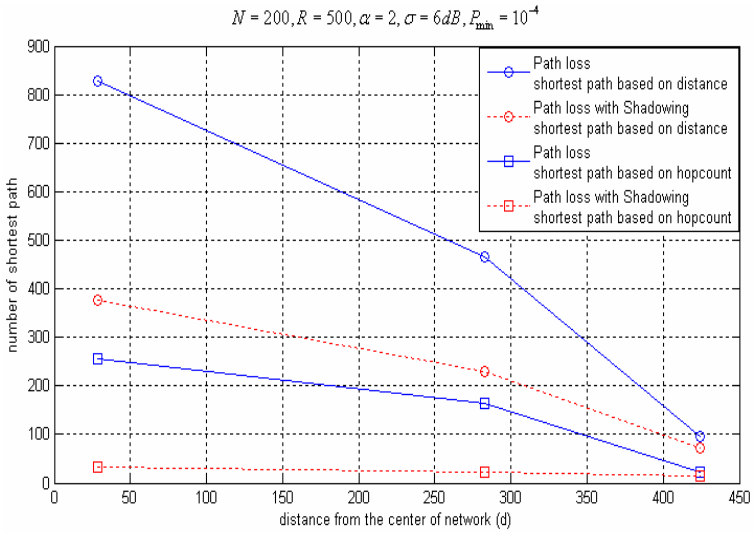


Figure 10.1: Traffic load on a node as a function of distance $\mathfrak{d}$ of the node from the center of the network ($G = 0.03mW$, $K = 7.01 \times 10^{-4}$, $P_{tx} = 100mW$, $P_{\min} = 10^{-4}mW$).

is low, equation 10.11 is not accurate. Also, 10.11 gives the load based on all source-destination pairs. The actual load on a node depends on number of source-destination pairs.

Figure 10.1 shows the simulation results for traffic load in a low density network with different path loss models and shortest path routing based on hopcount and distance metrics. The nodes ($N = 200$) are uniformly distributed in a circular area with radius $R = 500m$. The simulations indicate that shortest path routing based on hopcount leads to much better load balancing than shortest path routing based on distance. Moreover, due to long range contacts in shadowing, the traffic is distributed much more evenly in combined path loss and shadowing as compared to path loss alone.

Additional simulations show that shortest path routing based on distance metric performs similar to the shortest path routing based on hopcount in terms of expected hopcount and better in terms of expected distance.

10.4 Lifetime of Ad hoc Wireless Network

Network lifetime is usually defined as the time for first node to die, or as the time for a certain percentage of nodes to die [12, 111]. We need to know the energy consumed by wireless nodes for data transmission to determine the lifetime of ad hoc networks. The energy consumed by a wireless node has two components [91]. The receiving or processing energy is the energy consumed by the node in standby mode or when

receiving data. The transmission energy is the energy needed to transmit data over the radio link. The energy spent in transmitting and receiving a $j-$bit message is given by [91]:

$$E_T = E_{elec} * j + E_{amp} * j * r^{\alpha} \tag{10.12}$$
$$E_R = E_{elec} * j$$

where E_{elec} and E_{amp} are constants, r is the distance over which the message is transmitted and α is the path loss exponent.

With the same assumptions used to compute (10.11), the maximum energy spent by a node at distance $\mathfrak{d}$ from the center till time t is

$$E_{spent}(\mathfrak{d}) = \mathcal{E} * \left(\left(\pi \mathcal{R}^2 \delta - 1\right) \lambda + \frac{\pi \left(\mathcal{R}^2 - \mathfrak{d}^2\right)^2 \delta^2 \lambda \sigma}{2} \right) * t \tag{10.13}$$

where $\mathcal{E} = E_T + E_R = 2 * E_{elec} + E_{amp} * r^{\alpha}$. r is the transmission range of each node and for the path loss model $r = \rho$. The energy spent is maximum since we have assumed that each node transmits the maximum transmission range to transmit a packet to any neighbor within the transmission range.

Thus, the lifetime (time when first node fails) for a network when all nodes start with equal battery power E_0 is

$$\begin{aligned} t_{\text{lifetime}} &= \frac{E_0}{\mathcal{E}\left(\left(\pi \mathcal{R}^2 \delta - 1\right) \lambda + \frac{\pi (\mathcal{R}^2 - \mathfrak{d}^2)^2 \delta^2 \lambda \sigma}{2}\right)} \\ &= \frac{E_0}{\mathcal{E}\left(\left(\pi \mathcal{R}^2 \delta - 1\right) \lambda + \frac{\pi \mathcal{R}^4 \delta^2 \lambda \sigma}{2}\right)} \\ &= \frac{E_0}{\lambda \left(2 * E_{elec} + E_{amp} * r^2\right) \left((N-1) + \frac{N^2 \sigma}{2\pi}\right)} \\ &\cong \frac{c}{\lambda r^2 N^2} \end{aligned} \tag{10.14}$$

where $\alpha = 2$ and c is a constant. Using (10.14), it may be concluded that the lifetime of a network increases if the transmission range is reduced.

In the simulations to study lifetime of ad hoc networks, we assume that $E_o = 1J$, $E_{elec} = 50nJ/bit$, $E_{amp} = 100pJ/bit/\text{m}^2$ and $\alpha = 2$. We assume that the nodes ($N = 200$) are uniformly distributed in circular area with radius $R = 500$ m and the initial transmission radius (r) due to path loss alone is 200 m [41]. The size of packet for shortest path routing is assumed to be 1250 bytes and for searching with flooding and random walks the packet size is 125 bytes. We assume that in each time interval, a pair of nodes chosen uniformly exchange a single packet. Lifetime is chosen as the time when the first node dies [41].

Power control algorithms could be employed to extend the lifetime of ad hoc networks [12, 61]. We show the effect of adaptive power on the lifetime of network. We assume that each node tries to conserve energy by reducing its transmission power independently of other nodes i.e., the transmission radius is reduced by each node.

Figure 10.2 shows the lifetime of ad hoc network with shortest path routing based on hopcount and distance for a low density network. The simulations show that even though shortest path routing based on hopcount performs better load balancing, yet the lifetime of network is less than shortest path routing. This can be attributed to the fact that nodes need to transmit over larger distance using shortest path routing based on hopcount and thus require more transmission energy (10.12). These results show the pitfalls in modelling ad hoc networks. Seemingly contradictory results can be obtained by different assumptions and neglecting some of the parameters.

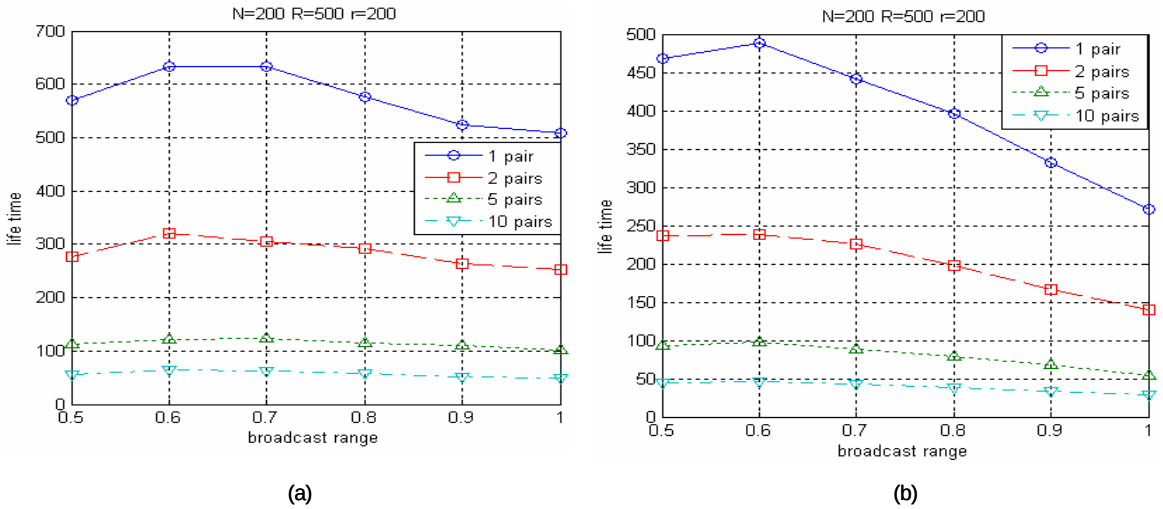


Figure 10.2: Lifetime of an ad hoc network (time for first node to die) with shortest path routing based on (a) distance and (b) hopcount for different transmission range.

Figure 10.2 also shows that by reducing the transmission power the lifetime of the network does not increase in contradiction to (10.14). This is attributed to the fact that though the decrease in transmission power reduces the energy consumption of a node, the number of paths passing through the node increases i.e., the expected hopcount for shortest path increases. Further analysis needs to be conducted to study the impact of adaptive power algorithms on the lifetime of networks.

Figure 10.3 shows the lifetime of ad hoc networks for flooding with sequence numbers and random walks for a low density network. The simulations show that overhead with flooding is much less than with random walks. Moreover, by reducing the transmission power the lifetime increases for flooding with sequence numbers. This is because each

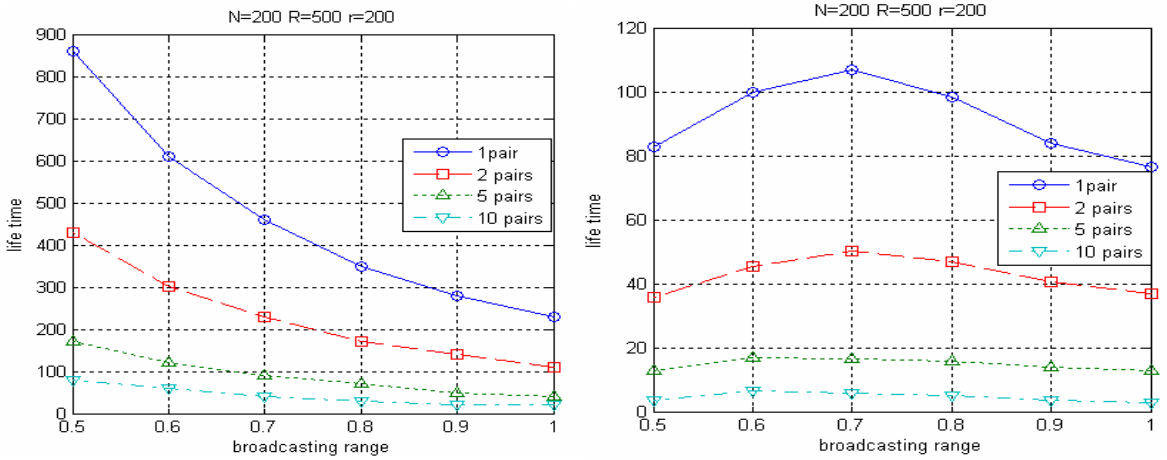


Figure 10.3: Lifetime of ad hoc network (time for first node to die) with searching by (a) flooding and (b) random walks for different number of source-destination pairs and different transmission range of each node.

node transmits the packet only once and with lower transmission power it spends less battery power. In random walks, with decrease in transmission power of nodes, the lifetime varies.

10.5 Conclusions

In this chapter, we analyzed the effect of different signal propagation models on the topology of ad hoc wireless networks. Shadowing leads to better load balancing as compared to path loss alone. The load balancing is much better for shortest path routing based on hopcount as compared to shortest path routing based on distance. However, the lifetime is worse with shortest path routing based on hopcount. The lifetime of ad hoc networks is not affected if all the nodes decrease their transmission power. Moreover, the lifetime analysis of ad hoc networks shows that overhead of searching with random walks is more than flooding.

Modelling of topology features of ad hoc networks such as degree distribution, connectivity, etc. has received extensive attention in the literature. However, the results have been obtained for limited scenarios and assuming that the nodes are static. Moreover, contradictory results can be obtained based on assumptions and assumed parameters. The increase in average node degree with shadowing but neglecting interference is one such example. With interference the probability of nodes transmitting at larger distance will decrease because of the larger interference. Thus, more analysis needs to

be done to study the effect of topology on lifetime of ad hoc networks.

Chapter 11

Estimation of Topology

11.1 Introduction

The development of topology estimation algorithms is the first step to build *intelligent*[1] topology based algorithms. In ad-hoc and self-configuring networks because of the dynamic topology and limited bandwidth, external measurement tools are harder to employ for topology estimation. Instead, techniques from statistics may be employed to estimate the topology of ad hoc networks on the fly. Once the topology of ad hoc networks is estimated, a number of parameters about the network such as connectivity, expected hopcount for shortest path, load on nodes, TTL for searching etc. can be obtained. These network parameters could be used for optimizing a wide variety of algorithms.

We know that the topology of ad hoc wireless networks can be modelled as a geometric random graphs with nodes the having Binomial degree distribution:

$$\Pr\left[\deg = j\right] = \binom{N-1}{j} p^j \left(1-p\right)^{N-1-j} \tag{11.1}$$

In this chapter, we consider the problem of estimation of topology of an ad-hoc network, which involves estimating the values of p and N, given the degree of m nodes. The estimation of parameters N and p for a Binomial distribution has been discussed extensively in the literature [35, 69, 79]. A particularly hard problem is to estimate the values of N and p when both are unknown and the sample size m is much smaller than N. In this case, we apply some recently proposed estimates for both N and p.

[1] An example is flooding with probability q, the probability q could be optimized based on the topology of the network.

11.2 Topology Estimation

A variety of estimators can be used in estimation of N or p when one of these parameters is known. Let $X_1, X_2, ..., X_m$ be the sample representing the degree of m nodes. The likelihood function and log-likelihood function of the Binomial distribution is

$$\mathfrak{L}(N-1,p) = \prod_{i=1}^{m} \binom{N-1}{X_i} p^{X_i} (1-p)^{N-1-X_i} \tag{11.2}$$

$$\mathcal{L}(N-1,p) = \sum_{i=1}^{m} \log \frac{(N-1)!}{X_i!\,(N-1-X_i)!} + \sum_{i=1}^{m} X_i \log \frac{p}{1-p} + m\,(N-1)\log(1-p) \tag{11.3}$$

The maximum-likelihood estimator (MLE) is obtained by differentiating the likelihood function $\mathcal{L}'(N-1,p) = \frac{\partial \mathcal{L}}{\partial p} = 0$, such that p_{MLE} is

$$p_{MLE} = \frac{\sum_{i=1}^{m} X_i}{m\,(N-1)} = \frac{\overline{X}}{N-1} \tag{11.4}$$

Thus, when N is known, the link density p can be estimated using (11.4).

The method of moments estimate of p can be obtained since the sample mean $\overline{X} = (N-1)\,p$ and sample variance $\hat{S}^2 = (N-1)\,pq$, where $q = 1-p$, hence the estimate of p is

$$\widehat{p} = 1 - \frac{\hat{S}^2}{\overline{X}} \tag{11.5}$$

The estimated value of N using method of moments is $\widehat{N}_{ME} = \frac{\overline{X}^2}{\overline{X}-\hat{S}^2} + 1$.

11.2.1 Estimation when both p and N are unknown

The simplest estimator of the value of N with m i.i.d. samples $X_1, X_2, ..., X_m$ is the sample maximum $X_{(m)} \to N$ and hence $X_{(m)} = N\,(a.s.)$ for all large m. This estimate works well even at low values of m when the link density p is large. However, it is well known that the sample maximum is not a reliable estimate of the binomial N parameter and results in severe underestimation of N, especially if either true N is large, or the value of p is small.

We first study the smallest value of m such that $\Pr\left(X_{(m)} \geq \mathfrak{m}\right) \geq 1-\theta$ for a given θ. The number m grows exponentially as shown by the following theorem by Dasgupta and Rubin [35].

Theorem 7 *: Let $\mathfrak{m}, N \to \infty$ such that $\frac{N}{\mathfrak{m}}$ is an integer, and $\chi = \mathfrak{m}/N > p$. Let $0 < \theta < 1$. Then the smallest value of m such that $\Pr\left(X_{(m)} \geq \mathfrak{m}\right) \geq 1-\theta$ is*

$$m \geq \frac{(-\log\theta)(\chi - p)}{p\sqrt{1-\chi}} \left(\frac{\chi (1-\chi)^{\frac{1}{\chi}-1}}{pq^{\frac{1}{\chi}-1}} \right)^{\mathfrak{m}} \sqrt{2\mathfrak{m}\pi} (1 + o(1))$$

We now study some of the estimators proposed for N and p when both are unknown. The method of moment estimate for p and N can be obtained by using (11.5). However, when the value of p is small i.e., the sample mean and variance are close to each other, the moments estimate is not stable. A moment-stabilized estimator has been proposed by Olkin *et al.* [79],

$$\widehat{N}_{MES} = \max\left\{ \frac{\hat{S}^2\phi^2}{\phi - 1}, X_{(m)} \right\} \tag{11.6}$$

where

$$\phi = \begin{cases} \frac{\overline{X}}{\hat{S}^2} & \text{if } \frac{\overline{X}}{\hat{S}^2} \geq 1 + 1/\sqrt{2} \\ \max\left\{ \frac{X_{(m)} - \overline{X}}{\hat{S}^2}, 1 + \sqrt{2} \right\} & \text{if } \frac{\overline{X}}{\hat{S}^2} < 1 + 1/\sqrt{2} \end{cases}$$

Another estimator that could be used is the stabilized MLE. The likelihood function described in (11.2) can be maximized when $N \to \infty$, $p \to 0$ and $p(N-1) = \overline{X}$. This limiting case of likelihood is denoted by $\mathfrak{L}(\infty, 0)$. The following theorem by Levin [69] gives the estimates.

Theorem 8 *:An MLE of N exists. If $\overline{X}/\hat{S}^2 \leq 1$, $\mathfrak{L}(N,p) < \mathfrak{L}(\infty,0)$, while if $\overline{X}/\hat{S}^2 > 1$, $\mathfrak{L}(N,p)$ is maximized by atleast one pair (N,p), with $X_{(m)} \leq N < \infty$.*

Substituting the value of p_{MLE} from 11.4 in 11.3,

$$\begin{aligned} \mathcal{L}(N-1,p) &= \sum_{i=1}^{m} \log \frac{(N-1)!}{X_i!(N-1-X_i)!} + \sum_{i=1}^{m} X_i \log \frac{\overline{X}}{N-1-\overline{X}} \\ &\quad + m(N-1)\log\left(1 - \frac{\overline{X}}{N-1}\right) \end{aligned}$$

Let us denote the differential,

$$\mathfrak{D}N = \frac{\partial \mathcal{L}}{\partial N}$$

If $\mathfrak{D}(0) \leq 0$, MLE $= X_{(m)}$ and if $\mathfrak{D}(0) > 0$ then the MLE is taken as solution of $\mathfrak{D}\left(\text{MLE} - X_{(m)}\right) = 0$. Thus, the stabilized MLE is given as

$$\widehat{N}_{MES} = \begin{cases} \text{MLE} & \text{if } \frac{\overline{X}}{\hat{S}^2} \geq 1 + 1/\sqrt{2} \\ X_{(m)} + \left(\frac{m-1}{m}\right)\left(X_{(m)} - X_{(m-1)}\right) & \text{if } \frac{\overline{X}}{\hat{S}^2} < 1 + 1/\sqrt{2} \end{cases}$$

Another two estimates for N when p and N are unknown have been proposed by Dasgupta and Rubin [35]. Consider the identity $N = \frac{N^{\mathfrak{r}+1}(Npq)^{\mathfrak{r}}}{(Np)^{\mathfrak{r}}(Nq)^{\mathfrak{r}}}$. Substituting the sample maximum $X_{(m)}$ for N, the sample variance $\hat{S}^2$ for Npq, the sample mean $\overline{X}$ for Np and $X_{(m)} - \overline{X}$ for Nq, we obtain the estimate for N

$$\widehat{N}_1 = \frac{X_{(m)}^{\mathfrak{r}+1}\left(\hat{S}^2\right)^{\mathfrak{r}}}{\left(\overline{X}\right)^{\mathfrak{r}}\left(X_{(m)} - \overline{X}\right)^{\mathfrak{r}}} \tag{11.7}$$

The second estimator proposed in [35] is bias corrected,

$$\widehat{N}_2 = X_{(m)} + \sum_{i=0}^{[\widehat{N}_1]-2} F^{-1}_{i+1,[\widehat{N}_1]-i}\left(\frac{1}{m}\right) \tag{11.8}$$

where $F^{-1}_{\mathbf{r},\mathbf{s}}$ denote the quantile function of the Beta$(\mathbf{r},\mathbf{s})$ distribution. The quantile function is defined as

$$F^{-1}(z) = \min\left\{x \in \mathbb{R}, z \leq F(x)\right\},\ z \in (0,1)$$

and the probability density function of a beta distribution for any $\mathfrak{i} > 0, \mathfrak{j} > 0$

$$f(x) = \frac{1}{\mathfrak{B}(\mathfrak{i},\mathfrak{j})} x^{\mathfrak{i}-1}(1-x)^{\mathfrak{j}-1}, 0 < x < 1$$

where $\mathfrak{B}(\mathfrak{i},\mathfrak{j}) = \frac{(\mathfrak{i}-1)!(\mathfrak{j}-1)!}{(\mathfrak{i}+\mathfrak{j}-1)!}$ when $\mathfrak{i}$ and $\mathfrak{j}$ are integers, is the beta function.

11.2.2 Results

Table 11.1 show the estimated values of N using different estimators. The average values are taken for 1000 iterations. In calculating the average value of $\widehat{N}_{ME}$, we have neglected the case where $\widehat{p}_{ME} \leq 0$. This makes the results look better than expected since only the stable values of $\widehat{N}_{ME}$ are used while in $\widehat{N}_{MES}$ even negative values lead to estimation of N. For small p, the estimator $\widehat{N}_1$ is the best while for large values of p, the estimator $\widehat{N}_2, \widehat{N}_{MES}$ and $\widehat{N}_{MLES}$ seem to perform good.

At small values of p, the estimators do not perform well. There seems to be underestimation of N when p is small in these methods. In the next section, we look into estimation of N when a subgraph is known.

11.2.3 A subgraph and average degree are known

Let $X_1, X_2, ..., X_m$ be the sample representing the degree of m nodes. The mean degree obtained from this sample is

$$\overline{X} = \frac{1}{m}\sum_{i=1}^{m} X_i = \widehat{p}\left(\widehat{N} - 1\right) \tag{11.9}$$

Table 11.1: Estimated value of N using different estimators.

N	p	m	$X_{(m)}$	$\widehat{N}_1$	$\widehat{N}_2$	$\widehat{N}_{ME}$	$\widehat{N}_{MES}$	$\widehat{N}_{MLES}$
200	0.05	25	24	38	30	219	37	18
200	0.1	25	39	82	62	431	71	33
200	0.3	25	84	278	199	347	165	93
200	0.7	25	164	531	400	199	198	183

Table 11.2: Estimation of N when a subgraph is known.

N	p	m	Z	$\hat{N}$	mse
400	0.05	25	40	411	5048
400	0.1	25	40	405	2176
400	0.4	25	40	401	349
400	0.8	25	40	400	58
1600	0.05	25	40	1651	79426
1600	0.1	25	40	1621	33509
1600	0.8	25	40	1602	886

Let us assume that a sub-graph having Z nodes ($Z < N$) is known. Let $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_{\mathfrak{t}}$ be the sample representing the degree of $\mathfrak{t}$ nodes within this sub-graph. The mean degree obtained from this sample is

$$\overline{\mathcal{X}} = \frac{1}{\mathfrak{t}} \sum_{i=1}^{\mathfrak{t}} \mathcal{X}_i = \widehat{p}\,(Z-1) \tag{11.10}$$

Thus using 11.9 and 11.10, the estimated value of N is:

$$\widehat{N} = \frac{\overline{X}}{\overline{\mathcal{X}}}\,(Z-1) + 1 \tag{11.11}$$

Table 11.2 shows the results for estimating N using 11.11. We assume that $\mathfrak{t} = Z$.

11.2.4 Influence of m and Z

We want to find the effect of m and Z on the estimated the value of N. When m is large, the error in estimation of $\overline{X}$ becomes negligible. Hence, the error is only introduced due to $\overline{Y}$ and when Z is large, the error would become small. Thus, large values of m

and Z give the best estimate for N. However, the estimate of N does not improve as m is increased keeping Z constant (11.1).

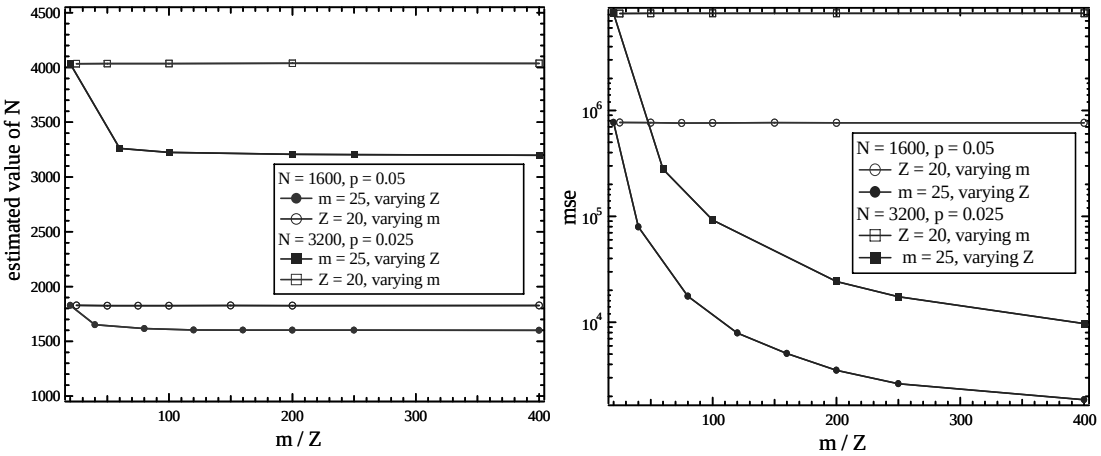


Figure 11.1: (a) Estimated value of N and (b) mean square error as a function of m and Z.

11.3 Conclusions

We presented a methodology for topology estimation given the degree of certain nodes in a random graph. It is difficult to estimate the values of N and p when both are unknown and the sample size m is much smaller than N. Results show that at small values of link density p, most of the estimators do not perform well. Further analysis of topology estimation needs to be carried out in order to develop topology dependant algorithms. Indeed, topology estimation is the first step for building such topology based algorithms.

Chapter 12

Conclusions

The emergence of ad hoc wireless networks and P2P networks has lead to an increasing interest in adaptive and decentralized algorithms for routing, resource and service discovery etc. A variety of applications are currently being developed based on these novel algorithms and protocols. For example, flooding and its variations had been the *de facto* standard for searching in different networks. However, new versions of P2P networks such as Gnutella and Gia use random walk strategies instead of flooding.

This thesis focused on three interrelated research topics for ad hoc networks:

- Ant Routing.
- Searching with single and multiple queries.
- Topology of ad hoc wireless networks.

In this chapter, we provide an overview of the results for each of these topics and the major contributions of this thesis. We include remarks for extension of work whenever possible.

Ant Routing

The first contribution of this thesis is in analyzing the performance of ant routing. Based on the analysis in this thesis, we are able to answer questions such as: How do different parameters affect the performance of ANTRALs and how does ant routing perform for ad hoc wireless networks where mobility and limited bandwidth play an important role. Extensive simulations of the AntNet algorithm show that ant routing algorithms are able to find paths that are close to the shortest path. We can state that

The performance of ant routing is close to shortest path routing for small sparse graphs.

As the network size increases and density of network increases, the AntNet algorithm is no longer able to find paths that are close to the shortest path. The performance of AntNet algorithm depends on parameters such as $\eta, W_{\max}$ and a. The optimal value of parameters such as η and $W_{\max}$ could be chosen depending on the size of network, while the optimal value of parameter a depends on the degree of nodes. Thus,

The performance and optimization of parameters in ant routing algorithms depend on the topology of the network.

Ant routing algorithms provide an automatic adaptation to link failure and traffic changes. However, the rate of topology change in mobile ad hoc wireless networks is large. In order to overcome this problem, on-demand protocols have been proposed which search for a path to the destination only when the nodes need to transfer data. We compared the performance of ant routing with on-demand protocols AODV and DSR using NS-2. Our implementation includes realistic features for ad hoc networks such as MAC layer, limited buffer size at nodes, node mobility etc. The analysis shows that

Overhead of ant routing is more than on-demand protocols such as AODV and DSR in ad hoc wireless networks.

Our work on ant routing could be extended further in new directions. The scalability of ant routing algorithms needs to be investigated further. Moreover, modifications to these algorithms such as using both data and ant packets for updating the routing tables could be investigated. Specifically, the AntNet algorithm would be considered a truly distributed and scalable algorithm if the nodes are able to learn and adjust the various parameters used in the algorithm.

Searching

The main contribution of section 3 is the definition and classification of different searching schemes. The searching strategies may be probabilistic and based on random walks or may use unique feature of the network topology such as high degree nodes. These strategies have been studied in bits and pieces in literature but a complete view of different searching schemes was missing. In chapters 7, 8 and 9 of this thesis, we have analyzed and studied the performance of searching schemes based on single or multiple queries. We distinguished between deadlocks and loops and examined the affect of deadlocks and loops on the performance of search strategies. An exact analysis for some of these strategies in Chapter 8 is another contribution of this thesis.

The probability distribution of hopcount for a random walk on a complete graph K_N is a geometric distribution while the probability distribution for a random walk with memory M is a uniform random variable on $[1, M]$ and a geometric variable on $[M + 1, N - 1]$. In many random walk strategies such as random walk using highest

degree, the expected hopcount is small but the probability of deadlocks is large. Indeed, we need to look into probability of deadlocks as well as the expected hopcount to determine the performance of search strategies.

The expected hopcount or expected weight of the path is not sufficient to characterize searching based on different random walk strategies.

The performance of different search strategies depends on the graph topology. Because of high degree nodes in a BA graph, the performance of search strategies using highest degree in BA graphs is better than in ER random graphs. In both these graph topologies, the performance of random walk with look-ahead is close to shortest path routing with small values of look-ahead. This is due to the small diameter $O(\log N)$ of these graph topologies.

With lookahead of $\log N$*, the performance of searching with* RW_{LA} *is close to that of shortest path routing.*

It can be concluded that more information a search strategy has, the better is its performance. For example, random walk using highest degree, look-ahead and memory performs better than random walk using look-ahead and memory.

A single RW query has an overhead and time to discover of $O(N \log N)$ for ER random graphs and power law graphs. To reduce the time to discovery, multiple random walk queries with smaller TTL may be used. The overhead of searching with multiple queries depends on TTL and the number of queries which in turn can be optimized by the topology of the network.

The overhead for searching with multiple random walk queries is less than for flooding.

However, in BA graphs, the improvement of searching with multiple random walk queries as compared to flooding is limited. With multiple copies of files in P2P networks, searching with random walk queries would be much more efficient than searching with flooding. Searching techniques proposed and analyzed in this thesis could be used with traditional searching methods such as flooding. For example, critical information could be spread by flooding while less critical information could be spread by random walks queries.

Thus, section 3 of this thesis provides answers for questions such as: How does memory and look-ahead affect the performance of random walk search strategies and what role the graph topology plays in performance of different search strategies.

Topology of ad hoc wireless networks

The first two sections of this thesis show that the topology of networks is one of the factors that determines the performance of different algorithms. The major achievement

of this section is understanding the topology of ad hoc wireless networks and the affect of topology on shortest path routing, lifetime etc.

In section 4, we investigated the effect of signal propagation models on the topology of ad hoc wireless networks. Increasingly, it is being realized that simple path loss model is not sufficient to characterize the topology of ad hoc wireless networks. Modeling of wireless networks with shadowing and fading shows that shadowing improves the connectivity while fading reduces the connectivity of these networks. Specifically, the change in average node degree with these models depends on the standard deviation of radio signal power fluctuations σ and the pathloss exponent α.

Shadowing increases degree of nodes in ad hoc wireless networks by a factor of $e^{\left(\frac{\sqrt{2}\sigma}{\alpha}\right)^2}$

We proved that the average number of neighbors of a node with shadowing is related to the well known formula for coverage in cellular networks. The appearance of long range links with shadowing increases the number of neighbors and distributes data traffic more effectively.

Shadowing improves the load balancing in ad hoc wireless networks.

A novel idea in Chapter 11 was the use of techniques from statistics to estimate the topology of ad hoc wireless networks. We used estimators such as method of moments and maximum likelihood to estimate N and p for random graphs. This analysis could be extended further to build topology dependant algorithms.

Appendix A

Average number of neighbors

The path loss model with shadowing is given by

$$f(\psi) = \frac{1}{\sqrt{2\pi}\sigma\psi} e^{-\frac{1}{2}\left(\frac{\log\psi + \log Kr^{-\alpha}}{\sigma}\right)^2}$$

The area (A) where received power P_r is greater than $P_{\min}$ is

$$A = \int_{r=0}^{\infty} 2\pi rp(\psi < \frac{P_t}{P_{\min}})dr = 2\pi \int_{r=0}^{\infty} rdr \int_{\psi=0}^{\frac{P_t}{P_{\min}}} \frac{1}{\sqrt{2\pi}\sigma\psi} e^{-\frac{1}{2}\left(\frac{\log\psi + \log Kr^{-\alpha}}{\sigma}\right)^2} d\psi$$

After substituting $x = \frac{\log\psi + \log Kr^{-\alpha}}{\sigma} = \frac{\log\psi Kr^{-\alpha}}{\sigma}$, we obtain

$$\begin{aligned}
&= 2\pi \int_{r=0}^{\infty} rdr \int_{x=-\infty}^{\frac{\log\left(\frac{P_t Kr^{-\alpha}}{P_{\min}}\right)}{\sigma}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\
&= 2\pi \int_{x=-\infty}^{\infty} dx \int_{r=0}^{\left(\frac{e^{-\sigma x} KP_t}{P_{\min}}\right)^{\frac{1}{\alpha}}} rdr \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} = 2\pi \int_{x=-\infty}^{\infty} \frac{1}{2} \left(\frac{e^{-\sigma x} KP_t}{P_{\min}}\right)^{\frac{2}{\alpha}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\
&= \pi \left(\frac{KP_t}{P_{\min}}\right)^{\frac{2}{\alpha}} \int_{x=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} e^{-\frac{2\sigma x}{\alpha}} dx = \pi \left(\frac{KP_t}{P_{\min}}\right)^{\frac{2}{\alpha}} e^{\left(\frac{\sqrt{2}\sigma}{\alpha}\right)^2}
\end{aligned}$$

Thus, the average number of neighbors for a node with path loss and shadowing are $\delta A = \pi\delta \left(\frac{KP_{tx}}{P_{\min}}\right)^{\frac{2}{\alpha}} e^{\left(\frac{\sqrt{2}\sigma}{\alpha}\right)^2}$.

We now calculate the number of neighbors within the transmission range $\rho = \left(\frac{KP_{tx}}{WP_{\min}}\right)^{\frac{1}{\alpha}}$ due to path loss alone.

$$\begin{aligned} A &= \int_{r=0}^{\rho} 2\pi r p(\psi < \frac{P_t}{P_{\min}}) dr \\ &= 2\pi \int_{r=0}^{\rho} r dr \int_{\psi=0}^{\frac{P_t}{P_{\min}}} \frac{1}{\sqrt{2\pi}\sigma\psi} e^{-\frac{1}{2}\left(\frac{\log\psi + \log Kr^{-\alpha}}{\sigma}\right)^2} d\psi \end{aligned}$$

After substituting $x = \frac{\log\psi + \log Kr^{-\alpha}}{\sigma} = \frac{\log \psi Kr^{-\alpha}}{\sigma}$, we obtain

$$\begin{aligned} &= 2\pi \int_{r=0}^{\rho} r dr \int_{x=-\infty}^{\frac{\log\left(\frac{P_t Kr^{-\alpha}}{P_{\min}}\right)}{\sigma}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\ &= 2\pi \int_{r=0}^{\rho} r dr \int_{x=-\infty}^{0} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx + 2\pi \int_{r=0}^{\rho} r dr \int_{x=0}^{\frac{\log\left(\frac{P_t Kr^{-\alpha}}{P_{\min}}\right)}{\sigma}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\ &= \frac{\pi\rho^2}{2} + 2\pi \int_{r=0}^{\rho} r dr \int_{x=0}^{\frac{\log\left(\frac{P_t Kr^{-\alpha}}{P_{\min}}\right)}{\sigma}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\ &= \frac{\pi\rho^2}{2} + 2\pi \int_{x=\frac{\log\left(\frac{P_t K\rho^{-\alpha}}{P_{\min}}\right)}{\sigma}}^{\infty} dx \int_{r=0}^{\left(\frac{e^{-\sigma x} KP_t}{P_{\min}}\right)^{\frac{1}{\alpha}}} r dr \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\ &= \frac{\pi\rho^2}{2} + \pi\rho^2 \int_{x=\frac{\log\left(\frac{P_t K\rho^{-\alpha}}{P_{\min}}\right)}{\sigma}}^{\infty} dx e^{-\frac{2\sigma x}{\alpha}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\ &= \frac{\pi\rho^2}{2} + \pi\rho^2 \int_{x=0}^{\infty} dx e^{-\frac{2\sigma x}{\alpha}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\ &= \frac{\pi\rho^2}{2} + \pi\rho^2 \frac{1}{2} e^{\left(\frac{\sqrt{2}\sigma}{\alpha}\right)^2} \operatorname{erfc}\left(\frac{\sqrt{2}\sigma}{\alpha}\right) \end{aligned}$$

Thus, the average number of neighbors for a node within the transmission radius due to path loss alone (ρ) are $\delta A = \frac{\pi\rho^2}{2} + \pi\rho^2\frac{1}{2}e^{\left(\frac{\sqrt{2}\sigma}{\alpha}\right)^2}\,\mathrm{erfc}\left(\frac{\sqrt{2}\sigma}{\alpha}\right)$.

Abbreviations

ANTRAL	Ant routing algorithm
AODV	Ad hoc on-demand distance vector
DS	Direct sequence
DSDV	Destination-sequenced distance vector
DSR	Dynamic source routing
DSSS	Direct sequence spread spectrum
DHT	Distributed hash table
FDMA	Frequency Division Multiple Access
FH	Frequency Hopping
FTP	File transfer protocol
HIPERLAN	High performance radio local area network
ITU	International Telecommunication Union
LAN	Local area network
MAC	Medium access control
MLE	Maximum-likelihood estimator
MTSO	Mobile Telephone Switching Office
OSI	Open Systems Interconnection
PAN	Personal area network
PN	Personal Network
QoS	Quality of service
RIP	Routing Information Protocol
RW	Random Walk
TDMA	Time Division Multiple Access
WCDMA	Wideband Code Division Multiple Access
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WSN	Wireless Sensor Network
WWW	World Wide Web

Bibliography

[1] M. Abramowitz, I.A. Stegun (Eds.). *A Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*. Dover Publications, New York, 1974, ISBN0486612724, June.

[2] L. A. Adamic, R. M. Lukose, A. R. Puniyani and B. A. Huberman. Search in power-law networks. *Physical Review E*, vol. 64, 2001.

[3] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer Networks*, 47, pp. 445–487, 2005.

[4] R. Albert and A. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74, 2002.

[5] C. Avin and G. Ercal. On the cover time of random geometric graphs. *Lecture notes in computer science ISSN 0302-9743*, 2005.

[6] A. Barabási, R. Albert and H. Jeong. Mean-field theory for scale-free random networks. *PhysicaA* 272, pp. 173-187, 1999.

[7] P. Baran. On distributed communication networks. *IEEE Transactions on Communications*. pp. 1-9, 1964.

[8] Z. Bar-Yossef, R. Friedman and G. Kliot. RaWMS- Random Walk based Lightweight Membership Service for Wireless Ad Hoc Networks. *Proc. MobiHoc*, pp. 238-249, 2006.

[9] N. Bean and A. Costa. An analytic modelling approach for network routing algorithms that use "ant-like" mobile agents. *Computer Networks* 49, pp. 243–268, 2005.

[10] R. Beckers, J.L. Deneubourg, S. Goss. Trails and U-turns in the selection of the shortest paths by the ant Lasius Niger. *Journal of Theoretical Biology* 159, pp. 397–415, 1992.

[11] C. Bettstetter. On the Connectivity of Ad Hoc Networks. *The Computer Journal*, vol. 47, no. 4, 2004.

[12] M. Bhardwaj and A. Chandrakasan. Bounding the Lifetime of Sensor Networks via Optimal Role Assignments. *Proc. INFOCOM*, pp. 1587-1596 , 2002.

[13] N. Bisnik and A. A. Abouzeid. Optimizing Random Walk Search Algorithms in P2P Networks. *Computer Networks*, 2006.

[14] B. Bollobás, *Random Graphs*, Cambridge University press, second edition, 2001.

[15] A. Borodin, Y. Rabani and B. Schieber. Deterministic Many-to-Many Hot Potato Routing. *Proc. IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 6, pp. 587-596, 1997.

[16] J. Broch, D.A. Maltz, D.B. Johnson,Y. Hu and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. *Proc. IEEE/ACM MOBICOM*, 47:445–487, 1998.

[17] C. Busch. $\tilde{O}$(Congestion + Dilation) Hot-Potato Routing on Leveled Networks. *Proc. SPAA*, 2002.

[18] D. Camara and A.A.F. Loureiro. Ants A novel routing algorithm for ad hoc networks. *Proc. of the 33rd Hawaii International Conference on System Sciences*, 2000.

[19] M. Castro, M. Costa and A. Rowstron. Peer-to-peer overlays: structured, unstructured or both?. *Technical Report MSR-TR-2004-73*, 2004.

[20] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham and S. Shenker. Making Gnutella-like P2P Systems Scalable. *Proc. SIGCOMM Conference*, 2003.

[21] C.C. Chiang, H.K Wu, W. Liu and M. Gerla. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. *Proc. IEEE Singapore International Conference on Networks*, pp. 197-211, 1997.

[22] F. Chung and L. Lu. The Diameter of Sparse Random Graphs. *Advances in Applied Math. 26*, pp. 257–279, 2001.

[23] F. Chung and L. Lu. The small world phenomenon in hybrid power law graphs. *Complex Networks, (Eds. E. Ben-Naim et. al.)*, Springer-Verlag, pp. 89–104, 2004.

[24] F. R. K. Chung. *Spectral Graph Theory*. CBMS Conference on Recent Advances in Spectral Graph Theory, 1994. ISBN: 0-8218-0315-8.

[25] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). *Internet Draft, RFC 3626*, 2003.

[26] T.J.M. Coenen, P.T.H. Goering, A. Jehangir, J.L. van den Berg, R.J. Boucherie, S.M. Heemstra de Groot, G.J. Heijenk, S.S. Dhillon, W. Lu, A. Lo, P.F.A Van Mieghem and I.G.M.M. Niemegeers. Architectural and QoS aspects of Personal Networks. *Proc. First International Workshop on Personalized Networks (PerNets 2006)* USA, 2006.

[27] C. Cooper and A. Frieze. Crawling on web graphs. *Proc. 34th annual ACM symposium on Theory of computing*, pp. 419 - 427, 2002.

[28] C. Cooper and A. Frieze. The cover time of the preferential attachment graph. *Journal of Combinatorial Theory, Series B 97*, pp. 269-290, 2007.

[29] T.H. Cormen, C.E. Leiserson and R.L. Rivest. *An introduction to Algorithms.* MIT Press, Boston, 2000.

[30] Andre Costa. Analytic Modelling of Agent-based Network Routing Algorithms. *Ph.D. thesis*, School of Applied Mathematics, University of Adelaide, 2003.

[31] S. Cui, R. Madan, A. J. Goldsmith and S. Lall. Cross-Layer Energy and Delay Optimization in small-scale sensor networks. *IEEE Transactions on Wireless Communication*, pp. 3688-3699, vol. 6, no. 10, 2007.

[32] D. Culler, D. Estrin and M. Srivastava. Overview of sensor networks. *Proc. IEEE Computer, Special Issue in Sensor Networks*, Aug 2004.

[33] D. J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes.* Springer, 1988.

[34] S.R. Das, C.E. Perkins and E. E. Royer. Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks. *Proc. INFOCOM*, pp. 3-12, 2000.

[35] A. DasGupta and H. Rubin. Estimation of binomial parameters when both n, p are unkown. *Journal of Statistical Planning and Interference* 130, pp. 391–404, 2005.

[36] J.-L. Deneubourg, S. Aron, S. Goss and J.-M. Pasteels. The self organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior* 3, pp. 159–168, 1990.

[37] S. S Dhillon and P. Van Mieghem. Performance Analysis of the AntNet algorithm. *Computer Networks*, Vol. 51, No. 8, pp. 2104-2125, 2007.

[38] S. S Dhillon and P. Van Mieghem. Comparison of Random Walk Strategies for Ad Hoc Networks. *Proc. of the Sixth Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2007), Corfu, Greece,* pp. 196-203, June 13-15, 2007.

[39] S. S Dhillon, X. Arbona and P. Van Mieghem. Ant Routing in Mobile Ad Hoc Networks. *Proc. Third International Conference on Networking and Services (ICNS'07), Athens, Greece*, June 19-25, 2007.

[40] S. S Dhillon and P. Van Mieghem. Searching with multiple random walk queries. *Proc. 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE PIMRC'07), Athens, Greece*, September 3-7, 2007.

[41] S. S Dhillon and Y. Zhou. Topology, shortest path routing and lifetime of ad hoc networks. *Proc. 14th Symposium on Communications and Vehicular Technology in the Benelux (SCVT 2007)*, Delft, Netherlands, 2007.

[42] G. Di Caro and M. Dorigo. AntNet: distributed stigmergetic control for communication networks. *Journal of Artificial Intelligence Research* 9, pp. 317–365, 1998.

[43] G. Di Caro and M. Dorigo. Two ant colony algorithms for best-effort routing in datagram networks. *Proc. 10th International Conference on Parallel and Distributed Computing and Systems*, 1998.

[44] S. Dolev, E. Schiller and J. Welch. Random Walk for Self-Stabilizing Group Communication in Ad-Hoc Networks. *Proc. 21st Symposium on Reliable Distributed Systems*, 2002.

[45] M. Dorigo and G. Di Caro. *The Ant Colony Optimization Meta-Heuristic.* McGraw-Hill Press, 1999.

[46] O. Dousse, F. Baccelli and P. Thiran. Impact of Interferences on Connectivity in Ad Hoc Networks. *IEEE Transactions on Networking*, vol. 13, pp. 425-436, 2005.

[47] F. Ducatelle, Gianni Di Caro, L.M. Gambardella. Ant agents for hybrid multipath routing in mobile ad hoc networks. *Proceedings 2nd Annual Conference on Wireless On demand Network Systems and Services (WONS)*, 2005.

[48] P. Erdös and A. Rényi. On random graphs I. *Publ. Math. Debrecen.* 6, pp. 290-297, 1959.

[49] M. Faloutsos, P. Faloutsos and C. Faloutsos. On power-law relationships of the Internet topology. *Proc. ACM SIGCOMM, Cambridge, Massachusetts*, 1999.

[50] P Ganesan and G S Manku. Optimal Routing in Chord. *Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 169-178, 2004.

[51] C. Gkantsidis, M. Mihail and A. Saberi. Random walks in peer-to-peer networks: Algorithms and Evaluation. *Performance Evaluation* 63, pp. 241-263, 2006.

[52] C. Gkantsidis, M. Mihail and A. Saberi. Hybrid search schemes for unstructured peer-to-peer networks. *Proc. IEEE INFOCOM*, 2005.

[53] A. Goldsmith. *Wireless Communication*. Cambridge University Press, 2005.

[54] M. Güneş and O. Spaniel. Routing algorithms for mobile multi-hop ad-hoc networks. *Proc. IFIP Conf. on Network Control and Engineering for QoS (Net-Con)*, pp. 120–138, 2003.

[55] Z. J. Haas. A New Routing Protocol For The Reconfigurable Wireless Networks. *Proc. 6th IEEE International Conference on Universal Personal Communications*, pp. 562-566, 1997.

[56] Z. J. Haas, J. Y. Halpern and L. Li. Gossip-Based Ad Hoc Routing. *Proc. IEEE INFOCOM,* 2002.

[57] R. Hekmat. *Ad-hoc Networks: Fundamental Properties and Network Topologies*. Springer, 2006.

[58] R. Hekmat and P. Van Mieghem. Degree Distribution and Hopcount in Wireless Ad-hoc Networks. *Proc. 11th IEEE ICON*, 2003.

[59] M. Heusse, D. Snyers, S. Guerin, P. Kuntz. Adaptive agent-driven routing and load balancing in communication networks. *Rapport technique de l'ENST de Brestagne, RR-98001-iasc*, 1998.

[60] S. S. Iyengar and R. R. Brooks. *Distributed Sensor Networks*. Chapman & Hall/CRC, 2004.

[61] X. Jia, D. Li and D. Du. QoS Topology Control in Ad Hoc Wireless Networks. *Proc. INFOCOM*, 2004.

[62] D. Johnson, D. Maltz, Y.-C. Hu, and J. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks. *Internet Draft, draft-ietf-manet-dsr09-11.txt*, April 2003.

[63] J. Jonasson. On the cover time of random walks on random graphs. *Combinatorics, Probability and Computing* 7, pp. 265-279, 1998.

[64] B.J. Kim, C.N. Yoon, S.K. Han and H. Jeong. Path finding strategies in scale-free networks. *Physical Review E*, vol. 65, 2002.

[65] F.A. Kuipers. *Quality of Service Routing in the Internet: Theory, Complexity and Algorithms*. Ph.D. thesis, Delft University Press, The Netherlands, ISBN 90-407-2523-3, 2004.

[66] G. F. Lawler, O. Schramm and W. Werner. On the scaling limit of planar self-avoiding walk. *Fractal Geometry and applications, a jubilee of Benoît Mandelbrot, Proc. Symp. Pure Math.* 72, vol. II, 339-364, 2002.

[67] S.-J. Lee, W. Su, and M. Gerla. On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks. *Internet Draft, draft-ietf-manet-odmrp-02.txt*, Jan. 2000.

[68] A. Leon-Garcia. *Probability and Random Processes for Electrical Engineering*. 2nd edition, Prentice Hall, 1994.

[69] B. Levin and J. Reeds. Compound Multinomial Likelihood Functions are Unimodal: Proof of a Conjecture of I.J. Good. *Annals of Statistics* 5, pp. 79–87, 1977.

[70] L. Lovász. Random Walks on Graphs: A Survey. *Combinatorics, Paul Erdös is Eighty (Volume 2), János Bolyai Mathematical Society Budapest*, pp. 353-398, 1996.

[71] Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. *Proc. 16th ACM International Conference on Supercomputing*, 2002.

[72] R. Madan, S. Cui, S. Lall and A. Goldsmith. Cross-layer design for lifetime maximization in interference-limited wireless sensor networks. *IEEE Transactions on Wireless Communications*, pp. 3142-3152, vol. 5, no. 11, 2006.

[73] G. S. Manku, M. Naor and U. Wieder. Know thy Neighbor's Neighbor: the Power of Lookahead in Randomized P2P Networks. *Proc. 36th ACM Symposium on Theory of Computing*, pp. 53-64, 2004.

[74] S. Marwaha, C. Tham, and D. Srinivasan. Mobile agents based routing protocol for mobile ad hoc networks. *Proc. IEEE Globecom*, 2002.

[75] M. Mihail, A. Saberi and P. Tetali. Random Walks with Lookahead in Power Law Random Graphs. *Internet Mathematics*, 2006.

[76] D. Miorandi and E. Altman. Coverage and connectivity of ad hoc networks in presence of channel randomness. *Proc. IEEE INFOCOM*, vol. 1, pp. 491–502, 2005.

[77] M. E. J. Newman. The structure and function of complex networks. *SIAM Review* 45, pp. 167-256, 2003.

[78] I.G. Niemegeers and S.M. Heemstra de Groot. Research Issues in Ad-Hoc Distributed Personal Networking. *Wireless Personal Communications: An International Journal*, Vol. 26, Issue 2-3, pp. 149-167, 2003.

[79] I. Olkin, A.J. Petkau and J.V. Zedik. A Comparison of n Estimators for the Binomial Distribution. *J. of the American Statistical Association* 76, pp. 637–642, 1981.

[80] J. Orriss and Stephen S. Barton. Probability distributions for the number of radio transceivers which can communicate with one another. *Proc. IEEE Transactions on Communications*, vol. 51, no. 4, pp. 676–681, 2003.

[81] C. Perkins, E. Royer, and S. Das. Ad hoc on demand distance vector routing. *Internet Draft, draft-ietf-manet-aodv-11.txt*, August 2002.

[82] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *Proc. ACM SIGCOMM,* pp. 234-244, 1994.

[83] L. L. Peterson and B. S. Davie. *Computer Networks - A Systems Approach*. Second Edition, Morgan Kaufmann publishers, 2000.

[84] P. P. Pham and S. Perreau. Performance analysis of reactive shortest path and multi-path routing mechanism with load balance. *Proc. IEEE INFOCOM*, vol. 1, pp. 251–259, 2003.

[85] P. Purkayastha and J. S. Baras. Convergence Results for Ant Routing Algorithms via Stochastic Approximation and Optimization. *Proc. 46th IEEE Conference on Decision and Control*, New Orleans, 2007.

[86] S. Rajagopalan and C. Shen. ANSI: A unicast routing protocol for mobile ad hoc networks using swarm intelligence. *Proc. International Conf. on Artificial Intelligence*, pp. 24–27, 2005.

[87] T. S. Rappaport. *Wireless Communication - Principles and Practice*. Second Edition, Prentice Hall, 2002.

[88] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker. A Scalable Content-Addressable Network. *Proc. of ACM SIGCOMM*, 2001.

[89] M. Roth and S. Wicker. Termite: emergent ad-hoc networking. *Proc. of the 2nd Mediterranean Workshop on Ad-Hoc Networking*, June 2003.

[90] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. *Proc. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware),* Germany, pp. 329-350, 2001.

[91] A. Salhieh and L. Schwiebert. Power Aware Metrics for Wireless Sensor Networks. *Proc. 14th IASTED Conference on Parallel and Distributed Computing and Systems*, 2002.

[92] R. Schoonderwoerd, O. Holland, J. Bruten and L. Rothkrantz. Ant-based load balancing in telecommunications networks, *Journal of Adaptive Behavior* 5 (2), pp. 169–207, 1996.

[93] G. M. Schütz and S. Trimper. Elephants can always remember: Exact long-range memory effects in a non-Markovian random walk. *Physical Review E*, vol. 70, Issue 4, 2004.

[94] S. Singh and C. Raghavendra. PAMAS: Power Aware Multi-Access protocol with Signalling for Ad Hoc Networks. *Proc. ACM Computer Communications Review*, 1999.

[95] P. Sinha, R. Sivakumar and V. Bharghavan. MCEDAR: multicast core-extraction distributed ad hoc routing. *Proc. Wireless Communications and Networking Conference (WCNC)*, pp. 1313 - 1317, 1999.

[96] E. S. Sousa and J. A. Silvester. Optimum Transmission Ranges in a Direct-Sequence Spread-Spectrum Multihop Packet Radio Network, *Proc. IEEE Journal on Selected Areas in Communications*, vol. 8, no. 5, pp. 762–771, 1990.

[97] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *Proc. of the ACM SIGCOMM Conference*, 2001.

[98] D. Subramanian, P. Druschel and J. Chen. Ants and reinforcement learning: a case study in routing in dynamic networks. *Proc. 2nd Mediterranean Workshop on Ad-Hoc Networking*, June 2003.

[99] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[100] H. P. Thadakamalla, R. Albert and S. R. T. Kumara. Search in weighted complex networks. *Physical Review E*, vol. 72, 2005.

[101] G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life (Special Issue on Stigmergy)*, pp. 97–116, 1999.

[102] C.K. Toh. A Novel Distributed Routing Protocol To Support Ad hoc Mobile Computing. *Proc. IEEE 15th Annual International Phoenix Conference on Computers and Communications*, pp. 480-486, 1996.

[103] S. Valverde and R. V. Solé. Internet's Critical Path Horizon. *European Phys. J. B.* 38, pp. 242-252, 2004.

[104] R. van der Hofstad, G. Hooghiemstra, P. Van Mieghem. First passage percolation on the random graph. *Probability in the Engineering and Informational Sciences (PIES)*, vol. 15, pp. 225–237, 2001.

[105] P. Van Mieghem. *Performance Analysis of Computer Systems and Networks.* Cambridge University Press, 2005.

[106] P. Van Mieghem. Paths in the simple random graph and the Waxman graph. *Probability in the Engineering and Informational Sciences (PIES)* 15, pp. 535–555, 2001.

[107] P. Van Mieghem. *Data Communication Networking.* Techne Press, 2006.

[108] P. Van Mieghem, H. De Neve and F.A. Kuipers. Hop-by-hop Quality of Service Routing. *Computer Networks*, vol. 37/3-4, pp. 407-423, 2001.

[109] Y. Xue and B. Li. A location-aided power-aware routing protocol in mobile ad hoc networks. *Proc. IEEE Globecom*, vol. 5, pp. 2837-2841, 2001.

[110] J.-H. Yoo, R. J. La and A.M. Makowski. Convergence results for ant routing. *Proc. CISS*, Princeton University, NJ, 2004.

[111] H. Zhang and J. Hou. On the Upper Bound of α-Lifetime for Large Sensor Networks. *ACM Transactions on Sensor Networks*, vol. 1, no. 2, pp. 272-300, 2005.

[112] M. Zorzi and S. Pupolin. Outage Probability in Multiple Access Packet Radio Networks in the Presence of Fading. *Proc. IEEE Transactions on Vehicular Technology*, vol. 43, no. 3, pp. 604–610, 1994.

Acknowledgements

I would like to thank my advisor and promotor Piet Van Mieghem who gave me the opportunity to work with him at TU Delft. His enthusiasm and insights have been invaluable during my Ph.D. thesis work. I am indebted to all the committee members for their comments and suggestions. My special thanks to Sonia, Ignas, Hans and other people in the IOPGencom project for their comments and views during project meetings.

Life would have been less fun without the people on the 19th floor. My special thanks to Nauman, Umar, Ramin, Weidong, Tom, Jasmina, Milena, Almerima, Fernando, Biengjie, Rob and all other people on the 19th floor for the wonderful time and a friendly atmosphere. A special thanks to Marjon and Wendy for their help during these four years.

I would also like to thank all my friends in Delft for their advice and cooperation specially Andrei, Frederico, Marco and Robert. Last but not least, a special thanks to all my friends, family and well wishers whom I didn't see much in last few years but remain always in my heart.

Santpal Singh
Delft

Curriculum Vitae

Santpal Singh Dhillon was born on 10 September, 1979 in Nathana, Punjab (India). He graduated with a B.Tech (Honors) degree in Electrical Engineering from Indian Institute of Technology, Kharagpur in 2001. He graduated with a MS degree in Electrical and Computer Engineering from Duke University, U.S.A in 2003. In March 2004, he started his Phd degree at TU Delft in the Network and Architecture Services (NAS) group headed by Prof. Piet Van Mieghem.

Publications

- S. S Dhillon and Y. Zhou, "Topology, shortest path routing and lifetime of ad hoc networks ", Proc. 14th Symposium on Communications and Vehicular Technology in the Benelux (SCVT 2007) .

- S. S Dhillon and P. Van Mieghem, "Searching with multiple random walk queries", Proc. 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE PIMRC'07), September 3-7, Athens, Greece.

- S. S Dhillon, X. Arbona and P. Van Mieghem, " Ant Routing in Mobile Ad Hoc Networks ", Proc. Third International Conference on Networking and Services (ICNS'07), June 19-25, 2007 - Athens, Greece.

- S. S Dhillon and P. Van Mieghem, "Comparison of Random Walk Strategies for Ad Hoc Networks ", Proc. of the Sixth Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2007), June 13-15, Corfu, Greece, pp. 196-203.

- S. S Dhillon and P. Van Mieghem, "Performance Analysis of the AntNet algorithm", Computer Networks, Vol. 51, No. 8, June 6, p. 2104-2125, 2007.

- T.J.M. Coenen, P.T.H. Goering, A. Jehangir, J.L. van den Berg, R.J. Boucherie, S.M. Heemstra de Groot, G.J. Heijenk, S.S. Dhillon, W. Lu, A. Lo, P.F.A Van Mieghem and I.G.M.M. Niemegeers, "Architectural and QoS aspects of Personal Networks", Proc. First International Workshop on Personalized Networks, PerNets 2006, 21 July 2006, San Jose, CA, USA.

- S. S. Dhillon and K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks", Proc. IEEE Wireless Communications and Networking Conference, pp. 1609-1614, 2003.

- S. S. Dhillon, K. Chakrabarty and S. S. Iyengar, "Sensor placement for grid coverage under imprecise detections", Proc. International Conference on Information Fusion (FUSION 2002), pp. 1581-1587, 2002.

- S. S. Dhillon, K. Chakrabarty and S. S. Iyengar, "Sensor placement for effective grid coverage and surveillance", Workshop on Signal Processing, Communications, Chaos and Systems, Newport, RI, 2002.

- S. S. Dhillon and K. Chakrabarty, "A fault-tolerant approach to sensor deployment in distributed sensor networks", Proc. Army Science Conference (ASC), Paper ID: JP-05, 2002.

- S. S. Dhillon and S. Chakrabarti, "Power line interference removal from electrocardiogram using a simplified lattice based adaptive IIR notch filter", Proc. 23rd International Conference on Engineering in Medicine and Biology (IEEE-EMBC), pp. 3407- 3412, 2001.