AN OPTIMIZATION OF NEIGHBOR DISCOVERY IN MOBILE AD HOC NETWORKS

Felipe Jovel

Department of Computer Science

APPROVED:

Patricia Teller, Ph.D., Chair

Michael McGarry, Ph.D., Co-Chair

Shirley Moore, Ph.D.

Charles Ambler, Ph.D.
Dean of the Graduate School

to my

MOTHER and WIFE

with love

AN OPTIMIZATION OF NEIGHBOR DISCOVERY IN MOBILE AD HOC NETWORKS

by

Felipe Jovel

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

December 2014

Abstract

This thesis aims to improve the performance of Mobile Ad Hoc Networks (MANETs) that use soft-state signaling for neighbor discovery, specifically OLSR. In particular, it uses simulations and previous work based on experiments conducted on a physical test bed to study the behavior of the neighbor-discovery algorithm used by OLSR to identify and explore ways to optimize the neighbor-discovery process. Candidate optimizations include the actual and relative settings of the refresh and expiry timers used by the algorithm.

Previous studies focused on understanding how the settings of the algorithms Hello_Interval (δ) and TC_ Interval refresh timers affect network performance in terms of protocol message overhead and network throughput. In contrast, this thesis investigates how the setting of the Hello_Validity (τ) timer relative to the setting of the Hello_Interval timer affects network performance in terms of overall network packet loss and how the setting of the Hello_Interval timer affects energy consumption vs. packet loss.

Results obtained from our simulations indicate that the relationship between the settings of these timers impact network performance in terms of the percentage of overall packet loss. In particular, we discovered that: (1) For Hello_Validity values smaller than 2δ, as the Hello_Validity approaches the value of the Hello_Interval, the percentage of packet loss increases due to the proximity of these parameters and collisions; and (2) setting the Hello_Validity timer to twice the value of the Hello_Interval timer results in a configuration with little packet loss. We attribute these results to a phenomenon that we call *"neighbor flapping"*, where neighbor information for a node that is within range is repeatedly placed in and then evicted from the neighbor sets of other nodes. In terms of energy consumption, we discovered that as the Hello_Interval timer increases, energy consumption decreases but packet loss significantly increases.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis represents a continuation of a research effort to understand and optimize the performance of a specific type of wireless network called a Mobile Ad-Hoc Network (MANET). Its main goal is to comprehend the behavior of the neighbor discovery mechanism employed in MANETs, and to identify possible optimizations that will improve its performance in terms of both latency and energy consumption. The work described in this thesis extends, through simulation, the findings of previous work [4], which relied solely on physical experimentation. To facilitate the understanding of the research presented in this thesis, the reader is first introduced to computer networks in Sections 1.1 and 1.2. In Section 1.3 wireless networks and MANETs are described. Finally, Section 1.4 presents the contributions of this thesis and describes its organization.

1.1 Computer Networks

A computer network can be described as a set of computers capable of communicating with each other. Today, the Internet is the largest, well-known computer network. Like many large systems, the Internet is intricate in nature. Abstraction, a powerful concept used to simplify the design of complex systems, was applied in architecting the Internet. The principle of abstraction gave life to abstract layer models used to specify how end-to-end communication between computers is accomplished in the Internet. For example, the Transmission Control Protocol Internet Protocol (TCP/IP) became the most widely used model in computer networking. The Internet Engineering Task Force (IETF) developed TCP/IP. It is a five-layer stack model that consists (top to bottom) of the application,

transport, network, data-link, and physical layers.

1.1.1 TCP/IP stack model overview

Each of the TCP/IP layers provides services to and consumes services from the layer directly below it. To provide these services, protocols for each of the five abstraction layers of TCP/IP were designed. From the perspective of a source node, whenever communication to a remote node is necessary, data has to traverse the TCP/IP stack from top to bottom. Along the way, each layer encapsulates data into its corresponding data unit and then hands it to the layer below. The encapsulation process appends layer-specific information used to guide data to its destination. At the destination node, encapsulated data traverses the TCP/IP stack in a bottom-up fashion. Every layer then performs a decapsulation process, and passes the information to the layer above. This behavior is illustrated in in Figure 1.1.

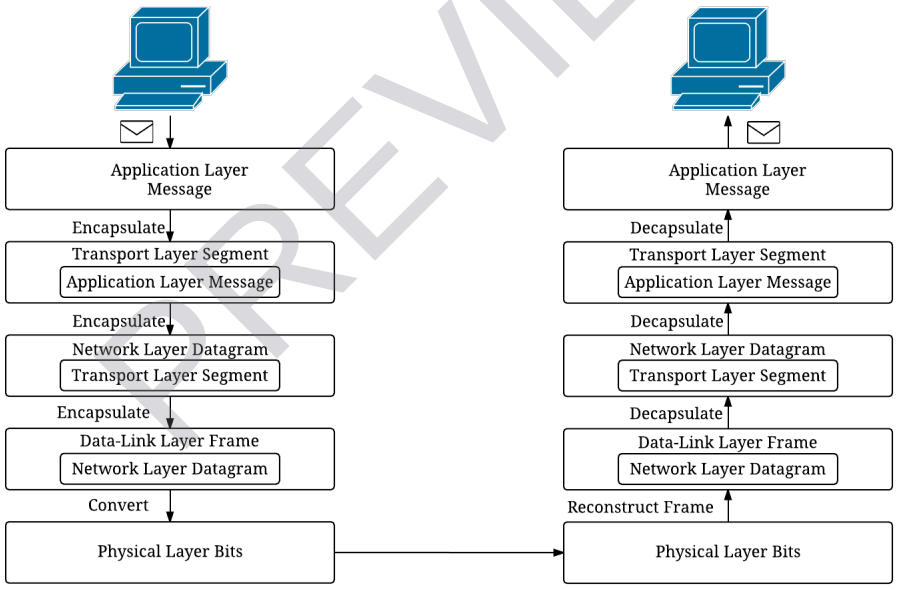


Figure 1.1: An illustration of how data traverses the TCP/IP stack.

At the application layer, protocols specify how applications should exchange messages, which is the data unit at this layer. An example of an application data protocol is the Secure

Shell (SSH) protocol, which enables the establishment of secure connections over insecure networks. In order to transmit messages, the application layer makes use of the process-to-process communication service provided by the transport layer. Depending on the needs of the application employing the transport layer, either TCP or the User Datagram Protocol (UDP) is used to transmit messages encapsulated in segments. In general terms, an application would use TCP when it needs a connection-oriented, reliable, congestion-controlling service. For example, the Post Office Protocol (POP) is an application layer protocol that is used to retrieve email, and because we all want our emails to be displayed exactly as they were sent, it uses TCP's reliable service. On the other hand, an application layer protocol that is intended for video streaming would normally employ UDP since a few frames of the streamed video could be missed without any major repercussions.

Next, to provide process-to-process communication the transport layer uses the network layer, which provides host-to-host communication services. The network layer takes segments and encapsulates them into datagrams. Datagrams are routed through the network from source to destination by means of routing protocols. The work described in this thesis is associated with one of the routing protocols in the network layer. Thus, Section 1.2 is devoted to expanding the description of the network layer.

Continuing down the TCP/IP stack model, the data-link layer receives network layer datagrams and encapsulates them into data-link layer frames. The data-link layer is responsible for transmitting frames over individual links. Note that the communication path from source to destination often involves different link types. For example, some may be wireless links, while others might be copper or optical. Thus, different link-layer protocols are needed to handle different link types along a communication path. Another important function of the data-link layer is that of arbitrating what device gets to transmit over a link. This is necessary when multiple computers have access to the same link. If multiple nodes try to transmit over the same link at the same time, collisions occur. To mitigate this, the data-link layer provides media access control mechanisms that coordinate communication. Finally, once the data-link layer hands a frame to the physical layer, the frame is sent over

the link as bits. The physical layer at the destination reconstructs the frame from the received bits, and delivers it to the data-link layer. This decapsulation process continues up the stack until the original message is received at the application layer.

1.2 Network Layer

As previously stated, the network layer is responsible for delivering packets from source to destination. Specialized devices called packet switches are used to guide packets through the network. A packet switch has both input and output ports. The purpose of a packet switch is to receive packets at an input port, and forward them to the correct output port. In order to determine the output port to which a packet should be forwarded, packet switches maintain forwarding tables that contain a mapping of destination addresses to output ports. When a packet arrives at an input port its header is parsed to determine its intended destination address. The destination address is then used to index into the forwarding table and the packet is forwarded accordingly.

It is worth noting that there are data-link and network layer switches. The forwarding table in a data-link layer switch is called an arp table, while a network layer switch has a routing table. To distinguish between the two, data-link switches are usually called link-layer switches and network switches are called routers.

Clearly, the routing of packets from source to destination is enabled by the routing tables in routers. Thus, an important related topic is routing-table configuration. This can be done in a static or dynamic manner. Static routing-table configuration can be done by having network administrators manually set the entries in routing tables. On the other hand, dynamic routing is made possible by designing routing protocols that routers execute. The function of a routing protocol is to obtain global or partial topology information and, based on this, calculate an optimal route to all network destinations. Most routing protocols fall into one of two broad categories: link-state and distance-vector routing protocols.

1.2.1 Link-state routing algorithms

To calculate an optimal path to all nodes in a network, link-state routing algorithms require as input a complete map of the network. This map provides link-state routing algorithms with a description of all the nodes in the network, the links between nodes, and the costs of traversing those links. Since routing algorithms are nothing more than a computer program that is executed by specialized computers called routers, an abstract data type called a graph is often used to logically represent the map of the network. Formally, a graph G = (V, E) consists of a set V of vertices and a set E of edges. Note each edge (x,y) is assigned a cost `c(x,y)`, which is used in the computation of the cost of a path.

To construct the graph of the network, link-state routing algorithms rely on link-state broadcast algorithms. There are three types of transmissions possible in computer networks: unicast, multicast, and broadcast. Unicast, as the name implies, is the transmission of data to a single destination identified by a unique address. Multicast transmissions target a subset of the network's nodes, while broadcast transmissions are intended for all nodes in a network. To provide the link-state routing algorithm with a complete graph of the network, link-state broadcast algorithms broadcast link-state messages. Every node in the network broadcasts link-state messages to advertise its neighbor nodes to message recipients. This information enables each of the nodes running the link-state routing algorithm to construct the graph of the network. Once the graph of the network is available, the link-state routing algorithm identifies an optimal path to every destination in the network and the routing table is configured according to these results. An example of a link-state routing algorithm is Dijkstra's algorithm, which solves the single source, shortest-path problem for a graph.

1.2.2 Distance-vector routing algorithms

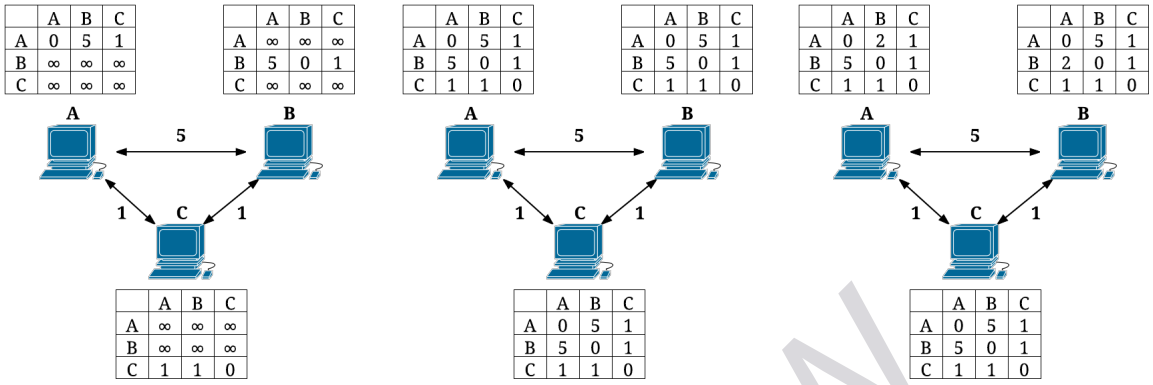
Unlike link-state routing protocols, distance-vector routing protocols do not have access to the global network topology. Instead, optimal paths are iteratively constructed by having each node transmit its distance vector to its directly attached nodes, i.e., its neighbor

nodes. The distance vector of a node consists of the distances (or costs) associated with the current best known paths from this node to all other known destination nodes in the network. For example, assume that a node 'a' has two directly attached neighbors, 'b' and 'c', each with a cost of 2 and 4, respectively. Then, the distance vector of node 'a' is (2,4). A distance vector for a particular node describes the node's current view of the network. In distance-vector routing, every node keeps its own distance vector as well as the distance vectors of its neighbor nodes.

Initially, every node is only aware of its own distance vector and is not aware of the distance vectors of its neighbor nodes. To inform neighbor nodes of its distance vector, every node transmits its distance vector to these nodes. Upon receiving a distance vector from a neighbor node, a recipient node uses it to update its local copy of the distance vector of the corresponding node and its own distance vector. When the receipt of a distance vector from a neighbor node causes a node to update its distance vector, the node transmits its updated distance vector to its neighbors. Eventually when the network becomes stable, nodes stop transmitting distance vectors because their routing tables contain optimal paths to all destinations.

It is easy to see that the distance vectors maintained by the nodes in a network allow them to identify the least-cost (optimal) paths to other nodes in the network. Next, we consider how routing tables are configured. This is accomplished using distance vectors. Whenever a node updates its distance vector, it updates its routing table. For example, consider the network in Figure 1.2a. Initially, node 'a' has a distance vector of (0,5,1), indicating that it can reach itself at a cost of 0, 'b' at a cost of 5, and 'c' at a cost of 1. At this point, the routing table of node 'a' indicates that it can send data to nodes 'b' and 'c' directly, i.e., in one hop. When all nodes send their initial distance vectors to their neighbors, the neighbor nodes store these distance vectors as shown in Figure 1.2b. Next, as shown in Figure 1.2c every node uses the recently acquired information to update its distance vector. For example, using the recently obtained distance vector of node 'c' , node 'a' realizes that 'c' can reach 'b' at a cost of 1. Since node 'a' can reach 'c' at a cost of 1,'a'

updates its distance vector to reflect that it can reach ‘b’ at a cost of 2, i.e., through ‘c’. This information is then used by node ‘a’ to add an entry to its routing table that shows that to send data to ‘b’, it must forward it through node ‘c’.



(a) Initial distance vectors.

(b) Distance vectors after first transmission.

(c) Distance vectors after recalculation.

Figure 1.2: Distance vector configuration.

1.3 Infrastructure vs. Infrastructure-less Wireless Networks

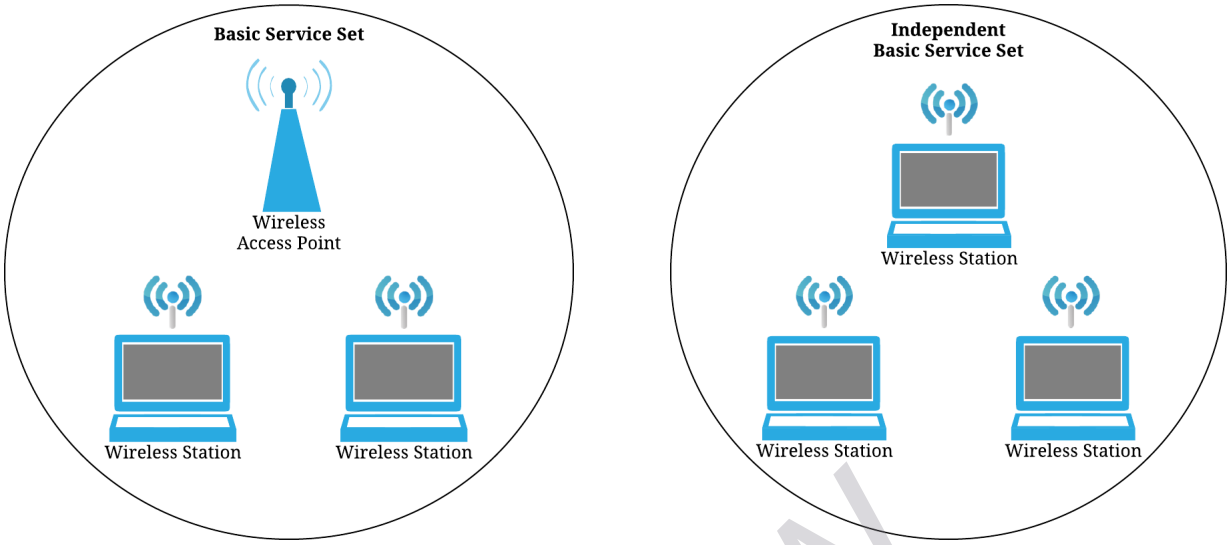
The wireless standard used in this work is the IEEE 802.11 standard, which is commonly referred to as WiFi. The IEEE 802.11 standard defines wireless stations and access points as the main participants in a wireless network. Wireless stations are the actual computers (or hosts) that use the wireless network to communicate. On the other hand, access points are used to establish and manage communication in the wireless network. In order to join a wireless network, a wireless station has to first associate itself with an access point. Once the wireless station does this, it can begin to communicate via the wireless network.

Communication takes place by relaying messages through an access point. Even in the case where two wireless stations, a and b, are within range, messages are first sent to the

access point. If a wants to communicate with b, it sends its message to the access point, and the access point forwards it to b. Since access points forward messages, it is common to have them also function as routers.

The mode of operation just described is referred to as an infrastructure. The fact that an access point is required for the wireless network to exist is what makes this an infrastructure wireless network. The basic building block in this network architecture is a basic service set. A basic service set is composed of an access point and one or more wireless stations.

On the other side of the spectrum there are independent basic service sets. An independent basic service set is composed of only wireless stations and, thus, it is usually referred to as infrastructure-less mode or ad-hoc mode. In an ad-hoc wireless network nodes communicate directly with other nodes within transmission range. A mobile ad-hoc network is defined as an ad-hoc network that is self-configuring. To be self-configuring, a mobile ad-hoc network must have a routing protocol. This work focuses on the Optimized Link State Routing (OLSR) protocol, which is a popular routing protocol for mobile ad-hoc networks (MANETs). Specifically, the neighbor discovery mechanics of the protocol are studied with the purpose of optimizing them.



(a) Infrastructure wireless network.

(b) Infrastructure-less wireless network.

Figure 1.3: Infrastructure vs. infrastructure-less wireless networks

1.4 Thesis Contributions and Organization

The neighbor discovery process is the first step towards establishing a MANET. In the OLSR protocol, neighbor discovery is accomplished through the periodic exchange of Hello_Messages that advertise nodes' links. This periodic transmission of control messages makes OLSR a good candidate for highly mobile networks, since a high frequency of control messages may enable it to quickly discover changes in topology. However, sending control messages too frequently can be wasteful in terms of throughput and energy consumption. Thus, the goals of this research are to:

- Understand how the setting of Hello_Validity timer of OLSR, relative to that of its Hello_Interval timer, affects network performance in terms of overall packet loss, and
- Understand how the setting of OLSR's Hello_Interval timer effects energy consumption versus packet loss.