DEVELOPMENT OF NEW BROADCAST PROTOCOLS IN A MOBILE AD HOC NETWORK BASED ON MACHINE LEARNING

by

Fuat Bilgin

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Mathematical and Computer Sciences)

Golden, Colorado

Date _____

Signed: _____
Fuat Bilgin

Approved: _____
Dr. Michael Colagrosso
Assistant Professor

Golden, Colorado

Date _____

_____
Dr. Graeme Fairweather
Professor and Head
Department of Mathematical and
Computer Science

ABSTRACT

A mobile ad hoc network (MANET) involves computers, typically wireless mobile nodes (MNs), that cooperatively form a network without specific user administration or configuration, allowing an arbitrary collection of MNs to create a network on demand. Broadcasting, the process in which one MN sends a packet to all the other nodes in the network, functions as a foundation of MANET communication. A number of unicast, multicast, and geocast protocols utilize broadcasting as a building block, providing important control and route establishment functionality. Therefore, any improvements to the process of broadcasting can be immediately realized by several MANET applications. Several efficient broadcast protocols have been proposed, but previous research based on simulation discovered that no single protocol for broadcasting works well in all possible network conditions in a MANET. Furthermore, each of the protocols fails catastrophically when the severity of the network environment is increased.

Two new broadcast protocols have been developed and presented in this thesis. The proposed protocols are based on a new approach in which rebroadcasting a packet is decided through an intelligent classification scheme. Each MN builds a classifier and trains it on data collected from the network environment. For an input vector describing a broadcast packet and current network conditions, the classifier returns a class label of “Rebroadcast” or “Drop”. Because each MN can adapt to changing network conditions, the result is a more robust communication protocol and more efficient use of network resources.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

I would like to express my profound gratitude to my advisor Dr. Michael Colagrosso, for his support and guidance in my research and accomplishing this Master of Science thesis.

I would like to thank Dr. Tracy Camp, for allowing me to participate in her TOILERS research group, for her continuous encouragement, and intellectual and financial support.

Further, I would also like to extend my sincerest thanks to Dr. Kadri Dagdelen without whom I would not have been able to attain this degree.

I am very thankful to my parents, Bekir Bilgin and Emel Bilgin, for their support and affection and for showing me the importance of knowledge, which inspired me to pursue this dream.

Chapter 1

INTRODUCTION

1.1 Overview of Ad Hoc Wireless Networks

A mobile ad hoc network (MANET) involves wireless mobile nodes (MNs) that cooperatively form a network without specific user administration or configuration, allowing an arbitrary collection to create a network on demand. MANETs present an environment in which the network topology is both dynamic and unpredictable. The lack of static infrastructure requires each MN to operate not only as an end-system, but also as a router to forward packets.

There are three salient features of MANETs:

- **Mobility:** Application can be used everywhere.
- **Peer-to-Peer:** Direct communication between peers is mandatory. Interaction is direct, without a central base.
- **Collocation:** Mere collocation of one or more MNs within a certain perimeter.

Ad hoc networking can be applied anywhere that there is no communication infrastructure, or the existing infrastructure is expensive or inconvenient to use. There are numerous scenarios that can be considered as typical ad hoc applications [49].

- **Military Battlefield:** To take advantage of commonplace network technology to maintain an information network between soldiers, vehicles, and military information headquarters.

- **Rescue:** Can be used in emergency/rescue operations for disaster relief efforts (e.g., fire, flood, and earthquake).
- **Local Level:** To autonomously link an instant and temporary multimedia network to spread and share information among participants (e.g., a conference or classroom). Some of the other potential local level network applications might be in civilian environments like taxicabs, sports stadiums, boats, and small aircrafts.
- **Personal Area Network (PAN):** Short-range intercommunication between mobile devices, such as laptops and PDAs.

1.2 Motivating Application

In addition to the sample applications previously listed, our motivating application is monitoring of an open pit mining environment. In this environment, the developed system is mounted inside a haulage truck cabin and receives the GPS signal from the GPS receiver mounted on the truck. The developed software, OptiTrack, calculates the position of the truck with respect to Digital Terrain Model (DTM) and sends this information to the other trucks in the network. Due to the difficulties of obtaining the wireless network coverage throughout the quarry based on 802.11b operating in infrastructure mode, the suggestion was made to operate the network in ad hoc mode. As such, the current research is undertaken to implement an ad hoc network to improve the developed system to increase the coverage of the wireless network. Therefore, the future phase of this system will include the implementation of the system in a MANET using various broadcast protocols and comparing them in an actual operating mining environment. Because the broadcasting protocol will be a building block of the system, it is imperative to have the most effective broadcast protocol possible for efficient delivery of the data.

The broadcast protocols that will be implemented in OptiTrack system must be adaptive and handle a wide range of network conditions; otherwise it jeopardizes the safety in the whole system.

1.3 Problem Statement

There has been recent interest in the development of network-wide broadcast protocols for MNs in an ad hoc network [1]-[14]. Network-wide broadcasting, simply referred to as "broadcasting" for the remainder of this thesis, is the process in which one MN sends a packet to all MNs in the network (or all nodes in a localized area). A number of research groups have proposed different broadcasting techniques. Each of these protocols operates differently and can be categorized into four groups according to their inherent characteristics. A performance evaluation of MANET broadcast protocols is available in [15].

Categorization of broadcast protocols may be listed as follows:

1. **Simple Flooding** [4]-[6]
2. **Probability Based Methods**
    a. Probabilistic Scheme [29]
    b. Counter-Based Scheme [29]
3. **Area Based Methods**
    a. Distance-Based Scheme [29]
    b. Location-Based Scheme [29]
4. **Neighbor Knowledge Methods**
    a. Flooding with Self Pruning [5]
    b. Scalable Broadcast Algorithm (SBA) [9]
    c. Dominant Pruning [5]
    d. Multipoint Relaying [12]
    e. Ad Hoc Broadcast Protocol (AHBP) [11]
    f. Connected Dominating Set (CDS)-Based Broadcast Algorithm [9]
    g. Lightweight and Efficient Network-Wide Broadcast (LENWB) [14]

Besides the broadcast protocols that are previously listed, there are other network layer protocols for MANETs in terms of unicast protocols, multicast protocols and geocast protocols. Unicast protocols (e.g., [16]-[24]) are utilized in a case where there is one sender node and one receiver node. In unicast communications, a packet is sent from a single source to a specified destination in the network. Multicast protocols (e.g., [4] and [26]) provide a delivery of the same packet simultaneously to a group of clients. In multicast communications, there can be one or more sender nodes that send a piece of information to a set of receiver nodes. Geocast protocols (e.g., [27], [28]) allow for the delivery of packets to a group of nodes in a specified geographical area.

Broadcasting is a building block for most other network layer protocols (unicast, multicast and geocast). For example, unicast routing protocols such as Dynamic Source Routing (DSR) [16], [17], Ad hoc On-demand Distance Vector (AODV) [18], [19], Zone Routing Protocol (ZRP) [20]-[22], and Location Aided Routing (LAR) [23] use broadcasting or a derivation of it to establish routes. Other unicast routing protocols, such as the Temporally-Ordered Routing Algorithm (TORA) [24], use broadcasting to transmit an error packet for an invalid route. Broadcasting is also often used as a building block for multicast protocols (e.g., [4] and [27]) and geocast protocols (e.g., [27], [28]).

The preceding network protocols typically use a simplistic form of broadcasting called Simple Flooding, in which each node (or every node in the localized area) retransmits each received unique packet exactly once. However, Simple Flooding often causes unproductive and harmful bandwidth congestion (e.g., called the "broadcast storm problem" in [29]) and wastes node resources. On the other hand, a number of research groups have proposed more efficient broadcasting techniques (see the preceding reference list), whose goal is to minimize the number of retransmissions while attempting to ensure that a broadcast packet is delivered to each node in the network.

The performance evaluation of MANET broadcast protocols in [15] illustrates that no single protocol for broadcasting works well in all possible network conditions in a

MANET. Furthermore, every protocol fails catastrophically when the severity of the network environment is increased.

1.4 Solution Methodology

All of the broadcast protocols based on probability, area, and neighbor knowledge use static algorithms. In contrast to these static protocols, [15] proposes a hand-tuned rule to adapt the main parameter of one protocol and demonstrates that hand tuning works well in many network environments. The adaptive rule used in [15], however, makes strong assumptions that are specific to the network conditions under which it was tested; from a machine learning perspective, it is desirable for the protocol to tune itself in a systematic, mathematically-principled way. This thesis proposes such protocols using machine learning classifiers and demonstrates their flexibilities in Chapters 4 and 5.

In this thesis, we first utilize Bayesian networks [30] for our learning models because of their expressiveness and more elegant graphical representation compared to other "black box" machine learning models. Bayesian networks, sometimes called belief networks or graphical models, can be designed and interpreted by domain experts because they explicitly communicate the relevant variables and their interrelationships. In network-wide broadcasting, mobile nodes must make a repeated decision (rebroadcast or drop), but the input features that MNs can estimate (e.g., speed, network load, local density) are noisy and, taken individually, are weak predictors of the correct decision to make. Our results illustrate a Bayesian network combines the input features appropriately and often predicts correctly whether to rebroadcast or drop.

Second, we utilize a model that learns via boosting, which is one of the most powerful learning methods introduced in the last 10 years. Boosting is the combination of many "simple" classifiers that produces a "complex" classifier. In network-wide broadcasting, when MNs make repeated decisions, a decision stump is used to create simple models. Individually simple and noisy input features (e.g.,

speed, network load, and local density) are chosen one-by-one to create a complex predictor for a correct decision. Our results present a Boosting classifier that combines the simple classifiers appropriately and predicts whether to rebroadcast or not.

Our learning methods require an objective function for MNs to use to assess their performance. We define the concept of a *successful retransmission* of a broadcast packet, and we require each MN to estimate when successful retransmissions occur. With respect to our objective function, MNs will change their behavior such that the posterior probability of future successful retransmissions is maximized. We describe our estimate of a successful retransmission in Chapter 2, which is conceptually simple, computationally efficient, and requires no communication overhead. We also enumerate the consequences of incorrectly assessing successful retransmission, and propose an augmented method (one that does require communication overhead) to reduce errors

1.5 Description of Work

Because a broadcast protocol is a building block of many other MANET routing protocols, it is imperative to have the most effective broadcast protocol possible in a given network. Because of this, this thesis carries out the development of two broadcast protocols that are the most efficient under the widest range of network conditions. Our simulation comparisons are designed to test the new broadcast protocols based on machine learning and to compare them with some of the proposed broadcast protocols under conditions where the effects of mobility, congestion, and node density are combined as in real networks.

We believe our broadcast protocols are competitive with previously published broadcast protocols; under high network severity, our protocols are the most robust. We argue that implementing these protocols in a MANET is feasible, and doing so would lead to improve network performance. In our MANET implementation (OptiTrack), due to the shortcomings of existing broadcast protocols, broadcast

protocols based on machine learning may be more suitable to the MANET established in a mining environment.

1.6 Organization

The remainder of this thesis is organized as follows. Chapter 2 begins with the categorization of the broadcast protocols in Section 2.1, including their common attributes. Section 2.2 describes known broadcast methods. We then discuss the limitations of these protocols in Section 2.3. In Chapter 3, Section 3.1 introduces the machine learning methodology. Section 3.2 describes the supervised learning approach to machine leaning. Section 3.3 then developes the feedback mechanism used to estimate performance. Chapters 4 and 5 present the new machine learning based broadcast protocols and demonstrate their algorithms. Chapter 6 deals with the simulation comparison of the new broadcast protocols and their training preparations. Section 6.1 provides an outline of the studies we performed and the simulation parameters we chose. In Section 6.2, we explain how the training sets for our protocols are created. Section 6.3 then gives the results from the simulations. Chapter 7 summarizes the motivating work for the development of the machine learning based broadcast protocols. It further discusses the developed system and required changes in order to implement the developed protocols. Lastly, Chapter 8 presents conclusions and future work.

Chapter 2

RELATED PREVIOUS WORK

2.1 Introduction

In the IEEE 802.11 MAC [31] protocol, the RTS/CTS/data/ACK handshake is designed for unicast packets to ensure that the transmission media is shared correctly. To send a broadcast packet, an MN needs only to assess a clear channel before transmitting. Because no resource is provided at a collision (e.g., due to a hidden node), an MN has no way of knowing whether a packet was successfully received by its neighbors. Thus, the most effective network-wide broadcasting protocols try to limit the probability of collisions by limiting the number of rebroadcasts in the network.

In previous work [15], broadcast protocols are categorized into four related areas:

- **Simple Flooding:** Nodes rebroadcast all received unique packets exactly once.
- **Probability Based Methods:** Nodes rebroadcast with a predetermined probability.
- **Area Based Methods:** Nodes rebroadcast if it will reach sufficient additional coverage area.
- **Neighbor Knowledge Methods:** Nodes rebroadcast based on its neighbor knowledge (algorithms are based on graph theory).

In [15], several existing protocols are presented with a detailed performance investigation; comparison of the protocols indicates that the performance of Neighbor Knowledge Methods is superior to the performance of other methods proposed for flat

network topologies. Therefore, we limit our discussion to the most commonly used Simple Flooding protocol and two Neighbor Knowledge Methods.

2.2 Review of Previously Developed Broadcast Protocols

2.2.1 Simple Flooding

The algorithm for Simple Flooding starts with a source node broadcasting a packet to all its neighbors. Each of those neighbors in turn rebroadcast the packet exactly once. This process of flooding continues until all nodes that have received the packet rebroadcast it. Since flooding has a control mechanism for minimal transfer delay, to propagate with the shortest path, it results in collisions and high redundancy in MANETs.

2.2.2 Scalable Broadcast Algorithm (SBA)

The Scalable Broadcast Algorithm (SBA) [10] requires that all nodes know their neighbors within a two hop radius. Each node achieves the knowledge of two-hop topology information, centered at itself, via "Hello" packets. When Node *B* receives a broadcast packet from Node *A*, Node *B* schedules the packet for delivery with a RAD (Random Assessment Delay: the amount of time that a node waits to make the decision) if Node *B* has additional neighbors not reached by Node *A*'s broadcast. For each redundant packet received, Node *B* again determines if it can reach any new nodes by rebroadcasting. This process continues until either the RAD expires and the packet is sent, or the packet is dropped. Let us consider the network topology in Figure 2.1.

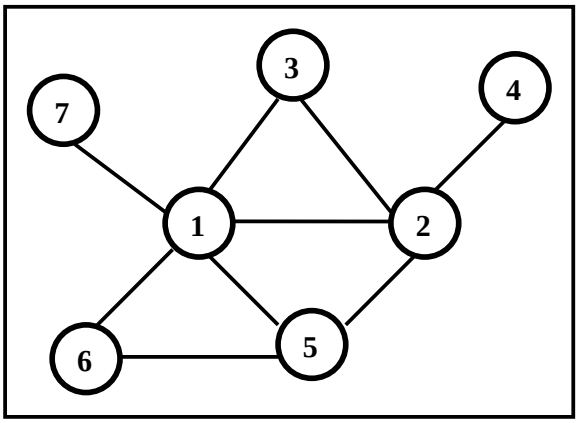

Figure 2.1: An Example of Network Topology for the Scalable Broadcast Algorithm. Nodes Share Neighbor Information via "Hello" Packets.

Node 1's two-hop topology information:

One Hop	**Two Hop Via the Given One Hop**
2	3,4,5
3	2
5	2,6
6	5
7	-

Suppose Node 1 receives a broadcast data packet from Node 2. Node 1 compares Node 2's one hop neighbors with its one hop neighbors to see if it has additional neighbors not reached by Node 2.

Node 1's one hop neighbors:

Node 2 => (sent the broadcast packet)
Node 3 => (received Node 2's transmission)
Node 5 => (received Node 2's transmission)
Node 6 => (did NOT receive Node 2's transmission)

Node 7 => (did NOT receive Node 2's transmission)

According to this situation demonstrated in Figure 2.2, Node 1 has additional neighbors (Nodes 6 and 7) which have not received Node 2's transmission of the broadcast packet. Thus, Node 1 schedules this data packet to be rebroadcast in a short time interval (a random assessment delay or RAD).

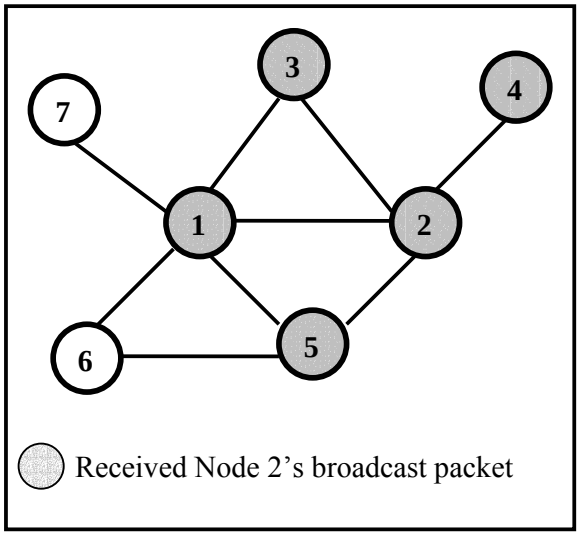

Figure 2.2: An Example of Network Topology for the Scalable Broadcast Algorithm (2)

Let's assume that Node 1 receives a second copy of the packet from Node 5 before its RAD expires, thus it again determines if it can reach any new nodes by broadcasting the received data packet.

Node 1's one hop neighbors (after receiving the redundant packet from Node 5):

Node 2 => (sent the broadcast packet)
Node 3 => (received Node 2's transmission)

Node 5 => (received Node 2's transmission)

Node 6 => (received Node 5's transmission)

Node 7 => (did NOT receive Node 2's nor Node 5's transmission)

As illustrated in Figure 2.3, receiving the redundant data packet from Node 5 implies that Node 6 also receives it. When Node 1's RAD expires, Node 7 is still uncovered, so Node 1 decides to rebroadcast.

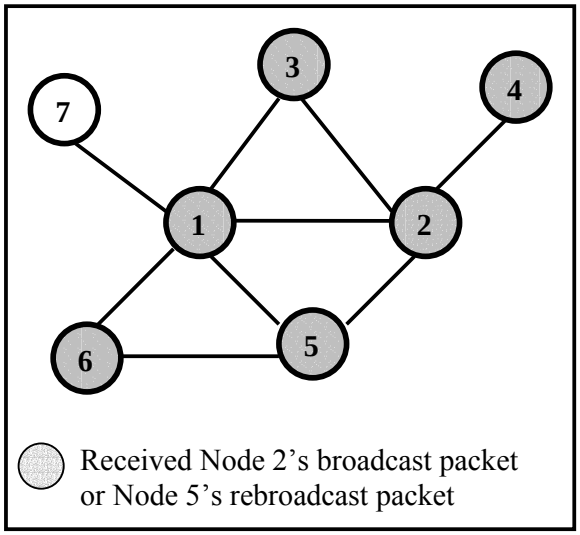

Figure 2.3: An Example of Network Topology for the Scalable Broadcast Algorithm (3)

As a result of the study in [15], it is found that higher assessment delay is effective in increasing the delivery ratio of SBA in congested networks. Because a lower assessment delay is desired in non-congested networks (to reduce end-to-end delay), [15] developed an adaptive SBA scheme. Specifically, if the node is receiving more than 260 packets per second on average, the node uses a RAD T_{max} time of 0.05 seconds. Otherwise, the node uses a RAD T_{max} time of 0.01 seconds.

2.3.4 Ad Hoc Broadcast Protocol (AHBP)

In the Ad Hoc Broadcast Protocol (AHBP) [11], only nodes designated as a Broadcast Relay Gateway (BRG) within a broadcast packet header are allowed to rebroadcast the packet. The algorithm for a BRG to choose its next BRG set is:

1. Find all 2-hop neighbors that can only be reached by one 1-hop neighbor. Assign those 1-hop neighbors as BRGs.

2. Determine the resultant cover set (i.e., the set of 2-hop neighbors that will receive the packet from the current BRG set)

3. From the remaining 1-hop neighbors not yet in the BRG set, find the one that would cover the most 2-hop neighbors not in the cover set. Assign this 1-hop neighbor as a BRG.

4. Repeat steps 2 and 3 until all 2-hop neighbors are covered.

5. When a node receives a broadcast packet and is listed as a BRG, that node determines which of its neighbors also received the packet in the same transmission. These neighbors are considered already "covered" and are removed from the neighbor graph used to choose the next hop BRGs.

6. AHBP is extended to account for high mobility networks. Suppose Node *B* receives a broadcast packet from Node *A*, and Node *B* does not list Node *A* as a neighbor (i.e., Node *A* and Node *B* have not yet exchanged "Hello" packets). In AHBP-EX (extended AHBP), Node *B* will assume BRG status and rebroadcast the packet.

As an example, consider the network topology given in Figure 2.4.

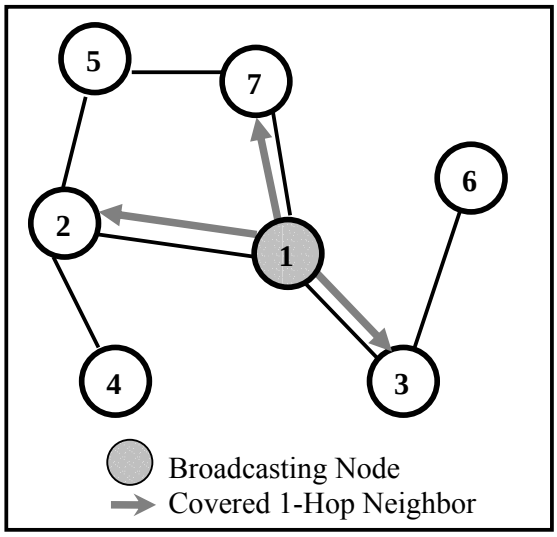

Figure 2.4: An Example of Network Topology for the Ad Hoc Broadcast Protocol

In this scenario, **Node 1**'s two-hop topology information:

One Hop	**Two Hop Via the Given One Hop**
2	4,5
3	6
7	5

Node 1 wants to send a broadcast packet, so it chooses Node 2 and Node 3 as BRGs; thus, it can cover all the two hop neighbors. According to Node 1's two-hop topology table, Node 1's two-hop neighbors are Node 4, Node 5, and Node 6. As it is shown in Figure 2.5, choosing Node 2 as a BRG covers Node 4 and Node 5; choosing Node 3 as a BRG covers Node 6.

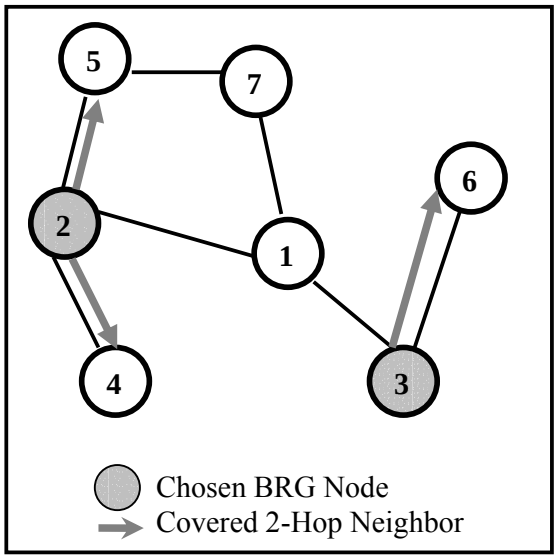

Figure 2.5: An Example of Network Topology for the Ad Hoc Broadcast Protocol (2)

When Node 7 receives the packet, it updates its table, but does not resend the packet because Node 1 did not choose it as a BRG. When Node 3 and Node 2 receive the packet, they also update their tables, and since they have been chosen as BRGs, they choose new BRGs from their neighbor list and rebroadcast the packet. These steps are recursively repeated until the message is propagated to all possible receivers.

2.3 The Shortcomings of Existing Broadcast Protocols

The shortcomings of existing broadcast protocols can be listed as:

1. Non-Neighbor Knowledge Methods require more rebroadcasts than the other methods studied, with respect to the number of retransmitting nodes [15].
2. The schemes that utilize a RAD (such as SBA) suffer in congestive networks unless a mechanism to adapt a node's RAD to its local congestion level is implemented [15].

3. The Neighbor Knowledge methods that do not use local information to determine whether to rebroadcast (such as AHBP) have difficulty in mobile environments [15]. The mobility extension for AHBP (AHBP-EX) marks an improvement over AHBP; however, it still under performs the other protocols. SBA naturally adapts to mobility by requiring more nodes to rebroadcast.
4. No single Neighbor Knowledge protocol evaluated performed the best in all studies [15].

Based on these shortcomings, none of the existing broadcasting protocols are satisfactory for wide-ranging MANET environments. Because the adaptive nature of the developed algorithm SBA showed significant improvements over the non-adaptive protocols, the development of new broadcasting protocols based on machine learning concepts appeared to be advantageous to pursue further work.

Chapter 3

MACHINE LEARNING APPROACH

3.1 Introduction

In Machine Learning (ML), computer systems that learn from experience and adapt to their environment are built. Machine learning systems require learning algorithms that specify how the system should change its behavior as a result of experience. ML deals with the design of computer programs and systems that are able to take advantage of data, examples, or experiences to improve their performance on a task. As shown in Figure 3.1, the process of training action is response to experience and involves the fields of statistics and decision theory.

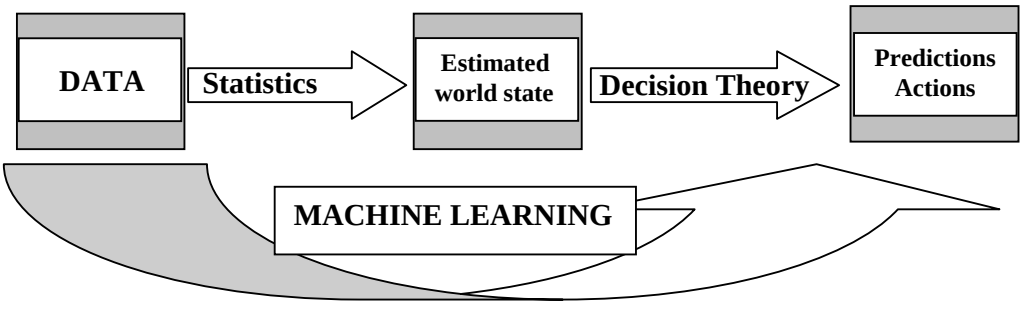

Figure 3.1: Machine Learning [46]

Our inspiration for applying machine learning to develop a broadcasting protocol stems from previous work summarized in Section 2.3. This work concludes that existing broadcast algorithms are too brittle to support a wide range of MANET environments. An initial adaptive protocol developed in [15] had only a single decision rule, yet it is one of the most robust protocols.

3.2 Supervised Learning

According to the learning process, if the learner is given a set of examples (training data) and each example shows what output will be returned for a given input, then this type of learning can be classified as supervised learning. Supervised learning is a technique for creating a function from training data, where the function's output can be a continuous, real number (called *regression*), or can predict a discrete class label of the input objects (called *classification*). The task of supervised leaning is to generalize the prediction from the presented data (training data which has been seen) to unseen situations in a reasonable way. Figure 3.2 shows a toy regression problem to illustrate the relationship between training data, output, and a model's representation.

A wide range of algorithms have been developed for this task including:

- **Bayesian Learning**
- K-Nearest Neighbor
- Decision Tree
- Neural Network
- Bagging (Ensemble Technique)
- **Boosting (Ensemble Technique)**

In our approach to broadcasting, first we determined the structure of the learned function and corresponding algorithms, which are Naive Bayes (Bayesian leaning) and AdaBoost (Boosting). Then we gathered a training set, which is a set of input objects (i.e., information about an incoming packet) and corresponding outputs (i.e., "Rebroadcast" or "Drop"). Because our outputs are discrete class labels, we treat the

broadcast decision as a classification problem. Lastly, to estimate performance we developed a feedback mechanism which is discussed in the following section.

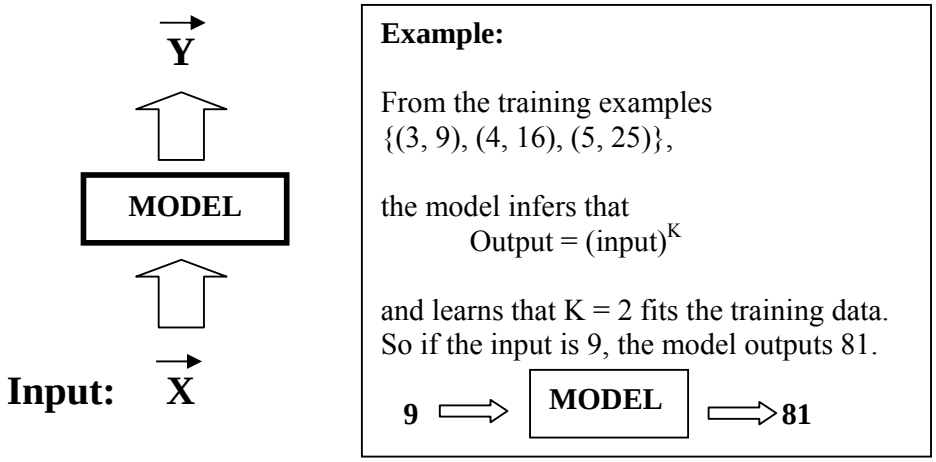

Figure 3.2: Supervised Learning Model

3.3 Feedback Mechanism to Estimate Performance

In two classification problems (i.e., when the output of the model are "+" and "-"), there are two types of classification errors, which are shown in Table 3.1. We treat the decision to rebroadcast a packet as a two-class classification problem in which "Rebroadcast" is the "+" class and "Drop" is the "-" class. If a classifier prediction matches the real life target then it is a good prediction, but if it does not match the target, then it is a bad prediction. If a wrong prediction is a False Positive, then it is called a Type I error. If a wrong prediction is a False Negative, then it is called a Type II error.

		CLASSIFIER: +	CLASSIFIER: -
Real Life	**+**	**True Positive**	**False Negative (TYPE II)**
	-	**False Positive (TYPE I)**	**True Negative**

Table 3.1: Classification Errors

We require an objective function that assesses whether a given mobile node is beneficially contributing to the network's delivery of broadcast packets. Each node will estimate this objective function and tune its behavior in order to maximize it. Intuitively, each mobile node must make a decision whether to retransmit an incoming broadcast packet, so our objective function should reflect whether the node made a good decision or not. As a first approximation, we define the concept of a successful retransmission:

Successful retransmission

> For given mobile node *A* and broadcast packet *X*, *A* considers *X* to be a successful retransmission if after broadcasting *X*, *A* hears one of its neighbors also broadcast *X*.

The goal of this definition is to capture the idea that once node *A* broadcasts a packet to its neighbors, if *A* hears one of them rebroadcast it, then *A* can infer that it has helped in propagating the message. The insight is that node *A* has no choice but to hear broadcasts of it neighbors, and can therefore collect this feedback without any communication overhead.

We identify two ways in which a misclassification can be made, using the terms from Table 3.1.

Type I error If node *A* retransmits packet *X*, but neighbor node *B* retransmits a copy of *X* it received from another node, node *A* will incorrectly infer a

successful retransmission. We anticipate these "false positive" errors to be more common with increasing congestion.

Type II error If, for example, node *A* is near the fringe of the network and delivers a packet *X* to neighbor *B*, who is on the fringe, *B* might decide not to retransmit the packet. Node *A* will incorrectly assume that this was an unsuccessful retransmission. This type of "false negative" error is harder to compensate for.

In machine learning, the goal is not to learn an exact representation of the training data set itself; the goal is to build a statistical model of the process which generates the data. This is required to have a good generalization performance. No model is perfect because each model has bias and variance. Bias measures the accuracy or quality of the algorithm (high bias means poor match). On the other hand, variance measures the precision or the specificity of the match (high variance means a weak match). We would like to minimize both of these (bias & variance). Unfortunately, we can't do this independently because as bias (variance) goes down, variance (bias) goes up. Figure 3.3 demonstrates the bias variance tradeoff. The goal is to minimize the training error in real life, which is shown as a dashed line. On the X-axis is a measure of model complexity. Simple models such as linear models have high bias and low variance. They have high bias because no matter what the training data are, they always model it as a line. They have low variance because if two linear models were trained on two different datasets taken from the same distribution, the two resulting linear models would be almost the same. On the other hand, complex models have low bias and high variance. A 7^{th} order polynomial is an example of a complex model. Such a model has low bias because it can model data that look like a line, a quadratic, or even more complex shapes (it is not "biased" toward a pattern). A 7^{th} order polynomial has high variance, however, because slight changes to the training data it is given to model can have large changes in the curve drawn to fit the data. That is, a complex model is more sensitive (variable) to outliers.

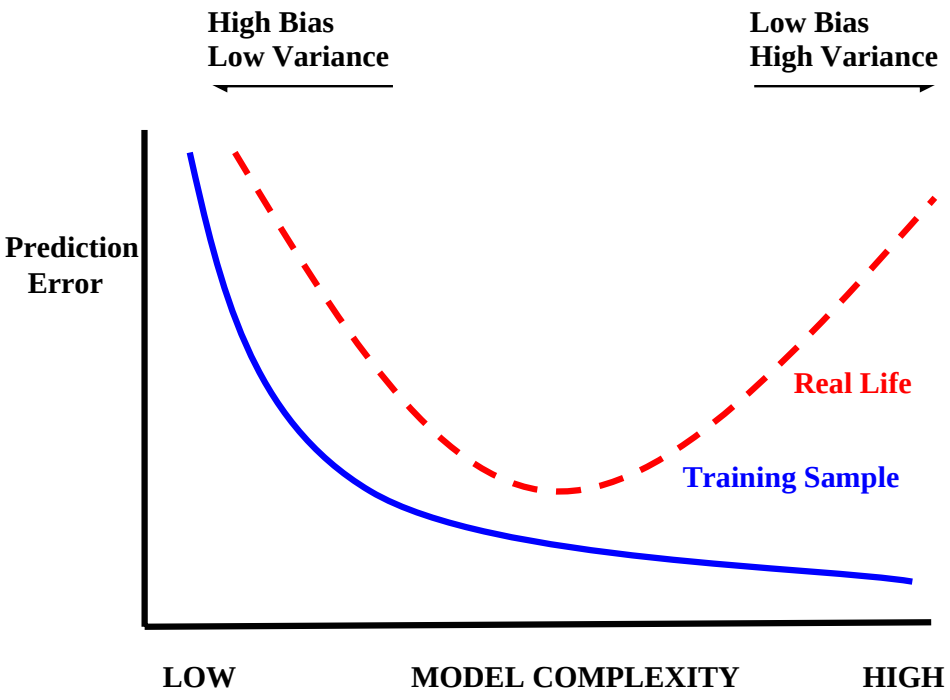

Figure 3.3: Bias – Variance Tradeoff

Figure 3.3 shows that a near-perfect fit of the training data can be found simply by considering more complex models, but this does not imply good performance in real life. As model complexity grows excessively high, the model starts to memorize the training data. Because the training data has noise, the model learns the noise and idiosyncrasies of that data, which hurts the model's performance when it must generalize to data it has never seen in real life. Finding the best machine learning model for a particular data set (i.e., finding the optimal bias/variance tradeoff) is an iterative process.

In the development of the new broadcast protocols that are proposed in this thesis, models are created by a naive Bayes classifier (Bayesian learning algorithm in Chapter 4) and AdaBoost (a Boosting technique in Chapter 5). The naive Bayes is a simple model (with a linear regression), which is not sensitive to the given data, so it has a low variance and high bias. On the other hand, AdaBoost works on simple

models and has high bias and low variance. It reduces the variance and also eliminates the effect of high bias of the weak learner.

Chapter 4

DEVELOPMENT OF THE NAIVE BAYES BROADCAST PROTOCOL

4.1 Introduction

The Naive Bayes Broadcast Protocol is one of the new broadcast protocols that are proposed in this thesis. This protocol uses a supervised learning classifier, which is a *naive Bayes classifier*, to make optimal classifications ("Rebroadcast" or "Drop"). Since naive Bayes is a Bayesian learning algorithm, before we discuss the naive Bayes approach to broadcasting in a MANET, it will be helpful to explain Bayesian learning in the following section.

4.2 Bayesian Learning

Bayesian learning algorithms calculate explicit probabilities for hypotheses. In our broadcast application, we consider two hypotheses: whether to rebroadcast or drop a received packet. Each hypothesis has a certain probability of being correct and each mobile node estimates that probability for every packet received. The characteristics, or features, of a packet influence the estimated probability of a hypothesis being correct through Bayes' theorem. The goal in applying Bayesian learning algorithms is finding the hypotheses with the highest probability of being correct.

4.2.1 Bayes' Theorem

To define Bayes theorem precisely, we introduce the following notation.

P = probability distribution.

h = a hypothesis. In our application, *h* can be take on two values: "Rebroadcast" or "Drop", which we abbreviate as $\oplus$ and $\ominus$.

D = a data set. The data set, *D*, contains the information known about a broadcast packet.

P (*h*), Prior Probability

The initial probability of hypothesis *h* being correct before any data has been observed. Because there are only two hypotheses in this application, two priors are estimated, P ($h = \oplus$) and P ($h = \ominus$), which must sum to one.

P (*D*), Evidence

The prior probability of data set *D* being observed. In our application, this is the probability of receiving a broadcast packet with the characteristics of *D*.

P (*D* | *h*), Likelihood

The probability of observing data set *D*, under the assumption of *h* being correct.

P (*h* | *D*), Posterior Probability

The probability of hypothesis *h* being correct, given a specific data set *D*, which reflects our confidence that *h* is the correct action after the data set *D* has been seen.

Bayes' Theorem

Bayes' theorem can be derived starting from the joint probability, P (*h*, *D*) which is the probability of *h* and *D* occuring at the same time:

$$P(h, D) = P(h, D)$$
$$P(h \mid D)\, P(D) = P(D \mid h)\, P(h) \qquad (4.1)$$

which means that the joint probability is equal to a conditional probability times the prior of the variable conditioned on. Dividing both sides of (4.1) by P (*D*) yields Bayes' theorem:

Posterior Probability ⟵ Likelihood

$$P(h \mid D) = \frac{P(D \mid h)\, P(h)}{P(D)} \qquad (4.2)$$

$P(D \mid h)\, P(h)$ ⇨ Prior Probability

$P(D)$ ⇨ Evidence

Bayes' Theorem is useful because it is a method to calculate posterior probabilities. In our broadcasting application, a node receives a broadcast packet with characteristics D, and must decide whether to rebroadcast or drop the packet. We treat this problem as estimating the posterior probabilities $P(h = \oplus \mid D)$ and $P(h = \ominus \mid D)$ (which must sum to one, just as the prior probabilities do), and choosing the hypothesis with the higher value.

4.2.3 Choosing the Most Probable Hypothesis

Computing the posterior probabilities for the broadcast problem involves the following two computations:

$$P(h = \oplus \mid D) = \frac{P(D \mid h = \oplus)\, P(h = \oplus)}{P(D)} \qquad (4.3)$$

$$P(h = \ominus \mid D) = \frac{P(D \mid h = \ominus)\, P(h = \ominus)}{P(D)} \qquad (4.4)$$

Because a mobile node needs only to compute which hypothesis is bigger, rebroadcast or drop, a simplification can be made. Noticing that $P(h = \oplus \mid D)$ and $P(h = \ominus \mid D)$ have the same denominator, a mobile node can compute only the numerators of (4.3) and (4.4):

$$P(h = \oplus \mid D) \approx P(D \mid h = \oplus)\, P(\oplus) \tag{4.5}$$

$$P(h = \ominus \mid D) \approx P(D \mid h = \ominus)\, P(\ominus) \tag{4.6}$$

And choose the hypotheses, rebroadcast or drop, which yields the higher value among (4.5) and (4.6). This is a significant savings because the probability of receiving one particular packet, P (*D*), is difficult to estimate unless a large number of packets have been received.

In subsequence discussion, we will omit the duplication of writing equations for P ($h = \oplus \mid D$) and P ($h = \ominus \mid D$) and simply write equations for P ($h \mid D$) with the understanding that the equation must be computed for both the rebroadcast and drop hypotheses.

4.3 Naive Bayes Approach to Broadcasting

The use of the Bayesian approach during the development of the broadcasting algorithm based on machine learning methodology involves a choice of Bayes learner (which is commonly referred to as *naive Bayes classifier*) as a learner (classifier). In the naive Bayes approach, one collects retransmit data to build the naive Bayes model as shown in Figure 4.1. The naive Bayes models are special cases of Bayesian networks, consisting of one parent node with the hypothesis and several children nodes that make up the input features of the data.

Figure 4.1 shows the components (features) of the naive Bayes model of successful rebroadcast. Circles represent random variables and arrows denote conditional independence relationships. (Arrows do not show the direction of information flow.) Inference in a Bayesian model is the process of estimating the unknown values of the unshaded circles with respect to the known shaded circles.

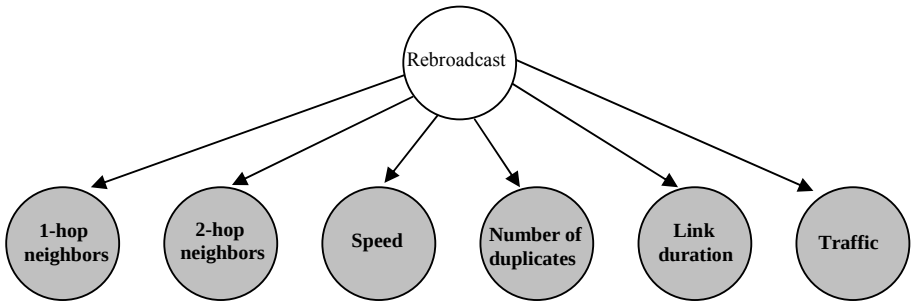

Figure 4.1: Naive Bayes Model Components (Features) for Broadcasting Algorithm

As shown in Figure 4.1, the hypothesis, h, is the unshaded, parent circle, and the features that make up D are shown as the shaded, children circles. We denote the individual features of D as (d_1, d_2, …, dn). The naive Bayes assumption is that the features of D, the d_i's, are conditionally independent. This is denoted in Figure 4.1 by the configuration of the arrows. In Bayesian networks, two variables are conditionally independent if there is not a directed path between them. In the graph in Figure 4.1, there is no way to follow the arrows from one shaded circle to another. We will use the naive Bayes assumption to simplify our calculations of the posterior probability, P (h | D), given in equation (4.5). Writing D in the terms of its features, equation (4.5) becomes

$$P(h \mid D) \approx P(d_1, d_2, \ldots, d_n \mid h)\, P(h) \tag{4.7}$$

Using the naive Bayes assumption, the features of D are conditionally independent, so their joint probability is equal to their product (conditional on h). That is,

$$P(d_1, d_2, \ldots, d_n \mid h) \overset{N.B.}{=} P(d_1 \mid h)\, P(d_2 \mid h) \ldots P(d_n \mid h) \tag{4.8}$$

Substituting (4.7) into (4.8) yields

$$P(h \mid D) \approx P(d_1 \mid h)\, P(d_2 \mid h) \ldots P(d_n \mid h)\, P(h)$$

$$\approx P(h) \prod_{i=1}^{n} P(d_i \mid h) \qquad (4.9)$$

Mobile nodes use equation (4.9) in our Naive Bayes broadcast protocol to calculate the posterior probabilities of rebroadcast and drop. Equation 4.9 and the naive Bayes assumption used to derive it are useful because it allows the likelihood terms for each input feature, $P(d_i \mid h)$, to be calculated independently. By calculating these terms independently, they can be calculated accurately using less data than would be required to calculate $P(d_1, d_2, \ldots, d_n \mid h)$ as one term.

Even when the naive Bayes assumption is violated (and the posterior probability estimates are wrong), there are conditions under which naive Bayes classifiers can output optimal classifications (rebroadcast or drop) [43]. The quantity, $P(h)$, the prior probability of h (h = rebroadcast or drop), is easy to estimate by counting the frequency with which each value of h occurs in the given training data. To estimate $P(d_i = x \mid h = rebroadcast)$, we make a subset of the training data for which rebroadcast is true and count the number of times d_i takes on the value of x.

We choose the input features for each broadcast packet based on our experience and consider the small amount of data that each MN has available to it. The features we found most useful are shown in Figure 4.1. For each input feature (e.g., speed, number of 1-hop neighbors, etc.), every MN maintains two tables: one conditional on rebroadcast and one conditional on drop. The parent stores one table: prior probabilities of rebroadcast and drop. The MN estimates density, congestion, and speed at the same time a packet is received. Deciding whether to rebroadcast or drop a packet is simple: (4.9) is evaluated once for the $\oplus$ (rebroadcast) class and once for the $\ominus$ (drop) class and an MN makes its decision based on which is bigger. Evaluating (4.9) for our model in Figure 4.1 requires seven table look-ups and six multiplications. The tabular data structures and threshold decision procedure (rebroadcast or drop) require less storage and computation than SBA and AHBP-EX, which both use algorithms based on graph theory.

An attractive feature of the naive Bayes model is that the likelihood entries, e.g., p (1-hop neighbors | rebroadcast), p (speed | rebroadcast), can be used to answer questions in a post hoc manner. For example, given that node *A* infers that the retransmission of packet *X* was unsuccessful, which hypothesis can best explain why? Candidate hypothesis include: (1) The node speed was so high that node *A* was out of transmission range before it could hear packet *X* being rebroadcast; (2) The congestion was so high that (a) there was a collision or (b) the neighbors already got packet *X* from another node; (3) The node density was so low that no neighbors were in range that needed the packet *X*. The hypothesis with the maximum likelihood dictates how the MN should adapt.

4.4 Demonstration of Naive Bayes Approach to Broadcasting

Let us consider a case where input features for each broadcast packet are receiver node's speed, receiver node's number of neighbors (1-hop), and the traffic. A node has a data set presented in Table 4.1 with 10 examples. Instead of numerical values for speed, number of neighbors, and traffic, we simplify the table with "High" and "Low" entries for illustration purposes. Suppose that the node has to classify a new incoming packet with the input features high speed, low number of neighbors, and low traffic.

By applying the naive Bayes classifier we discussed in the previous section, we will select the most likely classification by given the attribute values (d_1 = speed, d_2 = # of neighbors, and d_3 = traffic). If the hypothesis is to rebroadcast the packet, then h = Yes; if the hypothesis is to drop the packet, then h = No. That is, we calculate

$$P(h = \text{Yes} \mid d_1 = \text{High}, d_2 = \text{Low}, d_3 = \text{Low})$$

and

$$P(h = \text{No} \mid d_1 = \text{High}, d_2 = \text{Low}, d_3 = \text{Low})$$

Using equation (4.9), these calculations are

$$P(h = \text{Yes} \mid d_1 = \text{High}, d_2 = \text{Low}, d_3 = \text{Low})$$
$$= P(d_1 = \text{High} \mid h = \text{Yes})\, P(d_2 = \text{Low} \mid h = \text{Yes})\, P(d_3 = \text{Low} \mid h = \text{Yes})\, P(h = \text{Yes})$$

and

$$P(h = \text{No} \mid d_1 = \text{High}, d_2 = \text{Low}, d_3 = \text{Low})$$
$$= P(d_1 = \text{High} \mid h = \text{No})\, P(d_2 = \text{Low} \mid h = \text{No})\, P(d_3 = \text{Low} \mid h = \text{No})\, P(h = \text{No})$$

So we must calculate two posterior values and six likelihood values in total. As mentioned in the previous section, the prior probabilities are easy to estimate by counting the frequency with which each hypothesis occurs in the training data. In our example, $P(h = \text{Yes}) = 3/10 = 0.3$ and $P(h = \text{No}) = 7/10 = 0.7$.

Example No.	**Speed**	**# of Neighbors**	**Traffic**	**Rebroadcast?**
1	High	High	Low	No
2	High	High	Low	Yes
3	High	High	Low	Yes
4	High	High	High	No
5	High	Low	High	No
6	Low	Low	High	No
7	Low	Low	High	No
8	Low	Low	Low	Yes
9	Low	High	Low	No
10	Low	High	High	No

Table 4.1: The Created Dataset for the Naive Bayes Broadcasting Scenario

The likelihoods are also estimated by counts to estimate $P(d_i = \text{High} \mid h = \text{Yes})$, we use this equation:

$$P(d_i = \text{High} \mid h = \text{Yes}) = \frac{C}{N} \qquad (4.10)$$

N = Number of training examples for which the hypothesis is to rebroadcast.

C = Count of training examples for which d_i = High and h = Yes.

P (d_1 = High $\mid h$ = Yes)

We have three cases where h = Yes and in two of those cases (h = Yes), d_1 = High Speed, so for P (d_1 = High $\mid h$ = Yes), C = 2, N = 3. Thus,

$$P(d_1 = \text{High} \mid h = \text{Yes}) = \frac{2}{3} = 0.67$$

P (d_1 = High $\mid h$ = No)

We have seven cases where h = No and in three of those cases (h = No), d_1 = High Speed, so for P (d_1 = High $\mid h$ = Yes), C = 3, N = 7. Thus,

$$P(d_1 = \text{High} \mid h = \text{No}) = \frac{3}{7} = 0.43$$

P (d_2 = Low $\mid h$ = Yes)

We have three cases where h = Yes and in one of those cases (h = Yes), d_2 = Low # of Neighbors, so for P (d_2 = Low $\mid h$ = Yes), C = 1, N = 3. Thus,

$$P(d_2 = \text{Low} \mid h = \text{Yes}) = \frac{1}{3} = 0.33$$

P (d_2 = Low $\mid h$ = No)

We have seven cases where h = No and in three of those cases (h = No), d_2 = Low # of Neighbors, so for P (d_2 = Low $\mid h$ = No), C = 3, N = 7. Thus,

$$P(d_2 = \text{Low} \mid h = \text{No}) = \frac{3}{7} = 0.43$$

P (d_3 = Low $\mid h$ = Yes)

We have three cases where h = Yes and in three of those cases (h = Yes), d_3 = Low Traffic, so for P (d_3 = Low $\mid h$ = Yes), C = 3, N = 3. Thus,

$$P(d_3 = \text{Low} \mid h = \text{Yes}) = \frac{3}{3} = 1$$

P (d_3 = Low $\mid h$ = No)

We have seven cases where h = No and in two of those cases (h = No), d_3 = Low Traffic, so for P (d_3 = Low $\mid h$ = No), C = 2, N = 7. Thus,

$$P(d_3 = \text{Low} \mid h = \text{No}) = \frac{2}{7} = 0.28$$

Now we apply the naive Bayes equation (4.9) for h =Yes and for h = No by using the probability values previously computed:

$$\begin{aligned} &P(h = \text{Yes} \mid d_1 = \text{High}, d_2 = \text{Low}, d_3 = \text{Low}) \\ &\quad \approx P(h = \text{Yes}) * P(d_1 = \text{High} \mid h = \text{Yes}) \\ &\qquad * P(d_2 = \text{Low} \mid h = \text{Yes}) \\ &\qquad * P(d_3 = \text{Low} \mid h = \text{Yes}) \\ &\quad \approx 0.3 * 0.67 * 0.33 * 1 \end{aligned}$$

$\approx$ 0.066

P (h = No $\mid$ d_1 = High, d_2 = Low, d_3 = Low)

$\approx$ P (h = No) * P (d_1 = High $\mid h$ = No)

* P (d_2 = Low $\mid h$ = No)

* P (d_3 = Low $\mid h$ = No)

$\approx$ 0.7 * 0.43 * 0.43 * 0.28

$\approx$ 0.036

So a scenario of High Speed, Low Number of Neighbors and Low Traffic is classified as 'Yes', since 0.066 > 0.036.

Even though we never saw a packet during a secenario of d_1 = High, d_2 = Low, and d_3 = Low, we were able to generate from our experience. We simulate the Naive Bayes protocol we developed and compare it to previously proposed broadcast protocols, Flooding, Adaptive SBA, and AHBP-EX, over a range of network conditions in Chapter 6.

Chapter 5

DEVELOPMENT OF THE ADABOOST BROADCAST PROTOCOL

5.1 Introduction

The AdaBoost broadcast protocol is the second new broadcast protocol proposed in this thesis. This protocol uses a machine learning classifier known as adaptive boosting (AdaBoost) [48] to make classifications (rebroadcast or drop). Before we discuss AdaBoost approach to broadcasting in a MANET, we explain boosting in the next section.

5.2 The Boosting Approach to Machine Learning

One of the most popular methods for creating *ensembles* is boosting [48]. It is a recent method for improving the predictive power (accuracy) of any given learning algorithm. An ensemble is a collection of simple machine learning models. Each member of the ensemble is too simple to learn the entire training data set; instead each one becomes an expert on part of the data set, and when their predictions are combined, they are more accurate than any individual model. This behavior has earned ensembles the alternate name of *committee machines*. Intuitively, it is easier to find several, simple hypothesis than a simple, highly-accurate hypothesis that fits the entire training set. The only requirement on these simple hypotheses, called *base* hypotheses, is that they perform better than random guessing. That is, their hypothesis (rebroadcast or drop) is correct more than 50% of the time.

Boosting is based on the observation that finding many base prediction rules can be significantly easier than finding a single, highly accurate prediction rule. Boosting algorithm includes a “base” learning algorithm that predicts slightly better than

random guessing (better than 50%). This base algorithm is called repeatedly by feeding different weights over the given training examples; each time the base algorithm is called, it generates a base prediction rule.

There are three fundamental differences between boosting algorithms and traditional machine learning methods, such as the naive Bayes algorithm described in the previous chapter.

Weighting over the training examples: Each example in the training set has a weight associated with it, such that the sum of all the weights equal to one. Each example starts with equal weight and after each round, the misclassified examples get more weights to force the next hypothesis to focus on them (which are harder examples to learn) in the next round.

Choosing base learner algorithm: Technically, any machine learning algorithm that performs better than random guessing can act as a base learner, but a good base learner should:

- Have flexible features that can be correlated with most conceivable relations between feature vector and label.
- Avoid over-fitting.
- Allow the minimal weighted training error.
- Calculate predicted label very efficiently.

Combining base rules into a single strong rule: In addition to the training set examples having weights, the base learners themselves are assigned model weights in proportion to how accurate they are. These model weights are on a different scale and do not have to sum to one. Weighted majority vote is taken to combine the base rules into a single strong rule.

Weighted Majority:

- Learner 1 predicts “Rebroadcast” with weight 0.3
- Learner 2 predicts “Drop” with weight 0.5

- Learner 3 predicts “Rebroadcast” with weight 0.4

Weighted Majority Vote: Rebroadcast

As a specific boosting algorithm, we chose the AdaBoost method to implement in our boosting approach to broadcasting. Before we discuss the AdaBoost approach to broadcasting in a MANET, we explain our chosen base learner, which is a Decision Stump, in the next section.

5.3 Decision Stumps

Decision stumps can perform a single test on a single input feature. That is, it asks one question about training example *D*, and based on the answer to that question, it provides its hypothesis (rebroadcast or drop). It can be called a decision tree with only a root node because it has only one level of branching. The input feature, d_i, (e.g., # of 1-hop neighbors, speed, traffic, etc.) that is selected in decision stump is the most useful one for classifying the examples. This is measured by a statistical property called *information gain*. It is measured as the difference in entropy of the data set, *D*, before and after splitting it into subsets based on every training example’s value of d_i.

$$Gain\ (D, di) \equiv \underbrace{Entropy\ (D)}_{\textbf{Entropy Before}} - \underbrace{\sum_{v=Values(di)} \frac{|v|}{|D|}\ Entropy\ (v)}_{\textbf{Entropy After}} \qquad (5.1)$$

Entropy Before: The randomness of the training data set *D*.

Entropy After: The randomness of the subsets of the training data after partitioning the examples based on their values of d_i. The randomness of the subsets is summed as a weighted sum, according to the relative size of the subsets.

Information gain is the expected reduction in entropy, which characterizes the randomness of the set of training examples. Randomness in this context is measured as the mix of hypothesis (rebroadcast or drop) in the training set. Formally the entropy of the training set is defined as

$$Entropy\ (D) \equiv -p_{(+)} log_2 p_{(+)} - p_{(-)} log_2 p_{(-)} \qquad (5.2)$$

In equation (5.2), $p_{(+)}$ denotes to the proportion of positive examples (rebroadcast) in D and $p_{(-)}$ denotes to the proportion of negative examples (drop) in D. Because each of the training examples in D has a weight associated with it, it is these weights that are accumulated to calculate $p_{(+)}$ *and* $p_{(-)}$ That is, the weights of all the positive examples are added up to form $p_{(+)}$, rather than just counting all the positive examples.

$$w^+ = \sum_{i=1}^{|D|} \begin{cases} w_i \text{ if example } i \text{ is } + \\ 0 \text{ otherwise} \end{cases} \qquad (5.3a)$$

$$w^- = \sum_{i=1}^{|D|} \begin{cases} w_i \text{ if example } i \text{ is } - \\ 0 \text{ otherwise} \end{cases} \qquad (5.3b)$$

$$p_{(+)} = \frac{w^+}{w^+ + w^-} \qquad (5.4a)$$

$$p_{(-)} = \frac{w^-}{w^+ + w^-} \qquad (5.4b)$$

Similarly, the size of each subset, |V|, is computed by adding up the weights of the examples in the subset, not by counting the number of examples in the subset.

5.4 Adaptive Boosting (AdaBoost) Approach to Broadcasting

In our boosting approach to broadcasting, we chose the AdaBoost algorithm to output class labels (rebroadcast or drop). The AdaBoost algorithm was introduced in

1995 by Freund and Schapire [25]. AdaBoost is an iterative algorithm in which a base learner is trained to be an expert on those examples that previous learners got wrong. The iterative nature of the algorithm is shown in Figure 5.1.

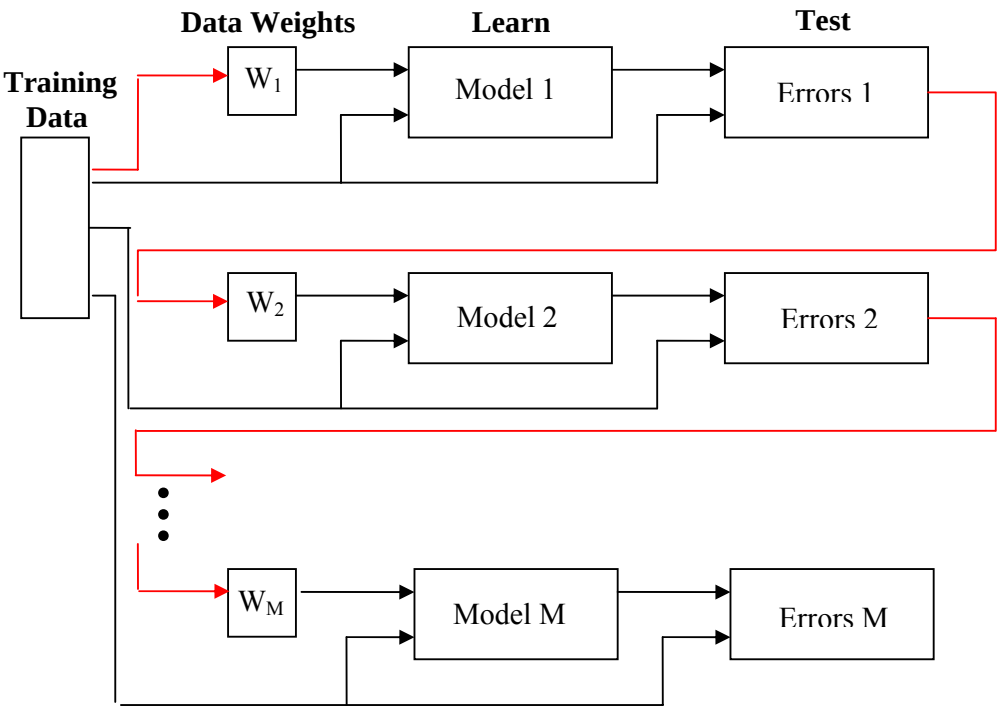

Figure 5.1: AdaBoost Training [47]

AdaBoost calls a given base learning algorithm (a decision stump in our implementation) repeatedly, creating an ensemble of models m = 1, …, M. One of the main ideas of the algorithm is to maintain a set of data weights over the training set; in each iteration, the base learner (decision stump) makes predictions based on slightly altered data (updated weighting over training set). After all the models are created, each model is given a model weight in proportion to its accuracy. To classify a new packet, the ensemble takes a weighted vote of all the models' hypotheses.

In the implementation of our AdaBoost approach to broadcasting, there are two phases. First phase is the training, second phase is the classification.

Phase I: Training the AdaBoost Broadcast Protocol

We collect the same data to build decision stumps as we do for the naive Bayes model in Section 4.3. An example dataset is shown in Figure 5.2 (many values are omitted for clarity). The decision stump chooses an input feature that best splits the data (i.e., the d_i that yields the highest information gain).

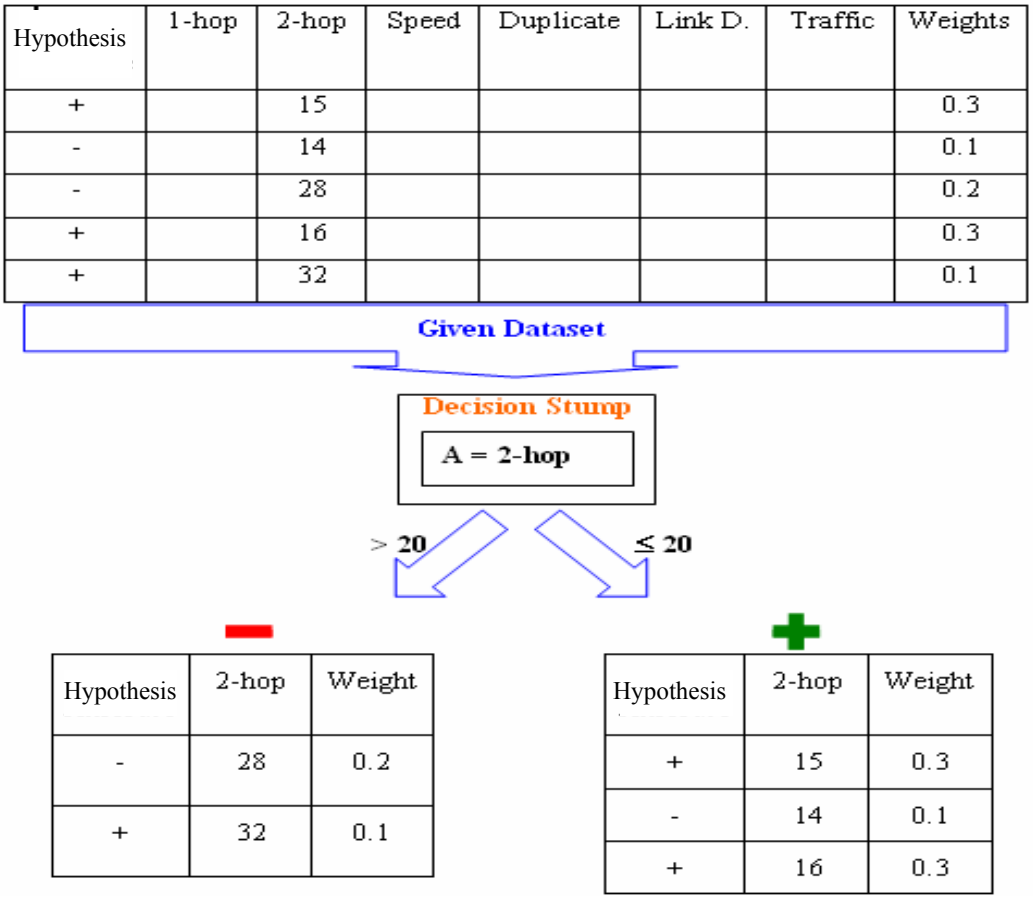

Figure 5.2: Base Model (Decision Stump)

In Figure 5.2, the decision stump chose to split the data on the number of 2-hop neighbors, yielding two subsets. In the left subset, the weight of the negative examples is greater than the weight of the positive examples, so all the examples in the left subset are classified as drop (yielding one error). Similarly, all the examples in the right subset are labeled rebroadcast (also yielding one error).

Notice that in this example, some weights are higher than others, which means that those examples with higher weight were previously misclassified by an earlier decision stump. Because of the way input features are chosen by weighted information gain (equation (5.1)), the current decision stump is encouraged to classify those highly-weighted examples correctly (which it does in this example). Initially, of course, all the data weights are equal (and constrained such that their sum is one).

Then for each model generation:

- We apply the learning algorithm (decision stump explained in the previous section) to weighted dataset. The decision stump picks one of the attributes (as shown in Figure 5.2) that has the highest gain and classifies by splitting on that attribute (e.g., speed, traffic and, etc.). The resulting model is saved.

- We compute error rate *e* (a fraction between 0 and 1) of the model on weighted data set and store it. If a classifier performs well on the weighted data then *e* is close to 0, otherwise e is close to 0.5. The error rate is the sum of all the weights of misclassified examples.

- If *e* is equal to zero (decision stump classified (rebroadcast or not) all of the examples correctly), or e is greater or equal to 0.5, we terminate model generation.

- Else
    - For each example in the dataset:
        - If an example is classified (rebroadcast or drop) correctly:

- Multiply the weight of the example by e / (1-e)

- Renormalize the data weights such that they sum to one.

Because the quantity e / (1-e) is always between zero and one, the weight of correctly classified examples goes down. After the subsequent normalization, the weight of misclassified examples goes up. The extra weight on misclassified examples will give extra incentive to the next decision stump to classify those "harder" examples correctly.

Phase II: Classification

The AdaBoost approach to classifying an incoming broadcast packet is according to the weighted majority of classifiers (models) created during the training. A new received broadcast packet includes the same features (e.g., speed, traffic, and so on) that each example in training data includes. So, each of the M models calculates a hypothesis (rebroadcast or drop) for that packet and how much an individual hypothesis contributes to the ensemble depends on the model weights. More accurate models have a higher model weight. During the training, we store the error rate of each model, which is used to calculate each model's weight (shown in equation (5.5)):

Model weight:

$$\text{Model weight} = -\ln \frac{e}{1-e} \tag{5.5}$$

After each model makes a prediction for the incoming broadcast packet, it votes with its model weight. The combination of the predictions into a single prediction is made by taking a weighted majority vote. The weighted majority vote's result makes the decision whether to rebroadcast an incoming broadcast packet.

Ideally each boosting base learner becomes a specialist in a part of the domain and complements the others rather than duplicating them. AdaBoost forces new models to become experts for instances classified incorrectly by previous ones. Each expert proposes a candidate hypothesis by covering a different part of the input space to explain why the transmission of a packet is successful or unsuccessful, and the combination of weighted hypotheses dictates how the MN should adapt.

To understand the application of the AdaBoost approach to improve broadcasting, an example of a broadcasting scenario is created based on a small data set. This example, presented in the next section, demonstrates how the AdaBoost classifier can be used to output optimal classifications (rebroadcast or drop).

5.5 Demonstration of AdaBoost Approach to Broadcasting

Consider a case in which each node has a small data set that includes 10 examples. Nodes will be trained on this dataset and will be tested on the same dataset. In this example, we demonstrate the combination of base predictions into one strong prediction to make the decision of whether to rebroadcast an incoming broadcast packet. To keep the example simple, we omit the details (calculations) of how our base learner (decision stump) predicts the outputs. All we need to know about a base learner is, since it is a simple model, it outputs a straight line because it is not sensitive to the given training set (as we discussed in Chapter 3, high bias / low variance).

In the training set given in Table 5.1, each example is assigned hypothesis “+” if the example was rebroadcasted, otherwise it is assigned “-” if it was dropped. According to our algorithm discussed in the previous section, equal weight is initially assigned to each example in the data set.

Example	Hypothesis	Weight of Example	Example	Hypothesis	Weight Of Example
1	-	0.1	**6**	-	0.1
2	-	0.1	**7**	-	0.1
3	-	0.1	**8**	+	0.1
4	+	0.1	**9**	+	0.1
5	+	0.1	**10**	+	0.1

Table 5.1: The Created Dataset the AdaBoost Broadcasting Scenario

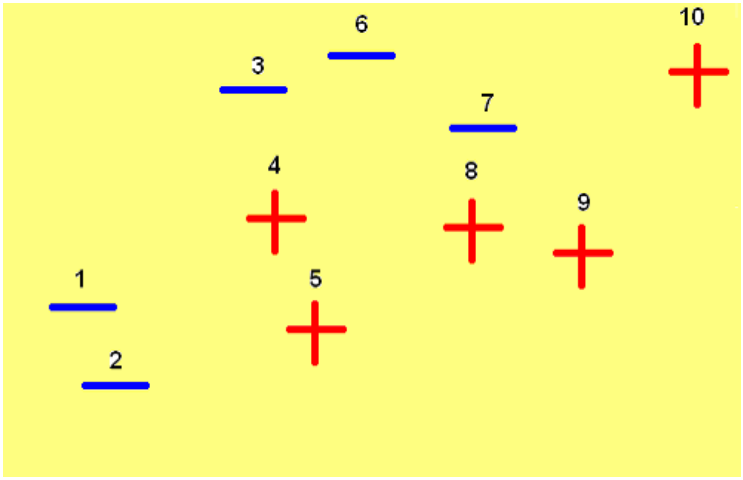

Figure 5.3: AdaBoost Example (Hypothesis Space)

According to the input features (numerical values not shown), if we consider them on a hypothesis space, they will be located as shown in Figure 5.3. Based on the algorithm discussed in the previous section, we will iteratively calculate different weighting over the preceding data set, creating three models. In the first round, the initial dataset will be used by our simple model with the assigned weights, and our

model will predict a straight line according to the given dataset. Then, we will test our model's prediction on the same given dataset and calculate its error rate, which will be used to calculate the model's weight and to update each example's weight in the dataset.

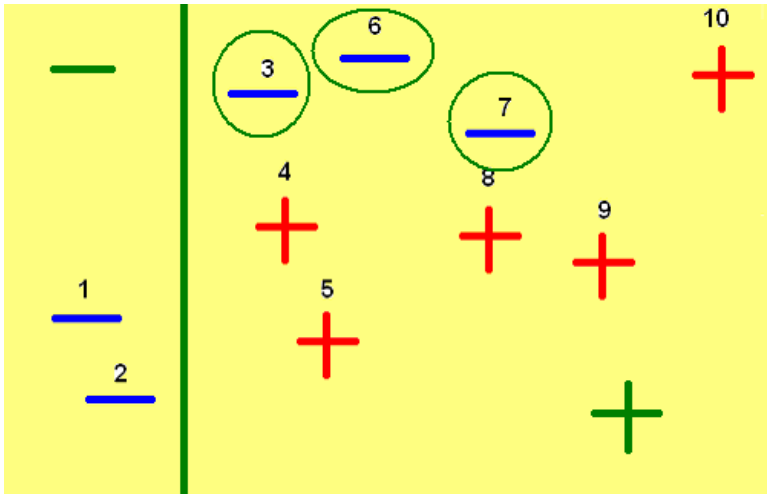

Figure 5.4: AdaBoost Model 1's Output

In the first iteration, the decision stump picks an input feature (not shown) that results in the data split into two subsets shown in Figure 5.4. Examples to the left of the line are classified as drop, and examples to the right of the line are classified as rebroadcast. According to the prediction, the error rate of the model is evaluated based on the assigned weights. In this case, as it is shown in Figure 5.4, packets 3, 6, and 7 are misclassified. They were supposed to be classified as drop, but were classified as rebroadcast. Then the error rate of the first model can be calculated as e_1 = 0.1 + 0.1 + 0.1 = 0.3. We multiply the correctly classified examples with (e_1 / 1- e_1) and normalize them. After updating weights and normalization, the new updated weights are assigned to the packet (see Table 5.2).

Based on the algorithm we discussed in Section 5.4, the model weight is also calculated. As it is given in the equation (5.5), the model weights are evaluated by using the error rate we calculated before. In our first iteration, the model weight may be calculated as:

Model weight 1 = $-\ln e_1/ (1-e_1) = -\ln 0.3/(1-0.3) \approx 0.84$.

Example	Target Attribute	Updated Weights	Example	Target Attribute	Updated Weights
1	-	0.07	**6**	-	0.17
2	-	0.07	**7**	-	0.17
3	-	0.17	**8**	+	0.07
4	+	0.07	**9**	+	0.07
5	+	0.07	**10**	+	0.07

Table 5.2: Dataset with Updated Weights after Iteration 1

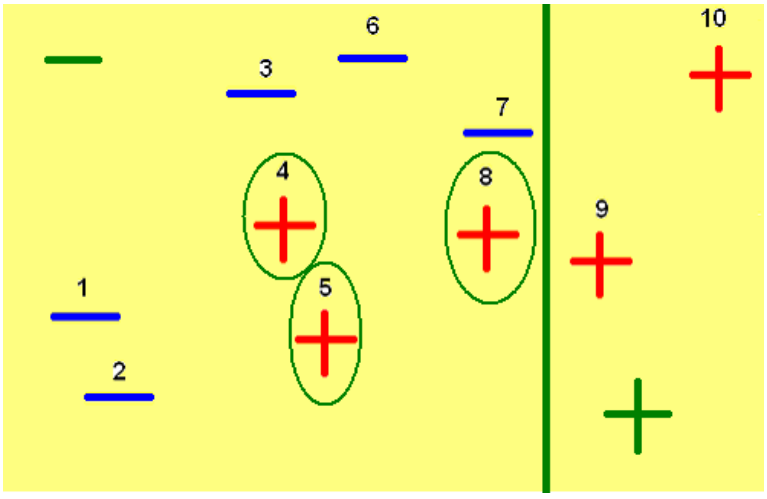

Figure 5.5: AdaBoost Model 2's Output

In Table 5.2, notice that the data weight of examples 3, 6, and 7 have gone up, and the others have consequently gone down.

In the second iteration, we use the updated dataset to build another model, and the same calculations (error rate, model weight) are done corresponding to the prediction of the model. As shown in Figure 5.5, this time (based on the updated weighting over the data set) the model classifies more of the hypothesis space as drop (all examples left of the line), and misclassifies three packets (4, 5, and 8). The model's error rate in this round is $e_2 = 0.21$ (based on the updated weighting over the dataset). According to our algorithm we update the data weights and normalize them (see Table 5.3). Using the error rate, the model weight is calculated to be 1.32. From the equation (5.5), notice that the second model has a higher model weight because it had a lower weighted error.

Example	Target Attribute	Updated Weights	Example	Target Attribute	Updated Weights
1	-	0.044	**6**	-	0.107
2	-	0.044	**7**	-	0.107
3	-	0.107	**8**	+	0.168
4	+	0.168	**9**	+	0.044
5	+	0.168	**10**	+	0.044

Table 5.3: Updated Weights after Iteration 2

According to the given preceding data set, the third model also processes the same steps and predicts an output (shown in Figure 5.6). The error rate of the third model is $e_3 = 0.132$ and the model weight is 1.88.

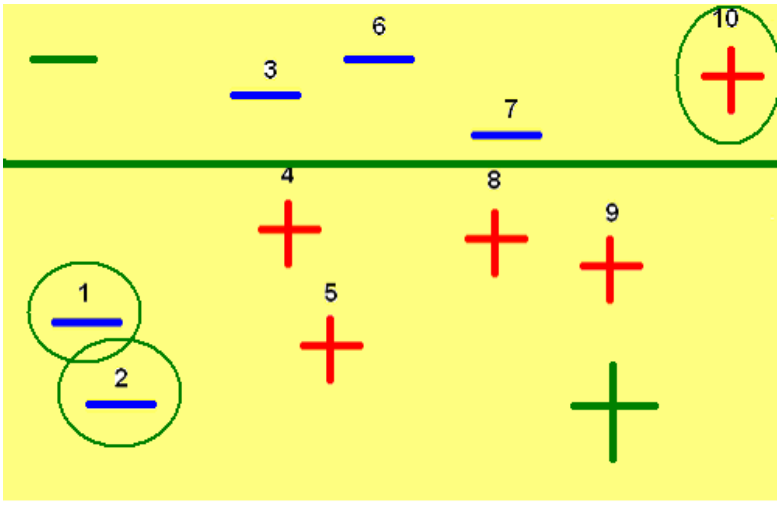

Figure 5.6: AdaBoost Model 3's Output

AdaBoost Classification:

Based on the preceding created models, now we classify the examples in the given data set by combining the three predictions. As we explained in Section 5.4, the predictions are combined by taking the weighted majority vote. Since we have three models, each model will vote with their calculated model weights and the weighted majority will be our AdaBoost protocols decision (shown in Table 5.4 and Figure 5.7).

Example	Model 1's Prediction	Model 1's Weight	Model 2's Prediction	Model 2's Weight	Model 3's Prediction	Model 3's Weight	AdaBoost Prediction
1	-	0.84	-	1.32	+	1.88	-
2	-	0.84	-	1.32	+	1.88	-
3	+	0.84	-	1.32	-	1.88	-
4	+	0.84	-	1.32	+	1.88	+
5	+	0.84	-	1.32	+	1.88	+
6	+	0.84	-	1.32	-	1.88	-
7	+	0.84	-	1.32	-	1.88	-
8	+	0.84	-	1.32	+	1.88	+
9	+	0.84	+	1.32	+	1.88	+
10	+	0.84	+	1.32	-	1.88	+

Table 5.4: Weighted Majority Vote

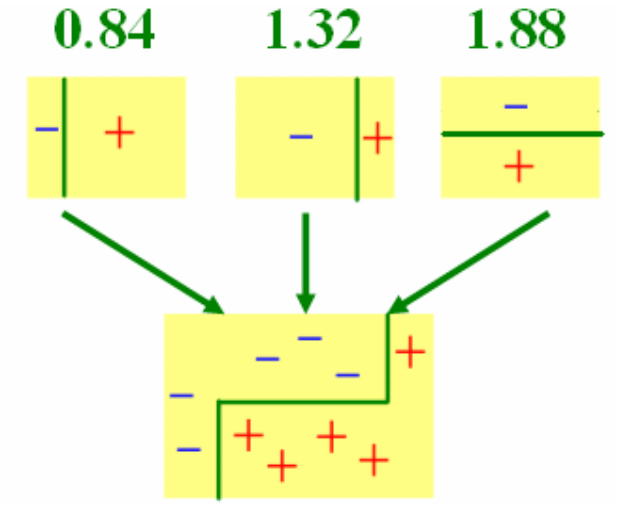

Figure 5.7: AdaBoost Classification

AdaBoost is a more complex algorithm than naive Bayes, and it is less computationally efficient. Nevertheless, it is a flexible model that can find an optimal bias/variance trade-off. As we show in the next chapter, it is capable of accurate broadcast predictions.

Chapter 6

SIMULATION COMPARISON OF NEW BROADCAST PROTOCOLS

6.1 Description of Studies

As we discussed in Chapters 5 and 6, two new machine learning based broadcast protocols are developed in this thesis. In this chapter, we present two different studies used in the development of our protocols. The first study explains the discovery of the training set created for our machine learning models to learn from. The second study compares the developed protocols over a range of network conditions including varying node density, node mobility, and traffic rates. In the second study, we show that the machine learning based broadcast protocols stand out as being appropriate for diverse conditions because of their ability to tune themselves in a systematic, mathematically-principled way.

In our simulation comparisons, we used the same NS-2 (1b-7a) simulation parameters as [15]. The parameters are outlined in Table 6.1. In particular, we report results testing increasing network severity; the nodes move according to the steady-state distribution for the Random Waypoint Mobility Model (see [44] for details). We increase the network severity by changing node density, congestion level, and node mobility simultaneously. This aggregation of the three variables demonstrates how the protocols react in real networks. It allows us to gauge the relative importance of each preceding variable to the overall performance of each protocol.

In each study the nodes are placed in the network area according to [44], and since these nodes are mobile, their number of neighbors keeps changing. Initially, if the average number of neighbors for different number of nodes is evaluated, each node covers approximately 20% of the nodes in the network shown in Table 6.2.

Simulation Parameter	Value
Simulator	NS-2 (1b7a)
Network Area	350 x 350 meter
Node Tx Distance	100 meter
Data Packet Size	64 bytes payload
Node Max. IFQ Length	50
Simulation Time	100 seconds
Number of Trials	10
Confidence Interval	95 %

Table 6.1: Simulation Parameters

Number of Network Nodes	Average Number of Neighbors
20	3.8
30	6.0
40	7.6
50	9.1
60	11.2
70	13.9
90	18.1
110	21.2

Table 6.2: Average Number of Neighbors for Different Number of Nodes

We tested our machine learning based protocols in five different trials shown in Table 6.3. The severity of the network environment is increased as the trial number increases.

Trial	1	2	3	4	5
Number of Nodes	40	50	60	70	90
Average Speed (m/sec)	1	5	10	15	20
Pkt. Src. Rate (pkts/sec)	10	20	40	60	80

Table 6.3: Trial Simulation Parameters

In Trial 1, the network is almost static (because node mobility is very low) and the congestion is very low (because the frequency of packet origination and number node are low). As the trial number increases, the network becomes unstable (dynamic) and gets congested.

As we previously mentioned, first we discuss the study that led to the creation of the training set. Then in Study 2, we evaluate the performance of each developed protocol by comparing them with Simple Flooding, Adaptive SBA, and AHBP-EX.

6.2 Study 1 – Dataset Creation

In Section 4.3 and Section 5.4, we discussed the classification approach to broadcasting with different supervised learning algorithms. In our generated models, some of the input features for each broadcast packet were shown. In this study, we discuss all of the features (attributes) that each input vector includes in our data sets. Choosing attributes and data set creation are one of the most important factors for our machine learning protocols. Adding irrelevant or distracting attributes affects the internal representation of a model and this makes training less efficient.

As discussed in Section 3.2, we use supervised learning models in our broadcast protocols. Supervised learning requires a teacher that knows what the right answer is for a large set of examples. We created training data composed of two parts: input features (i.e., the d_i') and output hypothesis (i.e., rebroadcast or drop).

We believe that we picked highly-predictive features in order to avoid over-fitting the training data and to keep the hypothesis space size small, so that our algorithms operate faster and more effectively. The chosen attributes for our models may be listed in four categories:

1. Neighbor Knowledge

a. Number of 1-hop neighbors: The number of 1-hop neighbors that a node has when it makes the decision whether to rebroadcast.

b. Number of unique 2-hop neighbors: The number of unique 2-hop neighbors that a node has when it makes the decision whether to rebroadcast. Unique denotes that the counted 2-hop neighbor can not be counted more than once, if there is a 2-hop link through different neighbors or if it is also a 1-hop neighbor.

c. Number of duplicate 2-hop neighbors: The number of duplicate 2-hop neighbors that a node has when it makes the decision whether to rebroadcast. Duplicate denotes that the counted neighbor might be counted more then once, if there is more then one 2-hop link through different neighbors or if it is also a 1-hop neighbor.

d. Number of all the neighbors: The number of all the neighbors (1-hop and 2-hop) in the neighbor list. All denotes that all the neighbors in the neighbor list are counted without eliminating duplicate 2-hop neighbors (the same neighbor can be counted more than once).

2. Link Duration Knowledge

a. Link change rate: Link change rate is a scan of the neighbor table. If a neighbor is entered to the neighbor table, a timer is set for it. The link duration is calculated by timing the duration of an entry in the neighbor table.

b. Cumulative average link change rate: Cumulative average of the link change rate during the whole simulation run time.

c. Current average link change rate: Average link change rate between the times a specific incoming broadcast packet is received and is decided whether to rebroadcast.

3. Traffic

a. Number of duplicates: The number of received duplicate packets during the RAD.

4. Mobility

a. Speed: Speed of a node when it makes the decision whether to rebroadcast.

The created training sets for our machine learning models, initially have the prior probabilities given in Table 6.4.

	Rebroadcast	**Drop**
Trial 1	31 %	69 %
Trial 2	35 %	65 %
Trial 3	35 %	65 %
Trial 4	44 %	56 %
Trial 5	65 %	35 %

Table 6.4: Initially Assigned Prior Probabilities for Each Trial

After the training datasets are created, our machine learning based protocols are ready to be trained and tested. The next sections present the simulation results of the new broadcast protocols that are trained with the datasets created in this study.

6.3 Study 2 - Simulation Tests

We divided our simulation tests into three subsections (studies). The first two studies show how it is desirable for the classification approached protocols to tune their selves in a mathematically-principled way to adapt in different network conditions. The third study compares the best performances of the proposed protocols (Naive Bayes and AdaBoost) based on machine learning classifiers with the chosen static hand tuned protocols (Flooding, Adaptive SBA, and AHBP-EX).

6.3.1 Study 2.1 – Naive Bayes Broadcast Protocol

The purpose of this study is to evaluate the core algorithm and show the flexibility of naive Bayes broadcast protocol for different network conditions listed in Table 6.3. While we show the naive Bayes' performance, we will compare it with Adaptive-SBA and AHBP-EX.

Before we run our Naive Bayes broadcast protocol with the training sets created in Study 1, we analyzed Adaptive SBA's and AHBP-EX's performances to manipulate the created data sets. According to the previous work [15], Adaptive-SBA is the highest performer through Trial 4, but after that, the congestion appears to catastrophically interfere with its ability to deliver packets. On the other hand, AHBP-EX is the worst performer through the first three trials. However, it degrades more gracefully, and in Trial 5 it outperforms Adaptive SBA by roughly 20%.

The results in [15] show that AHBP-EX performs badly through Trial 4 because it retransmits packets less than it should; in Trial 5, Adaptive SBA breaks down because it retransmits packets much more than it should. Figure 6.1 presents the prior probability of rebroadcasting for a quantity of nodes Adaptive SBA maintains in Trial 5. What the figure shows is that, while some nodes rebroadcast more than others, all these nodes rebroadcast too often.

.

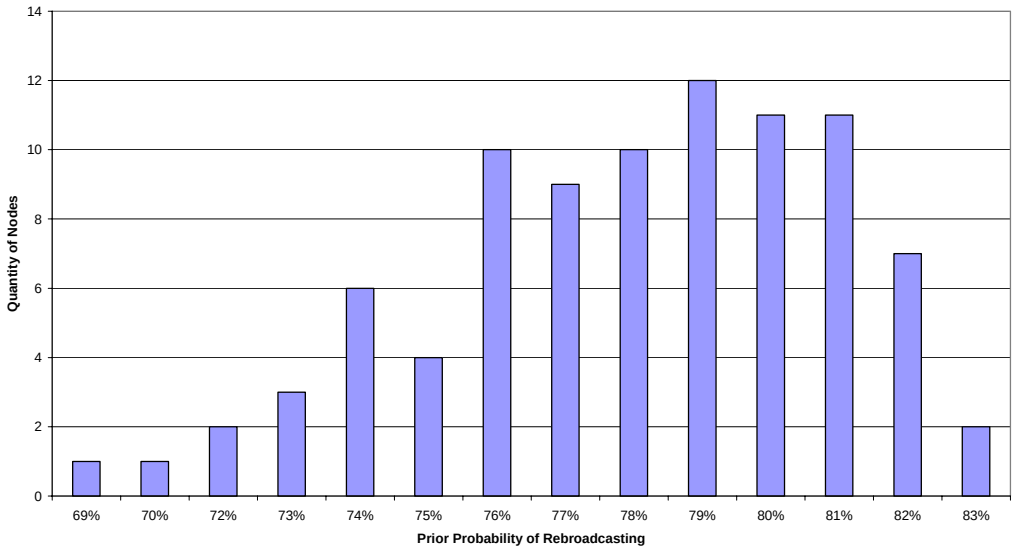

Figure 6.1: Prior Probability of Rebroadcasts in Trial 5 (Adaptive-SBA)

Therefore, we also tuned our machine learning models by cutting down the initial prior probabilities shown in Table 6.4. Since the prior probability affects our model's decision directly, this variable can be used as a knob to tune our model.

In our first attempt, where our data set has 30% reduction, naive Bayes yielded a 98.4% delivery ratio shown in Figure 6.2. In Trial 2 and Trial 3, naive Bayes maintained better than AHBP-EX and close to Adaptive SBA with 96.8% and 97.1% delivery ratios. When we look at Trial 4 and Trial 5, we can see that cutting the prior probability down with 30% pays off; naive Bayes broadcast protocol outperforms the others with 96.2% and 85% delivery ratios.

On the other hand, through Trial 4, the number of retransmitting nodes naive Bayes outputs is close to Adaptive SBA. However, in Trial 5, because of cutting the prior probability of rebroadcasting, naive Bayes retransmits much less than Adaptive SBA. Hence, it degrades more gracefully in Trial 5 and maintains higher delivery ratio than Adaptive SBA.

As shown in Figure 6.2 – Figure 6.4, our new broadcast protocol based on naive Bayes outperformed Adaptive SBA and AHBP-EX in Study 4 from [15]. Overall, the naive Bayes protocol maintains a high delivery ratio and low overhead. In all the trials, it maintains the highest or second-highest delivery ratio, including the most severe network environment.

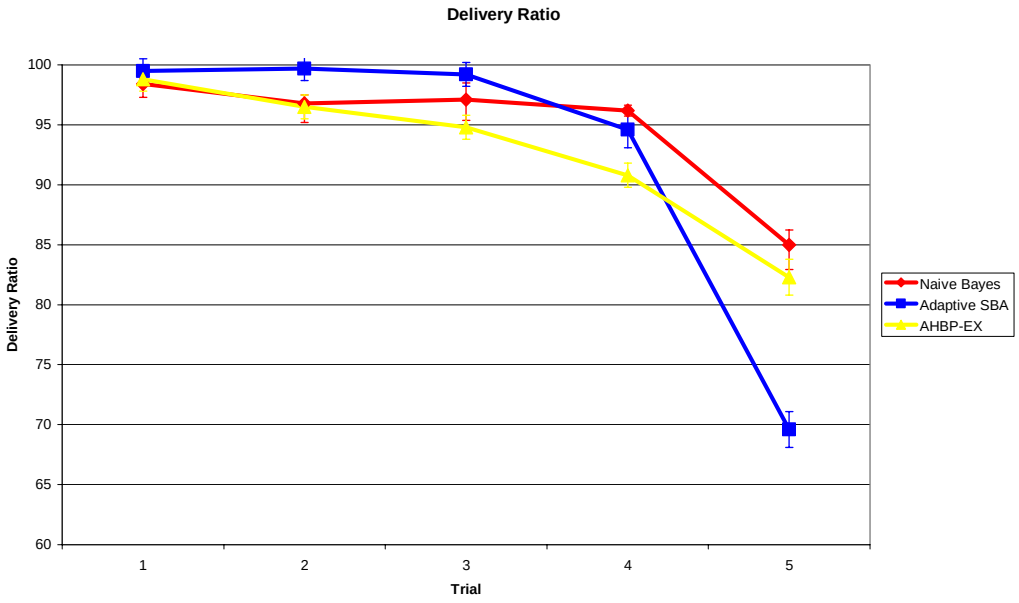

Figure 6.2: Delivery Ratio as Severity of Network Environment Increases

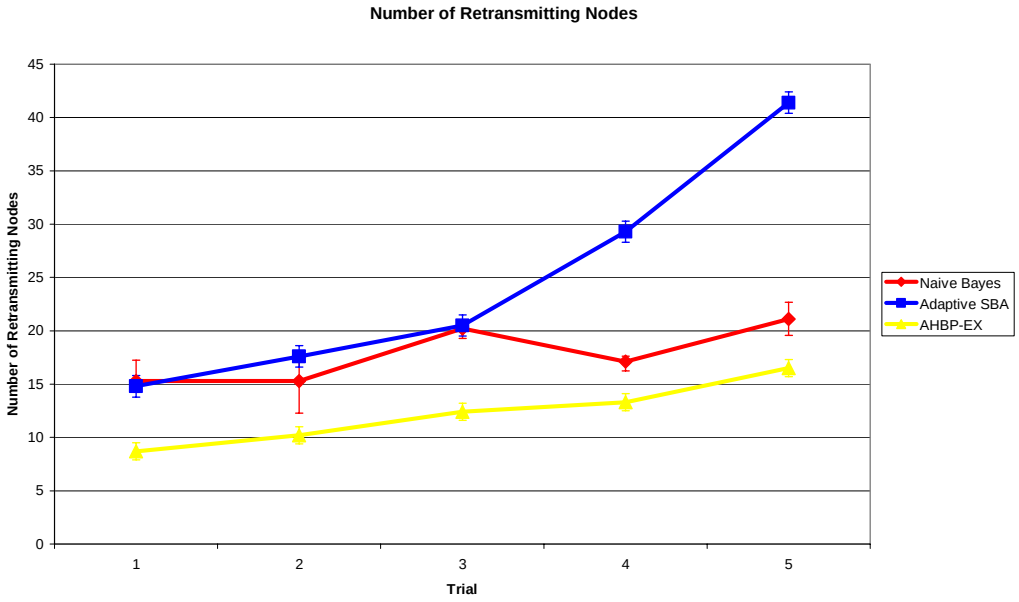

Figure 6.3: Number of Rebroadcasting Nodes as Severity of Network Environment Increases

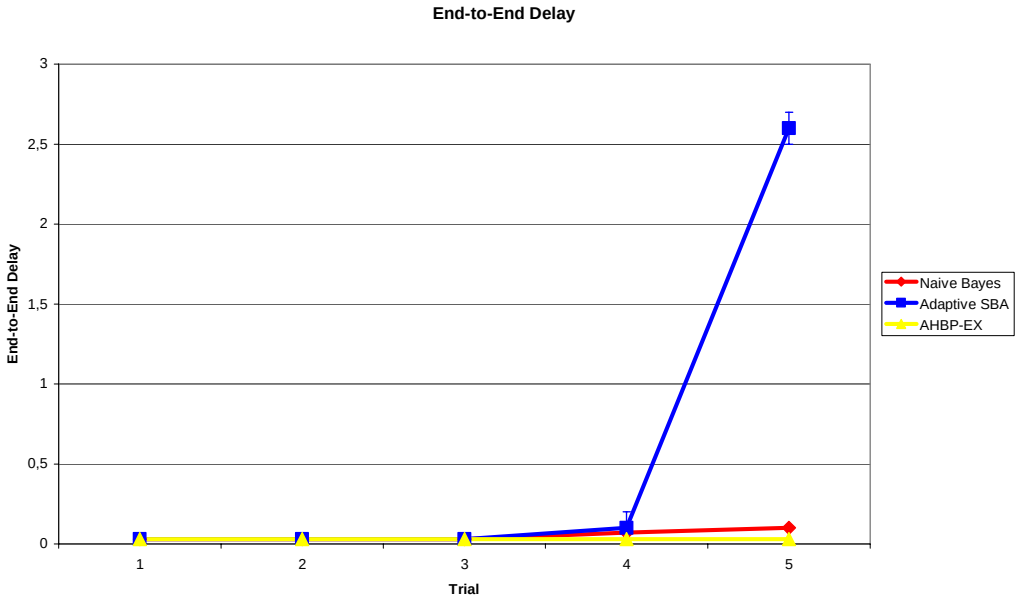

Figure 6.4: End-to-End Delay as Severity of Network Environment Increases.

We believe that the results presented in Figure 6.2 - Figure 6.4 display an optimized trade-off between delivery ratio and overhead. If, on the other hand, a network application specified that delivery ratio was more important than overhead, or vice versa, that knowledge can be directly incorporated into the naive Bayes model in Figure 4.1. If the delivery ratio is too low, the number of rebroadcasting nodes can be increased by increasing the prior probability of rebroadcasting in Trials 1 and 2, for example; a delivery ratio of 100% can be achieved by doubling the number of rebroadcasting nodes.

The trade-off between delivery ratio and overhead does not affect end to end delay in naive Bayes protocol and naive Bayes approach plots the same delay (as it is shown in Figure 6.4) no matter how the model is tweaked by the prior probability.

6.3.2 Study 2.2 - AdaBoost Broadcast Protocol

This study focuses on the ability of AdaBoost broadcast protocol in different network conditions listed in Table 6.3. Like naive Bayes, AdaBoost is also a data-driven generated classifier derived exclusively from the evidence that exists in the training data. As we experienced in the previous study, the dataset that we created by using Adaptive SBA was not labeled properly for Trial 5. Therefore, before we implemented AdaBoost protocol, we recreated a dataset for Trial 5 with approximately 44% prior probability of rebroadcasting and 56% prior probability of dropping an incoming packet.

After recreating the dataset for Trial 5, since initially there was no parameter to tune AdaBoost except the number of model creation, we decided to try the AdaBoost protocol with 50 models. In our first try, the protocol produced unexpected results. AdaBoost's delivery ratio for each trial is shown in Figure 6.5. According to the figure, with average delivery ratio of 93.3%, AdaBoost performed even worse than AHBP-EX in the first trial. At first glance, we thought that since we had a low delivery ratio in Trial 1, AdaBoost rebroadcasted less then it was supposed to, which was similar to AHBP-EX. Thus, it would perform better as the severity of network

increased. When we looked at the delivery ratio of Trial 2 through Trial 4, our guess was partially correct because AdaBoost maintained the second-highest delivery ratio in Trial 2 and Trial 3 with the average delivery ratio of 95.8% and 97.8%; in Trial 4 AdaBoost maintained 96.9% delivery ratio on average, which was the highest in the comparison. However, in Trial 5, unexpectedly even with the recreated dataset, AdaBoost maintained 76.7% average delivery ratio, which was better than Adaptive SBA, but worse than AHBP-EX. Figure 6.7 presents the end to end delay as the network severity increases, where the results follow the trends in Figure 6.5.

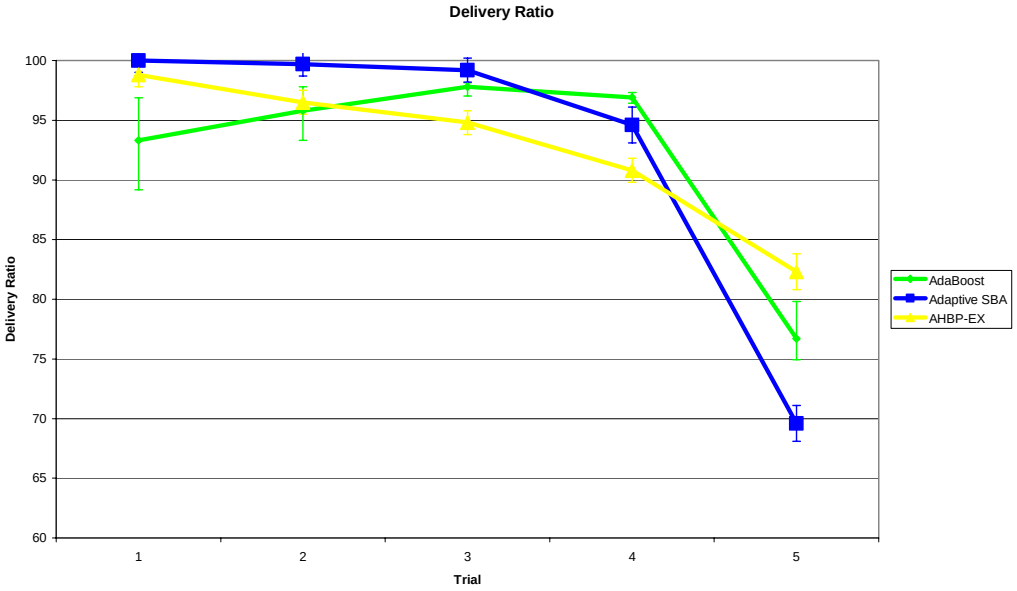

Figure 6.5: Delivery Ratio of AdaBoost as Severity of Network Environment Increases

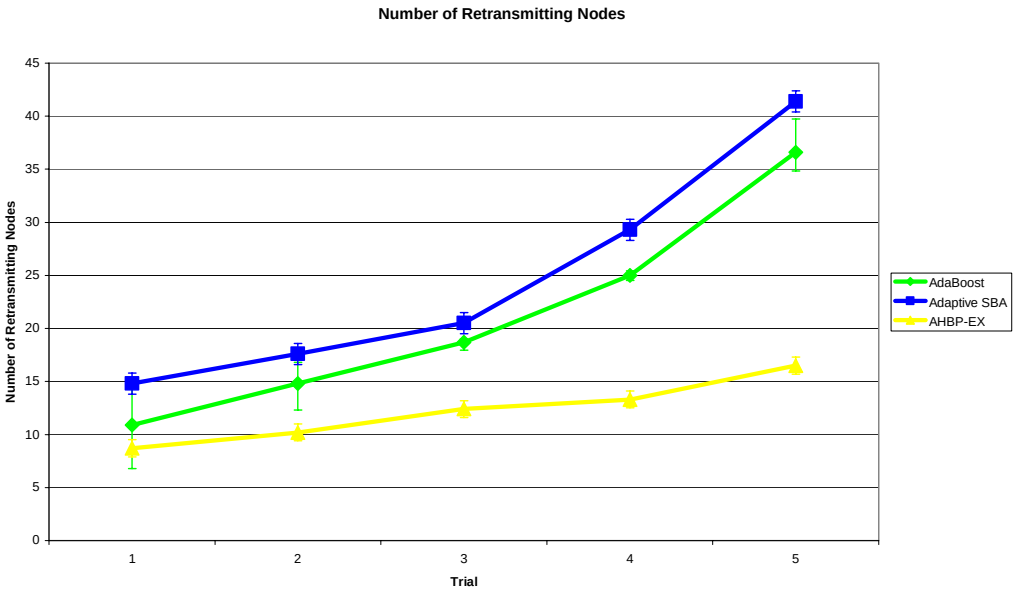

Figure 6.6: Number of Rebroadcasting Nodes as Severity of Network Environment Increases

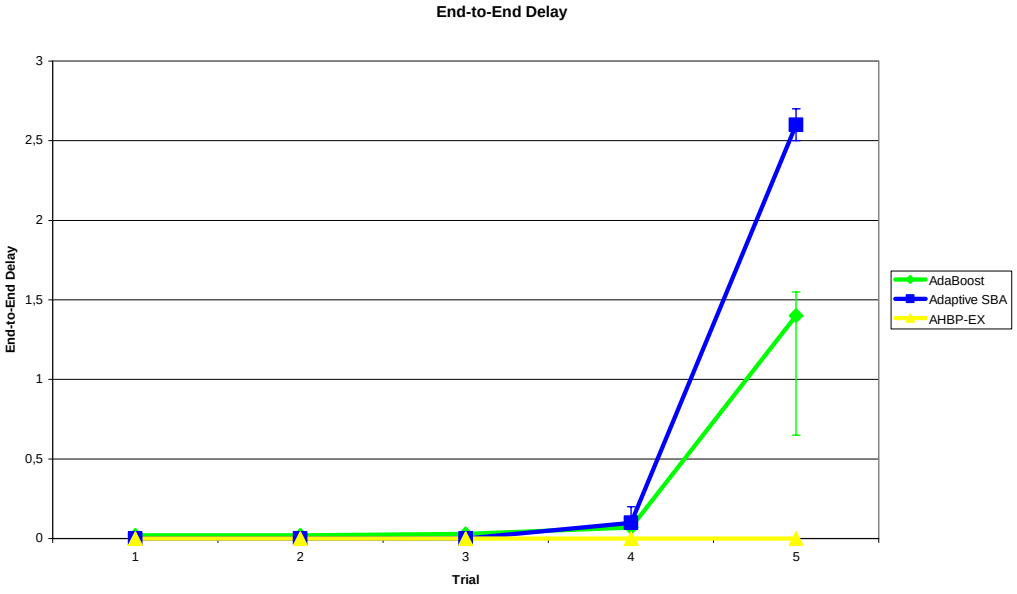

Figure 6.7: End-to-End Delay as Severity of Network Environment Increases

With these delivery ratios, AdaBoost was not as competitive as we desired, so we analyzed the number of retransmitting nodes AdaBoost maintained to figure out how we could improve AdaBoost protocol's performance. As it is demonstrated in Figure 6.6, we infer AdaBoost maintained low delivery ratio in Trial 1 because it rebroadcasted less than it was supposed to, which was similar to AHBP-EX. On the other hand, in Trial 5, AdaBoost rebroadcasted more than it was supposed to, which was similar to Adaptive SBA.

According to the results we had gotten so far (Figure 6.5 - Figure 6.7), we had two goals. The first one is to have more retransmitting nodes in Trial 1; the second one is to have less retransmitting nodes in Trial 5. As we know, even Flooding maintains 99%-100% delivery ratio in Trial 1. Therefore, to achieve our first goal, we forced AdaBoost protocol to behave like Flooding. This process would increase the number of retransmitting nodes as well as the delivery ratio.

As we discussed in Chapter 5, in AdaBoost, we assign weights to each instance in the training set, and we give more weights to the examples that are misclassified. This leads the next model to focus more on the misclassified examples. From this assumption, we increased the positive (rebroadcast) and correctly classified examples' weights more during the weights update. Therefore, our next model would focus on predicting more positive classifications (rebroadcast). To guarantee a high number of retransmitting nodes, we increased the weight of correctly classified rebroadcast examples by a factor of 1.5.

Forcing our model to behave like Flooding provided 100% delivery ratio in Trial 1, but AdaBoost maintained a very close number of retransmitting nodes to Flooding. Since our goal was to increase the delivery ratio of AdaBoost in low congested networks, we achieved our goal by increasing the number of retransmitting nodes.

As we mentioned before, our second goal was to improve the delivery ratio of AdaBoost protocol in Trial 5, which was the most severe network. The solution to achieve our goal was obviously decreasing the number of retransmitting nodes. Since we could increase the number of retransmitting nodes, we could also decrease it by using the same method. This time we updated the weights of correctly classified

negative examples (drop packet). We used the factor 1.2 and to keep the model simple, instead of creating 50 models as we had done before, we created only 15 models. After the changes, the average delivery ratio that AdaBoost protocol maintained in Trial 5 increased to 93.04% on average. This result also proved that AdaBoost provides the flexibility to tune its retransmission, which helped us to achieve our second goal successfully.

6.3.3 Study 2.3 - Comparison of Classification Approaches with Adaptive SBA, AHBP-EX and Flooding

In the previous two studies, we focus on each machine learning perspective and show how it is desirable for a protocol to tune itself in a systematic, mathematically-principled way. In order to show the flexibility of each individual new broadcast protocol, we varied their tuning ratios. This study attempts to show the evaluation of new classification approach protocols and their comparisons with Adaptive SBA, AHBP-EX, and Flooding. The previously proposed protocols and our newly classification based protocols have common and different features shown in Table 6.19.

Protocol	**RAD/ Jitter**	**GPS**	**Hello Packets**	**Complex Algorithm**	**Dominant Node Based**	**Reactive / Proactive**
Flooding	Jitter	No	No	No	No	Reactive
SBA	RAD	No	Yes	Yes	No	Reactive
AHBP	Jitter	No	Yes	Yes	Yes	Reactive
Naive Bayes	RAD	No	Yes	No	No	Reactive
AdaBoost	RAD	No	Yes	Yes	No	Reactive

Table 6.5: Protocol Features

As we demonstrated in Sections 6.3.1 and 6.3.2, the machine learning based protocols can be tuned according to the network conditions, but choosing the best tuned scheme is another issue that needs to be handled. The best behavior of each protocol can be chosen in two different ways.

First, behavior can be set statically through hello packets; the 2-hop neighbor knowledge is achieved, so the number of 1-hop neighbors and the number of 2-hop neighbors are known by each node. Also the number of received packets per second on average could be desirable as a feature. Therefore, according to these values, the classifiers can be set statically with the hard coded values of these attributes. This is similar to the method used to create Adaptive SBA in [15].

Second, the protocols can be tuned by using another model that predicts how to tune our protocols. The attributes used to train the tuner model can be the neighbor knowledge and the number of packets received per second on average. To keep the implementation simple, we would choose the naive Bayes model to tune our protocols; the circles again represent the random variables and the arrows denote the conditional independence relationships shown in Figure 6.8.

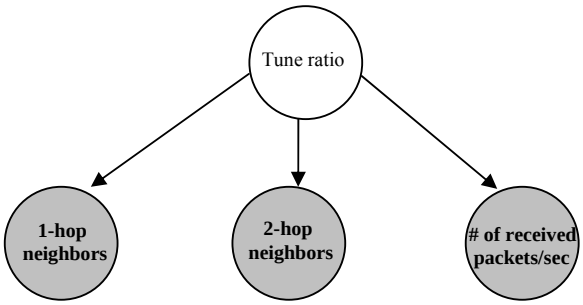

Figure 6.8: Naive Bayes Model that Predicts Tuning Ratio

Using the Bayesian approach, the instances included in training sets will be assigned with the most probable N target values (instead of binary). A very small

training data set can be used to train the classifier; since a supervised learning algorithm is implemented, this small training set can be created by hand and the target values can be assigned manually.

If we assume that the best tune ratio is chosen for each trial (in either static way or classification way), then the delivery ratio comparison of new machine learning based broadcast protocols will be as it is shown in Figure 6.9. According to Figure 6.9, if we compare the delivery ratio performance of each protocol, Flooding breaks first after Trial 2. In Trial 1, all of the protocols maintain close to 100% delivery ratio. In Trial 2 and Trial 3, naive Bayes and AdaBoost maintain between Adaptive SBA and AHBP-EX. As we discussed in Study 2.1 and Study 2.2 these values can be increased. However, to display the optimized trade-off between the delivery ratio and the overhead, they are not tuned to rebroadcast more. After Trial 3, our machine learning based broadcast protocols maintain the highest delivery ratio and outperform the others. Especially in Trial 5, AdaBoost protocol outperforms AHBP-EX roughly 10% with 93.04% delivery ratio on average.

On the other hand, when we compare the number of retransmitting nodes presented in Figure 6.10, naive Bayes rebroadcasts as much as Adaptive SBA from Trial 1 through Trial 4; after Trial 4, it gets close to AHBP-EX, which also explains how it maintains the second highest delivery ratio in Trial 5. As we discussed in Study 2.2, by forcing AdaBoost to rebroadcast more, the number of retransmitting nodes that AdaBoost has in Trial 1 is almost the same as Flooding. However, it pays off with 99.8% delivery ratio. From Trial 2 through Trial 4, AdaBoost protocol rebroadcasts almost as much as Adaptive SBA, but unlike Adaptive SBA, in Trial 5, it retransmits much less and gets close to AHBP-EX.

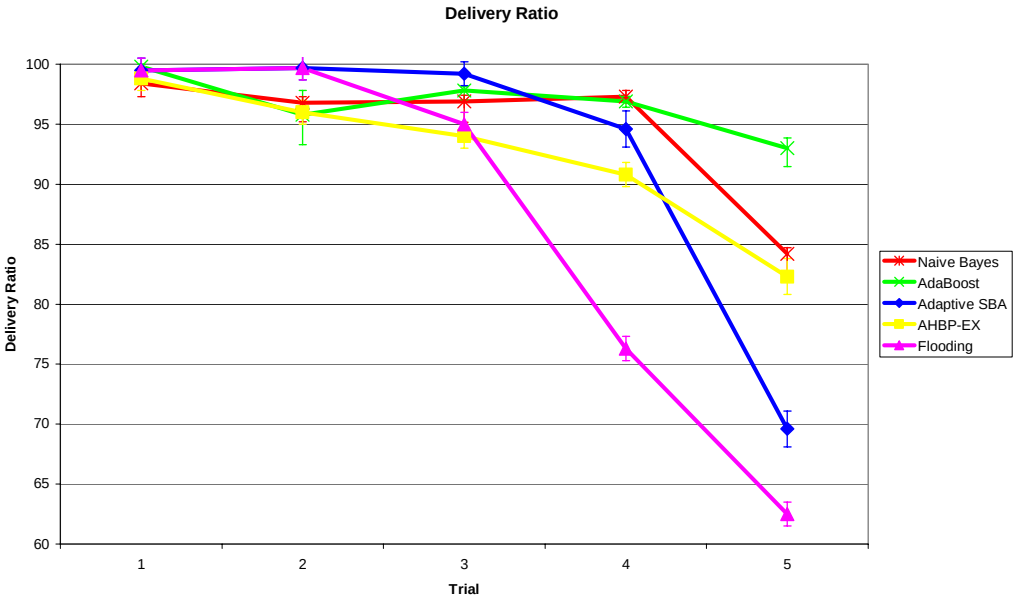

Figure 6.9: Comparison of Delivery Ratio as Severity of Network Environment Increases

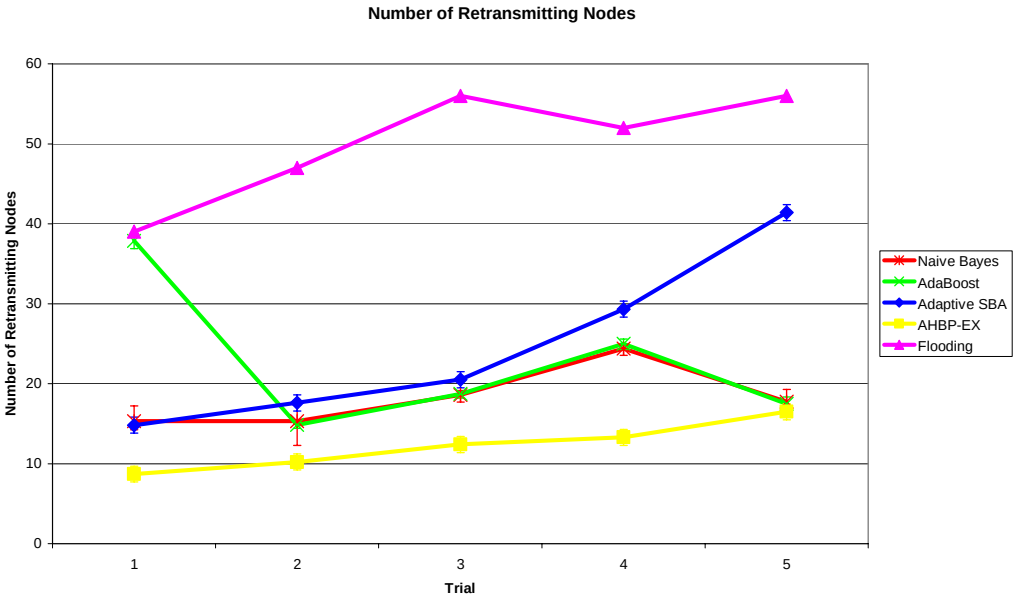

Figure 6.10: Comparison of Number of Rebroadcasting Nodes as Severity of Network Environment Increases

Overall, according to the results displayed in Figure 6.9, both machine learning based protocols outperformed Adaptive SBA, AHBP-EX and Flooding in Study 4 and Study 5. In each trial, as the network severity is increased, naive Bayes and AdaBoost maintain the highest or second-highest delivery ratios. AdaBoost maintains higher delivery ratio than naive Bayes in Trial 5, but AdaBoost requires significantly more retransmitting nodes than naive Bayes in Trial 1 to maintain a delivery ratio as high as naive Bayes.

Lastly if we compare end-to-end delay performances shown in Figure 6.11, both AdaBoost and Naive Bayes maintain almost zero second delay in all trials. In Trial 5, their delays are slightly more than AHBP-EX, where Adaptive SBA maintains more than 2.5 seconds end-to-end delay.

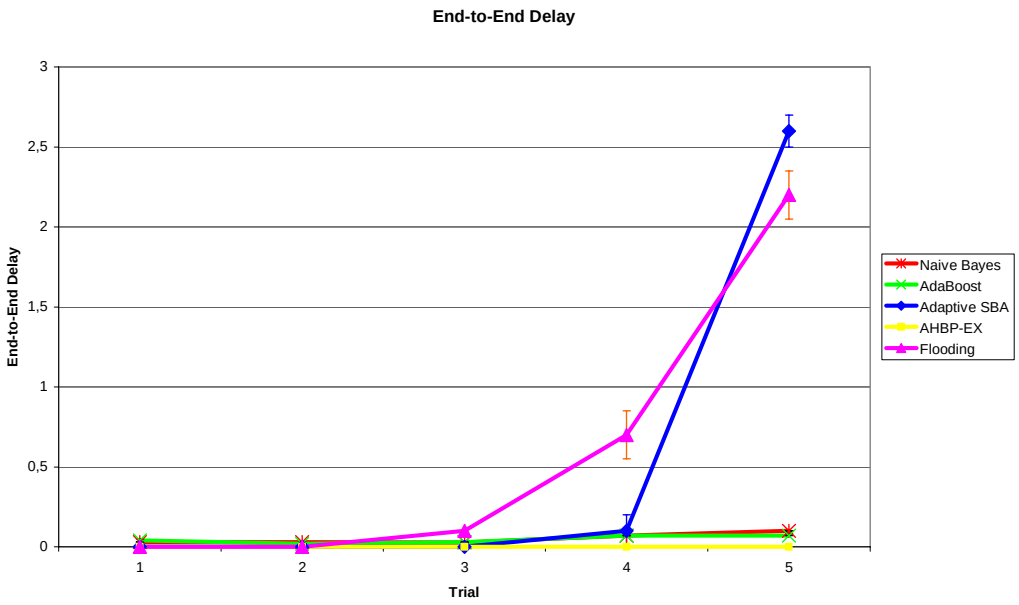

Figure 6.11: Comparison of End-to-End Delay as Severity of Network Environment Increases

Chapter 7

IMPLEMENTATION OF MANET IN AN OPEN PIT MINING ENVIRONMENT

7.1 Introduction

During the last decade, the Mine Safety and Health Administration (MSHA) has reported a significant number of off-highway dump truck accidents, which resulted in the loss of several human lives and millions of dollars in equipment costs. To reduce the number of such accidents, during the last four years, the researchers in Mining Engineering Department of the Colorado School of Mines (CSM) have been developing off-highway truck proximity warning and collision avoidance system using the Global Positioning System (GPS) and wireless networking technology [50]-[55].

This research project is primarily undertaken by Dr. Kadri Dagdelen in Mining Engineering Department at the Colorado School of Mines and funded by NIOSH (National Institute for Occupational Safety and Health). The past progress of this project involved:

- The development of the technology and the software for dump edge recognition in the lab at the Colorado School of Mines (CSM), using the Trimble GPS system with the Trimble 900 MHz radios.
- The development of a system using 802.11b wireless networking technology.
- Field tests of the system with respect to accuracy and its implementation at operating mines.
- The implementation of the system at an operating aggregate quarry for testing its reliability and operator acceptance.

The implementation of the collision avoidance system at the Lafarge Specialty Aggregating Quarry near I-70, Golden, Colorado involved a wireless 802.11b network operating in infrastructure mode throughout the quarry, the installation of the hardware and the software on two haulage trucks, and the development of the OptiTrack software. Due to the difficulties of obtaining the wireless network coverage throughout the quarry based on 802.11b operating in infrastructure mode, the suggestion was made to operate the network in ad hoc mode. As such, current research is undertaken to implement an ad hoc network to improve the developed system to increase the coverage of the wireless network. The prototype of the system is developed and tested at CSM for ad hoc and infrastructure type networks. However, only the infrastructure mode of the system is implemented at Lafarge Quarry. The future phases of this project will include implementation of the system as a mobile ad hoc network using various broadcast protocols discussed in this thesis and comparing them in an actual operating mining environment. Because the broadcasting protocols will be a building block of the system, it is imperative to have the most effective broadcast protocol possible for an efficient transfer of the data (i.e., truck location). Due to the shortcomings of an ad hoc network operating with existing broadcast protocols, more efficient broadcasting strategies are needed for an open pit mine with close to 100 MNs. Based on previous research on broadcast protocols [15], it appears that adaptive broadcasting strategy based on machine learning concepts may be more suitable to wireless networks established in a mining environment.

The remainder of this thesis will review the particular implementation of wireless networks and GPS technologies in an open pit mine environment. In Section 7.2, the implemented system is described and in Section 7.3, the implementation issues of a MANET in the developed system will be discussed.

7.2 Description of the System

The OptiTrack system involves different state-of-the-art technologies: the Global Positioning System, the mobile wireless network system, and a graphical interface

software system that provides 3D mapping. These technologies are combined in a real-time software system called OptiTrack, which is a new version of VirtualMine in [52].

The system is mounted inside the truck cabin and receives the GPS signal from the GPS receiver mounted on the truck. OptiTrack calculates the position of the truck with respect to the Digital Terrain Model (DTM) and sends this information to the other trucks in the network. At the same time, the truck receives the other trucks' information and displays it by mapping their coordinates with respect to the DTM of the mine.

Because the implemented system operates in an infrastructure network, the two way communications between the trucks are established via a central station. In the OptiTrack system there are two central stations: the repeater located at the Lafarge Quarry and the root access point placed at CSM. The transmitted (x, y, z) information is sent to each truck and the base station by the repeater. The base station receives the data through the root access point connected to the main virtual private network (VPN) in the Brown Building. Under this configuration, the vehicles can be monitored from the central office, through the VPN, using the OptiTrack software.

Mobile Trucks:

The mobile clients in the system are the haulage mine tucks. These trucks are designed for heavy conditions, so during the development of the system, rugged equipment is chosen to install in the trucks. The trucks include: 1-Rugged Tablet PC (Xplore / Getac), 1 GPS Unit (Trimble 4400 receiver), 1 Amplifier, 1 Wireless PCMCI Card (Cisco), 1 AC-DC Converter, 1 Lightning Arrestor, and cables. An equipped truck is shown in Figure 7.1.

Figure 7.1: Mounted GPS and Wireless Antennas on a Truck [55]

7.3 Implementation of OptiTrack in a MANET

The objective of the current research involves the compatibility of the developed system in a MANET. The purpose of switching from infrastructure to ad hoc is to solve some of the challenges that the developed system inherently includes because of operating in infrastructure mode. These challenges may be listed as:

- In infrastructure mode, if the repeater or the root access point goes down, wireless network communication between the trucks breaks down, including the communication with CSM.
- According to the topology of a given mine, covering all the surface of the pits at the mine with a number of repeaters (creating a wireless backbone) will increase the cost and difficulty of the implementation (since each repeater includes a trailer and a solar system).

To overcome the listed challenges above, it is desired to make the system compatible to operate both in infrastructure and ad hoc networks. Because the developed software works on the established network, first the wireless communication must support the ad hoc mode. Then, the software must include algorithms to run in a MANET.

7.3.1 Hardware Changes in the Network to Support Ad Hoc Mode

Since the MANET is going to be established only in the Lafarge Quarry, then there are two types of clients (mobile trucks and repeater) that must operate in ad hoc mode. In each mobile truck, a Cisco wireless adapter is installed. This adapter provides both ad hoc and infrastructure mode (the 802.11 standard defines two mode of operations), so each truck does not require any hardware alteration. However, all the trucks must be configured with the same Service Set Identifier (SSI) and the same Wired Equivalent Privacy (WEP) key values (or no WEP keys).

On the other hand, the repeater does require a new solution to operate in ad hoc mode because the one used in the developed system is a Cisco access point (AP) configured as a repeater. This AP cannot be a part of a MANET because it must operate in infrastructure mode; it does not listen (ignores) the clients in ad hoc mode (the clients that are not associated with itself). Instead of a repeater, there must be another device that will provide the communication between Lafarge Quarry and CSM. We propose installing a wireless router that will duplicate the functionality of the repeater. There are two options for wireless devices to route packets: a wireless router or a computer with the Linux operating system that has two wireless Ethernet adapters and is configured as a router. This system would operate without a central dependence, so even if the router fails, the communication between the trucks will still be available.

Since the system that operates in infrastructure mode has been implemented, the Cisco repeater and Cisco access point are the two devices that have been working without failure and without any maintenance problems. In this case, it is perceived that using a computer with the Linux operating system as a router becomes unprofitable because it is less reliable and would need periodic rebooting. The robustness of Cisco repeater inspires another idea to promote communication between the mobile trucks (operate in ad hoc mode) and the repeater (operate in infrastructure mode). Since we do not want to change the Cisco devices, then some of the trucks could have two wireless adapters. One adapter would be configured to operate in an

ad hoc mode; the other one would be configured to operate in an infrastructure mode. The trucks with two interfaces will be considered as cluster heads. When a cluster head receives a packet, it will route it to the repeater, which will send it to CSM. Two-way communication is also provided: the data coming from CSM will be delivered to the trucks via the cluster heads. This solution requires a second set of equipment (wireless omni antenna, amplifier, wireless PCMCIA adaptor, and etc.) installed on the chosen trucks (cluster heads). In this scenario, trucks would communicate with out a central point. If one of the cluster heads fails, the packets can be sent through another. If all the cluster heads fail, the communication between the trucks will still work even though the link between CSM (via the repeater) gets broken.

After one of the changes discussed above is implemented at the mine, the system seen in Figure 7.2 will operate as follows. Each truck receives the GPS signal and the developed software calculates the position of the truck with respect to the DTM. The truck updates its own position on the map and broadcasts the information (truck's coordinate) to all its neighbors. Each of those neighbors run its broadcast algorithm, based on machine learning, and decides whether to rebroadcast. The packets will be delivered to CSM via the nodes that are in routers' transmission range or via cluster heads which are in the repeaters transmission range. The collected data at the repeater/router will be sent to CSM. In this topology each device assists each other in transmitting packets through the network. If a new truck joins the network, the network will automatically incorporate it into the existing system. If one router (mobile truck) fails, the messages are sent through an alternate path by other routers (mobile trucks).

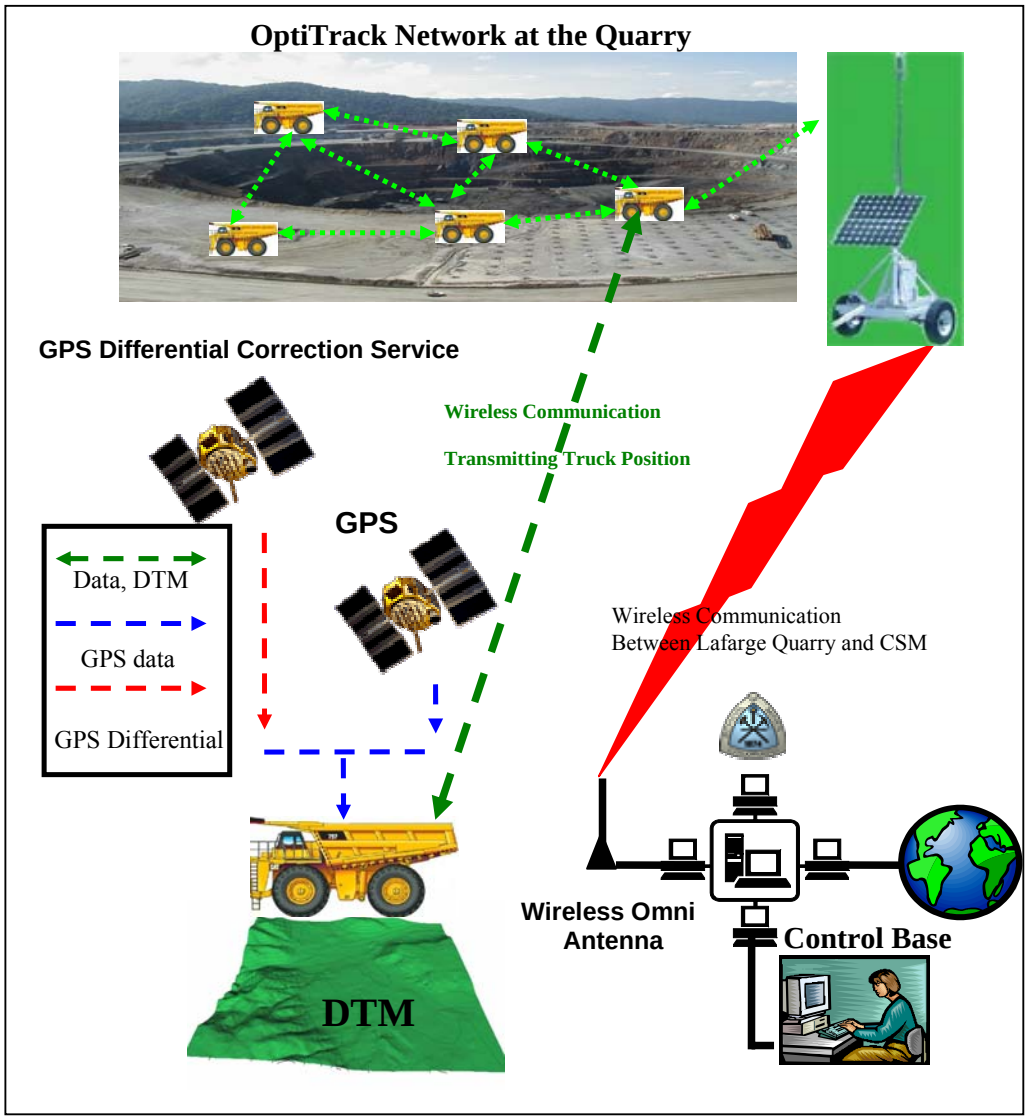

Figure 7.2: General Description of the System Operating in 802.11b Ad Hoc Mode [55]

7.3.2 Software Changes to Support Ad Hoc Networks

In the developed software OptiTrack, the Simple Flooding broadcast algorithm is already implemented. However, as it is proven in [15], Simple Flooding can not

handle congested networks. More efficient broadcasting strategies are needed for an open pit mine with close to 100 MNs. Based on the research on broadcast protocols [15] and the work done in this thesis, it appears that none of the existing protocols can handle wide range network conditions, and that an adaptive broadcasting strategy based on machine learning concepts may be more suitable to MANETs in mining environments. The results in Chapter 6 show that both naive Bayes and AdaBoost maintain high delivery ratio (highest or second highest) and low overhead in all trials including the most severe network environments.

Also, in Chapter 6, it is shown that there is a trade-off between delivery ratio and overhead. The new broadcast protocols (based on machine learning) can easily incorporate more important knowledge (delivery ratio / overhead) into their model. At this moment, the delivery ratio of the packets has the highest priority because a missing packet jeopardizes the safety in the whole collision avoidance system.

Chapter 8

FUTURE WORK AND CONCLUSIONS

A broadcast protocol is a building block of any MANET, so it is imperative to have the most efficient broadcast protocol possible for a reliable network. Because of this, there has been recent interest in developing network-wide broadcast protocols for MNs in a MANET.

This thesis carries out the development of two broadcast protocols (Naive Bayes and AdaBoost) based on machine learning methodology, which are the most efficient protocols under the widest range of network conditions. We developed and compared our new broadcast protocols based on machine learning with some of the proposed broadcast protocols where the effects of mobility (dynamic topology), congestion, and node density are combined. The effect of limited bandwidth on the wireless network using different broadcasting protocols is also evaluated in the simulation.

Based on the performance investigation in [15], the comparison of protocols indicates that the performance of Neighbor Knowledge is superior to the other methods proposed for a flat network topology. Due to their performance in [15], we compared our new broadcast protocols (based on machine learning) with the most commonly used protocol (i.e., Simple Flooding) and two Neighbor Knowledge Methods (i.e., Adaptive SBA and AHBP-EX).

We developed two protocols and demonstrated their flexibility in Chapter 6. The first developed broadcast protocol is Naive Bayes, which uses a supervised learning classifier commonly referred to as *naive Bayes classifier,* to make optimal classifications (rebroadcast or drop). The second developed broadcast protocol is AdaBoost, which is based on another supervised learning algorithm commonly referred to as *Adaptive Boosting*.

Our performance evaluation of machine learning based broadcast protocols shows the following:

1. From a machine learning perspective, it is desirable for the developed protocols to tune their selves in a systematic, mathematically-principled way (this shows their flexibilities).
2. There is a trade-off between delivery ratio and overhead. Therefore, if a network application specified that delivery ratio was more important than overhead, or vice versa, that knowledge can be directly incorporated into our machine learning models.
3. Overall, machine learning based broadcast protocols maintain a high delivery ratio and low overhead even within the most severe network environment.

Based on the performance results in [15], Adaptive SBA and AHBP-EX are preferred over other broadcast protocols. Unfortunately, there is no clear choice between them because they each fit different niche areas. We demonstrated that our machine learning protocols can be recommended for all expected scenarios (e.g., semi-static topology, extremely congested network, and etc.).

In all the trials, AdaBoost maintains the highest or second-highest delivery ratio including the most severe network environments. Especially in the most severe network environment, AdaBoost outperformed AHBP-EX by roughly 10% and Adaptive SBA by roughly 25%. However, in the low congested networks, to maintain 100% delivery ratio, AdaBoost required a higher number of rebroadcasting nodes than Adaptive SBA and AHBP-EX.

On the other hand, naive Bayes obtained high delivery ratio in low congested networks, but it was harder to train. In the severe network environments, Naive Bayes also outperformed Adaptive SBA and AHBP-EX. As well as AdaBoost, Naive Bayes can be tuned according to the network environments.

When we compare Naive Bayes and AdaBoost, the differences appear based on their machine learning concepts. AdaBoost is a more sophisticated classifier than naive Bayes; however, during our accuracy tests, naive Bayes performed as well as

AdaBoost. The training phase of the Naive Bayes protocol is shorter; as a node gets more local experience it is easy to automatically adapt itself to the network by changing the entries in its prior probability and likelihood probability tables.

On the other hand, AdaBoost is very successful at generating rules that make the most accurate predictions possible on new incoming packets. It also provides the elimination of inefficacious attributes in the training set by ignoring them.

We argue from our results that the broadcast protocols developed in this thesis are the most efficient under the widest rage of network conditions. Based on our performance evaluation, we conclude that both Naive Bayes and AdaBoost can fit in areas where none of the existing broadcast protocols can.

According to our investigations, the performance of the machine learning based protocols can not be significantly improved through the use of a more sophisticated classifier or another training procedure. Based on our conclusions from this thesis, further improvements will come from finding better, more predictive input features.

As an example of more predictive input features, we plan to incorporate location information. As we presented in Chapter 7, in real implementations, missing each packet jeopardizes the robustness of the whole system. Therefore, it is imperative to use a methodology that efficiently delivers a packet from one node to all other network nodes under the widest range of network conditions. Hence, further research in the implementation of a MANET in an open pit mining environment will include the developed broadcast protocols based on machine learning methodology. Since the MANET at the open pit mining environment will have the location information of the trucks, we will explore a new input to our machine learning models. The new input will be the distance between the sending node and the receiving node. In addition to the neighbor knowledge features, this feature will make our machine learning model more efficient. When a receiving node is close to the sending node, the model will learn to rebroadcast less often because it will reach less additional neighbors.

Lastly, a further research area is in the area of sensor networks. Traditional ad hoc routing techniques do not usually fit the requirements of sensor networks due to the different constraints. Therefore, the networking layer of sensor networks is usually

designed according to different principles. One of the most important principles is power efficiency. However, one of the techniques used for routing in sensor networks is Flooding. So instead of Flooding, a machine learning approach can be used because it is desirable for a protocol to decrease the number of retransmitting nodes. Therefore, this fact might be used to improve the computation of energy efficiency for minimum energy communication networks.

REFERENCES

[1] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward, "A distance routing effect algorithm for mobility (DREAM)," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM),* pp. 76-84, 1998.

[2] C. Chiang and M. Gerla, "Routing and multicast in multihop, mobile wireless networks," in *Proceedings of the IEEE International Conference on Universal Personal Communications (ICUPC)*, pp. 546-551, 1997.

[3] C. Chiang, H. Wu, W. Liu, and M. Gerla, "Routing in cluster head multihop, mobile wireless networks with fading channel," in *Proceedings of IEEE SICON '97*, pp. 197-211, 1997.

[4] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath, "Flooding for reliable multicast in multi-hop ad hoc networks," in *Proceedings of the International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication (DIALM)*, pp. 64-71, 1999.

[5] H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc networks," in *Proceedings of the ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM),* pp. 61-68, 2000.

[6] ____, "Flooding in wireless ad hoc networks," *Computer Communications Journal,* vol. 24, no. 3-4, pp. 353-363, 2001.

[7] C. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal on Selected Areas in Communications,* vol. 15, no. 7, pp. 1265-1275, 1997.

[8] G. E. Pagnani and G. P. Rossi, "Providing reliable and fault tolerant broadcast delivery in mobile ad-hoc networks," *Mobile Networks and Applications,* vol. 5, no. 4, pp. 175-192, 1999.

[9] W. Peng and X. Lu, "Efficient broadcast in mobile ad hoc networks using connected dominating sets," *Journal of Software – Beijing, China,* 1999.

[10] ____, "On the reduction of broadcast redundancy in mobile ad hoc networks," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC),* pp. 129-130, 2000.

[11] ____, "AHBP: An efficient broadcast protocol for mobile ad hoc networks," *Journal of Science and Technology – Beijing, China,* 2002.

[12] A. Qayyum, L. Vennot, and A. Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks," INRIA – Rapport de recherché, Tech. Rep. 3898, 2000.

[13] I. Stojmenovic and X. Lin, "Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, pp. 1023-1032, 2001.

[14] J. Sucec and I. Marsic, "An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks," September 2000, CAIP Technical Report 248 – Rutgers University. http://www.caip.rutgers.edu/~marsic/mobile/ (Accessed on February 22nd, 2004).

[15] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pp. 194-205, 2002.

[16] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing,* T. Imelinsky and H. Korth, Eds. Kluwer Academic Publishers, pp. 153-181, 1996.

[17] D. Johnson, D. Maltz, and Y. Hu, "The dynamic source routing protocol for mobile ad hoc networks," April 2003, Internet Draft: draft-ietf-manet-dsr-09.txt.

[18] C. Perkins, E. Beldig-Royer, and S. Das, "Ad hoc on demand distance vector (AODV) routing," Request for Comments 3561, July 2003.

[19] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications,* pp. 90-100, 1999.

[20] Z. Haas, "A new routing protocol for reconfigurable wireless networks," in *Proceedings of the IEEE International Conference on Universal Personal Communications (ICUPC)*, pp. 562-565, Oct. 1997.

[21] Z. Haas and M. Pearlman, "The performance of query control schemes for the zone routing protocol," *IEEE/ACM Transactions on Networking*, vol. 9, no.4, pp.427-438, 2001.

[22] Z. Haas and B. Liang, "Ad hoc mobility management with randomized database groups," in *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 1756-1762, 1999.

[23] Y. Ko and N. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM),* pp. 66-75, 1998.

[24] S. Carson and A. Ephremides, "A distributed routing algorithm for mobile wireless networks," *ACM Journal on Wireless Networks,* vol. 1, no. 1, pp. 61-81, 1995.

[25] Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting" in *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119-139, August 1997.

[26] S.-J. Lee, M. Gerla, and C.-C. Chiang, "On-demand multicast routing protocol," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1298-1302, September 1999.

[27] T. Camp and Y. Liu, "An adaptive mesh-based protocol for geocast routing," *Journal of Parallel and Distributed Computing: Special Issue on Routing in Mobile and Wireless Ad Hoc Networks,* vol. 62, no. 2, pp. 196-213, 2003.

[28] Y. Ko and N. Vaidya, "Geocasting in mobile ad hoc networks: Location based multicast algorithms," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA),* pp. 101-110, 1999.

[29] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pp. 151-162, 1999.

[30] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, 1997.

[31] I. S. Committee, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," in *IEEE 802.11 Standard.* IEEE, New York, 1996, ISBN 1-55937-935-9.

[32] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Discrete Mathematics,* vol. 86, pp. 165-177, 1990.

[33] V. Bharghavan and B. Das, "Routing in ad hoc networks using minimum connected dominating sets," in *Proceedings of the IEEE International Conference on Communications (ICC),* pp. 376-389, 1997.

[34] B. Das, R. Sivakumar, and V. Bharghavan, "Routing in ad-hoc networks using a spine," in *Proceedings of the International Conference on Computers and Communications Networks (ICCCN)*, pp. 34-39, 1997.

[35] S. Guah and S. Khuller, "Approximation algorithms for connected dominating sets," in *Proceedings of European Symposium on Algorithms (ESA),* 1996.

[36] ____, "Approximation algorithms for connected dominating sets," *Algorithmica,* vol. 20, no. 4, pp. 374-387, 1998.

[37] R. Svakumar, B. Das, and V. Bharghavan, "The calse vertebrata: spines and routing in ad hoc networks," in *Proceedings of the IEEE Symposium on Computers and Communications,* pp. 599-605, 1998.

[38] ____, "Spine routing in ad hoc networks," *Cluster Computing*, vol. 1, no.2, pp. 237-248, 1998.

[39] P. Wan, K. Alzoubi, and O. Friederi "Distributed constructions of connected dominating set in wireless ad hoc networks," in *Proceedings of INFOCOM,* pp. 1597-1604, 2002.

[40] J. Wu and H. Li, "On calculating connected dominating sets for efficient routing in ad hoc wireless networks," in *Proceedings of the International Workshop on Discrete Algorithms and methods for Mobile Computing and Communication (DIAL-M)*, pp. 7-14, 1999.

[41] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)", Request for Comments 3626, 2003.

[42] K. Fall and K. V. (Editors), "UCB/LBNL/VINT network simulator – ns (version 2)," http://www-mash.cs.berkley.edu/ns/. Accessed February 15th, 2004.

[43] P. Domingos and M. Pazzani, "Beyond independence: Conditions for the optimality of the simple Bayesian classifier," in *Proceedings of the 13th International Conference on Machine Learning*, pp. 105-112, 1996.

[44] W. Navidi, N. Bauer, and T. Camp, "Improving the accuracy of random waypoint simulations through steady-state initialization," in *Proceedings of the 15th International Conference on Modeling and Simulation (MS 2004),* pp. 319-326, 2004.

[45] Bayes, Rev. T., "An Essay Toward Solving a Problem in the Doctrine of Chances," *Philos. Trans. R. Soc. London 53*, pp. 370-418 (1763); reprinted in *Biometrika* 45, pp. 293-315 (1958), and *Two Papers by Bayes,* with commentary by W. Edwards Deming, New York Hafner, 1963.

[46] Y. Freund, "An Introduction to Boosting," http://www1.cs.columbia.edu/~freund/talks /IntroToBoosting.ppt. Accessed February 22nd, 2004.

[47] M. Hauskrecht, "Ensemble methods. Boosting," http://www.cs.pitt.edu/~milos/courses/ cs2750-Spring03/lectures/class21.pdf. Accessed February 22nd, 2004.

[48] R.E. Schapire, "The Boosting Approach to Machine Learning," in *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[49] J. Sun, "Mobile Ad Hoc Networking: An Essential Technology for Pervasive Computing," in *Proceedings of the International Conferences on Info-Tec & Info-Net*, Beijing, China, 2001.

[50] K. Dagdelen, and A. Nieto, "Development and testing of a vehicle collision avoidance system based on GPS and wireless networks for open pit mines", in *31st APCOM International Conference*, Cape Town, South Africa, May 2003.

[51] A. Nieto and K. Dagdelen, "Accuracy Testing of a Vehicle Proximity Warning System based on GPS and Wireless Communication Networks", in *International Journal of Surface Mining, Reclamation and Environment*, vol. 17, no. 3, pp. 156-170, 2003.

[52] A. Nieto, "Development of a real time proximity warning and 3-D mapping system based on wireless networks, virtual reality graphics, and GPS to improve safety in open pit mines", *PhD Dissertation*, Colorado School of Mines, Golden, Colorado, 2001.

[53] K. Dagdelen and A. Nieto, "Improving Safety of Off Highway Trucks Through GPS", in *Proceedings of 29th International Symposium on Applications of Computers and Operations Research in the Mineral Industry (APCOM),* Beijing, China, pp. 757-762, 2001.

[54] K. Dagdelen and A. Nieto, "Improving Safety and Productivity of Open Pit Mines Through GPS and Virtual Reality", in *Proceedings of 17th International Mining Conference and Exhibition of Turkey,* Ankara, Turkey, June 2001.

[55] K. Dagdelen, S. Reznik, and F. Bilgin, "Development of Vehicle Collision Avoidance and Proximity Warning System for off Highway Trucks using GPS and Wireless Networks", *Western Mining Resource Center Progress Report*, 2004.