Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2007, Article ID 10216, 16 pages
doi:10.1155/2007/10216

Research Article

Intelligent Broadcasting in Mobile Ad Hoc Networks: Three Classes of Adaptive Protocols

Michael D. Colagrosso

Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401-1887, USA

Received 10 February 2006; Revised 3 July 2006; Accepted 16 August 2006

Recommended by Hamid Sadjadpour

Because adaptability greatly improves the performance of a broadcast protocol, we identify three ways in which machine learning can be applied to broadcasting in a mobile ad hoc network (MANET). We chose broadcasting because it functions as a foundation of MANET communication. 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 higher-level MANET functionality and applications. While efficient broadcast protocols have been proposed, no single broadcasting protocol works well in all possible MANET conditions. Furthermore, protocols tend to fail catastrophically in severe network environments. Our three classes of adaptive protocols are pure machine learning, intra-protocol learning, and inter-protocol learning. In the pure machine learning approach, we exhibit a new approach to the design of a broadcast protocol: the decision of whether to rebroadcast a packet is cast as a classification problem. Each mobile node (MN) builds a classifier and trains it on data collected from the network environment. Using intra-protocol learning, each MN consults a simple machine model for the optimal value of one of its free parameters. Lastly, in inter-protocol learning, MNs learn to switch between different broadcasting protocols based on network conditions. For each class of learning method, we create a prototypical protocol and examine its performance in simulation.

1. INTRODUCTION: AD HOC NETWORK BROADCASTING

We introduce three new classes of broadcast protocols that use machine learning in different ways for mobile ad hoc networks. A mobile ad hoc network (MANET) comprises 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. Scenarios that might benefit from ad hoc networking technology include rescue/emergency operations after a natural or environmental disaster, or terrorist attack, that destroys existing infrastructure, special operations during law enforcement activities, tactical missions in a hostile and/or unknown territory, and commercial gatherings such as conferences, exhibitions, workshops, and meetings. Network-wide broadcasting, simply referred to as "broadcasting" herein, is the process in which one MN sends a packet to all MNs in the network (or all nodes in a localized area). There has been considerable effort devoted to the development of network-wide broadcast protocols in an ad hoc network [1–14]. A performance evaluation of MANET broadcast protocols is available in [15].

Broadcasting is a building block for most other network layer protocols, providing important control and route establishment functionality in a number of unicast routing protocols. 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, 25, 26]) and geocast protocols (e.g., [27, 28]).

The preceding protocols typically assume a simplistic form of broadcasting called simple flooding, in which each MN retransmits every unique received packet exactly once. The main problems with simple flooding are that it often

causes unproductive and harmful bandwidth congestion (e.g., called the broadcast storm problem in [29]) and it wastes node resources. The goal of an efficient broadcast technique is to minimize the number of retransmissions while attempting to ensure that a broadcast packet is delivered to each MN 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. In contrast to these static protocols, [15] proposed a hand-tuned rule to adapt the main parameter of one protocol and found it works well in many network environments. That adaptive rule, described further in Section 2.3, made 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.

In that spirit, we have identified three ways in which machine learning techniques can be incorporated naturally into broadcasting, and we use these ideas to create three new classes of broadcasting protocols: pure machine learning, intra-protocol learning, and inter-protocol learning. We believe that each class provides adaptability in a unique way, so we present example protocols from all three classes in this work. In Section 3, we train a probabilistic classifier and develop it into a pure machine learning-based protocol, which is an extension of our previous work [30]. We choose Bayesian networks [31] for our learning model 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 (to retransmit or not), 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 (Section 7) show that our Bayesian network combines the input features appropriately and often predicts whether to retransmit or not correctly. In Section 4, we develop an intra-protocol learning method, in which the machine learning model's job is to learn the optimal value of a parameter in a known broadcasting protocol. Although there are as many candidate protocols in this class as there are free parameters in the broadcasting literature, we present one adaptive broadcasting protocol that learns the value of a parameter that is particularly sensitive to two MANET variables, traffic and node density. The resulting protocol performs better than attempts by human experts to hand-tune that parameter. As the name implies, inter-protocol learning means learning between protocols, and we introduce that method in Section 5. Since no single broadcast protocol was found to be optimal in the previously cited survey, we propose an approach in which MNs switch between protocols based on network conditions. We develop a machine learning method that allows MNs to switch between two complicated broadcasting protocols, and, despite the logistical difficulties, the resulting combination performs better than either of the parts.

Since a broadcast protocol is a building block of many other MANET routing protocols, it is imperative to have the most effective broadcast protocol possible. We believe we have found three specific broadcast protocols that are efficient under the widest range of network conditions. Moreover, by identifying three new classes of adaptive protocols, we hope to inspire new development in the same vein.

2. STATIC BROADCAST PROTOCOLS

In addition to providing an overview of the broadcast literature, we describe two published broadcast algorithms in depth: the scalable broadcast algorithm and the ad hoc broadcasting protocol. We modify these two algorithms in Sections 4 and 5 by incorporating machine learning, and we compare the results of the static protocols, the modified protocols, and our pure machine learning protocol (Section 3) through simulation, presenting the results in Section 7. We name the protocols in this section static protocols because the protocol's behavior does not change or adapt over time; nevertheless, the protocols herein are certainly designed with mobile nodes in mind. We introduce two key concepts—the minimum connected dominating set and six families of broadcast protocols—before discussing the protocols in depth.

In the IEEE 802.11 MAC [32] protocol, the RTS/CTS/data/ACK handshake is designed for unicast packets. To send a broadcast packet, an MN needs only to assess a clear channel before transmitting. Since no recourse 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 possibility of collisions by limiting the number of rebroadcasts in the network. A theoretical "best-case" bound for choosing which nodes to rebroadcast is called the minimum connected dominating set (MCDS). An MCDS is the smallest set of rebroadcasting nodes such that the set of nodes is connected and all nonset nodes are within one hop of at least one member of the MCDS. The determination of an MCDS is an NP-hard problem [33]. Articles in the literature have therefore proposed *approximation* algorithms to determine the MCDS, for example, [13, 34–41].

We categorize existing broadcast protocols into six families: global knowledge, simple flooding, probability-based methods, area based methods, neighbor knowledge methods, and cluster-based methods. In [15], several existing broadcast protocols from all families are presented with a detailed performance investigation; that investigation found that the performance of neighbor knowledge methods is superior to all other families for flat network topologies. Thus, we choose neighbor knowledge protocols as the basis for our machine learning improvements in Sections 4 and 5, and

they serve as the benchmark for our performance comparison in Section 7.

2.1. The scalable broadcast algorithm

The scalable broadcast algorithm (SBA) [10] requires that all MNs know their neighbors within a two-hop radius. Two-hop neighbor knowledge is achievable via periodic "hello" packets; each "hello" packet contains the nodes identifier (IP address) and the list of known neighbors. After an MN receives a "hello" packet from all its neighbors, it has two-hop topology information centered at itself. When Node B receives a broadcast packet from Node A, Node B schedules the packet for delivery with a random assessment delay (RAD) if and only 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 MNs by rebroadcasting. This process continues until either the RAD expires and the packet is sent, or the packet is dropped if all two-hop neighbors are covered. The RAD is chosen randomly from a uniform distribution between 0 and $T_{\max}$ seconds, where $T_{\max}$ is the highest possible delay. It turns out that SBA's performance is sensitive to the value of $T_{\max}$. If $T_{\max}$ is high, an MN will wait longer for redundant rebroadcasts, possibly dropping its rebroadcast if all its two-hop neighbors are covered. Thus, the number of rebroadcasting MNs will likely be reduced, but the end-to-end delay (the time it takes for the last node to receive a packet) is increased. Choosing the right value of $T_{\max}$ must balance the desire for a small number of rebroadcasting nodes against the desire for a small end-to-end delay.

A simple method to dynamically adjust the length of the RAD to network conditions is proposed in [10]. Specifically, each MN searches its neighbor tables for the maximum neighbor degree of any neighbor node, $d_{N\max}$. It then calculates a RAD based on the ratio of $d_{N\max}/d_i$, where d_i is the node i's current number of neighbors. This weighting scheme is greedy: MNs with the most neighbors usually broadcast before the others. A completely different method that adapts the length of the RAD based on traffic rather than the number of neighbors was developed in [15], and is described in Section 2.3.

Before we present our machine learning protocols in Sections 3, 4, and 5, we investigate a simpler question: can we create a model that emulates SBA? That is, instead of creating a new broadcast protocol, we studied whether we could create a protocol that could learn to behave like SBA, without specifying the SBA algorithm. We collected data on MNs running the SBA protocol under the range of network conditions in Section 6. Every time an MN decided to rebroadcast or drop a packet, we recorded that event and annotated it with the current network conditions that the MN had available (e.g., see Figure 3). We collected 125 000 such events from different MNs in various environments, and treated these records in a database to be classified by a machine learning model. The inputs to the model are the instantaneous network conditions, and the desired output is SBA's decision of whether to rebroadcast the packet. We found that SBA could be fit with extremely simple models. Figure 1 shows a particularly simple yet accurate model of SBA. This decision tree [42] model matched the training database with 87% accuracy. What is striking about the decision tree model in Figure 1 is that it can be implemented as two "if-then" statements. This means that most of SBA's functionality, which requires maintaining a graph structure of two-hop neighbors and implementing a set-cover algorithm, can be emulated quite simply over a range of environments. We do not claim that this model does 87% as well as SBA; in some scenarios, this decision tree performs better, but SBA does better more often. On average, however, SBA and this simple model agree 87% of the time.

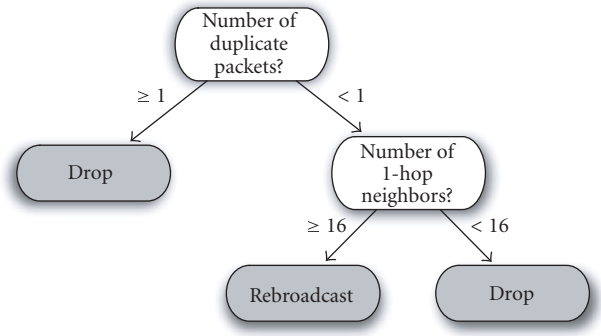


Figure 1: A simple decision tree model of the SBA protocol. Over a range of network conditions, this model makes the same rebroadcast/drop decisions as SBA 87% of the time.

2.2. The ad hoc broadcast protocol

Like SBA, the ad hoc broadcast protocol (AHBP) is in the neighbor knowledge family of protocols. Whereas SBA can be called a "local" neighbor knowledge protocol because each mobile node makes its own decision whether to rebroadcast or not, AHBP is a "nonlocal" neighbor knowledge protocol because a mobile node receives the instruction whether to rebroadcast or not in the header of the packet it receives. Since AHBP is based on another protocol, multipoint relaying, we describe them both in turn.

In multipoint relaying [12], rebroadcasting MNs are explicitly chosen by upstream senders. The chosen MNs are called Multipoint Relays (MPRs) and they are the only MNs allowed to rebroadcast a packet received from the sender. An MN chooses its MPRs as follows [12].

(1) Find all two-hop neighbors reachable by only one one-hop neighbor. Assign those one-hop neighbors as MPRs.
(2) Determine the resultant cover set—neighbors receiving packets from the current MPR set.
(3) Add to the MPR set the uncovered one-hop neighbor that will cover the most uncovered two-hop neighbors.
(4) Repeat steps 2 and 3 until all two-hop neighbors are covered.

Multipoint relaying is described in detail in the optimized link state routing (OLSR) protocol, an internet RFC [43]. In that implementation, the addresses of the selected MPRs are included in "hello" Packets.

In the ad hoc broadcast protocol (AHBP) [11], only MNs designated as a broadcast relay gateway (BRG) within the header of a broadcast packet are allowed to rebroadcast the packet. The algorithm for a BRG to choose its BRG set is identical to that used in multipoint relaying to choose MPRs, which is a sequence of steps that greedily approximates the MCDS. AHBP differs from multipoint relaying in two significant ways.

(1) In AHBP, when an MN receives a broadcast packet and is listed as a BRG, the MN uses two-hop neighbor knowledge to determine which 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 next-hop BRGs.

(2) 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.

While both SBA and AHBP use two-hop neighbor knowledge to infer node coverage, they use this knowledge in different ways. In SBA, when a node *receives* a broadcast or rebroadcast packet, it assumes that other neighbors of the sender *have been* covered. In AHBP, when a node *sends* a broadcast or rebroadcast packet, it assumes that neighbors of the designated BRG nodes *will be* covered.

2.3. The limitations of static protocols

An extensive evaluation of several broadcast protocols via simulation using NS-2 [44] is compiled in [15]. The goals were to compare the protocols over a range of network conditions, pinpoint areas where each protocol performs well, and identify areas where they need improvement. As a result of the study, higher assessment delay was found to be effective in increasing the delivery ratio of SBA in congested networks. Since a lower assessment delay is desired in noncongested networks (to reduce end-to-end delay), the balance proposed in [15] was to develop an adaptive SBA scheme; specifically, if the MN is receiving more than 260 packets per second on average, the MN uses a RAD with a $T_{\max}$ value of 0.05 seconds. Otherwise, the MN uses a RAD with a $T_{\max}$ value of 0.01 seconds. This simple adaptive SBA scheme leads to performance measures outperforming the original SBA scheme and AHBP under high congestion. Figure 2 illustrates how nodes implementing SBA protocol that switches between two different RADs can deliver more broadcast packets to the network over the range of traffic loads studied. The figure also demonstrates the fragility of SBA: if the value of $T_{\max}$ is set too low and there is no mechanism to adapt it, the performance of SBA can decline rapidly with increasing congestion. The difference is so dramatic that it represents the difference between SBA being the worst or the best protocol under high traffic [15].

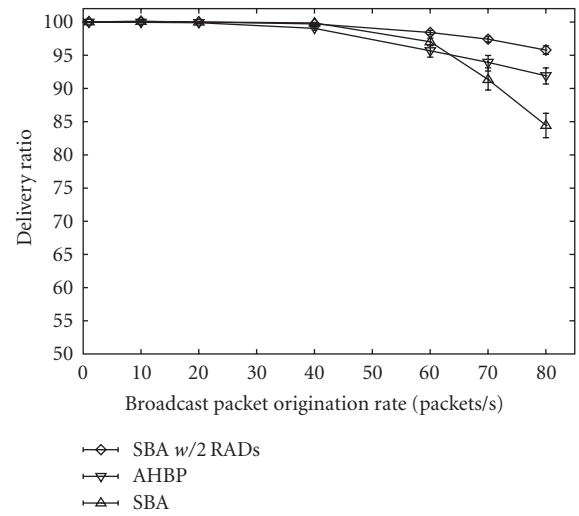


Figure 2: The delivery ratio of SBA is sensitive to the length of the RAD chosen by each mobile node. SBA performs the best when mobile nodes can choose a long RAD during high traffic and a short RAD during low traffic. This figure is recreated from a study performed in [15].

2.4. MANET intelligence

In the previous section, we motivated our approach for applying machine learning to broadcasting by providing an example in which an unsophisticated adaptive rule—a single "if" statement—outperformed static protocols. In this section, we give more background on the use of intelligent methods in MANETs for the purpose of explaining what is unique to the problem of broadcasting, and why we believe that more sophisticated methods can lead to further improvements.

Attempts to promote intelligence in MANETs usually involve application layer programs and intelligent agents [45], or autonomous vehicle projects, for example, [46], which also communicate at the application layer. These types of applications try to achieve complex goals and make multistep decisions. By comparison, *the broadcasting problem is a simple, single-step decision* (retransmit a packet or not) that must be made repeatedly. Because of the high frequency of actions taken and almost immediate feedback given during broadcasting, we argue that our models have more opportunity for online learning. Although our goals are not as ambitious as application layer autonomy, we believe that learning is more attainable.

At the network level, unicast routing algorithms have been analyzed [47] for the possibility of adaptation, but not to the extent of online, uniquely instantiated machine learning models for every MN as we propose herein. Instead,

unicast routing handles uncertainty by estimating a cost of routing a packet to an MN through a particular link and applying dynamic programming to compute the least cost route to each destination. When costs can be communicated easily without overhead, for example, included in ACK packets, cost-based routing has been shown to provide higher throughput than traditional routing algorithms.

3. A PURE MACHINE LEARNING BROADCASTING PROTOCOL

We exhibit a new approach to the design of a broadcast protocol: the decision of whether to rebroadcast a packet is cast as a classification problem. A classifier is simply a function that maps inputs into discrete outputs, which are called class labels. In this section, we describe our method of designing a classifier for the broadcast problem and how it learns from experience. Training a classifier is merely the process of adjusting how the function maps inputs to outputs, so we also describe how to formulate the inputs and outputs in a way that makes learning most effective.

Our proposed intra- and inter-protocol learning methods (Sections 4 and 5) take proven exiting protocols developed by experts and incorporate machine learning such that they become more robust. We propose a new method from the converse perspective: we will develop a pure machine learning model first and add expert knowledge and heuristics as needed. Using this method, each mobile node will contain an instantiation of a small model that it consults when deciding whether to rebroadcast a packet. Furthermore, we constrain this model to be of a certain type, regardless of how it is implemented: a binary classifier, a model with several inputs but only one output which can only take on two values (call them positive and negative). For each incoming packet, a mobile node will use its model to classify that packet as a positive (retransmit) or negative (disregard) example. In other words, this machine learning strategy will treat the decision to retransmit a packet as a classification task.

Our inspiration for applying machine learning stems from previous work concluding that existing broadcast algorithms are too brittle to support a wide range of MANET environments, and that even the hacked "if-then" rule to adapt the RAD of SBA described in Section 2.3 is more robust. We draw upon our previous work [30] applying Bayesian networks to a MANET. In network-wide broadcasting, mobile nodes must make a repeated decision (to retransmit or not), 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 show that a Bayesian network combines the input features appropriately and often correctly predicts whether to retransmit or not.

We desire that mobile nodes improve automatically through experience and adapt to their environment. Therefore, we require an objective function that assesses whether a given mobile node is beneficially contributing to the network's delivery of broadcast packets. Each MN 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 MN made a good decision or not. To this end, we define the concept of a successful retransmission.

Successful retransmission

For a 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 broadcasting 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 rebroadcasting 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 the broadcasts of its neighbors, and therefore it collects this feedback without any communication overhead.

We identify two ways in which mistakes can be made, with language borrowed from signal detection theory.
Type I error: If node A retransmits packet X, and then hears neighbor node B retransmiting a copy of X it received elsewhere, node A will incorrectly infer a successful retransmission. These "false-positive" errors are more common with increasing congestion because B receives more duplicate copies of X.
Type II error: If, for example, node A is near the edge of the network and delivers a packet X to neighbor B, which is also on the edge, then B might decide not to retransmit the packet (Because it has no other neighbors). Node A will incorrectly assume that this was an unsuccessful retransmission. We rarely find this type of "false negative" error whenever A has more than one neighbor, but it is more common when A has only one neighbor. (Since we implement a protocol with neighbor knowledge, we could choose to ignore unsuccessful retransmissions on nodes with only one neighbor.)

We collect retransmit data in the naive Bayes model shown in Figure 3. Naive Bayes models are special cases of Bayesian networks consisting of one parent node with the action or classification and several children nodes that make up the input features. They have the advantage over full Bayesian networks in that they are computationally simple and efficient with respect to space and CPU evaluation. We take $\oplus$ to denote a successful rebroadcast and $\ominus$ to denote an unsuccessful one, and each MN must consider each candidate hypothesis, $h \in \{\oplus, \ominus\}$. The Bayesian approach to classify a new broadcast packet is to choose the hypothesis with the highest posterior probability, also known as the (*maximum a posteriori*) hypothesis, h_{MAP}, given the n data attributes of the broadcast packet $(d_1, d_2, \ldots, d_n)$ that describe it. By applying Bayes' theorem, we arrive at the expression

$$\begin{aligned} h_{\text{MAP}} &= \underset{h\in\{\oplus,\ominus\}}{\operatorname{argmax}} P(h \mid d_1, d_2, \ldots, d_n) \\ &= \underset{h\in\{\oplus,\ominus\}}{\operatorname{argmax}} \frac{P(d_1, d_2, \ldots, d_n \mid h)P(h)}{P(d_1, d_2, \ldots, d_n)} \\ &= \underset{h\in\{\oplus,\ominus\}}{\operatorname{argmax}} P(d_1, d_2, \ldots, d_n \mid h)P(h). \end{aligned} \tag{1}$$

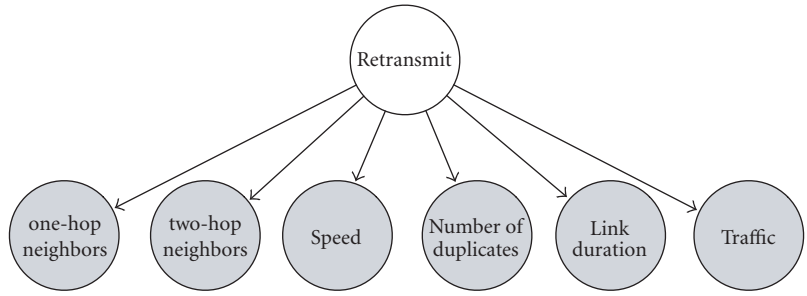


Figure 3: Naive Bayes model of successful retransmissions. Circles represent random variables and arrows denote conditional independence relationships. (Arrows do not show the direction of information flow.) Inference in a Bayesian model is the process of estimating the unknown values of the unshaded circles ("retransmit" in the figure) with respect to the known shaded ones.

The naive Bayes assumption is that the input features are conditionally independent given the action. Therefore, we can approximate the MAP hypothesis, $h_{\text{MAP}} \approx h_{\text{NB}}$, when the naive Bayes assumption is true:

$$h_{\text{NB}} = \underset{h \in \{\oplus, \ominus\}}{\operatorname{argmax}} P(h) \prod_{i=1}^{n} P(d_i \mid h). \tag{2}$$

Even when this assumption is violated (and the posterior probability estimates are wrong), there are conditions under which naive Bayes classifiers can still output optimal classifications (retransmit or not) [48]. We choose the input features for each broadcast packet based on our experience with the small amount of data that each MN has available to it. The features we found most useful are shown in Figure 3. Each input feature (e.g., speed, the number of one-hop neighbors, etc.) maintains two tables: one conditional on successful retransmissions and one conditional on unsuccessful retransmissions. The parent stores one table: the prior probabilities of success and failure. If, for example, a packet X was inferred as a successful retransmission, several tables must be updated: $P(\oplus)$, the prior probability of success; $P(\text{1-hop neighbors} \mid \oplus)$; $P(\text{2-hop neighbors} \mid \oplus)$; $P(\text{speed} \mid \oplus)$; and so on. The MN estimates the number of neighbors, traffic, and speed at the time of the successful retransmission. The tables that the input features store can be approximated and smoothed by replacing them with probability distributions. Deciding whether to rebroadcast or drop a packet is simple. Equation (2) 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 (2) for our model in Figure 3 requires seven table look-ups and six multiplications. The tabular data structures and threshold decision procedure (rebroadcast or not) require less storage and computation than other broadcast protocols, such as SBA [10] and AHPB-EX [11], which both use graph-theoretic algorithms.

An attractive feature of the naive Bayes model is that the likelihood entries, $P(\text{1-hop neighbors} \mid \oplus)$, $P(\text{speed} \mid \oplus)$, and so on, can be used to answer questions in a post hoc manner. For example, given that Node A decides that the retransmission of packet X was unsuccessful, which hypothesis can best explain why? Candidate hypotheses 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. Another useful feature is that there is diversity in the behavior of the MNs because they have different training experience. This means that each MN has its own classifier and naturally allows for some MNs to be more successful rebroadcasters of packets. An MN's priors and likelihoods,

$$P(\oplus) \tag{3}$$

and $P(d_i \mid \oplus)$, are updated through a node's membership in the network.

In this section, we designed a broadcast protocol based around a naive Bayes machine learning model. To this model, we added some expert knowledge about broadcasting and MANETs in general; we formulated the inputs to the classifier using variables we believe affect network performance. In the next two sections, we take a different approach: we take fully formed broadcast protocols that have been designed by human experts, and we try to add flexibility and adaptability to them. The flexibility and adaptability will come from the same place as in this section, by deploying small machine learning models on each of the nodes. As we described in this section, the naive Bayes model is conceptually simple and computationally efficient, and we will apply other models in this spirit. Using simple models is appropriate in this setting for several reasons. First, these models must be deployed on resource-constrained devices. Second, we are working with small dimensionality in the input and output spaces, where more complicated machine learning models would probably be overkilled and would probably overfit the data. Last, we want to spread the acceptance and adoption of machine learning methods by demonstrating that they can be applied simply, in which the benefits are achieved because of the dynamic nature of the environment and not any special ability hidden in the model.

4. INTRA-PROTOCOL LEARNING

In the previous two sections, we have described protocols designed by human experts and protocols that learn their behavior, respectively. In this section, we present the first of two new classes of broadcast protocols that use a hybrid approach; we employ an existing broadcast protocol and make it adaptive by using machine learning models. We call our first approach intra-protocol learning because a mobile node learns to change one of the free parameters *inside* a broadcast protocol. By contrast, we categorize MNs that can automatically learn to switch *between* different broadcast protocols as inter-protocol learners, and we discuss that method in Section 5.

With the exception of simple flooding, all the broadcast protocols we have identified in the literature have at least one *free parameter*, which we define as a parameter that the network programmer or implementer is free to set. Several studies in [15] confirm that the performance of a broadcast protocol is sensitive to the values of its free parameters. Moreover, the optimal value of a parameter varies as network conditions change. The value of $T_{\max}$ in the SBA protocol (Section 2.3) is a single example of how much improvement can be attained by properly setting a parameter and how different environments require different values.

We believe that the number of possible intra-protocol learning protocols is large; whereas the number of pure machine learning broadcast protocols relatively bounded by the number of reasonable classifiers, there can be as many intra-protocol learners as there are relevant and sensitive free parameters. We present two candidate protocols in this section. In Section 7, we use simulation results to assess the performance of our first candidate.

4.1. Adapting RAD-based protocols to density and congestion

We have noted earlier that the $T_{\max}$ parameter controls the length of SBA's RAD, and that this parameter is sensitive to the density of neighboring mobile nodes and congestion [10, 15]. We propose that a mobile node implementing SBA uses a simple regression model to estimate the value of $T_{\max}$ that is most appropriate for that node and its local conditions. While the naive Bayes classifier from Section 3 is a function that maps seven inputs into a discrete output, our present regression function will map two inputs into a continuous output. We choose two inputs to the regression, $\mathbf{x} = [x_1, x_2]^T$, where x_1 is the number of packets a node receives per second, and x_2 is the number of one-hop neighbors a node has. These inputs are a node's estimation of its local congestion and density, and each of these inputs can be computed easily and without extra communication overhead. After trying different forms of the regression function, we found the following equation to be both accurate and simple:

$$\hat{T}_{\max} \longleftarrow w_0 + w_1 \log x_1 + w_2 \frac{1}{x_2}, \tag{4}$$

where $\hat{T}_{\max}$ is the estimate of the correct upper bound on the RAD and the values of the coefficient vector, $\mathbf{w} = [w_0, w_1, w_2]^T$, are found during training. To train the model, we ran 25 different simulations, consisting of all combinations of 5 levels of congestion and 5 levels of node density. (See Section 6 for parameter values.) During each simulation, we choose one node at random and spotlight its behavior throughout the simulation to gather our training examples. All the other nodes in the network run the SBA algorithm described in Section 2.1, including the enhancement proposed by [15]. A single training example is created when the following conditions are met. When the spotlighted node receives a broadcast packet, it takes note of its estimates of number of packets received per second and the number of one-hop neighbors (x_1 and x_2), and implements SBA by covering its one- and two-hop neighbors. When not all of the spotlighted node's neighbors are covered after receiving the broadcast and any subsequent rebroadcasts, a training example is created. Along with x_1 and x_2, the node stores $\hat{T}_{\max}$, which is the node's estimate of the correct upper bound on its RAD. According to our method, the node chooses $\hat{T}_{\max}$ as twice the length of time between when a node receives the first copy of a broadcast packet and when it receives the last copy. Recall that nodes implementing SBA chose a RAD randomly in the uniform range $[0, T_{\max})$, so the expected value of the length of the RAD is half of $T_{\max}$. Thus, but choosing our estimate, $\hat{T}_{\max}$, as twice the interval that it takes for a node to receive all copies of a broadcast packet, we aim to ensure that nodes will wait before broadcasting most of the time. In the special cases when the node hears only one copy of the broadcast packet and no subsequent rebroadcasts (so that it cannot compute a length of time to double), we estimate $\hat{T}_{\max}$ as 0.01 seconds. Out of all the $(x_1, x_2, \hat{T}_{\max})$ generated during a simulation, we choose 40 at random, and over all the 25 simulations, these data comprise a training set of 1000 entries. Even though (4) is nonlinear, we treat it as a linear equation on the transformation of the inputs in order to learn $\mathbf{w} = [w_0, w_1, w_2]^T$ by least squares. That is, we write our training data as a matrix $\mathbf{D}$ and vector $\mathbf{y}$, where

$$\mathbf{D} = \begin{bmatrix} 1 & (\log x_1)_1 & \left(\frac{1}{x_2}\right)_1 \\ 1 & (\log x_1)_2 & \left(\frac{1}{x_2}\right)_2 \\ \vdots & \vdots & \vdots \\ 1 & (\log x_1)_{1000} & \left(\frac{1}{x_2}\right)_{1000} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} (\hat{T}_{\max})_1 \\ (\hat{T}_{\max})_2 \\ \vdots \\ (\hat{T}_{\max})_{1000} \end{bmatrix}. \tag{5}$$

Then the standard least squares solution for $\mathbf{w}$ is

$$\mathbf{w} = \left(\mathbf{D}^T\mathbf{D}\right)^{-1}\mathbf{D}^T\mathbf{y}. \tag{6}$$

We do not claim that the training procedure or the estimate of $\mathbf{w}$ is optimal, but they are simple and work well empirically.

4.2. *Adapting nonlocal decision neighbor knowledge methods to mobility*

Whereas we studied SBA in the previous subsection, we investigate AHBP here for the opportunity to improve its performance through learning. Recall that while AHBP is a neighbor knowledge broadcast protocol, it is part of the nonlocal decision family; nodes implementing AHBP do not decide whether to rebroadcast or not, but instead are instructed whether to do so in the header of the packet it receives.

AHBP-EX (which is AHBP plus the extension for mobility, described in Section 2.2) provided the best performance in the most severe network environment studied in [15]. Unfortunately, its sensitivity to node mobility produces the lowest delivery ratio in networks where the environment is dominated by topological changes. AHBP-EX requires a MN which receives a packet from an unrecorded neighbor (i.e., a neighbor not currently listed as a one-hop neighbor) to act as a broadcast relay gateway (BRG). In other words, AHBP-EX handles the case when a neighbor moves inside another node's transmission range between "hello" intervals. The extension does not handle the case when a chosen BRG is no longer within the sending node's transmission range. No recourse is provided in AHBP-EX to cover the two-hop neighbors that this absent BRG would have covered. That is, outdated two-hop neighbor knowledge corrupts the determination of next-hop rebroadcasting MNs.

We propose to model high mobility by annotating each entry in a node's neighbor table with a confidence measure. This confidence measure represents the belief that a given entry in the neighbor table really is a node's neighbor at that moment in time. The most straightforward confidence measure is a simple probability that if the node sends a packet, the given entry in the neighbor table will receive it. If these probabilities can be inferred accurately, an MN can make more conservative decisions on which MNs should rebroadcast. While we do not implement this protocol, we expect that training will reveal heuristics to estimate the confidence values. By finding the *expected* number of neighbors, the MNs estimate of density will be less. These confidence values will be based on features such as local node speed and total number of neighbors. For example, the confidence value is set to 1 when a "hello" packet is received; the value then exponentially decays at a rate determined by the heuristics learned in training.

5. INTER-PROTOCOL LEARNING

With sufficient training data and expert knowledge, it is possible to train an MN to switch from one broadcasting protocol to another that is more suitable. In this section, we create an inter-protocol learner to automatically switch an MN between SBA and AHBP. We are switching between two complicated neighbor knowledge protocols to demonstrate that it is possible, but there are certainly simpler inter-protocol broadcastings that are just as useful. Any combination of broadcasting protocols that do not have specialized headers would be a good candidate, such as simple flooding, probabilistic, counter-based, distance-based, or location-based, (see [29] for descriptions of these schemes). One obvious inter-protocol learner is to use any advanced broadcasting protocol whenever possible, but fall back to simple flooding when the network conditions are too extreme. In the present case, however, we hope to combine SBA and AHBP into a protocol that performs better than either one individually because we know that neither one is always better than the other over a wide range of simulations [15].

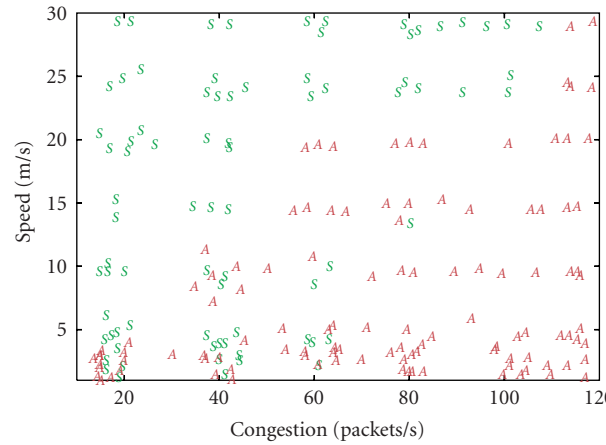


Figure 4: Training data for the inter-protocol learner is are the form of (speed, congestion) pairs as input, and whether SBA (green "S") or AHBP (red "A") performed better at that input. The inter-protocol learner uses these training data to build a model to switch between the protocols.

Both SBA and AHBP have special conditions that require the protocol to default to retransmit. Recall that if an SBA node receives a packet from a new neighbor, it is unlikely to know of any common one- or two-hop neighbors previously reached; thus the node is more likely to rebroadcast. In other words, local decision neighbor knowledge methods appear to adapt to mobility more easily than nonlocal decision neighbor knowledge methods. However, local decision neighbor knowledge methods (such as SBA) suffer more from congestion than nonlocal decision neighbor knowledge methods. Thus, we develop a protocol that will combine the benefits of these two types of neighbor knowledge methods. Specifically, a node in this combined protocol will track the amount of congestion and its speed to decide which protocol to use.

Figure 4 shows the training data we collected to train our inter-protocol learner. We ran SBA and AHBP simulations over the range of speeds and congestion levels given in Table 3 and the number of nodes fixed at 50, and we measured the delivery ratio of each node. Each data point in the figure represents five nodes, where we clustered the data by finding the five nearest neighbors and plotting the point at the centroid of each cluster; for each of the five nodes we take a majority vote of which protocol had the best delivery ratio, and we color the point with a green "S" if SBA was better

Table 1: SBA, AHBP, and the combined packet header of our inter-protocol learner.

SBA	AHBP	Inter-protocol
Packet type	Packet type	Packet type
Send time	Send time	Send time
Node ID	Node ID	Node ID
Packet route	Packet route	Packet route
Neighbor nodes	Neighbor nodes	Neighbor nodes
Neighbor count	Neighbor count	Neighbor count
—	BRG nodes	BRG nodes
—	BRG count	BRG count
Hop count	Hop count	Hop count
Origin address	Origin address	Origin address
Destination address	Destination address	Destination address
Data length	Data length	Data length
Node X/Y position	—	Node X/Y position

and with a red "A" if AHBP was better. We chose clusters of five nodes to eliminate some of the noise in the data, but the overall pattern is not sensitive to this choice. Note that because of the mobility model used in creating this training data (see Section 6), the data is a bit striated in bands across the speeds we studied, and that a large portion of the data is collected at low speeds of 0, 1, and 5 m/s. Also note that because we are plotting the centroids of clusters of five nodes, the exact location of the plotted points may be far from some node's actual behavior. Although the data are noisy, there exist intuitive patterns to build a model on. For this data, we will build a decision tree, in similar form to the one shown in Figure 1. Like that decision tree, our inter-protocol learner will use a *univariate* decision tree, meaning that it can ask about only one variable at a time. The consequence is that a decision tree separating the data in Figure 4 can draw only vertical and horizontal lines, so our inter-protocol learner will make a stair-step pattern roughly starting in the lower left corner and continuing to the upper right. For an environment of high speed and low congestion, a node will use SBA, and it will use AHBP when it encounters low speed and high congestion. When both speed and congestion are low or high, the delivery ratio is nearly equally good and bad, respectively, and the choice does not matter that much. (The model tends to choose SBA under low speed and low congestion and choose AHBP under high speed and high congestion.) The more critical choices are in the middle of Figure 4, and these are also the cases in which the network will have some nodes running SBA and some running AHBP.

To facilitate a node switching between SBA and AHBP, we make some changes to the structure and behavior of a broadcast packet. As shown in Table 1, the two protocols have a similar header format, so our combined header is the union of the two, with the most notable change including the BRG information from AHBP. Specifying the header also explains most of a node's behavior; it must implement a subset of both protocols, enough to fill the headers in a packet.

Table 2: Simulation details common to all trials.

Input parameters	
Simulation area size	300 m × 600 m
Transmission range	100 m
Derived parameters	
Node density	1 node per 3,600 m^2
Coverage area	31,416 m^2
Transmission footprint	17.45%
Maximum path length	671 m
Network diameter (max. hops)	6.71 hops
Mobility model	
Mobility model	Random waypoint [49]
Mobility speed	1, 5, . . . , 25, 30 m/s ±10%
Simulator	
Simulator used	NS-2 (version 2.1b7a)
Medium access protocol	IEEE 802.11
Number of repetitions	10
Confidence interval	95%

Table 3: Simulation parameters investigating network severity.

Trial	1	2	3	4	5	6	7
Number of nodes	40	50	60	70	90	110	150
Avg. speed (m/s)	1	5	10	15	20	25	30
Pkt. Src. Rate (pkts/s)	10	20	40	60	80	100	120

Number of nodes	Avg. number of neighbors
40	7.6
50	9.1
60	11.2
70	13.9
90	18.1
110	23.6
150	31.3

When sending a packet, a node must specify the BRG nodes, whether that node is implementing AHBP or not. This is not too much extra work for a node implementing SBA because that node already knows its uncovered two-hop neighbors, so it simply chooses the BRG in a greedy way. When a node receives a packet, it can choose to ignore the BRG fields in the header if it is implementing SBA, or follow them if it is implementing AHBP. In this way, the local behavior of SBA is preserved, and so is the nonlocal behavior of AHBP.

6. SIMULATION ENVIRONMENT

We believe that defining and explaining our three classes of adaptive protocols are a more important contribution than the details of the three specific protocols we created, but by simulating our protocols we confirm the concepts presented

in the preceding sections. We use the same NS-2 simulation parameters as [15]; see Tables 2 and 3 for details. In particular, we report results testing increasing network severity with respect to density, mobility, and congestion, according to Table 3. The MNs move according to the random waypoint mobility model, and their positions were initialized according to that model's stationary distribution (see [49] for details). In subsequent studies in which we vary only a single parameter, we choose to hold the others constant at their trial 3 values.

When simulating our pure machine learning protocol, we run our naive Bayes feedback mechanism in reverse, turning it into a naive Bayes classifier. For a fixed set of input features, the model in Figure 3 estimates the posterior probability of success. If the posterior probability is greater than 0.5, the strategy that minimizes errors on average is to retransmit the packet. As a node gets more local experience, it automatically adapts to the network by changing the entries in its prior and likelihood probability tables.

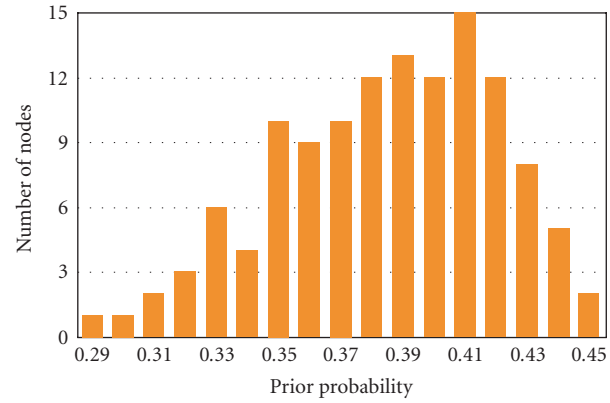


Figure 5: Histogram showing the spread of the prior probability of rebroadcasting, $P(\oplus)$. MNs have different priors because they have their own training examples from the MANET. Data are taken from trial 1 of Section 7.1.

7. RESULTS

We test three hypotheses in the following subsections, one for each learning method we propose. We demonstrate that our methods are indeed learning what we designed them to, and by doing so, that our protocols perform better than or equivalent to the static ones we derived them from. While we believe that the benefits of using machine learning are in the design phase, such as leading to simpler protocol designs that are more robust to change, the fact that they also are more efficient further advocates their adoption.

We also compare our three learned protocols to each other in Section 7.4. Since we have created only one example from each of our three learning methods, we expect that optimal protocols have yet to be found. Our comparison, however, informs on what performance can be attained from a protocol given the effort needed to create it, and we present this information to give insight and advice on what protocols should be used going forward. We expand on this insight and advice in the concluding section.

7.1. *Pure machine learning over increasing network severity*

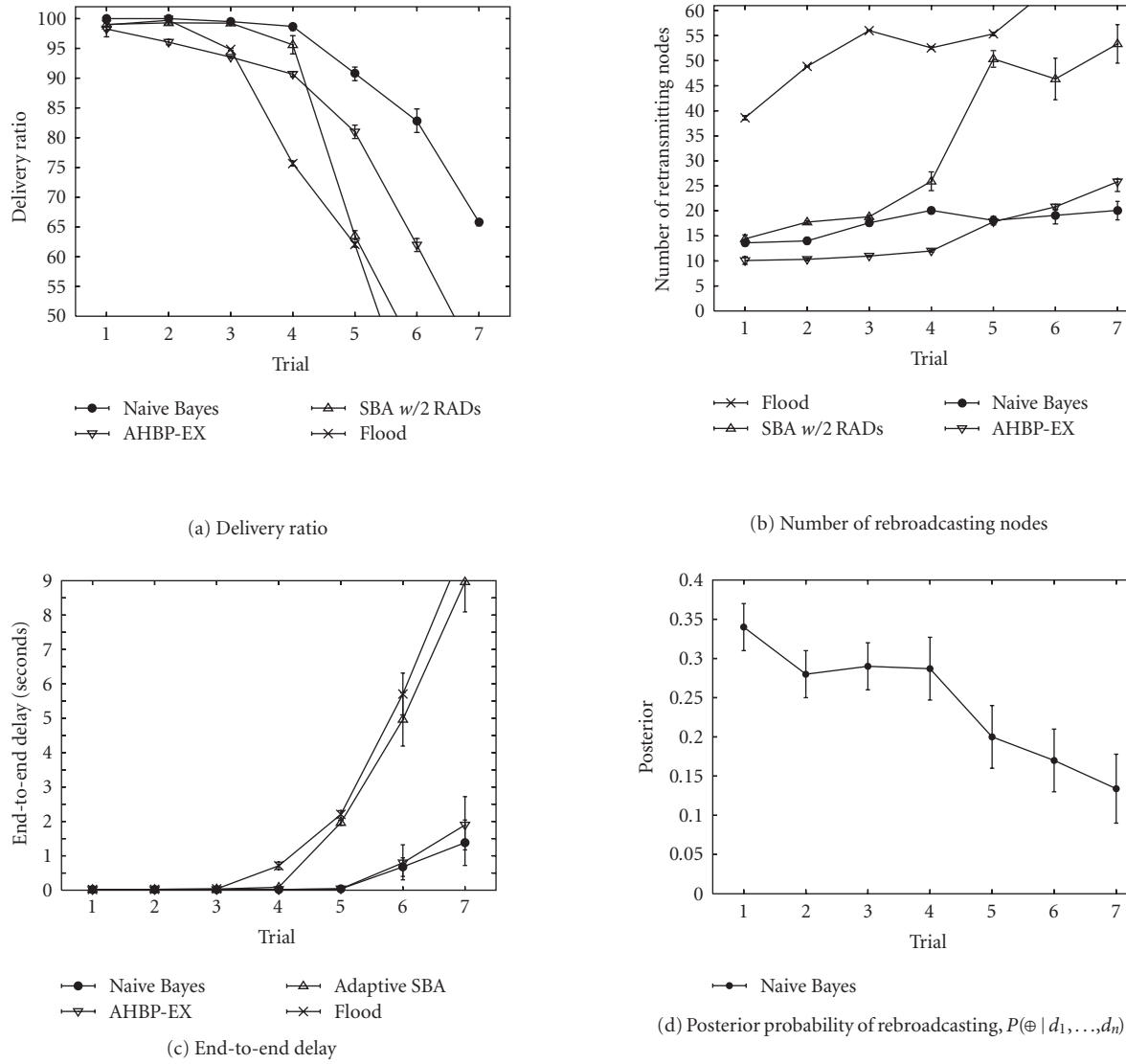
We created our naive Bayes broadcasting protocol to demonstrate that the pure machine learning method can be used to create a protocol that is robust over varied network conditions. To test this hypothesis, we replicate the most informative studies from [15] in which network severity increased from the combined effects of mobility, congestion, and node density. Table 3 shows that network severity increases as the trial number increases.

As shown in Figures 5 and 6, our naive Bayes broadcast protocol outperforms SBA and AHBP-EX, maintaining a high delivery ratio and low overhead. (Extremely poor results are clipped from the figure to preserve detail.) In all the trials, it maintains the highest or second-highest delivery ratio, and it is the only protocol that does not fail catastrophically reaching a "breaking point." In trials 1–4, AHBP-EX uses fewer rebroadcasting nodes, but also has a worse delivery ratio. In the extremely taxing trials, 5–7, the naive Bayes broadcast is the best in terms of delivery ratio and the number of rebroadcasting nodes. In these scenarios, MNs under SBA broadcast far too often as shown in Figure 6(b). The naive Bayes protocol, however, can adjust its prior probability of rebroadcasting, as shown in Figure 5, which shows the spread of prior distributions. The result is that very few MNs will rebroadcast, also shown in Figure 6(d), in which the posterior probability of a successful rebroadcast (2) decreases.

To ensure that there are no hidden effects from varying density, speed, and congestion at the same time, we vary them individually in Figures 7(a)–7(c), holding the others constant at their trial 3 values. We observe the same effects noted in [50], namely that AHBP-EX's performance decreases with increasing speed (because its two-hop neighbor knowledge is out of date), and SBA's delivery ratio decreases with increasing congestion (because its RAD is too short). As in Figure 6, our naive Bayes protocol has the highest delivery ratio.

7.2. *Intra-protocol learning over increasing congestion*

By creating our adaptive SBA protocol, we want to show that an intra-protocol learning method that automatically sets one sensitive parameter can perform better than setting that parameter by hand. We chose to learn $T_{\max}$ as specified in (4) by regression with $\mathbf{w} = [0.081, 0.011, 0.134]^T$ found by least squares. A node computes $T_{\max}$ using the instantaneous congestion and a number of neighbors, and we know that its value is sensitive to congestion. In Figure 8, we show how our learned protocol compares to static SBA and an adaptive SBA with only two different values of $T_{\max}$. At high levels of congestion, the delivery ratio is higher in the learned protocol



(a) Delivery ratio

(b) Number of rebroadcasting nodes

(c) End-to-end delay

(d) Posterior probability of rebroadcasting, $P(\oplus \mid d_1, \ldots, d_n)$

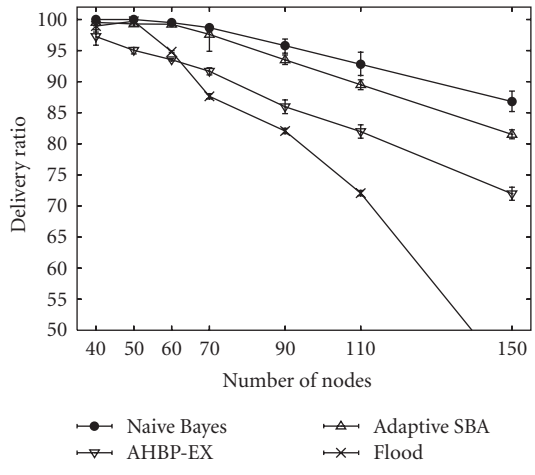
FIGURE 6: Performance of our naive Bayes protocol. Over a range of increasing network severity, our broadcast protocol built with a naive Bayes classifier maintains a high delivery ratio, low overhead, and low delay. In the most extreme scenario, it performs the best in all categories. All nodes compute a posterior probability of rebroadcasting, which decreases in these trials.

because $T_{\max}$ increases as congestion increases, so fewer unnecessary duplicate packets are sent. Figure 8(d) shows that our learned protocol has a smaller RAD than the simple adaptive protocol at all rates except for the last one, and the consequence is that our learned protocol has a smaller delay but more rebroadcasting nodes. At 80 packets per second, the relative size of the RADs is reversed and hence our learned protocol has a longer delay. In this simulation, we fixed the number of nodes at 60, the payload of size of each packet was set to 64 bytes, and the network was static, identical to [15].

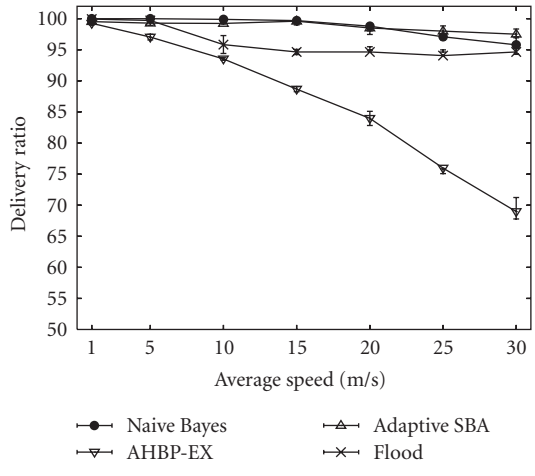
7.3. Inter-protocol learning over increasing speed

To test our inter-protocol learning method, we ask whether our protocol that switches between SBA and AHBP can perform at least as well as each protocol individually. If we hold the congestion level constant at 60 packets per second and increase the speed, we suspect that nodes will transition from AHBP to SBA. Figure 9 shows that nodes indeed do switch to SBA at higher speeds, but the figure also shows a performance gain over either protocol individually in some cases. (To remove any congestion effects, we used a null MAC, so the end-to-end delay is equivalent for all three protocols.) Figure 9(b) shows that there is some additional overhead cost in switching between protocols in terms of the number of rebroadcasting nodes. As Table 1 shows, there is also extra overhead in terms of the number of bytes per broadcast packet.

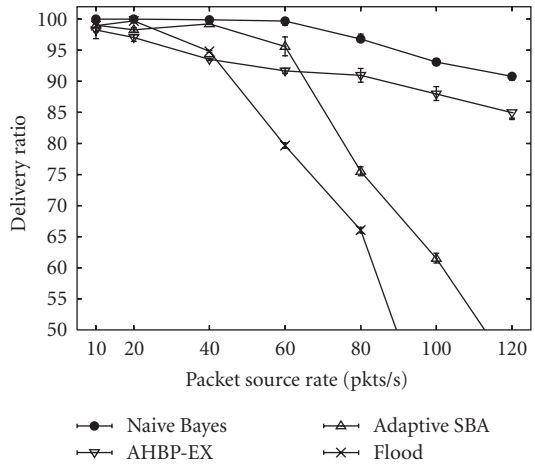
As expected, Figure 9(c) shows that nodes more often chose AHBP at low speeds and SBA at high speeds, and at the extreme cases the protocol follows AHBP or SBA completely. At moderate speeds, the behavior of our learned protocol



(a) Increasing density



(b) Increasing speed



(c) Increasing congestion

Figure 7: The delivery ratio of our naive Bayes broadcasting protocol is also the highest while varying density, speed, and congestion separately.

exhibits more of a gradual transition than a quick switch, and these are the same cases in which our protocol has a higher delivery ratio than AHBP or SBA. Providing a node with the flexibility to implement the protocol that is best suited for its local conditions is the cause of the improvement.

7.4. Comparing all three learned protocols

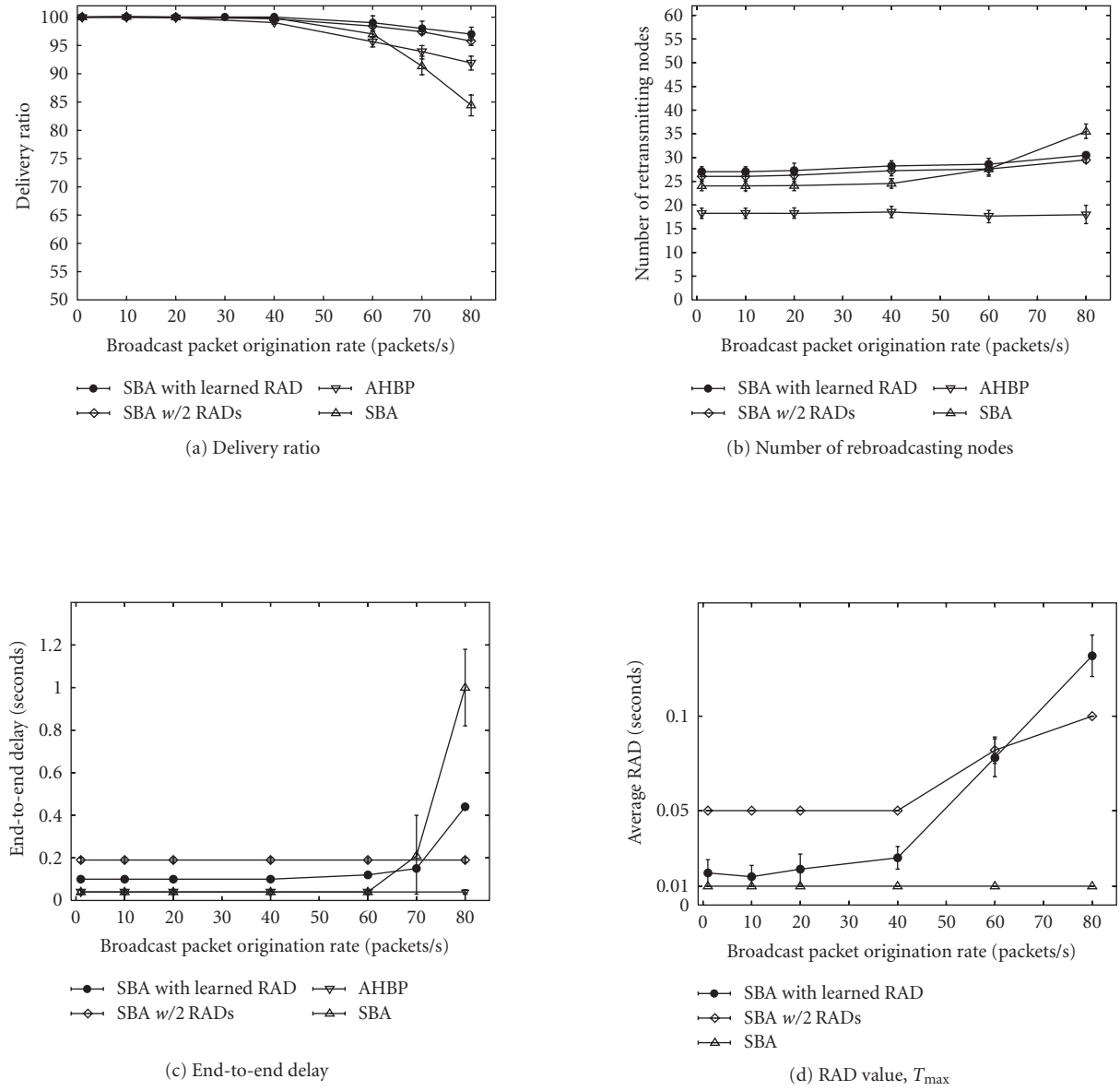
We believe that our learning methods can be applied to diverse network conditions to find an optimal broadcast protocol, and we have simulated many network conditions in this paper with our three exemplar protocols. We return to the increasing network severity study of Section 7.1 to compare our three learned protocols because that study is the most effective in separating out their performance. While they all perform well in some specific scenarios, this study chooses a wide range of network conditions, so it may give advice in the absence of any knowledge of the deployed network conditions.

Figure 10 shows that the naive Bayes protocol is the robust over varying network conditions, which is what we observed in Section 7.1. Our protocol that switches between AHBP and SBA becomes almost purely SBA at high speed, so its poor performance in trials 5, 6, and 7 is not surprising. Remember that the learning in the switching protocol is confined to the switching itself, because this is an inter-protocol learner; we do not modify the behavior of AHBP or SBA once the protocol has decided to use it. When we perform intra-protocol learning and learn SBA's RAD, the performance in terms of delivery ratio, overhead, and delay are improved relative to the switching, inter-protocol learning.

8. CONCLUSIONS

Advanced broadcast algorithms have been known in the literature, yet many unicast, multicast, and geocast protocols are still built using simple flooding. We believe that this gross misuse of network resources has perpetuated for two reasons. First, simple flooding is trivial to implement, while previous broadcast protocols require graph theory. Second, although previous broadcast protocols all outperformed simple flooding, there was no clear winner between them, especially under severe network conditions. Our naive Bayes protocol addresses both of these issues; it has a table-based implementation that is simple and efficient, and it is a robust performer over many network scenarios. Indeed, the naive Bayes protocol performed better than either of the other learned protocols we created. We believe that the results in Figure 6 display an optimized trade-off between delivery ratio and overhead. If, on the other hand, a network application specified that the delivery ratio was more important than overhead, or vice versa, that knowledge can be directly incorporated into the model in Figure 3. The prior probability of rebroadcasting, for example, can be scaled up or down.

We believe that our most important contribution is the identification of ways machine learning can be applied to broadcasting. We have identified three new classes of broadcasting protocols, and believe that there are many implementations that are ripe for discovery in each class. Just as there are many protocols under the heading "neighbor



(a) Delivery ratio

(b) Number of rebroadcasting nodes

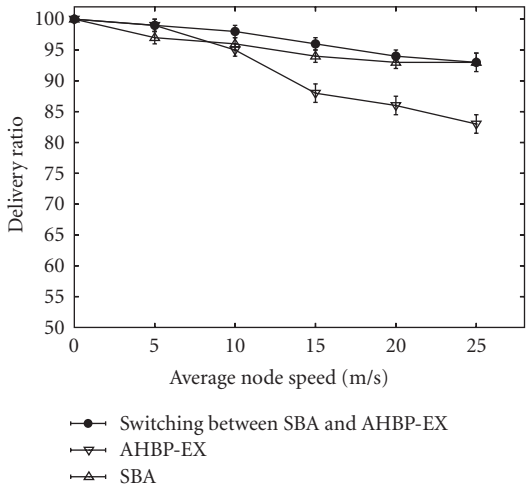
(c) End-to-end delay

(d) RAD value, $T_{\max}$

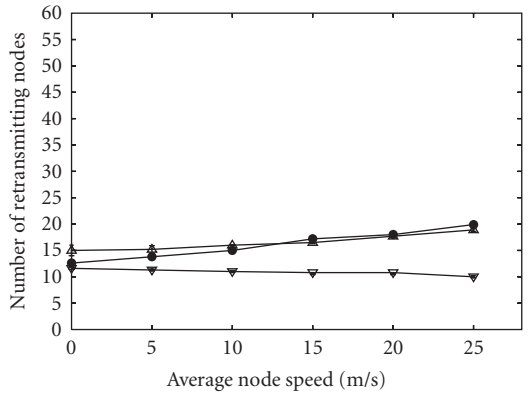
FIGURE 8: The intra-protocol learning method adjusts its value of $T_{\max}$ with changing congestion. Over the range of congestion levels studied, the learned protocol had the highest delivery ratio.

knowledge" and "location-based," we believe that there are many improvements to be found to pure machine learning, intra-protocol learning, and inter-protocol learning methods. None of the protocols we created under these headings add any extra communication overhead to the protocols themselves (aside from a slightly longer header in our intra-protocol learner), but perhaps future directions would consider the constraint of zero communication overhead to be too stringent. Allowing some communication between machine learning models on MNs might lead to lower overall communication. Moreover, we have not applied machine learning to other classes of broadcast protocols, which are important to consider when two-hop neighbor knowledge is not feasible or otherwise available.

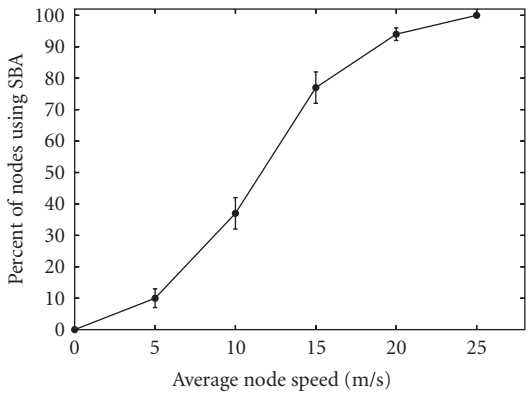
We hope to encourage improvements to other network-layer protocols through the use of our work by creating a taxonomy in which others can create simple, yet effective broadcast protocols. We conclude with a guideline on how to choose which class of adaptive protocols to use. If you are willing to create the protocol from scratch, a pure machine learning protocol is a simple one to create. If you are willing to change the internals of a known protocol, use an intra-protocol learner to modify its internals. If you have access to multiple protocols and are averse to modifying them, inter-protocol learning requires little additional effort except for possibly changing the packet header (which is often unnecessary). We believe that choosing any of these methods is more desirable than choosing a static broadcasting protocol.



(a) Delivery ratio

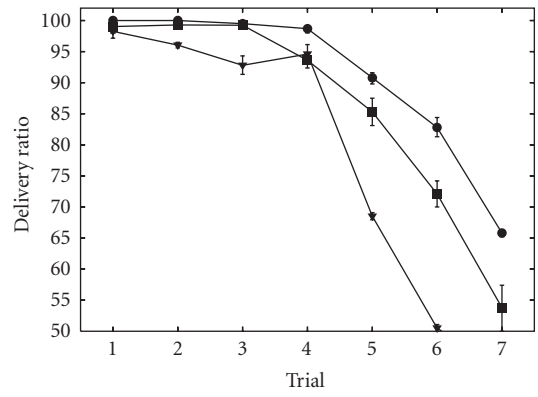


(b) Number of rebroadcasting nodes

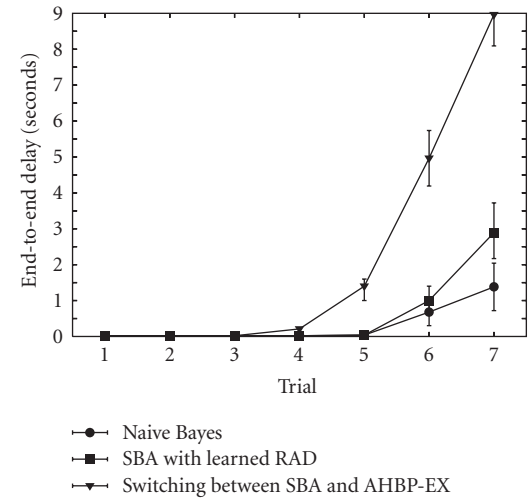


(c) Percent of nodes using SBA

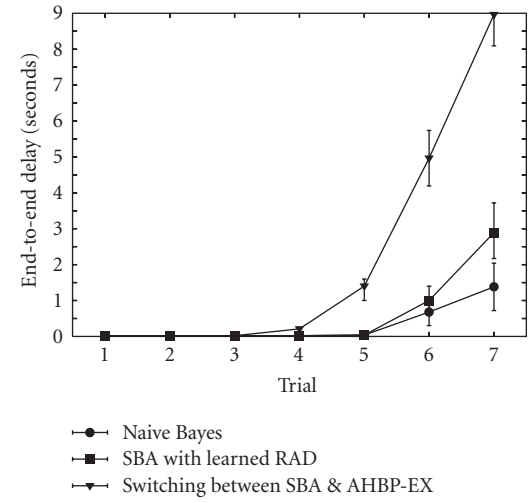
Figure 9: Nodes using the inter-protocol learning method switch between AHBP and SBA. For a fixed level of congestion, nodes prefer SBA at higher speeds.



(a) Delivery ratio



(b) Number of rebroadcasting nodes



(c) End-to-end delay

Figure 10: Comparing all three of our learned protocols over increasing network severity, we find that our naive Bayes learner is the most robust.

REFERENCES

[1] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward, "A distance routing effect algorithm for mobility (DREAM)," in *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98)*, pp. 76–84, Dallas, Tex, USA, October 1998.

[2] C.-C. Chiang and M. Gerla, "Routing and multicast in multihop, mobile wireless networks," in *Proceedings of the IEEE 6th International Conference on Universal Personal Communications (ICUPC '97)*, vol. 2, pp. 546–551, San Diego, Calif, USA, October 1997.

[3] C. Chiang, H. Wu, W. Liu, and M. Gerla, "Routing in clusterhead multihop, mobile wireless networks with fading channel," in *Proceedings of the IEEE Singapore International Conference on Networks (SICON '97)*, pp. 197–211, Singapore, April 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 '99)*, pp. 64–71, Seattle, Wash, USA, August 1999.

[5] H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc networks," in *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM '00)*, pp. 61–68, Boston, Mass, USA, August 2000.

[6] H. Lim and C. Kim, "Flooding in wireless ad hoc networks," *Computer Communications*, vol. 24, no. 3-4, pp. 353–363, 2001.

[7] C. R. 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] E. Pagani and G. P. Rossi, "Providing reliable and fault tolerant broadcast delivery in mobile ad-hoc networks," *Mobile Networks and Applications*, vol. 4, no. 3, pp. 175–192, 1999.

[9] W. Peng and X.-C. Lu, "Efficient broadcast in mobile ad hoc networks using connected dominating sets," *Journal of Software*, vol. 12, no. 4, pp. 529–536, 2001.

[10] W. Peng and X.-C. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," in *Proceedings of the 1st ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC '00)*, pp. 129–130, Boston, Mass, USA, August 2000.

[11] W. Peng and X.-C. Lu, "AHBP: an efficient broadcast protocol for mobile ad hoc networks," *Journal of Computer Science and Technology*, vol. 16, no. 2, pp. 114–125, 2001.

[12] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying: an efficient technique for flooding in mobile wireless networks," Rapport de Recherche 3898, INRIA, Cedex, France, 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," CAIP Technical Report 248, CAIP Center, Rutgers University, Piscataway, NJ, USA, September 2000, http://www.caip.rutgers.edu/~marsic/mobile/.

[15] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC '02)*, pp. 194–205, Lausanne, Switzerland, June 2002.

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

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

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

[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 (WMCSA '99)*, pp. 90–100, New Orleans, La, USA, February 1999.

[20] Z. Haas, "New routing protocol for the reconfigurable wireless networks," in *Proceedings of the IEEE 6th International Conference on Universal Personal Communications (ICUPC '97)*, vol. 2, pp. 562–566, San Diego, Calif, USA, October 1997.

[21] Z. Haas and M. Pearlman, "The performance of query control schemes for the zone routing protocol," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '98)*, pp. 167–177, Vancouver, BC, Canada, August-September 1998.

[22] Z. Haas and B. Liang, "Ad-hoc mobility management with randomized database groups," in *Proceedings of the IEEE International Conference on Communications (ICC '99)*, vol. 3, pp. 1756–1762, Vancouver, BC, Canada, June 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 '98)*, pp. 66–75, Dallas, Tex, USA, October 1998.

[24] S. Corson 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] C.-C. Chiang and M. Gerla, "On-demand multicast in mobile wireless networks," in *Proceedings of the International Conference on Network Protocols (ICNP '98)*, pp. 262–270, Austin, Tex, USA, October 1998.

[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 '99)*, vol. 3, pp. 1298–1302, New Orleans, La, USA, September 1999.

[27] T. Camp and Y. Liu, "An adaptive mesh-based protocol for geocast routing," *Journal of Parallel and Distributed Computing*, vol. 63, no. 2, pp. 196–213, 2003, Special issue on routing in mobile and wireless ad hoc networks.

[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 '99)*, pp. 101–110, New Orleans, La, USA, February 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 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '99)*, pp. 151–162, Seattle, Wash, USA, August 1999.

[30] M. D. Colagrosso, "A classification approach to broadcasting in a mobile ad hoc network," in *Proceedings of IEEE International Conference on Communications (ICC '05)*, vol. 2, pp. 1112–1117, Seoul, Korea, May 2005.

[31] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Francisco, Calif, USA, 1997.

[32] I. S. Committee, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," in *IEEE 802.11 Standard*, IEEE Computer Society, New York, NY, USA, 1996.

[33] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, no. 1–3, pp. 165–177, 1990.

[34] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *Proceedings of IEEE International Conference on Communications (ICC '97)*, vol. 1, pp. 376–380, Montreal, Que, Canada, June 1997.

[35] B. Das, R. Sivakumar, and V. Bharghavan, "Routing in ad hoc networks using a spine," in *Proceedings of the 6th International Conference on Computer Communications and Networks (IC-CCN '97)*, pp. 34–39, Las Vegas, Nev, USA 1997.

[36] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," in *Proceedings of the 4th Annual European Symposium on Algorithms (ESA '96)*, pp. 179–193, Barcelona, Spain, September 1996.

[37] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, 1998.

[38] R. Sivakumar, B. Das, and V. Bharghavan, "An improved spine-based infrastructure for routing in ad hoc networks," in *Proceedings of the 3rd IEEE Symposium on Computers and Communications (ISCC '98)*, Athens, Greece, June-July 1998.

[39] R. Sivakumar, B. Das, and V. Bharghavan, "Spine routing in ad hoc networks," *Cluster Computing*, vol. 1, no. 2, pp. 237–248, 1998.

[40] P.-J. Wan, K. M. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, vol. 3, pp. 1597–1604, New York, NY, USA, June 2002.

[41] J. Wu and H. Li, "On calculating connected dominating sets for efficient routing in ad hoc wireless networks," in *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication (DIAL-M '99)*, pp. 7–14, Seattle, Wash, USA, August 1999.

[42] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, Calif, USA, 1993.

[43] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," October 2003, request for comments 3626.

[44] K. Fall and K. Varadhan, Eds., *UCB/LBNL/VINT network simulator—ns (version 2)*, 1997, accessed on November 2002, http://www-mash.cs.berkeley.edu/ns/.

[45] G. Christensen, "Intelligent mobile networks," accessed on July 2002, http://www.mobilein.com/intelligentmobilenetworks.htm.

[46] B. Bellur, M. Lewis, and F. Templin, "An ad-hoc network for teams of autonomous vehicles," in *Proceedings of 1st Annual Symposium on Autonomous Intelligent Networks and Systems (AINS '02)*, Los Angeles, Calif, USA, May 2002.

[47] M. E. Streenstrup, "Routing under uncertainty: a comparative study," in *Proceedings of IEEE Wireless Communications and Networking Conference*, vol. 1, pp. 112–116, Chicago, Ill, USA, September 2000.

[48] 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 (ICML '96)*, pp. 105–112, Bari, Italy, July 1996.

[49] W. Navidi, T. Camp, and N. Bauer, "Improving the accuracy of random waypoint simulations through steady-state initialization," in *Proceedings of the 15th International Conference on Modeling and Simulation (MS '04)*, pp. 319–326, Marina Del Rey, Calif, USA, March 2004.

[50] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC '02)*, pp. 194–205, Lausanne, Switzerland, June 2002.