

Performance Analysis of Broadcast Protocols in Ad Hoc Networks Based on Self-Pruning *

Fei Dai and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

Abstract

Self-pruning is an effective scheme for efficient broadcasting in ad hoc wireless networks. In a self-pruning broadcast protocol, a node may not forward a broadcast packet if a certain self-pruning condition is satisfied based on the neighborhood information. In a static network with an ideal MAC layer, only a subset of nodes forward the broadcast packet and still guarantee the complete network delivery. Various protocols have been proposed with different self-pruning conditions. Recently, a generic self-pruning protocol was proposed by Wu and Dai [21], which combines the strength of previous conditions and is more effective. In this paper, we first propose an enhanced version of the generic protocol, which is more elegant in interpreting existing protocols and has a simpler correctness proof. Then we evaluate the performance of the family of self-pruning protocols under various network situations with ns2. The objective is to observe the efficiency and reliability of these protocols as a function of network density, congestion, and mobility, and provide a guideline of implementation in the “real world”. Our performance analysis reveals that the protocol reliability is barely affected by packet collision. However, most self-pruning protocols suffer from low delivery ratio in highly mobile networks. We further explore various techniques that improve the delivery ratio and show that both high efficiency and reliability can be achieved in highly mobile networks.

Keywords: Ad hoc networks, broadcasting, localized algorithms, ns-2 simulation, self-pruning.

*This work was supported in part by NSF grants CCR 9900646, CCR 0329741, ANI 0073736, and EIA 0130806.
Email: {fdai,jie}@cse.fau.edu.

1 Introduction

Ad hoc wireless networks (or simply ad hoc networks) are dynamic in nature. Due to this dynamic nature, global information/infrastructure such as minimal spanning tree is no longer suitable to support broadcasting in ad hoc networks. *Flooding* is a simple approach to broadcasting without global information/infrastructure; in flooding, a broadcast packet is forwarded exactly once by every node in the network. Blind flooding ensures the coverage; the broadcast packet is guaranteed to be received by every node in the network, providing there is no packet loss caused by collision in the MAC layer and there is no high speed movement of nodes during the broadcast process. However, due to the broadcast nature of wireless communication media, redundant transmissions in blind flooding may cause the *broadcast storm problem* [18], in which redundant packets cause contention and collision.

Self-pruning is an effective method in reducing broadcast redundancy. Unlike flooding, in self-pruning-based broadcast protocols [2, 3, 10, 13, 16, 17, 22], each node collects neighborhood topology information (i.e., *static information*) via exchanging “Hello” messages and extracts broadcast history information (i.e., *dynamic information*) from incoming broadcast packets. Each node decides its role in a specific broadcasting: it either becomes a *forward node* and forwards the broadcast packet, or becomes a non-forward node (i.e., is self-pruned) and does nothing. Collectively, forward nodes, including the source node, form a *connected dominating set* (CDS) and ensure the coverage. A set of nodes is a dominating set if every node in the network is either in the set or a neighbor of a node in the set. If the decision is made based on only static information, the corresponding protocol is a *static protocol*; otherwise, it is a *dynamic protocol*. Both static and dynamic protocols are *localized methods*, that is, the decision made on each node does not rely on global network information or any network infrastructures that exhibit “sequentialized propagation” in their building process. Although self-pruning does not provide a constant approximation ratio to the optimal solution, it exhibits similar average efficiency to several non-localized broadcast algorithms that provide constant approximation ratios.

In this paper, we evaluate the performance of the family of self-pruning protocols under various situations with network simulator *ns2* [4]. Our objective is to observe the efficiency (in reducing the number of forward nodes) and reliability (in terms of delivery ratio) of these protocols as a function of network density, congestion, and mobility, and determine if these protocols are practical in the “real world”. The second objective is to study the effects of several implementation options on the performance of the generic protocol and provide a guideline of tradeoffs under different network settings.

Our work has been inspired by two recent findings [20, 21]. In [21], a generic self-pruning scheme was proposed to combine the strength of several existing broadcast protocols. Five existing broadcast algorithms [2, 3, 16, 17, 22] were shown to be special cases of a *coverage condition*. A simulation study was conducted to compare the performance of the coverage condition and its special cases, as well as to examine the effects of several implementation options. However, those protocols were simulated in networks without collision or mobility. Since 100% delivery ratio is guaranteed in such “ideal” networks, evaluation on reliability was not conducted. In [20], Williams and Camp simulated a self-pruning protocol called Scalable Broadcast Algorithm (SBA) with *ns2*, and compared it with another CDS-based localized broadcast protocol called Ad Hoc Broadcast Protocol (AHBP) [11]. Their simulation results suggested that SBA has lower efficiency, causes higher end-to-end delay, and is more resilient to mobility. A variation of SBA called the neighbor coverage scheme, which is proposed and simulated by Tseng et al [19], shows similar properties in the simulation. However, SBA has too many unique properties to be a typical self-pruning protocol. More optimizing options are yet to be explored via a simulation study of the entire self-pruning family.

The generic protocol in [21] is used in our simulation study as a framework for comparing existing self-pruning protocols and various implementation options. In addition, we propose an enhanced version of the coverage condition called the *self-pruning rule*, which is more elegant in interpreting some existing protocols and has a simpler correctness proof. Seven existing protocols, including five protocols listed in [21], are shown to be special cases of the self-pruning rule. These protocols are simulated and compared with blind flooding and a new protocol derived from the self-pruning rule. AHBP uses another important broadcast method called *neighbor-designating* [7, 8, 12, 11], where the status of a node is determined by its neighboring forward nodes. A more general scheme that covers both self-pruning and neighbor-designating methods, and the performance comparison among them, are part of our future work and will not be discussed in the paper.

Our simulation study consists of two parts. In the first part, we consider three sources of unreliability: collision, contention, and mobility. Simulation results show that collision is not a serious problem when a small forwarding jitter delay ($\leq 1ms$) is used. The contention level can also be minimized by using the most efficient protocol to reduce redundancy. However, most existing protocols exhibit relatively low delivery ratio in mobile networks. One drawback of the self-pruning method is that it demands accurate neighborhood information and cannot ensure the coverage with outdated topology information. Without special treatment, most self-pruning protocols are not suitable for highly mobile networks. We try to solve this problem in the second part of our simulation

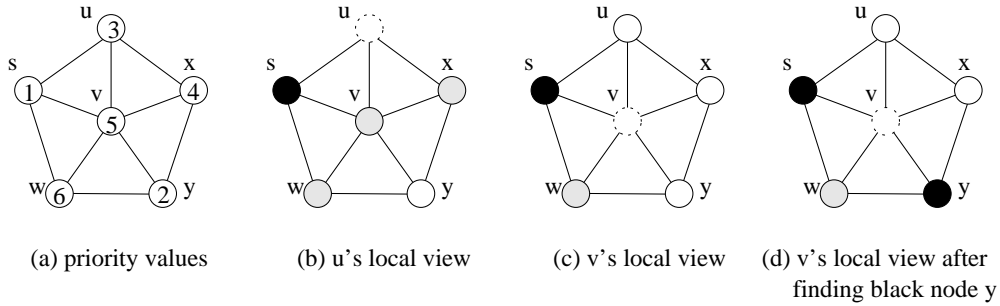


Figure 1. Local information for nodes u and v .

study. Five implementation techniques that improve reliability in mobile networks are discussed, and their effectiveness are evaluated. Four of them have been used previous protocols, and the fifth is newly proposed. Simulation results show that high reliability can be achieved with high efficiency in highly mobile networks.

The remainder of this paper is organized as follows: Section 2 reviews the original coverage condition and related implementation options. Section 3 introduces the new self-pruning rule. Section 4 examines seven existing self-pruning protocols as special cases of the self-pruning rule. Section 5 gives simulation results on the performance of existing protocols, and Section 6 evaluates various implementation options that improve reliability in mobile networks. Section 7 concludes this paper.

2 Preliminary

We consider an ad hoc network as a graph $G = (V, E)$, where V is a set of nodes and E is a set of bidirectional links. For each node v , $N(v) = \{u \mid (u, v) \in E\}$ denotes its neighbor set. A broadcast process can be denoted by the set of the forward nodes $F \subseteq V$. A broadcasting is *successful* if every node receives the broadcast packet; that is, $V - F \subseteq N(F)$, where $N(F) = \bigcup_{v \in F} N(v)$. We say a broadcast protocol *ensures the coverage* if it guarantees successful broadcasting, providing that G is connected and there is no topology change or packet loss during the broadcast process (the latter condition is relaxed in our simulation on $ns-2$). It is relatively easy to ensure the coverage with a large F (e.g. blind flooding) or with a small F which is based on expensive global information (e.g. MCDS [5]). The problem is to determine a small forward node set based on affordable local information.

Local information collected at each node can be divided into two categories: *static information*, including neighborhood topology and a certain node attribute that serves as a priority value, and

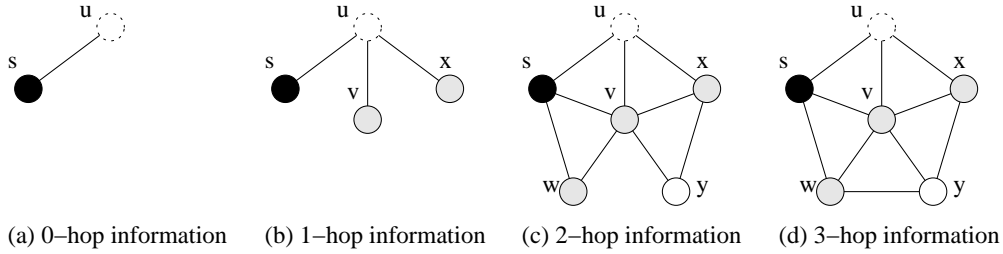


Figure 2. Neighborhood topology of node u .

dynamic information, including a small set of nodes that have forwarded the broadcast packet. The static information is independent of any broadcasting and can be collected by periodically exchanging “Hello” messages among neighbors. The priority values are used to establish a total order among nodes. High priority nodes usually bear more responsibilities in a broadcast process than do low priority nodes. The dynamic information depends on each broadcasting and is carried by the broadcast packet. More formally, for each broadcasting, the local information L_v collected at a node v is a triple (G_v, p, F_v) , where $G_v = (V_v, E_v)$ is a subgraph of G that usually represents the topology of a small vicinity, p is the priority function on V_v , and $F_v \subseteq F \cap V_v$ represents a list of forward nodes extracted from incoming broadcast packets. For the sake of clarity, we call nodes in F_v *black nodes*, non-black nodes with higher priorities than v *gray nodes*, and all other nodes *white nodes*. Note that the coloring of nodes is relative and depends on the view of each node. A gray node in the view of one node could be a white node in the view of another node. For example, in Figure 1, node x is a gray node in the view of node u but a white node for node v .

In the original generic self-pruning scheme [21], each node decides its own status (forwarding/non-forwarding) independently based on the following condition.

Coverage Condition: Node v is pruned (i.e., has a non-forward node status) if for any two neighbors u and w , a *replacement path* exists that connects u and w with black and gray intermediate nodes only.

According to the coverage condition, node u in Figure 1 (b) can be pruned, because its neighbor v is directly connected with neighbors s and x , and s and x are connected via a replacement path (s, v, x) with only gray node v as an intermediate node. Node v in Figure 1 (c) is not pruned, as no such replacement path can be used to connect neighbors s and x . It was proved in [21] that the coverage condition ensures the coverage. Five existing self-pruning protocols [2, 3, 16, 17, 22] were shown to be special cases of the coverage condition with different implementation options, including:

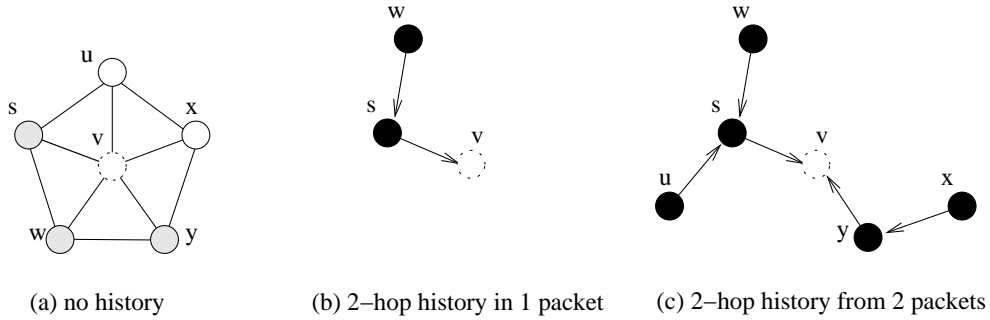


Figure 3. Piggybacked history information.

Neighborhood topology: i.e., G_v in each node v collected via exchanging “Hello” messages periodically among neighbors. We use the term “ k -hop information” to denote the topology information that can be collected after k rounds of “Hello” message exchanges. Figure 2 shows neighborhood topologies with different values of k . Generally, k -hop topology information includes nodes within k hops, links between any two nodes within $k - 1$ hops, and links between a node k hops away and a node $k - 1$ hops away. The coverage condition requires at least 2-hop information to apply. Using k -hop information with $k > 2$ can slightly increase the pruning efficiency, but also causes higher message overhead and slower convergence.

Priority value: i.e., $p(v)$ of each node v advertised in the “Hello” messages. We say $p(v)$ is a k -hop priority value if it depends on k -hop information of node v . For example, node id is a 0-hop priority value because it is independent of any topology information. Node degree, defined as the number of neighbors, is a 1-hop priority value. Neighborhood connectivity ratio (NCR), defined as the ratio of pairs of directly connected neighbors to pairs of any neighbors, is a 2-hop priority value. Node degree and NCR were proposed to reduce the number of forward nodes in relatively sparse networks. However, they also cause slower convergence.

Backoff delay: A node v is more likely to be pruned if it has a larger black node set F_v in its local information. For example, node v in Figure 1 can be pruned only after it identifies two black nodes in its neighborhood. The backoff delay scheme postpones the testing of the self-pruning rule for a backoff delay, hoping that new black nodes can be observed forwarding the same broadcast packet. Figure 1 (c) shows the situation when the first packet comes from node s , and Figure 1 (d) shows that another copy of the same packet is received from node y . Backoff delay has the drawback of longer overall delay.

Piggybacked history: Piggybacking a list of recently visited nodes in the broadcast packet is an inexpensive method to increase the number of black nodes in F_v . There are three available

options: 0-hop history (i.e., no black node, as shown in Figure 3 (a)), 1-hop history where the id of the last visited node can be extracted from the sender field of the incoming packet (Figures 1 (b) and 1 (d)), and k -hop history where id's of the last $k - 1$ visited nodes are piggybacked into the broadcast packet (Figure 3 (b)). In the last option, k is the same as that used in collecting k -hop information. Note that the piggybacked history can be combined with the backoff delay. For example, Figure 1 (d) combines backoff delay and 1-hop history information, and Figure 3 (c) combines backoff delay and 2-hop history information.

We say a self-pruning protocol is a static protocol if it does not collect or use any dynamic information; otherwise, it is a dynamic protocol. A static protocol has only gray nodes, whereas a dynamic protocol has both gray and black nodes. For example, node v in Figure 3 (a) can be pruned by a static protocol. The benefit of static protocols is that the forward node set can be applied prior to any broadcasting, which reduces the computation overhead. A more important consideration is to form a relatively stable CDS (also called backbone [14, 15]) that facilitates unicasting and multicasting as well. The drawback is that static protocols usually produce a larger CDS than the dynamic ones. Although the original coverage condition in [21] covers both static and dynamic protocols, it encountered difficulty in interpreting the situation with multiple disconnected black nodes as in SBA [10]. Furthermore, the correctness proof was complex and focused on static protocols.

Simulation results in [21] showed that the coverage condition is more efficient than existing self-pruning protocols. It was also shown that the pruning efficiency can be further improved using k -hop information ($k \geq 3$), h -hop broadcast history information ($h \geq 2$), and NCR as priority. However, the improvement is slight and does not justify the extra overhead. Note that simulations in [21] focused on efficiency (i.e., how many nodes forward the broadcast packet) instead of reliability (i.e., how many nodes receive the broadcast packet). Therefore, we assumed an ideal MAC layer without collision or contention, and always up-to-date neighborhood information in spite of mobility. This model no longer suffices, however, when the focus shifts to the protocol reliability under congestion and mobility.

3 An Enhanced Generic Self-Pruning Protocol

In this section, we propose an enhanced version of the original coverage condition in [21]. The enhanced condition, called the *self-pruning rule*, provides better interpretations for dynamic protocols and has a simpler proof than the original coverage condition.

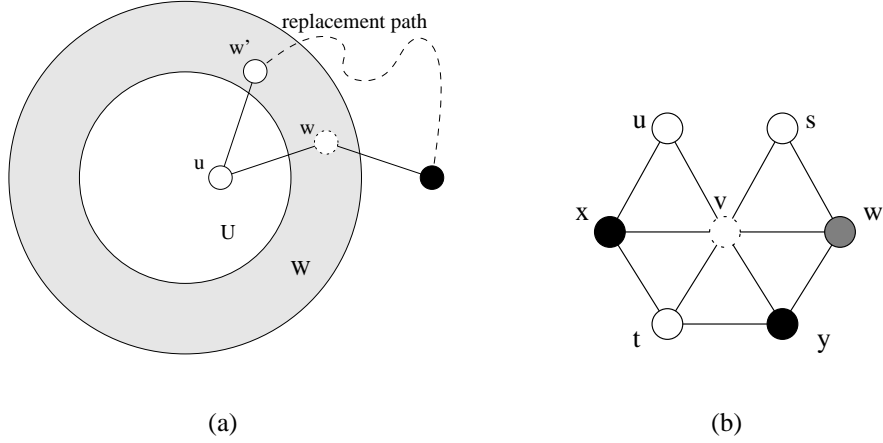


Figure 4. (a) The outer rim W that surrounds U , the set of nodes missing the broadcast packet. (b) Node v is a non-forward node based on the self-pruning rule.

Self-Pruning Rule: A node v is pruned if each of its non-black neighbors u is connected to a black node via a replacement path containing only gray nodes.

That is, if every node $u \in N(v)$ is either a black node, directly connected to a black node that has forwarded the broadcast packet, or indirectly connected to a black node via several gray nodes that have higher priorities than v , then v is a non-forward node; otherwise, it is a forward node. The following theorems show that the self-pruning rule is both correct and at least as powerful as the original coverage condition.

Theorem 1 *The self-pruning rule ensures coverage.*

Proof: We prove the theorem by contradiction. Suppose in a broadcasting that the set of nodes not receiving the broadcast packet, $U = V - F - N(F)$, is not empty, and let $W = N(U) - U$ be the “outer rim” of U that has received and dropped the broadcast packet. Apparently, $W \cap F = \emptyset$. Note that $W \neq \emptyset$; otherwise, a network partition exists that separates U from the source node. Let w be the node in W with the highest priority. From the assumption, there is at least one neighbor $u \in U$ of w , which must be covered by a black node according to the self-pruning rule. Note that any replacement path to u must contain at least one node $w' \in W$ (i.e., must pass the outer rim, as shown in Figure 4 (a)), and w' must be a gray node in w 's view. That contradicts the assumption that $p(w) > p(w')$. □

Table 1. Existing self-pruning protocols as special cases of the generic self-pruning rule.

Protocol	Static Info			Dynamic Info	
	topology	priority	round	delay	hist.
Wu-Li ^a	2	id/degree	2 ^b	No	0
Dai-Wu	2	id/degree	2 ^b	No	0
Span	3	connectivity	5 ^c	No	0
Rieck ^d	2	id	2	No	0
LENWB	2	degree	3	No	1
SBA	2	-	2	Yes	1
Stojmenovic ^a	1 ^e	degree	2	Yes	1

^aAllowing at most two gray nodes in each replacement path.

^bWhen restricted pruning rules are used. That is, $V_v \subseteq N(v)$

^cWhen the enhanced version is used.

^dAllowing at most one gray node in each replacement path.

^eRequiring a GPS device installed on each node.

Theorem 2 *Any nodes pruned by the original coverage condition can also be pruned by the self-pruning rule.*

Proof: The theorem is obviously true for dynamic protocols, as all white and gray nodes are connected to one or more black nodes via replacement paths. In static protocols, if a node v is pruned by the original coverage condition, then every pair of v 's neighbors are connected via gray nodes. In each broadcast process, regardless from which neighbor a node receives a broadcast packet, all other neighbors of v are always connected to this black node with gray nodes. Therefore, the self-pruning rule is still satisfied. \square

In general, the self-pruning rule is more powerful than the original coverage conditions. For example, node v in Figure 4 (b) cannot be pruned in the original coverage condition, as there is no replacement path connecting neighbors u and s . However, node v can still be pruned based on the self-pruning rule, as all white neighbors of v are connected to a black node.

4 Existing Self-Pruning Protocols

We examine seven existing self-pruning protocols as special cases of the self-pruning rule, as listed in Table 1. Most of them, except for Rieck's algorithm [13] and SBA [10] were shown to be special cases of the coverage condition [21] and are briefly discussed here for the sake of

completeness. All seven protocols are simulated and compared with blind flooding and a new protocol derived from the generic scheme in the next section.

Wu and Li’s algorithm (static): Wu and Li [22] proposed a *marking process* to determine a set of *gateways* (i.e., forward nodes) that form a CDS: a node is marked as a gateway if it has two neighbors that are not directly connected. Two pruning rules are used to reduce the size of the resultant CDS. According to pruning Rule 1, a gateway v becomes a non-gateway if all of its neighbors are also neighbors of another node u that has a higher priority value; that is, v ’s neighbor set is *covered* by u . According to pruning Rule 2, a marked node can be unmarked if its neighbor set is covered by two other nodes that are directly connected and have higher priority values. Two types of priority are used: node id and the combination of node degree and node id. A pruning rule is called *restricted* if the neighbor set of a gateway can be covered by its neighbors. 2-hop information is enough to implement the marking process and restricted Rules 1 and 2, and 3-hop information is required to implement non-restricted Rules 1 and 2. Wu and Li’s algorithm is a static protocol; the status of a node is computed only when its neighborhood topology has been changed. The computing time of the marking process on each node is $O(d^2)$ and that of restricted Rules 1 and 2 is $O(d^3)$, where d is the maximal node degree of a network.

Dai and Wu’s algorithm (static): Dai and Wu [3] extended the previous algorithm by using a more general pruning rule called Rule k : a gateway becomes a non-gateway if its neighbor set is covered by k other nodes that are connected and have higher priority values. Rules 1 and 2 are special cases of Rule k where k is restricted to 1 and 2, respectively. An efficient algorithm based on depth-first search was also proposed in [3] to implement the restricted Rule k with 2-hop information and $O(d^2)$ computing time. Simulation results show that the restricted Rule k is almost as efficient as the non-restricted one in reducing the forward node set.

Span (static): Chen et al [2] proposed the *Span* protocol to construct a set of forward nodes (called *coordinators*). A node u becomes a coordinator if it has two neighbors that are not directly connected, indirectly connected via one intermediate coordinator, or indirectly connected via two intermediate coordinators. Before a node changes its status from non-coordinator to coordinator, it waits for a backoff delay which is computed from its energy level, node degree, and the number of pairs of its neighbors that are not directly connected. The backoff delay can be viewed as a priority value, such that nodes with shorter backoff delay have a higher chance of becoming coordinators. Span cannot ensure a CDS since two coordinators may simultaneously change back to non-coordinators and the remaining coordinators may not form a CDS. To conduct a fair comparison of Span and other broadcast algorithms that guarantee the coverage, we use in this paper an

enhanced version of Span, where a node becomes a coordinator if it has two neighbors that are not directly connected or indirectly connected via one or two intermediate nodes with higher priority values. Span uses 3-hop information and requires $O(d^3)$ computing time. Note that Span does not use full 3-hop information, since the condition that two neighbors are indirectly connected via three intermediate coordinators could have been used.

Rieck's algorithm (static): Rieck et al [13] recently proposed a CDS algorithm that can be viewed as a special case of the enhanced Span. In Rieck's algorithm, a node v is in the CDS if it has two neighbors that are not directly connected or indirectly connected via one intermediate node with higher priority than v . Unlike Span, the case that two neighbors are indirectly connected via two intermediate nodes with higher priorities is not considered. Therefore, the resultant CDS is larger than that in Span. On the other hand, Rieck's algorithm requires only 2-hop information and $O(d^2)$ computing time.

LENWB (dynamic): Sucec and Marsic [17] proposed the *Lightweight and Efficient Network-Wide Broadcast* (LENWB) protocol, which computes the forward node status on-the-fly. Whenever node v receives a broadcast packet from a neighbor w , it computes the set C of nodes that are connected to w via nodes that have higher priority values than v . If v 's neighbor set, $N(v)$, is contained in C , node v is a non-forward node; otherwise, it is a forward node. LENWB is a dynamic protocol; the status of each node is re-computed during each broadcast process with 2-hop information and $O(d^2)$ computing time.

SBA (dynamic): Peng and Lu [10] proposed the Scalable Broadcast Algorithm (SBA) to reduce the number of forward nodes. As in LENWB, the status of a forward node is computed on-the-fly. When a node v receives a broadcast packet, instead of forwarding it immediately, v will wait for a backoff delay. For each neighbor w that has forwarded the broadcast packet, node v removes $N(w)$ from $N(v)$. If $N(v)$ does not become empty after the backoff delay, node v becomes a forward node; otherwise, node v is a non-forward node. As in LENWB, SBA uses 2-hop information and $O(d^2)$ computing time.

Stojmenovic's algorithm (hybrid): Stojmenovic et al [16] extended Wu and Li's algorithm in two ways: (1) Suppose every node knows its accurate geographic position, only 1-hop information is needed to implement the marking process and Rules 1 and 2. That is, each node maintains only a list of its neighbors and their geographic positions (connections among neighbors can be derived). (2) The number of forward nodes are further reduced by a neighbor elimination algorithm similar to the one used in SBA. Stojmenovic's algorithm is a hybrid scheme. Before any broadcasting,

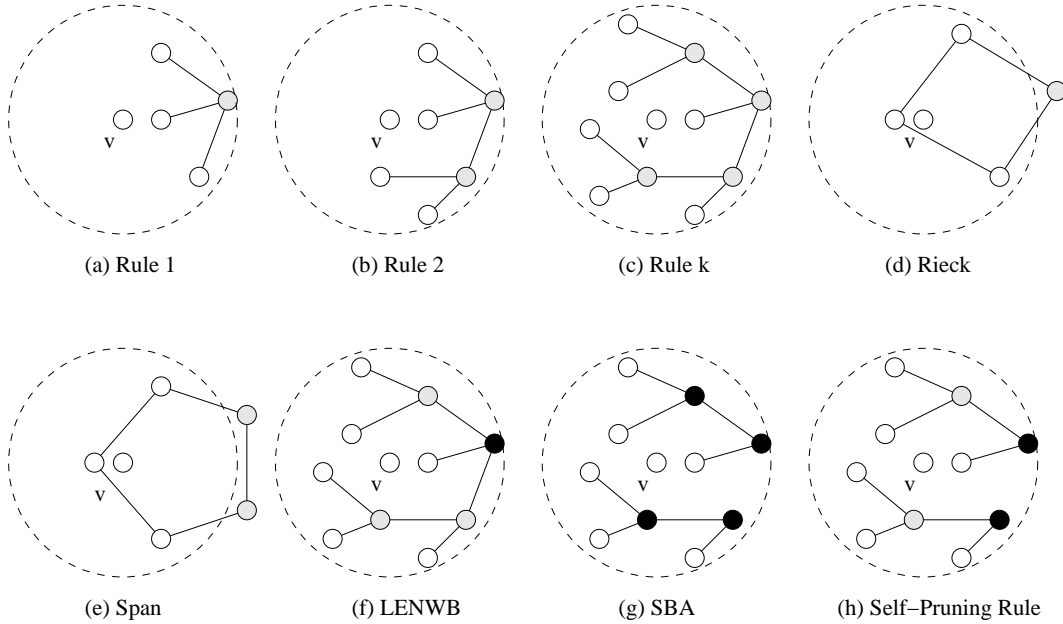


Figure 5. Node v in the center of each subgraph can be self-pruned by the corresponding protocol. Nodes in the transmission range of node v (the dashed circle) are neighbors of v .

a static algorithm (i.e., the marking process and restricted Rules 1 and 2) is applied on all nodes with 1-hop location information and $O(d^3)$ computing time. During each broadcasting, a dynamic algorithm (i.e., SBA) is applied on those nodes, which are temporarily marked as forward nodes by the static algorithm, with 1-hop location information and $O(d^2)$ computing time.

Figure 5 shows self-pruning conditions in above protocols. Node v in subgraphs (a) and (b) can be pruned by Wu and Li's algorithm. Node v in subgraphs (a), (b), and (c) can be eliminated by Dai and Wu's algorithm. Node v in subgraphs (a), (b), (d), and (e) can be removed by Span. Node v in subgraphs (a), (b), and (d) can be taken out by Rieck's algorithm. Note that gray nodes in Figures 5 (d) and (e) are node neighbors of v . All those static algorithms have a stronger pruning condition than the self-pruning rule. Because when v receives a broadcast packet from one of its neighbors, this neighbor (originally a white node) becomes a black node, and all other neighbors of v are connected to this black node, either directly or via a replacement path consisting of gray nodes. Node v in subgraphs (a) to (f) can be eliminated by LENWB. Node v in subgraphs (g) can be removed by SBA. Node v in subgraphs (a), (b), and (g) can be taken out by Stojmenovic's algorithm. Node v in all subgraphs can be pruned by the self-pruning rule. When 2-hop information is used, the generic protocol uses $O(d^3)$ computing time.

Figure 6 illustrates four self-pruning protocols with a small network with 11 nodes, where node

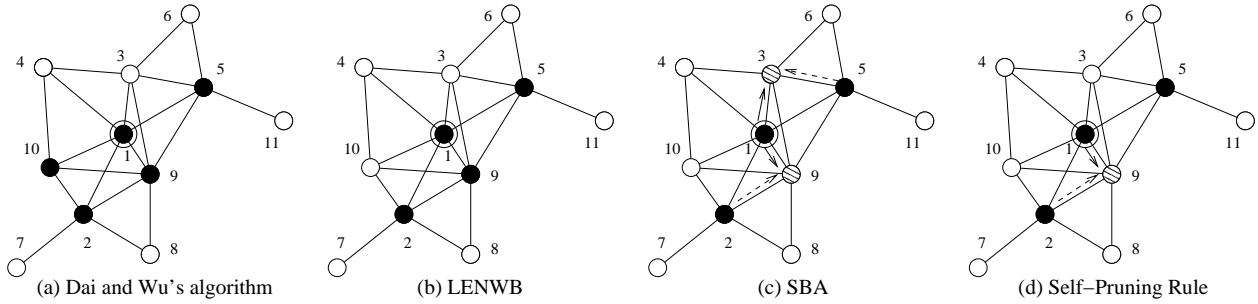


Figure 6. Broadcast processes of different self-pruning protocols.

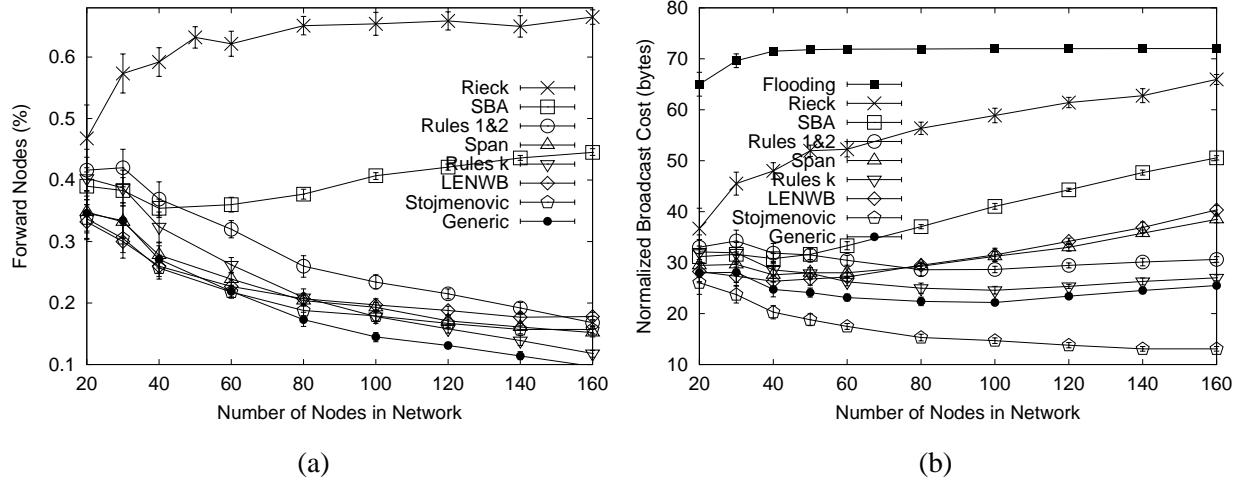
id serves as priority, and a broadcasting starts from node 1. Figure 6 (a) shows the broadcast process of Dai and Wu’s algorithm, which uses no dynamic information. The corresponding forward node set includes all gateway nodes (black nodes) and the source node (the black node marked by a circle). Among non-forward nodes, nodes 6, 7, 8, and 11 are unmarked by the marking process, and nodes 3 and 4 are unmarked by Rule k . Note that node 4 can also be unmarked by Rule 2. Figure 6 (b) demonstrates LENWB, which uses 1-hop history information. Compared with Dai and Wu’s algorithm, node 10 becomes a non-forward node because its neighbors are directly connected to the source node 1. Figure 6 (c) demonstrates SBA, which uses 1-hop history information and a random backoff delay. There are two nodes (shaded nodes) with random forward statuses. If node 3 receives the broadcast packet from only source node 1, it becomes a forward node because its neighbor 6 is not a neighbor of node 1. If node 3 has a smaller backoff delay than node 5, and receives the broadcast packet from both nodes 1 and 5, it becomes a non-forward node. Similarly, node 9 becomes a non-forward node when it has a smaller backoff delay than node 2. Figure 6 (c) shows the broadcast process of the generic protocol using a random backoff delay. Compared with LENWB, node 9 has a 50% probability of becoming a non-forward node. Compared with SBA, node 3 is always a non-forward node regardless of its backoff delay.

5 Performance Analysis

This section presents results from the first part of our simulation, i.e., the major threat to self-pruning protocols under congestion and mobility. Techniques that handle the threat are discussed in the next section. Unlike simulations in [21], simulations in this paper focus on reliability rather than efficiency, and are conducted on *ns-2(1b7a)* [4] and its CMU wireless and mobility extension [6], using the IEEE 802.11 MAC layer, limited queue space in the link layer, and the random waypoint mobility model [1].

Table 2. Simulation parameters

Parameter	Value
Network Area	$900 \times 900 m^2$
Transmission Range	$250 m$
Bandwidth	$2 Mb/s$
Data Packet Size	64 bytes
Maximal IFQ Length	50
Simulation Time	100 s
Number of Trials	20
Confidence Level	95%

**Figure 7. Percentage of forward nodes (a) and normalized broadcast cost (b).**

Nine protocols are simulated, including blind flooding (Flooding), Wu and Li’s algorithm (Rules 1&2), Dai and Wu’s algorithm (Rule k), a variation of Span that ensures the coverage (Span), Rieck’s algorithm (Rieck), LENWB, SBA, Stojmenovic’s algorithm (Stojmenovic), and a new protocol using the generic self-pruning rule (Generic). In order to avoid excessive packet collisions, all protocols use a random jitter between 0 and 1 millisecond before forwarding a received packet. In the default configuration, all self-pruning protocols use 1 second “Hello” interval and collect 2-hop information. The only exception is Stojmenovic’s algorithm, which uses 1-hop location information. Rules 1&2, Rule k and Generic use node id as priority, LENWB and Stojmenovic’s algorithm use node degree, and Span uses NCR. Only SBA and Stojmenovic’s algorithm use a random backoff delay (around $10ms$, depending on local topology). Other simulation parameters are listed in Table 2.

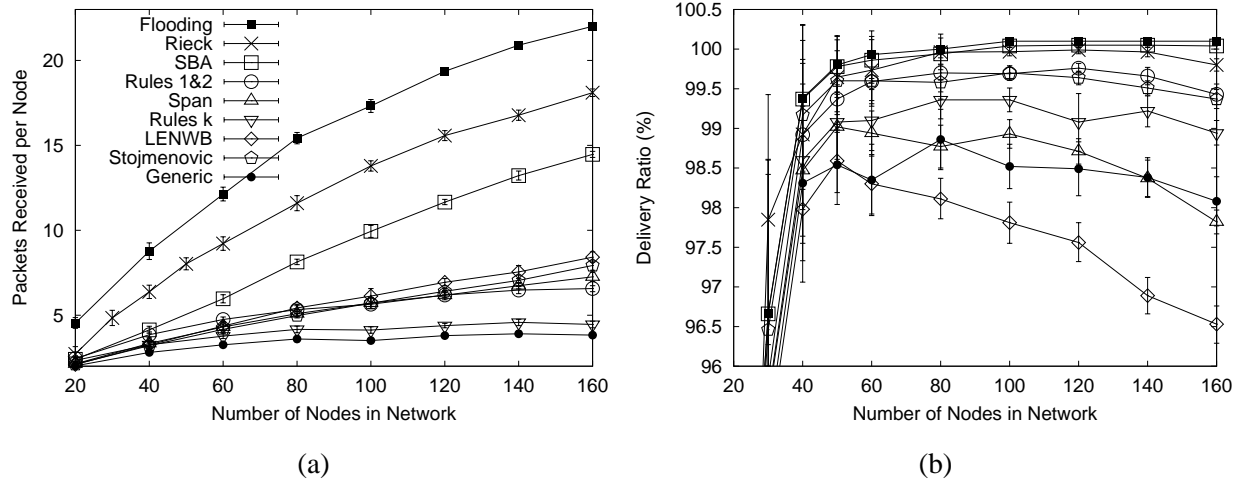


Figure 8. Broadcast redundancy (a) and delivery ratio (b) versus network size.

Efficiency: Figures 7 and 8 exhibit the performance of eight protocols in networks with relatively low traffic load (10 broadcast packets per second), low mobility (each node moving at 1 m/s on average), and varying sizes. Figure 7 (a) shows the pruning efficiency of different self-pruning protocols. Rieck’s algorithm is the most inefficient, because of its conservative pruning rule that keeps all shortest paths. SBA is inefficient in dense networks, partially caused by the relatively small backoff delay. Other protocols have similar efficiency; among them, Generic is the most efficient.

Note that the data packets sent by forward nodes represent only one part of the overall broadcast overhead; the other part is the overhead of “Hello” messages. Flooding uses no “Hello” message. In protocols using location information, the “Hello” messages are short and have a fixed size. For other protocols, each “Hello” message contains the list of neighbors of the sender, and its size is proportional to the node degree of the sender. A “Hello” message becomes smaller when node id is used as priority, as no extra bytes are used to accommodate priority values of neighbors.

We measure the overall broadcast cost of a protocol in terms of the normalized broadcast cost, i.e., average bytes sent to the MAC layer per node per broadcasting, including data packets and “Hello” messages. As shown in Figure 7 (b), Flooding has the highest cost, followed by Rieck’s algorithm and SBA, which have relatively large forward node sets. LENWB and Span, which use node degree and NCR as priorities, have higher cost than the three id-based protocols. The only location-based protocol, Stojmenovic’s algorithm, has the least overall cost. This is because Stojmenovic’s algorithm uses 1-hop location information, which has a very small constant packet size.

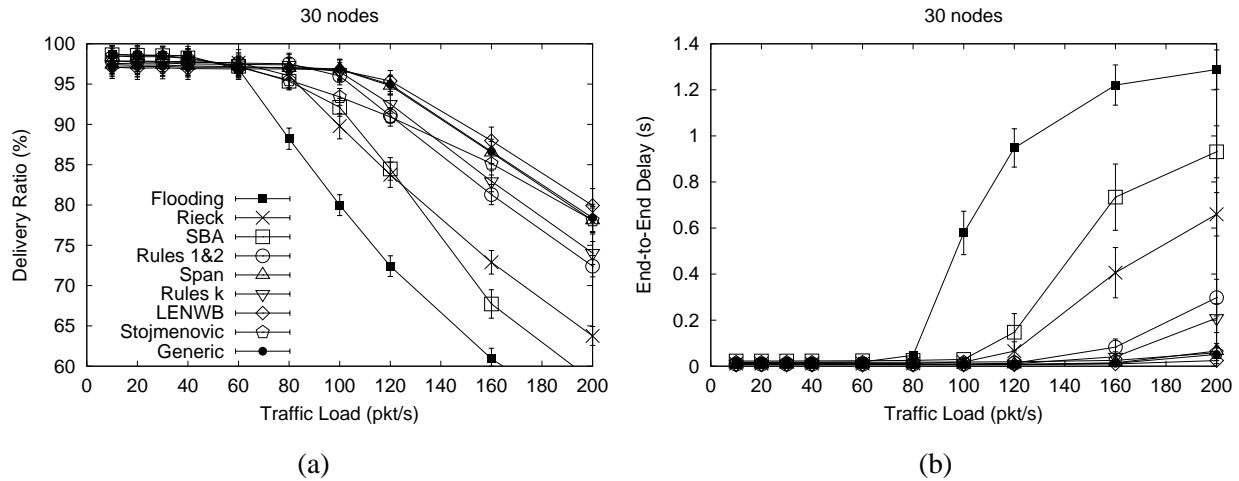


Figure 9. Delivery ratio (a) and end-to-end delay (b) versus traffic load in sparse networks.

Tolerance to collision: We measure reliability of broadcast protocols in terms of the delivery ratio, i.e., the percentage of nodes that received the broadcast packet. Low reliability may be caused by collision, contention, or mobility. Note that packet collision is different from contention. In a collision, several nodes send packets simultaneously because they cannot sense each other, and packets are lost due to interference. In a contention, a node senses a packet sent by another node and backs off, which causes extra waiting time and packets dropped from the sender’s queue. Redundancy can compensate for the effects of collision and mobility, but not for those of contention. Our focus is to identify the main threat to the reliability of a protocol under various environments.

Figure 8 (a) shows the broadcast redundancy of eight protocols. The broadcast redundancy of a protocol is defined as the average number of redundant packets received by each node. Four groups are observed in the redundancy graph: (1) Flooding, which has the highest redundancy, (2) Rieck’s algorithm and SBA, which have less redundancy than Flooding, but much higher than the others, (3) Rules 1&2, Stojmenovic’s algorithm, Span, and LENWB, which have moderate redundancy, and (4) Rule k and Generic, which have the lowest redundancy. Although high redundancy means low efficiency, moderate redundancy is critical for reliability.

Figure 8 (b) shows the delivery ratio of eight protocols. In a low traffic and low mobility environment, the dominating effect is collision. Simulation results show that collision is not a real threat to reliability under light traffic. High delivery ratio ($\geq 95\%$) is observed for all eight protocols. Both Rule k and Generic have very low redundancy, but both still have high delivery ratio ($\geq 98\%$). The only exception is in very sparse (30 nodes) networks, where network partition occurs. Five protocols with high or moderate broadcast redundancy have higher delivery ratio than other proto-

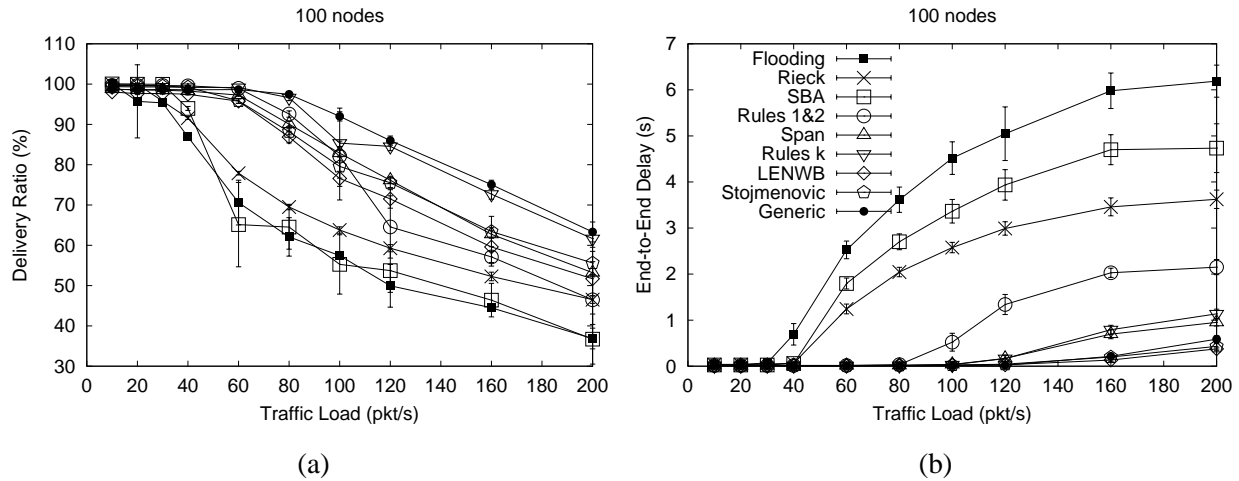


Figure 10. Delivery ratio (a) and end-to-end delay (b) versus traffic load in dense networks.

cols: Flooding, Rieck’s algorithm, SBA, Rules 1&2, and Stojmenovic’s algorithm. LENWB has lower delivery ratio than other protocols and its delivery ratio becomes lower when the network is denser. This is because LENWB uses node degree as priority and tends to select forward nodes from the center area, which increases the chance of collision in the center area, and decreases the redundancy in the border area.

Tolerance to congestion: In a congested network, low reliability is caused by either collision or contention. In the latter case, broadcast packets are dropped from the sender’s queue (a queue length of 50 at each node is used in the simulation) when the network bandwidth is exhausted by the heavy traffic. Efficient protocols like Generic and Rule k are more vulnerable to collision, while protocols with high redundancy, such as Flooding, Rieck’s algorithm, and SBA, suffer mainly from contention. When there is high contention, an increased average end-to-end delay will be observed. Figure 9 shows the scenario in relatively sparse networks (with 30 nodes and an average node degree around 6) with varying traffic load. Flooding collapses when the number of broadcast packets issued per second (pps) reaches 80, and Rieck’s algorithm and SBA collapse at 120; that is, their delivery ratios drop quickly below 90%. In the mean time, obvious increases of end-to-end delay are observed. Although SBA has lower redundancy than Rieck’s algorithm, the former has higher end-to-end delay than the latter, because SBA uses extra backoff delay. The delivery ratio of other protocols descends slowly as pps reaches 200. Since there is no big increase in their end-to-end delays, packet collision is the dominating factor. In relatively dense networks (with 100 nodes and average degree 18, as shown in Figure 10), Flooding exhausts the bandwidth when $pps = 40$, and Rieck’s algorithms and SBA do so when $pps = 60$. For the remaining protocols, Generic and

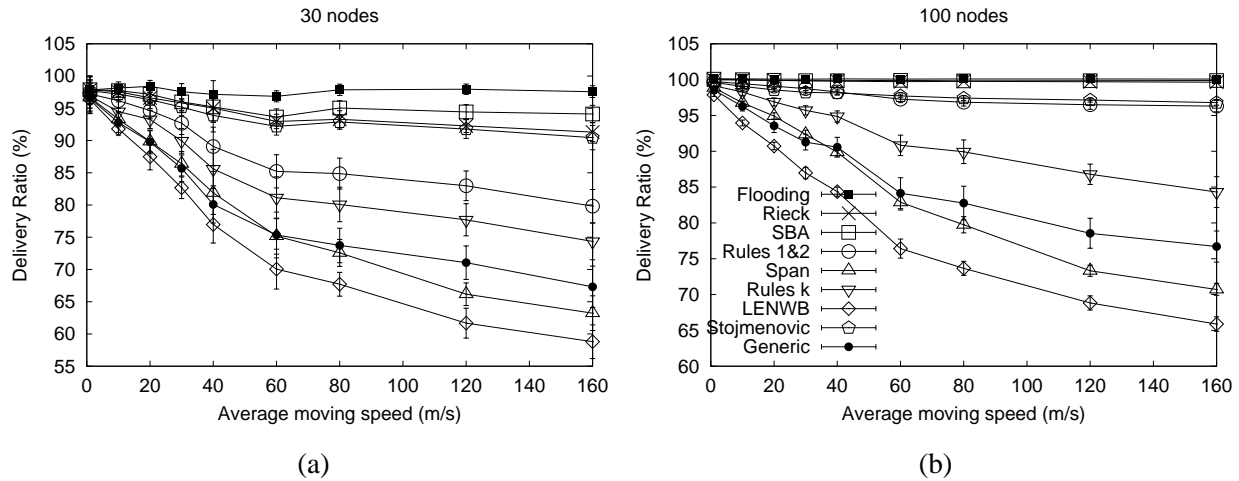


Figure 11. Delivery ratio versus network mobility in sparse (a) and dense (b) networks.

Rule k are obviously more tolerant to congestion than other special cases.

Resilience to mobility: Figure 11 shows the scenario under relatively low traffic load ($pps = 10$) and varying mobility level. The “Hello” interval is $1s$. The general rule is that protocols with the high broadcast redundancy (i.e., low pruning efficiency) have high delivery ratio. For example, Flooding always has 100% delivery ratio; Rieck’s algorithm and SBA have above 90% delivery ratio in relatively sparse networks and 100% in relatively dense networks. Rules 1&2 also have a very high delivery ratio in relatively dense networks. The delivery ratio of Generic, the protocol with the least redundancy, drops under 90% when the average node speed is above $20m/s$ in relatively sparse networks and $40m/s$ in relatively dense networks. This rule has two exceptions. First, Stojmenovic’s algorithm maintains a high delivery ratio under a high moving speed ($160m/s$). That means location-based protocols are more resilient to network mobility. Second, Span and LENWB have higher redundancy but lower delivery ratio than Generic, suggesting that node degree and NCR as priorities are less resilient to mobility than node id.

Simulation results in this section can be summarized as follows:

1. The new protocol (Generic) has higher efficiency than all existing self-pruning protocols.
2. When a small forwarding jitter ($\leq 1ms$) is used, all broadcast protocols achieve high delivery ratio ($\geq 96\%$) under a light traffic and low mobility environment. That is, collision alone cannot cause serious damage on reliability.
3. Heavy contention is observed under heavy traffic, and is the major cause of the unreliability in this case. This situation can be relieved only with high efficiency.

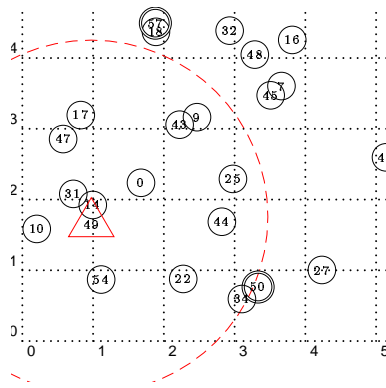
4. High mobility can seriously damage the reliability of several efficient protocols, including Rule k , Span, LENWB, and Generic.
5. When location information is provided, Stojmenovic’s algorithm is both efficient and reliable under high mobility.

In order to obtain high reliability under congestion and mobility, we must either make an efficient protocol more resilient to mobility, or make a reliable protocol more efficient. Using location information as in Stojmenovic’s algorithm seems to be an effective technique. More optimization techniques are discussed and evaluated in the next section. Note, however, that reliability is not an ultimate goal in many applications. For example, in route discovery of reactive protocols (including DSR and AODV), the route request message does not have to reach all nodes in the network. It is normally sufficient to find one node that has the knowledge of the destination. If reliable broadcast is an ultimate goal, special mechanisms have to be deployed, including ACK/NACK at link-to-link and end-to-end. Detailed discussion on this subject can be found in [9].

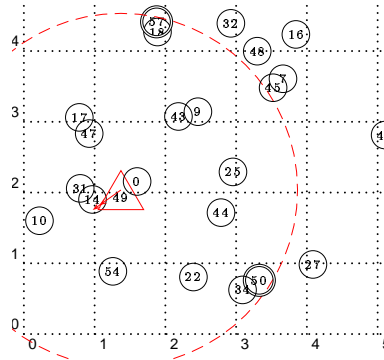
6 Optimization Techniques

The second part of our simulation study focuses on five implementation techniques that achieve resilience to mobility in self-pruning protocols while maintaining high efficiency. The first four techniques, related with “Hello” interval, priority value, backoff delay, and location information, are derived from existing protocols and have their limitations. We further propose the fifth technique, which uses two expiration timers in detecting link failures, to overcome those limitations.

“Hello” interval: In a mobile environment, nodes may be mistakenly pruned based on inaccurate neighborhood information. Massive pruning mistakes are the primary reason for the low delivery ratio in highly mobile networks. As shown in Figure 12 (a), when node 49 broadcasts a packet, none of its neighbors forward this packet. The reason is that all its neighbors have the outdated topology information, where two high priority nodes, 50 and 57, are still neighbors of node 49 and are supposed to forward the broadcast packet, as shown in Figure 12 (b). Suppose the interval between each “Hello” message is 1 second, and a link break is detected after missing two “Hello” messages, it takes at most 3 seconds to detect a topology change in a node’s 1-hop neighborhood and 4 seconds to detect one in the 2-hop neighborhood. A change of node priority, such as node degree and NCR, takes more time to detect. One way to reduce the number of pruning mistakes is to collect more accurate neighborhood information via more frequent “Hello” messages. For

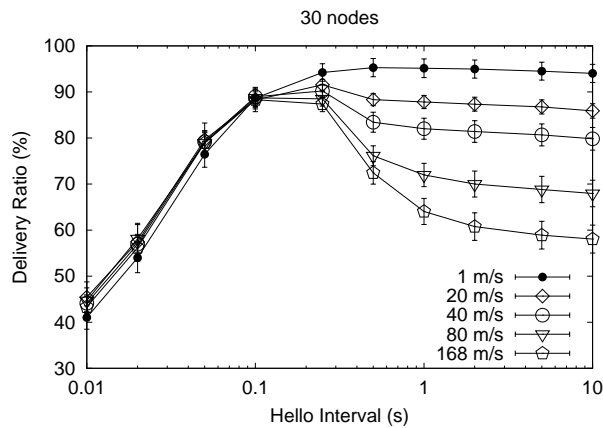


(a) A broadcast failure

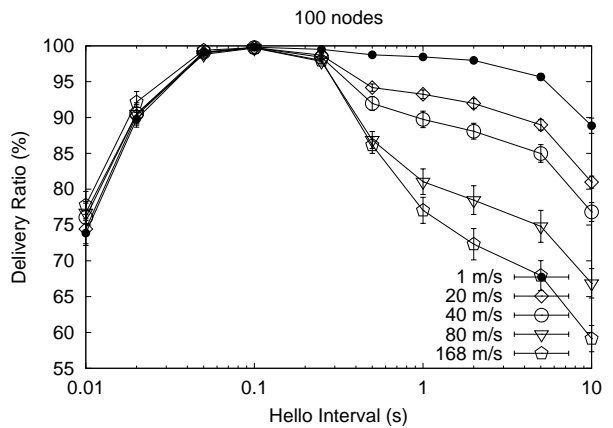


(b) The network 2 seconds ago

Figure 12. A broadcast failure due to outdated neighborhood information.



(a)



(b)

Figure 13. Delivery ratio versus “Hello” interval in sparse (a) and dense (b) networks.

example, if nodes 50 and 57 move out of the transmission range of node 49 one second ago, this change can be detected using a “Hello” interval of 0.25 second. However, this cannot solve the problem when the broadcasting starts immediately after the topology change.

Figure 13 shows the delivery ratio of the generic self-pruning rule (Generic) using different “Hello” intervals. It is not surprising that the delivery ratio is very low when the “Hello” interval is very large (10s), and becomes higher with smaller “Hello” intervals. When the “Hello” interval is around 0.1s, very high delivery ratio (90-100%) can be achieved under very high moving speed (160m/s). Using a smaller “Hello” interval, however, will not further improve the delivery ratio. In this case, packet collision rather than pruning error is the dominating factor. After the broadcast

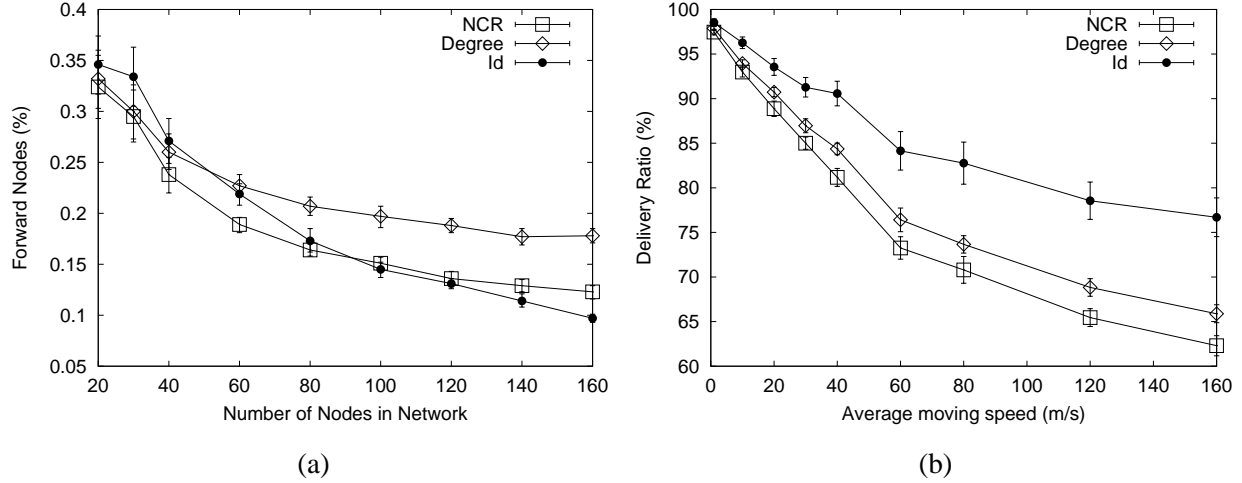


Figure 14. Pruning efficiency (a) and resilience to mobility (b) of different priority types.

packet is lost in the collision with frequent “Hello” messages, many nodes will not receive or forward the packet. As shown in Figure 13, the delivery ratio is very low with a very small “Hello” interval (0.01s).

On the other hand, the “Hello” interval cannot be reduced without limit. A self-pruning protocol is less efficient than blind flooding if it costs more than one packet per node for each broadcasting, including control packets (i.e., the “Hello” message) and data packets (i.e., the broadcast packet). Suppose 10 broadcast packets are issued per second, using a “Hello” interval of 0.1s is not a wise choice, because it uses more packets than blind flooding. Tradeoffs between efficiency and reliability must be done based on the network size, broadcast frequency, and network mobility.

Priority type: Here we consider three types of priority values: node id, node degree, and neighborhood connectivity ratio (NCR). Several previous literatures [16, 17, 21] suggested using node degree instead of node id as priority to improve the pruning efficiency, and NCR was proposed in [2] to be more efficient than node degree. This is also confirmed by simulation results in static networks in [21]. On the other hand, using node id (1-hop priority value) has faster convergence than node degree and NCR (1-hop and 2-hop priority values). In mobile networks, faster convergence means more accurate priority information, fewer pruning mistakes and higher delivery ratio. It seems that efficiency contradicts reliability in the selection of priority types.

An important discovery of this simulation study is that node id is actually more efficient than both node degree and NCR under certain situations. Figure 14 shows simulation results of several variations of Generic with different priority types. In sparse networks (40 nodes), node id as priority produces more forward nodes than both node degree and NCR. When the network becomes

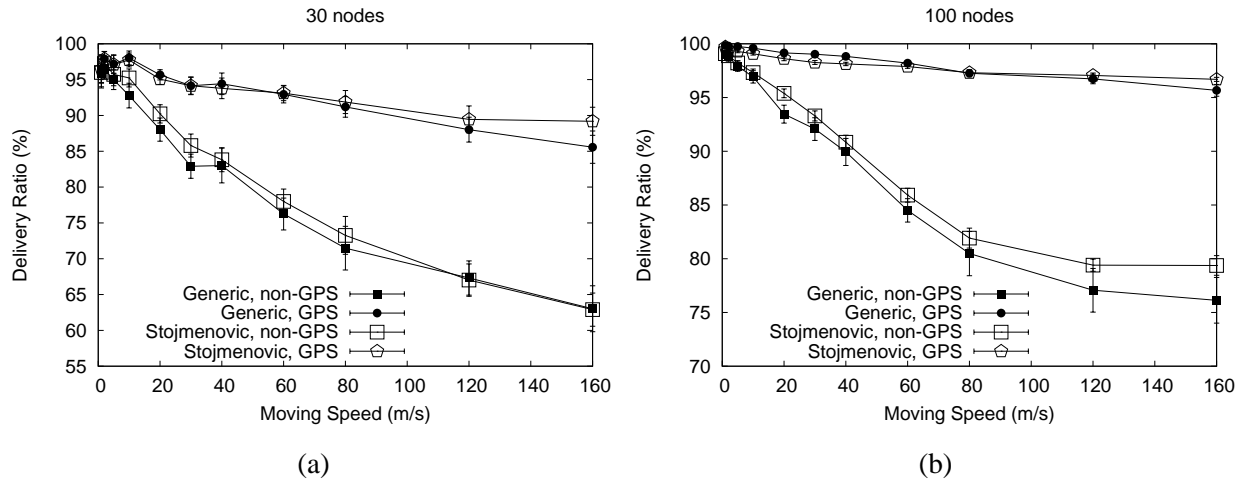


Figure 15. Delivery ratio versus average moving speed, with or without location information, in sparse (a) and dense (b) networks.

denser (60-80 nodes), node id is more efficient than node degree, but still less efficient than NCR. In networks with more than 100 nodes, node id becomes more efficient than both node degree and NCR, as shown in Figure 14 (a). When node id is used as priority, the forward nodes are evenly distributed to the entire area; when node degree is used as priority, most nodes are crowded in the center area and, therefore, cause higher redundancy. This is partially caused by the random waypoint mobility model we used in the simulation, where the center area tends to have higher node density than the border area. In very dense networks, only a few nodes with the highest priorities become forward nodes. If high priority nodes are evenly distributed, the entire network can be covered by fewer nodes with less redundancy; otherwise, more nodes will be forced to forward the broadcast packet.

Figure 14 (b) shows the delivery ratio of three variations of Generic in relatively dense (100 node) networks. Node id achieves higher delivery ratio than node degree, which in turn, has higher delivery ratio than NCR. The gaps become larger under higher mobility. Since node id is both efficient and reliable in mobile networks, it is our recommendation for all self-pruning protocols. However, using node id cannot solve the entire problem; the delivery ratio is still low (80%) under high mobility (160m/s).

Location information: Among existing self-pruning protocols, Stojmenovic’s algorithm achieves both relatively high pruning efficiency and high reliability. We believe that this high reliability is due to the use of location information. When location information is used, every node advertises its

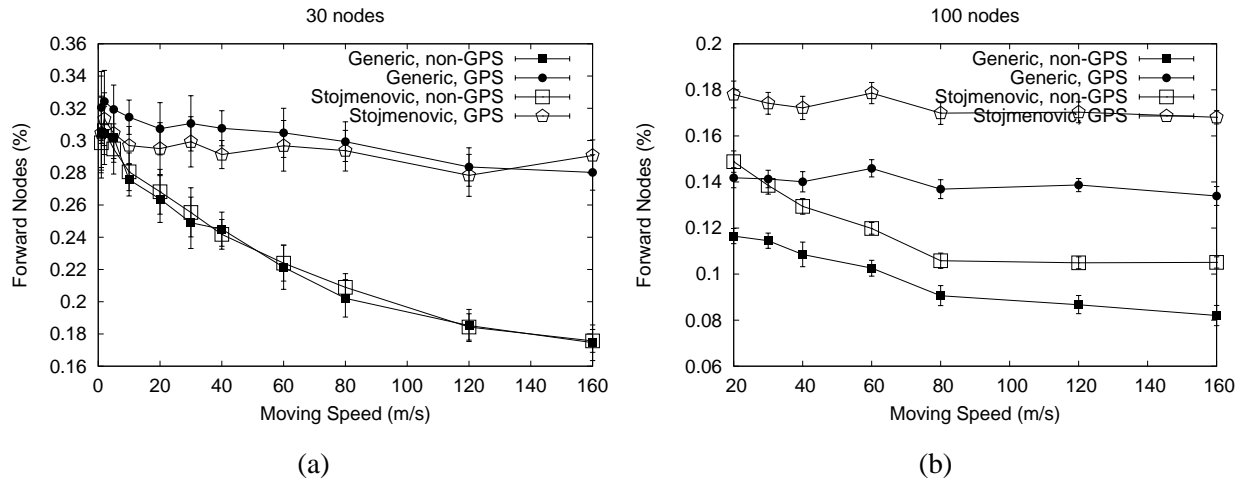


Figure 16. Percentage of forward nodes versus average moving speed, with or without location information, in sparse (a) and dense (b) networks.

location (obtained from a GPS device) in its “Hello” messages. Based on the location information collected from neighbors’ “Hello” messages, a node can compute the existence of wireless links among its neighbors based on their geographical distances and the transmission range. Using location information has two benefits: (1) the “Hello” message no longer contains the list of 1-hop neighbors and becomes smaller, and (2) wireless links computed from 1-hop location information are fresher and more accurate than those extracted from 2-hop topology information. For example, if node 43 in Figure 12 knows the recent locations of nodes 49 and 57, it can detect a broken link (49, 57) and forward the broadcast packet.

Naturally, we want to apply the technique to the generic protocol and expect the similar effect. In our simulation study, we implement two variations of Stojmenovic’s algorithm, one with the aid of location information and the other without. Similarly, we simulated two variations of Generic, with and without location information. Figure 15 shows that, when location information is available, both Generic and Stojmenovic’s algorithm achieve high delivery ratio ($\geq 85\%$ in sparse networks and $\geq 95\%$ in dense networks) under a very high average moving speed ($160m/s$). On the other hand, when location information is unavailable, the delivery ratio of both protocols drops dramatically as the moving speed increases. It is very clear that the high reliability of Stojmenovic’s algorithm comes from the use of location information, a technique that can be applied on the generic protocol and achieves the same high reliability. Other the other hand, Generic is more efficient than Stojmenovic’s algorithm. Figure 16 shows that the average number of forward nodes in Stojmenovic’s algorithm is 20% larger than in Generic in relatively dense networks. Stojmen-

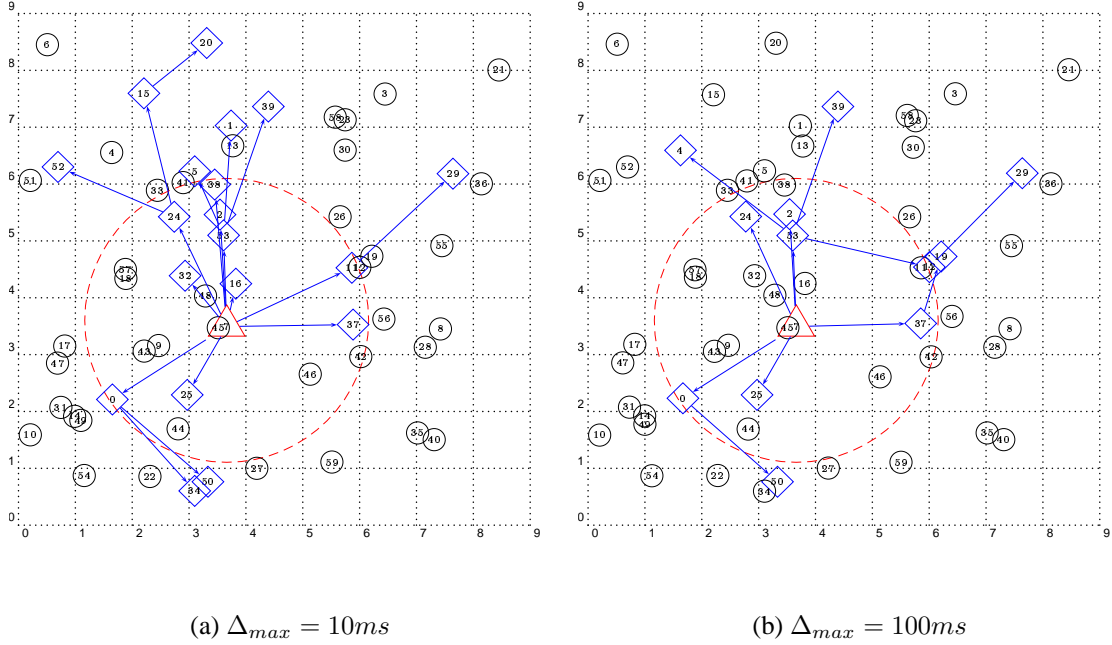


Figure 17. Broadcasting via SBA.

ovic’s algorithm is more efficient in sparse networks, because it uses node degree as priority, while Generic uses node id. Their difference, however, is very small.

On the other hand, using location information has its drawbacks. Location information providers, such as GPS devices, usually cause higher cost and energy consumption, and the obtained location information may be inaccurate. Furthermore, the actual transmission range of mobile hosts varies in different environments, and predicting the link existence among two nodes may be unreliable, even with the accurate location information.

Backoff delay: All above techniques improve the delivery ratio of an efficient self-pruning protocol. Here we try to solve the problem from another direction; that is, reduce the number of forward nodes in a reliable protocol, say, SBA. Backoff delay is used in SBA and Stojmenovic’s algorithm to discover more black nodes and further reduce the size of CDS. In the simulation, we compute the backoff delay of a node v as in SBA:

$$delay_v = \frac{1 + deg_v}{1 + deg_{max}} \Delta$$

where deg_v is the node degree of v , deg_{max} the maximum degree of v ’s neighbors, and Δ a uniform random variable between 0 and Δ_{max} . The number of forward nodes can be further reduced by using a larger Δ_{max} . Figure 17 shows two examples of SBA with different backoff delay coefficients. In the bottom of Figure 17 (a), both nodes 34 and 50 are forward nodes, because

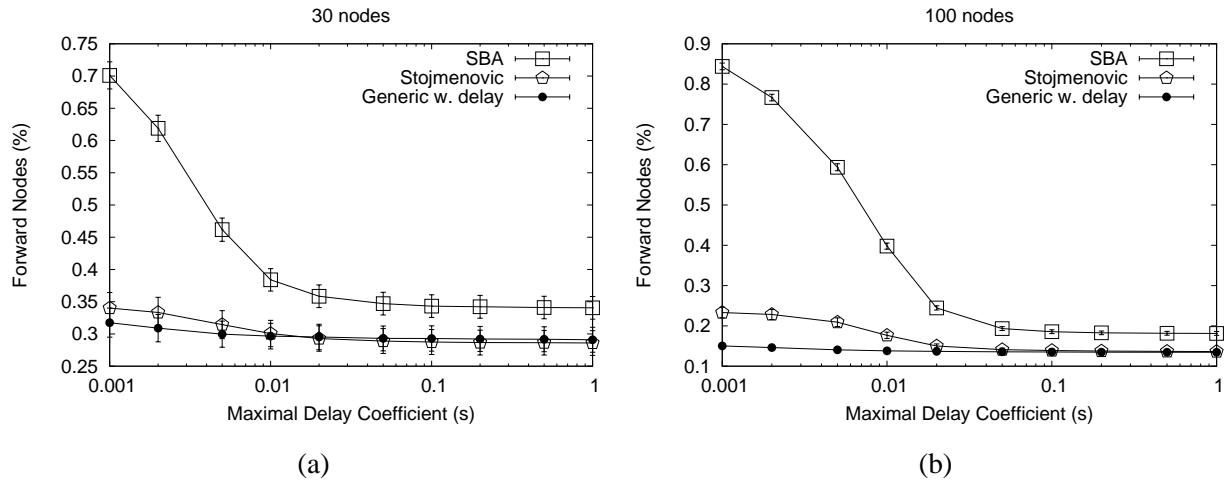


Figure 18. Percentage of forward nodes versus backoff delay in sparse (a) and dense (b) networks.

neither of them has discovered that the other node has forwarded the broadcast packet in their relatively short backoff periods ($\Delta_{max} = 10ms$). In Figure 17 (b), however, only node 50 becomes a forward node, as node 34 has waited long enough to discover the visit status of 50 ($\Delta_{max} = 100ms$).

We first examine the effect of backoff delay on efficiency of SBA, Stojmenovic’s algorithm, and a variation of Generic that uses backoff delay. Backoff delay has different effects on different protocols. As shown in Figure 18, a relatively large Δ_{max} is essential to the pruning efficiency of SBA, which performs poorly with a small backoff delay ($\Delta_{max} = 0.001s$). Increasing Δ_{max} improves the efficiency of SBA significantly. In SBA the average number of forward nodes decreases as Δ_{max} increases, until Δ_{max} reaches $0.1s$. After this point, increasing Δ_{max} will not improve efficiency significantly. Stojmenovic’s algorithm is less sensitive to the backoff delay than SBA, as Rules 1 and 2 are used to reduce the number of forward nodes before starting the backoff delay timer. Generic is the least sensitive to the backoff delay. The self-pruning rule is very efficient at the time the broadcast packet is first received, and only a few extra nodes can be pruned after the backoff delay.

The interesting thing is that the high reliability of SBA is not compromised by the higher efficiency achieved with larger backoff delay. As show in Figure 19, compared with Generic, variations of SBA with different values of Δ_{max} have almost the same high delivery ratio ($\geq 95\%$) in both sparse and dense networks, although they have different numbers of forward nodes. When

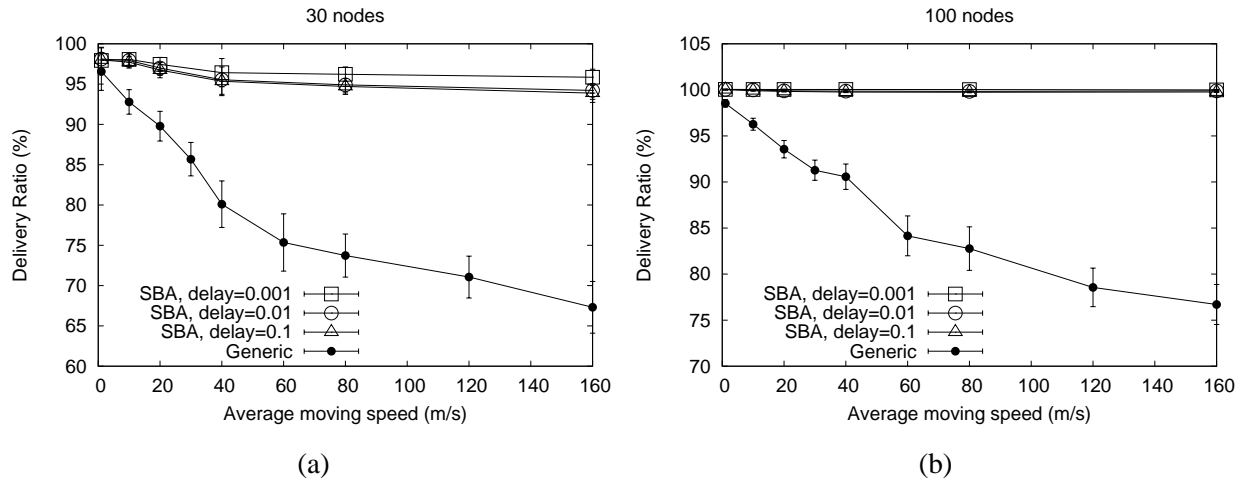


Figure 19. Delivery ratio versus mobility in sparse (a) and dense (b) networks.

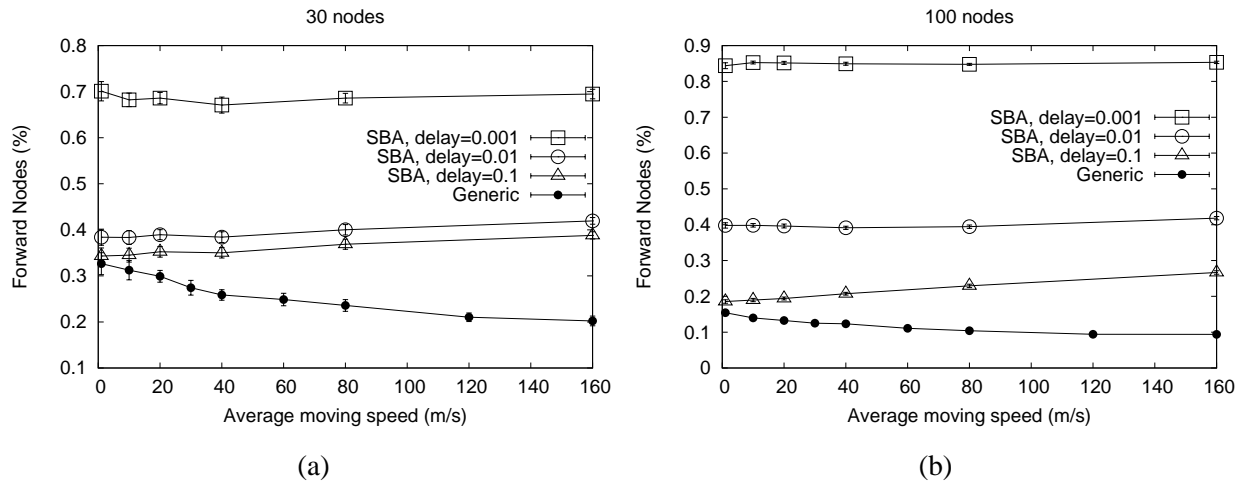


Figure 20. Percentage of forward nodes versus mobility in sparse (a) and dense (b) networks.

$\Delta_{max} = 0.1s$, SBA has almost the same number of forward nodes as in Generic, as show in Figure 20. In SBA, the number of forward nodes increases automatically to balance the increased mobility level. In the mean time, the number of forward nodes decreases in Generic, as more broadcast packets are lost under higher mobility.

Unfortunately, this technique works only for SBA, where no gray node is used in self-pruning. Even with a very large backoff delay ($\Delta_{max} = 1s$), SBA is still less efficient than Stojmenovic's algorithm and Generic with backoff delay. Another problem is the significant increase in end-to-end delay. With $\Delta_{max} = 0.1s$, the overall end-to-end delay of SBA is about $0.2s$, while the typical end-to-end delay of Generic without backoff delay is about $0.01s$.

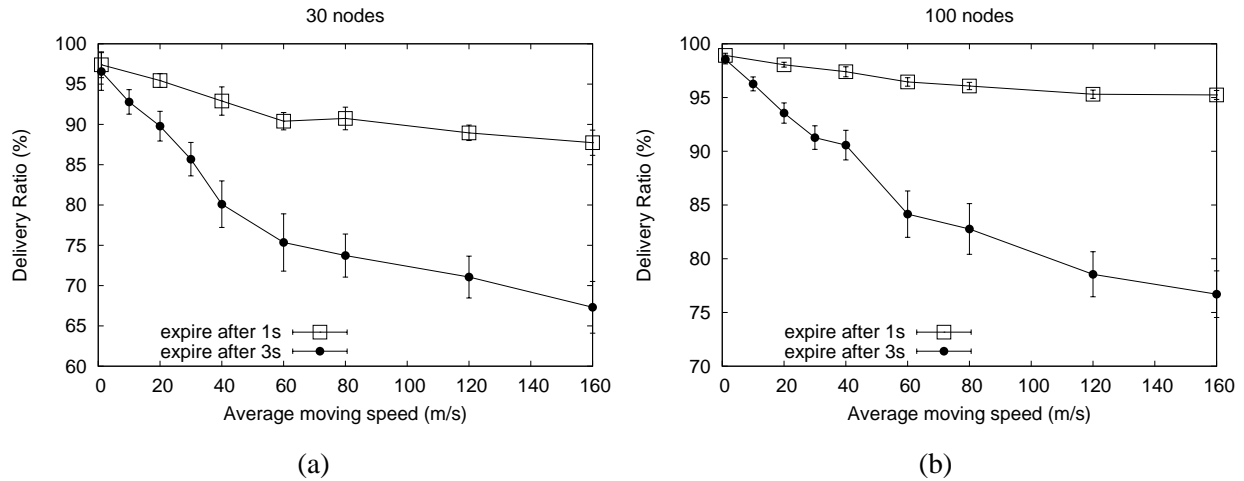


Figure 21. Delivery ratio versus mobility in sparse (a) and dense (b) networks.

Link failure detection: In our implementation of self-pruning protocols, the link failure detection mechanism is designed to tolerate two consecutive losses of “Hello” messages. That is, node v considers node u as its neighbor if v has received a “Hello” message from u within the last three “Hello” intervals. This mechanism can reduce the number of false alerts caused by collisions of “Hello” messages. However, it makes the detection of a link failure very slow. In the example of Figure 12, node 47 may become a forward node, and the broadcasting may succeed, if node 49 can remove node 57 from its neighbor set early, say, after the first missed “Hello” message from node 57, and advertise this information to node 47. However, an aggressive failure detector may cause other problems. If node v removes node u from its neighbor set by mistake, it may become a non-forward node based on the false information, leaving u uncovered.

Our solution is to use two expiration timers in link failure detection. After node v misses the first “Hello” message from node u , the failure of link (u, v) is advertised; that is, node u is removed from v ’s neighbor set embedded in the next “Hello” message. However, u is still viewed as a neighbor internally by v , until the loss of three consecutive “Hello” messages. Simulation results show that this enhancement is very effective in improving reliability. As shown in Figure 21, the enhanced Generic achieves high delivery ratio in both sparse and dense networks. According to Figure 22, the enhanced Generic has similar efficiency to the original protocol under low mobility, and the number of forward nodes increases automatically to balance higher mobility levels. Note that this scheme does not use any location information or cause extra end-to-end delay.

Simulation results in this section can be summarized as follows:

1. Using smaller “Hello” interval is more resilient to mobility. However, a very small ($< 0.1s$)

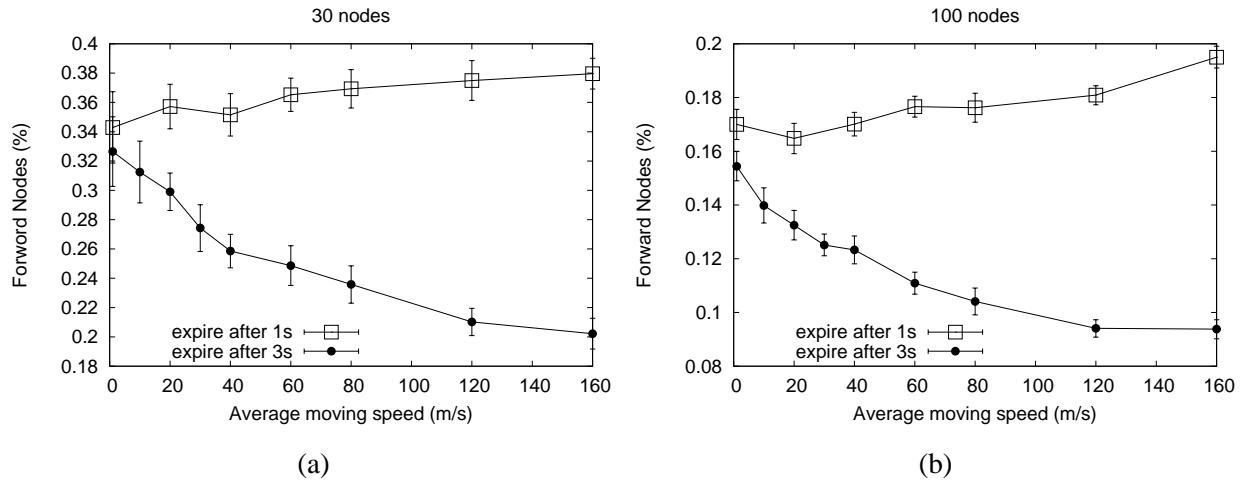


Figure 22. Percentage of forward nodes versus mobility in sparse (a) and dense (b) networks.

“Hello” interval may cause high collision, and has a negative effect on delivery ratio.

2. Using node id as priority is more efficient and more reliable than node degree and NCR.
3. SBA achieves relatively high efficiency and high reliability with a larger backoff delay.
4. When accurate location information is provided, self-pruning protocols are more resilient to mobility. However, location information may be inaccurate and causes extra cost.
5. Both high efficiency and high reliability can be achieved using the proposed fast link failure detection technique.

7 Conclusion

We have evaluated the performance of a family of self-pruning protocols via simulation study. A generic self-pruning rule is used as a framework for the comparison of existing protocols and various implementation techniques. The self-pruning rule is an enhancement of the coverage condition proposed in [21]. It provides better insight into existing protocols and has a simpler correctness proof. Seven existing self-pruning protocols and the generic protocol derived from the self-pruning rule are simulated. The simulation study focuses on the broadcast reliability under mobility and congestion. We have also evaluated several techniques to enhance reliability, and shown that both high efficiency and high reliability are feasible in mobile networks.

In a realistic network, low reliability can be caused by a combination of features such as contention, collision, and mobility. Simulation results show that collision among broadcast packets is

not a major issue here. Because a certain degree of redundancy exists even in the most efficient self-pruning protocol, and a large portion of collisions can be avoided by using a small forward jitter delay. In networks with high traffic and low mobility, contention for the shared wireless channel is a major problem, when a CSMA MAC layer such as IEEE 802.11 is used. We recommend using the generic protocol to reduce the number of forward nodes and conserve bandwidth consumption. In networks with low traffic and high mobility, the major problem is the inaccurate neighborhood information caused by mobility. We recommend using static protocols such as Wu and Li's algorithm to maintain a certain degree of redundancy.

Two existing protocols, SBA and Stojmenovic's algorithm, achieve both high efficiency and high reliability in mobile networks, and are suitable for networks with both high traffic and high mobility. However, both protocols have their limitations. In order to achieve high efficiency, a large backoff delay is required in SBA, which causes a large end-to-end delay. In Stojmenovic's algorithm, location information is used to reduce control overhead and for fast topology changes detection. However, acquiring location information introduces extra cost, and link detection based on geographic distance may be inaccurate in real networks. A technique to improve reliability in mobile networks is to use small "Hello" intervals. But this method also increases the control overhead. Using a small "Hello" interval can neutralize the effect of high mobility, but excessive "Hello" messages may be more expensive than blind flooding and are therefore not recommended. A new finding of our study is that node id as priority is better than node degree and NCR. Using node id is less efficient than node degree and NCR in sparse networks, but is more efficient in dense networks. In addition, protocols using node id has significantly higher reliability than those using node degree or NCR in mobile networks. Finally, we have proposed an enhanced link detection technique to achieve high efficiency and high reliability in mobile networks. When this technique is applied to the generic protocol, the enhanced protocol maintains the same efficiency in static networks, and is significantly more reliable than the original protocol in mobile networks, with only minor penalty in efficiency.

Our future work includes evaluating self-pruning protocols as route discovery mechanisms of on-demand routing protocols, and new efficient broadcast algorithms that are more resilient to node mobility.

Acknowledgement

We thank Brad Williams, Tracy Camp, and the Toiler group at Colorado School of Mines for sharing with us their code and experience, both of which are invaluable to our simulation study.

References

- [1] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.
- [2] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proceedings of MobiCom*, pages 85–96, July 2001.
- [3] F. Dai and J. Wu. Distributed dominant pruning in ad hoc wireless networks. In *Proceedings of ICC*, page 353, May 2003.
- [4] K. Fall and K. Varadhan. The *ns* manual. The VINT Project, UCB, LBL, USC/ISI and Xerox PARC, <http://www.isi.edu/nsnam/ns/doc/>, Apr. 2002.
- [5] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, Apr. 1998.
- [6] D. B. Johnson, J. Broch, Y.-C. Hu, J. Jetcheva, and D. A. Maltz. The CMU Monarch projects wireless and mobility extensions to ns. In *Proceedings of the Forty-Second Internet Engineering Task Force*, Aug. 1998.
- [7] H. Lim and C. Kim. Multicast tree construction and flooding in wireless ad hoc networks. In *Proceedings of MSWiM*, Aug. 2000.
- [8] W. Lou and J. Wu. On reducing broadcast redundancy in ad hoc wireless networks. *IEEE Transactions on Mobile Computing*, 1(2):111–123, Apr.-June 2002.
- [9] E. Pagnni and G. P. Rossi. Providing reliable and fault tolerant broadcast delivery in mobile ad hoc networks. *ACM/Baltzer Mobile Networks and Applications*, 4:175–192, 1999.
- [10] W. Peng and X. Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *Proceedings of MobiHoc*, pages 129–130, 2000.

- [11] W. Peng and X. Lu. AHBP: An efficient broadcast protocol for mobile ad hoc networks. *Journal of Science and Technology, Beijing, China*, 2002.
- [12] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying for flooding broadcast message in mobile wireless networks. In *Proceedings of HICSS-35*, Jan. 2002.
- [13] M. Q. Rieck, S. Pai, and S. Dhar. Distributed routing algorithms for wireless ad hoc networks using d-hop connected dominating sets. In *Proceedings of the Sixth International Conference on High Performance Computing in Asia Pacific Region*, Dec. 2002.
- [14] P. Sinha, R. Sivakumar, and V. Bharghavan. CEDAR: a core-extraction distributed ad hoc routing algorithm. In *Proceedings of IEEE INFOCOM*, pages 202–209, 1999.
- [15] P. Sinha, R. Sivakumar, and V. Bharghavan. Enhancing ad hoc routing with dynamic virtual infrastructures. In *Proceedings of IEEE INFOCOM*, volume 3, pages 1763–1772, Apr. 2001.
- [16] I. Stojmenovic, M. Seddigh, and J. Zunic. Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(1):14–25, Jan. 2002.
- [17] J. Sucec and I. Marsic. An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks. CAIP Technical Report 248, Rutgers University, Sep. 2000.
- [18] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless Networks*, 8(2/3):153–167, Mar.-May 2002.
- [19] Y.-C. Tseng, S.-Y. Ni, and E.-Y. Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. *IEEE Transactions on Computers*, 52(5):545–557, May 2003.
- [20] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of MobiHoc*, pages 194–205, June 2002.
- [21] J. Wu and F. Dai. Broadcasting in ad hoc networks based on self-pruning. *International Journal of Foundations of Computer Science*, 14(2):201–221, Apr. 2003.
- [22] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of DialM*, pages 7–14, 1999.