Towards Optimal Broadcast in Wireless Networks

Zygmunt J. Haas and Milen Nikolov
Wireless Networks Laboratory (WNL),
Cornell University
Ithaca, NY 14853
{zjh1, mvn22}@cornell.edu http://wnl.ece.cornell.edu

ABSTRACT

Broadcast is a fundamental operation in networks, especially in wireless Mobile Ad Hoc NETworks (MANET). For example, some form of broadcasting is used by all on-demand MANET routing protocols, when there is uncertainty as to the location of the destination node, or for service discovery. Being such a basic operation of the networking protocols, the importance of efficient broadcasting has long been recognized by the networking community. Numerous papers proposed increasingly more efficient implementation of broadcasting, while other studies presented bounds on broadcast performance. In this work, we present a new approach to efficient broadcast in networks with dynamic topologies, such as MANET, and we introduce a new broadcasting algorithm for such networking environments. We evaluate our algorithm, showing that its performance comes remarkably close to the corresponding theoretical performance bounds. Furthermore, we compare the performance of the proposed algorithm with other recently proposed schemes, especially in the context of various mobility settings.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design - *Network topology, Distributed networks*, *Wireless communication*; C.2.2 [**Network Protocols**]: Routing Protocols

General Terms

Algorithms, Design, Performance

Keywords

broadcast; stochastic routing; probabilistic broadcasting; efficient flooding; Minimum Connected Dominating Set; heuristics;

1. INTRODUCTION

Broadcast is a fundamental network operation allowing a source node to send a message to all other nodes in the network. In the context of *Mobile Ad Hoc Networks* (*MANET*), where topology can change rapidly, where all communications are carried over a wireless medium, and where the network nodes are limited in energy and computational power, an efficient broadcast mechanism is exceptionally important for the overall improvement of the network performance.

For instance, among the various proposed MANET routing protocols (e.g., AODV, DSR, OLSR, TRBF, ZRP), a prominent sub-group, referred to as *on-demand* or *reactive* routing protocols, is designed based on the philosophy that the discovery of a route in the network should be done only when there is an actual need to route traffic. The route discovery mechanism in on-demand routing protocols relies on some variant of broadcasting to locate a path between a source and a destination nodes. As another example, in highly reconfigurable topologies, where the lifetime of the routes may be shorter than the duration of a communication (especially connection-oriented communication) broadcast, by itself, could be used as a routing mechanism. Yet in other scenarios, when data dissemination to all the nodes in a MANET is needed, broadcast is an obvious solution.

The required features of a broadcast algorithm in the context of wireless reconfigurable networks are that:

(1) it reaches *all* the network nodes;

(2) it transmits the message as few times as possible (or, equivalently, reduces the number of times that the broadcast message is received by a network node, optimally to only once);

(3) each node requires only locally restricted information (e.g., only the knowledge of the *k*-hop topology for small value of *k*);

(4) it proceeds so that changes in topology during the broadcast propagation do not significantly affect (1) and (2) above.

Being such an essential network operation, it is not surprising that the importance of an efficient broadcast implementation has been widely accepted by the networking community. Unfortunately, few of the so-far-proposed approaches to the broadcast problem satisfy all the above requirements. Flooding-based protocols violate (2) above, as they lead to the notorious "broadcast storm" problem [1]. Probabilistic schemes, often based on percolation [3, 4, 5, 6, 7, 8, 9] do not satisfy (1) above.[1] Efficient backbone-based algorithms have also been proposed. They rely on finding minimum connected dominating sets (MCDS), which are constructed by identifying dominating sets, maximal independent sets, or Steiner trees prior to broadcast. Such algorithms do not tolerate dynamic topologies well [19, 20, 21, 22], thus are incompliant with (4). Furthermore, some of these algorithms are centralized [29, 30], thus violating (3). One recent example of a broadcast algorithm that satisfies (1) – (4) has been described in [25], though the algorithm requires the knowledge of the nodes' geographical position. (Section 7 provides more details and references to related work in the literature.)

[1] To ensure full coverage, a stochastic scheme would need to transmit with probability close to 1, in which case the scheme converges to flooding.

In this paper we describe, model, and simulate a novel approach to network broadcasting. The resulting algorithm satisfies the features (1) – (4) without relying on nodes' positional information. Furthermore, the algorithm is comparatively simple to implement, relying on the intuitive greedy heuristic to tackle the, otherwise, NP-hard problem [32] of finding the Minimum Connected Dominating Set (MCDS) in a Unit Disk Graph. The algorithm experimentally outperforms other schemes recently published in the literature, achieving close to the optimal performance of the broadcast operation for key metrics such as the number of transmissions. Significantly, the algorithm performs well even in highly dynamic scenarios, in the context of realistic mobility models such as the Gaussian Markov Mobility Model (GMMM) and the Reference Point Group Mobility (RPGM) model.

2. SYSTEM MODEL

The network model consists of N equal-capability nodes with unique IDs, randomly distributed in 2D plane.[2] The transmission range of all nodes is R [meters]. Two nodes are referred to as *1-hop neighbors* (or simply as *neighbors*) and can communicate directly if the distance between them is less than R. Thus, the network can be modeled as a Unit Disk Graph (UDG).

On the MAC layer, links are bidirectional and nodes share a single wireless channel. We assume a *perfect* MAC layer to eliminate other effects (e.g., congestion), so that the broadcast performance metrics reflect only the algorithmic efficiency. This also allows a meaningful comparison of various broadcast schemes, as different schemes could be differently affected by a particular MAC layer protocol and the choice of its parameters.

The system operation is time-slotted, and the network nodes are assumed to be coarse-grain synchronized. The latter is a standard assumption of many distributed algorithms and can be implemented in variety of ways. For instance, distributed, control-message-based methods for coarse-grain synchronization in multi-hop wireless networks would suffice; such schemes have been studied extensively in the literature (e.g., [28]). Recent advances in radio technologies could also be utilized [27]. Finally, all nodes are cooperative and trust each other.

3. SOLUTION DEMYSTIFIED

As hinted above, the problem of finding the most efficient broadcast scheme is to a large extent equivalent to finding an approximation of the MCDS in a Unit Disk Graph which satisfies (1) – (4). The *Time-Sequence* (TS) scheme and its variants, as proposed in this paper, aim at this goal through the use of a *greedy* heuristic. We start by providing a number of definitions:

Definition 1: A *broadcast session* is the operation (including all related events) of delivering a message, created at one node – the *source* – to all the other network nodes.

Definition 2: A *covered node* is a node that has received the broadcast message.

The source node of a broadcast session is always covered. A node that has not received the broadcast message at a particular time is referred to as an *uncovered* node at that time.

Definition 3: The *residual coverage* (*RC*) of a covered node s at a particular time, referred to as $RC(s)$, equals the number of its 1-hop uncovered neighbors at that time.

Definition 4: We define S as the set of all the network nodes, C as the set of all covered nodes at a particular time, and Q as the set of nodes that have already transmitted the message at a particular time. We further define $NE(s)$ as the set of all the neighbors of the node s, where $s \in S$. We note that at any time, $Q \subseteq C \subseteq S$ and that $|S|=N$.

We represent the network as a connected graph $G=(S, E)$, where E is the set of all the links that connect any two nodes in S. Also, we label the broadcast source node as s_0.

Consider first the following centralized greedy scheme, starting with the initial set of covered node $C=\{s_0\}$ and $Q=\varnothing$. The source node transmits first, covering its neighbors: $C = \{s_0 \cup NE(s_0)\}$, $Q=\{s_0\}$. An "oracle" greedily chooses a node, s_1, from the set $C\text{-}Q$ with the largest RC value to broadcast next; i.e., $\forall s \in (C\text{-}Q)$, $RC(s) \leq RC(s_1)$. After s_1 broadcasts, $C \leftarrow C \cup NE(s_1)$ and $Q \leftarrow Q \cup \{s_1\}$. This operation of selecting the next node to transmit as the node with the largest RC from the set $C\text{-}Q$ is repeated until all the network nodes are covered; i.e., until $C=S$, at which time the algorithm terminates. (The total number of transmissions in a broadcast session equals $|Q|$ at the algorithm termination time.) Furthermore, the choice of node s_i to transmit in the i^{th} iteration allows maximizing the number of covered nodes during the i^{th} transmission. This, in turn, would intuitively <u>tend</u> to minimize the total number of transmissions during the operation of the algorithm. Of course, the algorithm does not guarantee such a minimum, as in some cases choosing a node with smaller RC value first could, in fact, result in finding nodes with much larger RC value later, so that the overall number of transmissions is smaller. In [31], the authors presented in more details the failure of a similar greedy scheme in finding MCDS in general graphs.

Notwithstanding the above, restricting the network topology to a Unit Disk Graph relaxes the problem and the above centralized greedy heuristic finds on the average a rather close approximation of an MCDS,[3] as is demonstrated in Section 5. Of course, the challenge, similarly to other efficient MCDS approximation schemes, is to implement the "oracle" in a distributed manner; i.e., ordering nodes' transmissions based on their RC values, while utilizing only local topological information.

The following TS scheme distributively approximates the transmissions' order by allowing nodes with larger RC values to transmit before transmissions of nodes with smaller RC values. The timing of transmissions is enforced by a specifically structured sequence of time-slots, which are locally available to each node, and by a scheduling procedure, which is locally executed at each node. The scheduling procedure is based on associating a particular value of RC threshold with each time-slot and on restricting transmissions in a time-slot only to nodes with RC values not smaller than the time-slot's RC threshold.

Note that the goal of the TS scheme is not to control access to the wireless channel, akin the TDMA scheme, for instance. Namely, a time-slot in the TS scheme may accommodate concurrent transmissions of multiple nodes in the network. Section 4.3 discusses how, via the structure of the time-slot sequences in the TS scheme and the operation of the proposed algorithms, potential

[2] However, the results in this paper apply to 1D and to 3D networks, as well.

[3] This is often the case with the greedy algorithm in various other applications as well.

collisions and interference on the MAC and the physical layers can be avoided during the transmissions of the broadcast message.

4. THE TIME-SEQUENCE SCHEME

We start by defining a *Time Sequence,* and by presenting its structure and properties.

4.1 The Time-Sequence Structure

Let T be an ordered collection of vectors $\{T_x, T_{x-1}, \dots, T_1\}$, where $T_i = (u_i, m_i, l_i)$ and where $u_i, m_i,$ and $l_i \in \mathbb{N}^+$ for all i. The parameters u, m, and l are referred to as *upper*, *middle,* and *lower* values. Let T be the output of **Algorithm 1** given input u.

Proposition 1: $|T| = \dfrac{u(u+1)}{2}$.

Proof: The output, T, of **Algorithm 1** can be arranged in the following isosceles triangle with sides u: the triangle consists of u levels, where the last level (level 1) comprises u vectors:

(u, u, u) --{level u: $l = u$}
(u, u, u-1) **(u, u-1, u-1)** -----------------------{level $u\text{-}1$: $l = u\text{-}1$}
⋮ ⋮
⋮ ⋮
(u, u, 1) (u, u-1, 1) … **(u, 1, 1)** --------------{level 1: l=1}

The number of vectors at the i^{th} level equals $1 + u - i$. The total number of vectors is, then: $\sum_{i=1}^{u} u(u+1)/2$. □

The vectors in bold are called *edge slots* and form the "base" of the isosceles triangle.

Next, let $A = (S, I)$ be a set system with $S = \{s_1^{p_1}, s_2^{p_2}, \dots, s_n^{p_n}\}$, $p_i \in \mathbb{N}^+$ for all $i \in \{1, \dots, x\}$, and let I be the collection of subsets of S, such that $I = \{S_1, S_2, \dots, S_x\}$ and $S_i = \{s_j^{p_j} : p_j \geq m_j \geq l_j \geq 1\}$ given vector $T_j = (u_j, m_j, l_j)$ in T (note that $|I| = |T|$). Define a binary order relation on S, so that $s_j^{p_j} \leq s_i^{p_i} \Leftrightarrow p_j \leq p_i$.

Definition 5: An element $s_j^{p_j}$ is called *admissible in T_i*, iff $s_j^{p_j} \in S_i$.

A minimal admissible element in T_k, denoted by *min* (T_k), is $s_j^{p_j}$ such that $s_j^{p_j} \leq s_i^{p_i}$ for all $s_i^{p_i}$ admissible in T_k. Consider the ordered collection of vectors T^{λ} at the level λ of T ($\lambda \in \{1, \dots, u\}$), and the smallest element of all minimal admissible elements at this level, denoted *inf* (T^{λ}). Now, let L be the sequence (*inf*(T^{l}), *inf*(T^{l-1}),…, *inf*(T^{1})).

Proposition 2: The sequence L is decreasing.

Proof outline: The result follows trivially from the definition of I, the ordering of T, the order relation defined on S, and the definition of *inf*(T^{λ}).□

Let M be the sequence (*min* (T_k), *min* ($T_{k\text{-}1}$),…,*min* (T_1)), where T_i*'s*, $i \in \{1, \dots, k\}$, are in T^{λ} for some λ.

Proposition 3: The sequence M is decreasing.

Proof outline: Similarly to **Proposition 2**, the proof follows trivially from the definition of I, the ordering of T, the order relation defined on S, and the definition of *min* (T_k). □

Algorithm 1: Constructing the vector set T
Input: u

```
/* initialization */
upper := u
middle: = u
lower := u
T := ∅
T_1 := ( upper, middle, lower )
T := T_1
/* Computing the T vectors */
while middle > 1:   // Generating the next vector
    if middle == lower :
        lower := lower – 1
        mid := upper
        next_vector: = ( upper, middle, lower )
    else
    if middle > lower :  // lower remains the same
        middle := middle – 1
        next_vector := ( upper, middle, lower )
    T := T+next_vector
```

We can now define the structure of the Time Sequence (*TS*). Suppose time is slotted. Every x consecutive time-slots (naturally ordered in time) are mapped one-to-one to the vectors in the ordered collection T above. That is, each time-slot, t_j is uniquely associated with a vector in T as follows: $t_1 \leftrightarrow T_x = (u, u, u)$, $t_2 \leftrightarrow T_{x\text{-}1} = (u, u, u\text{-}1)$, …, $t_x \leftrightarrow T_1 = (u,1,1)$, where $x = 0.5u(u+1)$ from **Proposition 1.** A network node is admissible in t_j if it is admissible in $T_{x+1\text{-}j}$ This association "wraps around"; i.e., in general for $j \geq 1$, $t_j \leftrightarrow T_{kx+1\text{-}j}$ where $k = \lceil j/x \rceil$. The time sequence (*TS*) is the ordered collection of the time-slots, together with their corresponding vectors.

4.2 Distributing the Time-Sequence Scheme

As defined above, let S be the set of all the network nodes. Each node j is assigned its value $p_j = RC(j)$. More specifically, $S = \{n_1^{RC_1}, n_2^{RC_2}, \dots, n_N^{RC_N}\}$, where $|S| = N$. Note that as time goes by from a time-slot to the next, the values $RC(j)$ may change.

(Revisited) Definition 1: A *broadcast session* consists of all the events, starting from the transmission of the message by the source node and ending when the broadcast algorithm terminates.

Also, for now, we will assume that the broadcast session finishes in less than or equal to |T| time-slots. Then, we define the following rule:

Broadcast Rule: At every time-slot, t_i, a node considers broadcasting only if it has not broadcasted earlier in this session and it is admissible in T_i at the beginning of the time-slot t_i

Thus, **Proposition 2** implies that nodes with smaller *RC* values would not be able to broadcast (i.e., be admissible) at the beginning of the time sequence *TS*, but will potentially be admissible in later timeslots (i.e., at lower levels). And reversely, only nodes with larger *RC* values would be admissible and be able to broadcast in earlier timeslots (i.e., at higher levels) of the *TS*. Furthermore, **Proposition 3** implies that within given level of the *TS*, nodes with smaller *RC* values would again be admissible only in later timeslots at this level, and nodes with larger *RC* values would be admissible earlier in the given level.

For illustration purpose, consider the case of u = 4:

(4,4,4) ---{TS level 4: l=4}
(4,4,3) **(4,3,3)** -------------------------------------{TS level 3: l=3}
(4,4,2) (4,3,2) **(4,2,2)** -----------------------------{TS level 2: l=2}
(4,4,1) (4,3,1) (4,2,1) **(4,1,1)** ---------------------{TS level 1: l=1}

Example 1. The output of **Algorithm 1** for $u = 4$

In this example, in the top *TS* level (level 4) only nodes with RC $\geq$ 4 are allowed to transmit. In the next *TS* level (level 3), first nodes with $RC \geq 4$ and then nodes with $RC \geq 3$ will be allowed to transmit. In level 2, first nodes with $RC \geq 4$, then nodes with $RC \geq 3$, and finally nodes with $RC \geq 2$ will transmit. In the last level, first nodes with $RC \geq 4$, then nodes with $RC \geq 3$, then nodes with $RC \geq 2$, and finally nodes with $RC \geq 1$ (all nodes with at least one uncovered neighbor) will be allowed to transmit.

In summary, the structure of the *TS* in conjunction with the **Broadcast Rule** has admissibility property allowing for repetitive assignment of larger transmission priority to nodes with larger *RC* values as compared to nodes with smaller *RC* values. Hence, by the virtue of the admissibility property of the **Broadcast Rule**, the greedy "oracle" scheme is emulated.

4.2.1 Distributing the TS-based Broadcast

The above time-sequence structure is computed locally by every node in the network, so that covered nodes can determine, based on their current value of *RC*, the earliest time-slot (i.e., the phase of the time sequence) at which they should first attempt to transmit the message, thus satisfying the **Broadcast Rule.** Next, we describe the operation of the scheme in more details.

4.2.1.1 Network Deployment

Upon network deployment, **Algorithm 1** is run locally by each node.[4] The input to the algorithm, the parameter u, is fixed and set up administratively and network-wide at the time of deployment. The value of u should be judiciously chosen; a too small value of u does not allow to separate in time the transmissions of nodes with different values of *RC,* thus losing the ability to assign larger priority to nodes with larger *RC* values, while a too large value of u results in many empty time-slots, thus leading to an unnecessarily long broadcast session. Section 5.1.2 addresses the choice of the u parameter in more details, discussing its effect on the scheme's performance.

Preamble	*Broadcast Field*

Figure 1. The format of a time-slot.

Time-slots have the format shown on Fig. 1. The *Broadcast Field* is fixed to be the maximum duration needed to transmit the broadcast message. The *Preamble* is used to transmit control information between adjacent nodes, and its duration is negligible compared to the total length of the time-slot. The timestamp of the initial broadcast by the source is embedded within the *Preamble,* is propagated by the broadcasting nodes, and serves by the nodes to determine the current phase of the *TS.*

4.2.1.2 Node Scheduling

The source node broadcasts the message at the beginning of the first *TS*. As the broadcast propagates throughout the network, any node upon receiving the broadcast message for the first time determines the current *TS* level and the current time-slot within the *TS*. This is achieved by subtracting the initial broadcast timestamp (in the message's *Preamble*) from the node's current local time, and by dividing the difference by the duration of a time-slot. Knowing the generic *TS* structure and the number of time-slots that passed since the beginning of the *TS,* a node is able to determine the m and l values of the current time-slot.

Said differently, as the broadcast message propagates through the network, it prompts the receiving nodes to calculate the *TS* structure and to identify the current phase within the *TS* structure.

After a receiving node calculated the current time-slot within the *TS,* the node runs **Algorithm 2** to schedule its transmission for a future time-slot. It is easy to verify that **Algorithm 2** complies with the **Broadcast Rule**. For instance, if m of the current *TS* time-slot is lower than the node's *RC* value, **Algorithm 2** schedules the node's transmission for the next immediate time-slot. Otherwise, a further future time-slot is assigned to the node. A node is never scheduled to transmit if its $RC = 0$ (i.e., it is not admissible in any time-slot).

```
Algorithm 2: Node i self-scheduling
Input: $RC_i$;  ($u_{ct}$, $m_{ct}$, $l_{ct}$) // vector associated with the current
                                                          time-slot
/* initialization */
rc := $RC_i$
upper := $u_{ct}$
middle :=  $m_{ct}$
lower :=  $l_{ct}$
/*if $RC_i$ is larger than the current value of middle, the node
broadcasts in the next time-slot */
if rc > middle :
   broadcast_timeslot := next_timeslot
else
/* if $RC_i$ is not large enough for the current priority, the node's
future broadcast time-slot is determined by the value of $RC_i$ */
if rc ≤ middle AND rc ≥ lower :
  if ($u_{ct}$, $m_{ct}$, $l_{ct}$) is edge slot :
    if lower > 1 :
       broadcast_timeslot := ( upper,  rc,  lower-1 )
    else
       broadcast_timeslot := ( upper,  rc , 1 )
  else
     broadcast_timeslot := ( upper,  rc,  lower )
else
/* if $RC_i$ is even less than the value of lower, the node is
pushed to broadcast in a later level of the TS triangle; the
level depends on $RC_i$ */
if rc < lower AND rc ≥ 1 :
   broadcast_timeslot :=  ( upper,  rc, rc )
```

The value of a node's *RC* can change[5] between the time at which it had scheduled itself for transmission and the beginning of its scheduled-for-transmission time-slot, thus possibly rendering the node inadmissible in its scheduled time-slot. To avoid transmission in an incorrect time-slot, a node re-computes its *RC* value during the *Preamble* time of its scheduled-for-transmission timeslot and checks if it is still admissible in this time-slot. If so, it transmits the message during the *Broadcast Field* time. Else, it reschedules itself by employing the **Algorithm 2** again.

4.2.1.3 Residual Coverage Computation

The *RC* value of a node is needed prior to the node scheduling or rescheduling itself for transmission. This is done by a locally

[4] Practically, **Algorithm 1** should be run periodically.

[5] Due to transmissions of other nodes or due to mobility.

executed protocol – a version of a "Neighbor Discovery" protocol. Specifically, to compute its *RC* value, node *i* broadcasts a *Coverage Request* packet, *CReq*, to all its 1-hop neighbors. The *CReq* packet contains *i*'s ID. Upon receiving the *CReq* packet, each one of *i*'s uncovered neighbors replies with a *Coverage Reply* packet, *CRep*, which contains the neighbor's ID and *i*'s ID. Node *i* then counts the number of such replies (identified by its own ID in the *CRep* packets) during a time approximately equal to the duration of the *Preamble*. This simple protocol can be further improved in a variety of ways, but that is beyond the scope of this paper.

4.3 Variants of TS-based Broadcast Schemes

For clarity, let's summarize the basic *TS*-based broadcasting scheme. All nodes run **Algorithm 1** to construct the *TS*. When a node (re)broadcasts the message, all of its previously uncovered neighbors mark themselves as covered, compute their *RC*, and run **Algorithm 2** to schedule their broadcast time-slots. Just before its scheduled time to transmit (during the *Preamble* of the scheduled-for-transmission time-slot), a node re-computes and updates its *RC*. If the node's current value of *RC* has decreased (but *RC* > 0), the node determines its new time-slot assignment by re-running **Algorithm 2.** If, at any time, the *RC* value of a node equals 0, the node will never be scheduled for broadcast in this session.

We refer to this basic scheme as the *Naïve Time Sequence Scheme* (*NTSS*)[6]. In what follows, we present two variants of the NTSS, each utilizing varying degree of the neighborhood topology knowledge.

4.3.1 1-Hop Time Sequence Scheme (TSS-1Hop)

Suppose node *i* is about to transmit[7] in the current time-slot t_j. Then node *i* checks during the *Preamble* of the time-slot t_j whether any of its 1-hop neighbors are scheduled to broadcast within t_j as well. This check does not necessitate any additional transmissions, as the node can determine whether a particular neighbor is scheduled to transmit in t_j, if it receives the *CReq* message from that neighbor during the *Preamble* time. If more than one neighboring node is scheduled for t_j, the node with the largest *RC* is selected to transmit in t_i. The rest of the neighboring nodes reschedule themselves to transmit in the next time-slot.

To accommodate the TSS-1Hop operation, the *CReq* and the *CRep* control packets should also include the respective sender's *RC* value, in addition to the nodes' IDs. We omit here further details of this simple protocol.

4.3.2 2-Hop Time Sequence Scheme (TSS-2Hop)

In the TSS-2Hop scheme, a node *i* which is scheduled to transmit in the current time-slot t_j checks if there are either 1-hop or 2-hop neighbors scheduled to transmit within t_j as well. Among the set of all the 1-hop and 2-hop neighbors, the one with the largest *RC* is selected to transmit; the rest are rescheduled to the next time-slot.

Here, the control packets passed in the *Preamble* need further augmentation. The *CReq* packet contains now two IDs. Node *i* broadcasts a *CReq* packet to all its 1-hop neighbors, with the first ID set to *i* and the second ID to *null.* Each (1-hop) neighbor *n*, upon receipt of such a *CReq* packet, retransmits it to its own neighbors, with the first ID unchanged and the second ID set to *n.* Each (1-hop) neighbor of *n* (and, therefore, 2-hop neighbor of the node *i*), which is scheduled to transmit in this time-slot, replies to *n* with its *CRep* packet with its *RC* value and with the IDs unchanged. The node *n* then collects all such replies, determines the largest *RC* value from among the received *RC* values and its own *RC* value, and send its *CRep* with this largest value of *RC* to the node *i.* We omit here further details of this simple protocol.

4.4 Sample Execution of NTSS

Scheduled Nodes	*RC*/Scheduled Timeslot
a	**2** / (4,2,2)
b	**3** / (4,3,3)
c	**1** / (4,1,1)
d	**1** / (4,1,1)

Step 1

Scheduled Nodes	rc/Scheduled Timeslot
a	**2** / (4,2,2)
c	**0** / (4,1,1)
d	**1** / (4,1,1)
f	**1** / (4,1,1)
g	**0** / -
h	**0** / -

Step 2

Scheduled Nodes	rc/Scheduled Timeslot
c	**0** / (4,1,1)
d	**1** / (4,1,1)
f	**0** / (4,1,1)
g	**0** / -
h	**0** / -
e	**0** / -
k	**2** / (4,2,1)

Step 3

Step 4: All nodes are covered after timeslot (4,2,1). The time-sequence is exhausted at (4,1,1), where as well no nodes have RC > 0, and the algorithm terminates.

Figure 2. In the first timeslot, the source node *S* broadcast the message. Nodes *a*, *b*, *c*, and *d* receive it, mark themselves as covered, compute their RC, and schedule themselves to broadcast At **Step 1**, according to **Algorithm 2,** since node *b* has 3 uncovered neighbors it is scheduled for timeslot (4,3,3). Node *a* is scheduled similarly for (4,2,2), *c,* and *d* for (4,1,1). At **Step 2**, during the third timeslot (4,3,3) node *b* rebroadcasts the message. Nodes *g*, *h,* and *f* become newly covered and are added to the scheduled nodes list: *f* is scheduled for (4,1,1); *g* and *h* have *RC* = 0 and are not scheduled. Node *b* is removed from the list. At **Step 3**, in the sixth timeslot (4,2,2), node *a* check its *RC*. Since its *RC* remains the same, node *a* rebroadcasts the message. Nodes *e* and *k* become newly covered. Node k has two uncovered neighbors: *i* and *j*. It is scheduled for timeslot (4,2,1). Finally, at **Step 4**, during the preamble of the ninth timeslot node *k* has not changed its *RC* and rebroadcasts. All nodes are covered at this point. Hence, *c*, *d*, and *f* do not broadcast.

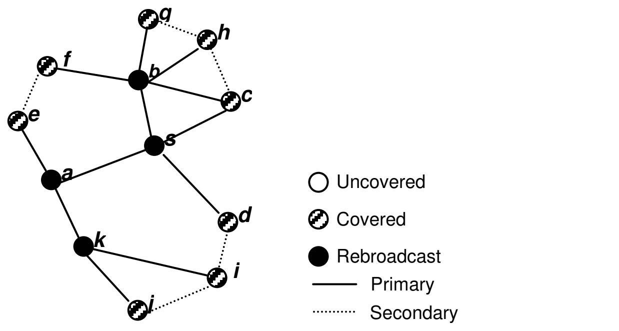

Fig 3: A network topology example, where {*s,b,a,k*} form an MCDS of the particular network graph. All nodes are covered after the broadcast session completes.

Consider the network of nodes shown on Fig. 3. For simplicity of presentation, only the NTSS variant of the *TS* scheme is discussed, which suffices to demonstrate the main mechanisms of

[6] As the name indicates, the NTSS possess some deficiency, which will be cured by the other variants of the scheme.

[7] I.e., that the current value of RC(*i*) matches the time-slot t_j.

the *TS* scheme. Suppose $u = 4$. Then, **Algorithm 1** constructs the *TS* as shown in **Example 1.** The sample run of NTSS is shown in Fig. 2, with the resulting network coverage depicted in Fig. 3.

Note that the *TS* structure allows **Algorithm 2** to give priority to nodes with higher *RC* that are scheduled later during the broadcast session. For instance, because of its larger *RC* value, node *k* will broadcast before nodes *c*, *d*, or *f,* albeit they were scheduled earlier. Ultimately, this eliminates the transmissions of nodes *c*, *d*, and *f*. Fig. 3 shows the network state after the broadcast session is completed. Edges between rebroadcast nodes, and between rebroadcast and covered nodes are labeled as primary, while edges between covered nodes as secondary.

4.5 Correctness of the TS-based Schemes

In the following proof of correctness, the network topology is assumed to be static during the broadcast session and the network graph is assumed to be connected.

Proposition 4: All *TS*-based schemes terminate in finite amount of time and guarantee full coverage of the network.

Proof (outline): Per **Algorithm 2**, for all *TS* schemes, a node is not scheduled to broadcast unless its *RC* is strictly greater than zero. Whenever a node *n* broadcasts, all of it neighbors receive the broadcast message and are marked as covered. Hence, the *RC* value of node *n* decreases to zero, and node *n* is not admissible in any future time-slot. Therefore, a node does not broadcast more than once during the execution of the algorithm. Since there is a finite number of nodes in the network, the algorithm terminates in a finite number of time steps.

Now, suppose that after a *TS*-based algorithm's termination there is at least one node, *D*, that is not covered. Since the network graph is connected, there exist at least one path from the source node, *S*, to the destination node *D*. Then, because *D* has not received the message, there are at least two neighboring nodes *X* and *Y* along this path, such that *X* has received the message and *Y* has not (note: *X* might be *S* and *Y* might be *D*). Therefore, *X*'s *RC* was greater than zero at algorithm's termination. This contradicts all *TS* schemes' termination condition. (Whenever a node is covered it computes its residual coverage. As long as node's residual coverage is greater than zero it is always scheduled to broadcast according to **Algorithm 2**.) Thus, by contradiction, all *TS*-based schemes cover all the network nodes. □

5. PERFORMANCE EVALUATION

We investigate the performance (as defined by a number of metrics) of various broadcast algorithms in three distinct network topology models: (1) a static network topology; (2) a mobile topology with *independent* mobility patterns of the nodes; and (3) a mobile topology with *correlated* (group) mobility pattern.

We compared the performance of the TS-based schemes with *position* aware *Responsibility Based Scheme* (*RBS*), suggested by Khabbazian and Bhargava [25]. In [25] authors show that RBS outperforms a few well known broadcast algorithms, such as the *Edge Forwarding* [13] algorithm, for example. Another broadcast protocol simulated here, the *Bordercast,* is the route discovery mechanism in the *Zone Routing Protocol* (*ZRP*) ([26]).

Bordercast relies only on local topological information to select the nodes to relay the broadcast message. To include an algorithm that constructs a *backbone* structure prior to the broadcast session we selected *Funke's* algorithm [21], which was shown to obtain one of the best approximation ratios to the size of the MCDS (i.e., |MCDS|), outperforming leading algorithms such as Wan's algorithm [20]. Finally, for comparison, we simulated Liu's algorithm [11] – a *node forwarding* algorithm that relies on 1-Hop positional information. Both RBS and Funke's algorithms are among the most efficient broadcast schemes in the literature.

In all the experiments, the simulation area is a 200[m] by 200[m] square; the inner square area is of dimensions (200 - R)[m] by (200 - R)[m], where R[m] is the transmission radius of all nodes and was set to R = 25[m]. The number of nodes in the network was varied to investigate the schemes' performance at different node densities. To eliminate the effects of the physical and MAC layers on the broadcast algorithms' performance, we assume no signal interference, no fading, and a "perfect" MAC Layer. All the broadcast algorithms were implemented in a JAVA discrete event simulator.

5.1 Static Network Topology

5.1.1 Transmission Complexity

The number of transmission during a broadcast session (i.e., "transmission complexity"), a crucial metric for an efficient broadcast algorithm, is investigated on Fig. 4. We were interested in understanding how our distributed algorithms compare with the centralized greedy approximation of the MCDS.

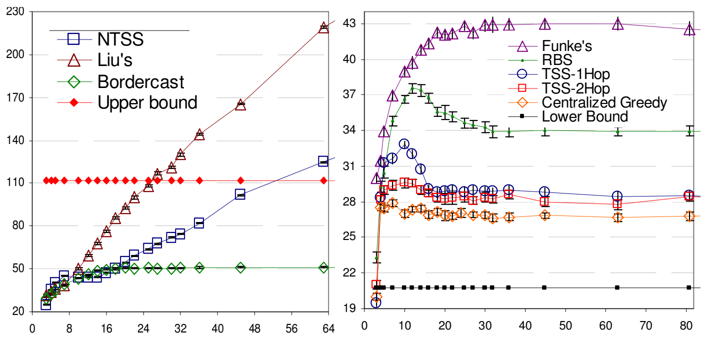

Figure 4. Number of transmissions (y-axis) vs. average node degree (x-axis) for different broadcast algorithms.

As an additional comparison of the transmission complexity, we calculated the number of transmissions using the *Linear Hexagon Coverage* technique, which provides a close approximation to the minimum number of transmissions needed to cover the entire network area, assuming sufficient node density. The upper bound of transmission complexity should be interpreted as the maximal number of transmissions that would be required by any broadcast algorithm that avoids duplicate coverage of nodes and, hence, is density independent. Due to space limitations, we omit here the derivation and an elaborate discussion of the bounds in Fig. 4.

The TSS-1Hop algorithm performs only slightly worse than the TSS-2Hop algorithm and mildly worse than the centralized greedy scheme.

Both, the NTSS and the Liu's algorithms are not density independent (surpass the upper bound). The NTSS algorithm does not perform well, because often a node is covered multiple times by transmissions of its 1-hop neighbors which occur in the same time-slot, thus resulting in redundant transmissions. This is especially common in time-slots towards the end of a broadcast

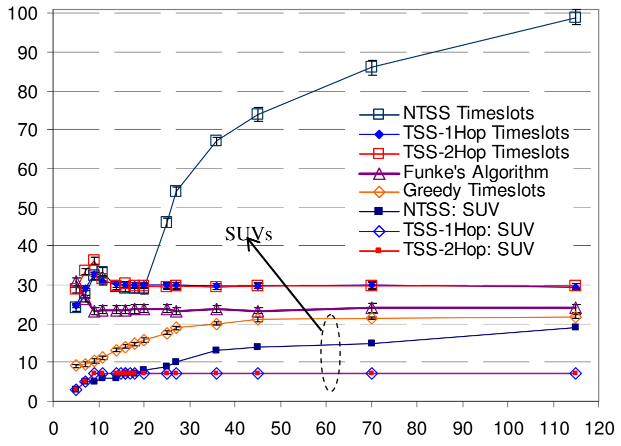

Figure 5. Delay (number of timeslots) (y-axis) vs. average node degree (x-axis). (SUV denotes Sequence Upper Values (the parameter *u*) for which the delay of *TS* schemes was measured.)

session. The number of such transmissions during a simulation run is proportional to the number of nodes in the network and, hence, the transmission complexity continues to increase with the node density.

5.1.2 Delay

The time delay – the number of time-slots needed to complete a broadcast session – is presented in Fig. 5. The delay performance of the three *TS*-based schemes was obtained with *u* set to the smallest value, such that the number of broadcast transmissions is still close to optimal. These smallest values of *u* were determined via simulations; it is worth noticing that beyond average node degree of 15, these *u* values for TSS-1Hop and TSS-2Hop remain essentially constant with respect to the network density. Thus the upper value could be fixed prior to network deployment, so that the resulting time delay and the number of broadcast messages remains close to the algorithm's optimum behavior. The results for TSS-1Hop and TSS-2Hop hint that a good practice would be to set *u* to the average node degree for networks of node degrees up to 20. For networks of larger node degrees, *u* is largely density- independent and can be simply fixed at 20.

In general, there is a tradeoff between the two performance metrics: the number of transmissions during a broadcast session and the duration of a broadcast session. Intuitively, it would be beneficial to set *u* to a large value for finer-grain resolution of priorities among the nodes with different *RC* values and, thus, for a better approximation to the greedy "oracle" scheme. However, large values of *u* have the disadvantage of increasing the delay, since the *TS* grows longer (quadratic in *u*, from **Proposition 1**) and large number of initial time-slots go by empty, as no nodes are then admissible

5.2 Changing Network Topology

For all of the mobility experiments presented here, each data point was obtained after a simulation warm-up time of 1000 [s] in which the mobility model could converge to a stationary node distribution.

5.2.1 Gauss-Markov Node Mobility

In the Gauss-Markov mobility model (GMMM) [33, 34], the time is split into time intervals. At the beginning of the k^{th} time interval, nodes' velocity is updated based on their velocity in the (*k*-1)th time interval and according to the following rule: $v_k = \alpha v_{k-1} + (1-\alpha)\bar{v} + \sqrt{(1-\alpha^2)}v_{x_{n-1}}$. Here, v_{k-1} is the velocity (speed and direction) of a node in the previous time interval, $v_{x_{n-1}}$ is a Gaussian random variable, $\bar{v}$ is the mean value of the velocity, and α is a parameter that determines the degree of which the current velocity depends on the previous velocity (the amount of "memory"). As α approaches 1, nodes' motion is constant; as α approaches 0 nodes' motion becomes more random. At the end of each interval, the position of a node in the network area is updated according to its velocity during this interval.

Velocity and position update interval	0.2 [s]
Velocity standard deviation	0.75 [m/s]
Velocity mean	Varied
Alpha	0.75

Table 1: The parameters of the Gauss-Markov Mobility Model.

Fig. 6 depicts the number of transmissions generated by the four investigated broadcast algorithms. As the average speed of the nodes increases, the number of broadcast transmissions saturates for all algorithms. This is an artifact of the finite network area – beyond a certain speed (about 20-30 [m/s] in this case), the locations of the nodes in the next time interval become random with respect to the locations in the previous time interval. This is equivalent to formation of a completely new random placement of nodes in each time interval, and increasing the speed further does not alter the performance of the algorithms anymore.

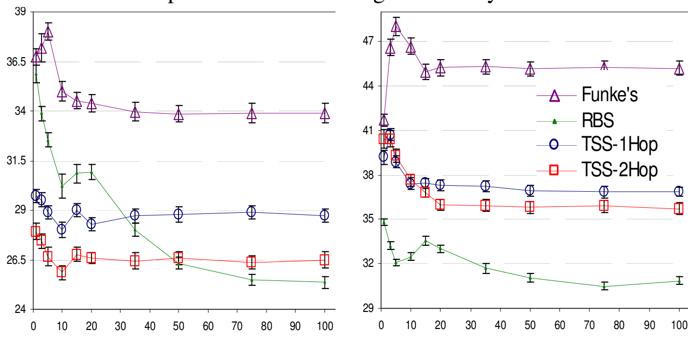

Figure 6. GMMM – number of transmissions (y-axis) vs. average speed [m/s] (x-axis); 200 and 500 nodes (*left*) and (*right*) respectively.

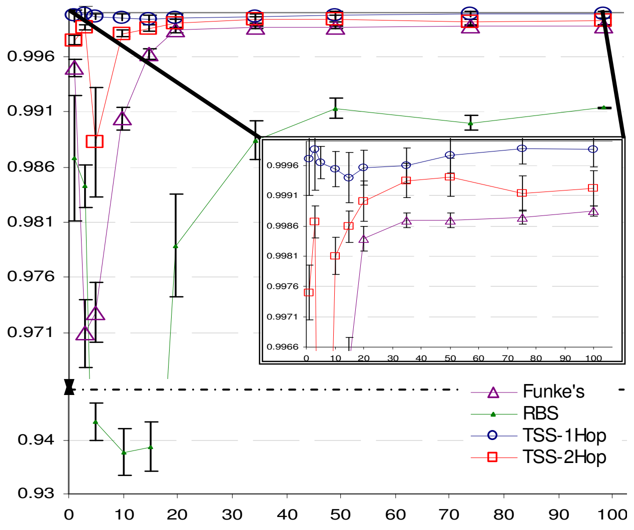

Figure 7. GMMM – the fraction of network nodes covered (y-axis) vs. average speed [m/s] (x-axis); 200 nodes.

However, the number of transmissions prior to this saturation point depends on the specific mechanisms of the broadcast algorithm in question. For instance, the decrease in the number of transmissions prior to saturation as mobility increases for the *TS*-based schemes results from the scheduling policy of these algorithms: the larger their residual coverage is, the sooner the nodes transmit. As mobility increases, there is a large chance that a node would move into an area containing larger concentration of uncovered nodes, thus increasing its *RC* and, overall reducing the total number of transmissions. Furthermore, the fraction of nodes covered increases as well, as shown in Fig. 7.

As opposed to the static case, where a broadcast session is bound to cover all the network nodes as long as the network is connected, with mobility there is a chance that some nodes will not be covered. Fig. 7 demonstrates the coverage (fraction of the network nodes reached during a broadcast session). The performance of all algorithms saturates beyond some mobility threshold. In the mobility region of 10-20 [m/s] and for smaller densities (200 nodes in the network), all algorithms are affected by disconnections in the graph during the broadcast session. Comparatively, for larger densities, all algorithms perform substantially better (the figure is omitted due to space limitations).

The notable drop in performance of the RBS scheme in Fig.10 for a range of low speeds is due to the specific RBS mechanisms: node *A* broadcasts if among its neighbors there is an uncovered node (call it node *B*), and node *B* is such that the distance between *A* and *B* is smaller than the distance between *B* and any other of the covered neighbors of *A*. For very low velocity, RBS performs well and covers almost the entire network; however, as the velocity increases, node *A*'s estimation of the distance between *B* and *A*'s neighbors becomes less accurate. Though, as the velocity of nodes increases further, RBS benefits from the fact that *A* could move into areas where only a small number of nodes is covered.

5.2.2 Group Mobility

Since independent individual node movement may be unrealistic for certain scenarios (e.g., hikers, tourist groups, military platoon, etc.), we have also simulated a group mobility model, which is

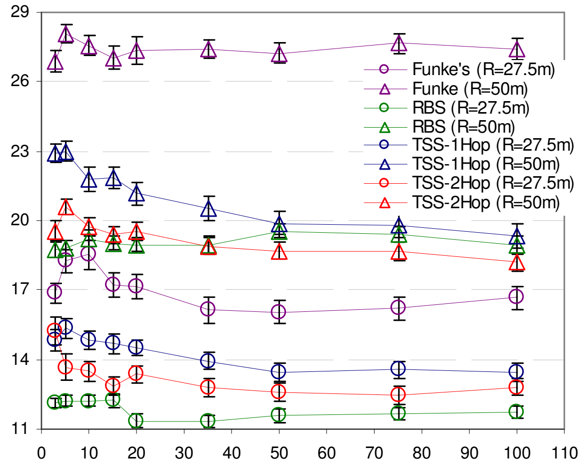

Figure 8. RPGM – number of transmissions (y-axis) vs. average speed [m/s] (x-axis); reference point (RP) radius equals 27.5[m] (*circles*) and 50[m] (*triangles*).

based on the *Reference Point Group Mobility* (*RPGM*) model described in [35]. The implemented version of RPGM follows the *Nomadic Community Mobility* model described in [34]. In this model the network nodes form groups that move together from one reference point (*RP*) in the simulation area to another. The RPs have a radius of influence and the group of nodes is free to move randomly within this radius. The simulation parameters (such as the group sizes and the number of groups in the network) used here were borrowed from [34] and [36] for the Nomadic and the RPGM models, respectively. The number of nodes was set to 200.

Pause time for mobility around the RP	0.3 [s]
Nodes velocity update frequency	0.5 [s]
Maximum reference point pause time	4 [s]
RP radius	Varied
Average group size	*20% of network size*
Nodes mean velocities	Varied

Table 2: The parameters of the RPGM mobility model.

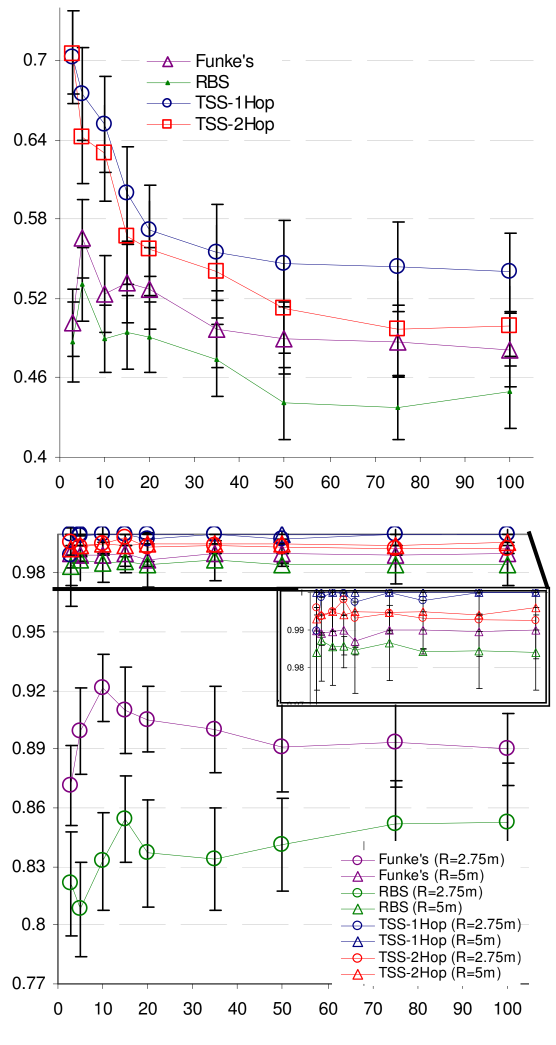

Figure 9. RPGM – fraction of nodes covered (y-axis) vs. average speed [m/s] (x-axis): (*top*) reference point (RP) radius equals 12.5[m]; (*bottom*) RP radius equals 27.5[m] and 50[m].

Figures 8 and 9 depict the number of transmissions and the network coverage. The RP radius (labeled as R in the Fig. 9), is varied to investigate the effects of node clustering about the reference point. As can be observed from these figures, the smaller the radius is, the more frequent are the disconnections and, consequently, the performance of the algorithms degrades.

As in the GMMM case, in the RPGM case too, the number of transmissions and the network coverage saturate as the network mobility increases. However, in the RPGM case there is a minimal decrease in the number of transmissions with respect to the increase in mobility; e.g., for 12.5[m] the decrease in the number of transmission is largely due to the highly clustered and disconnected network topology.

Overall, in dynamic scenarios, such as those when mobility is introduced, algorithms which rely on local topology information, such as TSS-1Hop and TSS-2Hop, outperform algorithms such as RBS and Funke's.

6. DISCUSSION

In this paper, we introduced a novel scheme for broadcasting in wireless networks that mimics in performance of the centralized greedy algorithm. This is accomplished through distributive prioritization of transmissions based on nodes' residual coverage (*RC*) and the particularly designed Time Sequence (*TS*) to schedule the nodes' transmissions. The basic NTSS scheme was improved to eliminate a major source of inefficiency − multiple coverage of nodes by more than one transmission − which resulted in the TSS-1Hop and the TSS-2Hop schemes. We proved the schemes' correctness (i.e., guaranteeing full coverage in finite time).

Through simulations and based on two metrics − the transmission complexity and the delay − we compared the performance of our schemes with other leading broadcasting schemes, initially in static networks. The TSS-1Hop and the TSS-2Hop schemes outperform all other schemes with respect to the number of message transmissions. Also, the TS-based schemes do not require additional equipment, such as GPS. Furthermore, this performance is achieved with bounded latency, and is independent of network density.

Next, we considered two types of mobility scenarios: individual node and group mobility. We showed experimentally that the TS-based schemes outperform the other schemes with respect to the network coverage by the broadcast transmissions.

Our study allows examination of the basic tradeoffs in the design of the broadcasting protocols, such as the tradeoff between transmission complexity and delay. It is worth noticing that the TS-based schemes facilitate control of this tradeoff through a single parameter only − the u parameter.

In addition, the current work invites interesting questions regarding the TS schemes performance that remain to be further studied. For instance, schemes' behavior in the presence of multiple simultaneous broadcast sources could be considered; the algorithms' performance given errors in the process of RC computation should also be investigated.

7. RELATED WORK

The problem of efficient broadcasting has been extensively studied in the technical literature. The initial simple concept of flooding evolved to more sophisticated schemes through building optimal network subgraphs. All through, the main algorithmic challenge has been to reduce the number of transmitted messages needed to reach all the network nodes.

Among the major shortcomings of pure flooding are the resulting large transmission complexity and the notorious broadcast storm [1]. The *Scalable Broadcast Algorithm* [2] alleviates somewhat this problem utilizing 1-Hop neighbor information.

A different approach is taken by probabilistic broadcast protocols that associate some (re)transmission probability to each node receiving the broadcast message. Schemes exploring such mechanisms were suggested in [3, 4, 5, 6, 7, 8, 9].

The interest in probabilistic broadcasting schemes is due to their inherent low transmission overhead, low processing complexity, and high tolerance of frequent and rapid topological changes. Balancing these benefits, though, is the disadvantage of inability to guarantee full network coverage.

In contrast, deterministic broadcast algorithms innately guarantee full network coverage (assuming ideal MAC layer). In the deterministic scheme of Multipoint Relaying proposed in [10], the set of retransmitting neighbor nodes is reduced from the set of all neighbors to the minimum subset of neighbors that cover the same area as that covered by the original set. This approach is an example of the "minimum forward-node set" strategy, and works such as [11, 12, 13, 14] provide approximate solutions to this NP-hard problem. To avoid the transmission of the list of forwarding nodes along with the broadcast message, a technique of self-pruning [15, 16] has been proposed.

The forward-node set and, consequently, the self-pruning problems can essentially be viewed as the task of solving the NP-hard "Minimum Connected Dominating Set" (MCDS) problem [13]. Several studies [18, 19, 20, 21, 22] have attempted to tackle the problem by constructing a communication backbone prior to the broadcast initiation. These schemes can sometimes dramatically reduce the number of transmissions. Nevertheless, as shown in [23], they do not tolerate well frequent network topological changes. For volatile communication environments, an approach to dynamically construct CDS is a better alternative. Works such as [24] and [25] offer potential solutions. However, further research is required to study the scalability of these approaches.

A number of studies have attended to the theoretical bounds of broadcast and related information dissemination mechanisms. For instance, [37, 38] provide bounds on gossiping, broadcast, and rumor spreading in general networks. Influential works by Peleg et al. in the spirit of [39] demonstrate important lower bounds on broadcast in radio networks.

8. ACKNOWLEDGMENTS

This work was supported by the NSF grant numbers ANI-0329905 and CNS-0626751 and by the AFOSR contract number FA9550-09-1-0121/Z806001. We would also like to thank the anonymous reviewers for their constructive critique and suggestions regarding this manuscript.

9. REFERENCES

[1] S.-Y. Ni, Y.-C. Tseng, Yuh-Shyan Chen, and J.-P. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Networks", In Proc. ACM/IEEE MobiCom, 1999, pp. 151 - 162

[2] W. Peng, and X.C. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," In Proc. ACM/IEEE MobiHoc, 2000, pp. 129–130

[3] Z. J. Haas, J. Y. Halpern, and L. Li, "Gossip-Based Ad Hoc Routing", In Proc. IEEE INFOCOM, 2002, pp. 1707-1716

[4] Y. Sasson, D. Cavin, and A. Schiper, "Probabilistic broadcast for flooding in wireless mobile ad hoc networks," in Proc. IEEE WCNC, 2003, pp. 1124-1130

[5] D. Scott and A. Yasinsac, “Dynamic probabilistic retransmission in ad hoc networks,” ICWN, 2004, pp. 158--164, CSREA Press

[6] J. Cartigny, D. Simplot, and J. Carle, “Stochastic flooding broadcast protocols in mobile wireless networks,” Tech. Report LIFL, Univ. Lille1, 2002-03, 2002

[7] J. Kim, D. J. Scott, and A. Yasinsac. “Probabilistic broadcasting based on coverage area and neighbor confirmation in mobile ad hoc networks,“ In Proc. IEEE Globecom, 2004

[8] A. Keshavarz-Haddad, V. Ribeiro, and R. Riedi, “Color-based broadcasting for ad hoc networks,” WiOpt, 2006

[9] S. Pleisch, M. Balakrishnan, K. Birman, and R. van Renesse, “MISTRAL: Efficient Flooding in Mobile Adhoc Networks,” ACM MobiHoc 2006.

[10] A. Qayyum, L. Viennot, and A.Laouiti, “Multipoint relaying for flooding broadcast messages in mobile wireless networks,” HICSS, 2002, Hawaii.

[11] H. Liu, P. Wan, X. Jia, X. Liu, and F. Yao. “Efficient Flooding Scheme Based on 1-hop Information in Mobile Ad Hoc Networks,” IEEE INFOCOM, 2006

[12] W. Lou and J. Wu. “Double-covered broadcast (DCB): A simple reliable broadcast algorithm in MANETs,” IEEE INFOCOMM, 2004

[13] Y. Cai, K. Hua, and A. Phillips, “Leveraging 1-Hop Neighborhood Knowledge for Efficient Flooding in Wireless Ad Hoc Networks,” IPCCC, 2005, pp. 347-354

[14] G. Călinescu , I. I. Mandoiu , P. Wan , A. Z. Zelikovsky, “Selecting forwarding neighbors in wireless ad hoc networks,” MONET, 2004, v.9 n.2, p.101-111

[15] M. Sun, W. Feng, and T. Lai, “Broadcasting in Ad Hoc Networks Based on Self-Pruning,” IEEE GLOBECOM, 2001, pp. 2842-2846,

[16] Jie Wu , Fei Dai, “A Generic Broadcast Protocol in Ad Hoc Networks Based on Self-Pruning,” IPDPS, 2003, p.291.

[17] J. Wu and H. Li, “On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks,” Third Int'l Workshop DIAL-M, 1999, pp. 7-14,

[18] A. Keshavarz-Haddad, V. Ribeiro, and R. Riedi, “DRB and DCCB: Efficient and Robust Dynamic Broadcast for Ad Hoc and Sensor Networks,” In Proc. IEEE SECON, 2007

[19] R. Misra, C. Mandal, "Minimum Connected Dominating Set Using a Collaborative Cover Heuristic for Ad Hoc Sensor Networks," IEEE Trans. Parallel and Distributed Sys, vol.21, no.3, pp.292-302, Mar. 2010

[20] P.-J. Wan, K. M. Alzoubi, O. Frieder, “Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks”, IEEE INFOCOM 2002, pp 1597-1604.

[21] S. Funke, A. Kesselman, U. Meyer, and M. Segal, “A Simple Improved Distributed Algorithm for Minimum CDS in Unit Disk Graphs”, ACM Trans. Sensor Networks, vol. 2, no. 3, pp. 444-453, 2006

[22] B. Gao, Y. Yang, H. Ma, “An Effective Distributed Approximation Algorithm for Constructing Minimum Connected Dominating Set in Wireless Ad Hoc Networks,” ICCIT, 2004, p.658-663

[23] N. Meghanathan, A. Farago, “On the stability of paths, Steiner trees and connected dominating sets in mobile ad hoc networks, Ad Hoc Networks,” vol. 6, no. 5, pp. 744-769, Jul 2008

[24] I. Stoimenovic, “Comments and Corrections to ‘Dominating Sets and Neighbor Elimination-Based Broadcast Algorithms in Wireless Networks”, IEEE Trans. Parallel and Distributed Systems, vol. 15, no. 11 Nov. 2004

[25] Khabbazian, M., Bhargava, V. “Efficient Broadcasting in Mobile Ad Hoc Networks,” IEEE Trans. on Mobile Comp., Vol.8, No.2, Feb. 2009

[26] Z. J. Haas and R. Barr, “Density-Independent, Scalable Search in Ad hoc Networks,” IEEE PIMRC, 2005

[27] X. Wang, R. Dokania, A. Apsel, "PCO Based Synchronization for Ad-Hoc Duty-Cycled Impulse Radio Sensor Networks", Special issue on cognitive sensor networks of IEEE Sensors Journal, 2010.

[28] R. Solis, V. Borkar, and P. R. Kumar, "A new distributed time synchronization protocol for multihop wireless networks," In Proc. IEEE CDC, 2006, pp. 2734-2739.

[29] X. Cheng, X. Huang, D. Li, W. Wu, and D.-Z. Du, “A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks,” Networks, 42, 2003, 202--208.

[30] H. Hunt, III , M. Marathe , V. Radhakrishnan , S. Ravi , D. Rosenkrantz , R. Stearns, “NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs,” Journal of Algorithms, v.26 n.2, p.238-274, Feb. 1998

[31] S. Guha and S. Khuller, “Approximation Algorithms for Connected Dominating Sets,” Algorithmica, vol. 20, pp. 347-387, 1998.

[32] B.N. Clark, C.J. Colbourn, and D.S. Johnson, “Unit Disk Graphs, ” Discrete Mathematics, Vol. 86, 1990, pp. 165-177.

[33] B. Liang and Z. Haas, “Predictive Distance-Based Mobility Management for PCS Networks,” In Proc. IEEE INFOCOM, 1999.

[34] T. Camp, J. Boleng and V. Davies, “A survey of mobility models for ad hoc network research.” Wireless Communications & Mobile Computing 2 5 (2002), pp. 483–502

[35] Hong X, Gerla M, Pei G, Chiang C. “A group mobility model for ad hoc wireless networks.” In Proc. ACM MSWiM, 1999.

[36] F. Bai, N. Sadagopan, and A. Helmy, "IMPORTANT: A framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks," In Proc. IEEE INFOCOM, 2003, pp. 825-835

[37] D. Liu , M. Prabhakaran,” On Randomized Broadcasting and Gossiping in Radio Networks,” In Proc.COCOON, 2002, pp.340-349

[38] D. Alistarh , S. Gilbert , R. Guerraoui , M. Zadimoghaddam, “How efficient can gossip be? (on the cost of resilient information exchange),” In Proc. ICALP, 2010

[39] N. Alon , A. Bar-Noy , N. Linial , D. Peleg, “A lower bound for radio broadcast,” Journal of Computer and System Sciences, v.43 n.2, p.290-298, Oct. 1991