Towards Optimal Broadcast in Wireless Networks

Zygmunt J. Haas, *Fellow, IEEE*, and Milen Nikolov

Abstract— Broadcast is a fundamental operation in networks, especially in reconfigurable wireless ad hoc networks. For example, some form of broadcasting is used by all on-demand mobile networks routing protocols, when there is uncertainty as to the location of the destination node, or for service discovery. In this work, we present a new approach to efficient broadcast in networks with dynamic topologies, and we introduce the Time Sequence Scheme (TSS), a new online local broadcasting algorithm for such networking environments. TSS ranks by priority, in a distributed way, candidate broadcasting nodes so that the overall number of re-broadcasts in the network is minimized. We evaluate TSS, showing that its performance comes remarkably close to the corresponding theoretical performance bounds, even in the presence of packet loss due, for example, to MAC-layer collisions. Furthermore, we compare our algorithm with a number of recently proposed schemes considering their performance in various realistic network mobility scenarios. We demonstrate that TSS performance is robust in the context of mobility induced topology reconfigurations – including temporal network partitioning – during propagation of the broadcast message.

Index Terms— wireless communication, distributed broadcast, efficient flooding, stochastic routing, Connected Dominating Set

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 networks where all communications are carried over a wireless medium and network nodes are limited in energy and computational power (e.g. sensor and mesh networks), an efficient broadcast mechanism is exceptionally important for the overall network performance. Similarly, a robust broadcast solution is required in networks where topology can change rapidly due to nodes' mobility (e.g. networks of unmanned aerial vehicles (UAVs), or interconnected devices attached to people).

For instance, among the various proposed wireless network 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 the source and the destination nodes. Also, in highly reconfigurable topologies, where the lifetime of network routes may be shorter than the duration of a communication (especially in the case of connection-oriented communication) broadcast, by itself, could be used as a routing mechanism. Yet in other scenarios, data dissemination to all nodes in a sensor network is needed and broadcast is an obvious solution. Being such an essential network operation, it is not surprising the importance of an efficient broadcast implementation has been widely accepted by the networking community. (Section 7 provides more details and references to a number of related works in the technical literature.)

Z. J. Haas is with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853 USA (e-mail: haas@ece.cornell.edu).
M. Nikolov, is with the School of Electrical and Computer Engineering Cornell University, Ithaca, NY 14853 USA (e-mail: mvn22@ cornell.edu).

The main contributions of this paper include the design of the Time Sequence Scheme (TSS), a novel broadcasting algorithm that ranks and orders in time the transmissions of broadcasting nodes, so that the overall number of re-broadcasts in the network is minimized. TSS utilizes only 1-hop topology information while *simultaneously* achieving full coverage of the network with close to optimal number of broadcast messages and with low delay.

Furthermore, the algorithm is *robust to rapid topological changes and network partitions*. It retains its performance in a full network stack implementation, where packet loss at network and MAC layers, for instance, may be present. We model, simulate, evaluate and compare the proposed broadcast scheme with the most efficient schemes to-date in the technical literature. We demonstrate that the proposed scheme outperforms them in the metrics considered. Also, TSS does not require positional information, as the latter is infeasible or costly to get in many network settings.

In summary, the algorithm discussed in this paper is efficient, distributed, performs well in highly dynamic network scenarios, and relies only on local coverage information. To the best of our knowledge, previously suggested schemes have *not* satisfied *all* these desiderata for practical broadcast.

Next, Section 2 describes the broadcast problem we tackle and the related challenges to its solution. Section 3 provides the system model and assumptions. Section 4 explains the basic intuition behind the TSS broadcast solution. More details and an execution example of TSS are provided in Section 5. Section 6 studies comparatively the performance of TSS in various networking scenarios, including realistic mobility models (e.g., both the *Gauss-Markov* [33, 34] and the *SLAW* [35] mobility models are considered). To evaluate the effects of noise, interference and practical MAC layer, for instance, the results of a full network stack simulation of the TSS are provided as well in Section 6. Sections 7 and 8

conclude the paper, placing it in the context of future and prior work, respectively. In the Appendix, simple closed-form bounds on wireless broadcast efficiency are derived for completion, providing benchmarks for broadcast algorithms.

2 The Efficient Broadcast Problem

Assume we represent the network as a connected graph $G=(S,E)$, where S is the set of all the network nodes and E is the set of all the links. Given a source node $s_0 \in S$ that transmits a broadcast message m, suppose it is desirable to reduce the number of rebroadcasts of m required to propagate m in the entire network. Consider the set of nodes $Q \subseteq S$ such that for each node $v \in S/Q$, v has a neighbor in Q. If Q is a connected subgraph of G, Q forms a connected dominating set (CDS). Notice that after the termination of *any* broadcast scheme that propagates m to *all* nodes in S, the set of nodes that the scheme has picked for broadcast forms a CDS. For pure flooding [1] this is the trivial CDS: S.

A Minimum Connected Dominating Set (MCDS) of G is a CDS in G with minimum cardinality. If just all the nodes in a MCDS broadcast message m, all nodes in G will receive m and the number of broadcast nodes is minimized. In the context of wireless networks, we observe that, first, minimizing the number of rebroadcasts (and rebroadcasting nodes) would expend substantially less energy and bandwidth especially as compared to flooding [45, 50]. Second, recent practical variants of flooding, such as Glossy and Flash [47, 50], can be very rapid and m reaches all nodes with remarkably low latency sans finding MCDS. However, as noted in [50], in many cases the design and application of these flooding schemes is orthogonal to the task of finding MCDS. I.e., the number of transmissions can be minimized by, first, finding the set, Q, of network nodes forming a MCDS; and, second, constraining the flooding only within Q one can minimize latency. Such approach was shown to significantly reduce the energy cost of the Flash flooding protocol [50], for instance.

Challenges: We are not the first to note the importance of MCDS to solving and modeling the wireless network broadcast problem (the reader is referred to Section 7 for related work). However, finding a MCDS of a graph G is an NP-hard problem (even if G is a Unit Disk Graph (UDG) [32]) and one needs to consider an appropriate heuristic algorithm for only approximate CDS. The desired features and challenges for a such *practical and efficient* algorithm for dynamic wireless reconfigurable networks are that it should

(1) reach *all* the network nodes;
(2) transmit the broadcast message as few times as possible (or, equivalently, reduce the number of times that the broadcast message is received by a network node, optimally to only once);
(3) minimize delay (i.e., the time needed for the broadcast message to be received by the entire network);
(4) require only locally available information (e.g., only knowledge of the 1-hop neighborhood topology);
(5) minimize the effects, on (1), (2), and (3) above, of topological changes during a broadcast propagation caused by mobility, and due to packet loss.

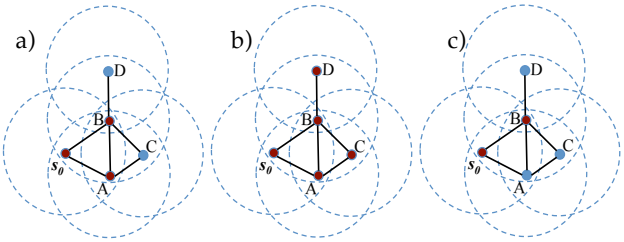

Fig. 1: The output of four algorithms given a sample network on 5 nodes, where the source node is s_0. The broadcast nodes in the output CDS are in red. a) RBS and CCS output broadcast nodes s_0, *A*, and *B*; b) Funke's algorithm outputs s_0, *A*, *B*, *C*, and D; c) TSS outputs the optimum solution in this case which is s_0 and *B*.

Unfortunately, few of the so-far-proposed approaches to the broadcast problem satisfy all the above desiderata. Pure *flooding* protocols violate (2) above; furthermore, they may lead to the notorious "broadcast storm" problem [1]. *Probabilistic schemes* [3 – 9, 49] do not satisfy (1) above. To ensure full coverage, stochastic schemes would need to transmit with probability close to 1, whereby the scheme degrades to flooding. Also, albeit simple, fast, and flexible they are not as efficient in terms of finding small size CDS. Following a different strategy, *backbone-based* algorithms have also been proposed (e.g., the reader is referred to [44] for comparison of such schemes). They rely on finding approximate MCDS which are constructed by identifying dominating sets, maximal independent sets, or Steiner trees *prior* to broadcast. While satisfying (1) – (3) above and often guaranteed to find a CDS with size within a constant of the MCDS, such algorithms do not tolerate dynamic topologies well [19 – 22]. Thus, they are incompliant with (5). Furthermore, some of these algorithms are centralized [29, 30] and violate (4). Two examples, [25, 45], of broadcast algorithms that satisfy (1) – (5) have been described most recently. In [25] though, the algorithm requires the knowledge of the nodes' geographical position, which is not always feasible. The proposed algorithm in [45] requires only 2-hop local topology information in lieu of geographical position and is thus more closely related to the goal of our paper. However, as the authors of [45] note, in highly mobile networks the algorithm's performance drops due to the 2-hop topology knowledge requirement.

To address these challenges and improve on the state-of-the-art schemes, our TSS's design carefully avoids subtle algorithmic inefficiencies that undermine the performance of broadcast schemes in the current technical literature. Illustrating some of these inefficiencies, Fig. 1 shows a very simple network topology that is input to three algorithms we found to be among the top performing in the context of reducing the number of broadcast transmissions and finding small size CDS.

Fig. 1a shows the output of the *position-aware Responsibility Based Scheme* (*RBS*), suggested by Khabbazian and Bhargava [25]. Per RBS a network node i receiving broadcast message m transmits m if 1) among i's neighbors there is a node j that has not received m yet and 2) the Euclidean distance between i and j is smaller than the Euclidean distance between j and any other of the neighbors of i that has already received m. Given source node s_0, this broadcast rule

requires the transmission of nodes A (since $AC < BC$) *and* B (since D has not received m after A broadcasts).

Incidentally, Fig. 1a also shows the output of the more recent novel broadcast algorithm utilizing only *2-hop* topology information (sans position information) developed by Khabbazian and Bhargava in [45]. We here dub this scheme the *Coverage Condition Scheme (CCS)*. In a nutshell, each node j maintains a list L_j of neighbors. L_j is gradually pruned by j. Based on information piggybacked in each retransmitted broadcast message and 2-hop neighborhood topology knowledge, j removes nodes from L_j. According to j the removed nodes have received or will receive m from a different node in the network. Per CCS, only nodes selected for broadcast transmit m. If L_j is non-empty, node j is eligible to select node $i \in L_j$ to broadcast next only if j itself has been previously selected for broadcast by a neighboring node k. The selection of i by j is done at random from L_j. If at some point L_j is empty the *coverage condition* of j is satisfied, and j does not need to broadcast if it has not been previously selected to broadcast by k. Otherwise, if j was selected to broadcast by k, j broadcasts but does not select a forwarding node. Here s_0's list contains the nodes A and B. Node s_0 selects node A to broadcast next (this happens with probability of 50%). Node C receives m from A, but node D has to receive m from B. Thus again both A and B broadcast m.

To include an algorithm that constructs a *backbone* structure *prior* to the broadcast session, Fig. 1b shows the output of the algorithm in [21] developed by Funke et al.. Funke's algorithm provably obtains one of the best approximation ratios to the MCDS. The algorithm is distributed and finds a CDS with size guaranteed to be within 6.94 from the optimum MCDS solution for UDGs. The algorithm has behavior similar to the Wan-Alzoubi-Frieder's algorithm from [20]. Funke's algorithm finds a solution CDS, which is a union of an independent set and a connected set in G. In the case of this example topology, the algorithm needs to select all the nodes in the network as a final solution.

Finally, Fig. 1c show the optimal solution given a source node s_0. In this case the MCDS consists only of node B. The broadcasts of the source s_0 and node B would be sufficient to cover the entire network. All of the three algorithms considered above miss the optimum solution. Notice that by design all of the three algorithms approach the goal of reducing the number of broadcast nodes somewhat implicitly. RBS relies on expanding the *area* covered by the broadcast nodes quicker. CCS relies on *pruning redundant* transmissions, but the effectiveness of the selected broadcast nodes' transmissions is not considered as much. Funke's algorithm exploits interesting relationships between graph theoretic properties of the independent and connected dominating sets to provide a good theoretical guarantee for the *final* solution. However, it does not necessarily target the minimization of the number of broadcasting nodes locally at each step, or the maximization of the number of nodes that receive m for the first time.

Thus per all three algorithms the main utility of a broadcast node – the number of nodes that have not received message m yet – is not explicitly pursued or maximized, leaving room for inefficient broadcast decisions. In contrast, TSS's target is to *explicitly maximize the utility of each broadcast*: every broadcast node i should reach as many as possible neighboring nodes that have not received m yet. If there were a node j in the network whose transmission would potentially reach fewer such neighboring nodes, j should wait until i finishes its broadcast first. This principle and its careful distributed implementation allow TSS to find the optimum solution shown in Fig. 1c. Both nodes A and B receive the message m from s_0, however node B has two neighbors (nodes C and D) that have not received m at this time, and node A has only one neighbor (node C). TSS thus considers both A and B as broadcast candidates and assigns them respective tentative broadcast times. B has higher rank and receives higher priority than A. The broadcast time of B is earlier than the broadcast time of A. B broadcasts and at the time for broadcast assigned to A, A no longer has neighbors who have not received m. Following TSS policy A does not broadcast in this case. Thus TSS finds the optimal solution here: the broadcasts of the source node s_0 and B are sufficient.

Although the optimal performance of TSS in this extremely simple example is not an accident, the actual distributed design and implementation of TSS of course requires more care and justification. In the remaining of this work we outline the algorithms behind TSS and some of the unexpected benefits of the scheme in highly dynamic mobile network scenarios. We start by providing a few definitions to setup the system model and context next, which will facilitate the description of the proposed scheme.

3 System Model

The network model consists of N equal-capability nodes with unique IDs, randomly distributed in a 2D plane. Though, the results in this paper apply to 1D and to 3D networks, too. 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 Euclidean distance between them is less than R [meters]. Thus, the network is modeled as a Unit Disk Graph (UDG).

On the MAC layer, we consider two scenarios: a) a *perfect* MAC layer to isolate other effects (e.g., collisions, links asymmetry, etc.), so that the broadcast performance metrics reflect only the algorithmic efficiency; and b) *packet loss* (at the MAC and other network layers) to evaluate the performance of the algorithm in practical network settings, where collisions and links asymmetries, among other deleterious effects, are present.

The system operation is time-slotted, and the network nodes are assumed to be only 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 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 (e.g., [27]).

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

Definition 2: A *covered node* is a node that has already re-

ceived the broadcast message in a prior transmission of the broadcast session. To simplify notation, in what follows we assume only a single message m needs to be propagated in the network, during the duration of each broadcast session. However, TSS can be trivially extended to handle the broadcast of multiple different messages simultaneously originating from different sources in the network.

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

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

Definition 4: We define 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 ($s \in S$). We note that at any time, $Q \subseteq C \subseteq S$ and that $|S| = N$.

Finally, we assume all nodes are cooperative.

4 Solution Demystified

The problem of finding the most efficient broadcast scheme is equivalent to finding an approximation of the MCDS, and satisfying (1) – (5) above. Since finding the MCDS is an NP-hard problem [32], one needs to consider an appropriate heuristic, with the centralized *greedy algorithm* being one such an alternative (e.g., [31]). The basic idea of the greedy algorithm finding the MCDS is to repeatedly select nodes for transmission, such that in each round a node whose transmission *covers* the largest number of uncovered nodes is selected. Thus, each transmission "removes" the largest possible number of nodes from the set of uncovered nodes and, eventually, results in covering the whole network with a minimized number of transmissions. However, such a centralized greedy algorithm violates the requirement (4) of Section 1. Consequently, we propose in this paper a particular distributed heuristic, which approximates the operation of the centralized greedy algorithm. The operation of this distributed greedy heuristics relies on each node "scheduling" its transmission based on the value of its RC – the larger the value of RC, the sooner the node is scheduled to transmit.

To explain the operation of the proposed distributed heuristic, we consider first the operation of the centralized greedy scheme. We start with the initial set of covered node $C=\{s_0\}$ and $Q=\emptyset$. 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 $C \setminus Q$ with the largest RC value to broadcast next; i.e., $\forall s \in (C \setminus Q)$, $RC(s) \le RC(s_1)$. After s_1 transmits, $C \leftarrow C \cup NE(s_1)$ and $Q \leftarrow Q \cup \{s_1\}$. Then, repeatedly, the next node with the largest RC is selected to transmit from the set $C \setminus Q$, 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 intuitively only tends to minimize the total number of transmissions during the operation of the algorithm. 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 values later, reducing the overall number of transmissions.

Although in [31], the authors discuss the inefficiency of a greedy scheme in finding MCDS in general graphs, the above centralized greedy heuristic finds on the average a rather close approximation of a MCDS in UDG as demonstrated in Section 6. 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. Surprisingly this path towards optimizing broadcast in wireless networks has not been investigated in previous works on the problem.

The following distributed *Time Sequence Scheme* (TSS) approximates the centralized greedy transmissions' order in time, by allowing nodes with larger RC values to transmit before nodes with smaller RC values.

TSS's blueprint is given as follows:

- Upon network deployment each node runs *Algorithm 1*. *Algorithm 1* generates sequence T of time-slots spanning the duration of the broadcast session.
- Source s_0 transmits message m and covers its neighbors.
- Each node i receiving m for the first time marks itself as covered and computes $RC(i)$. The local RC computation by nodes is discussed later in the paper.
- Next, node i runs *Algorithm 2* to schedule itself for later transmission time-slot T_b depending on $RC(i)$.
- If node i is scheduled to transmit in some time-slot T_b, i computes $RC(i)$ in the beginning of T_b. If the residual coverage of i has decreased (but is still positive) since the time-slot in which i has scheduled itself, i re-runs *Algorithm 2* and schedules itself for a new, later transmission time-slot. Else, still in T_b and prior to broadcast, i checks whether any of its 1-hop neighbors are scheduled to transmit within T_b as well. If more than one neighboring nodes are scheduled for T_b, the node with the largest RC transmits in T_b. The rest of the neighboring nodes schedule themselves to transmit in the next time-slot.

Next, we describe the details of the time sequence T's structure as generated by *Algorithm 1*. We also discuss the scheduling *Algorithm 2*, and how it exploits the structure of T to rank, prioritize and order nodes' transmissions in time.

5 Structure, Schedule, Sequence Schemes

The timing of nodes' transmissions is enforced by the particular structure of the time sequence T of time-slots. Each time-slot x, $0 < x \le |T|$, is associated with a specific RC threshold $\boldsymbol{\tau}_x$. Only nodes with RC values greater or equal to $\boldsymbol{\tau}_x$ are allowed to transmit in time-slot x.

5.1 Time Sequence Structure and Algorithm 1

What is the rationale for determining the RC threshold $\boldsymbol{\tau}_x$ at each time-slot x? Consider the following naïve scheme, which attempts to order the transmissions of the nodes, so that nodes with larger RC transmit first. Let T be a sequence

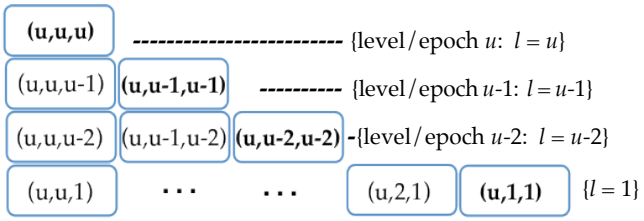

Fig. 2: Time Sequence *T*: output of Algorithm 1 with input *u*, arranged in epochs to implement threshold resets required for node scheduling.

of time-slots and assume $\boldsymbol{\tau}_x < \boldsymbol{\tau}_{x+k}$, $0 < k \le |T| - x$. Namely, each subsequent time-slot is associated with a *strictly* lower threshold value of *RC* than the previous time-slot's threshold. Upon receiving a broadcast message, node i marks itself as covered, determines $RC(i)$, and schedules itself to transmit in a future time-slot T_b. Since i can only schedule itself for a time-slot T_b such that $\boldsymbol{\tau}_b \le RC(i)$, the higher $RC(i)$ the earlier the scheduled T_b. That is, nodes with higher *RC* would tend to broadcast earlier than nodes with lower *RC*. This simple scheme *does not* take into account the fact that as scheduled nodes transmit, for instance in time-slot T_b, the set of *newly covered* nodes may contain nodes with *RC* values larger than $\boldsymbol{\tau}_b$. In other words, the time-slots following T_b cannot be used to time-order the transmissions of such newly covered nodes, since these nodes' *RC* values are greater than the thresholds of *all* time-slots following T_b.

To address this problem, we modify the time sequence T utilizing *Algorithm 1*, so that T contains repeated reordering of time-slots within *epochs* (also referred to as *levels*). Each *epoch* now contains a sequence of time-slots, and each subsequent time-slot within an epoch has *RC* threshold strictly lower than the previous time-slot's threshold. However, at the beginning of each epoch, the *RC* threshold is reset: the first time-slot in each epoch has *RC* threshold equal to the *RC* threshold $\boldsymbol{\tau}_1$ of the first time-slot in T. Suppose $\boldsymbol{\tau}_1 = u$. Fig. 2 shows the structure of the resulting time-sequence T, which is the output of *Algorithm 1* with input u.

For example, consider the case of $u = 4$. At the top level of T (level 4) only nodes with RC ≥ 4 are allowed to transmit. In the next level (level 3), first nodes with $RC \ge 4$ and then nodes with $RC \ge 3$ will be allowed to transmit. In level 2, first nodes with $RC \ge 4$, then nodes with $RC \ge 3$, and finally nodes with $RC \ge 2$ will transmit. In the last level, first nodes with $RC \ge 4$, then nodes with $RC \ge 3$, then nodes with $RC \ge 2$, and finally nodes with $RC \ge 1$ (all nodes with at least one uncovered neighbor) will be allowed to transmit.

To unambiguously label each time-slot, instead of using the threshold value only, we use a vector of three values: (*upper, middle, lower*). The *upper* is simply equal to u: the maximum value of the threshold, which is associated with the first time-slot in T. The *lower* is the number of the level/epoch and equals the *RC* threshold of this epoch. I.e. nodes with *RC* less than *lower* cannot transmit in this epoch. The *middle* is the threshold value of a time-slot. I.e. nodes with *RC* less than *middle* cannot transmit in this time-slot, but they may transmit later in this epoch, given their *RC* is greater than *lower*. Note that the values of *lower* and *middle* for all time-slots in T depend *only* on *upper*, which is equal to the parameter u. Hence, the number of timeslots $|T|$ in a

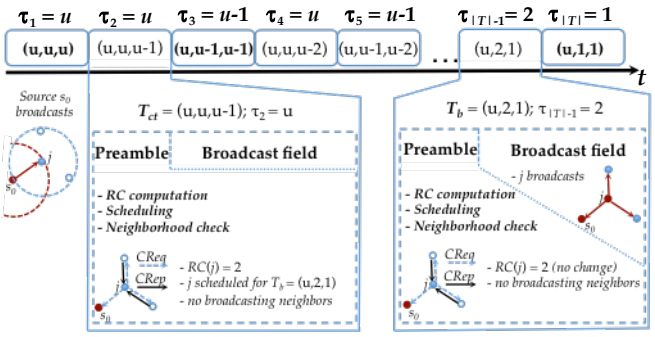

Fig. 3: The output of Algorithm 1 with input *u*; temporal format of timeslots and the temporal blueprint of TSS. Only nodes with RC greater or equal than a timeslot threshold τ_x can transmit in timeslot *x*. Timeslots in bold are called *edge* slots. Each timeslot comprises a *Preamble* followed by a *Broadcast Field*. Control messages are exchanged during the preamble of a timeslot. Nodes transmit a broadcast message during the broadcast field of a timeslot. T_{ct} is the current timeslot (*j* receives the broadcast message); T_b is a later timeslot for which *j* is scheduled using Algorithm 2, based on *RC*(*j*) computed during the *Preamble* of T_{ct}. In this case *RC*(*j*) has not changed during the time between T_{ct} and T_b; *j* broadcasts during the *Broadcast Field* of T_b.

broadcast session is a function of u. The parameter u is fixed and set up administratively and network-wide at the time of network deployment.

The value u should be judiciously chosen. A too small value of u does not allow separating 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 (i.e. larger $|T|$ and delay). Section 6.2.2 addresses the choice of appropriate parameter u in more detail, discussing its effect on the scheme's performance.

Fig. 3 shows the ordering of time-slots in time during the duration of a broadcast session. Each "edge" slot (in bold) demarcates the end of an epoch. The time-slot from the uppermost epoch occurs first, followed by the time-slots in the next, lower level. Also, in each level the time ordering of

Algorithm 1: Constructing the vector set T

Input: u
Output: ordered collection of vectors $T = \{T_u, T_{u-1}, \ldots, T_1\}$
Algorithm:
1: $T \leftarrow \varnothing$
2: $upper \leftarrow u$
3: $middle \leftarrow u$
4: $lower \leftarrow u$
5: $T_1 \leftarrow (\, upper, middle, lower \,)$
6: $T \leftarrow T_1$
7: **while** $middle > 1$ **do**
8: **if** $middle == lower$
9: $lower \leftarrow lower - 1$
10: $middle \leftarrow upper$
11: **else**
12: **if** $middle > lower$
13: $middle \leftarrow middle - 1$
14: $T_{next} \leftarrow (\, upper, middle, lower \,)$
15: $T \leftarrow T \cup T_{next}$

the time-slots is from left to right. The uppermost epoch (which contains a single time-slot) is associated with the largest threshold of *RC* (which is set to u), allowing transmissions only of nodes with *RC* value of at least u. The second epoch is associated with the *RC* threshold of u-1, allowing only nodes with *RC* values of at least u-1 to transmit. However, notice that, as the transmission in the first epoch might have revealed newly covered nodes with *RC* value larger than u, the second epoch contains two time-slots: the first allowing transmission of nodes with *RC* value of at least u, followed by a time-slot allowing transmission of nodes with *RC* value of at least u-1. This process continues until the last epoch (associated with *RC* threshold of 1 and containing u time-slots) allows ordered transmission of nodes with *RC* values of at least u down to nodes with *RC* values of at least 1.

5.2 Node Scheduling and Algorithm 2

Given the time-sequence (*TS*) structure described above, each node locally schedules its time of transmission, after receiving broadcast message m so that, overall, nodes with higher *RC* transmit earlier than nodes with lower *RC*. The *TS* serves as a common reference for all nodes.

The broadcast session begins when source node s_0 broadcasts message m. As m propagates throughout the network, any node j upon receiving m for the first time determines the *current time-slot* T_{ct} within the *TS*. This can be implemented as in 5.3.1 below.

The TS schemes' temporal flow proceeds as shown in Fig. 3. Node j determines its residual coverage $RC(j)$, as described in 5.3.2 below. After determining $RC(j)$, node j runs *Algorithm 2* to schedule its transmission for a future time-slot. Given T_{ct} and $RC(j)$, *Algorithm 2* schedules node j for a transmission timeslot, T_b, later in the broadcast session. T_b could be the next timeslot immediately after T_{ct} provided $RC(j)$ is large enough (i.e., $RC(j) > middle_{ct}$). Otherwise, *Algorithm 2* attempts to schedule node j for a time-slot at the current level, if $RC(j) \geq lower_{ct}$. If the current time-slot is an edge slot (see Fig. 2), *Algorithm 2* attempts the next level of the time-sequence. Else, node j is scheduled to transmit at a later, lower level. In general, the larger $RC(j)$, the earlier is the level and the earlier is the scheduled transmission time-slot T_b within that level. If $RC(j) = 0$ the node is not scheduled for transmission at all.

It is important to note that the value of $RC(j)$ can change between the time at which j had scheduled itself for transmission and the beginning of j's scheduled-for-transmission time-slot T_b, due to transmissions of other nodes or due to mobility. This may render j inadmissible in T_b. To avoid transmission in an incorrect time-slot, j re-computes its *RC* value prior transmitting in T_b and checks if it still can transmit in T_b. If so, j transmits the message in T_b. Else, it reschedules itself by employing *Algorithm 2* again with inputs $T_{ct} = T_b$ and the latest recomputed $RC(j)$.

5.3 TSS Protocol Details and Variants

In this section we discuss possible implementations of the Time Sequence Schemes based on the algorithms described above. We also provide details regarding implementations of procedures such as residual coverage computation, determining the current time-slot, and checking if a node has the largest *RC* in its neighborhood. We start by defining a general temporal format of a time-slot and define what procedures occur at different times in the time-slot.

Algorithm 2: Node j self-scheduling

Input: $RC(j)$; $T_{ct} = (u_{ct}, m_{ct}, l_{ct})$ // *vector associated with the current time-slot*
Output: transmission time-slot $T_b \leftrightarrow t_b$
Algorithm:
1: $rc \leftarrow RC_j$
2: $upper \leftarrow u_{ct}$
3: $middle \leftarrow m_{ct}$
4: $lower \leftarrow l_{ct}$
/* if $RC(j)$ is larger than the current value of *middle*, j transmits in the next time-slot */
5: **if** $rc > middle$
6: $T_b \leftarrow T_{ct+1}$
7: **else**
/* if $RC(j)$ is larger than *lower*, T_b is in the current level depending on the value of $RC(j)$ */
8: **if** $rc \leq middle$ **and** $rc \geq lower$
9: **if** (u_{ct}, m_{ct}, l_{ct}) **is** *edge_slot*
10: **if** $lower > 1$
11: $T_b \leftarrow$ (*upper*, *rc*, *lower* – 1)
12: **else**
13: $T_b \leftarrow$ (*upper*, *rc*, 1)
14: **else**
15: $T_b \leftarrow$ (*upper*, *rc*, *lower*)
/* if $RC(j)$ is even less than the value of *lower*, T_b is in a later level of the *TS*; the level depends on $RC(j)$ */
16: **else**
17: **if** $rc < lower$ **and** $rc \geq 1$
18: $T_b \leftarrow$ (*upper*, *rc*, *rc*)

As shown in Fig. 3, each time-slot consists of a *Preamble* part immediately followed by a *Broadcast Field* part. The *Broadcast Field* is fixed to the maximum duration needed to transmit the broadcast message depending on the maximum message size. The *Preamble* is used to transmit short control messages between adjacent nodes, and its duration is small compared to the *Broadcast Field* length of the time-slot. The timeslot duration t_D equals the duration of the *Preamble* added to the duration of the *Broadcast Field* and is known to all nodes in the network at deployment time.

1) Determining the Current Timeslot

The source node transmits the message m at the beginning of the *Broadcast Field* of the first time-slot in the *TS*. The initial transmission's timestamp is piggybacked by the broadcast message. Upon receiving m, node j computes the current time-slot T_{ct} by subtracting the *initial transmission timestamp* from j's current local time and dividing the difference by t_D to obtain the number of elapsed time-slots since the beginning of the broadcast session. Knowing the generic *TS* structure as per Fig. 2, a node is able to determine the vector of the current time-slot: $T_{ct} = (upper_{ct}, middle_{ct}, lower_{ct})$.

2) Residual Coverage Computation

The *RC* value of a node is needed prior to the node's scheduling (during the *Preamble* of timeslot T_{ct}) or rescheduling (during the *Preamble* of timeslot T_b), as noted in 5.2

above and shown in Fig. 3. This is done by a locally executed protocol – a version of a "Neighbor Discovery" protocol – where *Coverage Request* (*CReq*) and a *Coverage Reply* (*CRep*) messages are exchanged between neighboring nodes.

This simple protocol can be further improved in a variety of ways, but this is beyond the scope of this paper and is only discussed briefly in the discussion section of the paper. In Section 6.2, we simulate TSS over a full network stack and demonstrate that a standard 802.11b MAC layer readily handles all the control messages generated; as the network density increases, the performance of the TS schemes remain robust (Figures 7, 8 and 9).

3) The Naïve Time-Sequence Scheme

For clarity, we first summarize a basic *TS*-based broadcasting scheme. Upon network deployment, all nodes run *Algorithm 1* to construct the *TS*. After a node transmits the broadcast message, all of its previously uncovered neighbors that receive the message mark themselves as covered, compute their *RC* as in *2)*, and run *Algorithm 2* to schedule their transmission time-slots. Just before its scheduled time to transmit (during the *Preamble* of T_b), node *j* re-computes and updates its *RC*(*j*), as per Fig. 3. After the update, if the *RC*(*j*) has not decreased, *j* broadcasts during the *Broadcast Field* of T_b; if *RC*(*j*) has decreased (but *RC* > 0), the node determines its new time-slot assignment by re-running *Algorithm 2*. If, at any time, the computed *RC* value of a node equals 0, the node will never be scheduled for transmission in this broadcast session. We refer to this basic scheme as the *Naïve Time Sequence Scheme* (*NTSS*). As the name indicates, the NTSS possess some important deficiencies, which will be cured by the other variant TSS of the scheme presented next.

4) The Time-Sequence Scheme (TSS): Neighborhood Check

TSS operates as NTSS, but incorporates a 1-Hop neighborhood check within the scheduled-for-transmission timeslot T_b. To accommodate that, each timeslot's *Preamble* is split in two parts: *Preamble 1* and *Preamble 2*. Suppose node *j* is scheduled to transmit in T_b. During *Preamble 1*, *j* computes its *RC*, by sending *CReq* and receiving *CRep* packets. Next, *j* checks during *Preamble 2* of T_b whether any of its 1-hop neighbors are scheduled to transmit within T_b as well. This check *does not* necessitate any additional transmissions. Node *j* can determine whether a particular neighbor *i* is scheduled to transmit in T_b, if *j* has received the *CReq* message from *i* during *Preamble 1*. If more than one neighboring nodes are scheduled for T_b, the node with the largest *RC* is selected to transmit in T_b. In our implementation of TSS this is done as follows. Immediately after *Preamble 1*, node *j* picks a random time within *Preamble 2* to send a $RCPacket_j$ containing *RC*(*j*) (as found by *j* in *Preamble 1*) and *j*'s ID. If *j* receives $RCPacket_i$ from a neighboring node *i* and if *RC*(*i*) > *RC*(*j*), *j* does *not* send its $RCPacket_j$ and reschedules itself for broadcast to the next timeslot. Otherwise, *j* sends its $RCPacket_j$. Since the set B_N of all neighboring nodes broadcasting in the same timeslot T_b follows this protocol, it is easy to check that at the end of *Preamble 2* all but the node j^* with highest *RC* in B_N broadcasts in T_b. The nodes in $B_N \setminus \{j^*\}$ are rescheduled for the next time-slot T_{b+1}.

This 1-Hop check avoids redundant transmissions whereby neighboring nodes *i* and *j* broadcast *m* to the uncovered nodes in their respective neighborhoods and the shared neighbors of *i* and *j* receive *m* redundantly *both* from *i* and *j*. The latter severely degrades the performance of NTSS as we see in section 6.

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.

Fig. 4: In the first timeslot, the source node s_0 transmits the message *m*. Nodes *A*, *B*, *C*, and *D* receive *m*, 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* transmits 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* checks its *RC*. Since its *RC* remains the same, node *A* transmits the message. Nodes *E* and *K* become newly covered. Node *K* has two uncovered neighbors: *I* and *J*. *K* 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 transmits. All nodes are covered at this point. Hence, *C*, *D*, and *F* do not transmit.

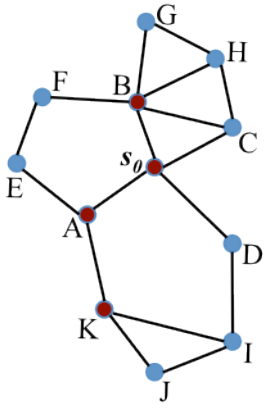

Fig. 5: A network topology example, where the TS-based scheme picks $\{s_0,B,A,K\}$ for broadcast and forms a MCDS of the particular network graph. All nodes are covered after the broadcast session completes.

5.4 Sample Execution of TSS

Consider the network of nodes shown on Fig. 5. Suppose

$u = 4$ and *Algorithm 1* constructs the *TS* as shown in Fig. 2. The sample run of TSS is shown in Fig. 4, with the resulting network coverage depicted in Fig. 5. Note that the *TS* structure allows *Algorithm 2* to give priority to nodes with higher *RC*, which could be covered and scheduled *later* during the broadcast session. For instance, because of its larger *RC* value, node *K* transmits before either of nodes *C*, *D*, or *F*, in spite of the fact that *C*, *D*, and *F* received the broadcast message *and* were scheduled earlier than *K*.

Ultimately, this eliminates the transmissions of nodes *C*, *D*, and *F*. Fig. 5 shows the network state after the broadcast session is completed. In this example, the MCDS (in red) equals the nodes selected by the TS scheme.

5.5 Formal Properties of the TS-based Schemes

Here we derive more formally the properties of the *TS* structure based on *Algorithm 1*. In 1), the correctness of the TS-based schemes is proved. *Algorithm 1*'s runtime is derived in 2). In 3), we discuss why along with the scheduling *Algorithm 2*, the TSS scheme emulates approximately the behavior of the centralized greedy algorithm.

1) Correctness of the TS-based Schemes

In the following proof of correctness, the network topology is assumed to be static[1] during the broadcast session and the MAC layer to be perfect. The network graph is assumed to be connected[2].

The performance evaluation of the TSS in the next section, demonstrates that even in the presence of high mobility and potential packet loss, the scheme achieves full or almost full network coverage.

Proposition 1: The *TS*-based schemes terminate in finite amount of time and guarantee full coverage of the network.

Proof: Per *Algorithm 2*, for all *TS* schemes, a node is not scheduled to transmit unless its *RC* is strictly greater than zero. Whenever a node *n* transmits, 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 transmit more than once during the execution of the algorithm. Since the number of nodes in the network is finite, the algorithm terminates in a finite number of steps.

Now, suppose that after the termination of a *TS*-based algorithm 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*. 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 received the message (note: *X* might be *S* and *Y* might be *D*). Therefore, since *Y* has not been covered, $RC(X) \geq 1$ at algorithm's termination. This is a contradiction. Per both TS schemes, node *X* computes $RC(X)$, after receiving the broadcast message. Per *Algorithm* 2 (lines 5-17), as long as $RC(X) > 0$, *X* is *always* scheduled to transmit in a later timeslot T_b.

Thus, the *TS* based schemes cover all the network nodes. □

2) Algorithm 1's Complexity

Let *T* be an ordered collection of vectors $\{T_x, T_{x-1}, \ldots, T_1\}$, where $T_k = (u_k, m_k, l_k)$, u_k, m_k, and $l_k \in \mathrm{N}^+$, and where the parameters *u*, *m*, and *l* are equal to the *upper*, *middle*, and *lower* values. Let *T* be the output of *Algorithm 1* with input *u*.

Proposition 2: The time complexity of *Algorithm 1* is $O(u^2)$; and the length of the generated time sequence is $x = |T| = u(u+1)/2$.

Proof: The output, *T*, of *Algorithm 1* can be arranged in an isosceles triangle with sides *u* as shown in Fig. 2. The triangle consists of *u* levels, where the last level (level 1) comprises *u* vectors. The number of vectors at the i^{th} level equals $1 + u - i$. The total number of vectors is then:

$$\sum_{i=1}^{u} i = u(u+1)/2 .$$

At each iteration of *Algorithm 1* there is exactly one vector generated. Hence there are $O(u^2)$ iterations and $x = |T| = u(u+1)/2$. □

As discussed in Section 6.2.2, the value of *u* is rather low ($u \approx 7$) in practice and does not depend on the network density. Hence, *Algorithm 1* is not computationally expensive and could be run on resource-constrained nodes.

3) Greedy Transmission Priority

Let $A = (S, I)$ be a set system with ground set $S = \{j: 1 \leq j \leq N\}$, the set of all network nodes. Each node *j* has weight equal to $RC(j)$ at a given time *t*. As time advances from time-slot to the next, the values $RC(j)$ may change. Let $I=\{S_1, S_2, \ldots, S_{|T|}\}$ be the collection of subsets of *S*, where $S_k=\{j: RC(j) \geq m_k \geq l_k \geq 1\}$, for given vector $T_k =(u_k, m_k, l_k) \in T$. Note that $|I| = |T|$.

Definition 5: A network node *j* is called *admissible* in vector T_k, iff $j \in S_k$.

Let $x = |T| = u(u+1)/2$. Every *x* consecutive time-slots (ordered in time as in Fig. 3) are mapped one-to-one to the vectors in the ordered collection *T* above. That is, each time-slot, t_k is uniquely associated with a vector in *T*: $t_1 \leftrightarrow T_x = (u, u, u)$, $t_2 \leftrightarrow T_{x-1} = (u, u, u\text{-}1)$, …, $t_x \leftrightarrow T_1 = (u,1,1)$.

Formally, a time-sequence *TS* is the ordered collection of the time-slots together with their corresponding vectors in *T*. A network node is admissible in time-slot t_k if it is admissible in vector T_{x+1-k}. This association "wraps around"; i.e., in general for $k \geq 1$, $t_k \leftrightarrow T_{d+1-k}$, where $d=\lceil k/x \rceil$.

(Revisited) Definition 1: A *broadcast session* consists of all the events, starting from the transmission of the message *m* by the source node and ending when the broadcast algorithm terminates after $|T|$ time-slots.

Broadcast Rule: At every time-slot t_k, a node considers transmitting only if it has not transmitted earlier in this broadcast session and if during the *Preamble* of t_k, it is admissible in T_{x+1-k}.

It is easy to verify that *Algorithm 2* complies with the *Broadcast Rule*. For instance, if $middle_{ct}$ of the current *TS* time-slot is lower than a 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).

Definition 6: $j \leq i$, $\forall i, j \in S$, iff $RC(j) \leq RC(i)$.

Definition 7: A *minimal admissible element* $min(T_k)$ in vector

[1] For an arbitrary mobility model even flooding cannot ensure full network coverage; similarly, this is true if package loss probability is positive.

[2] If the underlying graph of a static network is disconnected no broadcast algorithm can ensure full network coverage.

T_k is an *element* $j \in S$ such that $j \le i$ for all i admissible in vector T_k.

Consider the ordered collection T^q of vectors at level q, $q \in \{1, \ldots, u\}$, of T.

Definition 8: *inf* (T^q) is the smallest element among all minimal admissible elements in the vectors at level q.

Let L be the sequence (*inf* (T^u), *inf* (T^{u-1}),…, *inf* (T^1)). Note that the sequence L is decreasing, from definitions 6, 7, and 8 applied on the set system I described above.

Since L is decreasing, given the TS structure and scheduling algorithm, covered nodes with low *RC* values would not be able to transmit (i.e., be admissible) in earlier *higher levels* of the time sequence *TS*, but will potentially be admissible in later, *lower levels*. And reversely, only nodes with larger *RC* values would be admissible and be able to transmit in earlier *higher levels* of the time sequence. We next note that a similar observation holds for the time-slots *within* each level of the *TS*.

Let M be the sequence (*min* (T_i), *min* (T_{i-1}) …, *min* (T_1)), where vectors T_i, T_{i-1},…, T_1 are in collection T^q for a given level q. The sequence M is decreasing again from applying definitions 6, 7, and 8 on the set system I.

Since M is decreasing, within level q of the *TS*, nodes with smaller *RC* values would be admissible only in later timeslots at level q, and nodes with larger *RC* values would be admissible earlier at level q.

In summary, the structure of the *TS* (implemented via *Algorithm 1*) in conjunction with the *Broadcast Rule* (implemented via *Algorithm 2*) has admissibility property allowing for repetitive assignment of larger transmission priority to nodes with larger *RC* values 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 approximately.

6 Performance Evaluation

We investigate the performance, defined by a number of metrics, of various broadcast algorithms in four distinct network topology models. For a static network topology, we consider the case of a perfect MAC-layer (no packet loss due to collisions). Next, we implemented NTSS and TSS in full network stack, discrete event simulator (JiST/SWANS [26]), where packets may be lost at different network layers (e.g. due to collisions, noise etc.). Finally, we consider two types of realistic mobile models: one generating independent mobility patterns of the nodes; and another generating correlated (group) mobility patterns.

We compared the performance of the TS-based schemes against the most efficient schemes found in the technical literature to the best of our knowledge. In particular, we simulated the *RBS (*[25]). In [25] the authors show that RBS outperforms a few well-known broadcast algorithms such as the *Edge Forwarding* [13] algorithm, for example. We also simulate the more recent *CCS* ([45]). Another broadcast protocol we implement for comparison in this paper is *Bordercast:* the route discovery mechanism in the *Zone Routing Protocol* (*ZRP*) ([26]). *Bordercast* relies only on local topological information to select the nodes, which forward the broadcast message. Per ZRP a zone of node A in the network includes all nodes that are within k hops from A. Border nodes are those nodes in the zone whose minimum hop distance from A is exactly k. According to the *Bordercast* algorithm, the goal is to cover most efficiently only all of the border nodes in its zone.

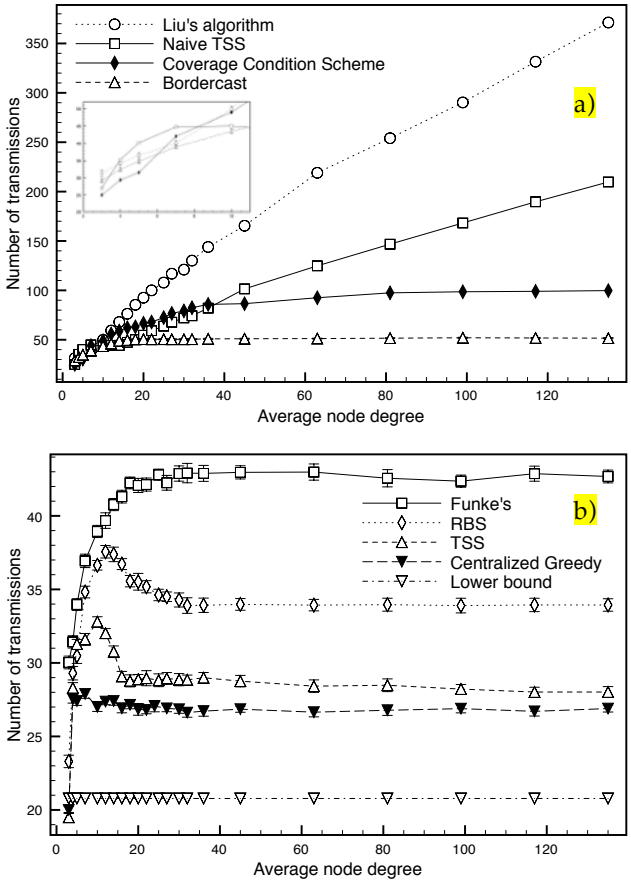

Fig. 6: The number of transmissions in a static collision free network (full coverage) of eight different broadcast algorithms.

To include an algorithm that constructs a *backbone* structure *prior* to the broadcast session, we selected *Funke's* algorithm [21], as it arguably provides one of the theoretically closest constant approximation ratio to MCDS: 6.94. Finally, for comparison, we also simulated Liu's algorithm [11] – a *node forwarding* algorithm that relies on 1-Hop positional information.

In all the experiments, unless otherwise indicated, the simulation area is a 200[m] x 200[m] square; the inner square area is of dimensions (200 - r)[m] x (200 - r)[m] to avoid edge effects. r [m] is the transmission radius of all nodes and is set to r = 25[m]. The number of nodes in the network varies from 200 up to 3000 nodes to investigate the schemes' performance at different node densities.

6.1 Static Network Topology with Lossless MAC

In this case the broadcast schemes performance reflect only the algorithmic efficiency, without the effect of collisions or noise in the environment.

The number of transmission (i.e., "transmission complexity"), equivalent to the number of broadcasting nodes during a broadcast session is a crucial metric for an efficient broadcast algorithm, and is investigated in Figures 6 (a) and

(b). The performance of the centralized Greedy algorithm is plotted in Fig. 6 (b) for comparison with the TSS. As expected TSS emulates well the centralized Greedy algorithm and both generate similar number of transmissions. Interestingly, without utilizing positional information TSS achieves about 15% lower number of transmissions compared to RBS and completes the broadcast session with approximately 67% fewer transmissions than CCS, while utilizing only 1-Hop topology knowledge. As an additional benchmark for transmission complexity, we calculated the number of transmissions using the Linear Hexagon Coverage technique, which provides a very close approximation to the minimum number of transmissions needed to cover the entire network area, assuming sufficient node density (Fig. 6(b)). The upper bound of transmission complexity in Fig. 6(a) 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*. More detailed discussion of these bounds can be found in the Appendix.

6.2 Full Network Stack Simulation

We implemented TSS and NTSS in the JiST/SWANS network simulator [26] to evaluate the effect of collisions, noise, fading, link asymmetries, etc. inherent in practical network scenarios.

The simulation parameters are given in Table 1. Figures 7, 8, and 9 show the performance of the TS schemes in terms of transmission complexity, delay, and fraction of network covered. Since some packets may be lost, full network coverage cannot be 100% guaranteed. However, the performance of the TSS scheme is rather robust as node density increases, despite more load at the MAC layer is present in the form of control messages. For comparison, we have implemented in JiST/SWANS the state-of-the-art RBS ([25]) and CCS ([45]) algorithms discussed above.

1) Transmission Complexity

The simulation results in terms of number of transmissions (Fig. 7), both for the TSS and NTSS algorithms, resemble their respective performance trends in the case of ideal MAC layer above. The number of transmissions is higher in Fig. 7 since the average transmission radius is about 22[m] here vs. 25[m] in the ideal MAC layer case. Also, some broadcast messages may be lost, requiring further retransmissions. TSS generates 25-30% and 53% fewer transmissions than RBS and CCS respectively. Notice however that,

TABLE 1
PARAMETERS OF THE JIST/SWANS SIMULATION

Simulator	JiST/SWANS v1.0.6
MAC Layer	IEEE 802.11b
Propagation Model	Free Space
Packet Size	64 – 7081 bytes
Radio Frequency	2.4GHz
Area	Square: 200[m]x200[m]
Transmission Range	~22[m]
Number of nodes	[150-1000]
Time-slot duration	TSS/NTSS: 100[ms]

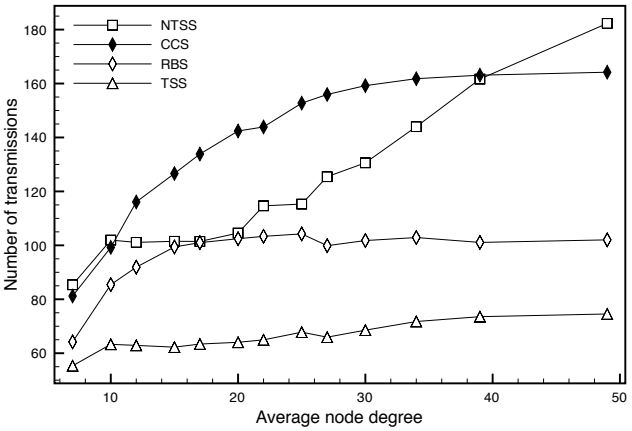

Fig. 7: Number of retransmissions (full network stack simulator).

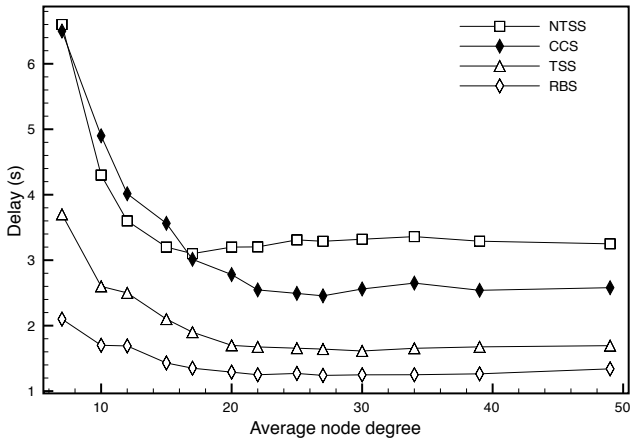

Fig. 8: Delay (full network stack simulator)

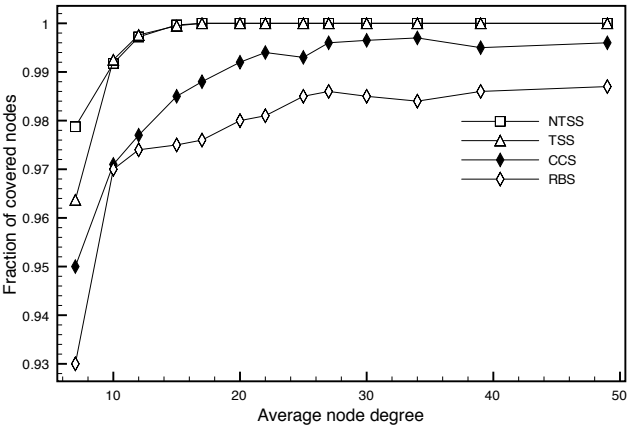

Fig. 9: Network coverage (full network stack simulator)

simultaneously, TSS maintains higher network coverage (Fig. 9): for instance, RBS leaves 25-30 uncovered nodes for networks of average node degree about 40.

2) Delay

The delay – the time needed to complete a broadcast session – is presented in Fig. 8. The delay performance of the TS-based schemes was obtained with parameter u set to the smallest value possible, so that the number of broadcast transmissions is still minimized. Decreasing u further would decrease the delay but would lead to more transmissions.

Setting the parameter u: On one hand, the number of timeslots would be lower if u is lower. However, smaller

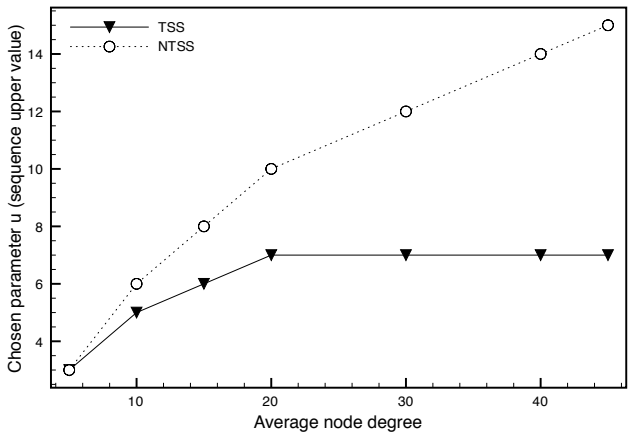

Fig. 10: The values of the parameter *u* determined via simulations for NTSS and TSS, so that the trade-off between number of re-broadcasts and delay is optimized.

number of timeslots allows for more nodes with different *RC's* being admissible and able to broadcast in the same timeslot. Hence, nodes' transmissions greedy prioritization could be coarser and less efficient. Larger values of *u* lead to potentially better prioritization of nodes' transmissions. However, the probability that a time-slot does not contain any transmissions is increased, leading to unutilized time and higher delay (quadratic in *u* as shown in Section 5.5.2). For instance, initial time-slots go by empty since their corresponding *middle* and *lower* values are too high compared to the nodes with highest *RC*. Thus, very few broadcasting nodes are admissible then.

Optimal value of parameter *u* is robust to network density variation: The optimal values of *u* for different network densities were determined via simulations (Fig. 10). We observed that beyond average node degree of 15, the optimal *u* values for TSS remain essentially constant with respect to the network density. Increasing *u* further, does not lead to notable decrease of transmission complexity. Thus the upper value could be fixed prior to network deployment, resulting in transmission complexity and delay that are close to the TSS optimal performance. The results for TSS hint that a good practice would be to set *u* to the average node degree for networks of node degrees up to 7. For networks of larger node degrees, *u* is largely density-independent and may remain fixed to 7. In our implementation, each node constructs a local copy of the *TS* that can be *extended* if at the last timeslot of the *TS*, the node schedules itself for transmission and its residual coverage is greater than 0. The delay of the broadcast (Fig. 8) accounts for the number of timeslots in the time sequence at the node with the *longest* local time sequence (i.e. the time elapsed between the first and last transmission).

The delay of TSS is comparable but slightly larger than that of the RBS algorithm, which however relies on GPS information and leaves larger number of nodes in the network uncovered (Fig. 9). The delay of the TSS scheme is about 50% lower compared to the CCS scheme, which utilizes only topology information.

3) Network Coverage

The network coverage (Fig. 9) of the TS-based schemes for lower node densities is ~96% since disconnections in the network are possible. However, at higher network densities the TS-schemes performance remains robust and they achieve virtually *full* coverage. The results indicate that the MAC layer readily copes with the number of control messages, as density increases. For all network densities TSS maintains higher coverage than both RBS and CCS.

TABLE 2
THE PARAMETERS OF GMMM

Velocity and position update interval	0.2 [s]
Velocity standard deviation	0.75 [m/s]
Velocity mean	1-20 [m/s]
Alpha	0.75

6.3 Dynamic Network Topology

In addition to TSS, we evaluated two state-of-the-art online, dynamic broadcast schemes (i.e. RBS, and CCS) under two mobility models within JiST/SWANS. Under the *Gaussian-Markov Mobility Model (GMMM)* ([33, 34]) each node follows independent realistic trajectory of movement. Under the *Self-Similar Least Action Model* (*SLAW*) model from [35], subsets of the network nodes follow correlated paths.

1) Individual Movement: Gauss-Markov Mobility Model

Per GMMM, time is split into time intervals (independent of the TS-based schemes time-slots). At the beginning of the k^{th} time interval, nodes' velocity is updated according to the following rule: $v[k]=\alpha v[k-1]+(1-\alpha)\overline{v}+(1-\alpha^2)^{0.5} z[k-1]$

Here, $v[k-1]$ is the velocity (speed and direction) of a node in the $[k-1]^{th}$, time interval; $z[k-1]$ is the observation of a Gaussian random variable at time interval $[k-1]$; $\overline{v}$ is the mean value of the velocity; and α is a parameter that determines the degree to which the current velocity at step k depends on the velocity at time interval $[k-1]$. As α approaches 1, nodes' motion becomes more constant; as α approaches 0 nodes' motion becomes more random. Table 2 summarizes the values of the parameters we used.

The number of transmissions and the corresponding achieved network coverage by the four algorithms at different average speeds is shown in Fig. 11a. The TSS performance is robust since each node checks its residual coverage at least twice (at the time a broadcast message is received and prior to transmission in the scheduled-for-transmission time-slot). Notice also that TSS is partially resilient against temporal disconnections of nodes from the network due to mobility.

More specifically, suppose in the Preamble of timeslot T_{ct}, j has a relatively smaller number of neighbors compared to the average node degree in the network. Respectively, j has higher chance of being temporarily disconnected from the network. In this case, it is likely that $RC(j)$ would also be lower in comparison to the *RC* of other nodes in the network. Then, node j is scheduled for a broadcast time-slot T_b that is late in the TS. This gives j more time to potentially move in areas with higher number of neighbors and *increase* $RC(j)$. Furthermore, even if meanwhile j becomes disconnected from the network at some timeslot between T_{ct} and T_b, that would not affect j's decision to transmit or not dur-

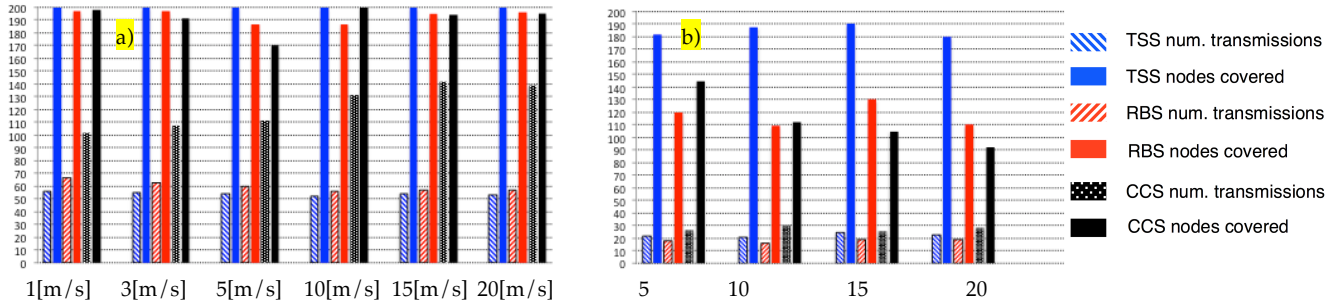

Fig. 13: Mobile network: a) Gaussian-Markov Mobility Model: $N = 200$ nodes. Num. transmitting and covered nodes (non-solid and solid color respectively); b) Self-similar Least Action Model: $N = 200$ nodes; x-axis: varying number of fractal waypoints {5,10,15, and 20}.

TABLE 3
THE PARAMETERS OF THE SLAW MOBILITY MODEL

Pause time	10-120[s]
Levy Exponent	1
β	1
Cluster range	27.5[m]
Distance weight	3
Nodes velocities	0.5 – 2 [m/s]
Waypoint ratio	5
Hurst Value	0.75
Number of Fractal Waypoints	Varied: [5-20]

ing T_b. Only $RC(j)$ in the Preamble of T_b determines whether j would transmit or not in T_b.

In contrast, suppose node j has larger than average number of neighbors during the Preamble of timeslot T_{ct}. In this case it is likely that $RC(j)$ would also be larger in comparison to the RC of other nodes in the network. Hence, j would be scheduled to transmit sooner in the TS, thus reducing the chance that the number of j's neighbors would decrease due to mobility.

As a result, TSS generates ~15% fewer transmissions than the position-aware RBS while achieving full network coverage. RBS leaves ~5% (8-10 nodes) uncovered. Furthermore, TSS generates 50-60% fewer transmissions than CCS, while the latter leaves up to 15% of the network nodes uncovered.

2) Group Movement: Self-Similar Least Action Model

Since independent individual node movement may be unrealistic for certain scenarios (e.g. hikers, tour groups, military platoon, etc.), we have also simulated a group mobility model, which is based on the *Self-Similar Least Action Model* (*SLAW*) model described in [35]. SLAW aims specifically at modeling realistically human mobility statistical features and has been verified to match a number of traces generated by real people's mobility. The simulation parameters (such as β, Levy Exponent, Waypoint Ratio, Hurst Value, and node velocities) used here were borrowed from [35]. Table 3 lists the parameters' values.

Figures 11 (b) depict the number of transmissions and the corresponding network coverage for different number of fractal waypoints governing the clustering in the network. TSS achieves at least 90% network coverage. The performance of the remaining broadcast algorithms degrades substantially since the network tends to be temporally partitioned. Here, each transmission covers larger number of nodes (due to clustering) and fewer transmissions are required to cover the network. TSS generates more transmissions than RBS, however it covers substantially larger fraction of the network nodes.

7 Related Work

The problem of efficient broadcasting has been extensively studied in the technical literature. The initial simple concept of flooding evolved into more sophisticated schemes through building optimal network subgraphs. Among the major shortcomings of pure flooding are the large transmission complexity and the notorious *broadcast storm* [1]. The *Scalable Broadcast Algorithm* [2] alleviates somewhat this problem utilizing 1-Hop neighbor information. All through, the main algorithmic challenge has been to reduce the number of transmissions needed to reach all network nodes.

More recent work on flooding offers a different and ingenious approach to circumvent the "broadcast storm" problem: Ferrari et al. ([47, 48]) design a system that harnesses constructive interference of concurrent neighboring transmissions and achieves remarkably low broadcast latency. Finding an approximation CDS to the minimum connected dominating (MCDS) set first, as done by the proposed TSS scheme, can significantly reduce the number of transmissions generated by such novel flooding-based algorithms. Constraining the flood to the nodes in the CDS could lead to significantly reduced energy expenditure, for instance, as observed in [50]. Consequently, these flooding-based approaches are orthogonal to the application of schemes such as TSS in finding approximations to the MCDS.

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 [3-9, 49]. The interest in probabilistic broadcasting schemes is due to their inherent low transmission overhead, low processing complexity, and high tolerance to frequent and rapid topological changes. Balancing these benefits, though, is the disadvantage of inability to guarantee full network coverage and, still, the presence of many redundant transmissions.

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 "mini-

mum forward-node set" strategy, and works such as [11]-[14] provide approximate solutions to this NP-hard problem. To avoid the transmission of the list of forwarding nodes along with the broadcast message, the 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]-[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. For instance, Funke's algorithm in [21] provably guarantees a 6.94-approximation ratio to the size of the MCDS. Nevertheless, as shown in [23], such backbone schemes 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], [25], and [45] offer initial solutions. However, [25] relies on positional information such as GPS, and GPS is not always feasible. The algorithm in [24] does not scale well for higher density networks. Finally, we show that the TSS scheme proposed in this paper is significantly more efficient in reducing the number of retransmissions (i.e. finding better approximation to the MCDS) than the CCS algorithm in [45].

Bounds on the size of $|\text{MCDS}|$ in unit disk graphs (UDG) have been studied extensively in the literature. Typically these bounds are given via ratio between the cardinalities of the Maximal Independent Set (MIS) and MCDS. Let $\mu = |\text{MCDS}|$ and $\chi = |\text{MIS}|$. Wu et al. demonstrate that $\chi \le 4\mu + 1$ [41]. Further improved band is derived in [42], where $\chi \le 3.43\mu + 4.82$. Finally, a tight (non-improvable) bound is derived by Vahdatpour et al.: $\chi \le 3\mu + 3$ ([43]).

A number of studies have attended to the theoretical complexity bounds of broadcast and related information dissemination mechanisms for topology (not area) bounded networks. For instance, [37, 38] provide such complexity bounds on gossiping, broadcast, and rumor spreading in such networks. Also, influential works by Peleg et al. in the spirit of [39] demonstrate important complexity lower bounds on broadcast in radius-2 radio networks: the broadcast procedure requires $\Omega(\log^2 n)$ transmissions.

8 Discussion

In this paper, we introduced a novel scheme, TSS, for broadcasting in wireless networks based on finding a distributed approximation of the wireless network MCDS. 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. The TSS scheme outperforms all other schemes with respect to the number of broadcast message transmissions, without requiring additional equipment, such as GPS. Furthermore, this performance is achieved with bounded latency, and is independent of network density.

To evaluate the effect of collisions, environment noise, attenuation, etc. on the performance of time sequence based schemes (NTSS and TSS), we implemented them in the JiST/SWANS full network stack simulator along with the state-of-the-art CCS and RBS broadcast algorithms. In this practical setting where control messages could be lost or simply erroneous due to link asymmetries, TSS improves on the performance of CCS and RBS with respect to number of transmissions and network coverage, while achieving comparable or better delay than the two algorithms.

Next, we considered the performance of TSS in mobile networks. We showed that TSS possesses network partitioning immunity and outperforms the other schemes with respect to network coverage in all mobility models simulated, achieving almost full coverage. This feature makes TSS one of the few broadcast alternatives to flooding in the context of mobile networks.

Thus, we can conclude that TSS is among the first broadcast algorithms, satisfying the efficient broadcast desiderata (1)-(5) in the introduction of this paper, and applying the TSS scheme in various network scenarios such as deployment of UAVs and sensor networks is a practical option.

Limitations: We note that with the current implementation of the TSS broadcast algorithm, its deployment in *extremely dense and dynamic networks* may not be practical. The number of control packets exchanged to compute RC in the latter settings may be rather large. An alternate strategy of estimating RC that reduces the number of control packets in such cases may be needed. For instance, instead of replying to a residual coverage request message 100% of the time, only a fraction of the nodes may reply (depending on the network density; the higher the density the smaller the fraction). Thus, a node may only have an estimate of the RC. We leave the implementation and study of such alternate RC computation for future work. The TSS performance though is robust both in mobile and static networks containing up to a few hundred nodes.

Even though theoretical bounds have been suggested before in the literature, we derived in the Appendix of the current study simple general upper and lower bounds on the number of retransmissions of any broadcast algorithm operating over a finite-area network. Not being the main contribution of the paper, we note that the benchmark Linear Hexagon Packaging lower bound on the number of broadcast retransmissions we obtain is rather close to the centralized Greedy performance and the TSS schemes for that metric. This indicates that Greedy may be a suitable candidate algorithm for theoretically low constant approximation algorithm for finding MCDS in dense UDGs. We leave the theoretical proof of this statement for future work.

References

[1] S.-Y. Ni, Y.-C. Tseng, Y. Chen, and J.P. Sheu, "The broadcast storm problem in a mobile ad hoc Networks", ACM MOBICOM, 1999.
[2] W. Peng, and X.C. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," ACM MOBIHOC, 2000.
[3] Z. J. Haas, J. Y. Halpern, and L. Li, "Gossip-based ad hoc routing", IEEE INFOCOM, 2002.
[4] Y. Sasson, D. Cavin, and A. Schiper, "Probabilistic broadcast for flooding in wireless mobile ad hoc networks," IEEE WCNC, 2003.
[5] D. Scott and A. Yasinsac, "Dynamic probabilistic retransmission in ad hoc networks," ICWN, 2004.
[6] J. Cartigny, D. Simplot, and J. Carle, "Stochastic flooding broadcast protocols in mobile wireless networks," Tech. Report LIFL, Univ. Lille1, 2003.

[7] J. Kim, D. J. Scott, and A. Yasinsac. "Probabilistic broadcasting based on coverage area and neighbor confirmation in mobile ad hoc networks," 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.
[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 INFOCOM, 2004.
[13] Y. Cai, K. Hua, and A. Phillips, "Leveraging 1-Hop neighborhood knowledge for efficient flooding in wireless ad hoc networks," IPCCC, 2005.
[14] G. Călinescu , I. I. Mandoiu , P. Wan , A. Z. Zelikovsky, "Selecting forwarding neighbors in wireless ad hoc networks," MONET, 2004.
[15] M. Sun, W. Feng, and T. Lai, "Broadcasting in ad hoc networks based on self-pruning," IEEE Globecom, 2001.
[16] Jie Wu , Fei Dai, "A generic broadcast protocol in ad hoc networks based on self-pruning," IPDPS, 2003.
[17] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," DIAL-M, 1999.
[18] A. Keshavarz-Haddad, V. Ribeiro, and R. Riedi, "DRB and DCCB: efficient and robust dynamic broadcast for ad hoc and sensor networks," 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, 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.
[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, 2006.
[22] B. Gao, Y. Yang, H. Ma, "An effective distributed approximation algorithm for constructing minimum connected dominating set in wireless ad hoc networks," CIT, 2004.
[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, 2008.
[24] I. Stoimenovic, "Comments and corrections to 'Dominating sets and neighbor elimination-based broadcast algorithms in wireless networks," IEEE Trans. Parallel and Distributed Sys., vol. 15, no. 11, 2004.
[25] Khabbazian, M., Bhargava, V. "Efficient broadcasting in mobile ad hoc networks," IEEE Trans. on Mobile Comp., vol. 8, no. 2, 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," IEEE CDC, 2006.
[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.
[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, vol. 26 no. 2, 1998.
[31] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," Algorithmica, vol. 20, 1998.
[32] B.N. Clark, C.J. Colbourn, and D.S. Johnson, "Unit disk graphs," Discrete Mathematics, vol. 86, 1990.
[33] B. Liang and Z. Haas, "Predictive distance-based mobility management for PCS Networks," 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 vol. 2, no. 5, 2002.
[35] K. Lee, S. Hong, S. J. Kim, I. Rhee, S. Chong, "SLAW: A New Mobility Model for Human Walks" IEEE INFOCOM 2009.
[36] F. Bai, N. Sadagopan, A. Helmy, "IMPORTANT: A framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks," IEEE INFOCOM, 2003.
[37] D. Liu , M. Prabhakaran," On randomized broadcasting and gossiping in radio networks," COCOON, 2002.
[38] D. Alistarh , S. Gilbert , R. Guerraoui , M. Zadimoghaddam, "How efficient can gossip be? (on the cost of resilient information exchange)," ICALP, 2010.
[39] N. Alon , A. Bar-Noy , N. Linial , D. Peleg, "A lower bound for radio broadcast," Journal of Computer and System Sciences, vol. 43 no. 2, 1991.
[40] Fejes Tóth, Uber die dichteste Kugellagerung. *Math. Zeit.* 48, 1943.
[41] W. Wu, H. Du, X. Jia, Y. Li and S. C.-H. Huang, "Minimum connected dominating sets and maximal independent sets in unit disk graphs," Theoretical Computer Science, 352(1-3):1-7, 2006.
[42] M. Li, P.-J. Wan, F. Frances Yao: Tighter approximation bounds for minimum CDS in wireless ad hoc networks, ISAAC 2009.
[43] A. Vahdatpour, F. Dabiri, M. Moazeni, and M. Sarrafzadeh, "Theoretical bound and practical analysis of connected dominating set in ad hoc and sensor networks", DISC 2008.
[44] S. Basagni, M. Mastrogiovanni, A. Panconesi, and C. Petrioli, "Localized protocols for ad hoc clustering and backbone formation: a performance comparison," IEEE Trans. Parallel and Distributed Sys., vol. 17, no. 4, 2006.
[45] M. Khabbazian, I. Blake and V. Bhargava, "Local Broadcast Algorithms in Wireless Ad Hoc Networks: Reducing the Number of Transmissions," IEEE Trans. on Mobile Computing, 2012.
[46] Y. Chen and J. L. Welch, "Location-based broadcasting for dense mobile ad hoc networks," ACM MSWiM, 2005.
[47] F. Ferrari, M. Zimmerling, L. Thiele, O. Saukh, "Efficient network flooding and time synchronization with Glossy," IPSN 2011.
[48] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. "Low-power wireless bus," ACM SenSys 2012.
[49] F. Stann, J. Heidemann, R. Shroff, and M. Z. Murtaza. "RBP: robust broadcast propagation in wireless networks," ACM SenSys 2006.
[50] J. Lu, K. Whitehouse, "Flash Flooding: Exploiting the Capture Effect for Rapid Flooding in Wireless Sensor Networks," IEEE INFOCOM, 2009.

Milen Nikolov received his undergraduate degree in Computer Science and Mathematics from State University of New York at Brockport, NY, and his M.Sc. degree from Cornell University, Ithaca, NY. He is currently a Ph.D. candidate conducting research in the department of electrical and computer engineering at Cornell University, where his interests are in the areas of wireless communications and information networks. He has recently been studying algorithms for networks with dynamic topologies, relay networks topology control, and collaborative communication in information networks.

Zygmunt J. Haas (F'07) received his Ph.D. degree from Stanford University, Stanford, CA, in 1988. He then joined the Network Research Department at AT&T Bell Laboratories, Holmdel, NJ. There, he pursued research on mobility management, wireless networks, wireless communications, fast protocols, optical networks, and optical switching. In 1995, he joined the faculty of the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY, where he is now a Professor, and where he directs the Wireless Networks Lab (WNL). Dr. Haas is an author of numerous technical papers and holds 18 patents. His interests include mobile and wireless communication and networks, biologically-inspired systems, and performance evaluation of large and complex systems. He has organized several workshops, delivered numerous tutorials at major IEEE and ACM conferences, and served as an Editor for several journals and magazines, including the IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE *Communications Magazine*, and Spinger's *Wireless Networks*. He has been a Guest Editor of several IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS issues and served as a Chair of the IEEE Technical Committee on Personal Communications. For more information, see http://wnl.ece.cornell.edu.