On Local Approximation of Minimum-Latency Broadcast Scheduling in 3D MANETs

Yilin Shen, Ying Xuan, My T. Thai
CISE Dept, University of Florida, Email: {yshen, yxuan, mythai}@cise.ufl.edu

***Abstract*—Minimum-latency broadcast scheduling problem has been a long-studied problem on the basis of conflict avoidance. However, no algorithms with theoretical performance guarantees have been proposed for this problem in 3D mobile ad hoc networks (MANETs), due to the significant hardness brought by various node mobilities in 3D space. As the first attempt in the literature to study this problem in 3D MANETs, we provide a localized approximation algorithm (LBS) with both theoretical and experimental guarantees with respect to time latency and message complexity.**

I. Introduction

Broadcast has been a fundamental mechanism of message dissemination in many applications, for example, the rapid assignment of orders to soldiers and vehicles plays a critical role in the battlefield. The intrinsic broadcasting nature of radio communications can either speed up the communications by transmitting the message to all neighbors or slow down the communications because of the conflicts with other transmissions. Intuitively, simple flooding [17] leads to the broadcast storm problem [17]. Thus, it is non-trivial and crucial to optimize the broadcast schedule with minimum time latency and no transmission conflicts [8], [15].

Unfortunately, Chlamtac and Kutten [3] established the NP-hardness of this problem in general networks, which denied the existence of a prompt optimal solution and followed by many approximation and heuristic solutions [8], [15], [4], [6], [7]. Later on, Parthasarathy *et al.* [12] further claimed that the problem remains NP-hard even in unit disk graphs.

As more mobile wireless services have been introduced since the last decade, *Mobile Ad Hoc Networks* (MANETs) becomes a dominant network type. Therefore, the problem of optimizing broadcast scheduling in MANETs exhibits more interest, yet meanwhile much more difficult.

The main challenges in this problem are three-fold: robustness toward various node mobilities, nodal self-organization and efficiency in 3D space, which are out of the capabilities of most existing solutions. Specifically, first of all, various node mobilities incur possible changes in network topologies all the time, which brings up the implementation complexities of many existing solutions [9], [10], [17]. Secondly, since a majority of the mobile devices nowadays are of low-power, which are incapable for energy-consuming calculations, and it is hard to control the MANETs in a centralized way, the *localized* ([9], [10]) instead of *centralized* ([8], [15]) algorithms are more preferable such that each node can be self-organizing. Nevertheless, most existing algorithms for MANETs with theoretical performance guarantees in the literature are centralized [8], [15], which thus limited their application to general MANETs topologies. Last but not least, most existing attempts focused on the 2D network models, whereas a majority of real applications are in 3D space. An important application of MANET is the vehicular ad-hoc network (VANET), which takes the moving cars as nodes to create a mobile network [1]. Due to the changes in elevation of the road faces and various car manufactures, VANET turns out a 3D MANET. From 2D to 3D, the broadcast scheduling problem poses larger intractability and it is non-trivial to adapt existing 2D solutions to the 3D problem without a significant decay in the performance.

To this end, in this first attempt to study the minimum time-latency broadcast scheduling problem in 3D MANETs, we propose a novel localized algorithm with both approximation theoretical and experimental performance guarantees to overcome the three difficulties mentioned above. The main idea of our approach is to handle the mobility by letting each node determine its schedule at the beginning of each period of time-slots, which is referred to as intervals. The target is to transmit the message from all nodes already having it to all their neighbors. To do this, each node exchanges indicator messages with its neighbors by taking advantage of the space tiling and coloring technique and the idea of independent subset. In a big picture, our approach not only handles the nodal mobilities by transmitting the message to all neighbors in each interval but also shorten the broadcast latency due to the conflict-avoidance schedules.

The rest of paper is organized as follows. We introduce the problem definition and network models in Section II. In section III, we present the tiling and coloring technique which will be used throughout the whole paper. The localized approximation algorithm in 3D MANETs is proposed in Section IV. Section V illustrates the experimental evaluations and the whole paper is concluded in Section VI.

II. Definition and Network Models

In a MANET, a source node s holds message m and wants to broadcast m to all the other nodes (one-to-all broadcast). A node can send m only after it receives m in a conflict-aware transmission. A schedule is referred to as the time a node i receives or transmits m. The latency of a broadcast is the period between the beginning of broadcast and the first time when all nodes receive m. Our goal is to find a conflict-aware broadcast schedule with minimum latency for one-to-all broadcast. We assume that time is discrete and each message transmission takes M unit time-slots. A broadcast schedule is called *conflict-aware* if the transmission can avoid the following two types of conflicts:

Receiver Collision: A receiver cannot successfully receive a message if it is within the *transmission range* of two or more senders.

Receiver Interference: A receiver cannot successfully receive a message if it is within the *interference range* of two or more senders.

In our MANET model, all nodes V move randomly in synchronous discrete time-slots. Also a node can join or leave the network at any time-slot. For each node $v_i \in V$, its neighbors, called $N(v_i)$, can change at various time-slots. Each node knows it location at any time-slot by being equipped with a functional GPS or using some existing localization approaches [11]. It is easy to see that the model includes most existing MANET models, i.e. reference point group model, random walk model, random way point model, etc. We assume that the network is connected in each time-slot and each node will not move during and after determining its schedule until the end of its transmission. In a mobile network, each node v_i has the same transmission range and interference range r. Node v_j is a neighbor of v_i if and only if the distance between v_i and v_j is no larger than r, i.e. $dist(v_i, v_j) \leq r$. Topologically, the network can be represented by a unit ball graph.

III. Tiling and Coloring Technique

We use a local *Tiling and Coloring* technique [16] to color the nodes. For each subset of nodes V_i of the same color i, a subset nodes G_i of V_i are further selected for two purposes: (1) all nodes in each G_i can transmit simultaneously without any conflicts; (2) all other nodes $V \setminus (\cup_i G_i)$ in the whole network can receive the message directly from at least one node in the union of all G_i.

The technique is two-fold: on one hand, tiling technique is to classify nodes into different cells by partitioning the space into cells and associating each node with a cell according to its location such that all nodes in each cell can communicate with each other directly. On the other hand, the cells are further clustered into groups such that any node in a specific cell of a group can transmit at the same time of some other node in the corresponding cell of any other groups, where the correspondence of cells among different groups can be achieved by coloring the nodes in the cells of each group using a predefined coloring pattern. In this case, G_i consists of the union of at most one specific node of color i in each group such that both purposes mentioned above can be satisfied.

In particular, we tile the 3D space using truncated octahedrons of side $r/\sqrt{10}$ as cells and each truncated octahedron is bottom-closed and up-open. Clearly, since such truncated octahedron has a diameter equal to r [2] and all nodes in network have transmission range r, the nodes in each truncated octahedron can communicate with each other directly. Then, according to the previous work [16], there exists a way to cluster the truncated octahedrons into groups locally where each group has $\left\lceil \left(\sqrt{\frac{3}{5}}\frac{2r}{r}+1\right)\right\rceil^2 \cdot \left\lceil \frac{2r}{r}\frac{2}{\sqrt{5}}+1\right\rceil = 27$ truncated octahedrons and has the same predefined coloring pattern such that any two nodes in different truncated octahedrons to_1 and to_2 of the same color have the Euclidean distance $dist(to_1, to_2) > 2r$. Fig. 1 illustrates two groups of truncated octahedrons. For simplicity, we define $\mathcal{C}_3 = 27$ to be the sufficient number of colors and $\mathcal{P}$ as the predefined coloring pattern.

IV. **LBS** Approach in 3D MANETs

This section proposes a localized broadcast scheduling algorithm (LBS) to determine an *on-the-fly* broadcast schedule.

In LBS, we define *an interval* to be a synchronized period of time-slots in which the set of nodes with m transmit it to all their neighbors successfully. The set of nodes having m is referred to as *source nodes* and the number of source nodes is monotonously increasing with the intervals since at least one node will receive m in each interval. And m is broadcast interval by interval until all nodes receive it.

The idea of LBS is to handle the mobility by determining different schedules for each node at the beginning of each interval using localized interval broadcast scheduling algorithm (LIBS). When some nodes move within one interval, the schedules of themselves and their neighbors can further be adjusted adaptively at the beginning of each time-slot in the interval using adaptively localized time-slot broadcast scheduling algorithm (ALTBS). Fig. 2 illustrates an example of a whole broadcast process by using LBS algorithm.

For each node v, the schedule in one interval, spanning l time-slots, is associated with a time-slot sequence $\mathcal{T}^v = \langle T_1^v, T_2^v, \ldots, T_l^v \rangle$ where each T_i^v is denoted as a binary variable: $T_i^v = 1$ if node v transmits m in time-slot i and $T_i^v = 0$ otherwise. We later prove that the length l of an interval is a constant in Lemma 3. The detailed LBS approach is shown in Algorithm 1. Note that a node does nothing after the termination of broadcast.

Algorithm 1: LBS in Node v

```
// Iterate all intervals
while true do
    // Operations in one interval
    if $v$ does not have $m$ then
        Notify all its neighbors;
    end
    else
        if $v$ is not notified then
            Wait until the beginning of next interval;
            Continue;
        end
    end
    Reset all time-slots in the time sequence $\mathcal{T}^v$ to be 0;
    Determine its schedule $\mathcal{T}^v$ using LIBS algorithm;
    for each time-slot $i$ do
        Adjust its schedule using ALTBS algorithm;
        Transmit $m$ according to $\mathcal{T}_i^v$;
    end
end
```

By applying the tiling and coloring technique as described in Section III, we begin with the LIBS algorithm as follows.

A. LIBS Algorithm

We first classify all nodes into three types:

- *Source Nodes*: The nodes which have received m;
- *Candidate Nodes*: The nodes which are neighbors of some source node and have not received m;
- *Spare Nodes*: All other nodes.

The idea of LIBS is to divide the time sequence $\mathcal{T}^v$ into two subsequences $\langle \mathcal{T}_1^v, \mathcal{T}_2^v \rangle$, where $\mathcal{T}_1^v = \langle T_1^v, \ldots, T_\tau^v \rangle$ of length τ and $\mathcal{T}_2^v = \langle T_{\tau+1}^v, \ldots, T_l^v \rangle$ of length $l - \tau$, and set the schedule by determining the transmission of each time-slot in the time-slot subsequences respectively. $\mathcal{T}_1^v$ and $\mathcal{T}_2^v$ are used by *connectors* and *dominators* to transmit respectively, where *dominators* refer to the set of nodes which is a maximal independent set of candidate nodes and *connectors* refer to the

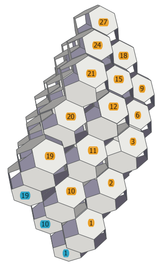


Fig. 1. Two Sample Truncated Octahedrons Tiling in 3D Space: The number of labels is the color labeled to all nodes located in each truncated octahedron.

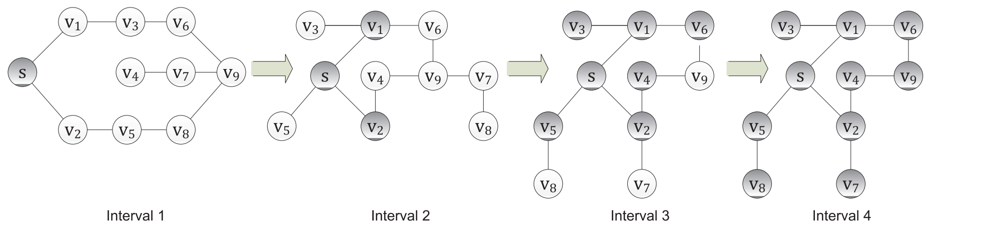


Fig. 2. An Example of Broadcasting: There are one source node s and 9 other nodes. A node i is labeled grey if it has received m and white otherwise. In interval 1, only node s is scheduled to transmit m to its neighbors v_1 and v_2. In interval 2, node v_3, v_4, v_5 and v_6 move to be neighbors of node v_1, v_2 and s, which are scheduled to transmit m to them all. In interval 3, node v_1, v_2, v_4 and v_5 are scheduled to transmit m to their new neighbors v_7, v_9 and v_8. At the beginning of interval 4, all nodes have received m and the broadcast terminates.

subset of source nodes which consists of a minimal covering of dominators. Note that the set of connectors is a subset of source nodes and the set of dominators is a subset of candidate nodes. $\mathcal{T}_2^v$ is determined before $\mathcal{T}_1^v$ in LIBS algorithm. Note that the division of $\mathcal{T}^v$ lends a hand to better utilize the properties of dominators and tiling techniques respectively.

In detail, each node v runs LIBS algorithm to determine its schedule as follows: Initially, all its elements in $\mathcal{T}^v$ are 0 and v regards itself as a spare node. v first colors itself as C_v using the tiling and coloring technique mentioned above. Then, v checks if it has received m. If so, it sets itself to be a source node and sends a query message E to all of its neighbors, where the query message E indicates that v has received m. Otherwise, it keeps idle until it receives a query message E from some of its neighbors and sets itself to be a candidate node. If v is a candidate node, it further determines if it is a dominator by using the localized maximal independent set algorithm (LMIS), which is proposed by Schneider *et al.* [13] as a localized algorithm to find the maximal independent set within time $O(\log^* |V|)$ on growth-bounded graphs (i.e. unit ball graphs is a subset of growth-bounded graphs). If so, the time-slot subsequence $\mathcal{T}_2^v$ is determined by its color label C_v and the predefined coloring pattern $\mathcal{P}$. Then v runs localized iterative minimal covering (LIMC) (Algorithm 3) to determine if it is a connector, and therefore set its time-slot subsequence $\mathcal{T}_1^v$. Note that in each interval, the broadcast schedules can be determined just by dominators and connectors according to their time-slot sequence $\mathcal{T}$. LIBS is shown in Algorithm 2.

Before introducing its subroutine LIMC, we first define some indicator messages with respect to each node v: $\mathfrak{C}$: v is covered; D: v is a dominator; $\mathfrak{D}$: v is dominated; S: v is a source node; T: v requires its neighbors to transmit; N: v does nothing.

LIMC algorithm: *The idea of LIMC is to set the time-slots in $\mathcal{T}_1$ iteratively such that all dominators can receive m at the end of LIMC and, in each iteration, a subset of dominators can receive m successfully without any conflicts by scheduling the transmission of the only one source node incident to each dominator.* In iteration i, each node first runs LMC to determine if it is a connector. Then for a connector v, it is scheduled to transmit by setting its time-slot $\mathcal{T}_i^u$ to 1 if there exists at least one dominator u incident to v but not incident to any other connectors. As shown in Lemma 2, our LIMC algorithm determines the transmission schedule of these

Algorithm 2: LIBS in Node v

```
// Initial Steps
Initialize the time-slot sequence $\mathcal{T}^v$;
Set $v$ to be a spare node;
Determine its color label $C_v$ using the tiling and coloring technology;
// Determine time-slot subsequence $\mathcal{T}_2^v$
if $v$ has $m$ then
    Set itself to be a source node;
    Send a query message $E$ to its neighbors;
end
else
    if $v$ receives at least one $E \wedge v$ is a spare node then
        Set itself to be a candidate node;
        Determine if $v$ is a dominator using LMIS algorithm;
        if $v$ is a dominator then
            Set $\mathcal{T}_2^v$ using its color label $C_v$ and the predefined coloring
            pattern $\mathcal{P}$;
        end
    end
end
// Determine time-slot subsequence $\mathcal{T}_1^v$
Set $\mathcal{T}_1^v$ using LIMC;
Terminate LIBS in node $v$.
```

dominators within constant time-slots by taking advantage of their properties as independent subsets. The detailed LIMC algorithm is shown in Algorithm 3.

Algorithm 3: LIMC in Node v

```
for $i \leftarrow 1$ to $\tau$ do
    Determine if $v$ is a connector using LMC algorithm;
    // A candidate node requires to receive $m$
    if $v$ is a candidate node then
        Send an indicator message $C$ to $N(v)$;
        Wait until receiving all replies from $N(v)$;
        if $v$ receives only one indicator message $S$ from node $u$ then
            Send an indicator message $T$ to the node $u$;
            Send an indicator message $N$ to all other $N(v)$;
            Set itself to be a spare node;
        end
    end
    // A source node set its transmit schedule according
       to the requirements from candidate nodes
    if $v$ receives at least one indicator message $C$ then
        if $v$ is a connector then
            Reply an indicator message $S$ to all sending nodes;
            Wait until receiving all replies from $N(v)$;
            Set $T_i^v$ to 1 true if $v$ receives an indicator message $T$;
            Terminate the algorithm in node $v$;
        end
        else
            Reply an indicator message $N$ to all senders;
        end
    end
end
Terminate LIMC in node $v$.
```

LMC algorithm: *As a subroutine in LIMC, LMC algorithm is proposed to determine if a source node is a connector.* (Each connector further schedules to transmit as described above in

LIMC.) The set of connectors is referred to as a minimal subset of source nodes covering all dominators. A node is called *covered* if at least one of its neighbors is selected as a connector.

In LMC, we first define some more notations: let $\kappa_v(\mathcal{M})$ be the number of neighbors sending a particular indicator message $\mathcal{M}$ to node v, where $\mathcal{M}$ can be any type of indicator message mentioned above. Let D_v and C_v be the number of valid replies for a dominator v and a connector v respectively; let $|N(v)|$ be the number of neighbors of node v.

The idea of LMC algorithm is to greedily select out the connectors until all dominators are covered via the message exchanges among spare nodes, dominators and source nodes. In detail, for each node v, we consider the following distinct cases corresponding to the above three types (Algorithm 4):

(1) If v is a spare node, it will send an indicator message N to its neighbors and simply terminate.

(2) If v is a dominator, it will send an indicator message D to its neighbors and then determine D_v to be $|N(v)| - \kappa_v(D) - \kappa_v(N)$ after receiving indicator messages from all neighbors. Then v will determine to be covered if it receives at least one indicator message C. If v is covered, it further sends an indicator message $\mathfrak{C}$ to its neighbors indicating that it has been covered. Otherwise, it will send an indicator message N to its neighbors and update D_v to be $D_v - \kappa_v(C) - \kappa_v(N)$ to decrease D_v by ignoring the set of connectors. The loop will not stop until v is covered. Notice that there are at most 11 iterations since any node in 3D space has at most 11 independent neighbors.

(3) If v is a source node, it will send an message S to its neighbors and set C_v to $|N(v)| - \kappa_v(S) - \kappa_v(N)$ after receiving indicator messages from all its neighbors. Then it will set a counter t to 11 initially. It will set itself to be a connector if C_v is no less than t. If v is selected as a connector, it will send an indicator message C to it neighbors. Otherwise, t will be decremented by 1 and C_v will be decremented by $\kappa_v(\mathfrak{C})$ after receiving all C_v indicator messages. The loop will not stop until there is no request or v is set as a connector. Again, there are at most 11 iterations.

Algorithm 4: LMC in Node v

```
// LMC in a spare node
if $v$ is a spare node then
    Send an indicator message $N$ to $N(v)$;
    Terminate LMC in node $v$;
end
// LMC in a dominator
else if $v$ is a dominator then
    Send an indicator message $D$ to $N(v)$;
    Wait until receiving all indicator messages from $N(v)$;
    $D_v \leftarrow |N(v)| - \kappa_v(D) - \kappa_v(N)$;
    while $v$ is not covered do
        if $v$ receives at least one indicator message $C$ then
            Send an indicator message $\mathfrak{C}$ to $N(v)$;
            Terminate LMC in node $v$;
        end
        else
            Send an indicator message $N$ to $N(v)$;
            $D_v \leftarrow D_v - \kappa_v(C) - \kappa_v(N)$;
        end
    end
end
// LMC in a source node
else if $v$ is a source node then
    Send an indicator message $S$ to $N(v)$;
    Wait until receiving all indicator messages from $N(v)$;
    $C_v \leftarrow |N(v)| - \kappa_v(S) - \kappa_v(N)$;
    $t \leftarrow 11$;
    while $t > 0$ do
        if $C_v \geq t$ then
            Set $v$ as a connector;
            Send an indicator message $C$ to $N(v)$;
            Terminate LMC in node $v$;
        end
        else
            Wait until receiving all $C_v$ indicator messages;
            $C_v \leftarrow C_v - \kappa_v(\mathfrak{C})$;
            $t \leftarrow t - 1$;
        end
    end
    Send an indicator message $N$ to $N(v)$;
    Terminate LMC in node $v$.
end
```

TABLE I Mobility Events

Event	Description
candidate node v arrival	at least 1 source node receives L from v
candidate node v departure	all nodes receives L from v do not have m
source node v arrival	at least 1 spare node receives L from v
link (u, v) arrival	v receives L from a new neighbor node u
link (u, v) departure	v does not receive an ACK from its previous neighbor node v after sending L to all its neighbors

[a] Note that when a candidate node v has received m and v moves, this is an event of source node arrival rather than candidate node departure.
[b] Link arrival and departure only happen among the union of source nodes and candidate nodes.

B. ALTBS Algorithm

After determining the schedule using LIBS algorithm at the beginning of one interval, *ALTBS will be further applied at the beginning of each time-slot in this interval to adjust the schedule properly according to the nodal mobilities*. Their mobilities lead to five events, i.e. *candidate node arrival, candidate node departure, source node arrival, link arrival and link departure*, as shown in TABLE I.

In ALTBS, when a candidate node or a source node moves at the beginning of some time-slot, it will send an indicator message L to its neighbors indicating the change of its location. We call the set of new candidate nodes as *Fresh Candidate Nodes*.

Candidate Node v Arrival: when a candidate node v arrives, it will first send an indicator message $\mathfrak{D}$ to its neighbors to ask if it is dominated. (A node v is called dominated when at least one of its neighbor nodes is a dominator.) If not, it will send an indicator message T to its neighbors to require the transmission. Then if the current time-slot i is less than τ and there is no source nodes incident to v scheduled to transmit between time-slot i and τ, an arbitrary source node u connect to v will be scheduled to transmit m in time-slot i.

Candidate Node v Departure: same schedule.

Source Node v Arrival: when a source node v arrives, the spare nodes will set themselves to be fresh candidate nodes if they receive L from v. Then each fresh candidate node u will update the schedules as the event of candidate node u arrival.

Link (u, v) Arrival: *only the arrival of a link between two dominators will affect the schedule.* Hence, the two nodes u and v will check if both of them are dominators. If so, they will redetermine their color labels C_u and C_v using their locations. Then node u and v will exchange their color labels to check if they are the same. If so, the one having less number of neighbors will not transmit if both of them have not transmitted. Otherwise, the schedule keeps the same.

Link (u, v) Departure: when a link (u, v) leaves, there are two cases which will affect the schedule:

(1) *The departure of a link between a connector and a dominator:* Without loss of generality, assume node v is a dominator. If node v has not received m, it will send an indicator message T to its neighbors. If no source nodes incident to v are scheduled to transmit between time-slot i and τ, an arbitrary source node u connecting v will be scheduled to transmit m in this time-slot i.

(2) *The departure of a link between a dominator and a candidate node but not a dominator:* Again without loss of generality, assume node v is a candidate node. Node v will send an indicator message $\mathfrak{D}$ to its neighbors to ask whether it is dominated or not. If not, v is regarded as a new arrival candidate node and update the schedules as the event of candidate node v arrival.

Remarks: Here we briefly introduce the way to handle the potential interference induced by indicator messages. Since each node has the same transmission range and interference range, a node can regard itself to receive a specific indicator message according to the synchronized time-slot since the indicator messages are only required to transmit and receive within the particular time-slots.

C. Theoretical Analysis

Lemma 1. *There is at most one dominator in each truncated octahedron.*

Proof: Assume that there are at least 2 dominators u, v in some truncated octahedron, in which each pair of nodes can transmit directly in one truncated octahedron since each truncated octahedron has the side $r/\sqrt{10}$. This leads to a contradiction according to the concept of independent subset. ■

Lemma 2. *In LIMC algorithm, $\tau \leq 11$.*

Proof: In each time-slot, since the set of connectors are the minimal covering of all dominators, for an arbitrary connector, there is at least one dominator incident to it but not incident to other connectors. This implies that the set of dominators scheduled to receive in different time-slots is disjoint. According to [5], a node v has at most 11 independent neighbors in unit ball graphs, that is, a connector v has at most 11 incident dominators and $\tau \leq 11$. ■

Theorem 1. *The message complexity in* **LBS** *for each node is $O(\imath)$, where $\imath$ is the number of intervals.*

Proof: In **LBS**, the message complexity is determined by LIBS and the number of intervals. In each interval, $O(1)$ messages are sufficient in LIBS according to [13]. In LIMC, according to Lemma 2, τ is upper bounded by a constant. Also in LMC, there are at most constant number of iterations. Therefore, each node at most exchanges $O(1)$ messages with its neighbor in each interval in LIMC. Thus, The message complexity in **LBS** is $O(\imath)$ for each node. ■

Theorem 2 (Correctness)**.** *All node can successfully receive m in certain time-slot during the whole broadcast process.*

Proof: We use the mathematical induction to show the proof. According to the assumption, the nodes will not move until the end of transmission if it has been scheduled to transmit. We show that at least one node will receive m in each interval.

Basis: in the first interval, since the whole network is connected, there is at least one node connecting the source node will receive m after the transmission of source node s.

Inductive step: we show that if $|V_k|$ nodes receive m after k intervals, at least $|V_k| + 1$ nodes will receive m after interval $k + 1$. The proof follows from the connectivity of the whole network in any time-slot. With this in mind, there exists at least one candidate node at the beginning of each interval. In other words, there is at least one dominator in each interval. Since a dominator is scheduled to transmit in such interval, it will not move until the end of its transmission according to the assumption. As a result, the dominators can successfully receive m after this interval. The proof is complete. ■

Lemma 3. *Within each interval, the length of time sequence l is at most $38 + o_M(1)$.*

Proof: As described in LIBS, the number of time-slots is determined by the transmission of connectors and dominators, i.e. $\mathcal{T}_1$ and $\mathcal{T}_2$. According to Lemma 2, τ iterations are sufficient for the transmission of connectors. For dominators, since the schedule is determined by the number of color labels. Since there are $O(1)$ indicator messages as shown in Lemma 1 and each indicator message is only required to transmit and receive within one time-slot, $\tau + \mathcal{C}_3(2) + o_M(1) = 11 + 27 + o_M(1) = 38 + o_M(1)$ is sufficient for the length of time sequence. ■

The following theorem shows that **LBS** is $38 + o_M(1)$ approximation algorithm with high probability under some constraint of network size.

Lemma 4. *The approximation ratio of* **LBS** *is $38 + o_M(1)$ if the network topology keeps the same within one interval, where $\imath$ is the number of intervals and M is the number of time-slots needed to transmit m.*

Proof: Since the number of intervals $\imath$ is a trivial lower bound of time latency in MANETs if the network topology is invariant within one interval as all source nodes transmit m to all their neighbors in each interval, the proof follows from Lemma 3. ■

Lemma 5. *For any two nodes u and v, the link (u, v) is invariant within one interval with probability $\frac{76S \max v}{\min w \min r}$, where w is the data transfer rate, r is the transmission range of each node, S is the size of broadcast m and v is the moving velocity of each node.*

Proof: Within one interval, the distance between u and v changes in each time-slot is at most $\frac{2S \max v}{\min w}$. Since there are at most 38 time-slots according to Lemma 3, the probability that the link between u and v changes is $\frac{76S \max v}{\min w \min r}$. ■

Lemma 6. *Within one interval, the network topology is invariant with high probability when the network size is no larger than $\sqrt{\frac{\log 1/2}{\log\left(1 - \frac{\min w \min r}{76S \max v}\right)}}$.*

Proof: When all edges are invariant within one interval, the network topology keeps the same. Consider that a network having n nodes has at most $\binom{n}{2}$ edges. According to Lemma 5, we need $\left(1 - \frac{76S \max v}{\min w \min r}\right)^{\binom{n}{2}} \geq 1/2$. The proof follows. ■

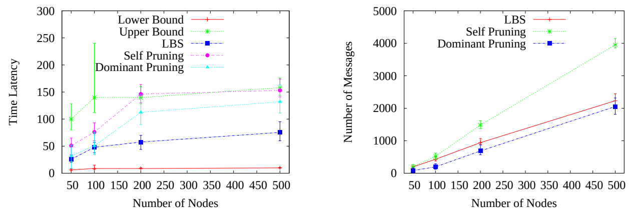
Theorem 3 (Performance)**.** **LBS** *algorithm has $38 + o_M(1)$ approximation ratio with high probability when the network* *size is no larger than* $\sqrt{\frac{\log 1/2}{\log\left(1-\frac{\min w \min r}{76S \max v}\right)}}$.

Proof: The proof follows from Lemma 4 and 6. ■

V. Experimental Evaluation

In our experiment, we compare our LBS approach with the two existing localized schemes, i.e. self-pruning (SP) [9] and dominant pruning (DP) [10], which consider the neighbor list for each node in forwarding procedure but have no theoretical guarantees. Also, we show our approach performs much better in practice than the provided theoretical guarantees. We measure the performance using *time latency* and *message complexity* in the whole broadcast process.

The setup in our experiment is as follows. Initially, all nodes are placed in a fixed cube (500m$\times$500m$\times$500m) at random with different number of nodes. Each node has a fixed transmission range of 50m and $\alpha = 1$. In the broadcast process, we randomly select a source node s and assume each message transmission takes one time-slot. We test on random walk model and reference point group model. For each model, we use the average of 100 runs of different random initial seeds.

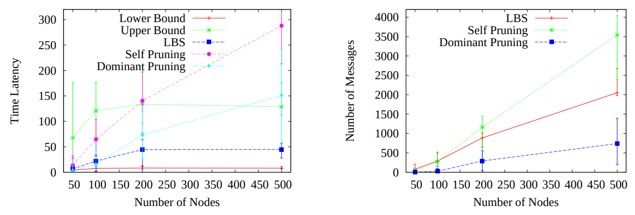


(a) Time Latency (b) Message Complexity

Fig. 3. Random Walk Model

Fig. 3(a) reports the time latency in random walk model. Compared with SP and DP, our LBS approach outperformances both of them by more than 50% in terms of the time latency. As revealed in Fig. 4(a), the performance is more remarkable in reference point group model. In this 500-node scenario, the average ratio is only 5 times the lower bound and outperformances SP and DP approach by more than 80% and 50% respectively. The reason is that our proposed LBS approach has a clever mechanism to determine the transmission node while the transmission schedule is determined only by each node itself and the forwarding list according to the IDs in SP and DP. Since these mechanisms cannot completely avoid the conflicts, the broadcast latency is heavily affected. In addition, these two figures also illustrate that our proposed LBS approach outperformances the theoretical upper bounds by 60% in both random walk model and reference point group model, in which the average approximation ratio is around 10 times the number of intervals (i.e. lower bound) even in the worst case.

As illustrated in Fig. 3(b) and Fig. 4(b), the number of messages using LBS in random walk model and reference point group model is proportional to the number of nodes, which exactly follows our above theoretical analysis. Compared with SP and DP approaches, the message is much less than SP but a little more than DP. In SP, the nodes need to exchange the neighbor list between each other, which increases the message complexity due to a large number of neighbor lists. However, the message complexity is comparatively low in DP due to



(a) Time Latency (b) Message Complexity

Fig. 4. Reference Point Group Model

its clever decision of selecting transmission nodes. Especially in reference point group model, the transmission strategy can be determined only by the group leader, which substantially reduce the exchange messages.

Moreover, the comparison between Fig. 3 and Fig. 4 derives an interesting observation, that is, the broadcast in reference point group model is much faster and less costly than in random walk model. This demonstrates the advantage of group leaders in many real networks, through which all other nodes can be informed directly.

VI. Conclusion

In this paper, we studied the minimum latency broadcast scheduling problem in 3D mobile ad hoc networks. To solve this NP-hard problem, we proposed the LBS approach in 3D MANETs and showed its approximation ratio theoretically with high probability. The experiments further illustrated that the performance overwhelmed both the existing localized approaches and the guaranteed theoretical upper bound in practice.

Acknowledgement

This work is partially supported by NSF Career Award ♯ 0953284, DTRA, Young Investigator Award, Basic Research Program ♯ HDTRA1-09-1-0061 and DTRA ♯ HDTRA1-08-10.

References

[1] http://en.wikipedia.org/wiki/Vehicular_ad-hoc_network.

[2] A. E. Abdallah, T. Fevens, and J. Opatrny. 3d local algorithm for dominating sets of unit disk graphs. In *CCCG*, pages 35–38, 2010.

[3] I. Chlamtac and S. Kutten. On broadcasting in radio networks–problem analysis and protocol design. *Communications, IEEE Transactions on*,1985.

[4] F. Cicalese, F. Manne, and Q. Xin. Faster centralized communication in radio networks. In *ISAAC*, pages 339–348, 2006.

[5] J. H. Conway, N. J. A. Sloane, and E. Bannai. *Sphere-packings, lattices, and groups*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[6] M. Elkin and G. Kortsarz. A logarithmic lower bound for radio broadcast. *Journal of Algorithms*, 52:8–25, 2004.

[7] M. Elkin and G. Kortsarz. Improved schedule for radio broadcast. In *Proceedings of SODA '05*, pages 222–231, 2005.

[8] S. C.-H. Huang, P.-J. Wan, X. Jia, H. Du, and W. Shang. Minimum-latency broadcast scheduling in wireless ad hoc networks. In *INFOCOM*, 2007.

[9] H. Lim and C. Kim. Flooding in wireless ad hoc networks. *Computer Communications*, 24(3-4):353 – 363, 2001.

[10] W. Lou and J. Wu. On reducing broadcast redundancy in ad hoc wireless networks. *IEEE Transactions on Mobile Computing*, 1(2):111–123, 2002.

[11] D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AOA. volume 3, pages 1734–1743 vol.3, April 2003.

[12] S. Parthasarathy and A. Mishra. Minimizing broadcast latency and redundancy in ad hoc networks. In *Proc. of MOBIHOC'03*, pages 222–232. ACM Press, 2003.

[13] J. Schneider and R. Wattenhofer. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In *Proceedings of PODC*, 2008.

[14] W. Shang, P. Wan, and X. Hu. Approximation algorithms for minimum broadcast schedule problem in wireless sensor networks. *Frontiers.Mathem.China*, 2010.

[15] R. Tiwari, T. N. Dinh, and M. T. Thai. On approximation algorithms for interference-aware broadcast scheduling in 2d and 3d wireless sensor networks. In *WASA '09*, pages 438–448. Springer-Verlag, 2009.

[16] R. Tiwari and M. T. Thai. On centralized and localized approximation algorithms for interference-aware broadcast scheduling. Technical report, 2010.

[17] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. *Wirel. Netw.*, 8(2/3):153–167, 2002.