A Cluster-Based Backbone Infrastructure for Broadcasting in MANETs *

Wei Lou and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431, USA
E-Mail: {wlou, jie}@cse.fau.edu

Abstract

Broadcasting is a fundamental service in mobile ad hoc networks (MANETs). Two categories of algorithms, based on source-independent and source-dependent connected dominating sets (CDSs), are proposed in literature to reduce the broadcast redundancy. In this paper, a cluster-based backbone infrastructure is proposed for broadcasting in MANETs. The backbone of the network takes advantage of the cluster structure and only requires clusterheads and some selected gateways to forward the broadcast packet. The static backbone (cluster-based source-independent CDS) consists of fixed clusterheads and selected source-independent gateways. Each clusterhead individually selects its gateways to connect all the clusterheads in its coverage set. The dynamic backbone (cluster-based source-dependent CDS) consists of fixed clusterheads and dynamically selected gateways. It is constructed step by step as the broadcast packet traverses the network. Each clusterhead selects some gateways to forward the packet when it sends the packet to all the clusterheads in its coverage set. Both the static and dynamic backbone structures have a constant approximation ratio to the minimum CDS. Simulations are conducted to compare both the static and dynamic backbones with another cluster-based source-independent CDS algorithm proposed recently.

1 Introduction

Mobile ad hoc networks (MANETs) are collections of autonomous mobile hosts without the help of center base stations. Applying such networks into practice brings many challenges to the protocol design, such as routing in highly dynamic networks, allocating shared wireless channels and saving limited bandwidth. Trade-offs are needed in the protocol design to achieve these conflicting goals.

*This work was supported in part by NSF grants CCR 9900646 and ANI 0073736.

Broadcasting is a fundamental service in MANETs. The broadcast nature of wireless transmissions, that all the neighbors of a host will receive the packet when the host transmits a packet, extremely limits the scalability of the network. When the size of the network increases and the network becomes dense, even a simple broadcast operation may trigger a huge transmission collision and contention that may lead to the collapse of the whole network. This is referred to as the *broadcast storm problem* [9]. Therefore, building some type of backbone infrastructure for a network can enhance the performance of the whole network when the network becomes dense. Basically, the backbone of a network converts a dense network to a sparse one to relieve the communication overhead of the whole network. The cluster structure is a simple backbone infrastructure which has only two levels of hierarchical structure. The network is partitioned into a group of clusters. Each cluster has one clusterhead that dominates all other members in the cluster. Two clusterheads cannot be neighbors. Gateways are those non-clusterhead nodes that have at least one neighbor that belongs to other clusters. It is easy to see that clusterheads and gateways form a backbone of the original network.

Theoretically, we can describe a MANET as a unit disk graph $G = (V, E)$, where the node set V represents a set of wireless mobile hosts and the edge set E represents a set of bi-directional links between the neighboring hosts, assuming all hosts have the same transmission range r. Two hosts are considered neighbors if and only if their geographic distance is less than r. We use $N^k(v)$ to represent v's k-hop neighbor set, including v itself. Generally, a backbone infrastructure of a network can be considered as a *connected dominating set* (CDS) of a given graph. A *dominating set* (DS) is a subset of nodes such that every node in the graph is either in the set or has an edge linked to a node in the set. If the subgraph induced from a DS of the graph is connected, the DS is a CDS. Another concept, an *independent set* (IS), is defined as a set of nodes of the network, in which each pair of nodes are not neighbors. In a cluster network, the set of clusterheads is an IS and the set of the clusterheads and

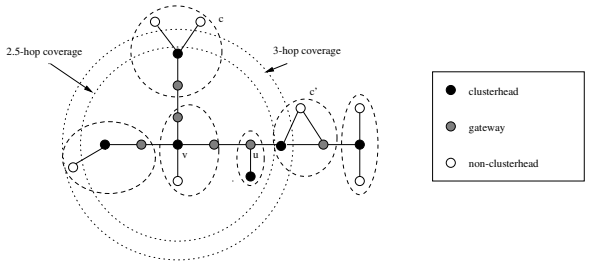

Figure 1. 3-hop and 2.5-hop coverage areas.

gateways is a CDS. It has been proved that finding a *minimum CDS* (MCDS) in a given graph is NP-complete; this applies to a unit disk graph as well. In cluster networks, the challenge is to select a small subset of gateways to connect clusterheads to form an approximation of the MCDS. Even when an MCDS is identified, maintaining such a backbone infrastructure in a mobile environment is a costly operation.

In this paper, the notion of *coverage set* is introduced. A node v's coverage set $C(v)$ is a set of clusterheads that are in a specific coverage area of v. It can be a *3-hop coverage set*, which includes all the clusterheads in its 3-hop neighbor set $N^3(v)$, or a *2.5-hop coverage set*, which includes all the clusterheads in $N^2(v)$ and the clusterheads that have members in $N^2(v)$. Specifically, $C(v)$ consists of $C^2(v)$, a set of clusterheads that are 2 hops away from v, and $C^3(v)$, a set of clusterheads that are 3 hops away from v. In Figure 1, the clusterhead of c' is in v's 3-hop coverage set, but not in v's 2.5-hop coverage set. In general, the size of a clusterhead's 2.5 hop coverage set is less than that of its 3 hop coverage set. Therefore, the cost of maintaining the 2.5-hop coverage set is less than that of the 3-hop coverage set.

We propose a cluster-based backbone infrastructure for broadcasting in MANETs. The backbone of the network takes advantage of the cluster structure and only requires clusterheads and some selected gateways to forward the broadcast packet. For each broadcast, the backbone can be formed statically or dynamically. The static backbone, or *cluster-based source-independent CDS*, consists of the fixed clusterheads and selected source-independent gateways. Each clusterhead individually selects its gateways to connect all the clusterheads in its coverage set. Only nodes in this CDS will participate in forwarding a broadcast packet. The source-independent CDS can be used for all broadcasting (i.e., it is irrelevant to the location of the source). The dynamic backbone, or *cluster-based source-dependent CDS*, is constructed at the time when the broadcast starts and is dependant on the location of the source of the broadcast. It consists of the fixed clusterheads and dynamically selected gateways that are used for the delivery of the broadcast packet. The CDS backbone can be constructed step by step as the broadcast packet traverses the network, where a clusterhead selects some gateways to forward the packet when it sends a broadcast packet. Through

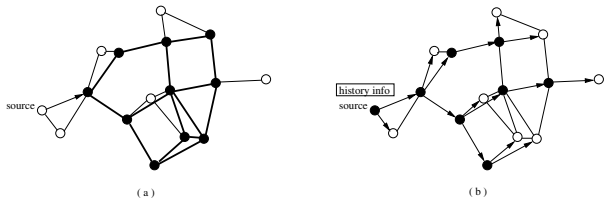

Figure 2. (a) The SI-CDS and (b) the SD-CDS.

the selected gateways, the clusterhead connects all the clusterheads in its coverage set. Because the backbone is constructed step by step, the information of the network can be propagated from upstream nodes to downstream nodes. Therefore, some pruning techniques can be used to further reduce the transmission redundancy. The fact that most source-dependent CDSs have no theoretical constant approximation ratio to the MCDS leads to the result that they have poor performance in the worst case when the network is dense. Both static and dynamic backbone structures have a constant approximation ratio that gives an upper bound for the worst case. Simulation also shows its good performance for the average case. The communication and time complexity of such a backbone are linear to the size of the network, which means the algorithm is message-optimal. The performances of both the static and the dynamic backbones are compared with another cluster-based source-independent CDS algorithm proposed in [1].

2 Preliminaries

Source-Independent CDS vs. Source-Dependent CDS Two categories of algorithms, based on the *source-independent CDS* (SI-CDS) and the *source-dependent CDS* (SD-CDS), are proposed in literature for broadcasting in MANETs. The construction of a SI-CDS is proactive and irrelevant to the source of the broadcast. Many algorithms have been proposed to form a SI-CDS, such as the Spine [3], the marking process with rules 1 and 2 [13], the spanning-tree-based CDS [2], and the message-optimal CDS[1].

Broadcasting in a SI-CDS works as follows: (1) The broadcast starts from the source by sending the broadcast packet to all its neighbors. (2) When a node in the CDS receives the broadcast packet for the first time, it forwards the packet among its neighbors; otherwise, it does nothing. (3) When a node that is not in the CDS receives the broadcast packet, it does nothing. Figure 2 (a) shows a network with a SI-CDS. The black nodes form a CDS, and they are connected by the marked edges. For a broadcast, the source delivers the packet to its black neighbors, and only the black nodes forward the packet.

In a MANET, the backbone infrastructure is not a real physical backbone, but a virtual one that always changes when the topology of the network changes. Therefore, maintaining a static SI-CDS backbone for broadcasting is

costly. The more efficient way is to form a CDS backbone on-demand. The construction of a temporarily set-up SD-CDS backbone is activated by the source of a broadcast. The CDS is constructed step by step and is dependent on the location of the source. A sending node selects some of its neighbors and requests those neighbors forward the broadcast after they receive the broadcast packet. When the broadcast process terminates, the nodes that forward the packet form a SD-CDS. Since the broadcast packet traverses the network in steps, the information of upstream nodes can be propagated with the packet to the downstream nodes so that the downstream nodes can use this information when they select their neighbors for the forwarding purpose. Therefore, the SD-CDS backbone usually generates a smaller number of forward nodes than the SI-CDS backbone. SD-CDS algorithms often differ in the neighbor selection process, such as the multi-point relay [12], the dominant pruning [7] and its extension the partial dominant pruning [8], and the ad hoc broadcast protocol [11].

Broadcasting protocols that use a SD-CDS conduct a broadcast process as follows: (1) The broadcast starts from the source. (2) Each selected forwarding node (including the source node) executes a neighbor selection process: When it receives a broadcast packet for the first time, it selects some of its neighbors to forward the packet. The selected nodes need to cover the senders' 2-hop neighbor set to sufficiently guarantee the success of the broadcast process. Information of the upstream nodes can be attached with the broadcast packet in favor of the use of the downstream nodes. (3) A node that is not selected does nothing when it receives a broadcast packet. In Figure 2 (b), a SD-CDS is constructed when the broadcast packet traverses the network. The black nodes are the selected forwarding nodes and the directed links indicate the paths of the propagation of the broadcast packet. While the packet traverses the network, the history information of the upstream nodes can be used by the downstream nodes to select their forward nodes. We can see that the SD-CDS is dependent on the location of the source, and the number of nodes that forward the broadcast packet is less than the ones in the SI-CDS.

Cluster-Based Broadcasting The distributed clustering algorithm, lowest-ID clustering algorithm [4], is initiated by electing as a clusterhead the node whose ID is locally the smallest one among all its neighbors. At the beginning, all nodes in the network are candidates. When a candidate finds itself to be the one with the smallest ID among all its 1-hop candidate neighbors, it declares itself as the clusterhead of a new cluster and notifies all its 1-hop neighbors. When a candidate receives a clusterhead notification from a neighboring clusterhead, the candidate joins in the cluster, changes itself to a non-clusterhead member of the cluster, and announces its non-clusterhead state to all its neighbors. If it receives more than one clusterheads' declaration, it joins in the cluster whose clusterhead has the smallest ID. Non-clusterheads that have neighbors belonging to other clusters become gateways. The network will eventually be partitioned into clusters where each cluster has one clusterhead and several gateway/non-clusterhead members.

Jiang et al [5] proposed a cluster-based routing protocol (CBRP) that forms a cluster structure by first electing clusterheads and then letting each clusterhead select one or one pair of gateways to connect to each clusterhead in its adjacent clusters.

In [10], Pagani and Rossi set up a cluster-based forwarding tree for a reliable broadcast process. The forwarding tree is rooted at the clusterhead of source and follows the order of clusterhead, gateway, then clusterhead again to build the tree. The gateway that connects the clusterheads records its upstream and downstream clusterheads in the tree when it receives and forwards the broadcast packet. The forwarding tree, thus, can be built level by level until all the clusters join in the tree. Apparently, such a forwarding tree is hard to maintain in MANETs.

Kwon and Gerla [6] proposed a passive clustering scheme that constructs the cluster structure during the data propagation. A clusterhead candidate applies the "first declaration wins" rule to become a clusterhead when it successfully transmits a packet. Then, its neighbor nodes can learn the presence of this clusterhead and change their states to become gateways if they have more than one adjacent clusterhead or ordinary (non-clusterhead) nodes otherwise. The passive clustering algorithm has the advantages of no initial clustering phase, no need of the complete neighborhood information for the clusterhead election and no communication overhead for maintaining cluster structure or updating neighborhood information, but it suffers poor delivery rate and global parameter requirement.

Alzoubi et al [1] proposed a cluster-based message-optimal CDS which is formed with two steps: In the first step, clusterheads are determined by the lowest-ID clustering algorithm. A clusterhead knows all its 2-hop and 3-hop clusterheads with two rounds of neighborhood information exchanges. In the second step, each clusterhead selects a node to connect each 2-hop clusterhead and a pair of nodes to connect each 3-hop clusterhead. All the clusterheads and selected nodes form a CDS of the network. The authors proved that the size of generated CDS has a constant approximation ratio to the MCDS, and the time complexity and message complexity for the construction are both linear to the size of the network, which means the algorithm is message-optimal. Note that this message-optimal CDS is a SI-CDS.

3 A Cluster-Based Backbone Infrastructure

Broadcasting in a Cluster-Based SI-CDS Backbone The cluster-based SI-CDS backbone of the network consists

of all the clusterheads and the selected gateways that connect adjacent clusterheads. The network is partitioned into clusters where each cluster consists of one clusterhead and several non-clusterheads. Clusterheads are elected by the lowest-ID clustering algorithm. Each clusterhead gathers neighbor set information to build its coverage set. It can be a 3-hop coverage set or a 2.5-hop coverage set. Each clusterhead u applies a neighbor selection process that heuristically determines a set of gateways to connect all clusterheads in $C(u)$. The selected gateways will be informed by u to become members of the backbone. Notice that both backbones generated with the 3-hop and 2.5-hop coverage sets are CDSs.

The construction of the backbone, including how clusterheads gather neighbor information and how they select gateways to build the backbone is described as follows. Only the process with the 2.5-hop coverage set is specified here. The process with the 3-hop coverage set is similar.

Each node can learn its neighbors' IDs through HELLO messages. Nodes are grouped into clusters by applying the lowest-ID clustering algorithm. A clusterhead will broadcast a CLUSTER_HEAD message and a non-clusterhead will broadcast a NON_CLUSTER_HEAD message to inform its neighbors.

After the clusters have been formed, each node knows all its 1-hop neighbors. A non-clusterhead u broadcasts a CH_HOP1(u) message which includes all its 1-hop neighboring clusterheads. Once another non-clusterhead v receives the message CH_HOP1(u) from u, v works as follows: If the clusterhead of u is a neighbor of v, v ignores the message; Otherwise, v checks if the clusterhead of u is a new 2-hop clusterhead of v. If so, v creates a new 2-hop clusterhead entry that contains u and the clusterhead of u. When v receives the CH_HOP1 messages from all its non-clusterhead neighbors, it broadcasts a CH_HOP2(v) message that contains all its 2-hop clusterhead entries.

When the clusterhead u receives the CH_HOP1 and CH_HOP2 messages from all its non-clusterhead neighbors, u builds its 2.5-hop coverage set $C(u) = C^2(u) \cup C^3(u)$, where $C^2(u)$ consists of all elements in CH_HOP1 and $C^3(u)$ consists of all elements in CH_HOP2. If a clusterhead appears in both $C^2(u)$ and $C^3(u)$, the one in $C^3(u)$ is removed.

A clusterhead u selects gateways that connect all the clusterheads in $C(u)$ as follows: The neighbor node that directly covers (i.e., connects directly) the maximum number of the clusterheads in $C^2(u)$ is first selected as a gateway. A tie is broken by selecting the node that indirectly covers (i.e., connects via a non-clusterhead) more clusterheads in $C^3(u)$. Node ID is used if the tie still exits. When a neighbor node, say v, is selected, all of the clusterheads in CH_HOP1(v) are removed from $C^2(u)$. If v can indirectly cover some clusterheads in $C^3(u)$ at the same time, these clusterheads are also removed and the corresponding non-clusterheads in CH_HOP2(v) that are associated with these clusterheads are selected as gateways. The selection process repeats until $C^2(u)$ is empty. At this time, if there are any clusterheads in $C^3(u)$ left, the pairs of non-clusterheads that connect u to these clusterheads are also selected as gateways.

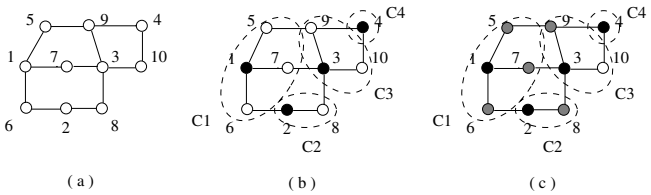

Figure 3. The construction of a cluster-based SI-CDS backbone.

After a clusterhead determines its gateways, it broadcasts a GATEWAY message that contains all the selected nodes among its 2-hop neighbor set by setting the time-to-live field (TTL) of the message to 2. The selected nodes will be informed to become gateways when they receive the GATEWAY message and will forward the message if the TTL field of the message does not reach 0.

Figure 3 shows the construction process of a cluster-based SI-CDS backbone. The black node represents a clusterhead, the gray node represents a gateway, the white node represents others. At the beginning, all the nodes are candidates (Figure 3 (a)). After applying the lowest-ID clustering algorithm, nodes 1, 2, 3 and 4 become clusterheads and create clusters labelled as C_1, C_2, C_3 and C_4, nodes 5, 6 and 7 join in cluster C_1, node 8 joins in cluster C_2, nodes 9 and 10 join in cluster C_3 (Figure 3 (b)). Node 9 sends a CH_HOP1(9) message including nodes 3 and 4, that is, CH_HOP1(9) = $\{3^*, 4\}$, where 3^* means that node 3 is the clusterhead of node 9. When node 9 receives a message CH_HOP1(5) = $\{1^*\}$, node 9 builds a message CH_HOP2(9) = $\{1[5]\}$, where $1[5]$ means that node 9 connects to node 1 via node 5. Also, node 5 sends a message CH_HOP2(5) = $\{3[9]\}$ after it receives the message CH_HOP1(9). Note that node 4 is not added to node 5's 2-hop neighbor clusterhead set since only the clusterheads of those 1-hop neighbors of node 5 (In this case, node 3 is the clusterhead of node 9) will be included. Similarly, nodes 6, 7, 8 and 10 send messages CH_HOP1(6) = $\{1^*, 2\}$, CH_HOP1(7) = $\{1^*, 3\}$, CH_HOP1(8) = $\{2^*, 3\}$ and CH_HOP1(10) = $\{3^*, 4\}$. After nodes 1, 2, 3 and 4 receive the messages CH_HOP1 and CH_HOP2, they build their 2.5-hop coverage sets, respectively; that is, $C(1) = C^2(1) = \{2, 3\}$, $C(2) = C^2(2) = \{1, 3\}$, $C(3) = C^2(3) = \{1, 2, 3\}$ and $C(4) = C^2(4) \cup C^3(4) = \{3\} \cup \{1\} = \{1, 3\}$. Node 1 selects nodes 6 and 7 as gateways and sends a GATEWAY message GATEWAY(1) = $\{6, 7\}$. Similarly, nodes 2 selects nodes 6 and 8 as gate-

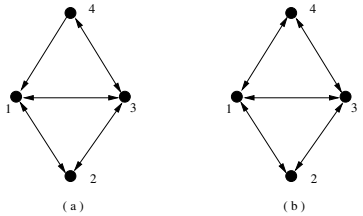

Figure 4. The cluster graphs with: (a) the 2.5-hop coverage set and (b) the 3-hop coverage set.

ways and sends message GATEWAY(2) = $\{6, 8\}$; node 3 selects nodes 7, 8 and 9 as gateways and sends message GATEWAY(3) = $\{7, 8, 9\}$; node 4 selects nodes 5 and 9 as gateways and sends message GATEWAY(4) = $\{5, 9\}$. Note that node 4 selects node 9, not node 10 as a gateway to directly cover node 3 because node 9 can also indirectly cover node 1. When nodes 5, 6, 7, 8 and 9 receive the GATEWAY messages, they become gateways (Figure 3 (c)).

In [14], Lou and Wu proved that the cluster graph $G^{'}$, generated from a connected graph G by using either the 3-hop or 2.5-hop coverage set, is a strongly connected graph. The cluster graph $G^{'}$ is constructed from the clusterheads of a given clustered network G: Each vertex of G' stems from a cluster in G and is represented by the clusterhead of the cluster. Each directed link (v, w) of G' is from clusterhead v to each clusterhead w ($w \in C(v)$). When the 3-hop coverage set is applied, for each pair of clusterheads v and w, if $w \in C(v)$, then $v \in C(w)$. Both directed links (v, w) and (w, v) exist in graph G'. But for the 2.5-hop coverage set, there may exist a pair of clusterheads v and w, where $w \in C(v)$, but $v \notin C(w)$. For the network in Figure 3 (b), the cluster graphs generated with the 2.5-hop and 3-hop coverage sets are shown in Figures 4 (a) and (b).

Theorem 1 *The generated static backbone of the network is a SI-CDS.*

Proof: The clusterheads and gateways form a backbone of the network. The construction process of the backbone that each clusterhead selects gateways to connect all clusterheads in its coverage set maps to the process that generates a cluster graph from the network, where each clusterhead corresponds to a vertex of the cluster graph and each selected gateway(s) corresponds to a directed link between two vertices of the cluster graph. Thus, the strongly connected property of the cluster graph suggests the connected property of the generated backbone. Notice that the set of clusterheads are a DS of the network; therefore, the backbone that consists of clusterheads and gateways is a CDS. Since it has no starting source to construct the backbone and all gateways are individually selected by each clusterhead, this backbone is also source-independent. □

The cluster-based SI-CDS backbone can be used for a broadcast in the same way as any other SI-CDS:

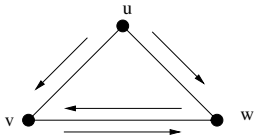

Figure 5. An illustration of the transmission redundancy in a network with three nodes.

Broadcasting in a Cluster-Based SI-CDS Backbone

1. The broadcast starts from the source by sending the broadcast packet to all its neighbors.
2. When a node in the backbone receives the broadcast packet for the first time, it broadcasts the packet among its neighbors; otherwise, it does nothing.
3. When a node that is not in the backbone receives the broadcast packet, it does noting.

Broadcasting in a Cluster-Based SD-CDS Backbone In this part, we consider the case that the backbone of the network consists of the fixed clusterheads and dynamically selected gateways that depend on the source of a broadcast; that is, the gateways are selected at the time when a clusterhead needs to relay the packet. Since this backbone is constructed step by step as the broadcast traverses the network, some pruning techniques can be used to reduce the broadcast redundancy.

Generally, pruning techniques can eliminate some redundant broadcasting operations between two downstream neighbors of a sender if these two neighbors know that they have received a broadcast packet from the same upstream sender. For a simple network with 3 nodes in Figure 5, suppose node u broadcasts a packet, both nodes v and w receive the packet, then they rebroadcast the packet to each other. Apparently, the last two transmissions are redundant.

There are many ways to reduce this kind of transmission redundancy. When a node receives a broadcast packet, if it can back-off a short period of time before it relays the packet, it may receive more copies of the same packet from its other neighbors. If all of its neighbors can be covered by these already received broadcast copies, it can resign its role of re-broadcast operation. For the network in Figure 5, when both v and w receive the packet from u, if both v and w have a random delay before they relay the packet, and w receives the duplicated packet from v before its delay times out, w realizes that all its neighbors (u and v) have already received the packet. Therefore, it does not relay the packet. In this case, one redundant transmission is saved. Another way to reduce transmission redundancy is to piggyback the covered nodes with the broadcast packet when the sender broadcasts a packet. From the information of the piggybacked packet, each receiver can compute which subset of

its neighbor set has already received the packet. For example, in Figure 5, u broadcasts a packet that piggybacks v and w because they have received the packet when u broadcasts the packet. At the time that v and w receive the packet, they know that all of their neighbors (for v they are w and u; for w they are u and v) have received the broadcast, therefore, none of them will relay the packet again. In this case, two redundant transmissions are saved. Of course, these methods will introduce some extra cost, for example, the first one will lead to more delay time and the second one will increase the message length.

By using the pruning technique of attaching the sender's coverage set and selected gateways with the broadcast packet, the broadcast process in a cluster-based SD-CDS backbone works as below:

Broadcasting in a Cluster-Based SD-CDS Backbone

1. If the source is not a clusterhead, it just sends the broadcast packet to its clusterhead.
2. When a clusterhead receives the broadcast packet from its upstream clusterhead sender for the first time, it executes the selection process: It chooses some gateways, called forward nodes, to forward the packet to all the clusterheads in its coverage set. Its coverage set is updated by excluding the clusterhead sender and those clusterheads in the sender's coverage set that are piggybacked with the broadcast packet. The coverage set of this clusterhead, together with its selected forward nodes, are piggybacked with the broadcast packet for the forwarding purpose. A clusterhead will do nothing if it receives a duplicated packet.
3. When a non-clusterhead node receives the broadcast packet for the first time and if it is a forward node, it relays the packet; otherwise, it does nothing.

The selection process is then modified: When a clusterhead v receives a broadcast packet coming from clusterhead u, attached with u's forward node set $F(u)$ and u's coverage set $C(u)$, v knows that all the clusterheads in $C(u) \cup \{u\}$ are covered by $F(u)$, and they do not need to be covered again when v computes its forward node set $F(v)$. Therefore, v can use the heuristic algorithm to determine $F(v)$ to cover the clusterheads in the updated $C(v) = C(v) - C(u) - \{u\}$. Note that both the 3-hop coverage set and the 2.5-hop coverage set can be used here. If the 2.5-hop coverage set is used, $F(u)$ may cover some extra clusterheads in addition to $C(u) \cup \{u\}$. More specifically, if clusterhead v is 3 hop away from u, and u uses a path (u, f, r, v) to deliver the broadcast packet to v, clusterheads in $N(r)$ also receive the broadcast packet. These clusterheads can also be excluded from $C(v)$. Therefore, the updated $C(v) = C(v) - C(u) - \{u\} - N(r)$.

In a clustered network, traversing all nodes in a cluster-based SD-CDS backbone can be viewed as traversing all vertices and edges of the cluster graph which is generated from the original network. As we can see, traversing all vertices of the cluster graph is enough for fulfilling a broadcast process in this network. Eliminating unnecessary edges in the cluster graph, therefore, can reduce the number of nodes that forward the packet in this clustered network. For example, suppose a broadcast starts from node 1 in the network shown in Figure 3 (c). From the view of the cluster graph, the edges $(2, 3)$ and $(4, 1)$ in the cluster graph (Figure 4 (a)) can be eliminated, which suggests that nodes 8 and 5 of the original network (Figure 3 (c)) do not need to forward the broadcast packet. Note that node 9 still needs to forward the packet to clusterhead 4.

Theorem 2 *The generated dynamic backbone of the network is a SD-CDS.*

Proof: Based on the selection process, each clusterhead selects a set of gateways to cover all the clusterheads in its updated coverage set at the time it relays the broadcast. A clusterhead's coverage set is updated to exclude those clusterheads that are in the coverage set of its upstream clusterhead sender. This action corresponds to eliminating an edge between two vertices in the cluster graph of the network if they are downstream vertices of the same upstream vertex, that is, assume u is an upstream vertex in the cluster graph, and both v and w are downstream vertices of u (i.e., there exists edges (u, v) and (u, w)), then, edge (v, w) (and (w, v)) can be eliminated. Eliminating edge (v, w) does not affect the connectivity of these three vertices in the cluster graph. This elimination can be applied to each group of three directly connected vertices in the cluster graph, and the cluster graph after eliminating edges is still strongly connected. This suggests that the backbone is connected. As we know, the set of clusterheads forms a DS of the network. Therefore, the backbone is a CDS. With different sources, the eliminated edges in the cluster graph may be different, that is, the backbone is source-dependent. □

Illustration We illustrate the broadcast process both in a cluster-based SI-CDS backbone and a cluster-based SD-CDS backbone in the network shown in Figure 3 (c). The 2.5-hop coverage set is applied here.

The cluster-based SI-CDS backbone consists of nodes 1, 2, 3, 4, 5, 6, 7, 8 and 9 when the 2.5-hop coverage set is applied, as shown in Figure 3 (c). Suppose node 1 is the source, and all the nodes in the backbone will forward the broadcast packet. In total, 9 nodes (nodes 1, 2, 3, 4, 5, 6, 7, 8 and 9) will forward the packets.

For the cluster-based SD-CDS backbone, suppose the source is node 1, since node 1's 2.5-hop coverage set $C(1)$ is $\{2, 3\}$, it selects nodes 6 and 7 to forward the packet to clusterheads 2 and 3. The broadcast packet piggybacks the forward node set $F(1) = \{6, 7\}$ and the 2.5-hop coverage set $C(1) = \{2, 3\}$. When clusterhead 2 receives the

broadcast packet from clusterhead 1, it updates $C(2) = C(2) - C(1) - \{1\} = \{1,3\} - \{2,3\} - \{1\} = \phi$; then, it only locally broadcasts the packet. When clusterhead 3 receives the packet from clusterhead 1, it updates $C(3) = C(3) - C(1) - \{1\} = \{1,2,4\} - \{2,3\} - \{1\} = \{4\}$; therefore, clusterhead 3 selects node 9 to forward the packet to clusterhead 4. $F(3) = \{9\}$ and $C(3) = \{1,2,4\}$ are piggybacked with the packet. After clusterhead 4 receives the packet, it only locally broadcasts the packet since all clusterheads in $C(4)$ have received the packet. In total, 7 nodes (nodes 1, 2, 3, 4, 6, 7 and 9) will forward the packets.

From this example, we can see that broadcasting in a cluster-based SD-CDS backbone can reduce more transmission redundancy than that in a cluster-based SI-CDS backbone.

4 Performance Evaluation

Performance Analysis We analyze the performance of both the static backbone (cluster-based SI-CDS) and the dynamic backbone (cluster-based SD-CDS) in the following three aspects:

Approximation ratio to the MCDS In a static backbone, as proved in [14] and [1], the size of a static backbone has a constant approximation ratio to the MCDS, which is an upper bound for the worst case. In a dynamic backbone, the backbone eliminates some connections between two adjacent clusterheads if they receive a packet from the same upstream clusterhead sender. This suggests that the size of the dynamic backbone is smaller than that of the static backbone of the same network. Therefore, the constant approximation ratio keeps for the dynamic backbone.

Communication complexity In a static backbone, when the clusters are constructed, each node will send a CLUSTER_HEAD or NON_CLUSTER_HEAD message. A non-clusterhead node will send a CH_HOP1 message that consists of the 1-hop neighbor clusterheads and a CH_HOP2 message that consists of the 2-hop neighbor clusterheads associated with their 1-hop gateways. A clusterhead sends a GATEWAY message that contains the selected gateways that connect the 2-hop and 3-hop neighbor clusterheads. Finally, only the selected gateways will forward the GATEWAY message for each clusterhead. Therefore, the communication complexity of the algorithm is $O(n)$, where n is the size of the network. As mentioned in [1], the algorithm with communication complexity $O(n)$ has already reached optimal. Therefore, the static backbone algorithm is also message-optimal. In a dynamic backbone, each clusterhead needs CH_HOP1 and CH_HOP2 messages to build its coverage set. The clusterhead does not send GATEWAY message, instead, it informs the selected gateways by attaching them with the broadcast packet and only selected gateways forward the packet. Therefore, the communication complexity of each broadcast process is still $O(n)$.

Time complexity In a static backbone, when the clusters are constructed by lowest ID clustering algorithm, the worst case happens when all the nodes are placed in a chain with node IDs that are monotonous from one end to the other end. The cluster construction needs n rounds of unit time to be formed. For each non-clusterhead node, it waits at most $O(\Delta)$ unit time to build CH_HOP1 and CH_HOP2, where Δ is the maximum node degree of the network. For each clusterhead, it also waits at most $O(\Delta)$ unit time for all of its non-clusterhead neighbors' CH_HOP1 and CH_HOP2 messages. Since a clusterhead v has a constant size of clusterheads in its $C(v)$, the selection process terminates at $\Theta(1)$ unit time. At last, the gateways are informed within 2 unit time. The overall time complexity of the algorithm is $O(n)$ in terms of round. In a dynamic backbone, the time complexity for forming clusters is $O(n)$. A clusterhead sender gathers clusterhead information of its coverage set in $O(\Delta)$ unit time. The dynamic backbone is constructed in steps when a broadcast packet traverses the network, which also needs at most $O(n)$ unit time. Totally, the time complexity of each broadcast process is also $O(n)$.

Simulations We measure the average sizes of the CDS backbones constructed by our algorithm (referred to as the static backbone for a cluster-based SI-CDS and the dynamic backbone for a cluster-based SD-CDS, both the 2.5-hop coverage set and the 3-hop coverage set are applied) and by the algorithm in [1] (referred to as the MO_CDS). The simulation runs under the following simulation environment: The confined working space is 100×100. Nodes are randomly placed in this area. The nodes have the same transmission ranges, and the link between two nodes is bi-directional. The network is generated with two fixed average node degrees: d = 6 and 18, which are the representatives of the common and highly dense networks. If the generated network is not connected, it is discarded. We only consider the traffic of the broadcast packets at the network layer. We assume that all the transmission collision and contention are taken care of at the underground physical and MAC layers. For each d, the number of nodes in the network ranges from 20 to 100. We repeat the simulation until the 99% confidential interval of the result is within $\pm 5\%$.

Figure 6 shows the average size of the CDS generated by the static backbone algorithm and by the MO_CDS algorithm with respect to d are 6 and 18. Both algorithms have the similar size of the CDS. Although the static backbone is better than the MO_CDS, the difference is insignificant. We also notice that the difference between the size of the backbone applied the 2.5-hop coverage set and that applied the 3-hop coverage set is less than 2%.

Figure 7 shows the average size of the forward node set for a broadcast process by using the dynamic backbone and the MO_CDS, when d are 6 and 18. The dynamic back-

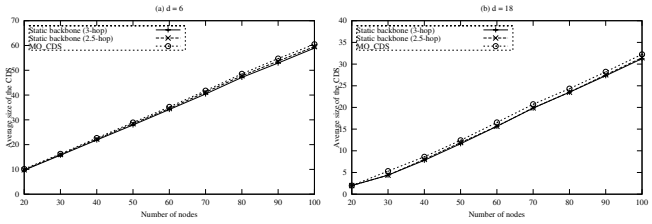

Figure 6. Average size of the CDS: (a) $d = 6$ **and (b)** $d = 18$**.**

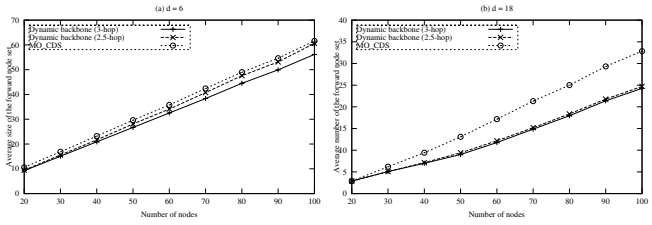

Figure 7. Average size of the forward node set: (a) $d = 6$ **and (b)** $d = 18$**.**

bone algorithm shows much better performance than the MO_CDS.

In Figure 8, we compare the average size of the forward node set between the static and dynamic backbones for a broadcast. Apparently, broadcasting in the dynamic backbone that uses the pruning technique has less broadcast redundancy than that in the static backbone. We also notice that the difference between algorithms with the 3-hop coverage set and the 2.5-hop coverage set is very small.

5 Conclusions

In this paper, a cluster-based backbone infrastructure is proposed for broadcasting in MANETs. We describe the construction of the cluster-based source-independent CDS backbone (static backbone) and the cluster-based source-dependent CDS backbone (dynamic backbone). Actually, the MO_CDS can be treated as a modified version of the static backbone with the 3-hop coverage set. We point out that maintaining a static backbone at all times for broadcasting is costly and unnecessary. Therefore, building a dynamic backbone on-demand is a better choice. Based on simulation results, we can conclude that the pruning technique greatly reduces the number of the forward nodes in a broadcast process. Also, the algorithm with the 2.5-hop coverage set has comparable performance to the one with the 3-hop coverage set while it reduces maintenance cost.

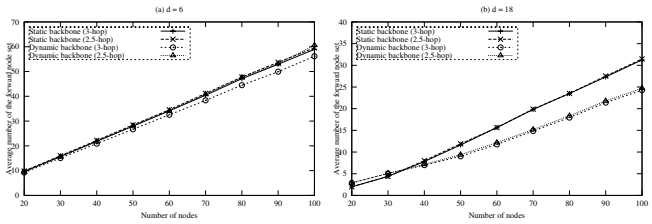

Figure 8. Average size of the forward node sets of the static and dynamic backbones: (a) $d = 6$ **and (b)** $d = 18$**.**

References

[1] K. M. Alzoubi, P. J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. *Proc. of ACM MOBIHOC'2002*, pages 157–164, 2002.

[2] K. M. Alzoubi, P. J. Wan, and O. Frieder. New distributed algorithm for connected dominating set in wireless ad hoc networks. *Proc. of 35th Hawaii Int'l Conf. on System Sciences (HICSS-35)*, pages 3881–3887, Jan. 2002.

[3] B. Das, R. Sivakumar, and V. Bharghavan. Routing in ad-hoc networks using a spine. *Proc. of the 6th Int'l Conf. on Computer communications and Networks (ICCCN'97)*, pages 1–20, Sept. 1997.

[4] A. Ephremides, J. E. Wieselthier, and D. J. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proc. of the IEEE*, 75(1):56–73, 1987.

[5] M. Jiang, J. Y. Li, and Y. C. Tay. Cluster based routing protocol (CBRP) functional specification. *IETF Internet draft*, Aug. 1999. http://www.ietf.org/ietf/draft-ietf-manet-cbrp-spec-01.txt.

[6] T. J. Kwon and M. Gerla. Efficient flooding with passive clustering (PC) in ad hoc networks. *ACM Computer Communication Review*, 32(1):44–56, Jan. 2002.

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

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

[9] S. Ni, Y. Tseng, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. *Proc. of ACM/IEEE MOBICOM'99*, pages 151–162, Aug. 1999.

[10] E. Pagani and G. P. Rossi. Providing reliable and fault tolerant broadcast delivery in mobile ad hoc networks. *Mobile Networks and Applications*, 4:175–192, 1999.

[11] W. Peng and X. Lu. AHBP: An efficient broadcast protocol for mobile and hoc networks. *Journal of Science and Technology*, 2002.

[12] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying for flooding broadcast message in mobile wireless networks. *Proc. of IEEE HICSS-35*, pages 3898–3907, Jan. 2002.

[13] J. Wu and H. Li. On calculating connected dominating sets for efficient routing in ad hoc wireless networks. *Proc. of ACM DIALM'99*, pages 7–14, Aug. 1999.

[14] J. Wu and W. Lou. Forward-node-set-based broadcast in clustered mobile ad hoc networks. *accepted to appear in Wireless Networks and Mobile Computing, a special issue on Algorithmic, Geometric, Graph, Combinatorial, and Vector Aspects*, 2003.