Multicasting in Ad Hoc Networks in the Context of Multiple Channels and Multiple Interfaces

Marco Aurélio Spohn and J.J. Garcia-Luna-Aceves

Abstract— **Multicast routing protocols based on shared trees employ one or more rendezvous points (usually called cores) for coordination. To address fault tolerance in case of core failure, multiple cores can be deployed. The location of cores is crucial for the performance of the protocol. In this context, the problem of finding the location for the cores is similar to the $(\mathbf{k}, \mathbf{r})$-*dominating set* problem, (k, r)-DS, in graph theory. That is, (k, r)-DS is defined as the problem of selecting a subset of nodes D such that the remaining nodes are within distance r from at least k nodes in D. In mobile ad hoc networks (MANETs), finding the location of cores should be computed distributively, because the topology may change frequently. We present a distributed solution to the (k, r)-DS problem, named DKR, which is used for core selection in a novel multicast protocol named *core hierarchical election for multicasting in ad hoc networks* (*CHEMA*). CHEMA is designed to operate in the context of multiple channels and multiple interfaces. One interface is dedicated for the communication among cores and members, using a non-interfering channel. The performance of CHEMA is compared against one of the best performing multicast protocols to date. CHEMA is shown to perform better in all scenarios considered.**

I. INTRODUCTION

Multicast routing protocols can be classified as tree-based and mesh-based. Tree-based can be further classified as single-source, shortest-path trees and shared, core-based trees. Core-based trees are more scalable compared to shortest-path trees, but usually present higher end-to-end delay and poor fault tolerance.

To improve the performance of core-based trees, multiple cores are deployed. The distribution of cores in the network has a direct impact on the performance, since cores placed closer to the receivers can reduce the end-to-end delay.

There are two *one-to-all* designs [1] with multiple cores: *senders-to-all*, and *members-to-all*. In *senders-to-all*, senders transmit to all cores, and members join to just one core (usually the nearest one). In *members-to-all* schemes, senders select one of the cores to send their data packets, and members need to join all cores. In the *one-to-one* approach, each of the cores must join to at least one other core [2]. In this case, senders send to just one of the cores, and receivers join to just one of the cores.

Senders-to-all scheme has several advantages compared to a members-to-all scheme. Both approaches use one multicast tree per core, but in a members-to-all scheme each tree connects all members, which increases the routing state in each router. In a senders-to-all scheme, members decide which core to join, allowing members to choose the core that better satisfy their requirements (e.g., lower end-to-end delay).

While the *one-to-one* approach combines the advantages of the two *one-to-all* designs, it requires a reachability and maintenance protocol between the cores.

Core placement has a direct impact on the performance of the protocol. If the number of cores is fixed, say k cores, then the problem is referred as k-center [3], and is defined as the problem of locating k cores in the network such that the distance from nodes to the cores is minimized. If the number of cores is not fixed, but the maximum distance to a core, say r, is fixed, then the problem is the same as the problem of computing d-hop *dominating sets* (DS) in graphs [4]. That is, d-hop DS seeks to select the minimum number of cores (i.e., dominating node) in the network such that each node is within distance r from at least one core. Both problems are known to be NP-Complete.

The selection of cores could be further extended to include a minimum number of cores within a maximum distance. In this context, the problem of finding the location of the cores is similar to (k, r)-*dominating sets*, (k, r)-DS, in graph theory. The (k, r)-DS problem is defined [5] as the problem of selecting a minimum cardinality vertex set D of a graph $G = (V, E)$, such that every vertex u not in D is at a distance smaller than or equal to r from at least k vertices in D. The problem of computing a (k, r)-DS of minimum cardinality for arbitrary graphs is also NP-complete [5].

When selecting cores, redundancy is achieved by choosing a value for the parameter k greater than one. At the same time, the distance parameter r allows increasing local availability by reducing the distance to the cores.

In mobile ad hoc networks (MANETs), finding the location of cores should be computed distributively, because the topology may change frequently. We present a distributed solution to the (k, r)-DS problem, named DKR, which is used for core selection in *core hierarchical election for multicasting in ad hoc networks* (*CHEMA*) in the context of multiple-channel and multiple-interface.

In CHEMA, each node is equipped with two interfaces. One for general communication, and the other for communication among cores and their members. Cores transmit packets to their members on a specific non-interfering channel via the dedicated interface, and receivers listen in the corresponding channel in the same interface. To reach all members with a single transmission, cores transmit packets with a larger power, such that all member within r-hops from the cores can successfully receive the packet. Therefore, all packets

The authors are with the Computer Engineering Department, University of California at Santa Cruz, Santa Cruz, CA 95064. The second author is also with the Palo Alto Research Center, 3333 Coyote Hill Rd, Palo Alto, CA 94304.

transmitted by the cores are expected to be successfully received by the members.

The rest of this paper is organized as follows. Section II presents a brief review of the related work. Section III describes DKR, which is applicable to ad hoc networks, given that it relies on information limited to the neighborhoods of nodes. Section IV presents CHEMA, a novel approach for multicasting in MANETs with DKR as the core selection mechanism. Section V presents the performance results for CHEMA compared to PUMA, one of the best known multicasting protocol for MANETs. And Section VI concludes this work.

II. Related Work

Many centralized multi-core selection algorithms have been proposed for wired networks [1], [6], [7]. These solutions are greedy variations of either the d-hop DS problem, or the k-center problem. They differ among each other by the metric applied for the selection of cores, which can be delay-bound or hop-count.

Max-Min [8] is an election-based distributed solution for the $(1, r)$-DS problem, which takes $2r$ rounds to complete. cores are computed during the first r rounds, and nodes decide which dominating nodes are going to be their cores during the subsequent r rounds. The authors also show that the problem of computing the minimum $r-$hop dominating set is NP-complete for unit-disk graphs [9].

Liang and Hass [10] proposed a distributed algorithm to compute $(1, r)$-DS. The algorithm is a distributed version of *Greedy Set Cover* (GSC), producing dominating sets with the same cardinality as the centralized solution for this problem. However, their solution requires the $2r$-hop neighborhood information.

Joshi et al [5] have provided centralized solutions for solving the (k, r)-DS problem in *interval graphs* (IG). Even though their solutions are optimal, IGs are limited to very simple network topologies.

III. Distributed Solution to Core Selection Using (k,r)-Dominating Sets

We propose DKR, which is a distributed algorithm for core selection using (k, r)-DS. DKR is well suited for both synchronous and asynchronous networks. In the synchronous network model, nodes exchange messages in synchronous rounds. In the asynchronous network model, nodes take steps at arbitrary times. Even though there are no rounds in the asynchronous model, it is possible to simulate rounds [11]. In order to do that, a node tags the message with its round number x. The recipient waits to receive round x messages from all its neighbors before transitioning to the next round.

A. Summary Description

We assume that nodes have unique identifiers (IDs), and that nodes know who their neighbors are. The latter can be implemented by means of a neighbor protocol with which nodes exchange hello messages [12], as part of the MAC protocol, or using periodic *HELLO* messages as part of the protocol itself.

The *status* of a node reflects its role during the core selection process. Initially, there is no established hierarchy among nodes, and the nodes assume an *unknown* status. As the nodes organize themselves, their status change to reflect their role in the network, which can be one of the following:

- *Dominating*, the node is a *core*.
- *Pending Dominating*, the node may become a *core*.
- *Dominated*, the node has *at least* k cores within distance r.
- *Gateway*, in addition to being *dominated*, the node connects other nodes to their cores.

A **round** of messages is defined as the successful transmission of a message $\mathbf{m}$ by any node n to all its one-hop neighbors. If rounds are numbered, a round x is deemed complete only after all nodes have sent the messages for round x. DKR has two phases:

- *Phase One* (Election Phase): Each node elects the k smaller ID nodes (including the node itself) within distance r. Elected nodes are just *candidates to be cores*. Because each node has its own set of k elected nodes within distance r, the sets of elected nodes *dominate* all non-elected nodes in the network.
- *Phase Two*: During this phase *cores* are assigned, and nodes are affiliated to their cores.

Clearly, there must be at least k nodes in every node's r-hop neighborhood for the required *multiple domination* to be satisfied. In the subsequent description of DKR, we assume that multiple domination can be satisfied at each node.

It is possible that not all nodes elected during *Phase One* become *cores*, because some redundant candidates are identified, and pruned. The rationale for choosing node IDs over node degree for the election process is that elections based on node degree can result on high turnover of dominating nodes when the topology changes, because the degree of a node is much more likely to change than the node ID relative to its neighborhood [13].

B. Phase One

This phase takes r rounds to complete in a static topology. For asynchronous networks, rounds are simulated as described previously. At the beginning of a new round, a node advertises its list of $K \leq k$ nodes with smaller IDs (including the node itself), and the respective distance to each node listed. After a number of rounds (r rounds in a static topology), a node i in the network learns the set of up to k nodes with smaller IDs (including the node itself) within distance r from it. We denote such a set by $D_i^{'}$.

An elected node can elect itself or be elected by other nodes. A node that elects itself is called *properly-elected* if the node is not elected by any other node, and is called *self-elected* if the node is elected by at least one other node. A node that does not elect itself and is elected by other nodes that are not elected is called *neighbor-elected*. After the election, any node i in the network changes its status as follows: If node

i is properly-elected or self-elected, node i changes status to pending dominating. Otherwise, node i has status dominated.

Note that a *properly-elected* node must become dominating, because there are at most $k-1$ other elected nodes in node i's r-hop neighborhood. Because identifying properly-elected nodes would incur extra overhead, they are implicitly notified of their dominating status after not hearing from enough dominating nodes within a given period of time.

A *neighbor-elected* node i is elected by at least one node, call it n, which is not elected and for which node i is strictly required. That is, there is no self-elected node in node n's r-hop neighborhood that could possibly replace node i; otherwise, node n would have elected that self-elected node. Even though in some cases a properly-elected node could replace node i, initially DKR chooses to select all neighbor-elected nodes as cores.

C. Phase Two

During *phase two*, some or all nodes elected during *phase one* become cores. In addition, the rest of the nodes are affiliated to their cores.

The messages used during this phase are:

- *Local Advertisement* (LA): A message having the list of nodes elected by the sender, and the respective next-hop to each one of the elected nodes.
- *Neighborhood Advertisement* (NA): A message advertising a core.
- *Notification*: A message sent to notify a node that must become core.
- *Join*: A message sent to notify, or to connect to a given core.

Because neighbor-elected nodes are not aware of their election, a notification mechanism is needed to notify them. Depending on the coverage provided by neighbor-elected nodes, some self-elected nodes may be ruled out as cores.

At the beginning of *phase two*, dominated nodes send to their one-hop neighbors an LA message containing their elected nodes. Any dominated node i proceeds as follows upon receiving an LA message:

- If node i is listed in the advertisement, node i changes its status to dominating, triggering an NA message announcing node i as core to all its r-hop neighbors. This is accomplished by broadcasting the NA message using restricted *blind-flooding* with the *time-to-live* (TTL) field set equal to r.
- If node i is not listed in the LA message but is listed as a next hop to any advertised node, then node i changes its status to gateway.
- For any advertised node $n \in LA$ that is not among the nodes elected by node i (i.e., $n \ni D_i^{'}$) it sends a *Notification* message to node n. Upon receiving the notification, if the notified node is not yet dominating, the node advertises itself via an NA message.

Definition 1 *For any node i, and for all $n \in D_i^{'}$, node n is deemed* ***validated*** *only upon the reception of the respective NA message advertising node n; otherwise, node n is not yet* ***validated****.*

Any neighbor-elected node n eventually changes its status to dominating by either receiving a *Notification* message originated at some node k within distance $1 < d < r$, or by receiving an LA message from some one-hop neighbor. In any case, once node n becomes dominating, it sends an NA message.

Because an NA message is sent only when a node changes its status to dominating, nodes receiving an NA message advertising node n know that this is a core. When processing an NA message for node n, any node i inserts an entry for node n in $D_i^{'}$.

A *pending dominating* node i does not become a *core* if ***(a)*** node i has *at least* k validated dominating nodes within distance r, and ***(b)*** every node n, which elected node i during *phase one*, is also covered by a set of validated dominating nodes.

Definition 2 ***Wait Period*** *is the minimum time required for reaching an agreement in* phase two.

In the worst case, a *Notification* for node n is initiated by some neighbor i located $r-1$ hops away from node n. The correspondent NA message initiated by node n reaches the most distant neighbors (i.e., r hops away from node n) after r successfully transmissions of message NA. Therefore, **Wait Period** should be larger than the time required for $2r$ transmissions of a message. After a period of time equivalent to *Wait Period* any node i in the network checks its coverage. If node i is *pending dominating*, and it does not have enough validated entries in $D_i^{'}$, then node i changes status to *dominating*, and sends an NA message. This means that node i does not have enough information for ensuring its own coverage (i.e., node i does not know if it has at least k dominating nodes within distance r). Otherwise, any non-*dominating* node i sends a *Join* message to k nodes from $D_i^{'}$ (including the nodes already validated). If there are more than k validated entries in $D_i^{'}$, node i chooses the closest ones (ties are broken choosing the node with smaller ID).

Like a *Notification*, which also serve for assigning *gateways* while the message is being routed to its destination, *Join* messages also serve to notify any *pending dominating* node that is still required as core. That is, even though some *pending dominating* node i finds itself covered, there might be some node j that still needs node i (i.e., without node i, node j does not have the required number of dominating nodes). While a *Join* message is being routed to destination n, a node i processing the message does not need to relay it if a *Notification*, or another *Join* message, had already been sent to node n. So, we assume that every node keeps track of recent *Notification* and *Join* messages sent by the node. After *Join* messages reach their destinations, all regular nodes are connected to their cores.

D. Example

Consider the example presented in Figure 1, where nodes are computing a $(2, 3)$-DS of the network. During *Phase One*

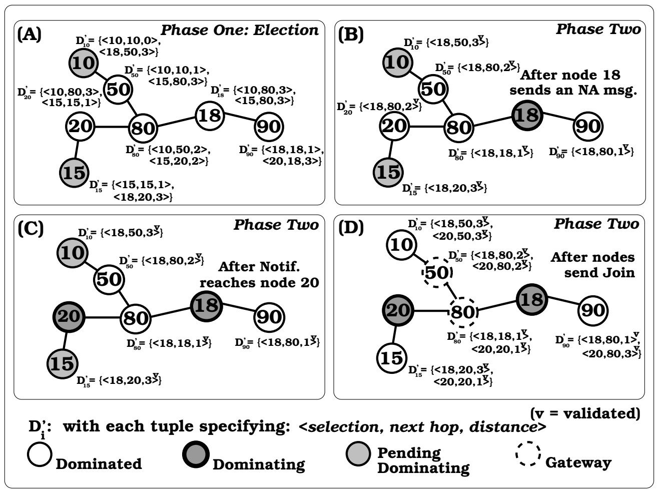
(A)
Phase One: Election
D'_10 = {<10,10,0>, <18,50,3>}
D'_50 = {<10,10,1>, <15,80,3>}
D'_20 = {<10,80,3>, <15,15,1>}
D'_18 = {<10,80,3>, <15,80,3>}
D'_80 = {<10,50,2>, <15,20,2>}
D'_90 = {<18,18,1>, <20,18,3>}
D'_15 = {<15,15,1>, <18,20,3>}
10
50
20
80
18
90
15
(B)
Phase Two
After node 18 sends an NA msg.
D'_10 = {<18,50,3>v}
D'_50 = {<18,80,2>v}
D'_20 = {<18,80,2>v}
D'_80 = {<18,18,1>v}
D'_90 = {<18,80,1>v}
D'_15 = {<18,20,3>v}
(C)
Phase Two
After Notif. reaches node 20
D'_10 = {<18,50,3>v}
D'_50 = {<18,80,2>v}
D'_80 = {<18,18,1>v}
D'_90 = {<18,80,1>v}
D'_15 = {<18,20,3>v}
(D)
Phase Two
After nodes send Join
D'_10 = {<18,50,3>v, <20,50,3>v}
D'_50 = {<18,80,2>v, <20,80,2>v}
D'_80 = {<18,18,1>v, <20,20,1>v}
D'_90 = {<18,80,1>v, <20,80,3>v}
D'_15 = {<18,20,3>v, <20,20,1>v}
(v = validated)
D'_i: with each tuple specifying: <selection, next hop, distance>
Dominated
Dominating
Pending Dominating
Gateway

Fig. 1. Computing a $(2, 3)$-DS of the network

nodes elect the two nodes with smaller IDs in their 3-hop neighborhood (Figure 1(A)). Nodes 10 and 15 are *self-elected*, and nodes 18 and 20 (both elected by node 90) are *neighbor-elected*. *Self-elected* nodes assume status *pending dominating*. The remaining nodes assume status *dominated*, and send an LA message advertising their list of elected nodes. After receiving the LA message from neighbor 90, node 18 changes status to *dominating*, and sends an NA message that eventually reaches all nodes within distance 3 from node 18. Figure 1(B) show the status of nodes, and their corresponding *validated* dominating entries. Besides sending the NA message, node 18 also sends a Notification to node 20. Figure 1(C) presents the status of the network after the notification has reached node 20. The notification makes node 20 change its status to *dominating*, triggering an NA message that eventually reaches all neighbors within distance 3 from node 20. After all affected nodes process this NA message, we notice that all dominated nodes are satisfied, because each of them have 2 validated entries in their $D^{'}$ lists. *Wait Period* should be set appropriately, so that by the time the event *CheckStatus* happens all NA messages have already been delivered and processed. Figure 1(D) shows the status of the network after all dominated nodes have sent out their *Join* messages to their dominating nodes (assume that *Join* messages are grouped together whenever different dominating nodes are reachable through the same node). In this case, because nodes 50 and 80 have either relayed a *Notification*, or a *Join* message, they serve as *gateways* for other dominated nodes.

E. Analysis of DKR

To prove the correctness of DKR, we have to show that it is *safe* (i.e., the algorithm computes a (k, r)-DS of the network), and that it is *live* (i.e., it completes within a finite period of time).

Lemma 1 Phase one *of DKR has time complexity of $O(n\delta r)$, where n is the number of nodes in the network, δ is the largest node degree, and r is the distance parameter.*

Proof: During each round, nodes send messages to all their one-hop neighbors. *Phase one* takes r rounds. Assuming a network of n nodes, and that nodes have at most δ links, *phase one* of DKR requires $O(n\delta r)$ messages to complete. Therefore, the time complexity of *phase one* is $O(n\delta r)$. ■

Lemma 2 *After r rounds of* **successful** *transmission of message $\mathbf{m}$, the message is propagated up to r hops away from the originating node.*

Proof: This can be proved by induction on the distance d from the originating node. The base case is when $d = 0$, and corresponds to the originating node n_0. Now consider a node v at distance r from n_0. A neighbor u of node v at distance $r - 1$ received the message. Therefore, node u sends the message to all neighbors, including v. Eventually node v receives the message. ■

Theorem 1 Phase One *of DKR correctly computes a (k, r)-DS of any arbitrary connected graph $G = (V, E)$.*

Proof: We assume that nodes know their one-hop neighbors. The system is either synchronous or asynchronous. In the latter case, rounds are simulated by tagging advertisements with the round number. By Lemma 2, after r rounds a node ID is propagated at most r hops away. Because nodes advertise their $K \leq k$ known smaller IDs, after r rounds every node $n \in V$ learns the $K \leq k$ nodes, $D_n^{'}$, with smaller IDs located within distance r. Lets assume that S is the set composed of *proper-elected*, *self-elected*, and unsatisfiable nodes (i.e., a node i is deemed unsatisfiable if $|D_i^{'}| < k$), and that R is the set of satisfiable non-*proper/self-elected* nodes (i.e., $R = V - S$). It follows that the set $D = \{\bigcup_{n \in R} D_n^{'} + S\}$ is a (k, r)-DS of G. ■

Theorem 2 *Phase Two of DKR correctly connects dominated nodes to* **at least** *k dominating nodes* **at most** *r hops away.*

Proof: At the beginning of *phase two*, any *dominated* node i advertises its list of elected nodes, $D_i^{'}$, by locally broadcasting the list via an LA message. Any node n with status *dominated*, or *gateway*, upon receiving an LA message from neighbor m, changes its status to *dominating* if the advertisement lists node n, implying that node $n \in D_m^{'}$. In this case, node n sends an NA message which is then propagated to all nodes within distance r from node n. For every node k in the LA message such that $m \ni D_n^{'}$, a notification is sent to node k if node n is on the path to node k (i.e., for node n the next-hop to node k is known, and it is not m). If this is the case, at least one neighbor of node m has a route to node k, because nodes are elected based on the advertisements sent by one-hop neighbors during *phase one*. Eventually, the LA message sent by node m reaches all its one-hop neighbors, including the nodes with routes to the nodes elected by node m. A Notification for node n, when necessary, is issued (initiated or relayed) just once by any node i. A *dominated* node relaying a notification changes its status to *gateway*. Once a *Notification* reaches the destination, if the status of the destination is not *dominating*, it changes its status to *dominating*, and advertises itself sending an NA message. If any node i relaying an NA messages currently has fewer than k validated entries in $D_i^{'}$, then an entry with the validated node is inserted in $D_i^{'}$. After a period of time equal to *Wait Period* (starting from the beginning of *Phase Two*), every node i in the network checks the number of *validated* entries in $D_i^{'}$. If node i's status is *pending dominating*, and it has $l < k$ validated entries in $D_i^{'}$, then node i changes its status to *dominating*, and it sends an NA message. Otherwise, if node i's status is not *dominating*, it sends a *Join* message to all its dominating nodes (validated or not). Any node n receiving a *Join* message, for destination d, does not need to relay the message in case node n had already initiated, or relayed, a *Join* or *Notification* message to node d before. After all the *Notification*, or *Join*, messages have reached their destinations, the paths from dominated nodes to their respective dominating nodes are formed by nodes that are either *dominating* or *gateway*. Because all nodes must check their status after a finite period of time (i.e., *Wait Period*), and any non-*dominating* node i must send *Join* messages to all its *dominating* nodes, (k, r)-coverage is guaranteed. ■

IV. Core Hierarchical Election for Multicasting in Ad Hoc Networks (CHEMA)

CHEMA uses DKR for the election of cores, and is designed to work in the context of multiple channels and multiple interfaces. CHEMA's main features can be summarized as follows:

- Deploy multiple cores, with DKR as the core selection mechanism. This allows flexibility in terms of redundancy, and a bounded distance to the selected cores. *Core announcements* are used to disseminate core information throughout the network. To reduce overhead, a single announcement aggregates information about all known cores.
- Use the senders-to-all approach. To reduce overhead, the packet header lists which neighbors should relay the multicast packet on a core basis. Instead of sending one packet toward each core, nodes relay the packet whenever they are listed at least once as a next-hop to any core. Before relaying the packet, the header is updated with entries for those cores for which the node is requested to forward the packet.
- Multi-channel and multi-interface: Each node has at least two interfaces. One is dedicated to receiving multicast transmissions from cores, and the other is used for any other transmission. Each core transmits in a channel different than any possible interfering core. That is, through core announcements nodes learn about all cores in the network, and the distance to each one. Using this information, cores select channels so that they do not interfere with other cores.
- Single *shot* approach: Once a multicast packet reaches the core, the packet is transmitted just once via the dedicated interface. In order to reach all receivers, the packet is transmitted with an increased power so that all nodes within r hops from the core can receive it. Because cores use different channels, and an interface is dedicated for receiving packets from the core, receivers should receive all transmissions sent by the core.

Multiple cores are selected via DKR. While cores have not yet been elected, multicast data packets are transmitted via *blind flooding*. After cores are elected, receivers join the nearest core by sending a join message to the core. Nodes aggregate all the fresh core announcements they receive, and broadcast them periodically every *core announcement interval* (which by default is set to be $3s$). Core announcements also include the number of members each core has. To let cores know about any associated members, an explicit *multicast join* message is sent from the receiver to the desired core whenever a node wants to join a multicast group. Note that this is not the same as the association provided by the join messages sent during the execution of DKR, which provides for connectivity from any node to at least k cores within distance r.

Senders send multicast packets to all cores with members. Instead of sending one packet per core, the sender broadcasts just one packet with all the information regarding the cores that need to be reached. That is, the packet header includes an entry for every core and the corresponding next hop toward the core (recall that in DKR nodes keep information about which neighbors are used to reach each core). A node receiving the packet for which it is listed as a next hop to any core forwards the packet. Before relaying the packet, entries for which the node is listed as a next-hop are updated with the current information, and any other entries are excluded.

Because multicast packets are broadcast unreliably, a node may retransmit a packet up to N times, unless it receives an implicit acknowledgment. That is, for every multicast packet transmitted the node relaying the packet keeps record of the packet and which neighbors should relay the packet. After a period of time equivalent to an *acknowledgment timeout*, the node checks if it has overheard any of the relayers transmissions. If the node fails to hear any of the relayer's transmission, the node retransmit the packet including only those nodes from which it has not yet heard from. Ideally, the length of an *acknowledgment timeout* should be set dynamically, because it depends on the level of contention, which is higher with a larger number of transmitting nodes.

Nodes are equipped with two radio interfaces. One is used for communication among cores and receivers, and the other is for general communication. More specifically, cores use a dedicated interface for transmitting multicast packets to their members, and the receivers use the same interface to receive packets from their cores. To allow for multiple cores to transmit simultaneously without interference, we assume that each core transmits on a different channel than any possibly interfering core. Therefore, the dedicated interface is set for transmitting and listening using a specific channel.

The problem of assigning non-interfering channels to cores is similar to the *graph coloring* problem in graph theory. Considering a core with transmission range of r, any core within distance $2r$ may interfere (i.e., even though the two cores may be out of range of each other, they may have members within range of both cores). In the context of MANETs, any distributed approximation to the graph coloring problem could be applied for assigning channels to the cores. Instead, we choose to limit the total number of cores in the network to the maximum number of orthogonal channels available for the dedicated interface. Because nodes learn about all cores in the network (through the periodical core announcements), channels are assigned lexicographically.

To reduce delay, and to avoid retransmissions from nodes between the core and members located more than one hop away from the core, cores transmit multicast data packets with a larger power. The transmit power should be set so that the packet can be successfully received by any node up to r hops from the core. Even though this approach increases energy consumption, it is expected to reduce the end-to-end delay and control overhead, because a single transmission from the core is supposed to reach all core members at the same time.

V. Performance

In this section we present performance results for the efficiency of DKR compared to *Max-Min* (recall that *Max-Min* is for computing $(1, r)$-DS, and that the (k, r)-DS problem is NP-Complete), and results comparing CHEMA against the *protocol for unified multicasting through announcements* (PUMA) [14].

A. Efficiency: DKR compared against Max-Min

When comparing DKR against *Max-Min*, we consider only $(1, r)$-DSs because the latter works only for this particular configuration. To focus on the efficiency of the heuristics themselves, we use a customized simulator for ad hoc networks, and assume an ideal MAC protocol with which no collisions can occur. This is the same approach adopted in all prior work [15]–[18] to compare the efficacy of heuristics. As discussed previously, DKR works in both synchronous and asynchronous networks. However, for the simulations we assume synchronous networks.

Experiments are repeated for 100 trials with different network topologies, varying the number of nodes and terrain size. Nodes are randomly placed over the terrain, and connectivity is tested to ensure that the network is connected. The radio range is set to $250m$. The results represent the average over the 100 different networks . The network size is varied from 100 nodes to 500 nodes, with increments of 50 nodes. For the same number of nodes, we vary the terrain size according to two configurations so that we can test the algorithms for different node density. Configuration 1 has a node density of $50\ \frac{\text{nodes}}{km^2}$, and Configuration 2 has $100\ \frac{\text{nodes}}{km^2}$.

Because DKR discards self-elected nodes whenever possible, in the worst case all elected nodes become *cores*. However, in *Max-Min* [8] all nodes elected at the end of the first r rounds become dominating; but, it is only during the second set of r rounds that some of the elected nodes find out about their *dominating* status. Besides that, in certain scenarios *Max-Min* generates dominating nodes that are on the path between a node and their elected nodes. In that case, only during the *convergecast* (which is used to connect regular nodes to their dominating nodes) that nodes adjust their selections to

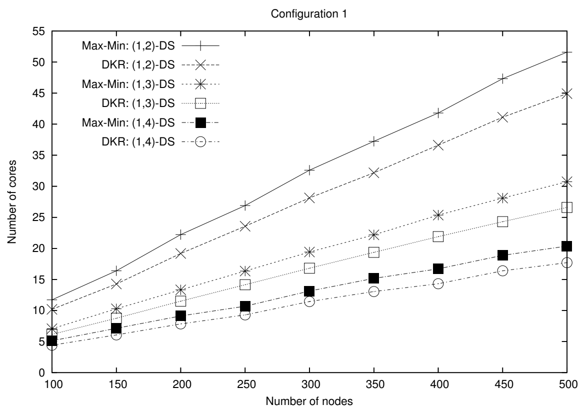

Fig. 2. DKR versus *Max-Min*, Configuration 1: number of *cores*.

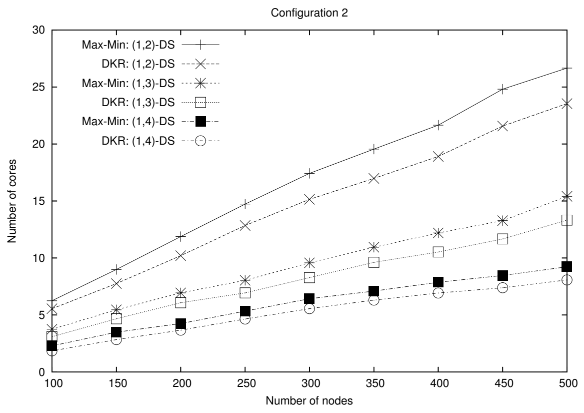

Fig. 3. DKR versus *Max-Min*, Configuration 2: number of *cores*.

the closest dominating node. To show how DKR reduces the number of cores compared to *Max-Min*, we present simulations for different values of the distance parameter.

Figure 2 presents the results for the total number of cores for Configuration 1, varying the distance parameter from 2 to 4. And Figure 3, presents the results for Configuration 2. For both configurations DKR always selects fewer cores compared to *Max-Min*, meaning that usually some *self-elected* nodes are ruled out as cores.

B. CHEMA versus PUMA

PUMA has been shown to outperform two of the state of the art multicast routing protocols for MANETs (i.e., ODMRP [19] and MAODV [20]). PUMA presents the following characteristics: receiver-oriented; core based (one core per group); mesh-based, providing multiple routes from senders to receivers.

Only the core performs control packet flooding in PUMA. In CHEMA it is the same, but information about multiple cores are aggregated to reduce control overhead (PUMA also applies aggregation when flooding information about multiple groups).

We compared CHEMA against PUMA using the Qualnet$^{\text{TM}}$ [21] network simulator. Each simulation was run with four different random-number seeds. Timer values

TABLE I
SIMULATION PARAMETERS

Simulator	Qualnet^{TM}3.5
Simulation time	$350s$
Terrain Size	1000 X 1000 m
Number of nodes	50
Node placement	Random
Mobility	Static
Radio Range	$250m$
MAC protocol	802.11
Channel Capacity	2 Mbps
Data packet size	512 Bytes

(i.e., *core announcements* in CHEMA, and *multicast announcements* in PUMA) were set to three seconds. Table I presents details about the simulation parameters.

In CHEMA, cores use a multiple of the regular radio range (i.e., for distance domination r, cores have a radio range of $r \cdot 250m$) for the dedicated interface. DKR is executed every $16s$ for core assignment. Because only static topologies are considered, the cores remain the same throughout the simulation.

Four performance metrics are evaluated:

- Packet delivery ratio: The ratio of the data packets delivered to the receivers to those data packets expected to be delivered (i.e., data packets sent times the number of receivers).
- Average end-to-end delay for data packets, including all possible delays caused by queuing at the interface, retransmission delays, and propagation and transfer times.
- Control overhead: The number of control packets transmitted per data packet delivered.
- Total overhead: The ratio of the total packets transmitted (i.e., control + data) to the data packets delivered.

For the simulation scenario, traffic load is varied across $\{1, 2, 5, 10, 25, 50\}$ packets/s. There are 5 senders, and 20 receivers for one multicast group. That is, the number of packets expected to be delivered varies from 20 packets/s to 1000 packets/s. Both senders and receivers are chosen randomly among the nodes in the network, and traffic load is equally distributed among all senders.

Even though DKR allows a myriad of scenarios for core selection, we consider just a few configurations for the purpose of simulation. For a 50-nodes network, at most eight cores are allowed (and there are 8 orthogonal channels for the dedicated interface). Values 3 and 4 are tested for the *distance domination*, and at least one core is selected within the specified distance. These two configurations are presented in the graphs as CHEMA $(1,3)$-DS, and CHEMA $(1,4)$-DS, respectively. For the networks considered, three cores are elected in average in the first configuration, and fours cores in the second configuration.

Figure 4 presents the results for packet delivery ratio. CHEMA delivers almost 100% of the data packets for all trafic loads considered. But PUMA cannot deliver more than 70% of packets for traffic load of 50 packets/s, due to increasing contention and collision of packets. On the other hand, mainly because CHEMA applies the *one shot* approach and the non-interfering channels for cores, once packets are transmitted by the core the packets are successfully received by the receivers.

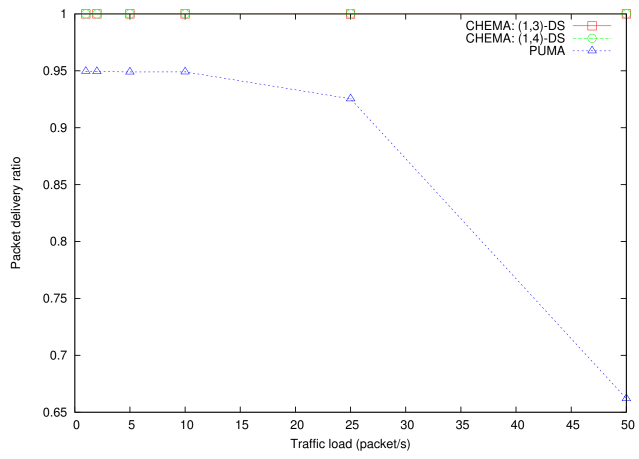

Fig. 4. Packet delivery ratio

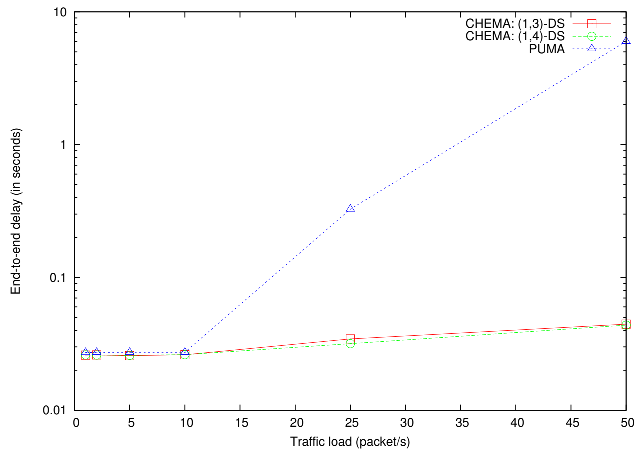

Fig. 5. End-to-end delay

For flows of up to 10 packets/s both protocols have similar end-to-end delay (Figure 5). While CHEMA has a small increase in terms of end-to-end delay for flows larger than 10 packets/s, PUMA experiences an exponential increase in average end-to-end delay. These results, together with the delivery ratio, indicate that CHEMA not only delivers more packets but does so incurring smaller end-to-end delays. This shows that it pays off sending packets to multiple cores and using a single transmission per packet from the cores to their members.

Even though CHEMA sends more control packets (mainly due to the election process) compared to PUMA, both protocols present similar control overhead because CHEMA delivers more packets (as shown in Figure 6). However, in terms of total overhead CHEMA incurs less than half total overhead compared to PUMA (Figure 7). CHEMA requires fewer transmissions for every data packet delivered, specially because once the packets reach the targeted core it takes just one transmission per data packet to reach all core's receivers.

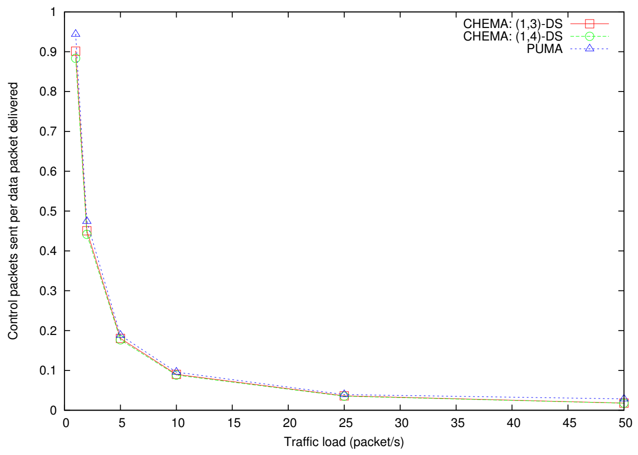

Fig. 6. Control overhead

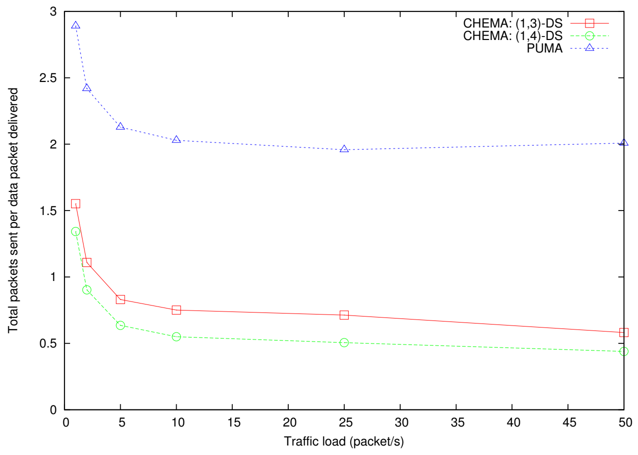

Fig. 7. Total overhead

VI. Conclusion

We have presented a novel approach for core election using (k, r)-*dominating sets* in multicast protocols based on shared trees. DKR is the first distributed solution to the (k, r)-DS problem, which allows the selection of k cores within distance r for any regular node in the network. DKR is applicable to ad hoc networks, given that it relies on information limited to the neighborhoods of nodes.

We proposed a novel multicast protocol named *core hierarchical election for multicasting in ad hoc networks* (*CHEMA*), which is designed to work with multiple channels and multiple interfaces. CHEMA applies DKR for core election, with a dedicated interface using non-interfering channel for communication among cores and any multicast member. Because cores use a larger radio range for the dedicated interface, it requires just one transmission per data packet for any core member to receive the packet.

CHEMA is compared against the *protocol for unified multicasting through announcements* (PUMA), which is one of the best performing multicast routing protocols for MANETs. CHEMA is shown to outperform PUMA in all aspects. CHEMA delivers more packets, incurs small end-to-end delays, and drastically reduces the total control overhead.

Acknowledgments

We would like to thank Ravindra Vaishampayan for providing us with the code for PUMA. This work was supported in part by CNPq (Brazil), the Baskin Chair of Computer Engineering at UCSC, the National Science Foundation under grant CNS-0435522, the UCOP CLC under Grant SC-05-33, and by the U. S. Army Research Office under grant No. W911NF-05-1-0246. Any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] D. Zappala, A. Fabbri, and V. Lo, “An evaluation of shared multicast trees with multiple cores,” *Telecommunication Systems*, vol. 19, no. 3-4, pp. 461–479, March - April 2002.

[2] T. Ballardie, P. Francis, and J. Crowcroft, “Core based trees (cbt),” in *SIGCOMM ’93: Conference proceedings on Communications architectures, protocols and applications*. ACM Press, 1993, pp. 85–95.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability*. Freeman, San Francisco,, 1978.

[4] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, Eds., *Fundamentals of Domination in Graphs*. Marcel Dekker, Inc., 1998.

[5] D. Joshi, S. Radhakrishnan, and C. Narayanan, “A fast algorithm for generalized network location problems,” in *Proceedings of the ACM/SIGAPP Symposium on Applied Computing*, 1993, pp. 701–8.

[6] H. F. Salama, “Multicast routing for real-time communication on high-speed networks,” Ph.D. dissertation, North Caroline State University, 1996.

[7] A. Karaman and H. Hassanein, “Dcmc - delay-constrained multipoint communication with multiple sources,” in *IEEE International Symposium on Computers and Communications*, 2003.

[8] A. D. Amis, R. Prakash, T. Vuong, and D. Huynh, “Max-min d-cluster formation in wireless ad hoc networks,” in *IEEE INFOCOM*, March 2000.

[9] B. Clark, C. Colbourn, and D. Johnson, “Unit disk graphs,” *Discrete Math*, vol. 86, pp. 165–177, 1990.

[10] B. Liang and Z. J. Haas, “Virtual backbone generation and maintenance in ad hoc network mobility management,” in *INFOCOM (3)*, 2000, pp. 1293–1302.

[11] E. Gafni, “Round-by-round fault detectors: Unifying synchrony and asynchrony,” in *ACM PODC*, 1998.

[12] M. Mosko and J. J. Garcia-Luna-Aceves, “A self-correcting neighbor protocol for mobile ad hoc wireless networks,” in *IEEE ICCCN*, 2002, pp. 556–560.

[13] M. Gerla and J. Tsai, “Multicluster, mobile, multimedia radio network,” *ACM Baltzer Journal of Wireless Networks*, vol. 1, no. 3, july 1995.

[14] R. Vaishampayan and J. J. Garcia-Luna-Aceves, “Efficient and robust multicast routing in mobile ad hoc networks,” in *IEEE MASS*, 2004.

[15] J. Wu and F. Dai, “Broadcasting in ad hoc networks based on self-pruning,” in *INFOCOM*, 2003.

[16] ——, “A generic distributed broadcast scheme in ad hoc wireless networks,” in *23rd ICDCS*, May 2003.

[17] F. Dai and J. Wu, “Distributed dominant pruning in ad hoc wireless ad hoc networks,” Feb 2002.

[18] W. Lou and J. Wu, “On reducing broadcast redundancy in ad hoc wireless networks,” *IEEE Transactions on Mobile Computing*, vol. 1, no. 2, April-June 2002.

[19] S.-J. Lee, M. Gerla, and C.-C. Chiang, “On-demand multicast routing protocol,” in *Wireless Communications and Networking Conference*, 1999.

[20] E. M. Royer and C. E. Perkins, “Multicast operation of the ad-hoc on-demand distance vector routing protocol,” in *MobiCom ’99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 1999, pp. 207–218.

[21] “Scalable network technologies,” 2003, qualnet simulator. http://www.scalable-networks.com.