MCEDAR: Multicast Core-Extraction Distributed Ad hoc Routing

Prasun Sinha, Raghupathy Sivakumar, Vaduvur Bharghavan
Coordinated Science Laboratory, University of Illinois at Urbana-Champaign
1308 W. Main Street, Urbana, IL 61801
{prasun, sivakumr, bharghav}@timely.crhc.uiuc.edu

Abstract— **In this paper, we present the MCEDAR (Multicast Core Extraction Distributed Ad hoc Routing) multicast routing algorithm for ad hoc networks. MCEDAR is an extension to the CEDAR architecture and provides the robustness of mesh based routing protocols and the approximates the efficiency of tree based forwarding protocols. It decouples the control infrastructure from the actual data forwarding infrastructure. The decoupling allows for a very minimalistic and low overhead control infrastructure while still enabling very efficient data forwarding.**

I. Introduction

Ad hoc networks are multihop wireless networks that are characterized by scarce resources and dynamic topologies. The inherent characteristics of ad hoc networks place two key demands on any routing protocol (unicast and multicast) designed for such environments: *efficiency* and *robustness*. The scarcity of the resources entail a very efficient low overhead protocol while the dynamic topology entails a robust protocol that does not require frequent route recomputations (or does route recomputation in a lightweight manner).

Existing approaches to unicast routing in ad hoc networks [1], [2], [3], [4] address the two demands reasonably well. However, approaches to multicast routing have addressed either efficiency or robustness but not both simultaneously. In particular, current approaches to multicasting can be classified into two based on the multicasting infrastructure that they use. While one class of approaches use a tree based infrastructure that provides high data forwarding efficiency at the expense of low robustness, the other class of approaches use a mesh based infrastructure that provides better robustness (breakage of a link on the mesh need not trigger a recomputation) at the expense of higher data forwarding overhead. Clearly, there is a need for a multicast routing approach that combines the advantages of both the tree based and the mesh based routing protocols.

In this paper, we present the MCEDAR (Multicast Core-Extraction Distributed Ad hoc Routing) multicast routing algorithm that addresses both the issues of robustness and efficiency in one framework. In a related work, we have proposed an efficient and robust unicast routing architecture for ad hoc networks called CEDAR. MCEDAR, while extending the CEDAR architecture to support multicast routing is also based on the principles that guided the CEDAR architecture (enumerated in Section 2). In addition, MCEDAR has the following key design goals:

- *To support the robustness of a mesh based routing protocol.* MCEDAR uses a mesh as the underlying mesh infrastructure. Hence, every link breakage in the underlying graph does not necessitate a reconfiguration of the infrastructure.
- *To provide the efficiency of a tree based forwarding protocol.* MCEDAR uses a unique forwarding mechanism on the mesh that creates an implicit source based forwarding tree. This ensures that although MCEDAR maintains a mesh routing infrastructure the data forwarding occurs only on a source rooted minimum height tree.
- *To decouple the control and data forwarding infrastructures.* In line with one of CEDAR's chief goals, MCEDAR decouples the control and data forwarding infrastructures. The decoupling allows for a very minimalistic control infrastructure while still enabling very efficient data forwarding.
- *To support mechanisms for efficient joins, leaves and pruning.* Besides the above two goals, MCEDAR also provides efficient mechanisms for joining, leaving and pruning to improve the overall efficiency of the multicast routing algorithm.

We believe that the uniqueness of MCEDAR lies in the fact that it provides the robustness of a mesh based multicast infrastructure while still approximating the efficiency of a tree based multicast forwarding protocol. The rest of this paper is organized as follows. Section 2 provides a high level overview of the CEDAR architecture and its components. Section 3 describes the MCEDAR multicast routing algorithm. Section 4 presents key issues that need to be addressed in MCEDAR and concludes the paper.

II. Overview of the CEDAR Architecture

CEDAR, the unicast routing protocol for ad hoc networks is designed with the following guiding principles:

(a) Route computation must be distributed because centralized routing in a dynamic network is impossible even for fairly small networks. (b) Route computation should not involve the maintenance of global state, or even significant amounts of volatile non-local state. In particular, link

state routing is not feasible for highly dynamic networks because of the significant state propagation overhead when the network topology changes. (c) As few nodes as possible must be involved in state propagation and route computation, since this involves monitoring and updating at least some state in the network. On the other hand, every host must have quick access to routes on-demand. (d) Each node must only care about the routes corresponding to its destination, and must not be involved in frequent topology updates for parts of the network to which it has no traffic. (e) Stale routes must be either avoided, or detected and eliminated quickly. (f) Broadcasts must be avoided as far as possible because broadcasts are highly unreliable in ad-hoc networks. (g) If the topology stabilizes, then routes must converge to the optimal routes, and (h) It is desirable to have a backup route when the primary route has become stale and is being recomputed.

Briefly, CEDAR achieves the above goals by establishing a "core" of the network, the core then functioning as the route management infrastructure, incrementally and intelligently propagating link state information to the nodes of the core, and performing on-demand route computation using a fully distributed algorithm in which core nodes participate using purely local state. Thus, the key components of CEDAR are: (a) Core Extraction, (b) Core Broadcast, (c) Link State Propagation, and (d) Route Computation. However, MCEDAR relies only on the first two components of the CEDAR architecture and hence we focus only on the first two components in the rest of this section.

A. Core Extraction

Ideally, the *core* comprises of the nodes in a minimum dominating set V_C of the ad hoc network $G = (V, E)$. However, we are using a robust and simple constant time algorithm which requires only local computations and generates good approximations for the MDS in the average case.

Consider a node u, with first open neighborhood $N^1(u)$, degree $d(u) = |N^1(u)|$, dominator $dom(u)$, and effective degree $d^*(u)$, where $d^*(u)$ is the number of its neighbors who have chosen u as their dominator. The *core* computation algorithm which is performed periodically, works as follows at each node u.

1. u broadcasts a beacon which contains the following information pertaining to the *core* computation: $< u, d^*(u), d(u), dom(u) >$.
2. u sets $dom(u) \leftarrow v$, where v is the node in $N^1[u]$ with the largest value for $< d^*(v), d(v) >$, in lexicographic order. Note that u may choose itself as the dominator.
3. u then sends v a unicast message including the following information: $< u, \{(w, dom(w)) \mid \forall w \in N^1(u)\} >$. v then increments $d^*(v)$.
4. If $d^*(u) > 0$, then u joins the *core*.

Essentially, each node that needs to find a dominator selects the highest degree node with the maximum effective degree in its first closed neighborhood. Ties are broken by node id.

When a node u joins the *core*, it issues a 'piggybacked broadcast' in $N^3(u)$. A piggybacked broadcast is accomplished as follows. In its beacon, u transmits a message: $< u, DOM, 3, path\_traversed = null >$. When node w hears a beacon that contains a message $< u, DOM, i, path\_traversed >$, it piggybacks the message $< u, DOM, i-1, path\_traversed + w >$ in its own beacon if $i - 1 > 0$. Thus, the piggybacked broadcast of a *core* node advertises its presence in its third neighborhood. This guarantees that each *core* node identifies its nearby *core* nodes, and can set up virtual links to these nodes using the *path_traversed* field in the broadcast messages. The state that is contained in a *core* node u is the following: its nearby *core* nodes (i.e. the *core* nodes in $N^3(u)$); $N^*(u)$, the nodes that it dominates; for each node $v \in N^*(u)$, $< \forall w \in N^1(v), < w, dom(w) >>$. Thus each *core* node has enough local topology information to reach the domain of its nearby nodes and set up virtual links. However, no *core* node has knowledge of the *core* graph. In particular, no non-local state needs to be maintained by *core* nodes for the construction or maintenance of the *core*. Note from steps 2 and 4 that over a period of time, the *core* graph prunes itself because nodes have a propensity to choose their *core* neighbor with the highest effective degree as their dominator.

Maintaining the *core* in the presence of network dynamics is very simple. Consider that due to mobility, a node loses connectivity with its dominator. After listening to beacons from its neighbors, the node either finds a *core* neighbor which it now nominates as its dominator, or nominates one of its neighbors to join the *core*, or itself joins the *core*.

B. Core Broadcast

As with most existing ad hoc networks, CEDAR requires the broadcast of route probes to discover the location of a destination node, and the broadcast of some topology information (in the form of increase/decrease waves). While most current algorithms assume that flooding in ad hoc networks works reasonably well, our experience has shown otherwise. In particular, we have observed that flooding probes, which causes repeated local broadcasts, is highly unreliable because of the abundance of hidden and exposed stations. Thus, we provide a mechanism for '*core* broadcast' based on reliable unicast (using RTS-CTS etc.). Note that it is reasonable to assume a unicast based mechanism to achieve broadcast in the *core*, because each *core* node is expected to have few nearby *core* nodes. Besides, our *core* broadcast mechanism ensures that each *core* node does not transmit a broadcast packet to every nearby *core* node. CEDAR uses a close coordination between the medium access layer and the routing layer in order to achieve efficient *core* broadcast.

Recall that a virtual link is a unicast path of length 1, 2, or 3. Recall also, that CSMA/CA protocols use a RTS-CTS-Data-ACK handshake sequence to achieve reliable unicast packet transmission. Our goal is to use the MAC state in order to achieve efficient *core* broadcast using $O(|V|)$ messages, where $|V|$ is the number of nodes in the network.

In order to achieve efficient *core* broadcast, we assume that each node temporarily caches every RTS and CTS packet that it hears on the channel for *core* broadcast packets only. Each *core* broadcast message M that is transmitted to a *core* node i has the unique tag $< M, i >$. This tag is put in the RTS and CTS packets of the *core* broadcast packet, and is cached for a short period of time by any node that receives (or overhears) these packets on the channel. Consider that a *core* node u has heard a CTS($< M, v >$) on the channel. Then, it estimates that its nearby node v has received M, and does not forward M to node v. Now suppose that u and v are a distance 2 apart, and the virtual channel $[u, v]$ passes through a node w. Since w is a neighbor of v, w hears CTS($< M, v >$). Thus, when u sends a RTS($< M, v >$) to w, w sends back a NACK back to u. If u and v are a distance 3 apart, using the same argument we will have atmost one extra message. Essentially, the idea is to monitor the RTS and CTS packets in the channel in order to discover when the intended receiver of a *core* broadcast packet has already received the packet from another node, and suppress the duplicate transmission of this packet.

1. The *core* nodes do not explicitly maintain a source-based tree. However, the *core* broadcast dynamically (and implicitly) establishes a source-based tree, which is typically a breadth-first search tree for the source of the *core* broadcast.
2. The number of messages is $O(|V_C|)$ in the average case. In particular, the only case we transmit extra data messages is when two nearby *core* nodes are a distance 3 apart.
3. Since the trees are not explicitly maintained, different messages may establish different trees. Likewise, changes in the network topology do not require any recomputation. However, the coordination of the MAC layer and the routing layer ensures that the *core* broadcast establishes a tree, and that a *core* node typically does not receive duplicates for a *core* broadcast.

While our approach for the *core* broadcast is very low overhead and adapts easily to topology changes, the RTS and CTS packets corresponding to a *core* broadcast need to be cached for some time after their reception.

In MCEDAR, the core broadcast enables the tree based forwarding protocol on the mesh infrastructure.

III. The MCEDAR Algorithm

In this section, we describe the MCEDAR multicast routing algorithm. Essentially, MCEDAR uses two of CEDAR's components, namely: the core and the core broadcast. The infrastructure for a multicast group resides entirely within the core and the core broadcast mechanism is used to perform data forwarding on the infrastructure. Briefly, for each multicast group MCEDAR extracts a subgraph of the core-graph to function as the routing infrastructure. The subgraph is a mesh structure and is called the *mgraph* for the multicast group. Once the *mgraph* is extracted for a multicast group, data forwarding is done on the *mgraph* using the core broadcast mechanism described in the previous section. MCEDAR, thus has four key components: (i) the *mgraph* which is the multicast mesh routing infrastructure, (ii) the join protocol, (iii) the core broadcast based forwarding protocol and (iv) the leaving, pruning and reconstruction protocols. The rest of this section describes in more detail each of these components.

A. *The* Mgraph *Infrastructure*

MCEDAR uses a mesh structure called the *mgraph* as its multicast routing infrastructure. The inherent redundancy present in meshes increases the robustness of the *mgraph*. Hence, recomputation of the *mgraph* may not be necessary for every link breakage. This is a property that is critical to ad hoc environments where link breakages occur often due to node mobility. However, unlike other mesh based approaches [5], MCEDAR minimizes the number of nodes in the *mgraph* by requiring only core nodes to become members of an *mgraph*. Specifically, an *mgraph* is a sub graph of the core graph and not a sub graph of the underlying network. Thus, only core nodes can become members of an *mgraph*. When a node wants to become a member of a multicast group, it requests its dominating core node to join the appropriate *mgraph* and the dominator then performs the join operation.

MCEDAR does not differentiate between sending and receiving members of a multicast group. Hence all aspects of the MCEDAR algorithm are the same for both senders and receivers of a multicast group. MCEDAR associates a *robustness factor* R with each *mgraph*. The factor R represents the degree of robustness supported by the *mgraph*. The factor R signifies how "strongly" each member of a multicast group is connected to the rest of the corresponding *mgraph*. The robustness factor R is set as a function of the dynamics in the network. We elaborate on how the robustness factor R is achieved and maintained when we explain the join protocol.

Each member of an *mgraph* maintains a notion of which of its nearby core nodes are members of the same *mgraph*. This information is used in data forwarding where incoming data at a member is forwarded onto all neighboring *mgraph* members except the one from the data arrived. *Mgraph* members also maintain a notion of a local ordering among each other. Specifically, each *mgraph* member maintains a notion of *when* it joined the *mgraph* and *when* each of its *mgraph* neighbors joined the *mgraph*. This infor-

mation is used to prevent loop formations during *mgraph* reconstructions as follows: an *mgraph* member issues a *reconstruction request* only when it loses connectivity with all of the neighboring *mgraph* members who had smaller join times. An *mgraph* member responds to a reconstruction request only if its join time is lesser than the join time of the issuer of the request. MCEDAR maintains the notion of join times through *join IDs*. More elaboration on join IDs can be found in the description of the join protocol.

B. The Join Protocol

Since the *mgraph* consists only of core nodes, only a core node is allowed to perform a join operation in MCEDAR. When a non-core node wants to become a member of a multicast group it requests its dominating core node to perform the join operation. All further uses of the word "node" in this section refer to a "core node". A node performs the join operation by core broadcasting a *join request JOIN(MA, joinID)*. The join request consists of the address of the group the node wishes to join and the current joinID of the node, corresponding to the multicast group. The joinID for a freshly joining node is set to infinity. When a node that is not a member of MA receives the join request, it forwards the message to its nearby core nodes in accordance with the core broadcast mechanism. On the other hand, when an existing member of MA receives the join request, it sends a JOIN_ACK(MA, joinID) only if its joinID is lesser than the joinID that arrives in the request. It then forwards the JOIN request further down. However, if its joinID is greater than the incoming joinID, it forwards the request like a non-member. Ties in joinIDs are broken based on nodeIDs.

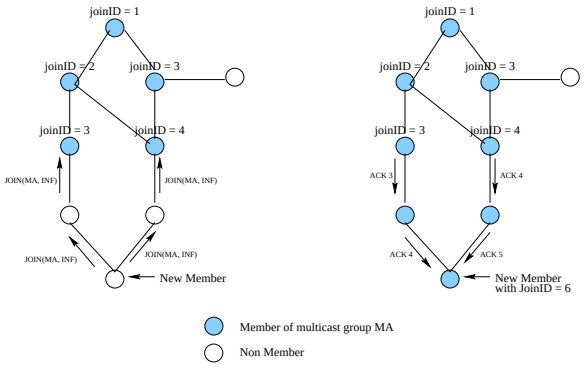

Fig. 1. The Join Protocol

The joinID in the JOIN_ACK message back to the node requesting the join is that of the replying node. When an intermediate node on the reverse path (from the replier to the requester) receives the JOIN_ACK message, it decides on whether to accept the JOIN_ACK or reject it based on the number of JOIN_ACKs it has already accepted for the particular multicast group. Specifically, an intermediate node accepts a JOIN_ACK if the number of such messages it has already accepted for that multicast group is less than R, the robustness factor. The member on accepting a JOIN_ACK sets its joinID to the maximum of its current joinID and the arriving joinID incremented by one. It then stamps the joinID of the JOIN_ACK with its new joinID. Each *mgraph* member maintains two other data structures, the *parent* set and the *child* set. When a node accepts a JOIN_ACK, it adds the upstream *mgraph* member to its parent set. Further, if the downstream node (as specified in the JOIN_ACK) is not already in its child set, it forwards the JOIN_ACK to the downstream node and adds the downstream node to its child set. On the other hand, when the intermediate node decides to reject a JOIN_ACK, it suppresses the JOIN_ACK and does an explicit leave from the upstream node so that its ID is removed from the upstream node's child set.

Every member should in the ideal case have atleast R/2 parents to maintain the robustness factor R of the *mgraph*. However, either due to the actual non-availability of paths during the initial join or due to breakages of links to parents, the cardinality of the node's parent set might be less than R/2. In such a case, the node periodically issues new join requests. The periodicity of these *background joins* is inversely proportional to the difference between R/2 and the size of the node's parent set and becomes infinity when the desired cardinality of the parent set is achieved. Figure 1 illustrates the join protocol of MCEDAR in a scenario where R = 2.

C. The Forwarding Protocol, Leaving, Pruning and Reconstructions

Although the *mgraph* for a multicast group is a mesh infrastructure, the forwarding of data on the infrastructure is done only on a source based tree, thus saving on redundant transmissions leading to efficient usage of the scarce bandwidth. The forwarding protocol of MCEDAR uses the core broadcast mechanism described in the previous section. Specifically, when a data packet arrives at an *mgraph* member, the member *attempts* to forward the data packet only to those nearby core nodes that is knows are also members of the same *mgraph*. Further, some of these attempts are suppressed by the core broadcast mechanism if it is found that a downstream member has already received the same data through a different path.

Such a forwarding protocol has two key advantages: (i) it eliminates redundant transmissions and hence saves bandwidth usage and (ii) it implicitly creates a source based tree that represents the fastest delivery structure, for each data packet of the multicast group.

A member of the *mgraph* issues a *leave* message to each of its parents it does not have any local members in its domain and its child list is empty. Since all members of the *mgraph* perform the leave operation if the two conditions are satisfied, the mesh is automatically pruned. A parent that receives the leave message from one of its children deletes the corresponding child's ID from its child

set. When a node loses connectivity with all of its parents, then there is a potential partition of the *mgraph*. In such an event, the node issues a join request with the joinID set to its current joinID (as opposed to infinity for a fresh join). Since only members that have lesser joinIDs respond to this request, the formation of partitions in the *mgraph* when the underlying graph is connected is eliminated.

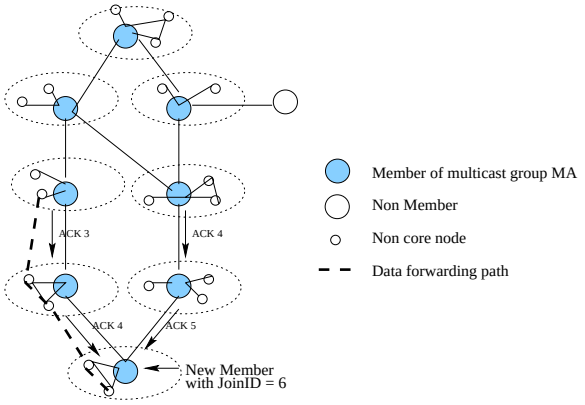

Fig. 2. Decoupling of Control and Data paths

D. Decoupling Control Infrastructure and Data Forwarding Infrastructure

Although the previous sections are based on the notion that the data forwarding also occurs on the *mgraph*, MCEDAR includes an optimization mechanism that reduces the overhead imposed on *mgraph*s in particular and the underlying core graphs in general. This optimization is performed by decoupling (as in the unicast case) the control infrastructure - the *mgraph*, from the actual data forwarding infrastructure. When a JOIN_ACK is propagated back to a newly joining node, besides performing the operations mentioned earlier, each intermediate node also computes a data path from its domain to its child's domain. It then adds information about the link from its domain to its child's domain to the JOIN_ACK it forwards downstream. It also updates the multicast routing table of the nodes in its domain appropriately. Thus eventually, when the join process is complete, besides the newly grafted sub graph of the *mgraph*, there is also a parallel data forwarding sub graph that is created alongside. Since the data forwarding infrastructure is closely tied with the *mgraph*, each node in the *mgraph* keeps track of the status of the data forwarding paths from its domain to the domains of its children and when there are breakages in any of the data paths, appropriate recomputations are done by the *mgraph* node. Figure 2 illustrates this decoupling in MCEDAR.

IV. Issues and Conclusions

In summary, the MCEDAR multicast routing algorithm consists of the following salient characteristics: (i) it uses a mesh based routing infrastructure that attempts to provide increasing robustness with increase in the network mobility, (ii) it uses a tree based forwarding protocol that approximates the optimal forwarding overhead in terms of the time taken for delivery of each data packet, (iii) it uses a notion of join times to eliminate creation of partitions in the mgraph when the underlying network is not partitioned, (iv) it has an implicit pruning mechanism that prunes the mgraph automatically, (v) it relies on the core broadcast mechanism for both control packet forwarding and data forwarding, and hence does not use the highly ineffective local broadcasts in its key mechanisms and (vi) it decouples the control infrastructure from the data forwarding infrastructure thus minimizing control overhead and maximizing data forwarding efficiency.

However, the key issue in the MCEDAR algorithm described in this paper is that of the parent child hierarchy that is maintained in an mgraph. The parent child hierarchy defines a global ordering on the join times (join IDs) of the members of the mgraph. Hence, any change in the joinID of one mgraph member can potentially create a cascade of joinID changes in the entire mgraph. We are currently working on an extension to the MCEDAR algorithm that does away with the maintenance of a global ordering on the members of an mgraph. We are also in the process of evaluating MCEDAR using simulations on ns-2 and plan to present the results in the near future.

References

[1] D. B. Johnson and D. A. Maltz, "Dynamic source routing in adhoc wireless networks," in *Mobile Computing*, (ed. T. Imielinski and H. Korth), Kluwer Academic Publishers, 1996.

[2] C. E. Perkins and E. M. Royer, "Adhoc on-demand distance vector (aodv) routing," in *Internet Draft draft-ietf-manet-aodv-03.txt*, June 1999.

[3] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proceedings of 1997 IEEE Conference on Computer Communications*, Apr. 1997.

[4] R. Sivakumar, P. Sinha, and V. Bharghavan, "Core extraction distributed ad hoc routing (cedar) specification," Internet Draft draft-ietf-manet-cedar-spec-00.txt, Sept. 1998.

[5] J. J. Garcia-Luna-Aceves and E.L. Madruga, "A multicast routing protocol for ad-hoc networks," in *Proceedings of the IEEE INFOCOM*, Mar. 1999.