Enhancing Ad hoc Routing with Dynamic Virtual Infrastructures

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— **Several routing algorithms for mobile ad hoc networks (MANETs) have been proposed in the recent past [1], [2], [3].**

With the exception of a few, these protocols (i) involve all nodes in the route management process, (ii) rely on the use of broadcast relays for route computation, and (iii) are primarily reactive in nature. Related work [4], [5] has shown that the capacity utilization in ad hoc networks decreases significantly when broadcast relays or "broadcast storms" are performed frequently. This effect is compounded when all nodes in the network take part in the route computation.

We propose and study an approach based on overlaying a virtual infrastructure (adaptation of the *core*, proposed in [3]) on an ad hoc network and operating routing protocols over the infrastructure. The core enables routing protocols to use only a subset of nodes in the network for route management and avoid the use of broadcast relays. Using the *ns-2* simulator [6] , we evaluate the performance of two ad hoc routing protocols, DSR [1] and AODV [2], when they are operated over the *core* and compare their performance against those of their basic versions.

Keywords— **Virtual Infrastructures, Ad hoc Routing.**

I. Introduction

Ad hoc networks are multi-hop wireless networks that are composed of mobile hosts communicating with each other through wireless links. These networks are typically characterized by scarce resources (e.g. bandwidth, battery power etc.), lack of any established backbone infrastructure, and a dynamic topology. A challenging but critical task that researchers have tried to address over the past few years has been the development of routing protocols that suit the characteristics of ad hoc networks. The scarcity of resources, lack of an infrastructure for performing routing, and the constantly changing topology render conventional routing protocols inappropriate for the target environment.

While several ad hoc routing protocols have been proposed in recent years [1], [2], [3], [7], [8], we choose two of these protocols, DSR and AODV, because of their prominence in the ad hoc networking research community and ready availability of *ns-2* code. While DSR and AODV have consistently been shown to perform well under varied network characteristics [4], [5], [9], the protocols share two fundamental traits that inherently bound their performance: (i) both protocols require flooding of route requests, and (ii) they perform the flooding using broadcast relays which use local broadcasts. We now briefly present intuitive reasons for why the two protocol characteristics cause performance degradation:

- *High Protocol Overhead:* The number of messages is of the order of the number of nodes in the network.
- *Unreliability of Broadcasts:* Several studies [5], [10] in the past have demonstrated the inefficacy of using local broadcasts - because of their unreliability - to convey information to all nodes in a wireless network.
- *Interference due to Broadcasts:* Since local broadcasts are not transmitted with the otherwise required RTS-CTS handshake[1] they can potentially result in collisions with the other packets (including other broadcasts) in the network resulting in an overall reduction in the network utilization.

In this paper, we study the impact of overlaying a virtual infrastructure on the ad hoc network and operating the routing protocols over the infrastructure. The infrastructure should ideally have a minimal number of nodes (subject to some constraints that we elaborate upon in the next section) and should support an efficient and robust means of propagating information to all nodes in the infrastructure. In CEDAR[3], a related work that proposes a QoS routing protocol for ad hoc networks, the authors propose a virtual infrastructure called the *core*[2] that approximates a *minimum dominating set* of the underlying network, and a QoS routing protocol that resides on the core. We propose a dynamic self-configuring infrastructure that is an adaptation of the *core* infrastructure, and schemes in which DSR and AODV can be made to operate over the core. We show through extensive simulation results that the core infrastructure enhances the performance of the two protocols under varied network characteristics.

The rest of the paper is organized as follows: In Section 2, we present an overview and characterization of AODV and DSR, and motivate the need to address the problems that arise because of network wide floods and broadcast storms. In Section 3, we present our approach by describing the core infrastructure and demonstrate how the core alleviates the fundamental problems identified in Section 2. In Section 4, we present the changes made to the basic

[1] We assume the use of an IEEE 802.11 MAC protocol in the rest of the paper. However, we discuss some MAC layer enhancements specific to our approach later in the paper.

[2] Similar approaches have also been proposed in [11], [12], [13], [14], [15], [16].

versions of DSR and AODV in order to operate them over the core infrastructure. In Section 5, we present simulation results to evaluate the performance of AODV and DSR when they operate over the core infrastructure against that of their vanilla versions and in Section 7 we conclude the paper.

II. Motivation

In this section, we first present an overview of DSR and AODV, the two ad hoc routing protocols that we consider in this work. We then characterize the protocols and identify common characteristics that limit their performance. Finally, we identify the key goals that need to be achieved to alleviate the problems.

A. Overview of Routing Protocols

A.1 Dynamic Source Routing (DSR)

DSR is an on demand routing protocol that makes use of source routing and an aggressive caching policy. Details on mechanisms of DSR can be found in [1].

Several features of DSR including on-demand route requests, source routing, and aggressive caching are desirable in ad hoc networks. On-demand routing optimizes the routing traffic by performing route computation only when necessary; Source routing precludes the need for route-loop detection mechanisms; and the aggressive caching policy is useful for limiting the number of nodes to which a route request propagates. However, DSR also suffers from the following problems [5], [9]: First, DSR floods route requests in the network using a series of local broadcasts. These local broadcasts which are transmitted as MAC broadcasts interfere with each other and with ongoing data traffic, as they are not protected using RTS and CTS control packets. Second, the aggressive caching in DSR heavily depends on source routing, which has high byte overhead and can lead to proliferated stale information. Finally, source routing and flooding in the network pose scalability concerns for large networks.

A.2 Ad hoc On-demand Distance Vector (AODV)

AODV is also a reactive protocol that computes routes only on-demand. Nodes in the network maintain distance vector tables to facilitate routing. Details on AODV mechanisms can be found in [2].

AODV's desirable features are its lower byte overheads in relatively static networks (recall that DSR stamps a source route on every data packet) and loop free routing using destination sequence numbers. However, it suffers from the problems discussed earlier: flooding of route requests and the use of MAC level broadcasts to propagate floods. In the rest of the section, we study the problems with DSR and AODV in more detail and identify the goals that need to be achieved to solve the problems.

B. Flooding and Broadcast Storms in DSR and AODV

The common properties of both the protocols which limit their performance are elaborated in the remaining section.

B.1 Flooding of Route Requests

Both DSR and AODV use network-wide floods as their base mechanism for computing routes. Although, DSR and AODV have specific mechanisms targeted towards limiting the number of nodes to which a route request is propagated, the mechanisms turn out to be effective only to a limited extent. In particular, the use of caching in DSR will be effective in limiting the area of propagation of a route request *as long as there exist active flows that pass through, originate from or are destined for the required destination and the cached information is present at enough number of nodes to prevent the route request from percolating through to a wider area of the network.* Furthermore, the caching cannot be made too aggressive since an overly aggressive caching policy can backfire in the form of proliferated stale information [9]. In AODV, the problem is more pronounced because intermediate nodes can respond to a *RREQ* message *only if they have an entry in their distance vector table for that particular destination, and a node will have an entry in its table only if a flow that originates from or is destined for that exact destination traverses the node.* Finally, the expanding ring search mechanism also has its own set of limitations including (i) an increase in route computation time due to the multi-phase route computation process, and (ii) ambiguity in computation of the timeout value for triggering a *RREQ* with a larger time to live value.

B.2 Inefficacy of Local Broadcasts

The second common characteristic that the two protocols share is the use of broadcast storms to achieve flooding which has the following problems:

- Local broadcasts are inherently unreliable because of the absence of any RTS-CTS-Data-ACK exchange, and the problem becomes even more significant when such broadcasts are performed in a series, one after the other. In [5], the authors extensively evaluate the performance of broadcast storms in an ad hoc network and demonstrate its inefficacy. [5] identifies three key issues with broadcast storms: (i) redundant transmissions by nodes after all neighbors have received a message through other paths, (ii) contention because of neighboring nodes trying to forward the same message that they just received to their respective neighbors, and (iii) collisions because of the absence of RTS-CTS-Data-ACK exchanges for broadcasts. In moderately sparse graphs the expected number of nodes in the network that will receive a broadcast message was shown to be as low as 80%. Figure 1 illustrates the unreliability of broadcasts with increase in network

load through a simulation study. The network consists of 50 nodes moving in a 1500x300m area using a random way-point model, with a pause time of 0 seconds and maximum speed of 20m/s. The simulation was run for 900s. Network wide broadcasts were issued for various number of background CBR flows, and the reachability of the broadcasts was measured. The average number of nodes reached when the network load is low (10 flows at 4 packets per second) is around 85%. However, as the network load increases, this value starts falling and reaches around 73% for 50 flows in the network.

- The second problem associated with local broadcasts is the interference they cause with the other traffic in the network. Broadcasts do not use a RTS-CTS-Data-ACK exchange and hence, can cause a considerable number of collisions. This in turn can reduce the overall network utilization. Figure 2 illustrates this phenomenon through a scenario similar to the one used for Figure 1. It shows the impact of local broadcasts on the data traffic in the network. The figure illustrates the degradation in the data delivery rate due to increase in the number of collisions with increase in the number of broadcast storms in the network. The degradation in the data delivery rate is not just due to the collisions and is in fact more seriously affected by other factors including MAC back-off because of collisions and consequent buffer overflows at the interface queue. Further, the broadcast storms cause collisions with not just data packets (see Figure 2) but also with other broadcasts.

Hence, an ideal solution would reduce the number of nodes involved in route computation and eliminate broadcast storms.

III. The Core Architecture

A. Overview

In this section, we show that overlaying a virtual infrastructure over the underlying network achieves the goals and we present the details of such an infrastructure. In a related work, [3] proposes a QoS routing protocol called CEDAR, for ad hoc networks, that employs a virtual infrastructure called the *core*[3] and uses the core for performing route computation. In CEDAR, a set of nodes is dynamically elected to form the core of the network by approximating a minimum dominating set (MDS) of the ad hoc network. The MDS is computed in a distributed fashion using only local computation and local state. Each core node maintains the local topology of the nodes in its domain, and also performs on-demand route computation on behalf of these nodes. By virtue of being a minimum dominating set, the core has fewer nodes than the underlying network. For the core computation, every node needs to send periodic beacons. [3] also proposes a broadcast scheme for the core, which however is not desirable for reasons stated later in the section. Hence, we borrow the fundamental idea of the core and its maintenance from CEDAR and propose our own core-broadcast mechanism that achieves efficient and robust floods on the core. In the rest of the section, we identify problems with CEDAR's core broadcast mechanism, and describe in detail the new core broadcast scheme that we use. Note that while CEDAR has other components including the *waves* and its own routing protocol, we do not use any of those components in this work.

[3]Similar approaches have also been proposed in [11], [12], [13], [14]. Although any of these approaches can be used as the virtual infrastructure in place of the core, the focus of this paper is merely to study the utility of virtual overlay infrastructures for ad hoc routing protocols and not to propose one particular infrastructure.

B. Core Broadcast

The CEDAR protocol has its own core broadcast mechanism. Briefly, each core node maintains an explicit tunnel with each of its "nearby" core nodes (core nodes in the 3 hop neighborhood). When it receives a core broadcast message, the core node uses the tunnels to make unicast transmissions of the message to all its nearby core nodes. However, the mechanism is not a desirable one because of the following reasons: (i) since core nodes need to know about their nearby core nodes, each core node sends periodic "core advertisement" messages that are propagated in the three hop neighborhood adding to the protocol overhead, (ii) the tunnels are set up using explicit exchange of messages between core nodes and hence contribute to the proactive overhead component. Further, static maintenance of tunnels between core nodes makes the core broadcast mechanism vulnerable to link failures, (iii) the tunnels are maintained using soft state and require periodic refreshing. If the refresh period is set to a large value (to reduce overhead), the tunnels can be stale leading to losses of core broadcast messages, (iv) since an explicit tunnel is maintained between every pair of nearby core nodes, overlapping of tunnels is possible, leading to redundant transmissions of a message.

Hence, we propose a new core broadcast mechanism in this section that has the following goals:

- *Efficiency:* The minimality of the core size implies that only a small number of nodes in the network need to be reached for every core broadcast. In addition, the number of messages required to reach the core nodes should be minimized.
- *Robustness:* The core broadcast mechanism used for reaching core nodes should be robust to link failures that can occur frequently in highly dynamic scenarios.
- *Low cost:* We have thus far discussed several problems attributed to broadcast storms. Hence, we want the core broadcast mechanism to avoid performing series of local broadcasts. Further, any computation done for achieving the core broadcast should ideally involve only local computations in order for the core broadcast

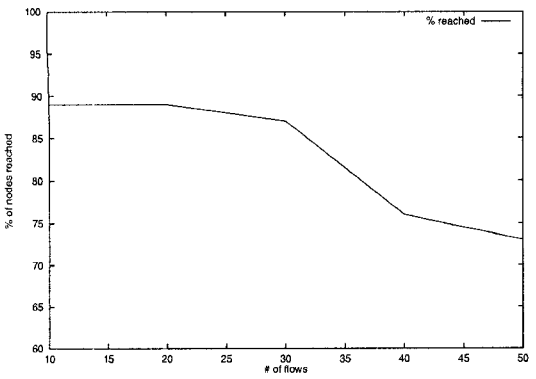

Fig. 1. Reachability of a network-wide broadcast

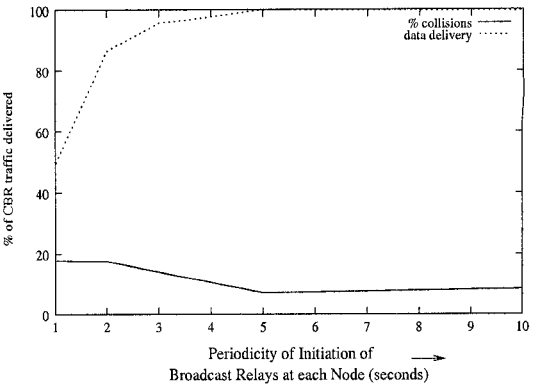

Fig. 2. Interference due to Broadcast Storms

to be scalable.

Our approach achieves the efficiency of a tree based forwarding mechanism without requiring any explicit tree or tunnel maintenance. Thus, we achieve the same efficiency as the core broadcast mechanism in [3], with a much more robust mechanism which also requires less messaging. The core broadcast traverses the links of a tree which reaches all the core nodes, using unicast messages. This tree is computed dynamically and locally using information carried in the beacons. The source node and the intermediate nodes compute a subset of its neighbor set, called the forwarding set, for forwarding a core broadcast. This set is computed dynamically and chosen in such a way that it guarantees that all core nodes in the same component of the network will receive the core broadcast. The core broadcast message is then sent individually to each of these selected neighbors in the forwarding set, using unicasts.

The algorithm for forwarding set computation and the proof of correctness are presented in the Appendix. Figure 3 illustrates the forwarding set computation by node 1, which is based on the partial topology learnt using the beacons. Computation of the subset of neighbors which will form the forwarding set is done at every node and it uses information contained in the beacons. The beacons from core nodes do not contain any core broadcast specific information, however, the ones from non-core nodes contain the list of core neighbors and the list of the dominators[4] of the non-core neighbors. Neighboring nodes 2 and 3 are included in the set as they are known to be core nodes, based on the beacons heard from them. Node 4 is included in the forwarding set as it has core neighbors (nodes 10 and 11). The core nodes which are part of the forwarding set, or are known to be reachable from the members of the forwarding set, constructed so far are said to be covered by the forwarding list. Node 5 is not included in the forwarding set as core node 11 (that it covers) is already covered by node 4. Node 6 is included as it leads to node 13 but node 7 is not included as the core node it covers (node 2), is already part of the forwarding set. Thus, nodes 2, 3, 4 and 6 form the forwarding set for node 1.

While the mechanisms described thus far achieve a broadcast on the core, we use added MAC functionality to further optimize the overheads of the core broadcast. The additional MAC functionalities include the provision for a negative CTS (NCTS) control message. This is used by nodes to avoid duplicate or unwanted reception of core broadcast messages. The MAC needs to maintain a cache of recently received core broadcast messages. Core broadcasts are identified by a unique core broadcast ID which consists of a sequence number[5] assigned by the originator of the core broadcast, and the source ID. The message id cache maintained in the MAC layer is updated based on both received and snooped (promiscuous mode) packets. This further leads to performance improvements in two cases: (a) when a node snoops out a core broadcast message targeted towards another node, it sends an NCTS later when an RTS arrives for the same message, and (b) if it is found through snooping that a neighbor has received a particular core broadcast message, the MAC layer abstains from forwarding the message to that neighbor even if the forwarding set includes the neighbor.

Since a node does not receive the same core broadcast twice, the efficiency of our core broadcast mechanism is at least as good as tree based forwarding. Further, our optimizations and promiscuous listening based core broadcast cache management mechanisms enable the core broadcast to perform even better than tree based forwarding. Unlike the core broadcast in [3], the links on which to forward the core broadcast are computed dynamically and hence is more robust to link failures. Thus we achieve the design goals of efficiency and robustness. Instead of flooding using local broadcasts, our core broadcast mechanism uses a series of unicasts, thus avoiding broadcast storms [5] in the network. As the forwarding set computation is local information based, we achieve message dissemination to all core nodes without using any global computation.

[4]see Appendix for explanation

[5]1 byte would suffice.

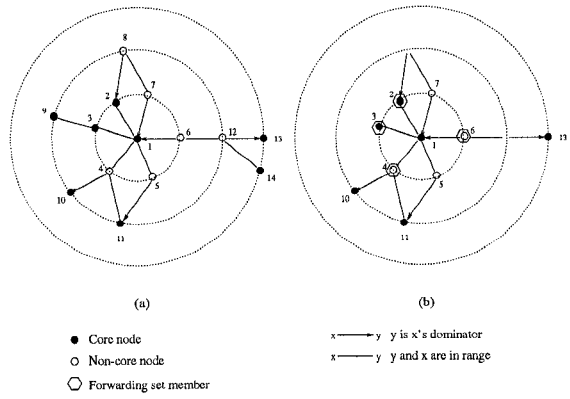

Fig. 3. Core Broadcast: Forwarding set computation for node 1. The dotted circles denote the 1st, 2nd and 3rd neighborhoods of node 1 of a large network. (a) 3 hop neighborhood of node 1 (b) Information available at node 1. Information about nodes 8, 9, 12 and 14 are not available through the beacons. Core node 13 however is known, based on the beacon heard from node 6.

IV. The Enhanced Protocols

A. DSRCEDAR

Recall from Section II-A.1 that DSR has the following key features: (i) the initiation and propagation of route requests (*RREQ*), gratuitous route replies (*GRREQ*), route replies (*RREP*) and route error (*RERR*) messages, (ii) aggressive route cache management, and (iii) source routing. DSRCEDAR layers DSR on CEDAR retaining the above features of DSR and bringing in the advantages of the core infrastructure. Like DSR, DSRCEDAR uses *RREP*, *GRREP* and *RERR* messages for route reply, gratuitous route reply and route error respectively. The route query mechanism however is based on the core broadcast, rather than a simple flooding of the *RREQ* messages. Beaconing, core computation and core broadcast, are the features infused into DSRCEDAR because of the core infrastructure. The merger alleviates some of the limitations due to flooding in DSR. The key features of DSRCEDAR which are distinct from DSR are as follows:

- *Implicit Ring Zero Search:* The information contained in the beacons allow every node to create a partial topology of up to three hops. If the destination is reachable in this partial topology, then the route obtained from this topology is the computed route. Thus the proactive information available from the CEDAR architecture enables DSRCEDAR to avoid a core broadcast in some cases. Note that the partial topology includes the neighbors of the node and hence unlike DSR, a Ring Zero Search is not needed for discovering a destination which is one hop away.
- *Core Broadcast:* When a route to the destination is not available in the cache and the proactive information has also failed to provide a route to the destination, a core broadcast is initiated for the destination, rather than a flood as in the case of DSR.
  Recall that unlike local broadcasts used in flooding, the core broadcast messages are sent as a sequence of unicasts. As discussed earlier (Section III-B), the reduced message complexity of the core broadcast results in significant savings in *RREQ* packet overhead as compared to DSR. However, this savings comes at the expense of periodic beaconing.
- *Fewer* RREP*:* In DSRCEDAR a node receives a core broadcast message only once, and therefore, any node sends at most one *RREP* per core broadcast. This is in contrast to DSR, where all route queries reaching the destination are responded to, resulting in a possible route reply flood. Although the packet overhead caused by a route reply flood could be offset by the resulting heavy caching, leading to higher cache hits, it has problems in networks with high load. In such scenarios, route error messages could be missed by promiscuous listeners resulting in stale caches and route reply floods could worsen the situation by aggressively spreading the stale information.
- *Proactive information based packet salvation:* In DSR, undeliverable packets are salvaged at intermediate nodes by using alternate routes from the route cache. In addition to this mechanism for packet salvation, DSRCEDAR benefits from information obtained from proactive beaconing. Currently, we use a simple mechanism to patch the source route, upon a link failure. On the remaining source route in the packet, we locate a node to which a path is known, based on the proactive information available through beacons. The source route in the packet is then accordingly modified. Since the route traversed is already in the packet, a repeated node in the new route indicates a loop and the packet is dropped.

B. AODVCEDAR

We refer to the version of AODV running over CEDAR as AODVCEDAR and the vanilla version of AODV as AODV in the rest of the paper. Recall from Section 2 that the AODV protocol has three key components: (i) the initiation and propagation of route request (*RREQ*) messages, (ii) initiation and propagation of route reply

(*RREP*) messages and (iii) the maintenance of the distance vector table. In AODVCEDAR, only the propagation of *RREQ* messages is different from that of AODV and the other two components are maintained as in AODV. Thus, while the propagation of *RREQ* messages on the network are done using the core broadcast mechanism, the original mechanisms of AODV come into play once the destination or an intermediate node having a route to the destination receives the *RREQ* message and replies to it.

When a source requires a route to a destination, it generates a core broadcast of the *RREQ* message. When the message reaches the destination's dominator, the dominator suppresses any further core broadcast and forwards the *RREQ* message directly to the destination. The destination then replies with an *RREP* message. The propagation of the *RREP* message is the same as in AODV. When the *RREQ* message reaches a domain in which one of the nodes has a route to the destination, the intermediate node replies with an *RREP* message as in AODV.

The propagation of *RREQ* messages using core broadcast proves to be beneficial to AODV since a significant portion of AODV's overhead stems from the propagation of its *RREQ* messages [4], [9]. We now elaborate on the benefits AODVCEDAR enjoys when operating over the core infrastructure:

- *Lower Route Request Overhead:* The propagation of *RREQ* messages over the core significantly reduces the route computation overhead. While AODV floods the *RREQ* message to a majority of the nodes in the network, AODVCEDAR restricts the *RREQ* propagation using core broadcast. Hence, AODVCEDAR saves considerably in terms of the packet overhead and byte overhead when compared to AODV. Further, the absence of broadcast storms (recall that the core broadcast uses only unicast transmissions) helps in better utilization of the network by avoiding the problems pointed out in Section 2. We show how this translates into better overall data delivery performance in the next section.
- *Implicit Zero-level Expanded Ring Search:* Like in DSRCEDAR, AODVCEDAR also performs an implicit zero-level ring search before performing a network wide core-broadcast and hence saves on protocol overhead in cases where the destination is in the neighborhood.
- *Route Salvation:* AODVCEDAR performs route salvation as in DSRCEDAR. However, AODVCEDAR lacks the source route information that DSRCEDAR has to perform route salvation. Hence, in AODVCEDAR, the core nodes maintain explicit information to perform re-routing of packets on a route failure. Core nodes maintain this information through snooped *RREP* information. Specifically, when an *RREP* message is forwarded back to the source, each intermediate core node that receives (or snoops) the message notes down the identifier of the next core node on the path to the destination. In the event of a route failure, the node upstream of the link failure sends the packet to be re-routed to its dominator. The dominator then uses its salvage information (the next hop core node for that particular destination) and core-broadcast tree information to forward the salvage packet to the next core node. Each core node then forwards the re-routed packet to the subsequent core node on the path to the destination till the packet reaches the dominator of the destination. The packet is then directly sent to the destination. Thus, AODVCEDAR utilizes the proactive information that is maintained in the core infrastructure to perform re-routing of salvaged packets.

In the next section, we evaluate DSRCEDAR and AODVCEDAR and compare their performance against those of their basic versions.

V. Simulations

In this section, we compare the performance of the enhanced protocols DSRCEDAR and AODVCEDAR against that of their vanilla versions, through simulations on the *ns-2* network simulator. We present the following set of results:

1. *Characterization of the core infrastructure:* Since the size of the core plays a significant role in the corresponding overheads in the enhanced protocols (route requests are flooded only among core nodes), in the first set of results we show the average size of the core infrastructure. We also show the average number of nodes that receive and hence, process a route request.
2. *Data delivery percentage:* In the second set of results, we present the data delivery percentages (total number of packets delivered over the total number of packets sent) of DSRCEDAR and AODVCEDAR along with that of their plain counterparts. We show that, across varied scenarios, the enhanced protocols perform better than their respective vanilla versions. We also discuss the reasons behind the improvements.
3. *Routing protocol overheads:* In the third set of results, we present the byte overheads (includes beaconing overhead) for the four routing protocols. The byte overheads are normalized with respect to the total number of received data packets and data bytes respectively. We show that, across different scenarios, the performance of the enhanced protocols have overheads similar to the vanilla versions. We then revisit the issues with DSR and AODV presented in Section 2 and briefly discuss how these issues are resolved by the core infrastructure leading to the overhead savings.
4. *End to end delay:* In the fourth set of results, we present the average end to end delay experienced by CBR packets. We show that the delay is higher in the enhanced protocols. We briefly identify the reasons

that result in the higher delay and discuss ways to reduce it.

The *ns* version used for the simulations was ns-2.1b4a and included the ad hoc wireless extensions provided by the Monarch research group at CMU. The *core protocol* was written as an independent agent in *ns* and was then interfaced with the routing protocols. Minimal changes were made to the basic routing protocols to enable them to operate over the core and the changes were restricted strictly to the modifications discussed in the previous section. The functionality of the MAC layer in the original CMU *ns* extensions was updated to include message ID caching and to send negative acknowledgments when required. The MAC layer otherwise retained the default characteristics of the IEEE 802.11 protocol. The transmission range and the channel capacity used were 250m and 2Mbps respectively. All other settings at both the MAC and the routing layers were retained as in the original distribution.

We show the simulation results for two different topologies: Topology 1 consists of a grid with dimensions 1500m $\times$ 300m and 50 nodes, while topology 2 consists of a 1500m $\times$ 1500m grid with 100 nodes. CBR traffic with a packet size of 512 bytes was used for the data traffic in all simulations. Beacons are sent out once every second. However, beacons are piggybacked onto data packets whenever possible. For each of the scenarios, we evaluate the protocols with different number of flows (10 and 20 sources) and different packet rates (1, 2, 3, 4 and 5 packets per second). Each simulation was run for a period of 450 seconds. Source-destination pairs were generated randomly and flows start randomly in the interval between 0 and 80 seconds. The mobility model used was the same as in [4], [5], [9] where nodes randomly pick a destination, move towards the destination at a speed uniformly distributed between zero and a maximum speed (maximum speed was 20 m/s for all simulations), and on reaching the destination stay there for pause amount of time[6], before repeating the whole process.

A. *The* Core *Infrastructure*

In this section, we present some results to provide an intuition to the benefits that the core brings with it to the performance of the routing protocols. We present two metrics: (i) the average number of core nodes in the network at any given instant, and (ii) the average number of nodes involved in a route computation process.

We show the average number of core nodes for two topologies (1500x300m and 1500x1500m). The pause time was varied from 0 seconds to 900 seconds for the two simulations and the simulations were run for 900s. Figure 4 shows the number of core nodes decreases with more stability. The number of core nodes in both the scenarios is a small fraction of the total number of nodes. Figure 5 shows the average percentage of nodes forwarding the route request message. The scenario used was 1500mx300m with 50 nodes and 900s simulation run. The percentages for the CEDAR versions are much lesser than that of the basic protocols. The reduction in the number of nodes involved is because of two reasons: (i) the reduced number of nodes (core nodes) to which a route request should be flooded in the worst case, and (ii) the use of MAC level suppression to perform a tree-based forwarding as opposed to a full flood.

B. *Data Delivery*

In this section, we show the percentage data delivered for the four protocols. For each topology, we use both 10 sources and 20 sources. The packet rates are varied between 1 packet per second and 5 packets per second. For larger packet rates, the network gets overloaded[4], [9] and the performance of all protocols suffer uniformly[7]. The pause times used were 0 seconds and 100 seconds respectively. Figures 6 -9 show the percentage data delivered by the four protocols in each of the scenarios.

It can be observed from the figures that the performance of the enhanced protocols improve over their basic versions uniformly in all the scenarios. Typically, the enhanced protocol that is the best in a particular scenario is the one whose basic version does better of the two basic versions. Specifically, DSR (hence DSRCEDAR) does better than AODV in the more dynamic scenario (pause time 0 seconds), while AODV (hence AODVCEDAR) does better than DSR in the more static scenarios. However, an interesting point to note is that the difference between the two enhanced protocols is considerably decreased when compared to the difference between the basic versions.

The improvement in the data delivery is because of three main reasons: First, one or two hop routes, which are obtained from the partial topology learnt from beacons, eliminate routing overhead to some destinations. Further, the partial topology is used for packet salvation. Second, since the route request is not flooded in the core enhanced protocols, the route response time is much lesser, thus reducing queuing time and improving performance. Additionally, in DSRCEDAR, the RREP is not flooded as in DSR. Third, the series of local broadcasts initiated by the route requests in DSR and AODV could collide with ongoing data transmissions lowering the performance of the basic protocols. The core enhanced protocols are not based on flooding, and hence have higher throughput.

[6]We would like to note here that extensive simulations were also done with other settings (packet sizes of 64, 256 and 1024 bytes, pause times of 25 and 50 seconds etc.) besides the ones presented here. While the results were similar in nature to the ones presented here, they have been excluded for lack of space

[7]However, we are currently working on improving the performance of AODVCEDAR and DSRCEDAR even in the case of overloaded scenarios. We provide some ideas on this in the next section.

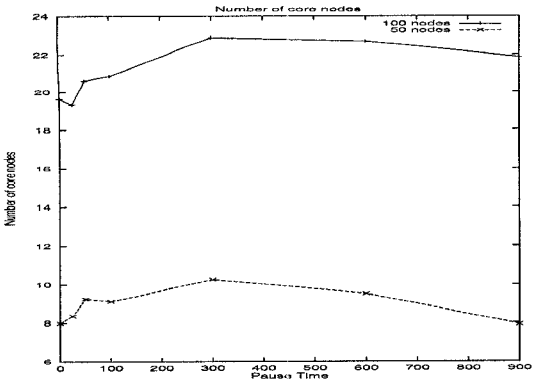

Fig. 4. Average Number of Core Nodes

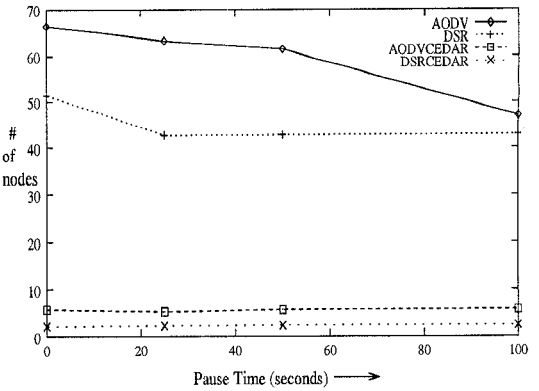

Fig. 5. Average Number of Messages Per Route Request

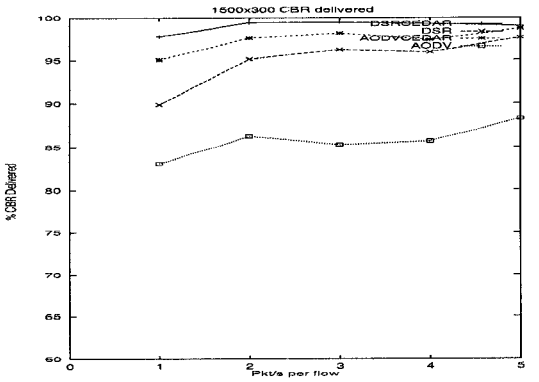

Fig. 6. 1500x300m: Data Delivery: 10 flows

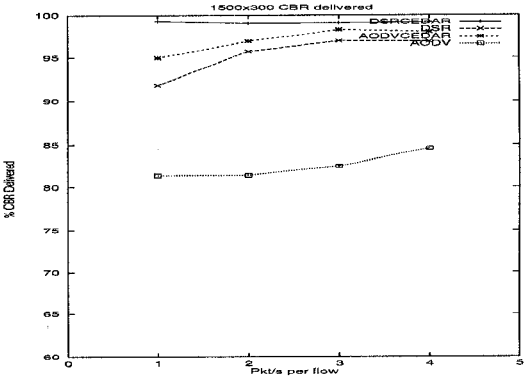

Fig. 7. 1500x300m: Data Delivery: 20 flows

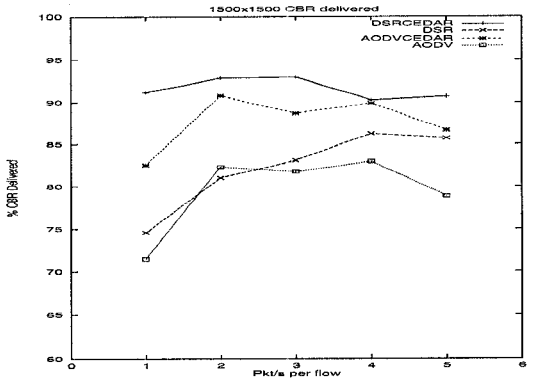

Fig. 8. 1500x1500m: Data Delivery: 10 flows

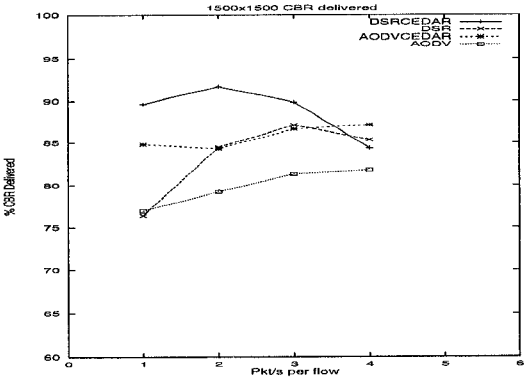

Fig. 9. 1500x1500m: Data Delivery: 20 flows

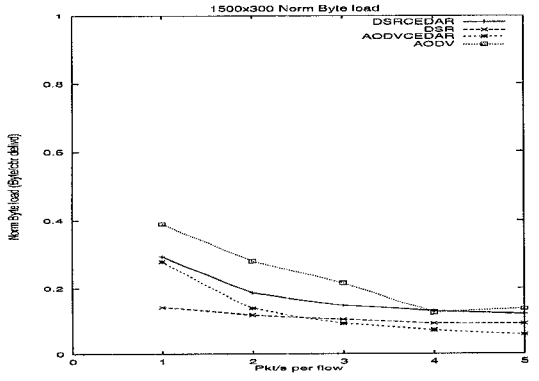

Fig. 10. 1500x300m: Byte Overhead: 10 flows

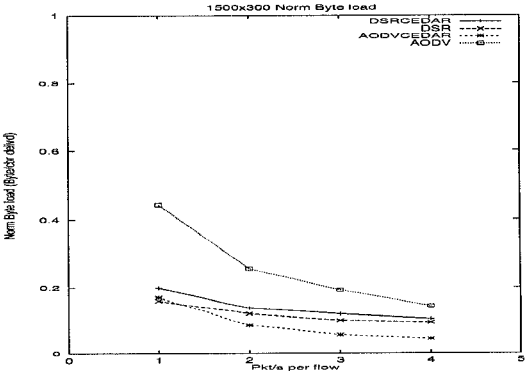

Fig. 11. 1500x300m: Byte Overhead: 20 flows

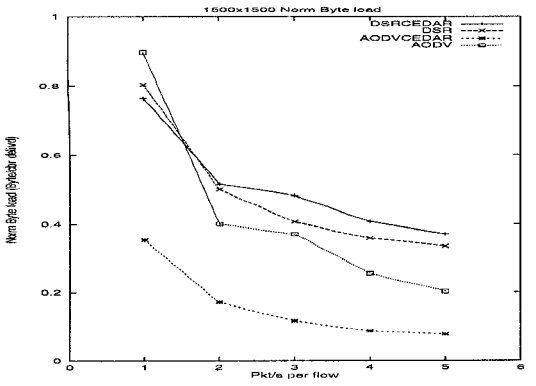

Fig. 12. 1500x1500m: Byte Overhead: 10 flows

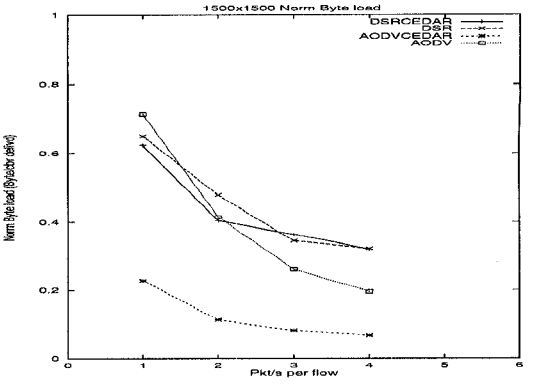

Fig. 13. 1500x1500m: Byte Overhead: 20 flows

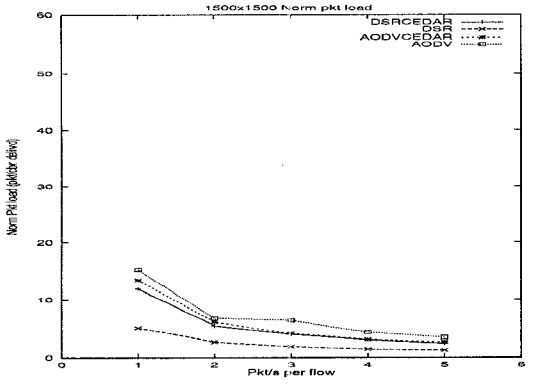

Fig. 14. 1500x1500m: Packet Overhead: 10 flows

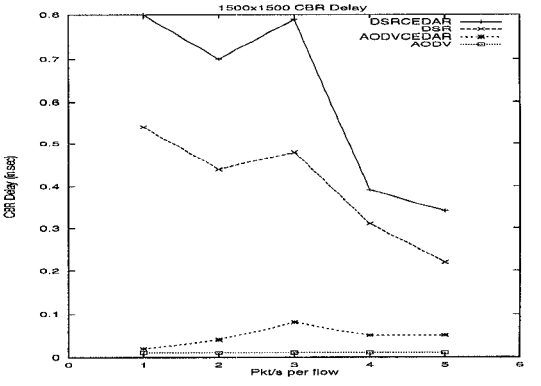

Fig. 15. 1500x1500m: End to End Delay: 10 flows

C. Protocol Overhead

In this section, we present the overheads for the four routing protocols in the scenarios that were described in the previous section. Although both packet and byte overheads were measured, due to lack of space we show only the byte overhead for all the scenarios in Figures 10-13 and show a representative plot for packet overhead results (Figures 14).

In Figures 10-13, AODVCEDAR performs significantly better than AODV (upto 50% in the more dynamic scenarios). The reasons are twofold: (i) AODV's route request is typically a flood of the network. However, in AODVCEDAR, the route request is flooded only among core nodes, and (ii) the flood among core nodes is achieved through the core broadcast mechanism that uses only unicast transmissions and does a tree based broadcast. Hence, the performance gains of AODVCEDAR is chiefly constituted by the savings in the propagation of route request messages. Since more number of route requests are generated in more mobile scenarios, the performance difference widens with increasing mobility. DSRCEDAR on the other hand performs worse than DSR in the 1500x300m scenario, and comparable with DSR in the 1500x1500m scenario. The reason for the absence of a significant improvement in the performance overhead of DSRCEDAR over DSR, is the aggressive caching policy that DSR employs. However, the aggressive caching policy turns out to be counterproductive in highly mobile scenarios (see Figures 6-9) where the cached information is more often than not stale and as a result of which DSR's data delivery percentage suffers. The packet overhead graph shown here and other studies show that the packet overhead of CEDAR enhanced protocols are comparable to that of the vanilla versions.

D. End to End Delay

The fourth set of results we present is the average end to end delay of CBR packets in the network. Figure 15 shows the average end to end delay experienced by CBR packets for each of the four protocols. The enhanced protocols show an increased end to end delay for CBR packets. The increase in delay is because of two reasons: (i) the enhanced protocols use unicast transmissions for their route request messages. Hence, each route request transmission undergoes an RTS-CTS-Data-ACK handshake and consequently takes more time and in the process delays transmission of packets further back in the interface queue, and (ii) there is no explicit decoupling of the data path from the core in the current implementation of the enhanced protocols. Consequently, core nodes typically participate in more number of routes and hence have a larger queue buildup leading to larger end to end delays.

We are currently experimenting with some preliminary measures to address the increase in end to end delay. The

measures include: (i) explicit decoupling of the data path from the core, (ii) load balancing in the network, and (iii) a lightweight core broadcast mechanism that requires fewer transmissions per core broadcast.

VI. Conclusions

Several infrastructure-less ad hoc routing protocols, such as DSR and AODV, involve all the nodes in the network for routing and use flooding as the principal mechanism for route querying. Flooding, which is typically achieved through repeated local broadcasts is plagued with problems such as, high overhead, low reachability and collisions with other data packets.

We present an approach that attempts to alleviate flood related limitations of ad hoc routing protocols by overlaying the core infrastructure [3] on the underlying network. The core broadcast mechanism presented is substantial modified from [3] and it achieves an efficient, robust, and low overhead mechanism to flood on the core. Through extensive simulations using *ns-2*, we demonstrate the effectiveness of the core in enhancing the performance of the routing protocols. In most cases we have observed up to 10% improvement in packet delivery percentage. While we observe significant improvements for packet and byte overheads in dynamic scenarios, the overheads are comparable in more static networks.

References

[1] J. Broch, D. B. Johnson, and D. A. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks," Internet Draft draft-ietf-manet-dsr-03.txt, Oct. 1999.

[2] C. E. Perkins, E. M. Royer, and Samir Das, "Ad Hoc On Demand Distance Vector (AODV) Routing," Internet Draft draft-ietf-manet-aodv-04.txt, Oct. 1999.

[3] R. Sivakumar, Prasun Sinha, and V. Bharghavan, "CEDAR: a Core-Extraction Distributed Ad hoc Routing algorithm," *IEEE Journal on Selected Areas in Communications (Special Issue on Ad-hoc Routing)*, vol. 17, no. 8, pp. 1454–1465, Aug. 1999.

[4] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," in *Proc. IEEE MOBICOM*, Dallas, TX, Oct. 1998.

[5] P. Johansson, T.Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks," in *Proc. IEEE MOBICOM*, Seattle, Aug. 1999, pp. 195–206.

[6] K. Fall and K. Vardhan, "*ns* notes and documentation," available from http://www-mash.cs.berkeley.edu/ns/, 1999.

[7] S. Murthy and J. J. Garcia-Luna-Aceves, "A Routing Protocol for Packet Radio Networks," in *Proceedings of ACM SIGCOMM '97*, Cannes, France, Sept. 1997.

[8] V. D. Park and M. Scott Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," in *Proceedings of IEEE INFOCOM*, Kobe, Japan, Apr. 1997.

[9] S. R. Das, C. E. Perkins, and E. M Royer, "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks," in *Proc. IEEE INFOCOM*, Tel-Aviv, Israel, Mar. 2000.

[10] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A medium access protocol for wireless LANs," in *Proceedings of ACM SIGCOMM '94*, London, England, Aug. 1994.

[11] P. F. Tsuchiya, "The landmark hierarchy: A new hierarchy for routing in very large networks," in *ACM SIGCOMM '88*, Stanford, California, 1988, pp. 35–42.

[12] W. T. Tsai, C. V. Ramamoorthy, W. K. Tsai, and O. Nishiguchi, "An adaptive hierarchical routing protocol," *IEEE Transactions on Computers*, vol. 38, no. 8, pp. 1059–1075, Aug. 1989.

[13] Z. J. Haas and B. Liang, "Virtual Backbone Generation and Maintenance in Ad Hoc Network Mobility Management ," in *Proc. IEEE INFOCOM*, Tel-Aviv, Israel, Mar. 2000.

[14] J. Sharony, "A mobile radio network architecture with dynamically changing topology using virtual subnets," *MONET*, vol. 1, no. 1, pp. 75–86, 1996.

[15] N. Shacham and J. Westcott, "Future directions in packet radio architectures and protocols," *Proceedings of the IEEE*, vol. 75, no. 1, pp. 83–99, Jan. 1987.

[16] J. Jubin and J. D. Tornow, "The DARPA packet radio network protocols," *Proceedings of the IEEE*, vol. 75, no. 1, pp. 21–32, Jan. 1987.

APPENDIX A

Forwarding set computation for Core Broadcast: By virtue of beaconing, each node knows all its core neighbors. All such nodes are included in the forwarding set. The non-core neighbors are added to the forwarding set if one of the following two conditions is satisfied:

- The non-core node has a core neighbor which is not *covered* by the forwarding list. The core nodes which are part of the forwarding set, or are known to be reachable from the members of the forwarding set, constructed so far, are said to be covered by the forwarding list.
- The non-core neighbor has a non-core neighbor whose dominator is not covered by the forwarding list.

The core broadcast cache is now used to delete nodes from the forwarding set which are known to have received this particular core broadcast. The updated forwarding set is then inserted in the core broadcast message which is used while promiscuously listening to core broadcast messages, as explained earlier. The correctness of the core broadcast mechanism relies on this forwarding set computation, and has been proved[8].

[8]Proof omitted due to space constraints.