Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

ELSEVIER

Computer Networks 41 (2003) 687–706

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Braving the broadcast storm: infrastructural support for ad hoc routing

Raghupathy Sivakumar [a,*], Prasun Sinha [b], Vaduvur Bharghavan [c]

[a] *School of Electrical and Computer Engineering, Georgia Institute of Technology, 777 Atlantic Drive, Atlanta, GA 30332-0250, USA*

[b] *Bell Labs, Lucent Technologies, Murray Hill, NJ 07974, USA*

[c] *Meru Networks, Santa Clara, CA 95054, USA*

Received 12 November 2001; received in revised form 26 June 2002; accepted 6 November 2002

Responsible Editor: B. Yener

Abstract

Several routing algorithms for mobile ad hoc networks have been proposed in the recent past [Broch et al., The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, Internet Draft draft-ietf-manet-dsr-03.txt, October 1999; Perkins et al., Ad Hoc On Demand Distance Vector (AODV) Routing, Internet Draft draft-ietf-manet-aodv-04.txt, October 1999; Haas and Pearlman, The Zone Routing Protocol (ZRP) for Ad Hoc Networks, Internet Draft draft-zone-routing-protocol-01.txt, August 1998; IEEE J. Select. Areas Commun. 17 (8) (1999) 1454]. 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 [Broch et al., Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, Proceedings of IEEE MOBICOM, Dallas, TX, October 1998; Johansson et al., Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks, Proceedings of IEEE MOBICOM, Seattle, August 1999] 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 [IEEE J. Select. Areas Commun. 17 (8) (1999) 1454]) 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. We evaluate the performance of dynamic source routing (DSR) [Broch et al., The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, Internet Draft draft-ietf-manet-dsr-03.txt, October 1999] and AODV [Perkins et al., Ad Hoc On Demand Distance Vector (AODV) Routing, Internet Draft draft-ietf-manet-aodv-04.txt, October 1999], when they are operated over the core and compare their performances against those of their basic versions. Through extensive simulations using *ns-2* [Fall and Vardhan, *ns* notes and

* Corresponding author. Tel.: +1-404-3852257; fax: +1-404-8947883.

E-mail addresses: siva@ece.gatech.edu (R. Sivakumar), prasunsinha@dnrc.bell-labs.com (P. Sinha), bharghav@merunetworks.com (V. Bharghavan).

1389-1286/03/$ - see front matter Published by Elsevier Science B.V.
doi:10.1016/S1389-1286(02)00423-1

documentation, available from http://www-mash.cs.berkeley.edu/ns/, 1999], we show that using a virtual infrastructure significantly improves the performance of both DSR and AODV, in terms of data delivery and routing overhead, under varied network characteristics.
Published by Elsevier Science B.V.

Keywords: Ad hoc networks; Virtual infrastructure; Broadcast storm

1. Introduction

Ad hoc networks are multihop 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 [1–4,8–10], we choose two of these protocols, dynamic source routing (DSR) and ad hoc on demand distance vector (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 [5,6,11], the two 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 give an intuition on 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 [6,12] 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. Specifically, [6] shows that in the moderately connected graphs, the fraction of nodes in the network that receive a flood goes down to around 80% in the worst case. This figure can be expected to go down even more in sparsely connected topologies. As a result, route requests might not reach the destination in a single flooding attempt resulting in multiple floods. In the rest of the paper, we borrow from terminology introduced in [6] and refer to broadcast relays or series of local broadcasts as *broadcast storms*.
- *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 Section 2) and should support an efficient and robust means of propagating information to all nodes in the infrastructure. In CEDAR [4], a related work that proposes a QoS routing protocol for ad hoc networks, the authors propose a virtual infrastructure called the *core* that approximates a *minimum dominating set* of the underlying network, and a QoS routing protocol that resides on the core. We

[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.

propose a dynamic self-configuring infrastructure that is an adaptation of the core infrastructure and schemes wherein 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. Note that several approaches for virtual infrastructures in ad hoc network have been proposed in related work [13–18]. Although any of these approaches can be used as the virtual infrastructure in place of the core, the focus of this paper is to study the utility of virtual overlay infrastructures for ad hoc routing protocols and not to propose one particular infrastructure.

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 versions of DSR and AODV in order to operate them over the core infrastructure. In Section 5, we present simulation results to evaluate the performances of AODV and DSR when they operate over the core infrastructure against that of their vanilla versions. In Section 6, we discuss several issues related to the core infrastructure approach, and in Section 7 we conclude the paper.

2. 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.

2.1. Overview of routing protocols

2.1.1. Dynamic source routing

DSR is an on demand routing protocol that makes use of source routing and an aggressive caching policy. Each node maintains a hop-by-hop route to other nodes in the network. When a route to a desired destination is not available in the cache, a route request (*RREQ*) flood is initiated by the source. If a node receiving a *RREQ* is not the destination or does not have a cached route to the destination, it forwards the *RREQ* using a local broadcast after stamping the packet with its ID. Otherwise, it sends a route reply (*RREP*), containing the complete discovered route, as a unicast message to the source by reversing [2] the path traversed by the *RREQ*.

On a link failure, a route error (*RERR*) message is unicast to the source. Upon receipt of the *RERR* message, the source initiates a route re-computation. The nodes that forward or snoop out the *RERR* packet also learn of the failed link and accordingly flush their caches. Cache purging in DSR, thus heavily relies on the *RERR* packets. To optimize the network traffic, DSR also attempts to reduce the route lengths for ongoing flows using gratituous replies (*GRREP*). A *GRREP* is initiated when a node promiscuously listens to an ongoing transmission and discovers the existence of a shorter route to the same destination through it.

Several features of DSR including on demand request, 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 *RREQ* propagates.

However, DSR also suffers from the following problems [6,11]: 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,

[2] We are assuming bidirectional links.

source routing and flooding in the network pose scalability concerns for large networks.

2.1.2. Ad hoc on demand distance vector

AODV is also a reactive protocol that computes routes only on demand. Nodes in the network maintain distance vector tables to facilitate routing. Corresponding to every routing entry, a node also maintains the list of "predecessor" nodes that it knows use that link in one of their routes. When a source A needs a route to a destination B, it broadcasts a *RREQ* message that is flooded throughout the network. Each node that receives the *RREQ* first updates its routing table with the new route information to the source. It then checks to see if it is the specified destination or if it already has a route to the destination in its distance vector table. In both cases, the node replies to the *RREQ* by sending a unicast *RREP* message to the source.

When a link breaks in the network, the node upstream to the link breakage sends a *GRREP* message with distance metric set to ∞ to each of the predecessor nodes that it knows uses the link for one of its routes. Each intermediate node that receives the *GRREP* deletes the appropriate entry from its routing table and forwards the message upstream in a similar fashion. When a source receives such a *GRREP*, it computes a new route by initiating a new *RREQ* message.

AODV, like DSR employs a one level expanding ring search before initiating a network wide flood. In addition, similar to DSR where intermediate nodes reply if they have cached information, in AODV, nodes reply on receiving a *RREQ* if they have a route to the destination in their distance vector table. These two mechanisms in tandem attempt at limiting the *RREQ*s from becoming network wide floods.

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 the DSR and AODV in more detail and identify the goals that need to be achieved to solve the problems.

2.2. Flooding and broadcast storms in DSR and AODV

Although the specific mechanisms used by the two routing protocols are different, the protocols, by virtue of belonging to the common genre of *reactive* protocols, do exhibit some common characteristics, which are elaborated in the remaining section, that however limit their performance.

2.2.1. Flooding of route requests

Both DSR and AODV use network-wide floods as their base mechanism for computing routes. Although, both DSR and AODV have specific mechanisms targeted towards limiting the number of nodes to which a *RREQ* 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 *RREQ* 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 *RREQ* 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 [11]. 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) the 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. As briefly mentioned before, the current version of the protocol specifications [1,2] specify only one level of expanding ring search (querying the next hop

neighbors) before a network-wide query is initiated.

2.2.2. 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 [6], the authors extensively evaluate the performance of broadcast storms in an ad hoc network and demonstrate its inefficacy. Ref. [6] 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%. Fig. 1 illustrates the unreliability of broadcasts with increase in network load through a simulation study. The network consists of 50 nodes moving in a 1500 × 300 m area using a random waypoint model, with a pause time of 0 s and maximum speed of 20 m/s. The simulation was run for 900 s. 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. Fig. 2 illustrates this phenomenon through a scenario similar to the one used for Fig. 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 backoff because of collisions and consequent buffer overflows at the interface queue. Further, the broadcast storms cause collisions with not just data packets (see Fig. 2) but also with other broadcasts.

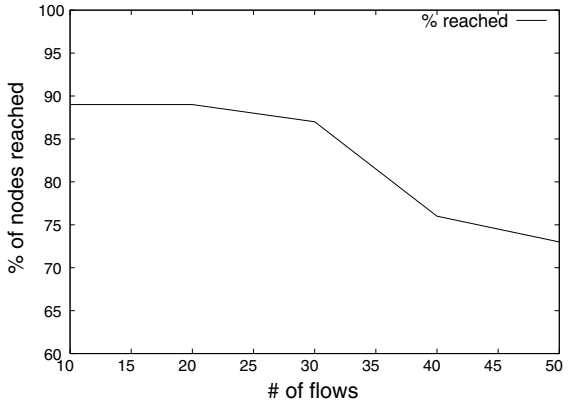

Fig. 1. Reachability of a network-wide broadcast.

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

3. The core architecture

3.1. 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, [4] proposes a QoS routing protocol called CEDAR, for

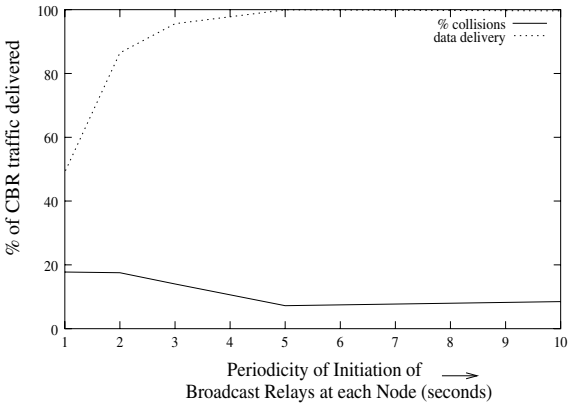

Fig. 2. Interference due to broadcast storms.

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 distributedly and dynamically elected to form the core of the network by approximating a minimum dominating set of the ad hoc network 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. [4] 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 (CB) mechanism that achieves efficient and robust floods on the core. Eliminating the details, we now briefly summarize the core computation and the CB mechanisms as they were proposed in [4].

Core computation and maintenance: The key idea of the algorithm is to maintain an approximate a set of dominators using only local computation. Each node transmits periodic beacons that contain the node ID, the number of neighbors, the number of dominated neighbors (which is 0 if it is not a core node) and the ID of the current dominator. The information in the beacons received from each neighbor is then used to compute its dominator (which could be itself). The algorithm prefers nodes that are already dominators and are also dominating a large number of nodes. In case there are no dominators in the neighborhood, it picks the node with the highest degree. Other details of the computation can be found in [4]. The dominator ID is then sent in the next beacon. If a node receives a beacon identifying the receiver as its dominator, the receiver marks itself as a core node. This computation is repeated whenever the link to the dominator breaks or if the node finds that no one besides itself has chosen it as a dominator.

Core broadcast: It is a mechanism for disseminating a message from any node to all the core nodes in the network. Each core node maintains an explicit tunnel with each of its "nearby" core nodes (core nodes in the three hop neighborhood). The originator hands over the CB message to its dominator. The dominator (core node) uses the tunnels to make unicast transmissions of the message to all its nearby core nodes. In order to eventually disseminate the CB to all core nodes in the network, it suffices to ensure that the algorithm guarantees that it reaches all core nodes within three hops (see Appendix for the proof). However, the mechanism proposed for CB, 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 CB 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

[3] Similar approaches have also been proposed in [13–16]. 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.

losses of CB 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.

In the rest of the section, we describe in detail the new CB scheme that we use. Note that while CEDAR has other components including *waves* and its own routing protocol, we do not use any of those components in this work.

3.2. New core broadcast

We propose a new CB 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 CB. In addition, the number of messages required to reach the core nodes should be minimized.
- *Robustness:* The CB mechanism used for reaching core nodes should be robust to link failures that can occur frequently in highly dynamic scenarios. Barring a partition, ongoing core broadcasts should still be completed.
- *Low cost:* We have thus far discussed several problems attributed to broadcast storms. Hence, we want the CB mechanism to avoid performing series of local broadcasts. Further, any computation done for achieving the CB should ideally involve only local computations in order for the CB 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 similar efficiency as the CB mechanism in [4], with a much more robust mechanism that also requires less messaging.

The originator of the CB computes a subset of neighbors such that forwarding only to those nodes rather than all the neighbors will suffice for the CB. This set is called the *forwarding set* and its computation requires passing some extra information in the beacons. The originator then sends the CB packet using unicasts to each forwarding set member. The recipients also compute their forwarding sets and repeat the same process. MAC layer optimizations along with promiscuous listening significantly reduce the size of the forwarding set. A smart forwarding set computation algorithm and the MAC layer optimizations ensure that the CB packet traverses the links of a tree that reaches all the core nodes, using unicast messages. This implicit tree is computed dynamically and locally using information carried in the beacons. The algorithm used for the forwarding set computation and the MAC layer optimizations are presented below.

Forwarding set computation: This subset of neighbors is computed by every node and is used for forwarding the CB. For computation and maintenance of the set of core nodes, periodic beaconing is used as mentioned above. In addition, these beacons also contain information that is used to compute the forwarding set at each node. Assuming that the network has become stable i.e., the topology is not changing any more, the algorithm for the forwarding set computation must guarantee that every core node in the network will eventually get the CB. As mentioned above and proved in the Appendix, the algorithm only needs to ensure that the CB reaches all core nodes within three hops.

The beacons from the core nodes do not contain any additional information for the forwarding set computation. However, the beacons from the non-core nodes contain the following additional information:

1. List of core neighbors.
2. List of dominators of non-core neighbors.

Both these pieces of information can be deduced from the core computation specific information in beacons from neighbors as they contain the ID of sender's dominator.

By receiving beacons from the neighbors, a node learns the following about nodes up to three hops away:

- *at one hop:* all the nodes at one hop.
- *at two hops:* all core nodes reachable through a non-core node at one hop.
- *at three hops:* all the core nodes that are reachable by hopping through two non-core nodes

and in addition, are also the dominators of the non-core node two hops away.

Based on the information contained in the beacons, each node learns a partial topology of its three hop neighborhood. It then computes a shortest path sub-tree on the partial topology graph such that the sub-tree reaches all core nodes. Ideally a sub-tree with the minimum number of internal nodes reaching all the core nodes in the partial topology must be computed. The next hops on the computed sub-tree is the set of nodes where the CB should be forwarded and hence, comprises the forwarding set. Observe that the tree is computed dynamically and is never stored in the packet. The subsequent nodes again compute the tree based on their information. Local computation of the next hops at every node adds robustness to the algorithm which is therefore able to tolerate inaccuracies in the information about the partial topology. This algorithm ensures that every node within three hops will receive the CB (see proof in Appendix). As discussed before, reachability to core nodes within three hops suffices to show reachability to all core nodes in the network, thus, proving the correctness of our new CB mechanism.

We now illustrate the forwarding set computation with an example. Fig. 3 shows the forwarding set computation for node 1. The three-hop network topology around node 1 is shown in Fig. 3(a). Based on the beacons heard from the neighbors, the partial topology learnt by node 1 is shown in Fig. 3(b). A tree to reach all the core nodes is computed which has nodes 2, 3, 4 and 6 as the next hops. It is easy to verify from the figure that this subset of neighbors is sufficient to reach all the core nodes and neighbors 5 and 7 are not needed as the next hops. Thus, nodes 2, 3, 4 and 6 comprise the forwarding set. Observe that IDs of some nodes such as 8 and 12 are not known however links to/from might be known.

MAC layer optimizations: While the mechanisms described thus far achieve a broadcast on the core, we use added MAC functionality to further optimize the overheads of the CB. The MAC layer maintains a cache (CB-cache) of recently received CB messages. Core broadcasts are

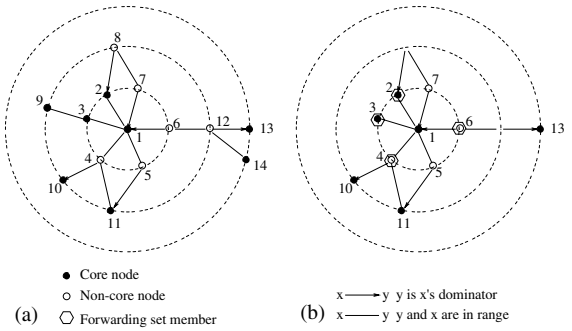

Fig. 3. Core broadcast: forwarding set computation for node 1. The dotted circles denote the 1st, 2nd and 3rd neighborhoods of node 1 of a large network. (a) three-hop neighborhood of node 1, (b) information available at node 1. Information about nodes 8, 9, 12 and 14 are not available through the beacons.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

identified by a unique core broadcast ID (CBID) which consists of a sequence number [4] assigned by the originator of the CB, and the source ID. Information about CB packets received by neighbors is also maintained in this cache.

The MAC layer will support negative CTS (NCTS). The provisioning of NCTS allows nodes to prevent duplicate reception of CB packets. The RTS packet therefore needs to contain the CBID that uniquely identifies the CB for which the RTS is sent. On reception of an RTS packet for a CB, the node sends a CTS except in the following cases when it sends an NCTS:

- *Duplicate reception:* The CB could have been received either directly from a node or by promiscuous listening. The latter requires the nodes to operate in promiscuous mode. The CB packet needs to carry the forwarding set. On receiving a CB packet promiscuously, the node checks the forwarding set stamped on the packet. If it is an intended recipient, the CB packet is accepted. As the CB is sent to all forwarding set members using separate unicasts, the node will eventually hear an RTS for the same CB. This will be responded with an NCTS. Note that the

[4] 1 byte would suffice.

CB-cache will be updated whether the packet is received directly or promiscuously.
- *Unwanted reception:* If the intended recipient of a CB is a non-core node that has a *null* forwarding set, then an NCTS will be used to avoid transmission of the CB packet. The CB is only intended for core nodes and hence this node does not need to receive it.

The CB-cache also helps in eliminating neighbors from the forwarding set if they are known to have already received the packets. The reduced forwarding set is stamped in the CB packet instead of the full forwarding set for aiding in promiscuous listening of CB packets as explained above.

Since a node does not receive the same CB twice, the efficiency of our CB mechanism is at least as good as tree based forwarding. Further, our optimizations and promiscuous listening based CB cache management mechanisms enable the CB to perform even better than tree based forwarding. This is achieved by promiscuous listening and NCTS capabilities that eliminate unicasts on some tree links. Unlike the core broadcast in [4], the links on which to forward the CB are computed dynamically and hence is more robust to link failures. Thus we achieve the design goals of efficiency and robustness. Instead of local broadcast based flooding, our CB mechanism uses a series of unicasts, thus avoiding flooding 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. The enhanced protocols

4.1. DSRCEDAR

Recall from Section 2.1.1 that DSR has the following key features: (i) the initiation and propagation of *RREQ*, *GRREQ*, *RREP* and *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, gratituous route reply and route error respectively. The route query mechanism however is based on core broadcast, rather than flooding of the *RREQ* messages. Beaconing, core computation and CB 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 CB 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 CB is initiated for the destination, rather than a flood as in the case of DSR. Recall that unlike local broadcasts used in flooding, the CB messages are sent as a sequence of unicasts. As discussed earlier (Section 3.2), the reduced message complexity of CB compared to a flood, 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 CB message only once, and therefore, any node sends at most one *RREP* per CB. This is in contrast to DSR, where all route queries reaching the destination are responded to, resulting in a possible *RREP* flood. Although the packet overhead caused by a *RREP* 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, RERR messages could be missed by promiscuous listeners resulting in stale caches and *RREP*

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.

4.2. 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 *RREQ* messages, (ii) initiation and propagation of *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 CB 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. In the event 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 CB proves to be beneficial to AODV since a significant portion of AODV's overhead stems from the propagation of its *RREQ* messages [5,11]. 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 CB. 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 CB 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 Section 5.
- *Implicit zero-level expanded ring search:* Like in DSRCEDAR, AODVCEDAR also performs an implicit zero-level ring search before performing a network-wide CB 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 rerouting 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 rerouted to its dominator. The dominator then uses its salvage information

(the next hop core node for that particular destination) and CB tree information to forward the salvage packet to the next core node. Each core node then forwards the rerouted 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 rerouting of salvaged packets.

In Section 5, we evaluate DSRCEDAR and AODVCEDAR and compare their performances against those of their basic protocols.

5. 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 [7]. 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 *RREQ*.
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 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 packet and byte overheads (includes beaconing overhead) for the four routing protocols. The packet and byte overheads are normalized with respect to the total number of received data packets and data bytes respectively. Again, we show that, across different scenarios, the enhanced protocols consistently save on the overheads. 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 acknowledgements 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 250 m and 2 Mbps 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 three different topologies: topology 1 consists of a grid with dimensions 1500 m $\times$ 300 m and 50 nodes, while topology 2 consists of a 2200 m $\times$ 600 m grid with 100 nodes. [5] The third topology consists of a 1500 m $\times$ 1500 m square grid with 50 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 over data packets whenever possible. For each of the scenarios, we evaluate the

[5] The first two topologies are the same as in [1,11].

protocols with different number of flows (10 and 20 sources) and different packet rates (1–5 packets per second). Each simulation was run for a period of 450 s. Source–destination pairs were generated randomly and flows start randomly in the interval between 0 and 80 s. The mobility model used was the same as in [5,6,11] 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]

5.1. 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 (1500×300 m and 2200×600 m). The pause time was varied from 0 to 900 s for the two simulations and the simulations were run for 900 s. Fig. 4 shows the number of core nodes decreases with more stability. There is a dip in the curves at very high mobilities because non-core nodes that move out of their dominator's domain travel fast enough to find themselves in the domain of a neighboring dominator and hence do not change the core set. The number of core nodes in both the scenarios is a small fraction of the total number of nodes. Fig. 5 shows the average percentage of nodes forwarding the *RREQ* message. The scenario used was 1500×300 m^2 with 50 nodes and 900 s simulation run. The percentages for the CEDAR versions are much lesser than that of the basic protocols. The reduction in the number of messages occurs because of two reasons: (i) the reduced number of nodes (core nodes) to which a *RREQ* 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.

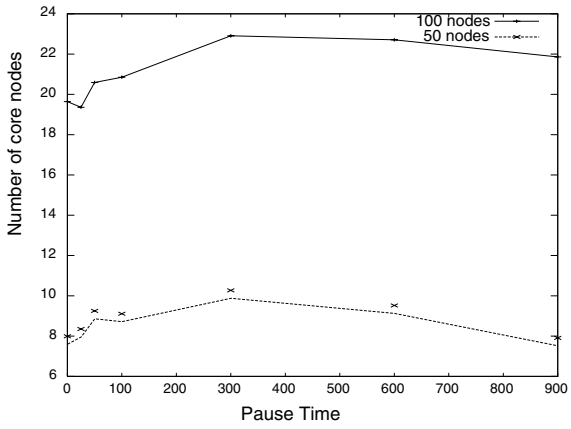

Fig. 4. Average number of core nodes.

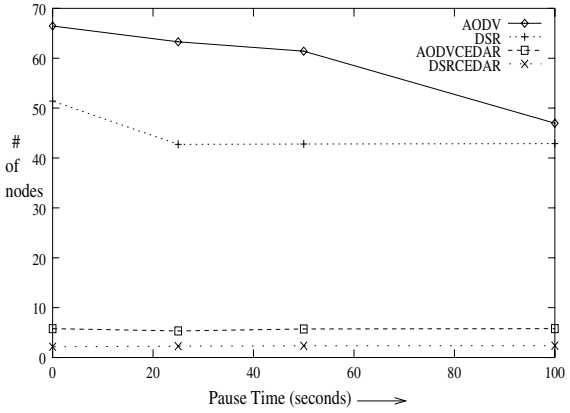

Fig. 5. Average number of messages per route request.

5.2. Data delivery

In this section, we show the percentage data delivered for the four protocols. Performance is measured in all three topologies (1500×300 m, 2200×600 m and 1500×1500 m). For each of the topologies, we use both 10 and 20 sources. The packet rates are varied between 1 packet per

[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 s, 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.

second and 5 packets per second. For larger packet rates, the network gets overloaded [5,11] and the performance of all protocols suffer uniformly. [7] The pause times used were 0 and 100 s respectively. Figs. 6–11 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 s), 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 learned from beacons, eliminates routing overhead to some destinations. Further, the partial topology is also used for packet salvation. Second, since the *RREQ* is not flooded in the core enhanced protocols, the route response time is much lesser, thus reducing queueing time and improving performance. 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.

5.3. 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

[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 Section 6.

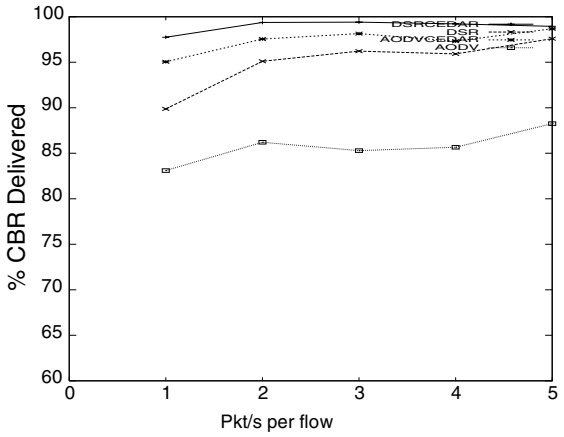

Fig. 6. 1500 × 300 m: data delivery: 10 flows.

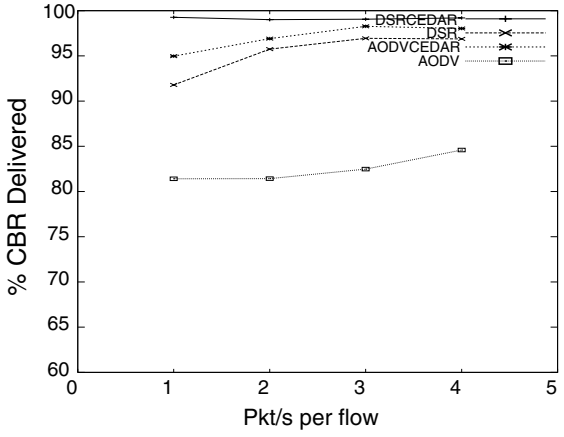

Fig. 7. 1500 × 300 m: data delivery: 20 flows.

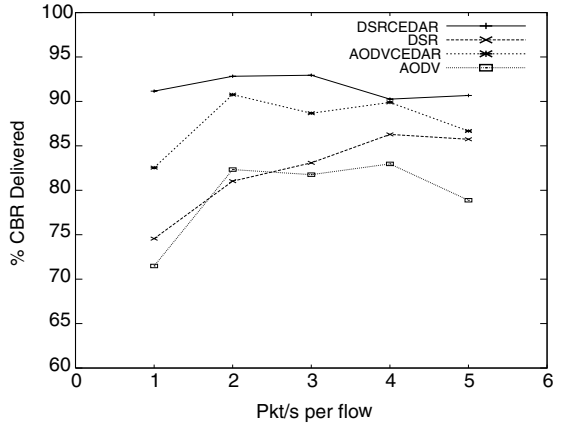

Fig. 8. 1500 × 1500 m: data delivery: 10 flows.

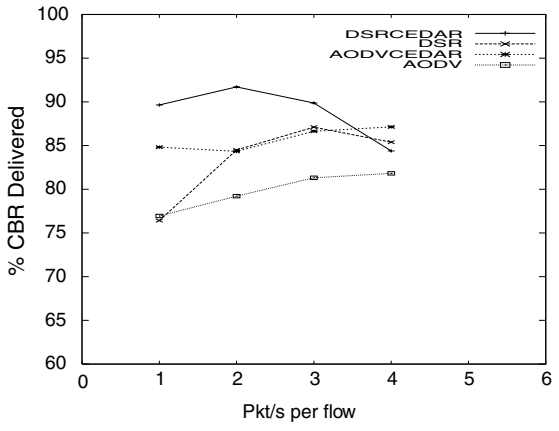

Fig. 9. 1500 × 1500 m: data delivery: 20 flows.

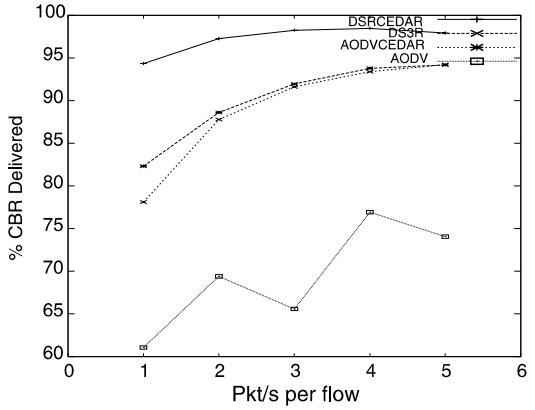

Fig. 10. 2200 × 600 m: data delivery: 10 flows.

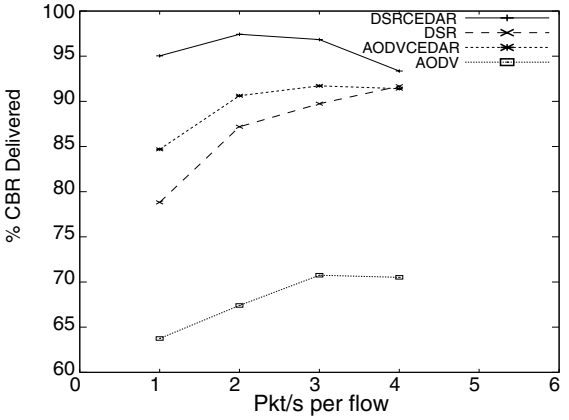

Fig. 11. 2200 × 600 m: data delivery: 20 flows.

packet and byte overheads were measured, due to lack of space we show only the byte overhead Figs. 12–17 for all the scenarios and show a representative subset of the packet overhead results (Figs. 18 and 19).

In Figs. 12–17, AODVCEDAR performs significantly better than AODV (upto 50% in the more dynamic scenarios). The reasons are twofold: (i) AODV's *RREQ* is typically a flood of the network. However, in AODVCEDAR, the *RREQ* is flooded only among core nodes, and (ii) the flood among core nodes is further 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 *RREQ* 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 1500 × 300 m scenario, comparable with DSR in the 1500 × 1500 m scenario and better than DSR in the 2200 × 600 m 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 Figs. 6–11) where the cached information is more

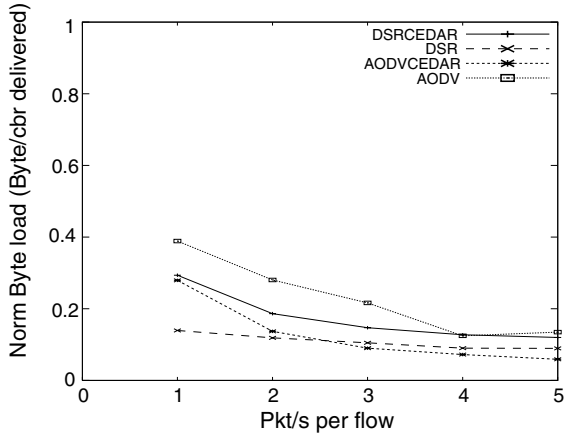

Fig. 12. 1500 × 300 m: byte overhead: 10 flows.

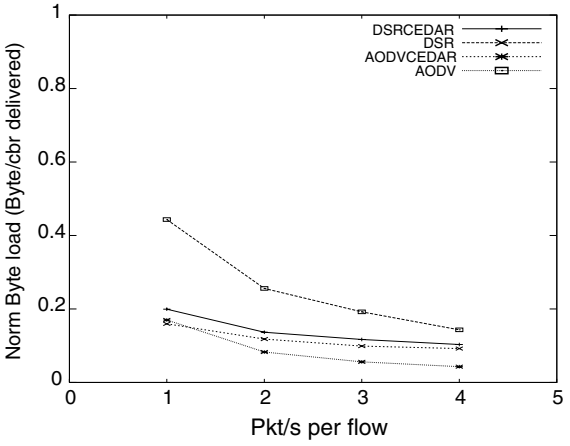

Fig. 13. 1500 × 300 m: byte overhead: 20 flows.

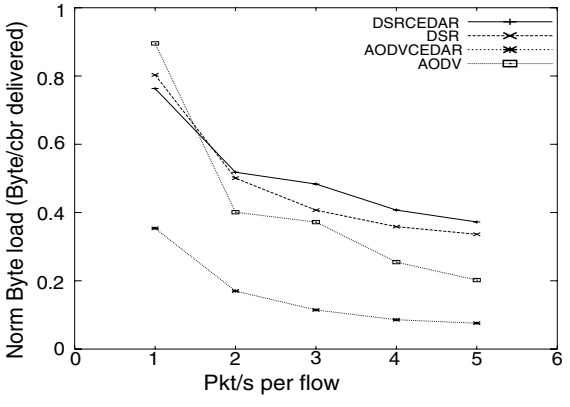

Fig. 14. 1500 × 1500 m: byte overhead: 10 flows.

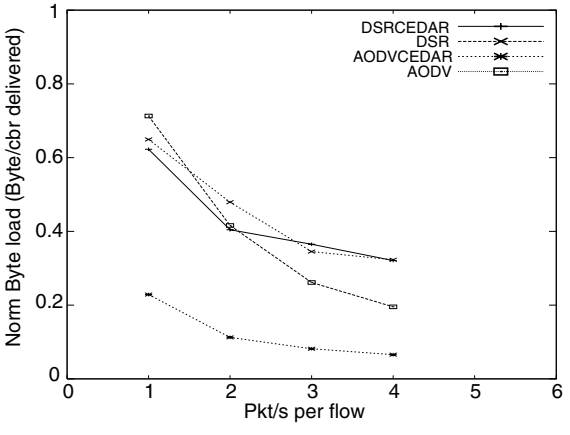

Fig. 15. 1500 × 1500 m: byte overhead: 20 flows.

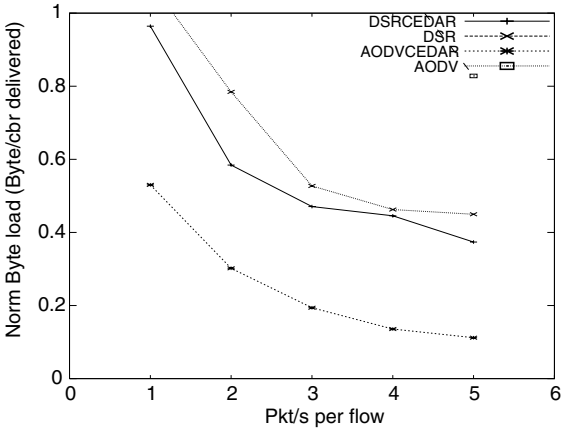

Fig. 16. 2200 × 600 m: byte overhead: 10 flows.

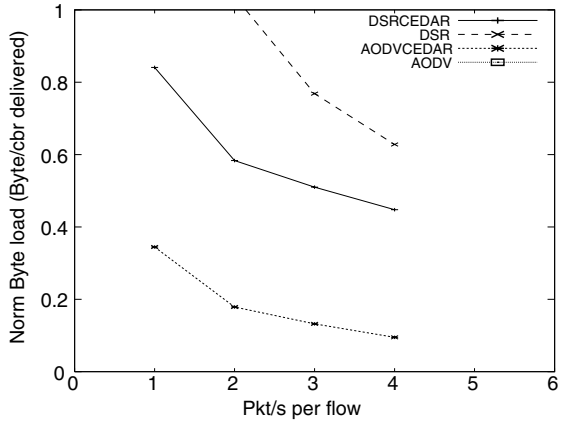

Fig. 17. 2200 × 600 m: byte overhead: 20 flows.

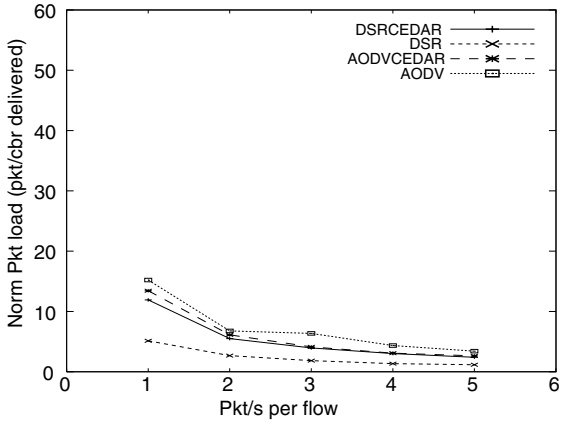

Fig. 18. 1500 × 1500 m: packet overhead: 10 flows.

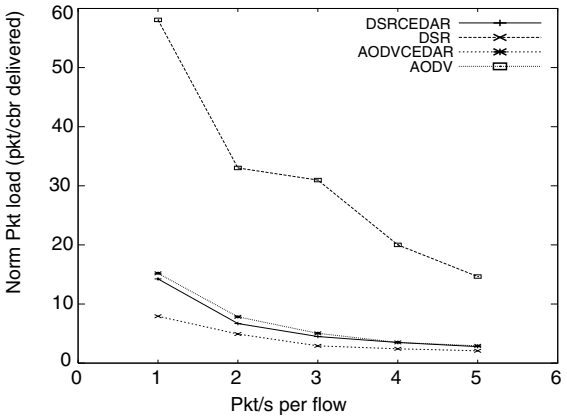

Fig. 19. 2200 × 600 m: packet overhead: 20 flows.

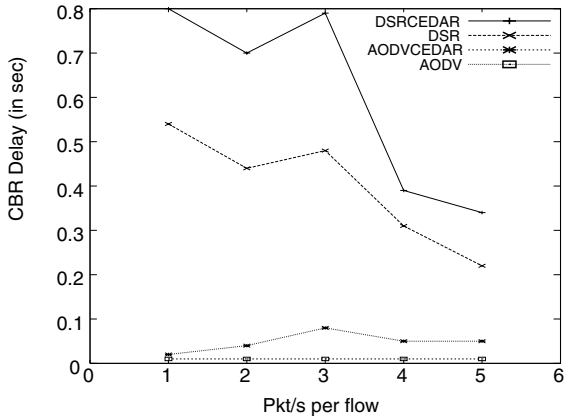

Fig. 20. 1500 × 1500 m: end to end delay: 10 flows.

often than not stale and as a result of which DSR's data delivery percentage suffers.

5.4. End to end delay

The fourth set of results we present is the average end to end delay of CBR packets in the network. Figs. 20 and 21 show 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 *RREQ* messages. Hence, each *RREQ* 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 CB mechanism that requires far fewer transmissions per CB.

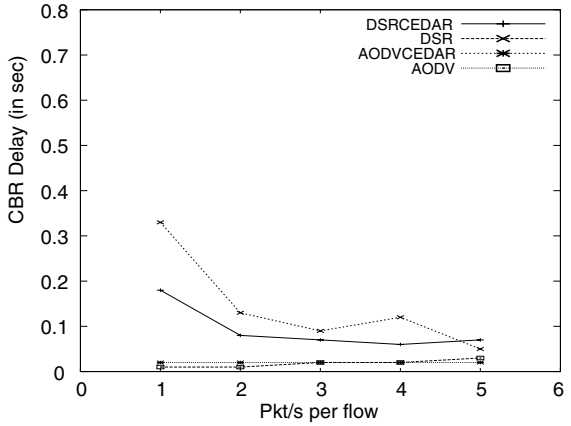

Fig. 21. 2200 × 600 m: end to end delay: 10 flows.

6. Issues and extensions

In this section, we first identify some key issues that need to be addressed in our approach and propose some solutions.

- *MAC level support for the routing protocol:* The current version of the CB algorithm relies on MAC level support for enhancing its performance. While the support is not critical for the correctness of the CB algorithm, it still brings about a considerable improvement in the performance of the algorithm. Specifically, MAC level snooping and caching enables nodes

to reply with an NCTS on receiving an RTS for a message already received or snooped, thus reducing the CB overhead. However, relying on changes in the MAC protocol arguably violates the layering principle and should be done away with in the ideal case.

Fortunately, most of the benefits demonstrated in this paper through MAC layer changes can still be garnered through appropriate routing layer mechanisms. The MAC layer mechanisms added to the basic IEEE 802.11 protocol belong to two main categories: (a) those that overhear packet transmissions, and therein enable nodes to proactively decide that forwarding a packet to another node is unnecessary, and (b) those that enable nodes which already have received a particular message to reply with a NACK in response to an RTS for that message. If the unchanged 802.11 protocol is to be used, nodes can still gain the benefits of (a) as long as they are neighbors of the transmitter, and the data packet has a message identifier. Similarly, the benefits of (b) can be attained through short routing layer control packet exchanges (that do not require RTS–CTS handshakes because of their small size) that mimic the RTS–NACK exchanges as described in this paper.

- *Decoupling data path from the core:* In both the enhanced protocols, no effort is currently taken to explicitly reduce the load on the core nodes by choosing paths away from the core. Hence, it is possible that core nodes tend to participate in more routes than their non-core counterparts (since *RREQ* message travel on the core and hence most of the nodes in the *RREQ* path will be core nodes). Although a simple solution to the problem would be to dynamically change core nodes so that all nodes in the network are made to participate in the core infrastructure in a fair manner, we are currently working on an approach that explicitly tries to decouple the data path from the core.
- *Dependence of core broadcast on beaconing:* The forwarding set used for core broadcasting, is computed based on the partial topology learnt from the recently heard beacons. The correctness of the CB mechanism, thus, depends on the accuracy of the partial topology available through beacons. Beaconing frequency as well the mechanism used to transmit the beacons has implications on this accuracy. In our implementation, we are making use of piggybacking of beacons on unicast transmissions, to improve the reliability of beacons, by implicitly protecting them using RTS–CTS. However, other mechanisms for improving the accuracy of the maintained partial topology need to be explored.
- *RTS–NCTS overhead:* The CB mechanism assumes that the RTS and NCTS packets are negligible in size compared to the CB packet. Thus, the MAC layer overhead of the CB is ignored while evaluating the efficiency of the CB. However, in the current simulation environment, the physical layer overheads, which are added to all the control and data packets, are significant compared to the RTS/NCTS packet sizes. Therefore, improvement in the physical layer overhead as well as a lower overhead RTS–NCTS messaging would improve the efficacy of CB.
- *Energy consumption and overloading:* The core nodes in CEDAR perform considerably more amount of work than the non-core nodes. Thus, an obvious issue is the higher drain on resources such as computing cycles, transmission slots, and battery power, at the core nodes. Note that the notion of co-operative relaying and the consequent unfair distribution of load among nodes is a much larger problem that is inherent to ad-hoc networking. However, when compared to traditional ad-hoc networking solutions, the use of a core does impose a heavier load on a subset of nodes in the network. Fortunately, this problem can easily be alleviated in CEDAR by appropriately changing the core computation process. Recall from Section 3 that core nodes are selected based on the tuple $(dom\_degree, degree, nodeId)$ where *dom_degree* represents the number of neighbors dominated by a node, *degree* represents the number of neighbors, and *nodeId* is the identifier of the node used to break ties. If fair battery power consumption is of importance, the amount of

battery power available at a node can be incorporated into the tuple. Thus, a tuple such as (*batter_power*, *dom_degree*, *degree*, *nodeId*) would bias the core computation in favor of nodes that have a larger battery power.

7. 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 which attempts at alleviating flood related limitations of ad hoc routing protocols by overlaying the core infrastructure [4] on the underlying network. The CB mechanism presented is substantial modified from [4] 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 the packet delivery percentage being improved to 99% or higher. While we observe significant improvements for packet and byte overheads in dynamic scenarios, the overheads are comparable in more static networks.

Appendix A

The following theorems related to the CB state the correctness of our algorithms. These two theorems together show that the CB does reach all the core nodes. It however assumes that the network topology has become static and the information received in the beacons are accurate. The theorems apply only to a connected component of the network.

Theorem A.1. *If the CB algorithm guarantees reaching all core nodes within three hops, all core nodes in the network will eventually receive it.*

Proof. Let us assume that the CB algorithm guarantees reaching all core nodes within three hops, but all core nodes do not receive the CB. Let C be the set of core nodes that received it and C' be the remaining core nodes. Let $c_1 \in C$ and $c_2 \in C'$ be such that the shortest path between c_1 and c_2 is the smallest for all possible values of c_1 and c_2. The distance between c_1 and c_2 (say d hops) must be more than three hops as the CB algorithm guarantees reaching all core nodes within three hops.

Let n_1 and n_2 be the first two nodes on the shortest path from c_1 to c_2. Both n_1 and n_2 must be non-core nodes. If any of them is a core node, then it is easy to see that we can find a pair of nodes one belonging to C and another to C' such that they are less than d hops apart, contradicting our choice of c_1 and c_2. Let $dom(n_2)$ be the dominator of n_2. If $dom(n_2) \in C$, then $dom(n_2)$ is a better choice for c_1 as its distance from c_2 is less than $d-1$ hops, which contradicts our choice of c_1. Otherwise if $dom(n_2) \in C'$, then $dom(n_2)$ is less than three hops away from c_1 which belongs to C, which leads to a contradiction since they should be more than three hops away.

Therefore, reachability to all core nodes within three hops suffices to guarantee reachability to all core nodes in the same connected component of the network. □

Theorem A.2. *The forwarding set algorithm* (*Section* 3) *ensures that all core nodes within three hops will receive the CB.*

Proof. The following argument holds for both a core and a non-core node as the originator. Note that the originator and the intermediate nodes use the same algorithm for forwarding set computation.

- *Core nodes at one hop*: By construction of the sub-tree for the partial topology graph, all core nodes in the first hop belong to the forwarding set.
- *Core nodes at two hops*: All nodes in the second hop are reachable through a core or a non-core node in the first hop. If it is reachable through a non-core node, then the bea-

con from the corresponding non-core node must have provided that path to the core node and the sub-tree computation will ensure that a neighbor reaching that core node is in the forwarding set. Otherwise, the core neighbor at first hop is anyway added to the forwarding set. This therefore reduces the problem to the one for reaching a core node at one hop which has been proved above.

- *Core nodes at three hops*: If there is a core node in the first hop which is two hops away or a core node in the second hop which is one hop away, from the core node at the third hop, then by virtue of the previous two arguments, these intermediate core nodes will receive the CB. The above two arguments can be applied again to ensure eventual reachability to the core node at the third hop.

Otherwise, there is a three hop path consisting of two intermediate non-core nodes, say the path is A–B–C–D where B and C are non-cores and D is a core node. If D is C's dominator, then B's beacon must contain information on D. Thus D will be reachable in the sub-tree computed for the forwarding set. If another core node E is C's dominator and not D, then E is either less than three hops away or exactly three hops away. In the former case, the previous arguments ensure that the CB will reach the core node E. In the latter case, the path A–B–C–E will be known based on the beacon from B and the sub-tree computation will ensure that a neighbor that is less than three hops away from E is selected in the forwarding set.

As node E could be at most two hops away from node D, it will eventually receive the CB, using the argument for two hops.

Thus the forwarding set computation guarantees that the CB will reach all core nodes within three hops. □

References

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

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

[3] Z.J. Haas, M.R. Pearlman, The Zone Routing Protocol (ZRP) for Ad Hoc Networks, Internet Draft draft-zone-routing-protocol-01.txt, August 1998.

[4] R. Sivakumar, P. Sinha, V. Bharghavan, CEDAR: a core-extraction distributed ad hoc routing algorithm, IEEE J. Select. Areas Commun. (Special Issue on Ad-hoc Routing) 17 (8) (1999) 1454–1465.

[5] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: Proceedings of IEEE MOBICOM, Dallas, TX, October 1998.

[6] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, M. Degermark, Scenario-based performance analysis of routing protocols for mobile ad-hoc networks, in: Proceedings of IEEE MOBICOM, Seattle, August 1999, pp. 195–206.

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

[8] V. Park, S. Corson, Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification, Internet Draft draft-ietf-manet-tora-spec-01.txt, August 1998.

[9] S. Murthy, J.J. Garcia-Luna-Aceves, A routing protocol for packet radio networks, in: Proceedings of ACM SIGCOMM '97, Cannes, France, September 1997.

[10] V.D. Park, M. Scott Corson, A highly adaptive distributed routing algorithm for mobile wireless networks, in: Proceedings of IEEE INFOCOM, Kobe, Japan, April 1997.

[11] S.R. Das, C.E. Perkins, E.M Royer, Performance comparison of two on-demand routing protocols for ad hoc networks, in: Proceedings IEEE INFOCOM, Tel-Aviv, Israel, March 2000.

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

[13] P.F. Tsuchiya, The landmark hierarchy: a new hierarchy for routing in very large networks, ACM SIGCOMM'88, Stanford, CA, 1988, pp. 35–42.

[14] W.T. Tsai, C.V. Ramamoorthy, W.K. Tsai, O. Nishiguchi, An adaptive hierarchical routing protocol, IEEE Trans. Comp. 38 (8) (1989) 1059–1075.

[15] Z.J. Haas, B. Liang, Virtual backbone generation and maintenance in ad hoc network mobility management, in: Proceedings of IEEE INFOCOM, Tel-Aviv, Israel, March 2000.

[16] J. Sharony, A mobile radio network architecture with dynamically changing topology using virtual subnets, MONET 1 (1) (1996) 75–86.

[17] N. Shacham, J. Westcott, Future directions in packet radio architectures and protocols, Proc. IEEE 75 (1) (1987) 83–99.

[18] J. Jubin, J.D. Tornow, The DARPA packet radio network protocols, Proc. IEEE 75 (1) (1987) 21–32.

Raghupathy Sivakumar received his B.E. degree in Computer Science from Anna University, India in 1996, and his M.S. and Ph.D. degrees in Computer Science from the University of Illinois at Champaign-Urbana, in 1998 and 2000 respectively. In 2000, he joined Georgia Institute of Technology as an Assistant Professor in the School of Electrical and Computer Engineering, where he is currently engaged in teaching and conducting research in the area of computer networking. His research interests are in the areas of wireless networks, mobile computing, and quality of service.

Prasun Sinha received his B.Tech. in Computer Science and Engineering from IIT Delhi, India in 1995, his M.S. in Computer Science from Michigan State University in 1997, and his Ph.D. in Computer Science from University of Illinois at Urbana-Champaign in 2001. He is currently with Bell Labs, Lucent Technologies. His research interests are in Computer Networking and Mobile Computing.

Vaduvur Bharghavan's research interests are in wireless networking and mobile computing.