Mobile Networks and Applications 8, 535–549, 2003

Explicit Multicasting for Mobile Ad Hoc Networks

LUSHENG JI
Fujitsu Laboratories of America, 8400 Baltimore Avenue, Suite 302, College Park, MD 20740, USA

M. SCOTT CORSON
Flarion Technologies Inc., Bedminster One, Bedminster, NJ 07921, USA

Abstract. In this paper we propose an explicit multicast routing protocol for mobile ad hoc networks (MANETs). Explicit multicasting differs from common approaches by listing destination addresses in data packet headers. Using the explicit destination information, the multicast routing protocol can avoid the overhead of employing its own route construction and maintenance mechanisms by taking advantage of unicast routing table. Our protocol – termed Differential Destination Multicast (DDM) – is an explicit multicast routing protocol specifically designed for MANET environment. Unlike other MANET multicasting protocols, instead of distributing membership control throughout the network, DDM concentrates this authority at the data sources (i.e. senders) thereby giving sources knowledge of group membership. In addition, differentially-encoded, variable-length destination headers are inserted in data packets which are used in combination with unicast routing tables to forward multicast packets towards multicast receivers. Instead of requiring that multicast forwarding state to be stored in all participating nodes, this approach also provides the option of stateless multicasting. Each node independently has the choice of caching forwarding state or having its upstream neighbor to insert this state into self-routed data packets, or some combination thereof. The protocol is best suited for use with small multicast groups operating in dynamic MANET environment.

Keywords: MANET, wireless networks, explicit multicasting, differentiated destination routing

1. Introduction

There are generally two mainstream approaches to multicast routing in fixed networks: Group Shared Tree (GST) and Source-Specific Tree (SST). Both construct data forwarding paths interconnecting all group members, where a group member is understood to be a multicast *receiver*. Data is firstly forwarded to the tree then along the tree paths to reach all group members. Several protocols include data sources as part of the forwarding tree as well. The GST approach builds one tree for the whole group regardless of where the data sources are located. The SST approach organizes multicast forwarding by data source. A "session" is associate with each source and is identified by the combination of group ID and source ID. For each session there is one distribution tree formed from the union of all shortest paths between the source and the group members. This form of multicasting usually results in lower end-to-end delay since data is forwarded using the shortest source-to-receiver paths, but typically consumes a greater amount of network bandwidth for constructing and maintaining multicast routes than shared-tree approaches because one tree is needed per session. Also each constructed tree usually contains more links than that of shared-tree approach.

A characteristic shared among these traditional multicast protocols is that multicast routing computation is *distributed* in the network. Not only are the forwarding states constructed and maintained by the network, group membership control (or lack thereof) is also distributed over the network. While this approach improves scalability with respect to group size, it has certain drawbacks. Firstly, distributed membership management may make aspects of security more difficult due to the lack of admission control. Without natural support for end-to-end signaling between sources and receivers, such traditional approaches need to rely on external mechanisms for security management. At the same time, billing management becomes more complicated as the information property owners, typically the sources, have no control and knowledge over how and to whom their property (data) is distributed. Secondly, distributed per group and per session forwarding state maintenance may result in large router resource usage. The number of possible multicast groups formed among n members grows combinatorially. While there is no efficient way to aggregate multicast routing table entries, the linearly growing multicast routing table (with respect to the number of active multicast groups) may quickly become too much of a storage burden for routers. The issue is worsen by the number of routers involved in forwarding since all routers along the multicast forwarding paths need to participate in multicast routing state maintenance.

To address this issue, recently there is an attempt of shifting towards stateless multicasting for small groups. Papers [4,19] lift multicasting out of routing layer so that multicasting no longer requires router support. Multicast data is encapsulated in unicast envelops and transmitted between end receivers. Therefore no multicast routing state is installed on routers. Papers [2,13] propose connectionless, small group multicast as a "stateless" approach to multicast routing. In these approaches, variable-length destination lists are placed in packet headers that are self-routed towards the destinations using the underlying unicast forwarding tables.

We consider the problem of multicast routing in Mobile Ad hoc Networks (MANETs). In MANETs all mobile nodes are equipped with wireless communication interfaces and can move at will. Here we assume their communications occur using omni-directional antennas over broadcast medium. The combination of node mobility and a wireless environment can result in MANET topology subject to rapid and unpredictable changes. Because of these dynamics, and the fact that communication is carried over a bandwidth-constrained broadcast medium, the multicast routing problem in MANETs differs from that in fixed networks significantly.

During recent years many multicast protocols have been designed specifically for MANETs (e.g., SMB [6], CAMP [5], ODMRP [11], MAODV [17], AMRoute [1], AMRIS [21], BEMRP [14], MCEDER [18], and LAM [7]). These protocols all follow the traditional multicast approaches, i.e., distributed group membership management and distributed multicast routing state maintenance. In addition to the security and resource use issues mentioned before, these approaches, especially when applied for use with small and sparsely distributed (potentially numerous) groups, may become even less efficient and more expensive to function in MANETs due to bandwidth constraints, network topology dynamics, and high channel access cost.

2. Protocol description

2.1. Overview of proposed approach

Aiming at the issues briefed in the previous section, we propose the Differential Destination Multicast (DDM) protocol. This approach is motivated, in part, by the approach to unicast routing of Dynamic Source Routing (DSR) protocol [10] and derived, in part, from the work of [2,13]. DDM is a unicast-dependent protocol. It requires the existence of a unicast routing protocol which provides unicast routes. However, DDM does not limit itself to any particular unicast protocol.

In DDM, the sources control multicast group membership to ease certain aspects of security administration. More importantly, and a departure from other proposed MANET multicast protocols, DDM encodes the destinations (i.e., the multicast group members to whom data needs to be delivered) in each data packet header in a fashion different from [2,13]. This "in-band" information can be used to establish soft-state routing entries if desired. Using such an approach has two advantages for MANETs. Firstly, there is no control overhead expended when the group is idle; a characteristic shared with DSR. Since many multicast applications do not have continuous traffic flows, it can be expensive in terms of network control overhead to maintain multicast forwarding state in routers during idle periods. In-band control avoids this problem because if there is no data traffic, there is no need for any control information either. Secondly, it is not necessary for the nodes along the data forwarding paths to maintain multicast forwarding state if they choose to run under stateless mode. When an intermediate node receives a DDM data packet, it only needs to look at the DDM header to decide how to forward the packet; another similarity to DSR. Assuming that routers can handle this processing cost, this stateless mode can be very reactive and efficient. This stateless approach also avoids loading the network with pure signaling traffic; a third trait shared with DSR. In so doing, the hope is that the unicast algorithm can converge much *faster*, with DDM then making immediate use of new unicast routing knowledge.

While the fixed networks and MANETs can both benefit from stateless, explicit multicasting because of its savings in storage complexity, it is even more desirable to follow this approach in MANETs due to the special characteristics of the MANETs. Firstly, in wired Internet, network topology change rarely occurs. Therefore if group membership is stable, after a multicast tree is constructed, the maintenance effort is small and tree links rarely need any repair. On the other hand, MANET topology is subject to constant and sometimes dramatic changes. Built in such an environment, multicast forwarding topology also requires constant repair even rebuild. Maintaining multicast routing states in MANET is much more expensive than in wired networks. This is even worsen by the tight bandwidth constrains of MANETs. Secondly, the cost of medium access is high in wireless broadcast networks due to the underlying MAC mechanism and broadcast transmission's blocking effect to neighboring nodes. Although packing routing information together with data traffic will enlarge data packet size, it reduces the total number of channel accesses because it reduces the number of pure control packets generated by the routing protocol. Therefore such an approach can be more efficient overall in many scenarios. Thirdly, in broadcast networks, when a node needs to send to more than one neighbor, it only needs to broadcast the packet once. So this approach has better bandwidth consumption and channel access properties in broadcast networks than in point-to-point networks. Lastly, the performance bottleneck for Internet routers is typically at forwarding processing. The relatively complex processing of the destination encoded headers prevents fast data forwarding at Internet routers. On the other hand presently in MANETs, the effect of per-packet processing on forwarding rate is less significant relative to bandwidth, medium access, and energy constraints. Also since mobile computing devices are typically less resourceful comparing to their stationary and wired counterparts due to weight and physical dimensional limits, saving storage resource use becomes more important in MANETs.

In scenarios where the stateless approach is not favorable, DDM may also operate in a "soft-state" mode. It is here that DDM markedly departs from the works of [2,13]. In this mode, as data packets with in-band control information are routed through the network, each node along the forwarding paths remembers the destinations to which it forwards data and how data is forwarded (i.e., which next hop is used for each destination). By caching this information, the protocol no longer needs to list all the destinations in *every* data packet header. Only when changes occur in unicast routes or destination list, an upstream node needs to inform its downstream neighbors (i.e., its next hops) regarding the *differences* in des-

tination forwarding since the last packet; hence, the name "Differential Destination" Multicast. Reporting only the differences significantly reduces DDM header size. Ideally, in a stable network where the topology and membership remain unchanged, only the first data packet needs to contain destination addresses and all subsequent packets would contain no destination information. In practice, the state kept at each node along the forwarding paths is "soft". Each time data forwarding occurs, this state is refreshed. Stale state eventually times-out and is removed. This mode is better suited for applications which generates small data packets at relatively high rate.

DDM is not a general purpose multicast protocol in the conventional sense. The header-encoded destination mechanism does not scale well with group size. However, there are many applications which do fall into the domain of "small group multicasting". On-line video gaming, panel discussion, and applications that maintain image consistency among multiple servers are all examples of this type of applications. Specifically in mobile computing area, small group multicasting may also be used for smooth hand-off. All these applications may benefit from DDM. Thus, the DDM serves rather as a compliment to traditional multicasting.

The stateless mode, however, does scale better with the number of multicast groups, as no per-session state is required in any routers. In MANETs, if the number of multicast groups is small enough to permit state storage, then the soft-state version using differentially-encoded header processing can be used to reduce average packet size and save bandwidth. This mode, however, has essentially little applicability to fixed networks as the complexity of the differentially-encoded header processing would significantly slow down forwarding rates in high-speed networks.

The rest of this section will give a brief introduction of the DDM protocol. A detailed description of the protocol can be found in [9].

2.2. Membership management

In contrast with traditional multicast algorithms, a multicast data source plays an important role in the DDM protocol. The protocol proceeds independently for each source sending to a multicast group (a session), and the remaining description applies to a single session. The source acts as an admission controller for the information itself is sending. When a node (a "joiner") is interested in a particular multicast session, it needs to join the session by unicasting a JOIN message to the source of that session. This JOIN message only needs to include the ID of the group to which the node wants to join. The source can find other useful information, such as the "joiner ID" and the "source ID", from the corresponding fields of the IP header. Then it is up to the source to decide if the JOIN can be accepted. The admission policies are beyond the scope of this paper.

Upon receiving a JOIN message, if the joiner passes the session's admission requirements, the source adds the joiner into its member list (*ML*) and the joiner becomes a receiver of the session (also referred as a destination of multicasting in following context). The source then acknowledges the JOIN message by unicasting an ACK message back to the joiner.

The sending of JOIN message may be repeated with exponential back off by the joiner if the join operation is not successful. The success of a join may be notified by the source using an ACK message. In addition, due to the explicit nature of DDM, a joiner will only receive data packets if the source has accepted the join. Thus, unless the security mechanism (if any) in use specifically requires an ACK, the receiving of data packets also hints a join success.

The *ML* kept at the source node needs to be refreshed from time to time to maintain up-to-date membership information. Due to the dynamic nature of wireless networks, node reachability may change unpredictably. When operating in hostile environments such as battle field, nodes may also be destroyed before any graceful exit procedure can be executed. Thus the source needs to be able to purge stale members. DDM employs a source-initiated periodical membership refreshing mechanism. When the source needs to refresh its *ML*, it sets a special POLL flag in the next outgoing data packet. Upon receiving such data packet, a multicast session member needs to unicast a JOIN message again to the source to express its continued interest. If a current member fails to respond to the poll for a predetermined number of times, it is assumed to be no longer interested in receiving data. The source then removes this member from its *ML*. JOIN "implosion" at the source is not expected to be a problem due to small expected group sizes. If necessary, on each receiver a random delay jitter may be inserted between the receiving of the poll and transmission of the JOIN to reduce congestive effects at the source.

This "polled" membership refreshment is used as a secondary mechanism to detect an absent member. An explicit LEAVE message is defined as the preferred way for a session member to leave the source's *ML*. It is a unicast message sent from the departing member to the session source. When received by the source, this LEAVE message terminates the member's membership. The member is removed from the *ML* and is excluded from future forwarding computations. To increase robustness, instead of sending just one LEAVE message, more than one copies of the LEAVE message may be unicast to the source when a node leaves the session. After these transmissions, a member node removes all information associated with the multicast session. The source may also dismiss a receiver by removing it from the *ML* at any time if membership control policy suggests so.

2.3. Forwarding computation

The key notion of DDM is forwarding computation based on destination (i.e., multicast receiver address) encoded headers, may or may not be differentially-encoded depending on in which mode a node operates. In this section, we describe the operation under soft-state mode. The operation for stateless mode is more straight forward.

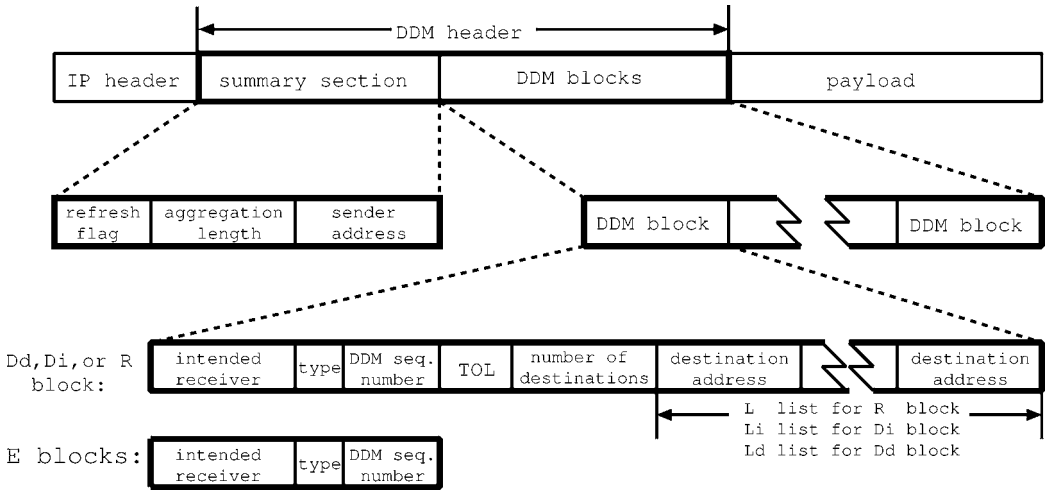

Figure 1. DDM header format example.

2.3.1. Packet formats

DDM employs two types of packets: control packets and data packets, where it is understood that data packets may also contain control information. There are four types of control packets: JOIN, ACK, LEAVE, and RSYNC. The first three are used only by the membership control part of the algorithm. Each of these control packets contains one field for the packet type and one for the group address. Since they are unicast packets, the source and destination of the packets are already included in the packet's IP header. The fourth packet type, RSYNC, is used by a node to request its upstream neighbor to resynchronize the stored destination lists on both nodes. The detailed used of RSYNC message will be explained later in sections 2.3.4 and 2.3.5. This is a unicast message with TTL of 1.

Each multicast data packet contains a payload and a DDM header. Each DDM header is composed of a *summary section*, and at least one *DDM block*. The summary section provides an overview of the blocks while each DDM block instructs corresponding downstream neighbor about the destination list change since the last forwarding. We use three types of DDM blocks: Empty (E) blocks, Refresh (R) blocks and Difference (D) blocks. Further a D block can be either an incremental block (D_i), or decremental block (D_d). When both D_d and D_i blocks are needed in a DDM block to describe a difference, both blocks are constructed and put into the header in a way that is consistent to all nodes (i.e., always put the D_d block immediately ahead of the D_i block).

The E block does not need any destination address. It is followed directly by the payload. Each R block has a destination list L which is intended to replace the destination list on the downstream node. Each D block contains a list of destinations which describes the destination changes since the last forwarding. Both Difference and Refresh blocks may be referred to as "informative" blocks in following text since they contain forwarding list update information.

When used in broadcast medium networks, DDM blocks for different downstream neighbors may be aggregated together and put in a single DDM header to reduce the number of transmissions. When this feature is in use, the summary section contains description for the aggregation. The *DDM header aggregation length* field indicates the number of DDM blocks being so aggregated. Each DDM block is tagged by its intended receiver so that when the packet is received, the receiver can locate the correct DDM block for itself. Figure 1 offers an example of an aggregated DDM data packet header. More detailed specification for packet format as well as other parameters can be found in [8].

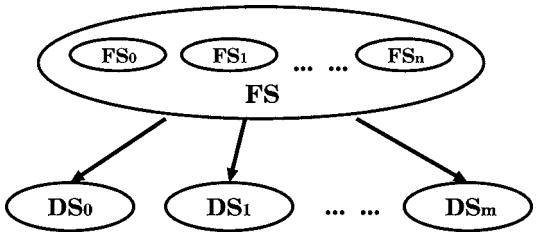

Figure 2. Forwarding computation data structure.

2.3.2. Data structures

The data structures involved in the forwarding computation is shown in figure 2. Such a data structure is only needed for nodes operating in "soft-state" mode. At each node involved in forwarding, there is a Forwarding Set (FS) for each active multicast session. It records to which destinations this node needs to forward multicast data. At the source node, the FS is the same as the ML. At other nodes, this FS is actually a union of several smaller sets. Each small FS_k set records the destinations included in data packets received from its corresponding upstream neighbor. After receiving a data packet from an upstream neighbor k, the receiving node will update the corresponding subset FS_k and then the overall union set FS,

$$FS = \bigcup_k FS_k, \quad \text{for all upstream neighbors } k.$$

Associated with each set FS_k, there is a sequence number which is used to record the DDM block sequence number

```
ProcessDDmDataPacket(P) {
  if (P has been seen before)
    Stop;
  if (fail to locate myDDMBlock in P's header)
    Stop;

  if (myDDMBlock.sequence > ForwardingSet[P.sender].sequence + 1) {
    Send RSYNC message to P.sender;
    Stop;
  }

  Update ForwardingSet[P.sender];
  Update overall ForwardingSet;

  for each destination D in overall ForwardingSet {
    nextHop = FindNextHop(D);
    add D into NewDirectionSet[nextHop];
  }

  newP = allocate packet with P's payload and empty header;
  for each NewDirectionSet[i] {
    if (exists DirectionSet[i])
      newDDMBlock = create R, D, or E DDM Block for nexthop i based on the
      difference between NewDirectionSet[i] and DirectionSet[i]
    else
      newDDMBlock = create R DDM Block for nexthop i
    replace the contents of DirectionSet[i] with NewDirectionSet[i]

    insert newDDMBlock to newP's header;
    if (newP's header is full) {
      send away newP;
      newP = allocate packet with P's payload and empty header;
    }
  }
}
```

Figure 3. DDM data packet forwarding processing.

seen in the last received packet. The reason for this sequence number is to detect any loss of DDM data packets containing destination list updates.

The overall Forwarding Set contains all the destinations to which this node needs to forward. However, these destinations may be reached via different paths (i.e., next hops). Therefore, the Forwarding Set needs to be partitioned according to these next hops. After the partition, each subset only contains destinations which are reached via the same next hop. We call these subsets after partition Direction Sets (DS_l). Again there is a sequence number for each Direction Set. When used together with the sequence numbers of the Forwarding Sets on downstream nodes, it can be used to detect destination list update losses.

The Forwarding Sets and Direction Sets are dynamically created and maintained by a node only if there is active traffic passing through the node. Thus, they only exist on nodes that are along the forwarding paths of the multicast group. If a node does not participant in forwarding, no data structure will be created on the node. After data path changes, data structures on nodes that are along the old path but not the new ones will be removed by time out.

2.3.3. Processing

Here we assume that DDM header aggregation is in use. Most of the processing consists of set operations. The computation complexity on each participating node is thus bounded by $O(n \log n)$, with n being the number of members in the multicast session. The forwarding computation can also be found in figure 3.

When a node receives a DDM data packet from an upstream neighbor, it first tries to locate the DDM block intended for itself by looking at the intended receiver tag of each enclosed block. If no such block can be found, the data packet is dropped and the forwarding processing stops. After finding its DDM block, the receiving node needs to check if the packet has been seen before. If so, the packet is discarded as well.

Then the receiving node accesses the locally stored Forwarding Set for the sender of the packet (the upstream neigh-

bor). If no such set is found the receiver tries to create one if possible. However, if the received DDM block is an Empty or Difference block, this means the receiving node is out of synchronization with the sender. Some packets containing destination list updates have been lost. In this case, the receiving node needs to initiate the resynchronization process.

Before a node proceeds further, it compares if the sequence number of its Forwarding Set matches with the sequence number contained in the received DDM block, which is copied from the corresponding Direction Set of the upstream neighbor. The reason for this step is to detect DDM update losses. Because of the differential approach, it is very important for DDM to quickly detect any losses of packets which contain destination list updates.

If there is a gap between the sequence numbers, which indicates that some packets containing informative DDM blocks have been lost. Thus the receiving node needs to start the resynchronization process if it can not construct the current destination list based on the received DDM block.

After the receiver checks sequence numbers successfully, it tries to update the destination list in its Forwarding Set based on information in the DDM block. If the block is a R block, the node simply replaces its destination list with the list L in the block. When a node receives a D block, it needs to check the next DDM block in DDM header to see if that block is also a D block (of different type) for itself. Then it updates its destination list according to the content and the type of the D blocks: removing the addresses listed in D_{d} block and adding the addresses appeared in the D_{i} block.

$$FS_k = (FS_k - L_{\mathrm{d}}) \cup L_{\mathrm{i}}.$$

After updating the Forwarding Set, the receiving node updates the overall Forwarding Set. Then it partitions the new overall Forwarding Set into Direction Sets according to the "next hops" to reach these destination as provided by the unicast routing table. These new Direction Sets are then compared with the Direction Sets generated by the last data forwarding to find their differences. Finally the differences are used to assemble the DDM blocks which are packed into the outgoing packets.

If DDM header aggregation is not in use, each Direction Set will produce one (or two, in the case of needing both D_{i} and D_{d} blocks) DDM block for the DDM header, which will be inserted into a copy of the packet and forwarded to the downstream neighbor. Otherwise, more DDM blocks intended for different downstream neighbors will be packed into the same DDM header until the header size reaches its limit. Then the filled data packet is sent out and a new data packet with the same payload is allocated. New DDM blocks are then inserted into the header of this new data packet.

2.3.4. Forwarding set synchronization

When DDM is operating in soft-state mode, usually only the *changes* of the destination lists are included in data packet headers. Therefore it is very important to keep the Direction Set on the upstream side and the Forwarding Set on the downstream side synchronized. The DDM algorithm uses sequence numbers to maintain synchronization.

Both the Direction Sets of the upstream neighbors and corresponding Forwarding Sets of the downstream neighbors keep their own sequence numbers. DDM blocks are then used as vessels to carry the sequence numbers from the upstream neighbors to the downstream neighbors so the sequence numbers can be matched and updated. Every time when an upstream neighbor sends out an informative block, it should increment the sequence number in the corresponding Direction Set.

When a downstream node notices a gap between a received sequence number and its local sequence number, it knows that it is out of synchronization with its upstream neighbor. In this case it is required to send a RSYNC (request for synchronization) message to the upstream neighbor. The upstream neighbor will, in turn, mark the Direction Set which is out of synchronization so that the next time when the upstream node is assembling DDM block for that particular Direction Set, it will include the full destination list instead of the difference. Optionally, if the upstream node caches the last data packet, it may also forward a copy of the data packet with a full destination list as soon as it receives the RSYNC message.

2.3.5. Timers and dual modes of DDM

DDM is a dual-mode algorithm where each participating node can independently operate in either "stateless" mode or "soft-state" mode based on its local situations such as computing resource availability, environment dynamics, etc. Using a combination of RSYNC message and timers, neighboring nodes can coordinate between themselves when they operate in different modes.

There is one timer associated with each Forwarding Set and Direction Set. When any of the timers expires, its corresponding set is removed. On the other hand every time when a set is used for forwarding computation, its lifetime is renewed. Each Direction Set may pick its own expiration time upon renewal based on local environment parameters. For instance, when the relative speed of surrounding nodes is high, a node tends to pick a smaller lifetime for its Direction Sets.

Each Direction Set may then instruct its downstream neighbors for how long they should keep their corresponding Forwarding Set by including the life time in the DDM blocks it sends to these down stream neighbors. The Direction Set should always set the life time for the Forwarding Sets on its down stream neighbors to be longer than the time used for its own expiration. This helps ensuring that, over a given link, the Direction Set on the upstream side will be removed before the Forwarding Set on the downstream neighbor. Otherwise it may happen that a downstream neighbor has its Forwarding Set removed but the upstream still has its Direction Set. When the upstream neighbor forwards the next data packet it will generate an Empty or Difference block which the downstream neighbor can not handle without requesting extra resynchronization. Although the timer setting scheme does not completely avoid this from happening (packet loss may still cause

the same problem), it should sufficiently reduce the probability of the occurrence of such undesired event.

The preceding applies to "soft-state" operation. If "stateless" forwarding is desired at a node, it needs to inform its upstream neighbor about its decision so the upstream neighbor will only provide full destination list in its DDM blocks intended for the "stateless" node. If the upstream neighbor is currently in "soft-state" mode, the "stateless" node needs to send back a RSYNC message with a special piece of instruction. After receiving such a RSYNC message the upstream neighbor will mark the corresponding Direction Set and from the next data packet onwards it will only assemble Refresh type of DDM block to satisfy the "stateless" downstream neighbor. Later on, if a "stateless" node ever wants to switch back to soft-state again, it only needs to send another RSYNC message to its upstream neighbor to cause the latter to remove this special mark.

2.4. Route correctness discussion

DDM is loop-free as long as the underlying unicast routing is loop-free. Since the DDM routing (*FS* to *DS*'s partition) calculation is carried out for every data packet, only the most recent unicast routing information is used. Transient loops are still possible during unicast routing convergence period, but will disappear as the unicast protocol recovers. During this period, some data packets may have been errantly sent using erroneous route information. In this case, these data packets will either be dropped when the IP TTL reaches zero, detected by duplicate detection mechanisms, or exit the loop and head towards the destination when some forwarding node finally corrects the loop.

Broken route is also handled by unicast routing. It is not necessary for DDM to learn about link status. The old *DS* and the old *FS* sets on two ends of the broken link are left alone to be deleted on time-out since no data packet is forwarded using them. If there is alternative route available, or as soon as the unicast routing finds one, DDM will use the next hop of the new route when forwarding the next multicast data packet. When there is new link to be added, DDM does not react either. In fact, DDM is not aware of the change if this new link is not used for reaching DDM destinations. Only after the unicast routing protocol designates this link as a next hop for some DDM destination will DDM utilize the new link. DDM will then set up a new *DS* for this new link. By relieving DDM from multicast forward path monitoring and maintenance, the only pure control traffic DDM generates during data forwarding is the RSYNC packet, which is only expected to be used rarely to correct inconsistency and coordinating between neighbors in different operation modes.

Data packets may be lost during transmission. If the lost packet contains only *E* block, it is simply a data loss with no implication to DDM algorithm. If the packet contains a *D* or *R* block, the loss may desynchronize the *FS* on the receiving end from the matching *DS* on the sending end. However, since all DDM blocks contain sequence numbers, the receiving end of the lossy link will discover the loss of previous informative DDM blocks when receiving the next data packet. A RSYNC is then sent back asking for a *R* block to resynchronize the *FS* and the *DS*.

3. Simulation studies

3.1. Simulation environment

The simulations are implemented using the NS-2 network simulation package [20]. The simulation environment models a MANET of 50 mobile nodes. At the beginning of each simulation run, nodes are randomly placed in an 1000 m by 1000 m area. Node movement follows the random way-point model of [3] with no pausing. When the simulation starts, each node randomly picks a destination and moves towards it with a random constant speed. After a node reaches its destination, it selects a new destination and starts to move towards that at a newly selected speed. Each simulation is executed for 900 seconds. There is no network partition during the course of simulation.

The network stack of each mobile node consists of a link layer, an ARP module, an interface priority queue, a MAC layer, and a network interface. IEEE 802.11 is used as the MAC protocol. The radio transmission range is approximately 250 meters. Each network interface transmits data at a rate of 2 Mbits/sec.[1]

In all simulation runs, CBR traffic flows are injected into the network from source nodes. The size of data payload is 512 bytes. Data packets are generated at each source at a rate of 4 packets per second. Group members and sources are randomly selected among all 50 nodes. Source nodes may or may not be group members themselves. To reduce side effects, membership control features are turned off. All group members join the multicast group at the beginning of the simulation and remain members till the end of simulation.

3.2. Simulation introduction

We study three aspects of explicit multicasting in MANET environment. Firstly the DDM protocol is compared with two other protocols, simple flooding and the On-Demand Multicast Routing Protocol (ODMRP) [11]. The ODMRP multicast routing agent implements the specification of [11] without mobility prediction. Because DDM is very different to traditional approaches to multicasting in many ways, it is difficult to find a protocol with similar behaviors. Because of the superior performance results of ODMRP as reported by [12], we decide to use ODMRP as a reference for traditional approaches to see how and in what situations DDM may compliment them. In simulation, ODMRP protocol parameters are set to the values used in [12], which was produced by

[1] Current 802.11b standard permits transmission rate of up to 11 Mbps. However, the rate is adjusted based on signal quality. While for one-to-one communication it is possible to find out the quality between two communicating nodes, it is not easy to do the same for multicast/broadcast communication. Therefore, to be safe, almost all implementations of the 802.11b standard limit multicast data rate to 2 Mbps.

the authors of ODMRP. In addition, the Forwarding Group time-out value is set to 4 times of the route refresh interval, as suggested by [11].

Secondly we study different flavors of the DDM protocol. Since the DDM is such a flexible protocol, its behavior may vary under different configurations. This part of the study focuses on three different settings of the protocol: DDM with aggregation, DDM without aggregation, and DDM tunneling. When DDM is forwarding data packet to multiple neighbors, it has the option of aggregating these packets together since they carry the same data payload. With aggregation, obviously the number of data transmissions is reduced. However, since aggregated packet can only be sent as broadcast packet, the reliability of packet transmission is reduced. This may affect the protocol throughput. In this part of the study, we would like to see the trade-offs between these two configurations. The DDM tunneling configuration is a setting we use for comparison. In this configuration data packets are tunneled directly from sources to each receiver so the performance we see is equivalent to "repeated unicast", a naive one-to-many communication mechanism used when multicasting is not available.

In the third part of the study, we study the impact of using different unicast routing protocols underneath DDM. The basic version of DDM used in previous two parts of the study runs above an omniscient unicast routing algorithm which always knows the full topology of the network and computes the shortest paths to destinations based on network topology. This decouples DDM from the characteristics of any given unicast algorithm, and allows us to focus on its own behavior under admittedly ideal circumstances. The performance obtained in this case is probably an upper bound on that achievable with DDM. In this part of the study, we also run DDM over two real MANET unicast routing protocols: the Ad Hoc On-Demand Distance Vector (AODV) protocol [16] and the (Destination-Sequenced Distance-Vector (DSDV) routing protocol [15]. Both routing protocols are already implemented in the NS simulation package. A major difference between these two protocols is that the AODV is an on-demand protocol while the DSDV is a full routing table protocol. The former only constructs and maintains a route when there is active data stream utilizing the route while the latter always maintains a full routing table containing routes for all nodes in the network. It would be interesting to see how DDM performs differently under these two environments.

3.3. Parameters

We alter the simulations by varying three parameters: the maximum node speed, the number of members in a group, and the number of data sources for the group. We only simulate one multicast group. The first variable represents different levels of node mobility, and thus the relative amount of network topology dynamics. Each node's speed is set to a value evenly distributed between 0 and the maximum node speed. The second parameter controls multicast group size. It translates into two factors, membership density and overall network traffic level, both of which affect the simulation at the same time. The last parameter controls traffic load. When the number of sources varies, the total amount of data traffic circulating within the network also changes.

The simulation runs under different combinations of parameter values. The standard case is for a multicast group with 2 sources and 10 members. All nodes are moving at a random speed between 0 and 2 m/s. When we are studying the effect of one particular parameter, it is set to different values while the other parameters are fixed at the standard case. The standard case has relatively low traffic load and low mobility. This way we can focus on the performance changes resulted from the change of the parameter of interest. For the effect of node mobility on protocol performance, the simulated range of maximum node speed is between 1 m/s and 20 m/s. For group sizes, simulations are run for groups with from 2 to 50 members. For the impact of number of sources, we use from 1 to 10 sources. For each simulation data point, results from multiple runs are averaged. These runs are for the combinations of 3 different random node placement/movement patterns and 3 different group member distributions.

3.4. Measurements

For each simulation run, we measure a handful of values. They are listed below. However due to the limitation of the length of the paper, not all measurement results against all variables will be shown.

- *Data packet delivery ratio.* Data packet delivery ratio is defined as the ratio of the number of data packets actually received by group members to the number of data packets that should have been be received. Since the membership is static during the entire simulation (and since there are no partitions), the number of data packets that should be received equals the product of the number of members and total number of data packets generated by all sources. The higher this value is, the more effective a protocol is at delivering data packets to receivers.
- *Data forwarding efficiency.* Data forwarding efficiency is defined as the "number of data packets transmitted per data packet delivered". This counts for all data packet transmissions by nodes including both source nodes and intermediate forwarding nodes. Since data payloads tend to be large in size compared to protocol control information, the number of data packet transmissions is usually the most important measure for bandwidth consumption of a multicasting protocol in terms of bytes. The number of data packets transmitted includes the transmissions of all packets containing a user payload. For DDM, this includes all data packets. For ODMRP, this includes both data packets and JOIN_QUERY packets since those contain user data as well.
- *Control packet overhead.* Control packet overhead is defined as the "number of control packets transmitted per data packet delivered". It shows relatively how much extra wireless channel access is required for the protocol to ex-

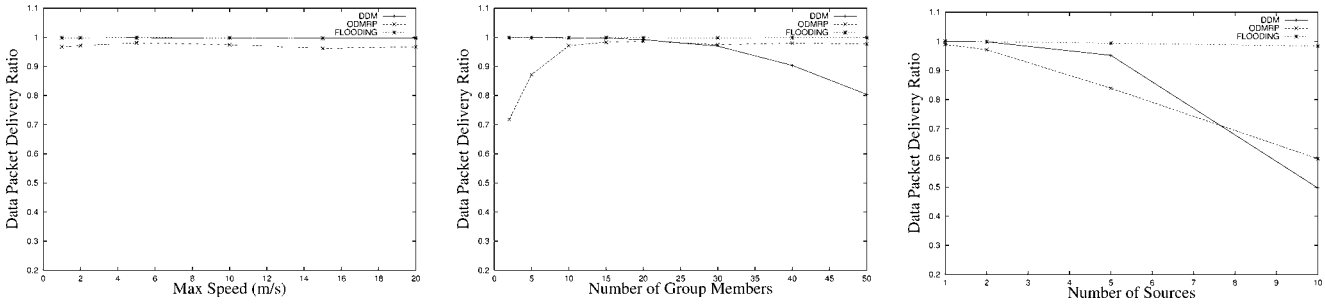

Figure 4. Data packet delivery ratio vs.: (a) max. node speed, (b) group size, (c) number of sources.

change control information. Here we only count pure protocol control packets that do not carry user payload. For DDM, this only counts for RSYNC packets. For ODMRP, the count includes only JOIN_REPLY and ACK packets. Such a measurement may also be referred to as "routing packet load" in some literatures.

- *Control byte overhead.* Control byte overhead is defined as the "number of control bytes transmitted per data byte delivered". The number of control bytes includes both the whole length of protocol control packets and embedded protocol control information in data packets. For DDM, the whole packet size of RSYNC packets and the DDM header bytes embedded in each data packet are both counted. For ODMRP, the control bytes only include those of the control packets. For JOIN_QUERY messages that contain data payload, only bytes used by ODMRP are counted. Such a measurement may also be referred to as "routing byte load" in some literatures.

3.5. Performance comparison between different protocols

3.5.1. Data packet delivery ratio

From figure 4(a) we can see that the throughput of ODMRP (which does not require any unicast routing support) is impressive across the whole mobility spectrum. So too is the delivery ratio of flooding. They are doing well because they use redundant transmissions for data delivery. ODMRP uses mesh forwarding topology instead of tree topology. Data packets are flooded within the mesh called "forwarding group" (FG). A simplified view of which is that the FG is constructed as the union of all Source Specific Trees for the multicast group. Given a group with 10 members and 2 sources, the FG can contain enough nodes to provide the redundancy needed for high delivery level. DDM over omniscient unicasting also offer good throughput at all mobility levels. Since it runs over nearly perfect routing information, this result is not surprising. In addition, since the network traffic level is low, there is little data packet collision.

When the group size changes, different protocols react differently. Flooding performs well for all group sizes. ODMRP offers high data delivery ratio when the multicast group is more populated. When the group is small, the FG contains a small amount of nodes and thus is relatively fragile. The FG is easily broken by node movements. When the group gets larger and larger, the FG size increases and thus becomes more tolerant to node movements, – if one path is broken, the FG may very well contain other backup paths. DDM behaves the opposite from ODMRP: it provide high data delivery ratio when the group is small but the performance degrades when the group gets larger. The DDM's data delivery ratio drops for large groups mainly due to high membership density. When membership density is high, communication channel is more likely becoming congested. Congestion related packet drop happens more frequently near sources. A data packet dropped near source has more negative impact on delivery ratio than a packet dropped father away. All these factors reduce the DDM's delivery ratio for larger multicast group.

Adding the number of sources increases network traffic level even faster. Very soon, the whole network is badly congested and all protocols suffer. Flooding is the least sensitive to network load as it's forwarding topology is the most redundant. And it does not need any control messages to maintain this forwarding topology. ODMRP is not as good as flooding due to two reasons. Firstly its forwarding topology is smaller than flooding's. Secondly ODMRP needs to exchange control message to construct and maintain its forwarding topology. These control messages can become victims too when the network is congested. When traffic level is very high, such control message loss costs the ODMRP delivery ratio. The DDM does not need control message other than RSYNC. However, as a tree forwarding protocol, DDM does not provide redundant forwarding path. Thus data packet losses have more serious impact on delivery ratio than packet losses in ODMRP.

3.5.2. Data forwarding efficiency

Figure 5 shows the average number of data transmissions required to deliver each data packet. For all simulated node mobility levels, ODMRP requires more data transmissions to deliver data packets to receivers. One reason for ODMRP's high data transmission number is its JOIN_QUERY message. This message, with data payload piggy-backed, is flooded throughout the whole network periodically. Another reason is that ODMRP uses mesh-wide flooding and the size of mesh is fairly large compared to the size of multicast group. Partially, the large size of the FG's is also the result of the time for a node to stay as an FG node. After an FG refreshing, a node which no longer on the sources-to-receivers paths may

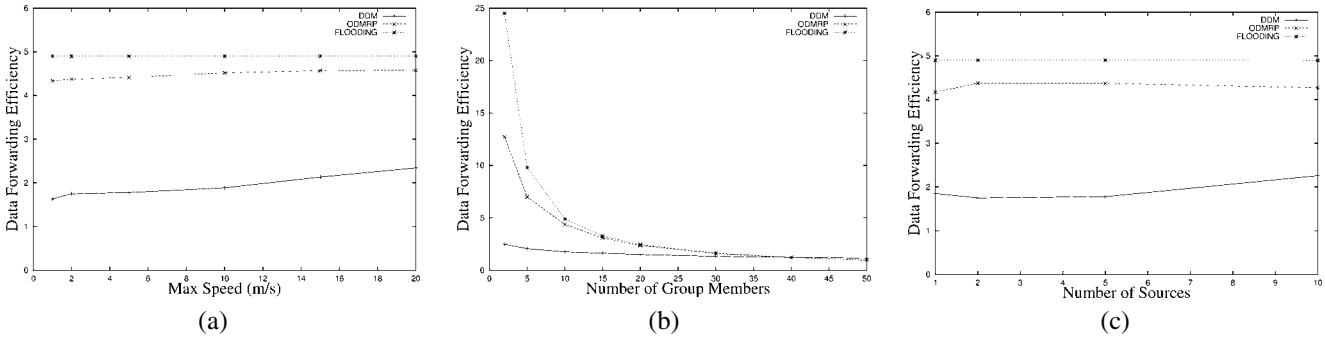

Figure 5. Data forwarding efficiency (number of transmissions to deliver one data packet) vs.: (a) max. node speed, (b) group size, (c) number of sources.

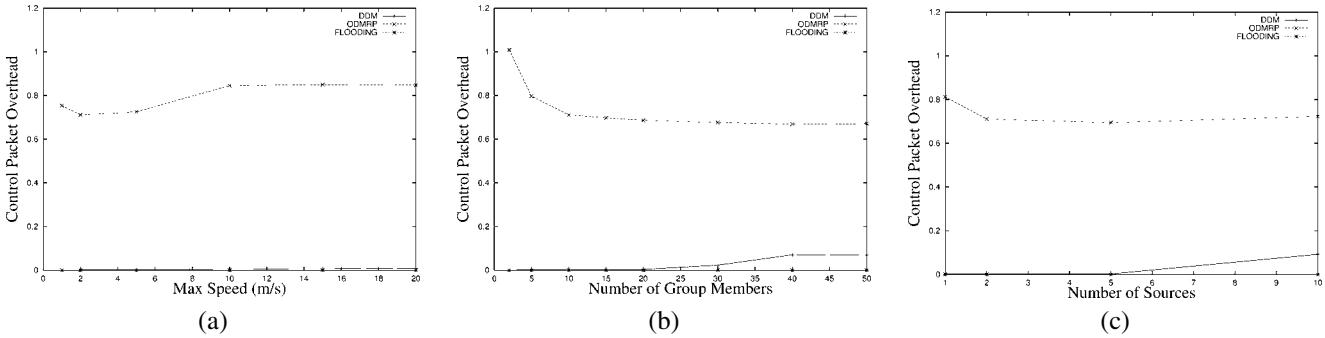

Figure 6. Control packet overhead vs.: (a) max. node speed, (b) group size, (c) number of sources.

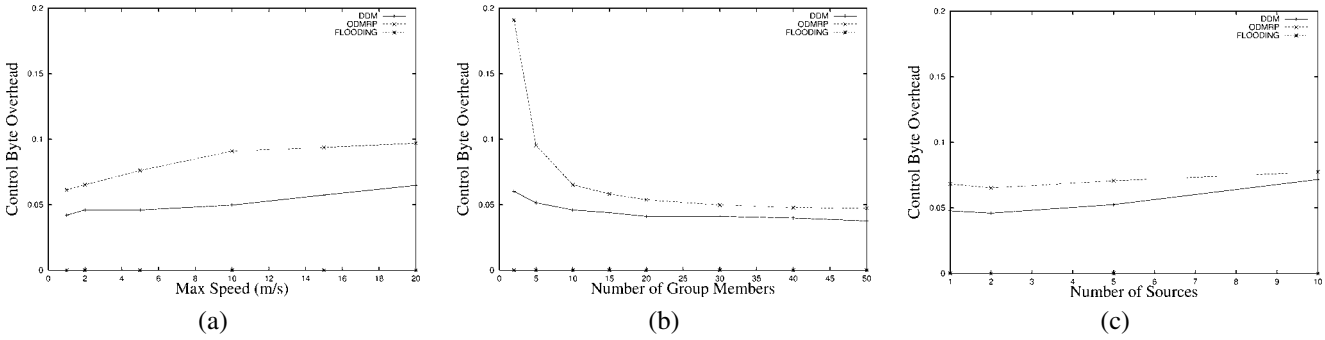

Figure 7. Control byte overhead vs.: (a) max. node speed, (b) group size, (c) number of sources.

still remain in the FG and keep forwarding data if its FG flag has not expired. Using tree forwarding, DDM uses the least number of transmissions to deliver each data packet.

When the group size increases, data transmission efficiency of both flooding and ODMRP increases due to the fact that more and more members are included in their forwarding topologies. DDM is relatively insensitive to group size changes. When group size increases, the overall forwarding efficiency tends to increase slightly because the paths between sources and receives are more likely to overlap. However, the increase is affected by the drop of number of data packets delivered as group size increases.

3.5.3. Multicast routing protocol load

DDM rarely uses pure control packets as it mostly rely on control information enclosed in data packets. Thus the number of protocol control channel accesses is kept low. The control packet overhead of DDM increases when data packet loss increases, which occurs when (a) node mobility increases, (b) group size increases, and (c) number of sources increases. There are two reasons for this increase. Firstly, the loss of data packets which contain destination list update will drive the downstream nodes to send out RSYNC messages to their upstream neighbors. Secondly, the lowered data delivery ratio decreases the denominator of the overhead and thus increases the overhead.

ODMRP's control packet overhead is higher than the rest. This is due to the mechanism it uses to construct the FG. When a group member receives a JOIN_QUERY, it replies with a JOIN_REPLY. This JOIN_REPLY will trigger all nodes between the originator of the JOIN_REPLY and sources to send more JOIN_REPLY towards the sources. After each member receives JOIN_QUERY, it needs to reply with JOIN_REPLY and such chain reaction is triggered again.

ODMRP's control packet overhead increases slightly as node mobility level increases. ODMRP does not employ

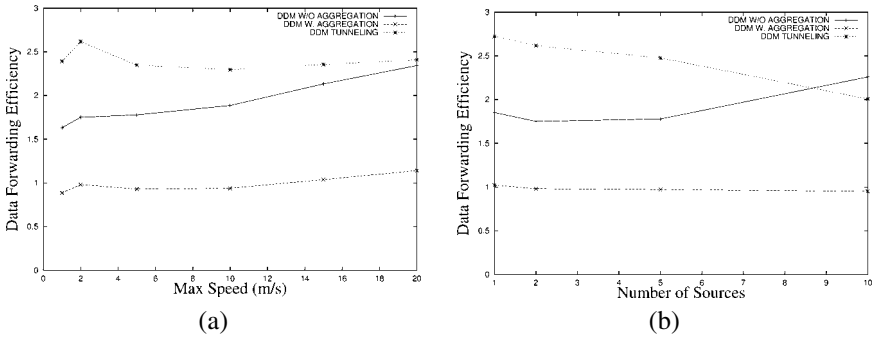

Figure 8. Data forwarding efficiency D vs.: (a) max. node speed, (b) number of sources.

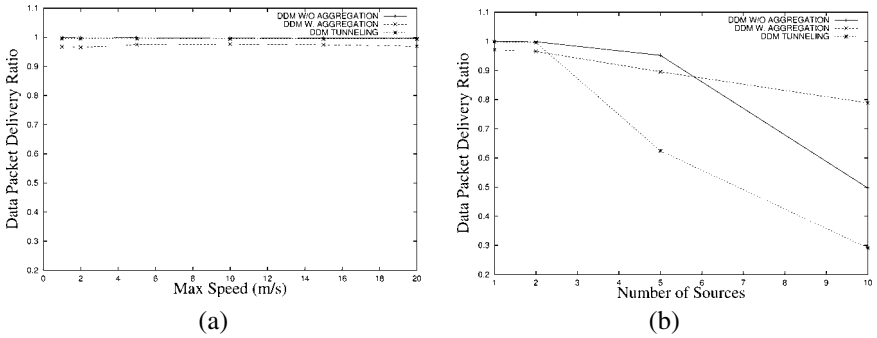

Figure 9. Data packet delivery ratio vs.: (a) max. node speed, (b) number of sources.

adaptive mechanisms to dynamically adjust FG refreshing period so the JOIN_QUERY's are flooded out at the same rate for all node mobility levels. The small increase is due to the following reason. When nodes move faster, the reverse routes to sources learned when receiving JOIN_QUERY become less stable as the next hops become more likely to move away. If the triggered JOIN_REPLY is targeting at a next hop which is no longer there, ODMRP will initiate broadcasting to seek routes, which causes the increase of control packet overhead when network topology changes faster. ODMRP's packet overhead decreases as group size increases and number of sources increases. Although the number of JOIN_REPLY messages generated also increases as the group size gets bigger and there are more sources, the denominator (the number of data packets delivered to members) increases faster.

ODMRP does not piggyback control information in regular data packets. Control information is enclosed in JOIN_QUERY packets. However, since these packets are flooded through the network at a fixed rate this portion of control bytes remains at the same level for different parameters. Other control bytes are from pure control packets. Therefore we see that roughly the change of byte overhead follows the change of packet overhead for ODMRP.

For DDM, things are different. Majority of the control bytes are from DDM headers enclosed in data packets. Thus we see less correlation between packet overhead and byte overhead. When the node mobility level increases, DDM has higher control byte overhead due to more frequent route changes. Thus DDM needs to use more D and R blocks to inform the downstream neighbors about the route changes. These blocks enlarge the DDM header size. When the group size increases, the byte overhead of DDM tends to decrease. Because paths between sources and receivers overlap each other more, one data transmission can effectively forward data for more members. Also since routes are relatively slow changing due to low mobility, most of the data packets only carry E blocks. However, this trend is offset by the drop of data delivery ratio. When number of sources increases, DDM experiences byte overhead increase. This is due to data packet loss, the use of more RSYNC, and large DDM header size caused by the use of R blocks for resynchronization.

3.6. Performance comparison between different flavors of DDM

Figure 8 shows the data forwarding efficiency for different flavors of DDM. Just as expected, the DDM tunneling is the least efficient in terms of the number of transmissions for delivering each data packet to receivers. This is also an evidence for why MANET can benefit from multicast routing. Even for relatively small groups, multicast routing uses significantly less network bandwidth for forwarding data packets than repeated unicasting.

DDM aggregation uses the least amount of data transmissions. This is because at forwarding path branching points, DDM aggregation only needs to transmit once to delivery a data packet to all downstream neighbors. On the other hand, DDM without employing aggregation would need to transmit one copy for each downstream neighbor.

Figure 9 shows the data delivery ratio for different variations of the DDM protocol. Because in this part of the study, all variations of the DDM's are running over omni-

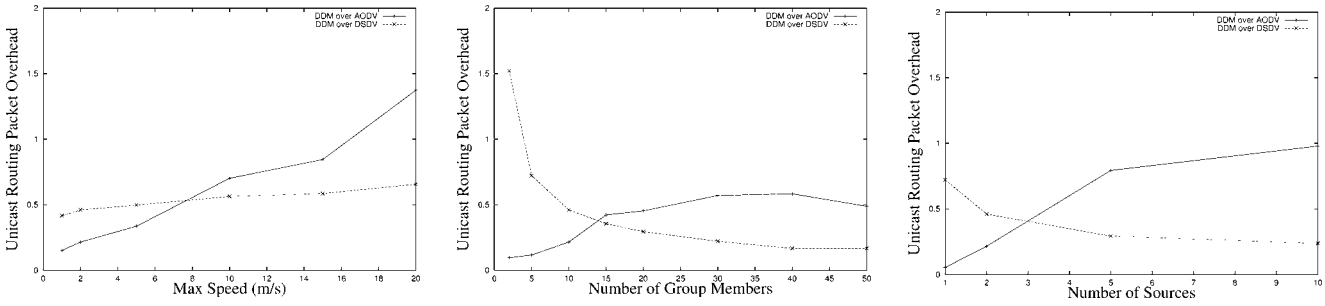

Figure 10. Unicast routing control packet overhead vs.: (a) max. node speed, (b) group size, (c) number of sources.

scient unicasting algorithm, there is little routing information error. This is the reason we see high data delivery ratio across all mobility levels. However, when there are more data circulating in the network, congestion becomes the main cause for packet loss and reduced delivery ratio.

From this figure we can also see that the impact of traffic level on protocol performance is heavy. The performance of DDM tunneling drops very fast because it uses the most data transmissions to deliver data. When the network bandwidth becomes congested, it is the first to suffer. Soon, DDM without aggregation also becomes the victim of network congestion. For DDM without aggregation, the delivery ratio drop rate after the turning point is even greater than DDM tunneling. This is because for DDM tunneling, one packet loss will only affect one receiver while for DDM without aggregation, one packet loss counts for all receivers downstream of where the packet loss occurs.

It is interesting to see how the delivery ratio of DDM with aggregation holding on. At low traffic level, the delivery ratio of DDM with aggregation is lower than the rest. This is because broadcast packets (which DDM aggregation uses for forwarding to multiple downstream neighbors) are easier to collide. On the other hand, a single packet loss also affect more downstream receivers. However, when the other variations of DDM suffer from poor delivery ratio, the benefit of having less data transmissions overcomes the negativities of using broadcast packets and the delivery ratio of DDM with aggregation exceeds the delivery ratios of its counterparts.

3.7. Performance comparison between DDM over different unicast protocols

In previous parts of the study, DDM is running above an omniscient unicast routing algorithm which always knows the network topology and the shortest paths towards destinations. This is obviously not realistic. In this section we present the measurements for DDM running above two unicast routing protocols designed for MANETs: the AODV and the DSDV. Another interesting point between these two protocols is that the AODV is an on-demand protocol and the DSDV is a full routing table protocol.

DDM has a way for adapting to on-demand type of unicast protocols. If a route for a receiver is in need but no available, the DDM will encapsulate the data packet in a unicast IP packet destined for the receiver and pass the packet to the unicast routing. In this case, a "need" for route is generated, and the unicast routing mechanism will be triggered to construct the requested route. Once the route becomes available, DDM can use the routing information without passing encapsulated data packets to the unicast routing protocol.

The performances for DDM over omniscient unicast routing is plotted in figures of this section for reference purpose.

Before we see each reading and their analysis, it is very useful to at first look at how much unicast control overhead is required to run each unicast routing protocol (figure 10). In these plots, unicast control overhead is defined by the number of unicast routing messages transmitted divided by the total number of data packets received by all receivers.

The absolute numbers of DSDV control messages are relatively stable. The change of DSDV overhead level curves is mainly due to the number of data packets received by receivers. This is reasonable because the route maintenance mechanism of DSDV is independent of traffic and topological parameters. On the other hand, AODV tries to adjust its behaviors based on network mobility level (link breakage) and traffic usage. Therefore we can see AODV is trying harder when the network is less stable and when more routes are needed (either because of larger group size or more data sources).

Figures 11–14 show the performance for DDM over different unicast routing protocols. We can see that in general, DDM over DSDV performs better than DDM over AODV. This is because in DSDV routes are always available and DDM can always use the route information. Some route information may be incorrect due to network dynamics, but those correct routes can still forward DDM data to receivers. On the other hand, to run above AODV the DDM often needs to trigger the route construction mechanism. Data packets sent before the route is constructed are lost.

The performance difference between DDM over DSDV and DDM over AODV widens as the network mobility level increases, multicast group size increases, and number of sources increases. This is partially because the increased number of control packets AODV injects into the network. Increased network traffic level affects the performance of DDM.

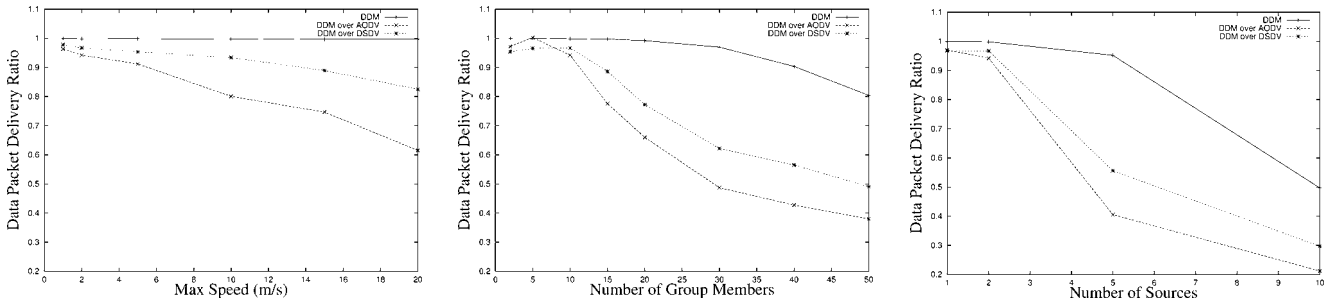

Figure 11. Data packet delivery ratio vs.: (a) max. node speed, (b) group size, (c) number of sources.

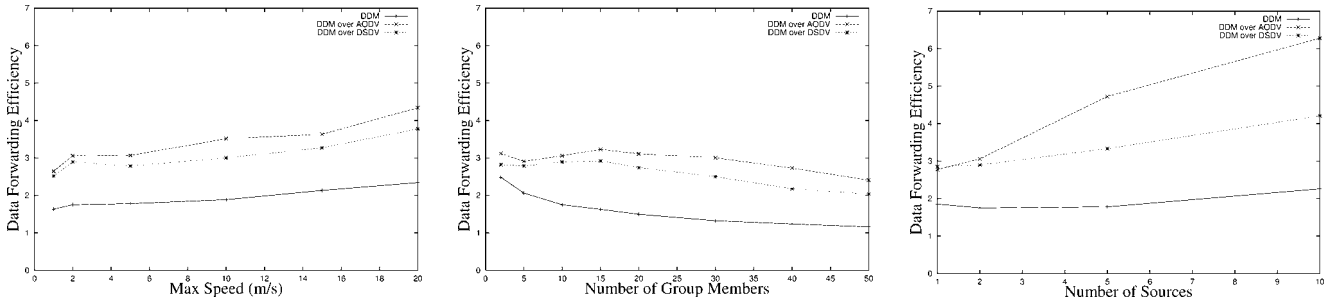

Figure 12. Data forwarding efficiency D vs.: (a) max. node speed, (b) group size, (c) number of sources.

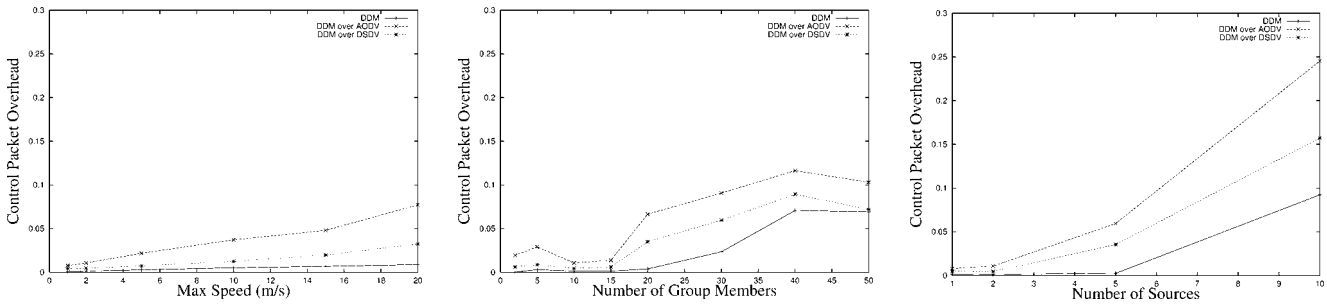

Figure 13. Control packet overhead vs.: (a) max. node speed, (b) group size, (c) number of sources.

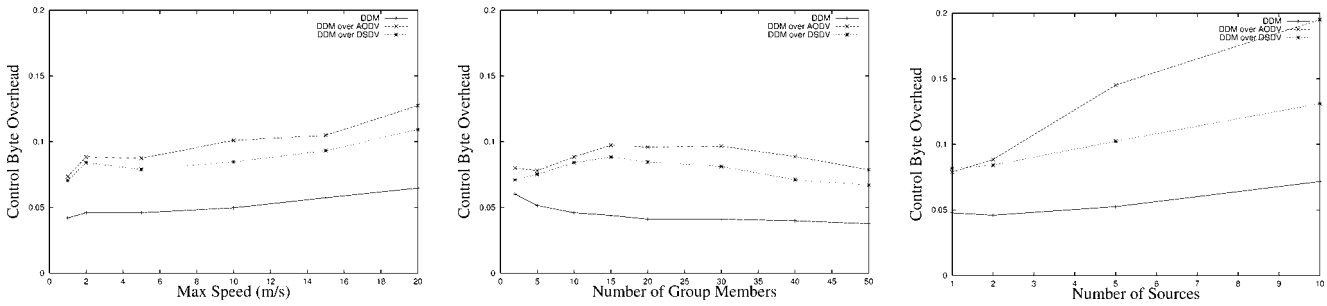

Figure 14. Control byte overhead vs.: (a) max. node speed, (b) group size, (c) number of sources.

4. Conclusion and future work

We have introduced an explicit multicast routing protocol intended for use with small multicast groups in mobile ad hoc networks of any size. The protocol is source-initiated and controlled, and uses in-band, destination-encoded data packet headers to control a distributed routing computation. The result is a flexible, efficient and robust protocol suitable for small multicast groups. The protocol offers the option of choosing between "stateless" and "soft-state" modes.

The simulation studies show that DDM is very efficient both in terms of data forwarding and control channel access. Especially for small groups, such characteristic stands out among all simulated protocols. When the network traffic level is low, the throughput of the protocol is affected by performance of the unicast routing protocol which DDM is operating upon. When the unicast protocol provides good route information, DDM delivery high throughput across all situations. DDM performs particularly well for small groups. It is nearly ideal for multicasting to small groups in networks that already have unicast routing support. On the other hand, ODMRP is better suited for another end of MANET multicasting, multicasting to large and dense groups.

The simulations also show that DDM is sensitive to network traffic level compared to protocols using redundant forwarding path. This is because data packet loss has more serious consequence on protocol performance. When data packet loss occurs, the DDM control information embedded in data packets is also lost. With no redundant data forwarding, packet loss also reduces packet delivery ratio.

Further studies suggest that DDM aggregation is better for applications which injects more data into the network because under this configuration DDM uses less data transmits to forward data. Therefore the increase of network traffic level is slower and the protocol performance is affected less. Compared to on-demand type of unicast routing protocol, DDM prefers a full routing table unicast protocol running underneath for borrowing routing information from. Full routing table type of unicasting has two advantages for supporting DDM: DDM does not need to wait for new routes to be constructed before DDM can use it; the unicast control traffic level is rather stable so for larger groups or large number of sources, the unicast control traffic does not increase too fast to affect DDM performances.

References

[1] E. Bommaiah, A. McAuley and R. Talpade, Amroute: ad hoc multicast routing protocol, Internet Draft, `draft-talpade-manet-amroute-00.txt` (February 1999), work in progress.

[2] R. Boivie, A new multicast scheme for small groups, IBM Research Report RC21512(97046) (June 1999).

[3] J. Broch, D. Maltz, D. Johnson, Y. Hu and J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: *Proceedings of ACM/IEEE MOBICOM'98* (October 1998) pp. 85–97.

[4] Y. Chu, S. Rao and H. Zhang, A case for end system multicast, in: *Proceedings of the International Conference on Measurements and Modeling of Computer Systems* (2000).

[5] J.J. Garcia-Luna-Aceves and E. Madruga, A multicast routing protocol for ad-hoc networks, in: *Proceedings of IEEE INFOCOM'99* (March 1999) pp. 784–792.

[6] J. Jercheva, Y. Hu, D. Maltz and D. Johnson, A simple protocol for multicast and broadcast in mobile ad hoc networks, Internet Draft, `draft-ietf-manet-simple-mbcast-01.txt` (July 2002), work in progress.

[7] L. Ji and M. Corson, Light-weight adaptive multicast, in: *Proceedings of IEEE GLOBECOM'98* (November 1998).

[8] L. Ji and M. Corson, Differential destination multicast (ddm) specification, Internet Draft, `draft-ietf-manet-ddm-00.txt` (July 2000), work in progress.

[9] L. Ji and M. Corson, Differential destination multicast – a manet multicast routing protocol for small groups, in: *Proceedings of IEEE INFOCOM 2001* (April 2001).

[10] D. Johnson and D. Maltz, Dynamic source routing in ad hoc wireless networks, in: *Mobile Computing*, eds. T. Imielinski and H. Korth (1996) pp. 153–181.

[11] S. Lee, W. Su and M. Gerla, On-demand multicast routing protocol (ODMRP), Internet Draft, `draft-ietf-manet-odmrp-02.txt` (June 1999), work in progress.

[12] S. Lee, W. Su, J. Hsu, M. Gerla and R. Bagrodia, A performance comparison study of ad hoc wireless multicast protocols, in: *INFOCOM 2000* (March 2000).

[13] D. Ooms and W. Livens, Connectionless multicast, Internet Draft, `draft-ooms-cl-multicast-01.txt` (October 1999), work in progress.

[14] T. Ozaki, J. Kim and T. Suda, Bandwidth-efficient multicast routing protocol for ad hoc networks, in: *Proceedings of IEEE ICCCN'99* (October 1999).

[15] C. Perkins and P. Bhagwat, Highly dynamic destination-sequenced distance vector routing (DSDV) for mobile computers, in: *Proceedings of SIGCOMM'94* (August 1994).

[16] C. Perkins, E. Royer and S. Das, Ad hoc on-demand distance vector (aodv) routing, Internet Draft, `draft-ietf-manet-aodv-04.txt` (October 1999), work in progress.

[17] E. Royer and C. Perkins, Multicast using ad hoc on-demand distance vector routing, in: *Proceedings of ACM/IEEE MOBICOM'99* (1999).

[18] P. Sinha, R. Sivakumar and V. Bharghavan, Mcedar: Multicast core extraction distributed ad hoc routing, in: *Proceedings of IEEE WCNC'99* (September 1999).

[19] I. Stoica, T. Ng and H. Zhang, Reunite: A recursive unicast approach to multicast, in: *Proceedings of IEEE INFOCOM'00* (March 2000).

[20] UC Berkeley and LBL and USC/ISI and Xerox PARC, ns notes and documentation, `http://www-mash.cs.berkeley.edu/ns` (October 1999).

[21] C. Wu, Y. Tay and C. Toh, Ad hoc multicast routing protocol utilizing increasing id-numbers (amris) functional specification, Intenet Draft, `draft-ietf-manet-amris-spec-00.txt` (November 1998).

Lusheng Ji is a Member of Research Staff at the Pervasive Computing Research Department of the Fujitsu Laboratories of America, Inc. Dr. Ji's current research interests include protocols and applications for mobile, wireless networks. He received his Ph.D. in computer science from the University of Maryland at College Park in 2001.
E-mail: lji@fla.fujitsu.com

M. Scott Corson is the Director of Networking at Flarion Technologies. Dr. Corson has worked on multiple access and routing technologies for mobile, wireless networks since 1987, and has been active in the Internet Engineering Task Force (IETF) since 1995. He co-organized and currently co-chairs the IETF Mobile Ad hoc Networks Working Group, a body chartered to standardize mobile routing technology for networks of wireless routers. He was previously on the faculty of the University of Maryland at College Park from 1995–2000, and was a consulting network architect for British Telecomm (BT) Labs working on the design of a fixed/mobile-converged IP network architecture. He has a Ph.D. in electrical engineering from the University of Maryland.
E-mail: corson@flarion.com