Internet Draft
Expiration: January 12, 2001

L. Ji, UMD
M. S. Corson, UMD

July 12, 2000

Differential Destination Multicast (DDM) Specification

<draft-ietf-manet-ddm-00.txt>

Status of this Memo

Abstract

This draft describes a multicast routing protocol for mobile ad hoc networks (MANETs). The protocol---termed Differential Destination Multicast (DDM)---differs from common approaches proposed for ad hoc multicast routing in two ways. Firstly, instead of distributing membership control throughout the network, DDM concentrates this authority at the data sources (i.e. senders) thereby giving senders knowledge of group membership. Secondly, 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 be stored in participating nodes,

this approach also provides the option of stateless multicasting. Each node independently has the choice of maintaining cached forwarding state, or requesting 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 networks of any size.

1. Introduction

The Differential Destination Multicast (DDM) protocol is designed with several goals in mind.

The first goal is to minimize multicast routing protocol's communication channel use especially the number of channel access. Due to the fact that commonly all MANET nodes operate in broadcast media, the extra cost of each protocol channel access imposed by lower layers is very expensive compared to the case that protocols working primarily with point-to-point links.

The second purpose of the design of DDM is to introduce a protocol which enables centralized group membership management. Most MANET multicast protocols [1,2,3,4] follow the footsteps of their wired network counterparts [5,6,7]. They distribute membership control (or lack thereof) over the network. The protocols themselves do not have the mechanism to prevent any party from joining the group. Thus, the sources have no way to control how its data is distributed. If a data source wants to limit the distribution scope, it has to use other external means such as key distribution. Even so, such external mechanisms do not prevent data being received by unauthorized parties. In many applications, especially for small multicast groups, a centralized approach is more in favor because it establishes a better ground for implementing security/pricing features.

The third goal is to make the protocol as flexible as possible. Due to the highly dynamic nature of MANETs, the protocol should leave enough configuration space so each node may adaptively choose its own operation parameters according to its own behavior and local environment.

The resultant protocol is significantly different from other MANET multicast protocols proposed to this group. Unlike in other protocols, DDM lets the sources control the membership. All

membership changes need to be directed to the sources. Also different from traditional multicasting, DDM encodes multicast receiver (destination) addresses in multicast data packets using a special DDM Data Header. This way it is not necessary for each node to maintain per-session multicast forwarding states. Thus, it is

more scalable with respect to the number of sessions. Although the number of multicast destinations of each session is limited by header/packet size, given the relatively small size (compared to the wired Internet) of MANETs, such approach may well serve a large population of applications.

2 Terminology

Node: A device that implements IP.

Multicast Session: The basic unit for multicasting organization. Each multicast session is identified by a group ID and a data source.

Multicast Destinations: Each node with multicast data receiving application running is a multicast destination.

Forwarding Set (FS): The address set each DDM node uses to record to which multicast destinations it needs to forward multicast data.

Direction Set (DS): The address set each DDM node uses to record via a particular next hop, to which multicast destinations multicast data are forwarded.

3 Protocol Overview

3.1 Protocol Description

The proposed approach is, motivated, in part, by the approach to unicast routing of Dynamic Source Routing (DSR) [10] and derived, in part, from the work of [8,9]. The latter has recently been named Explicit Multicasting (or xcast for short).

In DDM, the source controls multicast group membership to ease certain aspects of security administration. More importantly, and a departure from all proposed MANET multicast protocols to date, DDM

encodes the multicast destinations in each data packet header in a fashion different from [8,9]. 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. When one

intermediate node receives a DDM data packet, it need only 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 that much *faster*, with DDM then making immediate use of this knowledge.

While the fixed network and MANETs can benefit from stateless, explicit multicasting because of its savings in storage complexity, it is desirable to follow this approach in MANETs for other reasons as well. Firstly, media access is expensive in wireless broadcast channels. 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 control packets generated by the protocol. Therefore such approach can be more efficient overall in many scenarios. Secondly 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, one argument against explicit multicasting in the wired Internet is that the relatively complex header processing prevents fast path forwarding at routers. Presently in MANETs, the effect of per-packet processing on forwarding rate is not significant relative to bandwidth and energy constraints.

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 work of [8,9]. In this mode, as data packets with in-band information are routed through the network, each node along the forwarding path remembers the destinations to which it forwarded the last time and how the data was forwarded (i.e. which next hop was used for each destination). By storing this information, the protocol no longer needs to list all the destinations in every data packet header. When changes occur in the underlying unicast routing, an upstream node only needs to inform its downstream neighbors (i.e. its next hops) regarding the *differences* in destination forwarding since the last packet; hence, the name "Differential Destination" Multicast. Reporting only these differences significantly reduces DDM header sizes. 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 DDM routing 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.

Also different from Explicit Multicasting, DDM takes advantage of the broadcasting medium. Multiple DDM blocks, which is the per next hop forwarding information unit, may be aggregated together so one data packet transmission can forward data to multiple neighbors (next hops). On the other hand, DDM again provides the flexibility for each forwarding node to choose by itself. In an environment discriminates against broadcasting but in favor of unicasting, if high data throughput is critical to the application, forwarding nodes may decide not to aggregate DDM Blocks together. In this case, multicast data is forwarded in unicast envelop to each next hop. Within each such packet, the DDM header only contains the DDM Block(s) for the intended receiver.

DDM is not a general purpose multicast protocol in the conventional sense. The header-encoded destination mechanism does not scale well with group size. The stateless mode (in common with other xcast-based approaches), however, does scale well with the number of multicast groups, as no per-group 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 header size and save bandwidth. This approach,

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.

3.2 Route Correctness

DDM replies on the underlying unicast routing protocol to provide the "next hop" information. DDM is loop-free as long as the underlying unicast routing is loop-free. Since the DDM routing (FS to DS 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. Some data packets may have been errantly sent using erroneous route information. In this case, these data packets will either dropped when the IP TTL reaches zero, or will exit the loop and head towards the destination when some forwarding node finally corrects the loop.

The same argument holds for broken routes. In many other multicasting protocols, broken links, if used by the protocol forwarding topology, should be discovered as soon as possible so they can be repaired. In DDM, it is not necessary to inform DDM when a link used in multicast forwarding goes down. The old DS and the old FS on either end of a broken link are left alone to be deleted on time-out. As soon as the unicast routing recovers, DDM will use the new next hops (if available) when forwarding the next multicast data

packet. With these new next hops, DDM will use new and likely different DS's.

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 in DDM routing. 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 when set up a new DS for this new link.

Data packets may be lost during transmission. When nodes are opearting in "statefree" mode, each data packet is processed independently from the previous packets. Therefore the loss does not affect the forwarding correctness of future packets. When operating "soft-state" mode, if the lost packet does not contain any multicast destination updates, it is simply a data loss. No future forwarding

will be affected eiother. If the packet does contain updates, the loss may de-synchronize the FS sets on the receiving ends from the matching DS sets on the sending ends. However, since all packets contain DDM Block Sequence Number, the receiving end of the lossy link will discover the loss of an informative DDM header when receiving the next data packet. A RSYNC is then sent back asking for a R packet to re-synchronize the FS sets and the DS sets.

4 DDM Protocol Specification

DDM asumes that there is a unicast routing protocol running on each MANET node. Through this unicast routing protocol, DDM can obtain the "next hop" information for a particular destination. Also DDM may use this unicast routing protocol to deliver unicast packets.

4.1 Data Structures

At each source node, there is a Member List. This list is used to keep track of the admitted members for each session the source is originating data. It contains the addresses of all the multicast group members who have successfully joined the session.

At each node, there is one Forwarding Set (FS) for each multicast session. It records to which destinations this node forwards multicast data. At the source node, the FS contains the same set of node addresses as the ML. At other nodes, the FS is actually the union of several subsets. Each subset FS_k records the multicast destinations included in data packets received from upstream neighbor k. After receiving a data packet from an upstream neighbor k, the receiving node will update the corresponding subset FS_k set and then the unioned set FS is recalculated.

Associated with each set FS_k, there is one sequence number SEQ(FS)_k. This is used to record the last DDM Block Sequence Number seen in a received DDM data packet from upstream neighbor k. The reason for this sequence number is to detect the loss of DDM data packets containing forwarding set updates. For each multicast session, there is a data sequence number cache kept by each node. It is used to record recently seen data sequence numbers. This cache is used to avoid duplicated forwarding of the same data packet. In

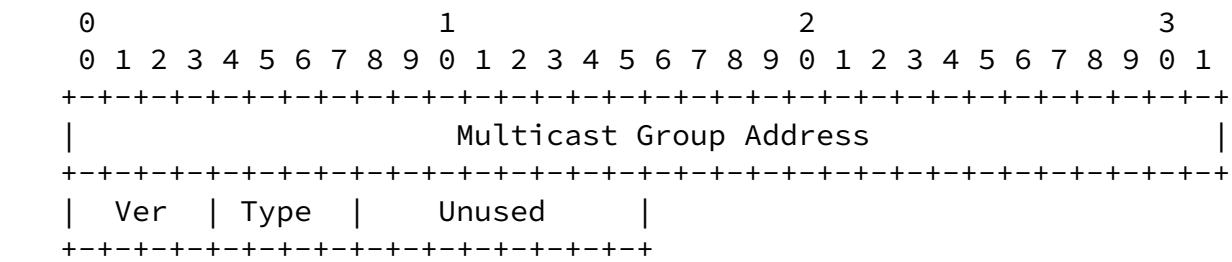
addition, each node needs to remember if itself is a receiver member of a particular multicast session.

When a node is forwarding a data packet received from an upstream neighbor k, it duplicates the data packet (if forwarding to multiple neighbors is necessary) and sends the packet(s) to the next-hop neighbor(s). The FS contains all the multicast destinations for this session to which this node needs to forward. However, these destinations may be reached via different paths (i.e. next hops). Therefore, the FS needs to be partitioned into subsets according to the next hops these destinations require. The destinations in the FS who use the same downstream neighbor (next hop) will be put into the same subset. These subsets resulting from partitioning the FS are called Direction Sets (DS) (a DS_l exists for each downstream neighbor l). For each DS_l there is also a sequence number SEQ(DS)_l. It is initialized to 0 when the DS_l is created for the first time. Each DS_l contains a "Forced Refreshing" (FR) flag. This flag has three states: no forcing, forcing once and forcing always. The use of this flag will be explained in sections 4.4 and 4.5.

4.2 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 five types of control packets: JOIN, ACK, LEAVE, RSYNC, and CTRL_DATA. All of them are unicast IP packets. The first three are used only by the membership control part of the algorithm. The fourth is used to coordinate between a pair of neighboring nodes. The last one is used to encapsulate multicast data so it can be delivered to a particular multicast destination via unicast routing. Regular multicast data packets are D-class IP packets with DDM Data Header. However, multicast data may also be encapsulated within unicast IP packets to be forwarded to the next hop neighbors if uncasting is used as the primary data transmission means. This encapsulation is different from the CTRL_DATA because it is only for downstream neighbors one hop away.

4.2.1 Base DDM Control Packet Format



All DDM control messages contain this common base. Depending on the control message type, certain extensions may follow. The "Ver" field contains the DDM version number. The "Type" field contains the control message type. The "Multicast Group Address" contains the ID of the multicast group of interest.

4.2.2 JOIN Packet

This packet is used to express a node's interest towards a particular multicast session. This packet is a unicast IP packet sent from the joining node to the multicast session source. The "Type" field of the base format is set to 0. The joiner's address is used as the source address of the IP packet. The multicast session source's address is used as the destination address of the IP packet. The TTL field of the IP packet is set to the estimated NETWORK_DIAMETER.

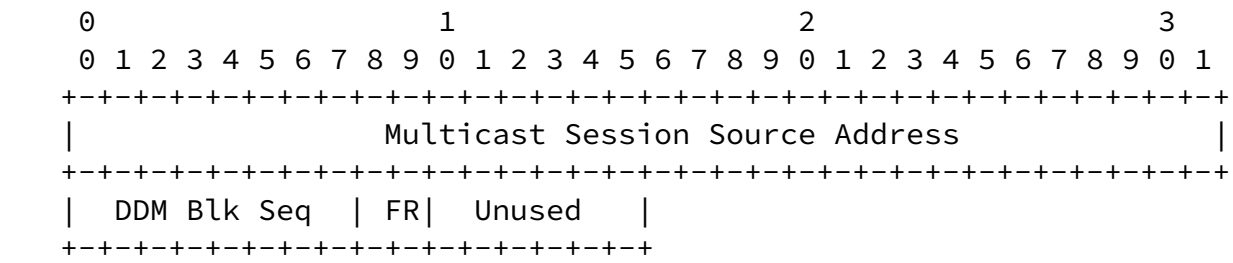
4.2.3 ACK Packet

This packet serves as a notification of admission to a particular multicast session. It is a unicast IP packet sent from the multicast session source to the interested joining node. The "Type" field of the base format is set to 1. The joiner's IP address is used as the destination address of the IP packet. The multicast session source's ID is used as the source address of the IP packet. The TTL field of the IP packet is set to the estimated NETWORK_DIAMETER.

4.2.3 LEAVE Packet

This packet serves as a resignation from a particular multicast session. It is a unicast IP packet sent from the multicast member node to the multicast session source. The "Type" field of the base format is set to 2. The leaving node's address is used as the source address of the IP packet. The multicast session source's address is used as the destination address of the IP packet. The TTL field of the IP packet is set to the estimated NETWORK_DIAMETER.

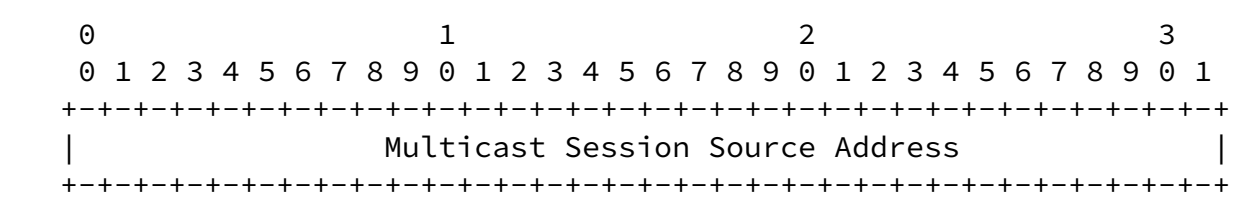
4.2.3 RSYNC Packet



RSYNC extension

RSYNC (Request to Synchronization) packet is used to synchronize the multicast destination address sets between a pair of neighboring nodes. It is unicast IP packet sent from the downstream neighbor to the upstream neighbor. The communicating neighbors are used as the source and destination fields of the IP header. The TTL field of the IP header is set to 1. The "Type" field of the base format is set to 3. Other than the common base, RSYNC packet also contains the above extension. In this extension, the re-synchronization requester needs to specify the multicast session source and the DDM Block Sequence Number. The "FR" (Force Refreshing) flag can have the value of "no forcing", forcing once", and "forcing always".

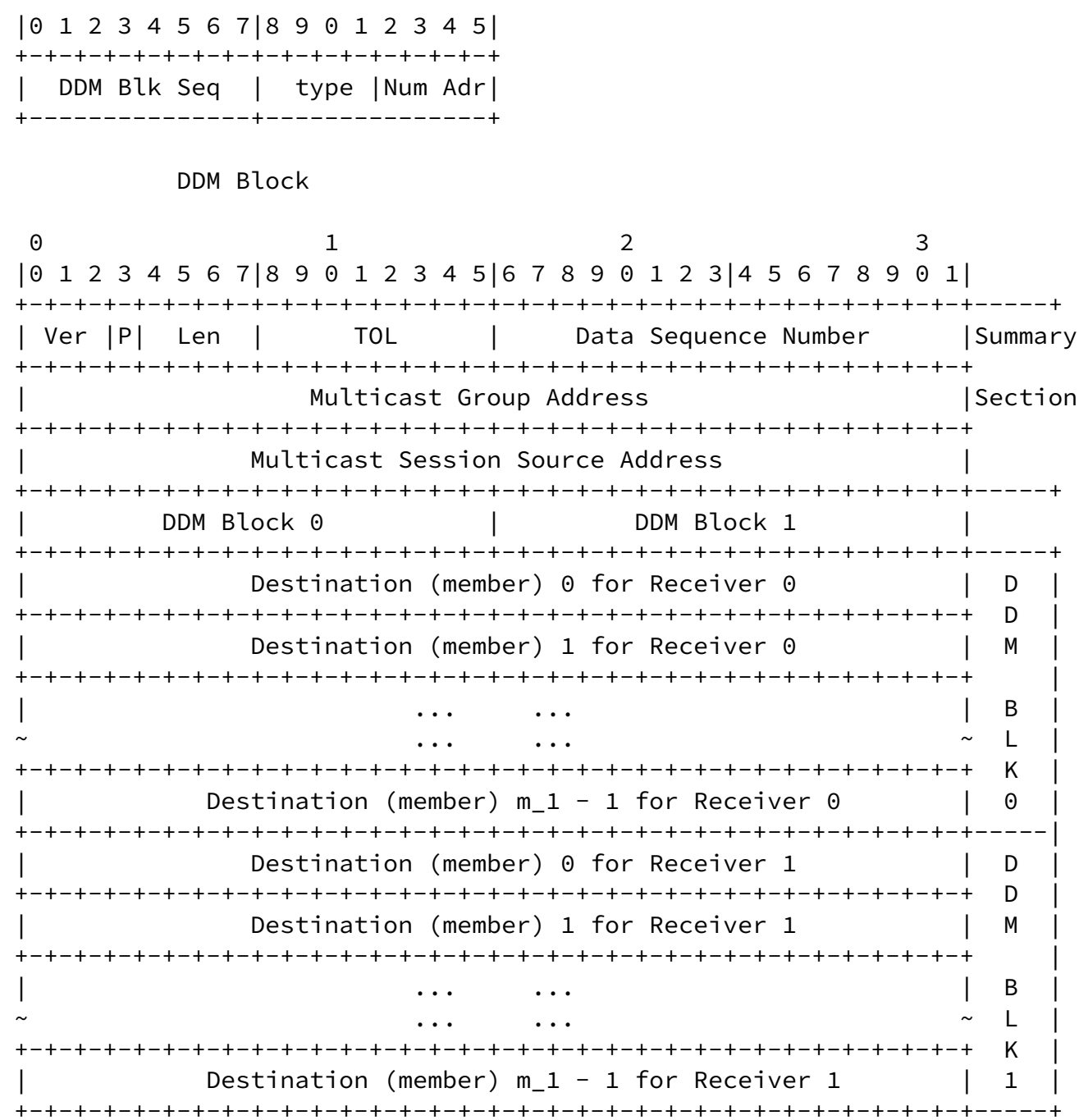
4.2.3 CTRL_DATA Packet



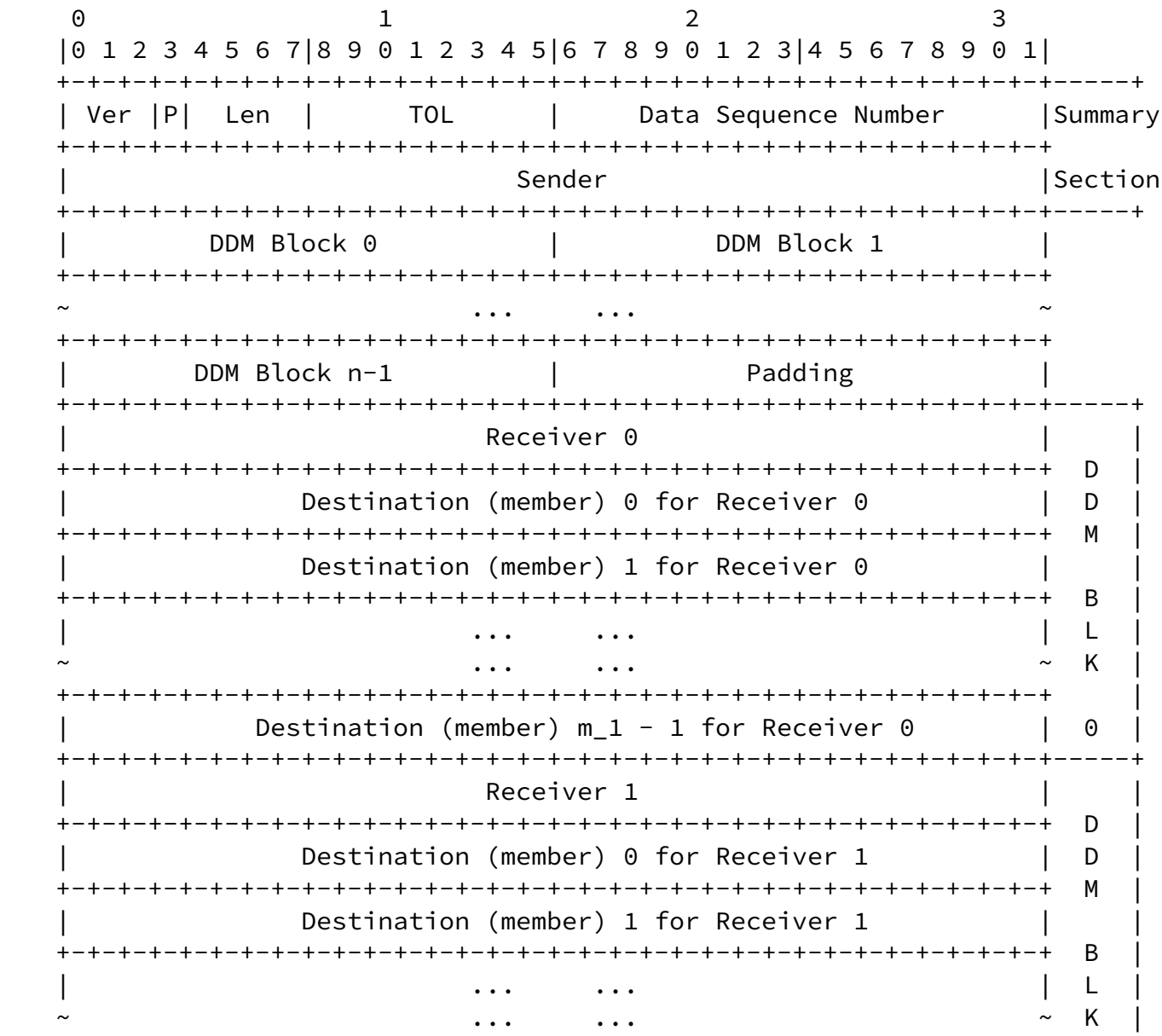
CTRL_DATA extension

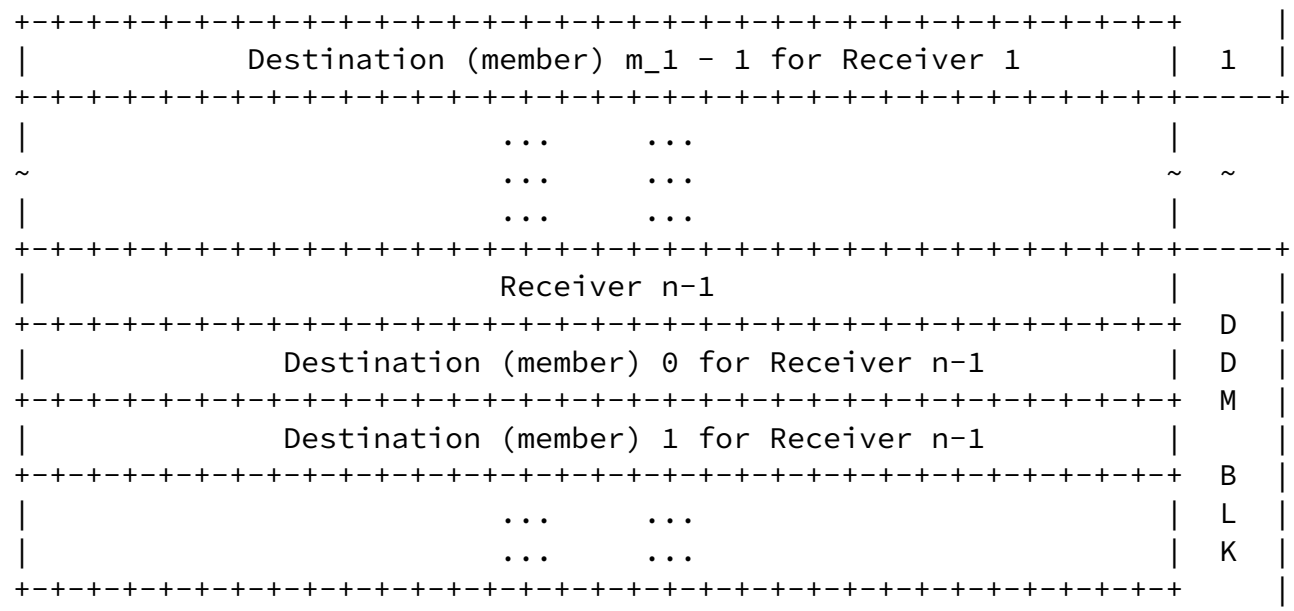
This packet is used to encapsulate multicast data in a unicast packet so it can be forwarding via unicasting to reach a particular multicast destination. It is a unicast IP packet sent from the forwarding node who needs to forward the data to the multicast destination. The TTL field of the IP packet is set to the estimated NETWORK_DIAMETER. The "Type" field of the base format is set to 4. It also carries the above extension so upon receiving such packet, the multicast destination can identify to which multicast session the data belongs.

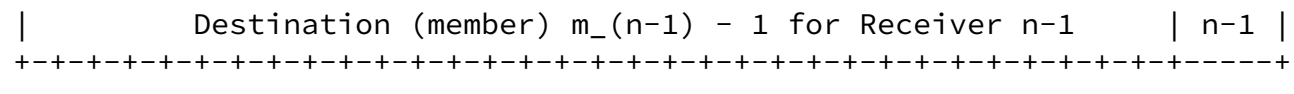
4.2.4 DATA Packet


|0 1 2 3 4 5 6 7|8 9 0 1 2 3 4 5|
DDM Blk Seq
type
Num Adr
DDM Block
0 1 2 3
|0 1 2 3 4 5 6 7|8 9 0 1 2 3 4 5|6 7 8 9 0 1 2 3|4 5 6 7 8 9 0 1|
Ver
P
Len
TOL
Data Sequence Number
Summary
Multicast Group Address
Section
Multicast Session Source Address
DDM Block 0
DDM Block 1
Destination (member) 0 for Receiver 0
Destination (member) 1 for Receiver 0
Destination (member) m_1 - 1 for Receiver 0
Destination (member) 0 for Receiver 1
Destination (member) 1 for Receiver 1
Destination (member) m_1 - 1 for Receiver 1
D
D
M
B
L
K
0
D
D
M
B
L
K
1

DDM Data Header Format (Unicast)







DDM Data Header Format (Multicast)

The above are the formats for DDM Data Header. This header is inserted into data packets before the data payload. The IP data packet's destination address may be set to the multicast group address while the multicast session source address is used as the source address of the IP packet. This is shown in "DDM Data Header Format (Multicast)". Or, when forwarding node chooses to use unicast to forward a data packet to a next hop, it needs to use its own address as the IP source and the next hop address as the IP destination address. The group address and the multicast session source address are hidden in the DDM Data Header. This is shown in "DDM Data Header Format (Unicast)".

At the beginning of the DDM Data Header it is the summary section. In the summary section, after the DDM version field, it is the POLL flag for membership refreshing. The "Len" field indicates how many DDM Blocks there are in the header. The "TOL" (Time Of Life) field indicates the life time of the FS set on the intended receivers of this packet. Then it is followed by the DDM data sequence number. The next is the address of the packet sender (the forwarding node). In the case of unicasting, the packet sender is in the IP source

field. Instead, the summary section contains the multicast group address and the session source address.

The rest of the header contains all the DDM Blocks and the addresses used by them. Each DDM Block contains a Block Sequence Number, a Block Type, and a Number of Addresses field. There are four types of DDM Blocks. They are Empty (E), Replace (R), Incremental Difference (Di), and Decremental Difference (Dd) blocks, and their types are 0, 1, 2, and 3 respectively. E Block does not list any multicast destination. R block lists all multicast destinations to which the receiving node needs to forward. Di and Dd Blocks (both may be referred as D blocks in the rest of the draft) contain different destination addresses from the last forwarding. The multicast destination lists used by the blocks will be referred as L list (for R header), L_i list (for Di Block), or L_d list (for Dd Block). Since R and D Blocks contain destination list updates, they may be referred as "informative" in the rest of the description.

When a node uses broadcast to forward data to next hops, the first node address used by any DDM Block is always the intended receiver for this DDM Block. However, when a node uses unicast to forward data to individual next hop, this intended receiver field is omitted since the intended receiver is the IP destination. Other addresses

used by a DDM Block are the multicast destinations to whom data needs to be forwarded by the intended receiver of this DDM block. To align the header better, all DDM Blocks are concatenated together, then followed by an archive of all the addresses used by all the DDM blocks. These addresses are serialized in the same order as the DDM Blocks using them.

4.3 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, and the remaining description applies to a single source. The source acts as an admission controller for the information it is sending. When a node (the "joiner") is interested in a particular multicast session, it needs to join the session by unicasting a JOIN message to the source for that session. This JOIN message include the ID of the group to which the node wants to join. The joiner's ID and the

source ID are used as the source and destination fields of the IP header of the JOIN message. After receiving the JOIN message and verifying its own role of being the source for the interested multicast session, the source decides if the JOIN can be accepted. The admission policies and protocols are beyond the scope of this draft.

Upon receiving a JOIN message, if the joiner passes the session's admission requirements, the source adds the ID of the "joiner" into its Member List (ML). The source then acknowledges the JOIN message by unicasting an ACK message back to the joiner.

Back at the joiner, after the JOIN message is sent, the joiner waits for one JOIN_WAITING_PERIOD. If it has not received an ACK till the end of this period, the joiner needs to resend the JOIN message. When sending the JOIN messages, the waiting period is exponentially backed off for every consequent JOIN sent. That is, the waiting period for the i-th JOIN message is:

(2 ** (i - 1)) * JOIN_WAITING_PERIOD

The sending of JOIN messages is stopped when either an ACK message is received or MAX_JOIN_RETRY JOIN messages have been sent and the waiting period for the last JOIN message has passed. The first case indicates a successful join while the latter indicates failure. Reception of a DDM data packet intended for this joiner prior to reception of an ACK may or may not indicate a successful join, depending on the security mechanism (if any) in use and whether or not an ACK is required as part of this mechanism. Security-related mechanisms are beyond the scope of this draft. This draft

essentially describes an insecure session. When a join process is successfully completed on ACK reception, any activate join waiting timer is canceled and no further JOIN messages are generated.

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. The source needs to be able to purge stale members. In DDM member refreshing is source-initiated. Once every MEMBERSHIP_REFRESH_PERIOD data packets, the source sets the 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 after MAX_REFRESH_TIMEOUT such polling data packets have been sent and there is still no JOIN message received from one member, the source assumes that this member has left the multicast session. This member is then removed from the ML and excluded from future forwarding computations. JOIN "implosion" at the sender is not expected to be a problem due to small expected group sizes and different hop distance between members and the source. If necessary, random delay jitter may be added to the transmission times of JOINs sent in response to POLL packets 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 leaving 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 message, MAX_LEAVE_RETRY LEAVE messages may be unicast to the source when a node leaves the session. After these transmissions, a member node removes any information associated with the multicast session. The source may also dismiss a receiver by removing it from the ML at any time if its security mechanism suggests so.

4.4 Forwarding Processing

Most of the processing consists of set operations. The computation complexity on each participating node is bounded by O(n log n), with n being the number of members in the multicast session.

When a node receives a multicast DDM data packet from an upstream neighbor k, it first tries to locate the "DDM Block" intended for itself. Walking through the DDM Blocks in the DDM Data Header, the receiving node matches the 1st node address used by each DDM Block

(the intended receiver) with its own. If it is indeed the intended receiver of one of the DDM Blocks, the processing continues. Otherwise, the data packet is dropped and the forwarding processing stops. If the received data is unicast, this step is not necessary

because only the intended receiver will receive such data packet.

After finding the DDM block, the receiving node compares the Data Sequence Number in the summary section of the DDM Data Header with the local data sequence number cache for the same multicast session. If this packet has been seen before, it is discarded and no further processing is needed. If the packet is new, its data sequence number is inserted into the cache. DDM also needs to see if itself is a multicast destination. If so, a copy of the data packet is made and the DDM Data Header is striped off from this data packet. The restored data is re-injected into the network kernel of the node for local delivery.

The receiving node extracts all multicast destination addresses for the DDM Block from the header. The upstream neighbor is also identified, either from the "Sender" field of the summary section or the IP source of the packet depending on if the data packet is received as D-class data packet. Then the receiving node accesses the FS_k set for this upstream neighbor k. If no such set is found and the received DDM Block is a R block, a new FS_k is create. The newly created set is initialized to empty. The SEQ(FS)_k number is initialized to be 1 lower than the DDM Block Sequence Number in the received DDM Block. However, if the block is E or D-type, this means that the FS_k set is out of synchronization with the Direction Set on the upstream sender end. In this case, the receiving node sends a RSYNC packet back to the sending upstream neighbor and the data packet is dropped. In this RSYNC message, the "DDM Block Sequence Number" field is set to the same sequence number as the received DDM Block's. The "Forced Refreshing (FR)" flag is set to "forcing once" unless the node is operating in "statefree" mode, which will be described later in section 4.6.

Before a node proceeds further, it compares its SEQ(FS)_k value with the "DDM Block Sequence Number" in the received DDM block. The reason for this step is to detect destination set update losses. Because of the differential approach, it is very important to quickly detect any packet losses which contain destination list updates. When a node sends out a DDM Block, it stamps the block with a DDM Block Sequence Number. This sequence number is incremented by one every time a node sends out an informative DDM Block. The sending of E headers does not change the sequence number since there is no change in the destination list. Still, an E header carries the current sequence number so a receiving node may detect previous losses.

If the packet "DDM Block Sequence Number" is at least one greater than the SEQ(FS)_k and the DDM block is an E or at least two greater and the DDM block is a D, the receiving node knows that at least one packet was lost and the packet(s) contains destination list updates. So the receiving node sends a RSYNC packet back to the sending upstream neighbor, and the data packet is dropped. This check is unnecessary for R blocks since they contain the entire multicast destination list for this node to forward.

After verifying the DDM Block Sequence Number, the SEQ(FS)_k value is updated to the DDM Block sequence number in the packet. If the received data packet has an E header, the receiving node only needs to forward one copy of the data packet to each downstream neighbor l whose current DS_l is not empty.

If the data packet has a R block, the node replaces its FS_k by the list L in the header.

FS_k = L

When a node receives a Di or Dd block, it updates the corresponding FS_k according to the D header: removing the addresses in the L_d from its FS_k then adding the addresses in the L_i into its FS_k set. If the received block is a Dd block, the receiving node needs to look one block beyond to see if the next block following the Dd is a Di block for itself. If so, the L_i list in that D_i block needs to be processed together.

FS_k = (FS_k - L_d) U L_i

After updating the FS_k, the receiving node updates the union FS. Then it partitions the new overall FS set into DS'_l sets according to the "next hops" l used by the unicast routes towards the destinations in the FS. All destinations in the same DS'_l share a common "next hop" l.

By comparing the contents of the new DS' sets and the existing DS sets, the node assembles new DDM Blocks for the outgoing DDM Data Header. If there is a DS'_l but there is no matching DS_l for a particular "next hop" l, an R-type DDM Block is constructed for l. The DDM Block Sequence Number for this DDM Block is set to 0. Alternatively, if there is one DS_l set but there is no DS'_l, no DDM Block is constructed.

For the case that there are both a DS'_l and a DS_l, the processing checks if the DS_l's "Forced Refreshing" flag is set to either "forcing once" or "forcing always". If so, a R block is constructed

which contains all destination addresses in the DS'_l. Otherwise the

contents of them need to be compared. If the DS'_l set is the same as the DS_l set, an E block is used since there is no change. Otherwise, R or D block(s) is constructed. By default D block(s) are tried first. All destination addresses that are in the DS'_l (new Direction Set) but not in the DS_l (old Direction Set) form the incremental list L_i. All addresses that are in the DS_l but no longer in the DS'_l form the decremental list L_d. Addresses that are common in both sets appear in neither list. If the total length of the L_i and L_d lists is not shorter than the destination list in DS'_l, a R block is used instead to cut down number of addresses in header. Otherwise, a Dd Block is constructed then followed by a Di Block. Of course when either L_i or L_d list is empty, its corresponding DDM Block is not needed. For D or R-type DDM Blocks, the SEQ(DS)_l is incremented by one while it remains the same for E blocks. The final SEQ(DS)_l is used as the DDM Block Sequence Number in the constructed DDM Block.

R header: L = DS_l or

D header: L_i = DS_l - DS'_l
L_d = DS'_l - DS_l

After the DDM Block is ready it is packed into the header of the data packet. The multicast destination list of the DS_l set are replaced by the one of DS'_l set and kept in memory for the next forwarding computation. After the construction of the outgoing DDM Block, the "forced refreshing" flag of the DS_l set is reset to "no forcing" if the current value if "forcing once".

If DDM header aggregation is not in use, the data packet can be forwarded to the downstream neighbor using unicasting right away. Otherwise, more DDM Blocks will be packed into the same DDM Data Header until the number of DDM Blocks in the DDM Data Header exceeds MAX_DDM_BLOCKS. The above DDM Block construction is repeated until all DS' sets are processed. New DDM Blocks are inserted into the packet header if the header size does not exceed the limit. Otherwise, the filled data packet is sent out, and a new data packet with the same payload is allocated. New DDM Blocks are inserted into the header of this new data packet.

4.5 Forwarding Set Synchronization

When operating in soft-state mode, usually only the differences of the multicast destination lists are included in data packet headers. 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 a DDM Block Sequence Number to maintain

synchronization. The sequence number on the DS (upstream) side must match with the sequence number of the FS (downstream) side.

This sequence number is carried inside each DDM Block and its advertisement is data-driven. Only nodes which are not exchanging data may remain out of synchronization for extended periods. If the receiver detects missing sequence numbers, it knows that some data messages containing informative DDM Blocks for it have been lost and its Forwarding Set is out of synchronization with the Direction Set on the upstream neighbor. It needs to notify its upstream neighbor using a RSYNC message. When used for resynchronization purpose, this message contains a flag telling the upstream neighbor to set the "Forced Refreshing" flag for the Direction Set to "forcing once". The message is unicast to the upstream neighbor.

On the upstream end, upon receiving a RSYNC message, it locates the DS set used for the sender of this RSYNC message and sets the "forced refresh" flag to what the message's FR flag says. When the upstream neighbor is forwarding the next data packet, if it sees the "forced refresh" flag is set for a DS set, a R Block is constructed. Since R block contains the full list, upon reception the downstream neighbor will have its FS set synchronized again.

4.6 Timers and Dual Modes of DDM

DDM is a dual-mode algorithm where each participating node can operate in "stateless" mode or "soft-state" mode. Timers, the "Forced Refresh" flag, and the TOL field in each DDM Data Header are the mode control parameters.

There are timers associated with each set (both FS_k sets and DS_l sets). When any of the timers expires, the corresponding set is removed. Every time there is a packet received from neighbor k, the

timer associated with the FS_k is reset to expire in TOL seconds as specified in the received DDM Data Header.

FS_k timer value = TOL

Every time a DS_l set is used for forwarding, the forwarding node picks an expiration period for the timer associated with this DS_l. Then in the outgoing packet, its TOL field is set to:

TOL = DS_l life timer value * R (R > 1)

This action helps ensuring that, over a given link, the timer for the Direction Set on the upstream side expires before the timer for the Forwarding Set on the downstream side. Otherwise if the DS set lives longer than its downstream FS set, the upstream neighbor may use an E

or a D header and the downstream node will only transmit an unneeded RSYNC packet. Although the timer setting scheme does not avoid this from happening completely (packet loss may still cause the same problem), it should sufficiently reduce the probability of this undesired event. Also, by making the timer value for the DS sets a local decision, nodes are given the opportunity to adapt the timer values to their local environment. For example, if a node moves rapidly relative to its surrounding nodes, it is reasonable to choose a smaller timer value.

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. If the upstream neighbor is also operating in "stateless" mode, there will be no need to do so since all DDM Blocks coming from it are R-type. If the upstream neighbor is in "soft-state" mode, when receiving an E or D header the downstream "stateless" node needs to send back a RSYNC message with the "force refreshing" flag set to "forcing always". After receiving such RSYNC message the upstream neighbor only sends R-Blocks from the next data packet onwards since the DS block corresponding to this "stateless" downstream neighbor has a "forcing always" flag. In a "stateless" node, all FS and DS set timers are set to zero so that none of the sets are kept. Later on, if a "stateless" node ever wants to switch back to "soft-state" again, it needs to send another RSYNC message to its upstream neighbor to cause the latter to reset its "force refreshing" flag to "no forcing".

Using a combination of RSYNC message, TOL, and "force refreshing" flag, neighboring nodes can operate in different modes but still work together.

4.6 Multicast Data Encapsulation

During the partition of the overall Forwarding Set, DDM needs to query the unicast routing protocol for the next hop information. In the case when there is no route, or the underlying unicast protocol is on-demand, such next hop information may not be available immediately. In this case DDM will encapsulate the multicast data packet in a CTRL_DATA control message. This message is a unicast message from this forwarding node to the multicast destination to whom the next hop is unknown. This message is passed to the unicast routing protocol for forwarding. During the forwarding of this unicast packet, the underlying unicast routing protocol will construct the route. This way, soon the next hop information will become available for DDM. In the case such multicast destination is unreachable, the data packet will eventually be dropped by the unicast routing protocol.

When a node receives a valid CTRL_DATA message, it decapsulates the original multicast data packet. Then it uses the information stored in CTRL_DATA header to restore the D-class IP header for the data packet. Finally, the restored multicast data packet is loop-back to the IP kernel of the same node for local delivery.

5 Conclusion

This draft proposes a multicast routing protocol intended for use with small multicast groups in 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 simple, efficient, and robust protocol suitable for small multicast groups. The protocol offers the option of choosing between "stateless" and "soft-state" modes.

Simulation shows that DDM is efficient both in terms of bandwidth and channel access. It is nearly ideal for multicasting to small groups

in networks that already have unicast routing support. Also DDM is highly robust and flexible.

Ongoing work on DDM is focusing in 2 areas: (a) more simulation studies to explore different modes and options of the DDM and (b) a Linux multicasting daemon implementation to study DDM's performance in real system.

6. ACKNOWLEDGMENTS

This work was performed under a grant from the Army Research Lab (ARL) ATIRP program.

References

[1] J. J. Garcia-Luna-Aceves and E. Madruga, "A Multicast Routing Protocol for Ad-Hoc Networks", Proceedings of IEEE INFOCOM'99, March, 1999.

[2] S. Lee, W. Su and M. Gerla, "On-Demand Multicast Routing Protocol (ODMRP)", Internet Draft draft-ietf-manet-odmrp-02.txt, work in progress, January, 2000.

[3] E. Royer and C. Perkins, "Multicast using Ad hoc On-Demand Distance Vector Routing", Proceedings of ACM/IEEE MOBICOM '99, 1999.

[4] L. Ji and M. Corson, "Light-weight Adaptive Multicast", In Proceedings of IEEE GLOBECOM '98, November, 1998.

[5] J. Moy, "Multicast Extensions to OSPF", RFC1584, March, 1994.

[6] A. Ballardie, "Core Based Trees (CBT version 2) Multicast Routing -- Protocol Specification", RFC2189, September, 1997.

[7] D. Estrin et al, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", RFC2362, June, 1998.

[8] R. Boivie, "A New Multicast Scheme for Small Groups", IBM Research Report RC21512(97046), June, 1999.

[9] D. Ooms and W. Livens, "Connectionless Multicast", Internet Draft draft-ooms-cl-multicast-01.txt, October, 1999.

[10] D. Johnson and D. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", Mobile Computing, 1996.

Author's Addresses

Lusheng Ji, M. Scott Corson
Institute for Systems Research
University of Maryland
College Park, MD 20742
(301) 405-6630
{lji,corson}@isr.umd.edu