Improving Delivery Ratios for Application Layer Multicast in Mobile Ad-hoc Networks

Peter Baumung
Institute of Telematics
University of Karlsruhe (TH)
Email: baumung@tm.uka.de

Martina Zitterbart
Institute of Telematics
University of Karlsruhe (TH)
Email: zit@tm.uka.de

Kendy Kutzner
Institute of Telematics
University of Karlsruhe (TH)
Email: kutzner@tm.uka.de

***Abstract*— Delivering multicast data using so-called overlay networks offers many advantages. Overlay networks only consisting of a multicast group's members that are connected by means of transport links, non-members (especially routers) no longer need to bother about multicast routing and keeping a group's state information. While existing application layer multicast protocols mainly were designed for the fixed Internet, first attempts to adapt such protocols for mobile ad-hoc networks start emerging. Because these adaptions however still suffer from a relatively high rate of packet losses, this paper outlines generic mechanisms that improve the overall delivery ratio for application layer multicast protocols. Although the proposed mechanisms are illustrated at the example of the NICE-MAN protocol (which is an adaption of the NICE protocol, originating from the Internet), they may be applied to arbitrary overlays. Furthermore, this paper shows how overlay networks combined with the developed mechanisms can be used, to significantly improve delivery ratios for existing multicast protocols (like ODMRP, M-AODV) that reside on the network layer.**

I. Introduction

The ability of delivering data not to only a single user, but to an entire group of receivers, becomes more and more significant as today's network bandwidth allow realtime streaming of audio and video data. Application layer multicast appears to be a promising approach of delivering multicast data using so-called overlay networks. Such networks are built on top of physical links. They consist of a multicast group's members that are connected through unicast transport connections. This means that non-members need to bother neither about a multicast group's state information, nor about any multicast routing. As data is forwarded between a group's members using existing unicast protocols, no changes to a network's infrastructure are required. In the context of the fixed Internet all of this results in an important relief especially for routers.

Although applications may differ, demand for multicast data delivery in mobile ad-hoc networks keeps growing. Streaming of whiteboard applications may be supported as well as multi player games and cooperation environments for spontaneously formed learning groups. Many application layer multicast protocols having been developed for the Internet [1], [2], [3], people look forward to operate these protocols in mobile ad-hoc networks. Although some of them seem promising, none of these protocols take the pecularities of mobile ad-hoc networks fully into account. Most of them neither sufficiently consider increased node mobility nor the broadcast medium.

First attempts in generically adapting application layer multicast protocols for the Internet have appeared [4]. Although these adaptions considerably improve a protocols performance in mobile ad-hoc networks, average delivery ratios still remain too low to admit streaming applications with low bandwidth or similar applications requiring almost reliable data transmission.

This paper's focus is on defining generic mechanisms for improving overall delivery ratios for application layer multicast protocols in mobile ad-hoc networks. These mechanisms comprise selective retransmissions requests as well as a buffer management and a simple congestion control. Furthermore this paper shows how application layer multicast protocols can be used to improve delivery ratios for existing multicast routing protocols.

As the mechanisms presented in this paper require a basic understanding on the operation of overlay networks, section II shortly presents the existing NICE protocol, which was designed for the fixed Internet. In section III, the NICE protocol is enhanced according to [4] (resulting in the NICE-MAN protocol) in order to effectively make use of the broadcast medium present in mobile ad-hoc networks. The work done in this paper is presented in section IV, which aims in further enhancing the NICE-MAN protocol by means of generic mechanisms for improving delivery ratios. An evaluation of these mechanisms is provided in section V. Section VI shows how overlay networks can be used to improve data delivery for existing multicast routing protocols. An evaluation of ODMRP enhanced by the NICE-MAN overlay and the mechanisms presented in this paper is given in section VI-A. Sections VII and VIII conclude by briefly referencing related projects and providing an overview of future work.

II. The NICE Protocol

The NICE protocol [1] is a scalable hierarchical application layer multicast protocol, that organises its group members using a cluster-based approach. Such clusters consist of nearby nodes, whose distances are measured by means of *roundtrip times (RTT)*. At all time of its group membership, each node has joined a cluster on one or more layers L_i ($i \geq 0$). The formation of these layers is described in the following.

According to NICE, clusters count at least k, but no more than $3k-1$ nodes, where k is a fixed constant. As the number of a cluster's members falls below k because of leaving nodes, it is merged with another nearby cluster. In case of joining nodes, a cluster is split into two clusters of equal size as soon as it counts more than $3k-1$ nodes. All members of a group have joined a cluster on layer L_0. Among the nodes of a cluster on layer i a leader in its center is chosen, which joins layer $i+1$. All nodes on layer $i+1$ find themselves also clustered. This recursive definition leads to a hierarchical structure as it is depicted in figure 1. Note that each member of a cluster on layer i (e.g. node A on layer 2 in figure 1) is leader of a cluster on all layers 0, ..., $i-1$.

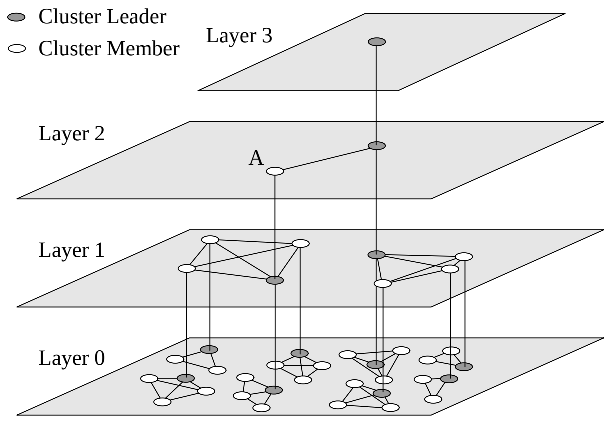

Fig. 1. NICE's hierarchical cluster approach.

Clusters are fully meshed structures, and, thus, a node knows about all the members of its clusters on all layers it has joined. Additionally any node n which is member (not leader) of a cluster on layer i knows about the members of its so-called *super cluster*, which is the cluster on layer $i+1$ to which n's leader on layer i belongs. Cluster memberships are kept as *soft states* and are periodically refreshed using dedicated *HeartBeat* messages.

NICE's hierarchy results in an overlay network as depicted in figure 2. The fully meshed clusters on different layers of the overlay are clearly visible. An important criteria affecting the scalability of an overlay protocol is given by the overhead that is needed for maintaining the overlay. In the case of NICE, the periodic *HeartBeat* messages contribute to this overhead, since they are used to communicate cluster memberships. Nevertheless NICE scales well, as it features a constant overhead for an average node and a logarithmic overhead in the worst case.

Because of the hierarchy's special structure, data delivery trees for all group members may be implicitly defined as follows. A node n receiving packets from another node n' relays the packets to all nodes inside all its joined clusters[1], excepted the cluster in which n and n' are neighbours. Obviously n will have to relay packets to all nodes inside *all* its joined clusters, if n is a multicast source.

[1]These do not include n's super cluster.

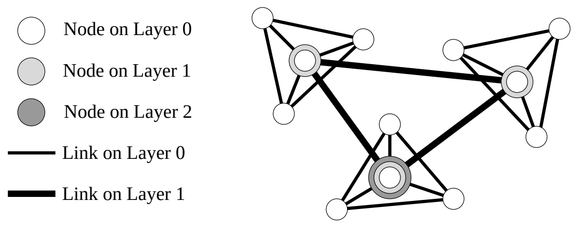

Fig. 2. An example overlay depicting the NICE hierarchy.

As the reliability mechanisms described in section IV are not specifically conceived to be used with NICE, the protocol is not presented in detail. Please refer to [1] for further information.

III. NICE-MAN: Enhancing NICE for Mobile Ad-hoc Networks

NICE appears to be a promising protocol, that can be used to efficiently realize multicast data delivery over the Internet. Its fully meshed clusters and the use of super clusters guarantee a good degree of flexibility and a high robustness when opposed to node failures. NICE's original specification features mechanisms considering a group's dynamics (nodes joining and leaving the group) as well as recoveries e.g. of a cluster leader's failure. The protocol's direct application in mobile ad-hoc networks however cannot be recommended, as its operation conditions will severely change. Because NICE considers neither the roaming of mobile devices nor the wireless medium, adaptions to the protocol are needed in order to meet both of these requirements. With NICE-MAN some interesting approaches to take the pecularities of mobile ad-hoc networks into account have been presented in [4]. This protocol fully adapts the NICE protocol for mobile ad-hoc networks. As section IV pays dedicated attention to so-called *local broadcast clusters (LBCs)* introduced with NICE-MAN, their concept is outlined in the following.

Imagine a scenario as depicted by figure 3, in which a NICE cluster with its leader and members is represented. Note that to the sake of clarity the cluster's shown overlay structure is not fully meshed. A packet of a multicast source reaches the leader through an overlay link A. As a multicast transmission is desired, the leader takes care of relaying the received packet inside its cluster. Although 4 out of 6 of the cluster's members are within the leader's direct reach, it explicitly relays the packet to each member by sending unicast packets through the appropriate overlay links (B, C, D, E, F, G). This results in an increased network load, as the leader needs to access the medium 6 times.

In order to improve scenarios as denoted above, local broadcast clusters have been designed. They effectively make use of the broadcast medium, by letting an overlay node relay data packets to an arbitrary number of group members, that are located within its transmission range, using one single broadcast message. As the overlay node no longer needs to separately relay a packet to each receiver inside its transmission range, network load is drastically reduced. Although local

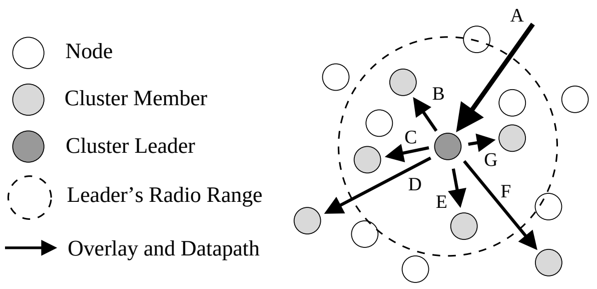

Fig. 3. Ineffective use of the wireless medium.

broadcast clusters were introduced in the context of the NICE-MAN protocol, they may be used in other overlay networks as well. In order to fully understand the idea behind local broadcast clusters, their formation is now explained.

Let a node n_O being member of the NICE overlay periodically send so-called *Local HeartBeat* messages. By doing so, n_O becomes leader of its local broadcast cluster. Let n_O furthermore repeat all received multicast packets using a broadcast message, in addition to relaying them along the overlay's links according to the overlay's routing protocol. A group member n_L, that finds itself within n_O's transmission range, will notice the overlay node's presence through its local heartbeat messages. Because n_L will receive all of the group's multicast data via n_O's broadcast messages, there is no explicit need for n_L to join the overlay. Instead, n_L can loosely float within n_O's local broadcast cluster and, thereby, become a locally joined group member. Such locally joined members do not introduce any control flow overhead, as they do not exchange state information with members of the overlay.

The mechanism of data forwarding inside the NICE overlay enhanced by means of local broadcast clusters is shown in figure 4. Multicast packets still reach the leader of a (NICE) cluster through link *A*. After having forwarded a packet to the members of its (NICE) cluster using the overlay links *B* and *C*, the leader now repeats the packet using a broadcast message *D*. By doing so, it relays the packet to all members of its local broadcast cluster.

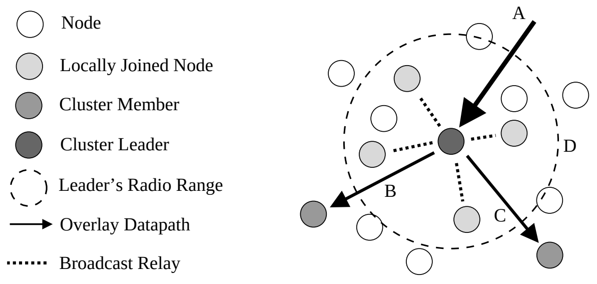

Fig. 4. Data forwarding using broadcast clusters.

A sample scenario showing an entire overlay that features local broadcast clusters can be seen in figure 5. While darker nodes are connected by the overlay, the lighter nodes loosely roam inside the respective broadcast clusters and obtain multicast data through broadcast messages. The overlay itself is now comparable to a multicast backbone which is used for relaying data over larger distances.

Local broadcast clusters can overlap, as a locally joined node may find itself within reach of two overlay nodes. An example of overlapping broadcast clusters is shown in figure 5. The nodes *A* and *B* are located within the transmission range of two members of the overlay and thus receive local heartbeats from two leaders of broadcast clusters. Because a node may only be member of one broadcast cluster, *A* and *B* decide to which cluster they belong. Such decisions can be based on measurements of the loss rate of a leader's local hearbeat messages.

As will be discussed below, locally joined nodes may experience the loss of their broadcast cluster's leader. In order to quickly recover from such events, the nodes *A* and *B* keep the second of their overlapping clusters in mind. Overlapping LBCs may have positive effects, as e.g. *A* and *B* will receive multiple instances of data packets and thus suffer from fewer packet losses.

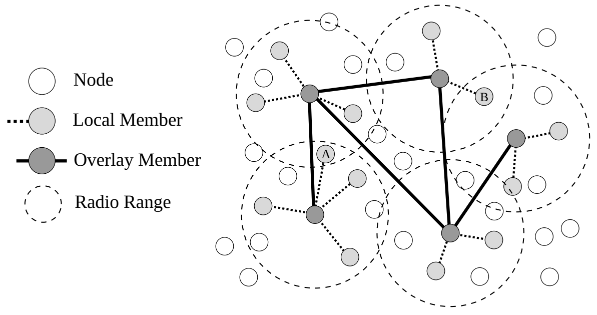

Fig. 5. Overlay featuring local broadcast clusters.

Local broadcast clusters may also collide. Such a collision occurs, when two members of the overlay are in each other's reach and thus receive each other's local hearbeat messages. Collisions between LBCs are explicitly unwanted, as the wireless medium thereby becomes ineffectively used. One of both colliding leaders therefore abandons its LBC and thereby leaves the overlay. While doing so, the concerned leader broadcasts a message, informing its locally joined members about the cluster's dissolution. These now find themselves in one of two possible states: either they were located inside an overlapping of the former LBC and another nearby LBC, or the dissolved LBC was the only cluster they knew about. Members still residing inside a LBC and having kept the leader's presence in mind, the first case needs no further attention. The second one however must be taken care of, as such members are no longer within an overlay node's reach and thus will not receive multicast data any longer. Similar scenarios may also occur when nodes move out of a LBC leader's reach because of their mobility. In order to obtain further multicast packets, such a node now needs to join the overlay and, by doing so, become leader of a LBC itself. To enable nodes to quickly join the overlay in such cases, local heartbeat messages include information about the overlay. In case of the NICE-MAN protocol this is the address of a NICE cluster's leader. Nodes having lost their LBC leader can thus quickly join the NICE-MAN overlay by directly contacting the leader denoted by the last heartbeat received.

Enhancing overlay networks using local broadcast clusters most often results in much smaller overlays, as fewer nodes need to join the overlay. Thus, the overall control overhead needed for the overlay's maintenance can drastically be reduced. More details about broadcast clusters and simulation results underlining their positive effects on overlay networks can be found in [4].

IV. Improving Delivery Ratios

As commonly known, the pecularities of a mobile ad-hoc network's wireless medium result in a particularly high rate of packet losses. Two nodes within each other's reach and simultaneously accessing the medium most often cause collisions between their packets. As a consequence such packets are dropped and therefore do not reach their destination. In order to improve the average packet delivery ratio for multicast data deliveries through NICE-MAN, some mechanisms for recovering from such packet losses will now be introduced. In general the presented principles can be applied to arbitrary application layer multicast protocol. Furthermore dedicated support is supplied for protocols, that organise their group members using local broadcast clusters whose leaders are connected through a so-called multicast backbone.

This section is structured as follows. First the basic receiver-oriented mechanism for recovering from packet losses using retransmission requests is presented. Then possibilities that may be used to reduce the amount of unnecessarily sent retransmissions requests are outlined. Thereafter, a suited buffer management is introduced. Finally, congestion control is addressed.

A. Selective Retransmission Requests

Basically packet losses are identified through gaps in the sequence numbers of received packets. Recovery from such packet losses is achieved, by letting an affected node *A* request retransmissions for each of the corresponding packets. The entire mechanism thus show a receiver-oriented character.

As can be seen in figure 6, data paths may be specific to each possible multicast source. In order to efficiently recover from packet losses, *A* should request retransmission from the node, that is superior on a specific source's data path. According to figure 6 *A* will thus request retransmissions of packets sent by *S*1 from *B*, while requests for *S*2 will be sent to *C*. Group members in general are therefore required to remember the address of the node, that data packets from a specific source are currently received from. This however is not of interest for group members having joined a local broadcast cluster, as these nodes can directly request missing packets from their broadcast cluster's leader.

For each missing packet a configurable number of retransmission requests is sent within a certain time interval if the previously sent requests remain unanswered. As the

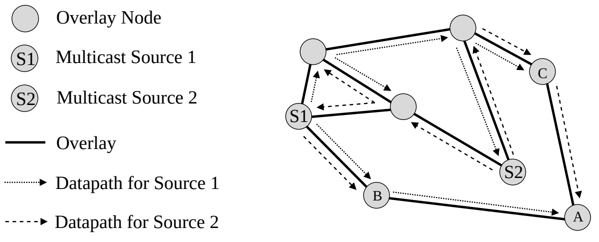

Fig. 6. Source-specific data paths.

sequence number's gap persists after the final retransmission was requested, the packet loss is declared as unrecoverable. Such events most likely result in broken connections and are to be handled by the multicast application itself.

Multicast packets themselves are retransmitted depending on the state of a node a retransmission request was received from. While requests originating from nodes being part of the overlay are answered directly by sending unicast messages along the corresponding overlay link, retransmissions for locally joined group members are broadcasted. This allows multiple locally joined nodes to recover from the same packet loss using only one retransmission.

Because of the mechanism's receiver-oriented character a dedicated flag inside each multicast packet is required to denote a transmission's final packet. A group member that neither received further multicast packets for a while, nor was informed about the transmission's end using the flag introduced above, automatically requests a missing packet from its superior node. As the node received packets up to a certain highest sequence number s_H, the sequence number of the requested packet is given by $s_H + 1$.

B. Avoiding Unnecessary Retransmission Requests

As will be shown below, many of the retransmission requests sent are redundant or inadequate. The reliability mechanism presented so far would thus most often result in superfluous accesses to the wireless medium. Extending the above mechanism by means of NACK avoidances can considerably reduce the amount of requests sent. At this point, it is important to distinguish both possible states of a node willing to send retransmission requests. While the focus will first be on nodes that locally joined a broadcast cluster, the sending of retransmission requests by members of the overlay will be discussed afterwards.

1) Locally joined Nodes: As depicted by figure 7.a), the loss of a specific packet is most probably experienced by several nodes being member of the same broadcast cluster. While the shown overlay node *A* relays a multicast packet via broadcast, a foreign node *B* simultaneously accesses the medium. Because *A*'s broadcast message is not covered by retransmission requests on the MAC layer, *B* by its access to the medium causes a collision for the marked nodes. This usually requires each of the affected nodes to individually request retransmissions from *A*. The resulting multiple retransmissions through *A* show up in an increased energy consumption and in an ineffective use of the wireless medium.

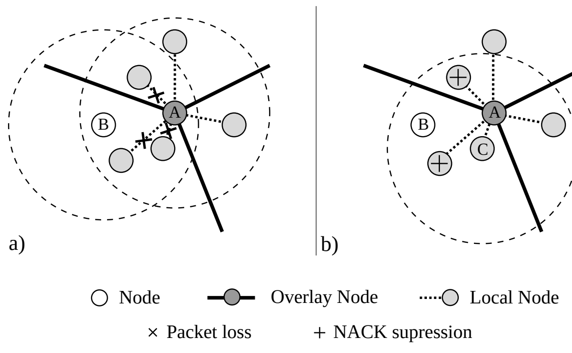

Fig. 7. Packet losses and NACK suppression within broadcast clusters.

Therefore, locally joined group members delay their retransmission requests for a randomly chosen time interval. Furthermore, requests are sent as broadcast messages, instead of directly addressing the cluster's leader using an unicast operation. This way any other locally joined group members, that have experienced the loss of the same packet and that are located within the member's transmission range, are informed about the ongoing retransmission and thereby suppress their own request. As shown in figure 7.b), *C*'s timer elapses first, alowing the node to immediately send its retransmission request. The request is intercepted by the marked nodes, that thereby suppress their own requests.

However transmission ranges will probably not be equal for all members. While it is guaranteed that a broadcast cluster's leader reaches all of its local members within one hop, locally joined members might not be able to reach their leader directly due to an inferior transmission range. Such scenarios prevent locally joined members to request retransmissions from their leaders using broadcast messages. This leads to an alternate design of the above mechanism, in which locally joined nodes directly address their leaders using unicast packets. In order to intercept retransmission requests and therefore enable local NACK avoidance, this would require all locally joined members that experienced packet losses to be in promiscuous mode.

2) Nodes having joined the Multicast Backbone: The necessity of providing a mechanism avoiding superfluous retransmission requests in a more global fashion inside the multicast backbone is shown in figure 8. In this scenario it is assumed, that a multicast source *A* already sent a total of 455 packets that were successfully received by the depicted receivers. While packet 456 is dropped on its way from node *A* to node *B* because of collisions, packet 457 successfully reaches its destination. By receiving that packet, node *B* recognises packet 456 as missing and requests its retransmission from *A*. To reduce overall latency node *B* however immediately relays packet 457 to the nodes *C* and *D*. Being successfully received, both nodes now also identify packet 456 as missing and request its retransmission through *B*. Because *B*'s request to *A* yet remained unanswered, the node at this point is unable to fulfill *C*'s and *D*'s requests.

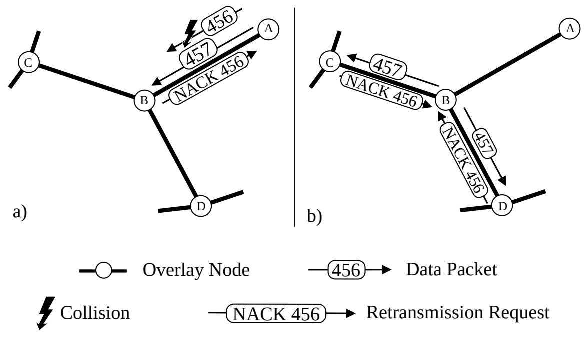

Fig. 8. Inappropriateness of C's and D's retransmission requests.

Such unnecessary accesses to the medium can easily be prevented by letting each member *n* of the multicast backbone inform its subordinated nodes about its highest available packet. This is the last packet a node received, before any packet losses occured. A simple way to realise this, is by adding a second field to each relayed multicast packet, containing the highest available packet's sequence number.

This results into an avoidance mechanism as it is shown in figure 9. While relaying packet 457 to the nodes *C* and *D* in figure 9.b), node *B* informs the recipients about its highest available packet having a sequence number of 455. By receiving that packet, both nodes recognize packet 456 as missing. However they also realize that it is not yet available from node *B* and thus suppress their retransmission requests. As *B*'s retransmission request 456 to *A* is eventually answered in figure 9.c), the node on one hand has successfully recovered from its packet loss. On the other hand the node is now able to increase the sequence number of its highest available packet from 455 to 457. In order to reduce overall latency, node

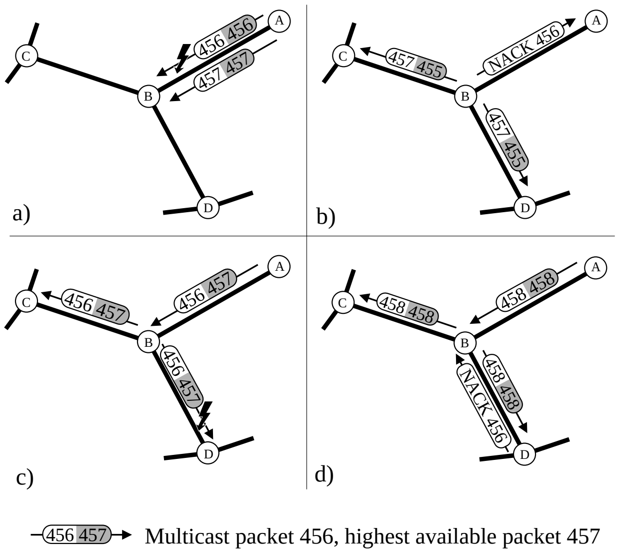

Fig. 9. Global NACK avoidance inside the Multicast Backbone.

B immediately relays packet 456 to C and D[2] and thereby informs both nodes about its new highest available packet. Because of another collision however, packet 456 is dropped on its way to node D. In figure 9.d) A relays packet 458 to node B. As the packet is successfully received, it is immediately forwarded to node C and D. Still missing packet 456, node D is now able to request its retransmission through B. This results from the fact, that D was informed by packet 458 about B's highest available packet now having a sequence number of 458.

C. Buffer Management

Because of its receiver-oriented character, the reliability mechanisms presented so far suffer from the well known problem of infinite buffers. Nodes relaying multicast packets are uncertain up to which sequence number packets were successfully received by their subordinated nodes. Retransmission requests concerning specific packets being possibly received at any time, nodes thus are not allowed to free their buffers if a reliable transmission is desired. To overcome this problem, a buffer management based on an error correction and a receive window is introduced.

Every packet that a node receives over time is inserted in the node's receive window, which has a capacity of c_R packets for all nodes of the multicast group. As packet losses occur during the transmission, a node's receive window may experience gaps in its sequence numbers.

Because of its limited capacity the window's lower bound has to progress with time. This is achieved by introducing so-called cumulative ACK packets. Let s_H^n denote the sequence number of the highest packet available from a node n and $\mathcal{S}^n$ the set of the overlay's members that are subordinated to node n. In order to inform n about successfully received packets, each node $n' \in \mathcal{S}^n$ periodically sends cumulative ACKs directly to n. Cumulative ACKs sent by the node n contain a sequence number

$$s_C^n \quad := \quad \min\left(\{s_H^n\} \cup \bigcup_{n' \in \mathcal{S}^n} \{s_H^{n'}\} \right)$$

where $s_H^{n'}$ is denoting the sequence numbers that n itself directly received from its subordinated nodes by means of cumulative ACKs. As shown in figure 10 the global minimum of all highest available packets this way propagates towards the multicast source. Members of the overlay thereby gain knowledge up to which sequence number their receive windows can safely be freed.

This however is only true for members having joined the multicast backbone. Because of the nature of broadcast clusters, locally joined nodes cannot reliably inform their cluster's leader about successfully received packets[3] and thereby ommit

[2]By doing so, C and D do not need to explicitly request the packet's retransmission and thus latency is reduced.

[3]This results from the fact, that locally joined nodes are unknown to the broadcast cluster's leader. Thus the leader is not able to accumulate sequence numbers from his locally joined subordinated nodes.

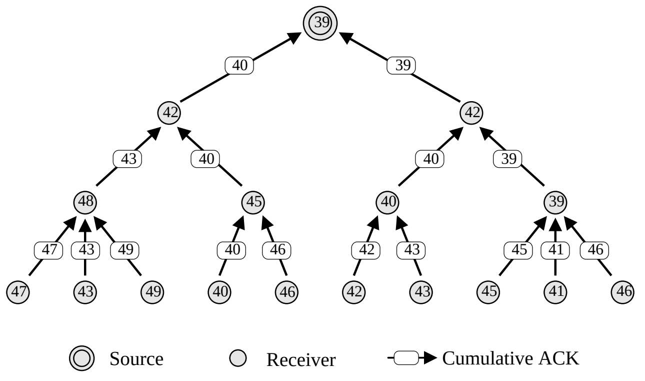

Fig. 10. Propagation of ACKs backwards along data paths.

sending any cumulative ACKs. This raises the problem that a member n of the overlay might free its receive window up to sequence number s_C^n, altough some of its locally joined members still lack packets with lower sequence numbers.

Instead of directly discarding packets as the receive window progresses, a member of the overlay shifts affected packets from the receive into the error correction window. The latter window has a capacity of c_E packets and thereby supplies a certain buffer in order to allow locally joined nodes to recover from packet losses. The progress of both windows over time is shown in figure 11 for a node A.

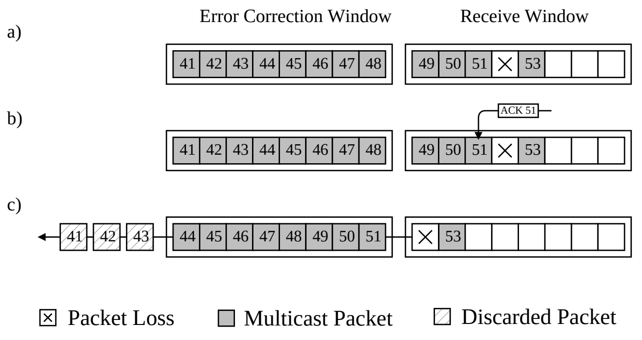

Fig. 11. Progress of the receive and error correction window.

In figure 11.a s_C^A apparently has a value of 48. Because A however successfully received all packets up to sequence number 51, this indicates that one of A's subordinated nodes B is responsible for the receive window not being able to progress. As depicted in figure 11.b the node B eventually informs A about having received all packets up to sequence number 51 and thus increases s_C^A by 3 to a value of 51 (figure 11.c). Note that the packets removed from the receive window now take place in A's error correction window. Because of this window's capacity being exhausted, some of its oldest packets need to be discarded.

This double-buffered mechanism is based on the assumption, that locally joined members quickly recover from packet losses. This assumption is supported by the fact, that recovery for locally joined members will be achieved over very low distances. With an increasing value for c_E these nodes will be able to recover from the loss of a specific packet for a longer time. However an absolute reliability can obviously not be guaranteed.

D. Congestion Control

The buffer management presented above can easily be extended into a simple congestion control. This can be achieved by explicitly making use of the receive window's limited size.

Let a multicast source S insert its sent packets into its receive window. According to section IV-C, the window's lower bound will progress as the source receives cumulative ACKs from the subordinated group members. In case of congestions (mainly being noticeable through extremely high rates of packet losses), these cumulative ACKs will either hold only slowly increasing sequence numbers (due to many losses of multicast packets), or not reach the multicast source at all. In both cases this results into a nearly unchanging value of s_C^S. Depending on the rate S wants to deliver data at, its receive window's capacity will rapidly be exhausted. Being unable to insert further packets into its receive window, the multicast source has to delay its data delivery and thus the network load will be reduced. As group members recover from packet losses, cumulative ACKs with increasing sequence numbers will propagate backwards along the data paths and eventually reach the multicast source. Its receive window slowly progressing again, the source will then be able to resume its data delivery[4].

Under certain circumstances however, this mechanism might result into a deadlock situation, in which a multicast source perpetually delays its data delivery. A source S is able to indentify such situations, as soon as s_C^S stays unchanged for a certain period of time. This problem can be solved, by letting the source repeat the packet causing the deadlock. This packet's sequence number is obviously given by $s_C^S + 1$.

E. Fast ACK Relay

Depending on the periodicity cumulative ACKs are sent by members of the overlay, the propagation of sequence numbers towards the multicast source might be heavily delayed. This is explicitly unwanted, as in many cases the source will have to wait for incoming sequence numbers, in order to let its receive window progress and thereby resume its data delivery. Increasing the rate cumulative ACKs are sent at would reduce the propagation latency on one hand, but also result in a much higher network load on the other hand. For this reason an adaptive mechanism called *Fast ACK Relay* is used. A node A sends an ACK packet additionally to all periodically sent ACK packets, if an important change of s_C^A is detected after having received a cumulative ACK from a subordinated node. Such a change is considered to be important, if s_C^A is increased by at least a certain fraction $\alpha \in [0,1]$ of the receive window's capacity c_R. If the value for α is cleverly chosen, this mechanism most often results in a quick propagation of important information towards the multicast source as well as in a low average network load.

[4]From the sources point of view, the receive window therefore represents the largest buffer that can be transmitted without needing any acknowledgements from a group's members. This is commonly known as a sliding window technique.

V. Results of Simulation Experiments

In order to evaluate the mechanisms outlined in section IV, they were applied to the NICE-MAN protocol [4] outlined in section III. For implementation and measurements the Qualnet simulation tool was used [5].

Simulations are lasting for 10 minutes. Every scenario uses an area of $1000m$ by $1000m$ in size. An area features 80 nodes with a transmission range of $150m$, in order to provide a good degree of connectivity. Each node's link bandwidth is set to 2 Mbit/s. The multicast group consists of 30 members (1 sender and 29 receivers) that move at a speed between 0m/s and 1m/s according to RPGM[5] with an average cluster size of 3 nodes and a radius of $80m$. Paths for the remaining nodes are calculated using the random waypoint model. These nodes move at a speed which is randomly chosen between 0m/s and 2m/s. A combination of RPGM and the random waypoint model is used, in order to simulate the behaviour of small cooperating learning groups consisting of pedestrians that slowly move among many nodes not participating in the group's activities. Any communication between nodes in the experiments is achieved by transmitting packets using the unreliable UDP [7]. The routing of unicast packets on the network layer is done using AODV [8]. All measurements are smoothed by averaging results of 20 scenarios simulated using different values for random seeds each.

In order to evaluate the effects of the mechanisms introduced in section IV, some measurements for the original implementation of NICE-MAN (without any of the mechanisms outlined in section IV) are provided as well. In this context packets of 512 bytes each were sent for 5 minutes. While a rate of 2 packets per second was delivered under the original implementation of NICE-MAN, the rate was doubled (4 packets per second) for NICE-MAN's enhanced version. Note that two classes of simulations experiments for NICE-MAN's enhanced implementation were conducted. While *Fast ACK Relay (FAR)* was enabled during one experiment, it has explicitly been disabled during the other experiment. The capacities for the receive and the error correction window were set to $c_R = 16$ packets and $c_E = 32$ packets respectively.

Most of this document's figures show so-called *node orders*. These types of representation give a clear overview of simulation results by averaging measurements separately for each node and ordering the outcomes from the best node (on the left) to the worst (on the right).

As can been seen in figure 12 delivery ratios[6] obtained using NICE-MAN's enhanced version out-perform the original implementation by far at any time, regardless of *Fast ACK Relay* being enabled or disabled. In fact 20 out of 29 receivers

[5]The Reference Point Group Mobility model organises nodes using clusters of different size. It is fully described in [6].

[6]The delivery ratios denotes the percentage of packets sent by a multicast source, that have successfully been received by the set of receivers.

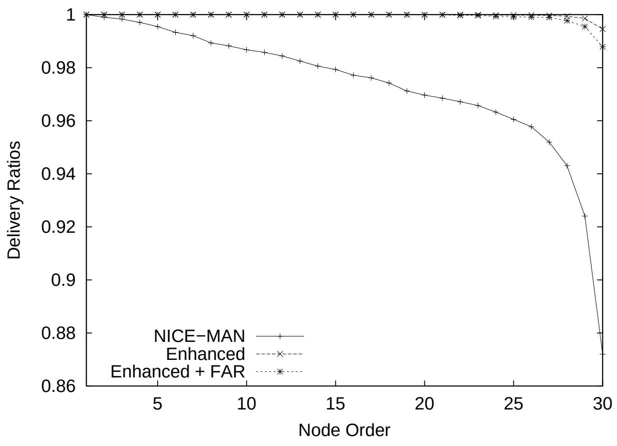

Fig. 12. Delivery Ratios under NICE-MAN.

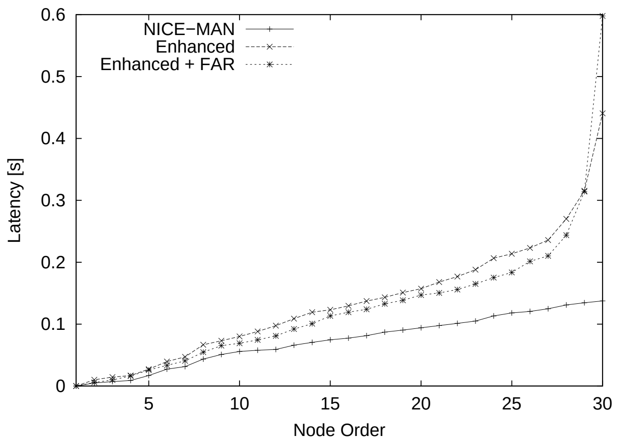

Fig. 13. End-To-End Latencies under NICE-MAN.

successfully recover from all occuring packet losses and thereby experience a reliable transmission.

The remaining receivers suffer from only very few packet losses ($< 0.3\%$), excepted the worst two nodes, for which delivery ratios of only about 99% are achieved. These unrecoverable packet losses originate from the error correction window's moderate capacity and the data delivery mechanism which is tightly bound to the overlay network. In case of major changes inside NICE-MAN's overlay, nodes may be unreachable for some time and therefore not able to receive any data packets. After having rejoined the overlay the provided buffers may then be too small to allow a full recovery.

Note that enabling *Fast ACK Relay* slightly worsens delivery ratios. As each node's windows are now able to progress more quickly, packets will sooner be discarded from error correction windows. Especially locally joined nodes will have to recover from packet losses within less time.

The end-to-end latencies, as they were obtainend during the expirements above, are illustrated in figure 13. Latencies for NICE-MAN's original implementation are the lowest, as no attention is paid to lost packets. Recovery from packet losses increases latencies, as retransmission request can only be sent after certain delays. This especially affects locally joined nodes, as back offs are needed to realise the local NACK avoidance. Similar to figure 12 the worst two nodes experience relatively high latencies. These result from delayed recoveries from packet losses, as a node might have been unreachable for some time, during important changes inside NICE-MAN's overlay.

A. Fast ACK Relay

According to the congestion control designed in section IV-D, a multicast source stops sending further data packets as soon as its receive window doesn't progress quickly enough. This raises the question of how long a source actually has to delay its data delivery because of missing acknowledgements.

Figure 14 gives an overview of the average duration, a multicast source assumes a congestion for, while delivering data using NICE-MAN's enhanced implementation. A data transmission starting at 180 seconds simulation time, positive congestion times may only occur beyond that point. The figure is e.g. to be read as follows: 2 minutes after a transmission with *Fast ACK Relay* disabled was started (at 300 seconds simulation time), a multicast source in average will have assumed a congestion for about 55 seconds and thus will have spent this time not sending any data packets. As can be seen, enabling *Fast ACK Relay* drastically reduces a source's idle times. A multicast source being able to resume its data delivery more quickly, an applications average throughput will raise.

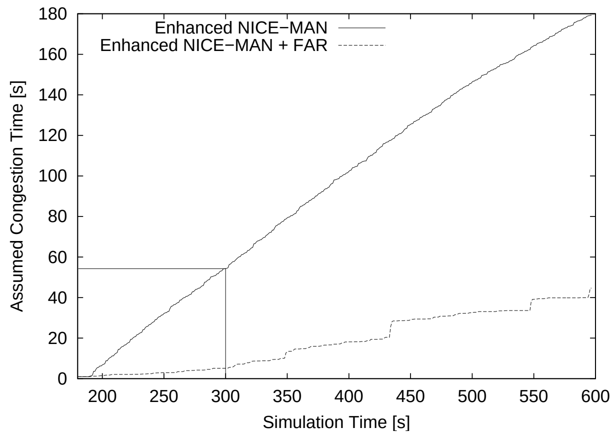

Fig. 14. Assumed congestion durations for NICE-MAN.

Note that the congestion times remaining for *Fast ACK Relay* mainly result from temporarily unreachable nodes, in course of drastic changes inside NICE-MAN's overlay. Because such a node is not member of the overlay for some time, its lastly sent cumulative ACK[7] will no longer affect the source. The latter one will thus continue its transmission, not noticing the nodes absence. As the node eventually rejoins the overlay, it has fallen back in the transmission. Its retarded

[7]This refers to the last cumulative ACK the node sent, while being a member of the overlay.

cumulative ACK packets will thus cause the multicast source to freeze until the node recovered from its packet losses.

VI. Utilizing Overlay Networks for Multicast Routing Protocols

Overlay networks can be used to improve delivery ratios for existing unreliable multicast routing protocols such as ODMRP [9] or M-AODV [8]. Such overlays are built on top of existing multicast routing protocols without neither affecting their functionality, nor requiring any changes to these protocols. While data packets are relayed according to the multicast routing protocol, the overlay's only purpose is to allow group members to recover from packet losses.

As nodes join an overlay, they gain knowledge about nearby group members. In case of a packet loss, these nodes can be used for recovery by requesting the respective packets to be retransmitted. When doing so, it is crucial not to request retransmissions from arbitrary neighbours on the overlay. Instead retransmissions should be requested from nodes that lie nearer to the multicast source. Because the overlay however is not used for relaying the multicast data itself, group members do not know which node on the overlay packets would have been received from, and thus which node lies into the source's direction. For this reason a multicast source is required to explicitly build data paths as they would be created during transmissions using the overlay. This can easily be achieved by letting the source periodically send dedicated packets along the overlay. Periodic refreshments are required, as the overlay's topology most probably will evolve with time. It can quickly be deduced that data paths along the overlay will most often not match the (network layer) multicast protocol's paths. This is shown in figure 15, in which NICE has been used as overlay.

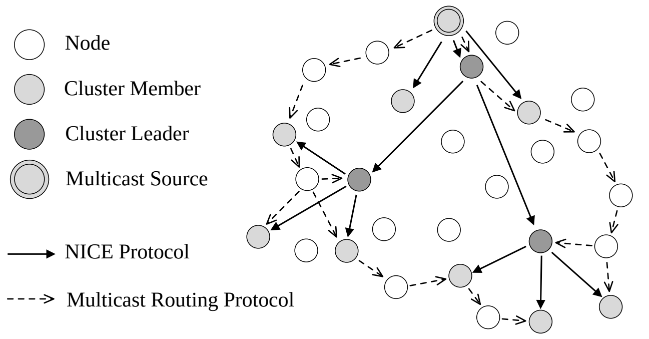

Fig. 15. Layer-dependent data paths.

A. Results of Simulation Experiments

In order to analyse how far NICE-MAN's overlay network may be used to improve delivery ratios for ODMRP, some experiments were conducted. The conditions, under which the simulations were run, correspond to the parameters presented in section V. While 2 packets per second of 512 bytes each were sent using ODMRP, the rate was increased to 4 packets per second for ODMRP enhanced by NICE-MAN's overlay network. In analogy to section V, simulations were run with *Fast ACK Relay* enabled on one hand, and explicitly disabled on the other hand.

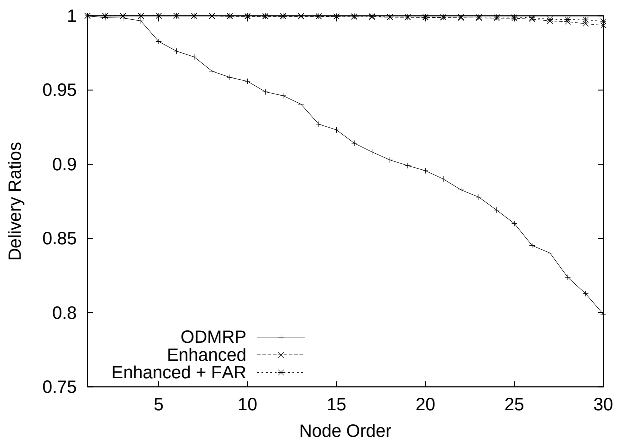

Fig. 16. Delivery Ratios under ODMRP.

Figure 16 shows the delivery ratios as they were obtained during the experiments. As can be seen, delivery ratios are considerably increased by letting nodes use NICE-MAN's overlay network for recovering from packet losses. 10 out of 29 receivers do not experience any unrecoverable packet losses. Most of the remaining receivers suffer only from very few packet losses ($< 0.25\%$). At this point it is unfortunately not fully understood why recovery performs better with *Fast ACK Relay* enabled.

This significantly increased delivery ratio comes however at the cost of highly elevated latencies. As can be seen from figure 17 the latency averaged for all nodes is increased from about $30ms$ to nearly $0.75s$. This results from the fact, that paths used for recovering from packet losses are totally independent from paths used for the multicast data delivery (see figure 15). Nodes receiving retransmission requests are thus often unable to fulfill these requests, as the corresponding data packet may not yet has been received. This will be further amplified as a node, receiving a retransmission request, itself may have lost the same packet.

B. Fast ACK Relay

Finally some measurements illustrating assumed congestion times for ODMRP enhanced by NICE-MAN's overlay network are provided. As can be seen from figure 18 *Fast ACK Relay* lowers a multicast source's idle times. The improvements achieved are however less significant than in section V-A. Recoveries from packet losses take much longer because of data paths being unbound from retransmission paths. As a result, the progress of a multicast source's receive window still is heavily delayed.

VII. Related Work

In order to achieve high delivery ratios in mobile ad-hoc networks, the delivery of multicast data needs to be combined with retransmission requests [10] as well as with an effective congestion control. The RALM protocol [11]

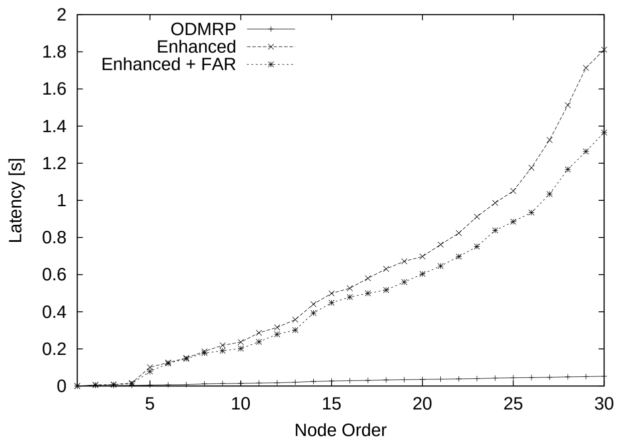

Fig. 17. End-To-End Latencies under ODMRP.

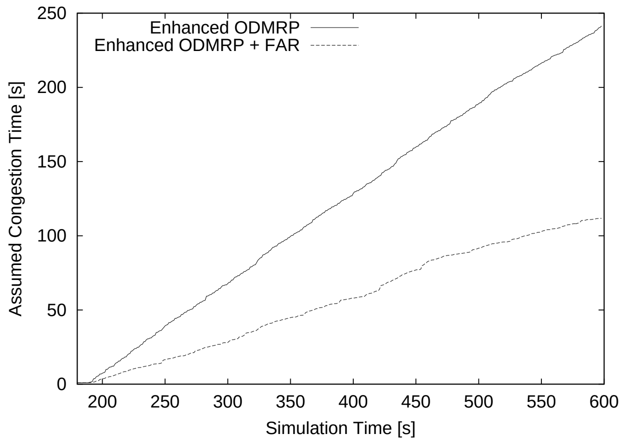

Fig. 18. Assumed congestion durations for ODMRP.

achieves reliable transmissions, by guaranteeing data delivery to perturbed receivers, according to a congestion control based on a send-and-wait mechanism. Nodes noticing packet losses directly alert the multicast source, whereby the source enters an error recovery mode. In this mode, the source contacts all troubled receivers in a round-robin fashion. It lets a node recover from packet losses by retransmitting missing packets to the entire multicast group.

RALM however has been designed for supporting only small sets of receivers. As members suffering from the loss of multicast data directly refer to the multicast source, NACK implosion will occur for large sets of receivers and frequent packet losses. This problem is avoided in NICE-MAN, by recovering from packet losses using a more hierarchical approach. According to section IV-A, NICE-MAN requests retransmissions directly from superior nodes, backwards along the data path. Thereby each node receives retransmission requests only from that nodes the node itself relayed multicast packets to.

Additionally RALM lets a multicast source retransmit data to the entire group. This results in unnecessary network load, in case packet losses occur for only a few nodes. This especially occurs for the so-called *crying baby problem*, which refers to scenarios in which one of the group's members constantly suffers from many packet losses due to a link's bad condition. In such cases data will be constantly retransmitted to the entire group. In NICE-MAN overall network load is reduced by handling the recovery from packet losses in a more local fashion. As retransmissions are requested from immediate neighbours on the overlay, retransmissions will be limited to a network's specific area.

VIII. Conclusion and Further Work

In this paper, basic mechanisms for considerably improving packet delivery ratios for application layer multicast in mobile ad-hoc networks were presented. While these mechanisms were implemented and widely evaluated for NICE-MAN, they show a generic character and can thus be applied to any application layer multicast protocol. As was shown, overlay networks may also be used to considerably increase the reliability of existing multicast protocols based on data delivery through the network layer.

Future work will mainly focus on further improving delivery ratios using overlay networks as well as decreasing a multicast source's idle time. By doing so, we look forward to provide reliable data delivery with acceptable latencies, to a set of 30 and more pedestrians. Poor performance is known to currently result from temporary inconsistencies inside the overlay. While links are established or dropped, an overlay's topology evolves with time. Changing routing information however propagates only slowly inside the overlay because of a mobile ad-hoc network's frequent packet losses. As in this case multicast data will not reach all group members when data is actively relayed using the overlay, assigning the overlay a more passive role for data delivery is to be considered. In addition, the use of multiple sources was not investigated in this paper and needs further attention as well.

References

[1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proceedings ACM Sigcomm 2002*, Pittsburgh, Pennsylvania, 2002.
[2] Y.-H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *ACM SIGMETRICS 2000*, Santa Clara, CA, 2000.
[3] S. Banerjee and B. Bhattacharjee, *A comparative study of application layer multicast protocols*. To appear, 2002.
[4] S. Bloedt, *Efficient End System Multicast for Mobile Ad-Hoc Networks.* To appear, 2003.
[5] S. N. T. Inc., *The QualNet Simulator*. Scalable Network Technologies, Inc., 2001. [Online]. Available: http://www.scalable-networks.com/
[6] X. Hong, M. Gerla, G. Pei, and C. Chiang, *A Group Mobility Model for Ad Hoc Wireless Networks*. ACM International Workshop on Modelling and Simulation of Wireless and Mobile Systems, 1999.
[7] J. Postel, *User Datagram Protocol*. IETF, RFC-768, 1980.
[8] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, 1999, pp. 90–100.
[9] S. Lee, M. Gerla, and C. Chiang, "On-demand multicast routing protocol," in *Proceedings of IEEE WCNC 1999*, New Orleans, 1999.
[10] R. Chandra, V. Ramasubramanian, and K. Birman, "Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks," in *Proc. 21st International Conference on Distributed Computing Systems (ICDCS)*, 2001, pp. 275–283.
[11] K. Tang, K. Obraczka, S.-J. Lee, and M. Gerla, "Reliable adaptive lightweight multicast protocol," in *Proceedings of IEEE ICC 2003*, 2003.