Stable, Congestion-Controlled Application-Layer Multicasting in Pedestrian Ad-hoc Networks

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

Abstract

Ad-hoc networks enable mobile devices to communicate without any fixed infrastructure. While reliable multicasting has been identified as a key application in this context, we analyze different aspects of ad-hoc networks and their impact on application-layer multicast. As a result, we propose our technique of local broadcast-clustering, making use of the wireless medium's broadcast capability. We further describe a simple congestion control as well as a technique for avoiding an overlay's routing inconsistencies and thus for stabilizing the delivery of multicast data.

1 Introduction

Pedestrian ad-hoc networks (PANETs) consist of moderately mobile devices, communicating via the wireless medium. While two devices located in one another's transmission range are able to communicate directly, intermediate devices will be necessary to relay data across multiple hops as the distance increases. Ad-hoc networks are predicted to be of major importance in the future, since many scenarios arise relying on networks that feature extreme ease of deployment. Sightseeing tours as well as fairs and eEducation scenarios can benefit from networks allowing people to spontaneously form freely roaming groups.

Despite their differences, many applications share their need for reliable multicast communication. While different proposals for multicast routing on the network layer have been made [8, 13, 9, 6], we focus on the delivery of multicast data using application-layer approaches [2, 5, 1]. These protocols link all members of the multicast group by means of unicast tunnels, resulting in a logical network called *overlay network*. While multicast packets are duplicated on the application-layer of group members, they are forwarded along unicast tunnels using the overlay's dedicated routing algorithm. Intermediate devices thus only need to handle unicast traffic and are completely unaware of the group's state information. Ad-hoc and overlay networks share different similarities, since in both all devices potentially act as senders, receivers and routers.

Several attempts for multicasting data in ad-hoc networks using application-layer approaches have appeared. AMRoute [10] connects group members using unicast tunnels upon which multicast trees are set up. While a tree is adapted in case of node mobility by including or excluding specific unicast tunnels, the overlay's topology however remains static. Since delivery trees thus not necessarily reflect the underlying network's topology because of node mobility, overall efficiency is likely to decrease over time. PAST-DM [7] overcomes this problem by adapting the overlay to the changes in the underlying network's topology. Multicast trees are set up using a source-based algorithm relying on steiner trees. While the overall tree cost is effectively kept low, the protocol's link-state routing induces an important amount of control flow information. Furthermore the wireless medium's broadcast capability is not addressed, resulting in a poor performance in environments with an increased density of mobile devices.

Approaches in making multicast in ad-hoc networks reliable have emerged. Anonymous gossip (AG, [4]) extends existing unreliable multicast routing protocols by providing a simple technique for error recovery. As a node notices packet losses, it starts gossiping with a nearby member of the multicast group. Both nodes exchange information about successfully received and missing packets, upon which error recovery is initiated. AG however does not include techniques for avoiding congestion by throttling the rate a multicast source sends data at. Newly transmitted data therefore steadily perturbs a receiver's error recovery, resulting in poor performance at higher data rates. ReACT [12] also provides reliability by extending existing multicast routing protocols using a local and a global error recovery. While local error recovery is applied when a packet loss is recognized to be caused by local transmission errors, the protocol reverts to global recovery in case of congested networks. In the latter case, a multicast source reliably retransmits packets to perturbed receivers using a stop-and-wait technique. Local transmission errors are discerned from global congestion by examining the IP-header's congested flag on the one hand, and the length of the packet-queue on the MAC-layer on the other hand.

In this work, which arose from the MAMAS project funded by the Deutsche Forschungsgemeinschaft, we address the peculiarities of PANETs and their impact on application-layer multicast. Section 2 presents our techniques for efficiently delivering multicast data on the application-layer. The experimental setup we chose for evaluating our findings is described in section 3. Simulation results are available in section 4. Section 5 concludes by giving a short summary and an overview of topics that will be handled in future work.

2 Design of Multicast Techniques

This section presents different techniques that were tailored to allow efficient delivery of multicast data through overlay networks in PANETs. An important design goal was to keep a generic character in order to make them applicable to arbitrary overlay networks. Each mechanism addresses one of the peculiarities of wireless environments, such as the broadcast medium, frequent packet losses, node mobility and congested networks.

2.1 Local Broadcast Clusters

Depending on applications and scenarios, a multicast group's receivers will most often find themselves clustered into small groups or teams consisting of a few members each. Dimensions of these groups will usually be limited to a few ten meters, resulting in an increased density of mobile devices. As is illustrated in figure 1, multicasting data using an overlay is highly inefficient in such scenarios since packets are forwarded along multiple overlay hops although devices probably are within direct transmission range. This problem can be overcome by extending an overlay network by means of *Local Broadcast Clusters* (*LBCs*). These are based on cross-layer information by explicitly making use of the wireless medium's broadcast capability.

A local broadcast cluster exists around every member of the multicast group that has joined the overlay. The latter nodes by this become leader of their LBC. Additionally to all operations usually associated with the overlay's maintenance, these nodes periodically broadcast dedicated heartbeat messages signaling the LBC's presence to nearby group members. A LBC's dimension thus is limited by the transmission range of its respective leader. Members of the multicast group receiving a LBC leader's heartbeat messages do not join the overlay and thus become locally joined

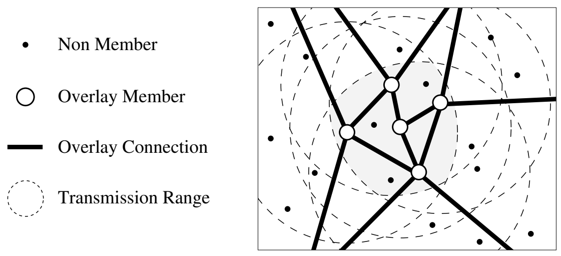

Figure 1. Overlay in dense node scenario.

nodes. To reduce total overhead, locally joined nodes do not exchange any control flow information with other nodes.

Note that a locally joined node may receive heartbeats from multiple LBC leaders as it might be located within transmission range of different overlay nodes. In this case the node computes the best quality leader (via the heartbeat's loss rate) and assigns itself to the respective LBC. This quality indicator is also used to detect the loss of LBC leaders. As soon as a LBC's quality drops below a certain threshold, it is declared as lost. Such behavior is usually expected when a locally joined node moves out of its LBC leader's transmission range. In order to receive further multicast data, the node now joins the overlay and thereby becomes leader of its own newly created LBC.

In analogy to the creation of a LBC by a specific node joining the overlay, a LBC's dissolution can be defined by letting the respective leader leave the overlay. Such a mechanism is required to achieve the most effective use of the wireless medium. LBCs indeed become redundant as soon as two LBC leaders, find themselves within one another's transmission range. This is avoided by letting a node retire from the overlay after having received a heartbeat from a nearby LBC leader with a higher node address. The retiring node thereby abandons its own LBC and joins the LBC indicated by the received heartbeat. Obviously, group members that had locally joined the former LBC now either need to assign themselves to another nearby LBC, or are required to join the overlay.

According to the above definition of LBCs, each member of the multicast group finds itself in one of two possible states during its entire group membership. It has joined either the overlay, or a LBC. While transitions between both states are possible at every time, a node handles the sending and forwarding of multicast data to group members depending on its current state. Members of the overlay send and forward data through unicast tunnels to nearby overlay nodes. This forwarding is done according to the overlay's routing protocol. Additionally, overlay nodes broadcast and thus forward data to their LBC members using only one medium access. Locally joined multicast sources simply

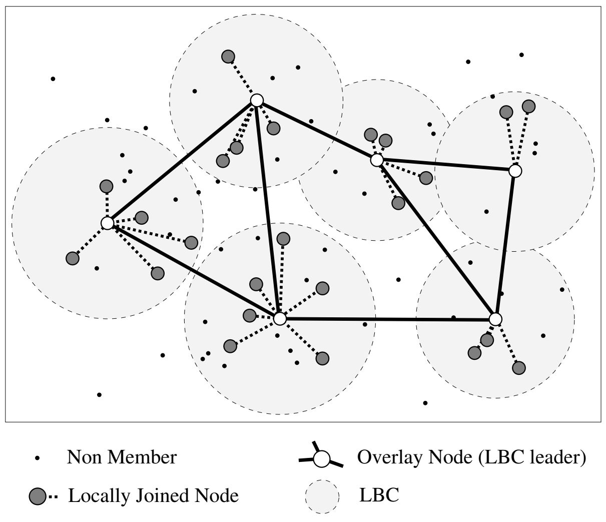

Figure 2. Virtual ad-hoc infrastructure.

unicast data to their LBC leader which then takes care of both data forwarding to overlay nodes via unicast tunnels and broadcasting to LBC members.

Figure 2 shows the resulting network topology which consists of LBC leaders connected through unicast tunnels. Depending on the transmission range of overlay nodes, LBCs might be of different size. One of the key benefits of LBCs is, that they might hold an arbitrary number of locally joined group members. Since the latter nodes do not introduce any control flow, the total overhead needed for the overlay's maintenance as well as for data forwarding is drastically reduced. It furthermore is limited by the area group members occupy and not by the number of group members itself. The overlay now becomes comparable to a '*virtual ad-hoc infrastructure*' that is used for data forwarding across multi-hop distances. This infrastructure is highly flexible and is automatically extended or released depending on a multicast group's needs. Furthermore it might preferably hold more powerful devices which results in a relief to weaker nodes.

2.2 Selective Retransmission Requests

Packet losses are commonly recognized by means of missing sequence numbers. Error recovery is preferably achieved by requesting retransmissions of specific packets using a receiver-oriented approach based on the sending of NACK-packets. According to section 2.1, the way retransmission are requested depends on a node having joined the overlay or a LBC.

A member of the overlay requests retransmissions from the node that multicast data is currently received from. This node will henceforth be called a member's *parent node*[1]. The parent node most often is a nearby overlay member, unless the multicast source happens to be a locally joined group member inside the requesting node's own LBC. Since this way retransmission requests are addressed in a hierarchical fashion, NACK-implosion at the multicast source is avoided and hence scalability is enhanced.

Locally joined nodes request retransmissions from their LBC leader. To avoid redundant accesses to the wireless medium, members of a LBC broadcast retransmission requests after a certain, randomly chosen delay. This enables nearby LBC members to notice ongoing retransmissions and thereby suppress their own requests.

To avoid the sending of inappropriate requests[2] an additional field is added to forwarded and retransmitted packets. It indicates the highest sequence number, up to which the forwarding node successfully received all packets, and up to which retransmissions may be requested. Since a few packets beyond the indicated sequence number might be available, this mechanism slightly increases latencies but effectively reduces medium accesses.

2.3 Buffer Management

Receiver-oriented reliability mechanisms cause the problem of infinite buffers. Because parent nodes are only informed about lost and not about successfully forwarded packets, they may not free their buffers since further retransmission requests can potentially be received at any time. To limit buffer sizes, nodes inform their parents about successfully received data using periodically sent ACK-packets. These include the highest sequence number, up to which data was successfully received. A parent node gathers these sequence numbers from all its child nodes, adds its own sequence number and computes the minimum. The resulting sequence number (called a node's *minimal cumulated sequence number*) indicates up to which packet buffers may safely be freed.

In conjunction with LBCs introduced in section 2.1 the problem arises, that locally joined members are unknown to their LBC leader. The latter hence is unable to gather sequence numbers from its LBC members. Locally joined group members for this reason omit sending ACKs to their LBC leader and therefore do not affect the leader's buffer management. In case an overlay node's buffer quickly progresses, members of the respective LBC might not be able to recover from packet losses. Instead of immediately discarding packets as indicated by the minimal cumulated sequence number, overlay nodes shift packets from their de-

[1]Note that each parent node forwards multicast data to a set of group members, which are called the parent's *child nodes*.

[2]A request is considered as inappropriate when the requested packet is not available from the parent node because of the parent's own packet losses.

fault buffer into their hereby introduced *local error correction buffer*. This second buffer is held by every node and its purpose is to extend the period of time, for which locally joined nodes are able to recover from packet losses. Since this buffer's capacity also is limited, packets may need to be discarded as other packets are shifted in.

2.4 Congestion Control

Congestion control has been identified as a crucial element for reliable multicast data delivery in ad-hoc networks. Since the wireless medium is highly sensitive to network load, the sending of new data by a multicast source is counterproductive as long as packet losses persist for a set of receivers. Newly sent packets complicate the recovery of packet losses as they cause network load to increase and packets to be dropped due to collisions.

In order to adjust its sending of data, a multicast source must be informed about the group's condition. The buffer management outlined in section 2.3 can easily be extended into a simple but effective congestion control by replacing the sequence number contained in ACK-packets through a node's computed minimal cumulated sequence number. By doing so, the *highest* sequence number, up to which the *worst* receiver successfully received all packets, propagates towards the multicast source. Upon this sequence number, a multicast source computes the difference between the worst receiver's state and the sequence number of the next multicast packet to send. As the difference increases, a source recognizes that one or more receivers hardly recover from their packet losses. The source hence decreases the rate it sends data at, in order to release the medium in favor of the receiver's error recovery. It might also be convenient to let a source completely stop the sending of further data as soon as the computed difference exceeds a certain threshold.

As with time receivers recover from packet losses, increases of the minimal cumulated sequence numbers will propagate towards the source. The periodicity at which ACK-packets are sent thereby becomes a key factor affecting the rate a source will send data at. If ΔT denotes the period of time between two ACKs, h the depth of multicast tree and if packet losses are neglected, information contained in ACK-packets reaches a multicast source with a delay of $h \cdot \Delta T$ in the worst case. Obviously information will further be delayed when packet losses are considered. To reduce overall latency, a node recomputes its minimal cumulated sequence number on reception of every ACK-packet and compares the result to the previously obtained sequence number. If the difference exceeds a certain threshold, the progress is declared to be important and is immediately forwarded to the node's parent. The entire mechanism thereby becomes adaptive to the receiver's error recovery.

2.5 Avoiding Inconsistencies of Overlay Routing

In order to keep application-layer delivery of multicast data efficient, an overlay's topology constantly needs to be adapted to the ad-hoc network's topology. As a consequence, an overlay often needs to adjust its routing to match the topology changes. Such routing updates are mostly realized by letting group members periodically exchange routing information. The propagation of routing information inside the overlay however is problematic, since wireless environments suffer from many packet losses. Relevant routing updates might thus be heavily delayed, often resulting in a partially inconsistent routing. As routing inconsistencies cause many consecutive packet losses, they are unacceptable considering the delivery of multicast data.

Since in case of topology changes parts of the overlay can not reliably be used for the routing of data, we argue that it is necessary to assign the overlay a more passive role considering the delivery of multicast data. We therefore decided to completely separate the routing of true multicast application data from the overlay's integrated multicast routing. The latter most often is based on a simple forwarding table held in every node. This table is from now on called a node's *Overlay Forwarding Table* (*OFT*). Using the definition of OFTs, routing inconsistencies might be described as follows. Let a node p_1 be parent of a child node c. c will thus appear in p_1's OFT. As the tunnel between both nodes degrades, it eventually is shut down resulting in c being deleted from p_1's OFT. In order to further receive multicast data, c must appear in another node's (called p_2) OFT. Since periodically sent routing updates, which would lead c to eventually appear in p_2's OFT, might be heavily delayed because of packet losses, c will not receive multicast data for an uncertain time.

While OFTs can still be used for routing of control flow information that does not rely on high reliability, data sent by multicast applications must hence be routed in a different manner. To achieve this, each member of the overlay holds a second forwarding table, called the *Application Forwarding Table* (*AFT*). Entries in a node's AFT originate from group members explicitly registering at the node for data forwarding. Each group member therefore designates a parent node which is contacted using dedicated packets of type `RegisterAsChild`. On reception of such a packet, the parent node places the child's address in its AFT and thereby becomes responsible for forwarding multicast data to the respective node.

Knowledge about which parent a node should register at, in order to efficiently acquire multicast data, is obtained from the overlay's OFTs. Each overlay member sends dedicated packets of type `UpdateMulticastTree` to the nodes indicated by its OFT. Since OFTs reflect the overlay's multicast routing, it is guaranteed that on reception of an

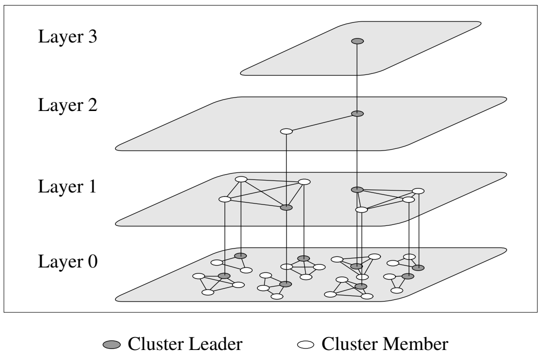

Figure 3. NICE's hierarchical cluster topology.

update-message, the recipient learns about its optimal parent node which is then contacted by means of `RegisterAsChild`-packets. Note that different optimizations can be achieved considering the overhead resulting from the presented mechanism. Packet of type `UpdateMulticastTree` need only be sent in case of topology changes. These become visible through new entries in a nodes OFT. Once a node has been registered as a child inside the AFT, the node obviously knows about its best parent and thus no longer needs to be informed about it.

The key aspect of this technique is, that in case of routing inconsistencies the child node c remains in p_1's AFT although it has been deleted from the OFT. The routing of multicast data hence is stable, since p_1 still is responsible for forwarding multicast data to c. After the inconsistency has eventually been resolved, c appears in p_2's OFT. Because of the change in its OFT, p_2 contacts c using `UpdateMulticastTree`-packets. As the node is informed about its new best performant parent, c thus registers as a child of p_2. On reception of the first multicast packet through p_2, c recognizes its registration as fulfilled. To stop p_1's data forwarding to c, the latter unregisters at p_1 using packets of type `UnregisterAsChild`.

Note that by using this technique, the routing of application data is strictly separated from the overlay's integrated routing. Since unicast tunnels are used by both routing algorithms, they may only be closed when no longer needed by both routing mechanisms. This of course only is of concern, when tunnels are based on connection-oriented transport protocols.

3 Experimental Setup

To evaluate the performance of the techniques presented in section 2, we first implemented the NICE [2] protocol in the simulation environment *QualNet*. NICE is a hierarchical protocol originating from the fixed Internet. It organizes its group members by means of hierarchical, fully meshed clusters, resulting in a network topology as depicted by figure 3. As can be seen, all group members are present on layer 0 of the hierarchy. On this base, the hierarchy is built according to the following recursive procedure.

- All nodes on layer i of the hierarchy are assigned to clusters so that each clusters counts at least k but no more than $3k-1$ members, where k is a fixed constant.
- Among all clusters on layer i, a node in each cluster's center is elected as its leader.
- All cluster leaders present on layer i join layer $i+1$.

The procedure is recursively repeated until a single node remains on the top layer. Distances to nodes are measured by means of *Round Trip Times* (*RTTs*) and periodically exchanged between all members of a cluster using dedicated heartbeat messages. NICE seems to be a promising protocol for realizing multicast data delivery not only in the fixed Internet but also in mobile environments. Indeed, NICE induces only a small amount of control flow (compared e.g. to Narada [5]) and includes basic mechanisms for adapting its overlay to the topology of the underlying network. This is achieved by enabling cluster members (not leaders) to move from one cluster to an adjacent cluster. Additionally, because of the hierarchy's special structure, the forwarding of data is implicitly defined. Since no extra routing protocol is needed, control flow overhead is kept low. Further details about NICE can be obtained from [2].

In a second step, we extended the NICE protocol by integrating our techniques as described in section 2. The key features of the resulting *NICE-MAN* protocol may be summarized as follows.

- NICE-MAN features a flexible overlay topology which is adapted to the underlying network using a *hop-count*-metric [3]. The latter are obtained using cross-layer communication to access a node's routing tables on the network layer.
- The wireless medium's broadcast capability is effectively used by means of LBCs.
- Reliability mechanisms are included via selective retransmissions (a packet's retransmission is requested up to 10 times) and NACK avoidance. Due to simplicity, NACK-packets request only one single packet's retransmission in the current version.
- Congestion control, although being integrated, is not yet rate-adaptive. Instead, it lets a multicast source temporarily suspend its sending of new multicast data in case of congestion. The threshold, at which the source reverts to congestion control, is given by the capacity of the node's default buffer.

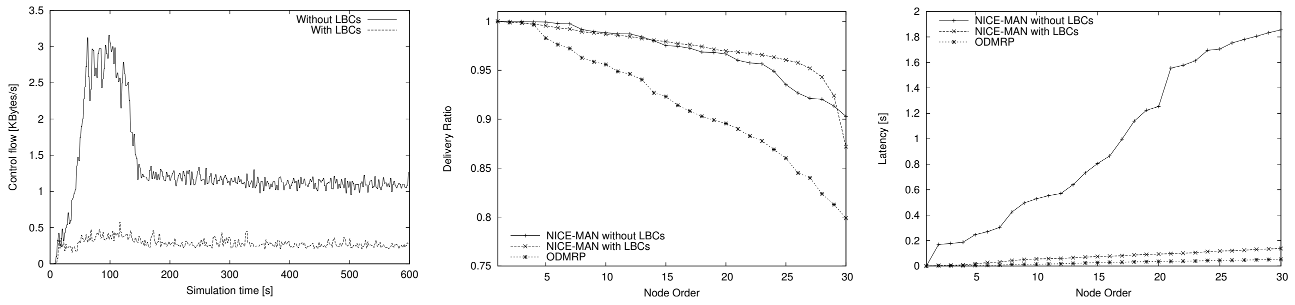

Figure 4. Effects of local broadcast clusters.

- The forwarding of an application's multicast data is unbound from the overlay's integrated multicast routing, in order to avoid routing inconsistencies as described in section 2.5.

Considering the characteristics of pedestrian ad-hoc networks, we run simulations with the following scenarios. A total of 30 members join the multicast group within the first two minutes of simulation time. The following 60 seconds are used to let the overlay's topology stabilize. After a total of three minutes simulation time, a source transmits data at different rates according to the experiments presented in section 4. While packet size is fixed to 512 *bytes*, the data rate is modified via the number of packets sent per second. Unless specified otherwise, the rate is set to 4 packets per second. Each data transmission lasts for 5 minutes, after which 2 more minutes of simulation time follow. These may be required to complete the transmission, since a source will suspend its data delivery in case of congestion.

All nodes roam on a surface of $1000m$ by $1000m$. Group members move with at most $1\frac{m}{s}$ and a pause time of $120s$ according to the Reference Point Group Mobility model (RPGM). RPGM clusters contain between 1 and 5 nodes and have a radius of $80m$. Additionally, 50 nodes (that do not join the multicast group) randomly move with at most $2\frac{m}{s}$ and a pause time of $30s$. Transmission ranges are set to $150m$, link bandwidth is $2\frac{Mbit}{s}$. Unicast tunnels are realized by using UDP as transport protocol. Unicast routing on the network layer is done using AODV [11]. IEEE 802.11 with the RTS/CTS extension is used for accessing the medium. In order to smooth measurements, 20 scenarios were pregenerated and run with different seed values.

4 Results of Simulation Experiments

In this section we present the results of our simulation experiments, in order to evaluate our multicast techniques as they were integrated in NICE-MAN. We first underline the importance of considering the wireless medium's broadcast capability in section 4.1. Afterwards we demonstrate the consequences of neglecting routing inconsistencies in section 4.2. The aspect of different loads transmitted by the multicast source is studied in section 4.3.

Some of the figures presented below show *node orders*. These give an overview of simulation results by averaging measurements for every node and sorting the nodes from the best (on the diagram's left) to the worst (on the right). We define *delivery ratios* as the fraction of multicast packets, that a node has successfully received. Additionally, the *fraction of reliably delivered packets* refers to the portion of multicast packets, that were delivered to all nodes.

4.1 Effects of Local Broadcast Clusters

To better evaluate the effects of using the wireless medium's broadcast capability, measurements in this section were obtained by disabling reliability (retransmission requests, congestion control and routing inconsistency avoidance).

Figure 4.a) shows the effect of local broadcast clusters (cf. section 2.1), considering the control flow overhead induced on the application-layer by the overlay's maintenance. Since group members join LBCs in case they are located within transmission range of an overlay node, the overlay's size is drastically reduced. As locally joined group members do not introduce additional control flow, the total overhead is decreased. Especially within the first 150 seconds, in which nodes join the multicast group and during which the NICE hierarchy is set up, the high volume of control flow data can successfully be avoided.

As can be seen from figure 4.b), the effect of introducing LBCs is somewhat positive on delivery ratios. Compared to ODMRP [8] (provided for comparison to a network-layer protocol), a noticeable increase of performance is visible. Successful packet delivery is more likely, since unicast tunnels (used for data dissemination) are covered by basic retransmissions on the MAC-layer.

Considering the latencies shown in figure 4.c), a drastic improvement can be achieved by making use of the wireless medium's broadcast capability. High latencies here re-

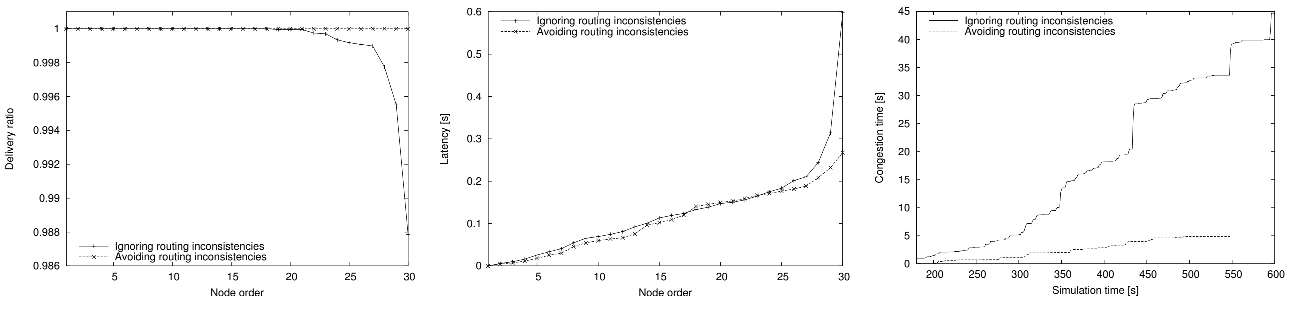

Figure 5. Effects of inconsistent overlay routing.

sult from several nodes being in one another's transmission range. When connected through an overlay, packet forwarding between nearby nodes initially results in frequent collisions. Since IEEE 802.11 with RTS/CTS is used for medium access, further attempts at transmitting a packet will be made by each node. While the MAC layer's contention window (used for collision avoidance) is exponentially increased as collisions persist, packets eventually are delivered with increased latency. The MAC-layer's retransmissions thus are counterproductive when low latencies are desired. Effectively making use of the wireless medium's broadcast capability in this case becomes a key aspect when delivering multicast data using application-layer approaches in conjunction with IEEE 802.11. Performance thereby gets comparable to ODMRP.

4.2 Effects of Inconsistent Overlay Routing

Measurements in this section were obtained using activated reliability mechanisms. While the default buffers size was set to 24 packets, the local error correction buffer's capacity was set to 48 packets.

As retransmission requests and congestion control are enabled, overall delivery ratios are increased as well as latencies. Unexpectedly however figure 5.a) shows that perfect delivery ratios are not achieved for all nodes. In course of simulation experiments we identified inconsistencies in the overlay's routing as the major cause of persisting packet losses. While degrading unicast tunnels are shut down and replaced as described in section 2.5, a node might temporarily not receive multicast data sent by the source due to its lacking parent node. Since the node does not yet know about its (future) parent node and since routing updates might be delayed, the node is not able to send acknowledgments. During the routing's inconsistency the node will therefore neither affect nor trigger the source's congestion control. As the source goes on sending new data, packet buffering in the overlay's nodes might not be sufficient to allow a node suffering under inconsistent routing to perform full error recovery. After separating the routing of the application's data from the overlay's integrated routing, perfect delivery ratios are achieved for all nodes.

Although routing inconsistencies only affect latencies of a few group members, improvements in figure 5.b) are nevertheless clearly visible for the worst nodes. This is achieved because all nodes now obtain multicast data during the entire transmission. Neglecting routing inconsistencies in this case leads to group members needing to recover from the loss of many consecutive packets after an inconsistency eventually is resolved. The longer the inconsistency was lasting, the higher an affected packet's latency will be. Stabilizing data delivery successfully avoids this problem.

We define a source's *congestion time* as the amount of time a source suspends its data delivery due to missing acknowledgments. As can be seen from figure 5.d), avoiding routing inconsistencies positively affects congestion time. As group members steadily receive multicast data, a multicast source will not be blocked by nodes that formerly suffered under routing inconsistencies.

4.3 Effects of Different Data Rates

While in this section buffer sizes are fixed to 20 packets for the default buffer and to 40 packets for the local error correction buffer, the load offered by the multicast source is varied from 8 $\frac{Kbit}{s}$ up to 32 $\frac{Kbit}{s}$.

Figure 6.a) shows the resulting fraction of reliably delivered packets as well as the worst receiver's latency. As can be seen, perfect delivery ratios are not achieved for the highest data rate, since the number of retransmission requests sent by a node is limited. Overall latency increases because error recovery now takes longer as collisions persist.

Figure 6.b) shows the respective congestion time and the effectively achieved data rate. The need of congestion control in ad-hoc networks becomes visible in this diagram. As the load offered by the multicast source increases, congestion time increases as well in order not to overburden the network. The effectively achieved data rate thus more and more differs from the data rate supplied by the source.

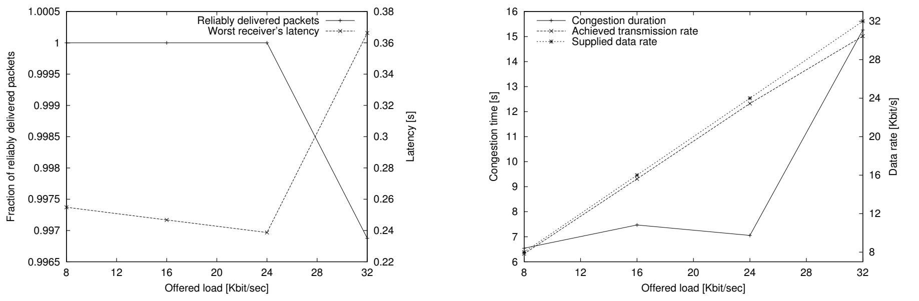

Figure 6. Effects of different data rates.

5 Conclusion and Future Work

In this work we present our first steps in providing efficient application-layer multicast in ad-hoc networks with low mobility. We proposed different mechanisms that take the peculiarities of these networks and their effect on application-layer protocols into account. The mechanisms comprise an effective use of the wireless medium's broadcast capability, a simple congestion control and a technique for stabilizing data delivery in case of routing inconsistencies inside the overlay. Our simulation experiments show that the presented mechanisms allow the delivery of multicast data with up to 3 $\frac{KBytes}{s}$ within acceptable latencies to a set of 30 pedestrians. At a data rate of 4 $\frac{KBytes}{s}$ overall delivery ratios remain high enough to support different types of applications that can bear with packet losses less than 0.3%.

Future work will further focus on reliability considering the delivery of multicast data using application-layer approaches. The sending of retransmission requests needs dedicated attention, since most applications require high delivery ratios and low latencies. These however can only be obtained by quickly recovering from packet losses. The effect of retransmission requests and packet retransmissions on congested networks needs to be closely analyzed to achieve maximum performance. In addition, the potential of extensive cross-layer communication to realize a true rate-adaptive congestion control will be examined. Since congestion control currently also suffers from single nodes slowing down the entire group, methods excluding such group members will be studied. Additionally the possibility of simplifying overlay networks to meet the requirements of multicast data delivery will be analyzed.

References

[1] S. Banerjee and B. Bhattacharjee. *A comparative study of application layer multicast protocols*. Unpublished, 2002.

[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *ACM SIGCOMM 2002*, Pittsburgh, PA, USA, June 2002.

[3] S. Bloedt. Efficient end system multicast for mobile ad hoc networks. In *The First International Workshop on Mobile Peer-to-Peer Computing*, Orlando, Florida, USA, March 2004.

[4] 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)*, pages 275–283, 2001.

[5] Y. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *ACM SIGMETRICS 2000*, pages 1–12, Santa Clara, California, USA, June 2000.

[6] S. Das, B. Manoj, and C. Murthy. A dynamic core based multicast routing protocol for ad hoc wireless networks. In *Proc. ACM MOBIHOC'02*, Lausanne, Switzerland, June 2002.

[7] C. Gui and P. Mohapatra. Efficient overlay multicast for mobile ad hoc networks. In *The Wireless Communications and Networking Conference (WCNC)*, New Orleans, Louisiana, USA, Mar. 2003.

[8] S.-J. Lee, M. Gerla, and C.-C. Chiang. On-demand multicast routing protocol. In *Proceedings of IEEE WCNC 1999*, pages 1298–1304, New Orleans, USA, Sept. 1999.

[9] S.-J. Lee, W. Su, J. Hsu, M. Gerla, and B. R. A performance comparison study of ad hoc wireless multicast protocols. In *Proceedings of IEEE INFOCOM 2000*, pages 565–574, Tel Aviv, Israel, Mar. 2000.

[10] M. Liu, R. R. Talpade, and A. McAuley. AMRoute: Ad-hoc multicast routing protocol. Technical Report TR 99-1, Center for Satellite and Hybrid Communication Networks, 1999.

[11] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (AODV) routing, Feb. 2003.

[12] V. Rajendran, K. Obraczka, Y. Yi, S.-J. Lee, K. Tang, and M. Gerla. Combining source- and localized recovery to achieve reliable multicast in multi-hop ad hoc networks. In *Proceedings of the Networking 2004*, Athens, Greece, May 2004.

[13] E. M. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Proceedings of MobiCom '99, Seattle, WA, USA*, pages 207–218, Aug. 1999.