LULEÅ TEKNISKA UNIVERSITET

2001:01

RESEARCH REPORT

A Literature Review of Recent Developments in Reliable Multicast Error Handling

Stefan Elf, Peter Parnes

Research report

Institutionen för Systemteknik
Avdelningen för Programvaruteknik

2001:01 • ISSN: 1402-1528 • ISRN: LTU-FR--01/01--SE

A Literature Review of Recent Developments in Reliable Multicast Error Handling

Stefan Elf[1], Peter Parnes
Luleå University of Technology
Centre for Distance-spanning Technology
Department of Computer Science
SE-971 87 Luleå, Sweden.
Email: {self,peppar}@cdt.luth.se

January 2001

[1]Stefan Elf is also with Ericsson Erisoft AB, SE-932 83 Ursviken, Sweden

Abstract

As the number of applications directed towards group collaboration and group media dissemination increases, the interest for multicast protocols has also grown during recent years.

Several applications have a need for reliability. This is true for the use of shared network editors, shared white-boards, networked games, etc. An increased popularity of media and application distribution over the Internet and the beginning growth of mobile computing with its inherent inhomogeneity and higher loss-rate, will create a demand for security which will increase the need for reliable transmissions.

This paper aims to study reliable multicast protocols from an error correction perspective. During recent years, a number of protocols have been proposed. The reliability properties of these lies partly with error correction measures, but to a great extent also with different topological and timing measures which are aimed at boosting protocol efficiency.

As it turns out, there are mainly two groups of error correction measures which are used. They can be divided into reactive measures, and proactive. The development in this field of research is very fast. Although a number of studies have already been done, this paper aims to summarise the most recent developments in the area of reliable multicast.

This is a literature study and the sources are mainly English-language references from the INSPEC database and the ACM, IEEE, and IEEE Computer Society's digital libraries, essentially from 1997 until 2000.

1 Introduction

The most common way to work today over the Internet has been over a point-to-point connection since both people and applications have interacted with each other on a node-to-node basis. A growing class of applications in which people collaborate has created a demand for something more efficient than simply broadcasting the data to all nodes in a network. This rather novel multicast technique turns out not to be very new. Multicast was first proposed some 20 years ago, but in spite of the increasing popularity of the applications it has until very recently been a rather limited facility. In the MBone, multicast capable routers are available and between these multicast islands, data is encapsulated, tunnelled, in unicast IP packets.

Multicast capabilities are implemented in several protocol layers (Fig. 1). In the link layer there is of course a need to support multicast addresses. In the network layer there is e.g. the IGMP protocol for handling joins and leaves to and from multicast groups and multicast routing protocols, such as MOSPF, DVMRP, and PIM.

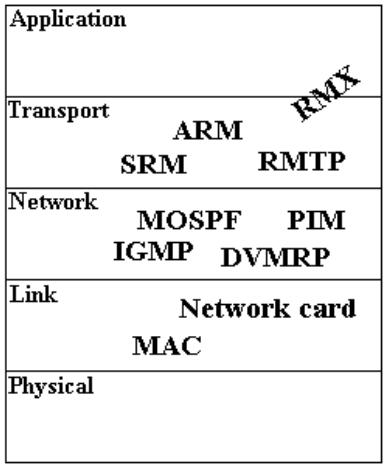

Figure 1: Multicast protocol layers.

The transport layer handles reliability such as in TCP and this is the case also when it comes to multicast. Most of the reliable multicast protocols are found in the transport layer. Some protocols operate also in the application layer. These are designed to take into account also the semantics of the multicast packets and to handle application specific requests on data. These protocols build on the ALF [1] principle. It should be pointed out that the border between what is considered transport or application layer is very thin.

In [2] Byers et al. suggest the following criteria for an ideal protocol. The protocol should be

- **Reliable**, guaranteeing delivery to all receivers
- **Efficient**, minimal processing overhead
- **On demand**, download may start at any time
- **Tolerant**, the protocol should handle a heterogenous receiver population

Apart from the criteria given above, it is important that a multicast protocol is scalable, since it aims to address a one-to-many or many-to-many scenario. The protocol must not degenerate as the number of receivers grow. Depending on the application, real-time requirements may also be pertinent.

This review will target the reliability and the tolerance and will study the current protocols in the view of how they handle errors, not how they are arranged regarding the strategy for information dissemination. The goal is not to grade the protocols such as by efficiency, but rather to make a small inventory of the error handling methods that are used, and how these are combined.

During the last couple of years, a number of studies of reliable multicast have already been done. Most of them are focused on one single or on a class of protocols or topologies. A study of error recovery techniques for audiovisual multicast applications was made by Carle and Biersack [3] and a comprehensive taxonomy of multicast protocols was done by Obraczka in 1998 [4]. In [5], Lacher et al. compares performance in centralized versus distributed error recovery.

The remainder of this paper is structured as follows. In section 2 the error handling methods of some reliable multicast protocols is discussed. In section 3 some recent work is described. Lastly, in section 4 conclusions and suggestions for further work into error handling in reliable multicast, is presented.

2 Error handling

In this section, the current methods of error handling in reliable multicast protocols will be reviewed. There are several ways to categorise these methods, such as between sender-initiated or receiver-initiated protocols, or between reactive (repeating) or proactive (using redundancy) protocols. There are, or course, also hybrids.

2.1 Reactive or proactive

Several reliable multicast protocols have been proposed but essentially, they all are mainly either reactive or proactive (Fig. 2) with regards to the way packet loss is handled. Reactive protocols will repeat information on detection of losses, while proactive protocols will use redundancy, or parity, information to supply the receivers with knowledge to repair losses. Among the reactive protocols are the ACK-based and the NAK-based.

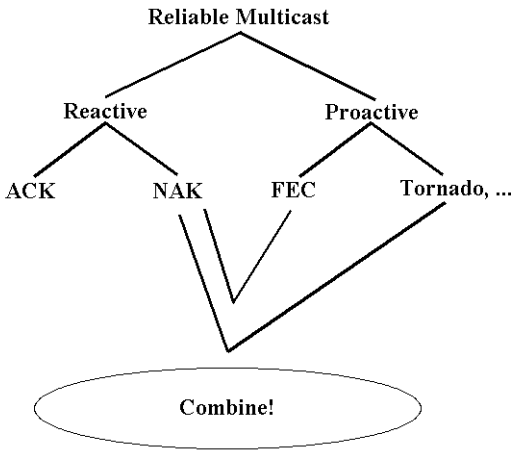

Figure 2: Protocol type breakdown.

ACK-based protocols belong to the group of protocols which are also called sender-initiated or sender reliable, in that the sender must keep track of all receivers and whether each receiver has ACK'ed a specific packet or not. Sender-initiated protocols can be said to poll the receivers for their state by sending out a data packet.

ACK-based protocols have an advantage in that the sender will always know when a packet can be discarded and the buffer space reclaimed. Pending that, the packet must be buffered by the sender. Unfortunately, this is about the only advantage this class of multicast protocols can exhibit. Maintaining information about all receivers and buffering un-ACK'ed packets draws CPU power and memory space from the sending node. Therefore this solution does not scale to a larger audience. When the number of receivers grow to be several thousands, the number of ACK's that must be sent to the sender will be overwhelming, and will eventually result in what is called an *ACK implosion*.

To handle this ACK implosion ACK-based protocols build ACK-trees

using techniques like expanding ring search, ERS. In these trees, some of the receivers will acquire the status of *designated receivers* and will act as proxies for the sender regarding ACK's and repetition of packets. There are not many protocols that rely solely on ACK's from the receivers, one example being RMTP [6].

In NAK-based protocols it is not necessary for the sender to know about every receiver. NAK-based protocols are also called receiver-initiated, since it is the receivers, through the use of negative acknowledgements, NAK's, poll the sender for corrective information.

NAK-based protocols have the advantage of being scalable to a large audience. This is possible while each receiver is responsible only for its own state. The sender need not keep the state or identity of each receiver. If there is a large number of receivers that experience packet loss of a magnitude, the sender will instead face a possible *NAK implosion*. The NAK's can instead be multicast.

Receivers use timers in order to determine that a packet has been lost and that a request for a retransmission is necessary. A NAK is never sent before this timer has expired. The expiry time is either a function of the receiver's distance from the sender, or a random value, or a combination as in Scalable Reliable Multicast [7]. The aim is that no two receivers shall have the same timer value. Another receiver in need of the same packet, will detect the multicast NAK and reset its own timer and then continue waiting for correction information. The retransmitted data is of course also multicast to the entire group in this source-driven recovery. This class of receiver-initiated protocols with NAK avoidance schemes, are sometimes referred to as RINA [8] protocols.

Since NAK-based protocols exhibit some advantages in relation to ACK-based, there have been substantial efforts to improve on this class of protocols.

There seems to be a rather overwhelming concensus that receiver-initiated protocols perform better than sender-initiated, the possible exception being XTP, the Xpress Transport Protocol [9, 10].

Repeating packets either on loss detection at the sender or on request from the receivers generates excessive traffic. Alternate proactive schemes have thus been explored. A proactive protocol is not dependent on an ACK or a NAK if the packet loss is within reasonable limits. Redundant information is incorporated in the packet stream. This allows the receivers to repair a certain amount of packet loss without any further actions of requesting more data. The methods used include erasure codes and forward error correction schemes.

The Digital fountain approach is discussed by Byers et al. [2]. A digital

fountain is a special application of efficient erasure codes called *tornado codes*, in which a data set is encoded in such a way, that if the original K packets are encoded, a receiver need only receive *any* $K(1+\epsilon)$ packets, i.e. slightly (ϵ can typically be 0.05) more than the original number, in order to be able to decode the original data set. If the sender keeps transmitting packets, there need not be any feedback at all, since a receiver that has packets missing, need only listen a while longer to obtain suitable packets from which the missing ones can be determined.

As the amount of packet loss cannot be foreseen at any given instant, these proactive schemes must be combined with a reactive. When a receiver does not obtain the amount of information necessary to decode the data, it must eventually request repair actions from the sender. The sender will then respond with the requested packets. These repair packets will be able to correct several packet losses, not only at the originally requesting host, but at any host. This represents a large efficiency gain.

The drawback carried by sending redundant information is obviously that the required bandwidth is increased whether it is needed or not. The data must also be blocked, and that this is not always suitable for streaming media. The blocking will introduce delays in the data delivery.

Using FEC only as a repair action on a lossless path will not generate any excess traffic, but a digital fountain will.

2.2 Error Concealment

Another technique that can be used in certain instances is to do nothing at all, i.e. to let the packet stay lost and leave it to the application to handle the error. The advantage is that no additional cost is imposed on the packet transport layer. On the other hand, this *error concealment* puts requirements on the ability of the application to handle error situations.

2.3 Efficiency issues

Since the most viable solution seems to be the NAK-based strategy, several researchers have aimed at improving on the drawbacks of NAK-based reliable multicast. The NAK implosion problem can be handled by using topology, i.e. to arrange the receivers into a tree also at the transport level. The NAK's need be sent only to the nearest "parent". These in turn, will send a NAK back to the next upper level, and ultimately, the sender will receive just one NAK from each branch.

Still, the sender will be responsible for re-sending requested packets to the multicast group. This is called source-based recovery, and it puts a large

burden on the sender. To lessen this work, the hierarchy of receivers can be used 3. Among the receivers, or routers, some can be dedicated to error recovery and thus the NAK's need be sent only to these nodes using TTL scoping. The same nodes are responsible for repair actions and the NAK's never reach the original sender. Such local repairs can hide most of the packet losses from the sender.

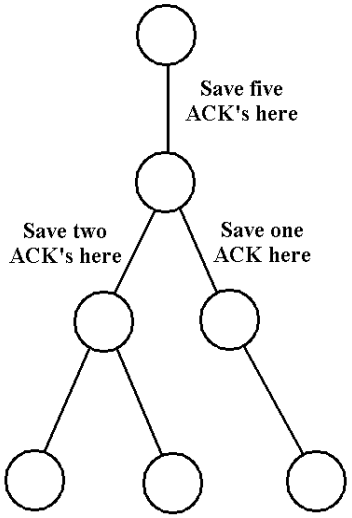

Figure 3: Saving ACK's by hierarchy.

Multicasting NAK's and repair actions seem to be an efficient way of conveying information, but this is not entirely true. If only a few receivers are located on lossy links, they will generate an abundance of NAK and repair messages, which will just consume resources at other problem-free receivers. This has been referred to as the *crying baby problem* [11].

Kasera et al. [12] compare server-based with receiver-based local recovery, and conclude that local receiver-based recovery performs well, but that server-based recovery performs even better. In server-based recovery, dedicated repair servers, co-located with routers in the network, take on the task of providing repair actions. The servers themselves are provided the same service from higher-order servers. The efficiency is due to that the repair servers can be equipped with fast CPU's and more memory, than can the average receiver. This is very similar to the Log-based receiver-reliable multicast proposed by Holbrook et al. in [11].

In [13], Li and Cheriton draws the conclusion from simulations that the

better the protocol is in itself, the less it will benefit from an FEC scheme.

Ring based protocols can be efficient and provide total ordering of packets to all receivers. They suffer from the fact that the single token holder must handle all NAK's. This can be avoided though, as shown by Gu and Garcia-Luna-Aceves [14]. Their scheme allows nodes other than the token holder to share the NAK handling load, thus balancing the stress among the ring members.

Layering has been used previously in multicast transmissions in order to cater for receivers of different capability. All links are not equal, and some links can offer substantially lower throughput than others. By splitting the data stream into a number of layers, each presenting all of the contents, but only the most crude information resolution in the base layer, and successive refinements in the higher layers, receivers can decide how many layers they are able to receive before congestion and packet loss appear. Thus, each receiver can digest an optimal amount of data and present to the user, the best information quality obtainable.

By instead applying this technique to error control, Rhee et al. [15] put FEC repair packets in a layered structure. Thus, a receiver that experiences only a minor packet loss subscribes to the base layer and the first order of repair layer, while a receiver on a lossy link, subscribes to a number of repair packets. Since, like in the traditional layering scheme, additional streams flow only where used, layered repair packets flow only towards links that are lossy. There might be a problem, though, in that the links that get the most layers, and thus, the highest number of packets, are the ones that are most likely to loose them.

The Adaptive Reliable Multicast, ARM, proposed by Yoon et al. [16], combines the use of retransmissions with the use of redundancy. In ARM, the sender emits *probe* packets in order to poll the receivers for a figure of the perceived loss. These figures are then used to dynamically adjust the amount of redundant information that is multicast by the sender.

Nonnenmacher et al. [17] also discuss using a combination of data repetition and proactive redundancy.

It is possible to distinguish between two types of ARQ-FEC hybrids, depending on whether parity data is transmitted already with the original transmission or only on request from a receiver. The former is sometimes labelled *hybrid ARQ type I*, while the latter is called *hybrid ARQ type II*.

3 Recent developments

Not one single technique will prove to solve all the complex problems of reliable multicast. Hybrid protocols for reliable multicast have used combinations of different error correcting schemes, such as combining repeating protocols with redundancy. In RMX, Chawate et al. [18] takes this a step further by combining best-effort UDP multicast with unicast TCP traffic.

RMX organises the receivers in a tree-like group structure, and heading each group is a reliable multicast proxy. Between these RMX'es the traffic is unicast over TCP, and is, therefore, already reliable.

RMX is an ALF implementation and includes application specific modules which enables an RMX to e.g. transform the information in order to fit it in the available downstream bandwidth. This part of RMX is not so much error handling per se, as adaptation to the environment.

This is not the first time that this kind of hybrid has been tried. In [19] Barbeau presents SCE, Single Connection Emulation, where an SCE layer is placed between IP multicast and a "user level TCP implementation".

Recently, new work on ACK-based protocols has appeared. In the work presented by Byung-Won et al. [20] an ACK-based hierarchical protocol is suggested. Notably, such sender-initiated protocols are applied to mobile or cellular networks, in which the node hierarchy comes very natural as e.g. a cellular base station becomes a designated receiver for the underlying subgroup of receivers (i.e. mobile terminals), connected via an air interface. The number of receivers of each such sub-group is small. The chance for an ACK implosion is therefore relatively small, as other parameters dictate a limited number of members in each subgroup.

Byung et al. expect multicast groups to be very large. They use a three-tier architecture in which they foresee a possible implosion problem, which they propose to handle by allowing both layers above the mobile terminal to alleviate the ACK's. Contrary to non-cellular scenarios though, the number of mobile terminals i.e. the size of the multicast group, will be fairly low.

In some applications, a message can render a previous message obsolete. This occurs e.g. when there is an update of some information contained in the payload. Previously, exploring this kind of semantic has been restricted to application level. It was recently proposed by Pereira et al. [21], that this could be applied also in the transport layer semantics.

Pereira et al. propose that as buffer occupancy reaches a high-water mark, a constructed semantic in the packet header becomes active. Each packet header contains information pertaining to which earlier packets it renders obsolete. This can be used to purge packets from receiver buffers. It could well be used to clean up sender or repair server buffers. Also, during conges-

tion, receivers may drop all packets that do not render any packet already in the buffer, obsolete. During normal, i.e. not congested, operation, this mechanism is not active.

This kind of solution is generic regarding the receiving end. At the sender, though, the construction of the obsolescence information if clearly application dependent.

The thought of using obsolescence has also been proposed by e.g. Rodrigues et al. [22] who use message deadlines as a criteria, and Baldoni et al. [23] who assign a lifetime to each packet. When a packet reaches a deadline or the lifetime has expired, it can be purged, possibly to be replaced by a subsequent packet.

These obsolescence techniques and the efficiency of them, all depend on application semantics. They also rely on the application to provide the metrics.

The IETF working group on reliable multicast transport (RMT) [24] is aiming at standardise reliable multicast. Recognising the problems with a *one-size-fits-all* solution, the RMT WG is going to present three protocols, a NAK based, a tree-based ACK, and an asynchronous layered coding protocol. This is currently all work in progress.

In RMT, protocols are broken down into building blocks (BB) and protocol implementations (PI). The BB's describe algorithms and interfaces towards other BB's and PI's. A PI, on the other hand, constitutes a set of BB's and other functions which make up a complete instance of a protocol.

The NAK based protocol, NORM [25] will incorporate means for NAK avoidance and dynamically adjustable timers. It will cater for measurement of the greatest round trip time for adjustment of protocol state and back-off timers. There will be dedicated header field to gather such information as to let NORM adapt to changing network conditions. NORM will also allow for proactive as well as reactive FEC. Though NORM is a NAK based protocol, ACK's can also be used by explicit configuration.

The tree-based PI is TRACK [26], which supports NAK's, proactive and reactive FEC and tree-based ACK's. Via General Router Assist, local repairs is also supported. Repair Heads correspond to the dedicated receivers or repair servers found in other protocols mentioned earlier.

TRACK has also a resemblance of the obsolescence property proposed in [22] in its sender-controlled recovery window. In each data packet, the sender may indicate that packets older than the given sequence number, are obsolete.

The layered protocol instantiation, LCT [27] the layering supports scalability with regards to link and host capacity, as well as congestion control and error recovery. As in [15] a receiver can obtain additional parity packets

by subscribing to more layers, which can result either in a faster decoding on account of a faster reception of a required number of packets, or an error recovery by acquiring a sufficient amount of parity information.

4 Conclusions and future work

There is currently a limited set of error handling techniques available. The missing data can simply be repeated, or parity information can be transmitted proactive or reactive. These methods are most often combined with NAK avoidance and grouping schemes. Such schemes can limit the number of messages that are otherwise generated in the error handling process by probing either the sender or the receivers. They will thus lessen the CPU and memory buffer burden of the corrective measures.

ALF has become a popular design principle in which the application assumes responsibilities to handle congestion and to take corrective measures in case or errors. The application has the advantage of knowing the payload semantics which can result in more options to error handling. On the other hand, it puts a larger burden on the application. The solutions for error handling also tends to be application specific.

There should be room for improvements here. Taking the data semantics into account has been proposed in e.g. RMX. But RMX is an ALF protocol and as such it involves the application in the error handling process. Agent technology seems not to have been explored, other than as a payload for existing protocols.

It is known that loss patterns are more or less correlated depending on where they occur. It should be possible to exploit such correlation patterns and to construct semantic measures from them. Using the inherent semantics in the payload of the multicast packets, possibly pointed out by information in the headers, or, constructing a semantic in the first place, might be an interesting way in introduce a new error handling element into reliable multicast without creating a need for knowledge about the application semantics.

In the end, it might be that the error need not be handled at all at one given instance, and that the problem should be dropped altogether. This is something that a transport layer semantic, be it constructed or analysed, possibly combined with loss correlation patterns, could be the answer to.

Acknowledgements

This work is supported by Ericsson Erisoft AB, Centre for Distance-spanning Technology, the Swedish Research Institute for Information Technology, and Internet 3.

References

[1] D. Clark and D. Tennenhouse, "Architectural considerations for a new generation of protocols," in *Proceedings of SigComm*, Philadelphia,PA, 1990, pp. 201–208.

[2] Byers J W, Luby M, Mitzenmacher M, and Rege A, "A digital fountain approach to reliable distribution of bulk data," *Computer-Communication-Review*, vol. 28, no. 4, pp. 56–67, 1998.

[3] G. Carle and E. Biersack, "Survey of error recovery techniques for ip-based audio-visual multicast applications," *IEEE Network*, Dec. 1997.

[4] Obrazcka Katia, "Multicast transport protocols: a survey and taxonomy," *IEEE Communications Magazine*, vol. 36, no. 1, pp. 94–102, Jan. 1998.

[5] Martin S. Lacher, Jörg Nonnenmacher, and Ernst W. Biersack, "Performance comparison of centralized versus distributed error recovery for reliable multicast," *IEEE/ACM Trans. Networking*, vol. 8, no. 2, pp. 224, April 2000.

[6] Lin JC and Paul S, "Rmtp: a reliable multicast transport protocol," *Proceedings of IEEE INFOCOM '96*, vol. 3, pp. 1414–24, 1996.

[7] Floyd S, Jacobson V, Liu C G, McCanne S, and Zhang L, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM-Transactions-on-Networking*, vol. 5, no. 6, pp. 784–803, Dec. 1997.

[8] Levine BN, Lavo DB, and Garcia Luna Aceves JJ, "The case for reliable concurrent multicasting using shared ack trees," *Proceedings of 4th ACM Multimedia Conference*, pp. 365–76, 1996.

[9] Atwood JW, Catrina O, Fenton J, and Strayer WT, "Reliable multicasting in the xpress transport protocol," *Proceedings of LCN - 21st Annual Conference on Local Computer Networks*, pp. 202–11, 1996.

[10] Johnstone GS and Williams GD, "Reliable multicast under error conditions," *Proceedings of 22nd Annual Conference on Local Computer Networks*, pp. 379–86, 1997.

[11] Holbrook HW, Singhal SK, and Cheriton DR, "Log-based receiver-reliable multicast for distributed interactive simulation," *Computer-Communication-Review*, vol. 25, no. 4, pp. 328–41, Oct. 1995.

[12] Kasera SK, Kurose J, and Towsley D, "A comparison of server-based and receiver-based local recovery approaches for scalable reliable multicast," *Proceedings IEEE INFOCOM'98 Conference on Computer Communications Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies Gateway to the 21st Century*, vol. 3, pp. 988–95, 1998.

[13] Li D and Cheriton DR, "Evaluating the utility of fec with reliable multicast," *Proceedings of ICNP'99: 7th International Conference on Network Protocols*, pp. 97–105, 1999.

[14] Lifan Gu and Garcia-Luna-Aceves JJ, "New error recovery structures for reliable multicasting," *Proceedings of Sixth International Conference on Computer Communications and Networks*, pp. 189–96, 1997.

[15] Rhee I, Joshi SR, Lee M, Muthukrishnan S, and Ozdemir V, "Layered multicast recovery," *Proceedings IEEE INFOCOM 2000*, vol. 2, pp. 805–13, 2000.

[16] Yoon J, Bestavros A, and Matta I, "Adaptive reliable multicast," *Proceedings of IEEE International Conference on Communications*, vol. 3, pp. 1542–6, 2000.

[17] Nonnenmacher J, Biersack EW, and Towsley D, "Parity-based loss recovery for reliable multicast transmission," *IEEE/ACM-Transactions-on-Networking*, vol. 6, no. 4, pp. 349–61, 1998.

[18] Chawathe Y, McCanne S, and Brewer EA, "Rmx: reliable multicast for heterogeneous networks," *Proceedings IEEE INFOCOM 2000*, vol. 2, pp. 795–804, 2000.

[19] Barbeau M, "Implementation of two approaches for the reliable multicast of mobile agents over wireless networks," *Proceedings of 1999 International Symposium on Parallel Architecture, Algorithms and Networks*, pp. 414–19, 1999.

[20] Byung Won On, Haesun Shin, Miae Choi, and Myong Soon Park, "A hierarchical ack-based protocol for reliable multicast in mobile networks," *Proceedings of IEEE ICON International Conference on Networks*, pp. 359–62, 2000.

[21] Pereira J, Rodrigues L, and Oliveira R, "Semantically reliable multicast protocols," *Proceedings of 19th IEEE Symposium on Reliable Distributed Systems*, pp. 60–9, 2000.

[22] Rodrigues R., Baldoni R., Anceaume E., and Raynal M., "Deadline-constrained causal order," *The 3rd IEEE International Symposium on Object-oriented Real-time distributed Computing*, Mar. 2000.

[23] Baldoni R., Prakash R., Raynal M, and Singhal M., "Efficient Δ-causal broadcasting," *Journal of Computer System Science and Engineering*, 1998.

[24] IETF Working Group for Reliable Multicast Transport (rmt), "Charter page," URL[1], Visited 2001-01-12.

[25] Adamson B., Bormann C., Floyd S., Handley M., and Macker J., "Nack-oriented reliable multicast protocol (norm)," IETF Internet Draft, November 2000, draft-ietf-rmt-pi-norm-00.txt, Work in progress.

[26] Whetten B., Chiu D., Paul S., Kadansky M., and Taskale G., "Track architecture, a scaleable real-time reliable multicast protocol," IETF Internet Draft, July 2000, draft-ietf-rmt-track-arch-00.txt, Work in progress.

[27] Luby M., Gemmell J., Vicisano L., Rizzo L., Handley M., and Crowcroft J., "Layered coding transport: A massively scalable multicast protocol," IETF Internet Draft, November 2000, draft-ietf-rmt-bb-lct-00.txt, Work in progress.

[1] <http://www.ietf.org/html.charters/rmt-charter.html>