A Configurable Transport Layer as a Cure for Crying Babies

Stefan Elf, 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

Abstract— **Group collaboration and media distribution applications have attracted more interest during recent years. With an increasing attention to wireless environments and business applications there is a growing need for reliable multicast communication protocols.**

There have been mainly two approaches, proactive or reactive, to error handling in reliable multicast. Proactive protocols transmit redundant information along with the original information, enabling the receivers to repair lost packets without the necessity of feedback to the sender. Reactive protocols rely either on positive or negative feedback from the receivers.

A case of special interest in relation to wireless connectivity is where essentially one receiver on a lossy link eventually will consume most of the error handling resources. This problem has been addressed by some researchers, e.g. by the use of probabilistic methods.

We propose to view reliable multicast as a special case of semantically reliable multicast. We propose to make use of the application's knowledge of its specific semantics to improve e.g. on the problem with single, lossy, links. We suggest the design of a semantic rule set for the transport layer protocol, which can be configured by the application. Using application semantics would allow the application to relax the reliability requirement at the transport level.

Keywords— **multicast, reliable multicast, error handling, semantic, dynamic, configure**

I. Introduction

It is not until very recently that the Internet has been available to a general public. As more homes are connected by cable or stable radio links, Internet business cases become attractive. Where there is business we need security and security requires reliability.

As a result of people being commonly connected, there is a growing interest for establishing a mobile computing platform, enabling mobile connectivity. Parallel to a growing wired network, we will therefore also see a growing wireless infrastructure with all its implications. In this new setting the location of hosts will not be static and links will be established and broken depending on the user's location changes and the possible shortcomings of the wireless infrastructure. As the computing platforms shrink in size, users will become more mobile and then expect to be able to stay on line while on the move. Links to wireless hosts will have a much less stable bandwidth than we have been used to from wired links.

Capabilities for reliability are mainly implemented in the transport layer. A number of reliable multicast protocols have been proposed over the years, each adapted towards a specific problem area. A consequence of the conflict between the need for a collected view of reliable multicast services and the very different requirements and possibilities offered by various applications, is drawn in the creation of the Application Level Framing principle, proposed by Floyd et al. [1]. ALF has consequently grown in popularity during recent years, shown e.g. by Chawathe et al. in the RMX protocol [2], which operates in the transport layer as well as in the application layer.

Making a multicast protocol reliable is one challenge. But at the same time, the protocol must work efficiently not only for two users, but for hundreds or thousands, depending on the application. The bandwidth can be exhausted by re-transmissions as well as by control message implosions. Single, failing, receivers on lossy links, must not disrupt the session for other, more stable, hosts, by consuming most of the resources for error correction. This latter phenomenon has been described as the *crying baby problem* [3] and it is even more pronounced in a scenario with a mix of wired and wireless hosts.

Various methods have been used in order to lighten the load from the sender or to share the load among the receivers, depending on whether a sender or receiver based strategy has been chosen. It would seem that, at least in scenarios where the number of receivers is fairly high, the tree-based receiver oriented approach, has been the most successful, both in terms of sharing load, improving on locality of repair, and on overall protocol efficiency.

In a tree-based protocol, the remedy to an inhomogeneous group is to split the group. But if the group contains one crying baby, splitting the group will not help, since the receiver on the lossy link will obviously belong to one of the resulting groups. There have been probabilistic approaches to load sharing and scalability, which also lead to a remedy to this problem, such as proposed by Xiao and Birman [4].

Application level framing obviously will enable error handling according to application specific semantics. This will allow for error concealment strategies where the transport layer would simply ignore lost packets guided by suitable

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

constraints and the application would conceal the lost information.

By allowing for the application to configure the transport layer protocol, we propose to use the best of both worlds, namely to combine the knowledge of the semantics from the application layer with the speed of processing of the transport layer. This will allow us to create a service that is "semantically reliable", where the reliability will be put in the context of application semantics. We need not recreate or request a repair of a packet that is not necessary for the application to fulfil its task.

The remainder of the paper is structured as follows. Current error handling methods are reviewed in section II. Next, in section III we propose and discuss a framework for a configurable transport layer protocol. The concept of semantic reliability is expanded on in section IV. Section V presents related work. Lastly, in section VI we conclude.

II. Multicast Error Handling

In this section, the current methods of error handling in reliable multicast protocols will be reviewed.

Essentially, reliable multicast protocols are all either mainly reactive or proactive 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 concurrently to the original data transfer. The reactive protocols are either ACK-based or NAK-based. This simply implies where the responsibility for detection of packet loss lies. ACK-based protocols leaves this responsibility with the sender, while NAK-based protocols share the responsibility among all the receivers.

ACK-based protocols have an advantage in that the sender will always know when a packet can be discarded and the buffer space reclaimed. On the other hand, they suffer from the drawback that non-ACK'ed packets must be buffered by the sender and that the sender must keep state information for each receiver. Therefore this solution does not scale as the number of receivers grows. The amount of resources demanded by the sender will eventually become too high, or the number of ACK's sent by the set of receivers will exhaust the network or sender host resources and create an *ACK implosion*.

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, but if there is a large number of receivers who experience packet loss of a magnitude, the sender will instead face a possible *NAK implosion*.

To avoid the implosion effect and to boost efficiency, a tree structure is often used. The tree structure enables reduction of the number of control messages and caters for a locality of repair.

Packet loss is detected by the use of timers. A NAK is sent when the timer expires. The expiry time is can be a function of the receiver's distance from the sender, or a random value. The Scalable Reliable Multicast [5] uses a combination of these methods to select the timer value. The receivers that did not send a NAK, will reset their own timers when then sense the NAK multicast by the "winning" receiver. Of course, repair data will also be multicast to the whole group.

Repeating packets either on loss detection at the sender or on request from the receivers generates excessive traffic that increases with the amount of loss. If the cause for the loss is congestion, this would seem to be somewhat counter-productive. 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 feedback to the sender. The methods used include erasure codes and forward error correction schemes.

The Digital fountain approach using a special application of efficient erasure codes called *tornado codes* is discussed by Byers et al. [6]. The Digital Fountain can be especially effective in e.g. distribution of predetermined amounts of information, since there is no need for feedback at all. A receiver that looses a packet will just have to wait a while longer in order to obtain *any* few more packets to be able to reconstruct the missing ones.

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 this is not always suitable for streaming media. The blocking will also introduce delays in the data delivery.

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 or creating a favourable topology. Among the receivers some can be dedicated to error recovery and thus the receiver 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.

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, Chawathe et al. [2] takes this a step further by combining multicast with unicast TCP traffic.

It would seem that the crying baby problem is partly addressed by tree-based systems, by forward error correction methods, and by probabilistic load-sharing systems.

III. A cure for crying babies

Handling crying babies without loading other hosts in the network requires that traffic to other receivers not be impaired and that proxies are not overloaded.

In order to fulfil this, there must be a low amount of feedback information generated per detected loss and servicing host. This can be accomplished by e.g. splitting groups to lessen the probability of there being a crying baby in the

group, or by enabling hosts to share the load among themselves. This seems not to be a widely addressed problem.

In this section, we propose a method for error handling especially in inhomogeneous environments in reliable multicast protocols, which should be possible to implement on top of some existing multicast transport protocol.

When applying ALF principles, it is understood that the error handling should be done in the application. The advantage with this is that the application knows the semantics of the application data, which are essentially unknown to the transport layer where the information appears as a packet payload. The drawback is that handling errors in the application can consume considerable resources such as buffer space and CPU power. If we are considering a wireless scenario also, this can be constraining properties.

One way to handle this problem is to be efficient in the application. There is always a tradeoff between having access to more information and being able to process faster, as you move down a protocol stack. We speculate that an optimal way to proceed would be to define, based on received data and knowledge of application semantics, an error handling rule set in the transport layer, which can be configured by the application to handle certain packets in a specific way. Therefore, we propose to create a configurable transport layer protocol based on the notion of semantic reliability IV.

By making the transport layer protocol configurable, the transport layer will probably be differently configured at each host. The configurable parts of the transport layer protocol will not be engaged in the interactions with other hosts. They will instead serve the local application with timely decisions regarding lost packets. Since all hosts can have different semantic configurations, this implies that a crying baby need not disrupt the session for other participants.

The configuration of the transport layer could either be done dynamically, by feedback from the application, or during session start-up having a sender start the session by uploading transport layer configuration parameters for the session. These are recognised by the transport layer directly as a dedicated packet type. The parameters can also be recognised by the application and returned by the application to the transport layer.

Receivers who join a session late and wish to update their session configuration can do so by multicasting a dedicated message to its neighbours. The session configuration is then multicast to the same scope. It is to be noted that receivers who do not have the session parameters, can still receive from and transmit to the group. This configuration would not be a prerequisite, but rather an enhancement that can be used with any protocol.

One example of a session configuration is the implementation of a deadline constraint, which can be described by the following scenario.

In an audio application it is not essential that all packets be delivered to the application. In fact, if the audio application has lost a packet, it can sometimes simply replay the last received packet instead of the lost one, without any serious deterioration to the output signal. But the transport layer would not know of this, and if there were packet losses, the transport layer would request lost packets, even if the application could decide that they were not necessary.

According to our proposal, the application would be able to configure the "deadline" semantic of the transport layer. If, at time t_0 the packet with a time stamp t_p was to be played out by the application, and if a packet must be placed in the playout buffer a time t_l before it is going to be played out, the application can configure the constraints $t_0 - t_p$ and t_l in the transport layer protocol. Each time a packet is received in the current stream, the transport layer would decide if for the current packet $t - (t_0 - t_p) - t_l < 0$. Were that the case, the packet would simply be discarded. The configuration parameters can preferably be updated by the application during run-time.

In a shared white-board application most of the time it is imperative that packets are not lost - but not always! Imagine that a user draws a circle and another user does not receive the circle-packet. But before this error can be corrected, the first user deletes the circle. Now, there is no meaning in re-sending the original circle-packet to receiving user. The sender can obviously detect this situation, and remove the original circle-packet from its buffer. Prior to sending this packet, it is marked as rendering the circle-packet obsolete and redundant. The receiver who is still missing both packets can, upon reception of the circle-delete-packet, remove the request for the originally lost packet, and drop the circle-delete-packet. Neither packets need ever reach the application.

This behaviour is the result of use of application specific semantics, which are downloaded to a configurable transport layer protocol at the start of the session.

IV. Semantic Reliability

This proposal is based on the concept of semantic reliability, which has been exemplified above. If we accept that a reliable transmission must not necessary imply that all packets reach their destination, but that the notion of reliability must be put in the context of the application semantics, we can begin manipulating with the receiving queues by adjusting the behaviour of the transport layer protocol.

The configuration information that is either distributed in a dedicated packet type, or downloaded from the application, contains a meta-semantic which describes the type of semantic to be used, and possible parameters. Which kind of semantic this would specifically be, depends on the application in question. A few examples (obviously, some of these have already been explored by other authors, as described in the following section) follow

- **ABSOLUTE** Every single packet must be received. Configured by a flag.
- **OBSOLESCENCE** The sending application will provide information as to which packets the current packet will render obsolete. Configured by listing packet sequence

numbers.

- **REDUNDANCY** The sending application provides information as to whether this information will make older information redundant. Configured by listing packet sequence numbers.
- **LIFETIME** The sending application supplies a measure of life time for each packet. Configured by a time constraint value.
- **EXPENDABLE** The packets that are marked expendable by the sending application can be dropped without hazard for the application semantics. The receiving application can also provide this information. Configured by a flag.
- **DEADLINE** If the packet arrives after the provided deadline, it must be discarded. This kind of deadline can be provided either by the sender or by the receiver. Configured by a time limit value.

More semantic measures can be included depending on the application. For each semantic, there will also be a set of parameters provided in the configuration, which will be used by the transport layer protocol.

V. Related work

In section I a brief survey of the closest related work is given. According to this, errors are handled either in the transport layer, where they are subject to re-sending original data upon detection of packet loss. There are various techniques to reduce the number of control messages by using hierarchic models or by multicasting control information and/or repair information. The RRMP proposed by Xiao and Birman [4] can also handle inhomogeneous environments, since it shares the burden among all receivers in a local group by using a probabilistic approach. This work is related to our proposal in the sense that it also solves the crying baby problem that this paper addresses.

The work presented by Pereira et al. [7], Rodrigues et al. [8], and Baldoni et al. [9], comes closest to our proposal.

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 congestion, 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 with regards to the receiving end. At the sender, though, the construction of the obsolescence information if clearly application dependent.

Rodrigues et al. use message deadlines as a criteria and Baldoni et al. 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 semantic to the transport layer.

Our proposal differs from the referenced work, in that we aim to create a configurable transport layer protocol in which the sender side application can feed information to the transport layer, which can be interpreted at the receiver side, and in which the receiving application can help the transport layer to adapt to the injected semantic as well as to payload properties, as interpreted by the application. At the receiver side, the deployment of error handling can thus be distributed between the application and the transport layer in that the application will enable a transport layer activity.

VI. Conclusions and future work

A reliable multicast protocol most often means that no packet must be lost. This is accomplished either by repeating the information on request, or by sending parity data along with the original information. These methods are most often combined with NAK avoidance schemes. Such schemes can limit the number of messages that are generated in the error handling process by probing either the sender or the receivers, thus enabling the error handling schemes and lessen the burden of the corrective measures. By building trees for control or corrective information, inhomogeneities can be handles to a certain extent. Probabilistic approaches handle this even better.

It would seem that reliability can be a relative expression, which can most favourably be interpreted in the context of the application semantics. This implies that in some contexts, the requirement for reliability is that every single packet be received. In other contexts, this might not be the case, and the absolute reliability becomes a special case of a "semantic reliability".

In order to be able to use the application specific semantics, the application level framing principle will be explored. As knowledge about the application semantics is present in the application layer, there is also where the processing is most expensive. Error handling is more efficient in the transport layer, where the application semantics unfortunately is not present.

By striving to handle the errors at the transport layer, but configure and tune the protocol from the application, we distribute the computing burden which on one hand relieves the application, and on the other hand caters for a more effective protocol, since each decision will be handled at a lower layer.

It is possible that this technique can be applied also to congestion control.

The most imminent work is now to implement a scheme according to the proposal in a network simulator. Based on the results from the simulations, a live implementation will be made.

Acknowledgements

This work is supported by Ericsson Erisoft AB and the Centre for Distance-spanning Technology.

References

[1] Floyd S, Jacobson V, McCanne S, Liu C.G., and Zhang L., "A reliable multicast framework for light-weight sessions and application level framing," *Computer-Communication-Review*, vol. 25, no. 4, pp. 342–56, Oct. 1995.

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

[3] 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.

[4] Xiao Z. and Birman K.P., "A randomized error recovery algorithm for reliable multicast," *Proceedings IEEE INFOCOM 2001*, April 2001.

[5] 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.

[6] 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.

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

[8] 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.

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