Applying Semantic Reliability Concepts to Multicast Information Messaging in Wireless Networks

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

ABSTRACT

Public use of the Internet increases and applications suitable for multicast distribution grow more popular. An increasing percentage of users require wireless connectivity. Multicast solutions must therefore be able to handle receiver and network link heterogeneity. This paper proposes an approach to handle such challenges.

According to a definition of *semantic reliability*, the reliability concept can be interpreted in terms of application semantics. By using semantic reliability traditional reliability constraints can be relaxed. Our approach is to use an application – transport layer communication to implement a dynamically configurable transport layer protocol whose error-handling rule set can be configured from the application or even from the sender in-session. It can also be initiated from the sender when the session starts.

INTRODUCTION

Protocols for mass-distribution of information have been researched during the last two decades. Although multicast scenarios mostly involve one-to-many situations, such as lecturing or advertising, there is also a growing class of group collaboration applications, which operate on a many-to-many basis.

Networking environments have until recently been more or less static. This situation is now changing as wireless network access is gaining public interest and availability. With the freedom of wireless connectivity comes a number of problems among which the most notable are that multicast trees will need to change more often as the user moves, that link capacity will fluctuate due to the nature of radio propagation, that user equipment will have restricted capacity both in terms of power and memory space.

It seems reasonable to assume that more widely available network connectivity will drive a need for many-to-many communications. Among such collaborative applications are shared network editors, on-line gaming, simulation applications, and shared whiteboards. Many of these applications require reliable services.

In MobileCity, an EU funded ongoing project in Skellefteå in northern Sweden, the International Fair Navigator is being developed. Visitors will have information about products and services, maps etc., stored in a personal digital assistant (PDA). The PDA will also receive information via multicast and users may interact. Since PDA's are connected with a wireless interface and users are highly mobile, they are subject to varying bandwidth and frequent disconnections (Figure 1). A reliable service is required to ensure content delivery. According to our approach, it is possible that reliability can be achieved even though all information does not reach the PDA.

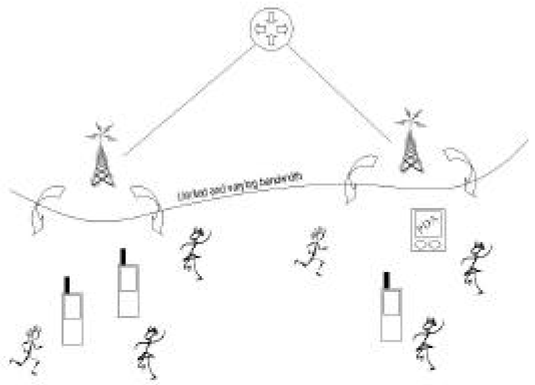

Figure 1. In a wireless scenario, excess traffic must be avoided.

A number of reliable multicast protocols have been designed during recent years in which commonly, the transport layer handles reliable services. Each protocol tends to address specific application requirements. Application Level Framing (ALF), first proposed by Clark and Tennenhouse (Clark DD & Tennenhouse DL, 1990), puts the responsibility with the application itself to handle end-to-end reliability. The advantage is that the application can adapt gracefully, while the drawback is that this must be implemented specifically in each application. ALF has grown in popularity shown e.g. by Chawathe et al. in RMX (Chawathe Y, McCanne S, & Brewer EA, 2000), which operates in the transport layer as well as in the application layer.

It is required that a reliable multicast protocol be scalable regardless of whether errors appear or not. It must be efficient for two users or for two thousand users alike. Control message traffic, delay, jitter, and buffer requirements must not cause the protocol to use most of the available bandwidth for control data.

Topological features such as hierarchic levelling, grouping, and clustering have been explored in order to handle the flow of control messages. Other approaches include use of multicasting control messages using multiple multicast groups to layer control message information.

Protocols handle errors either by detecting information loss and re-sending lost data or by transmitting redundant or parity information with the original data. Some transmit parity information only when a loss is detected. If repair packets are multicast to the group, each packet may repair several lost packets for different receivers.

The ALF approach will allow the application to handle most of these problems. ALF enables error concealment strategies. The drawback is that effectively a new protocol must be developed for each application.

We believe that a well-defined dynamic behaviour in the transport layer, configured by application level information, will enable it to use application semantic properties. We will build on previous work on semantic reliability and attempt to find a straightforward way to formalize the implementation in an application independent way.

The remainder of the paper is structured as follows. Next, we present related work. Current error handling methods are then shortly reviewed. Following, we present a relaxed notion of reliability and discuss a configurable transport layer protocol as a way to implement a semantic reliability. Last, we conclude.

RELATED WORK

Mauve and Hilt (Mauve M & Hilt V, 2000), discuss an application-programming interface (API) for reliable multicast supporting distributed interactive multimedia applications. Their approach is close to ours. They propose a subscription-oriented interface in which a sending application controls the use of forward error correction (FEC). The receiving application can specify its (dis-)interest in lost packets. These Quality of Service (QoS) levels can relieve the receiver from having to buffer packets that must otherwise be preserved due to outstanding earlier packets.

The work presented by Pereira et al. (Pereira J, Rodrigues L, & Oliveira R, 2000), Rodrigues et al. (Rodrigues L, Baldoni R, Anceaume E, & Raynal M, 2000), and Baldoni et al. (Baldoni R., Prakash R., & Raynal M./Singhal M., 1998), is also close to our approach. They explore application level semantic included in the packets.

Pereira et al. (Pereira J et al., 2000) 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. Packets can thus be purged from receiver buffers. Sender or repair server buffers can also be cleaned up using this approach. During congestion receivers may drop all packets that do obsolete packets already in the buffer. During nominal operation this mechanism is inactive. The sender is required to mark all packets that may be discarded, as obsolete.

Rodrigues et al. (Rodrigues L et al., 2000) use message deadlines as a criteria and Baldoni et al. (Baldoni R. et al., 1998) assign a lifetime to each packet. It can be purged when it reaches a deadline or its lifetime has expired.

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

Our approach differs from the referenced work in that we act at the transport layer and will create a “tunnel” between the application and transport layers. This allows the sender application to feed information to configure the receiving side transport layer. The receiving application will be able to configure the transport layer based on its knowledge of incoming payload. The semantics used are associated with application behaviour rather than QoS levels. This will decouple the specific application’s subjective involvement from lower layer error handling.

Where nodes are connected wireless to a fixed network or central server, it is also important that configuration actions not only be restricted to the local host, but to the nearest upstream router or server. This will assure that the least amount of traffic will pass the air interface.

ERROR HANDLING

Reliable multicast protocols are either reactive or proactive. Reactive protocols must have a mechanism to detect packet loss. Sender oriented protocols require receivers to acknowledge (ACK) receipt of each packet as in RMTP (Lin JC & Paul S, 1996). With a large receiver group the sender could risk being swamped by ACK's from all receivers resulting in an *ACK implosion*. The sender must keep state for all receivers and must buffer transmitted packets until all receivers have ACK'ed them.

In a hierarchy the number of control messages reaching the sender is lower. Each ACK needs to travel only to the next higher level in the tree. The host that receives the ACK's issues repair packets for the underlying sub-tree, which further lowers traffic in the multicast group and allows the sender to share the load with local repair servers.

Pure ACK-based protocols are not scalable but have lately attracted interest in wireless scenarios where the number of receivers is not very large, one example being the work by Byung et al. (Byung Won On, Haesun Shin, Miae Choi, & Myong Soon Park, 2000). Another example is the `pgmcc` (Rizzo L, 2000), wherein the worst-case receiver is appointed representative of its group, the *acker*.

To handle the scalability problem negative acknowledgements (NAK) are used. As each receiver is responsible for detecting its own lost packets, there will not be any significant load on the sender and the result is improved scalability. In NAK based protocols lost packets are detected using time-outs and message sequence numbers. Lost trailing packets are detected with *session messages* issued by a receiver announcing the currently largest packet sequence number.

NAK implosion may result from a large number of receivers experiencing simultaneous loss. In Scalable Reliable Multicast (Floyd S, Jacobson V, McCanne S, Ching Gung Liu, & Lixia Zhang, 1995) a scheme using timers causes a receiver to back off before a NAK is sent, in hope for another receiver to multicast a corresponding NAK. Multicast of control messages and repairs increases efficiency and lowers the chance for an implosion.

Hierarchy, grouping, or clustering of hosts increases efficiency in NAK-based protocols as well as in ACK-based.

In a proactive approach redundant information is transmitted with original data. Ideal proactive protocols require no feedback, which can be true in a real life situation only up to a certain level of packet loss. The cost for this convenience is a constantly higher data rate. A possible need for blocking data will introduce delays in the data transmission.

The Digital fountain approach discussed by Byers et al. (Byers J W, Luby M, Mitzenmacher M, & Rege A, 1998) is a special application of efficient erasure codes. With these *tornado codes* a receiver need only receive *any* subset of packets among those sent, in order to be able to decode original data. If the sender keeps transmitting packets, there is no need for feedback at all. The penalty is again a slightly increased bandwidth demand.

In practice, proactive protocols must be combined with reactive. Hybrid protocols for reliable multicast include RMX (Chawathe Y et al., 2000), where UDP multicast is combined with unicast TCP traffic.

Table 1 summarises some challenges, which must be met by multicast protocols in general.

Table 1. Multicast protocol challenges

Issue	**Remedy**	**Affected by semantic reliability**
Receiver heterogeneity	Grouping, hierarchy, worst case	Yes
Network heterogeneity	Grouping, hierarchy, worst case	Yes
Scalability	Grouping, hierarchy	Possibly
Congestion	TCP-like schemes	Possibly
Control message implosion	Hierarchy	Possibly
Application independence	-	Yes

RELAXING RELIABILITY

Applying semantics is not sufficient to implement a semantic reliability concept. We need a framework that explores semantics and decouples the application from actual packet handling.

Semantic Reliability

The basis for the idea of semantic reliability (Pereira J et al., 2000) is a redefinition of reliability as such. By looking at the data from the application's perspective we can make probable that it is not necessary to receive all information for the application to perform adequately. Thus, we can view a semantically reliable transport to be a transport that is reliable in relation to application semantics.

Configuration information is either distributed in a dedicated packet type or downloaded from the application. It contains a meta-semantic describing the type of semantic used, and possible parameters. The specific kind of semantic depends on the application, as shown in Table 2.

Each semantic is related to a set of parameters describing the desired behaviour of the transport layer, with respect to each received packet. The most strict semantic is ABSOLUTE, which corresponds to unrelaxed reliability where every packet must eventually be delivered to each receiver. The least strict semantic is BEST-EFFORT – “do not re-send lost packets”. Although there seems to be a “range” from ABSOLUTE to BEST-EFFORT, this is not the case. Different semantics do not represent a scale of higher or lower QoS, but rather different aspects of application data properties.

Table 2. Example semantics and properties

Semantic	Meaning	Property	Configured by
ABSOLUTE	All packets received	None	Sender/receiver
OBSOLESCENCE	Packets made obsolete	Seq.no.	Sender
REDUNDANCY	Redundant packet	Seq.no.	Sender
LIFETIME	Packet life time	Time	Sender
EXPENDABLE	No crucial information	Boolean	Sender
DEADLINE	Latest arrival time	Time	Receiver
CATCH-UP	Disregard packets	Seq.no.	Receiver
BEST-EFFORT	Fully relaxed	None	Receiver

ABSOLUTE and BEST-EFFORT are mutually exclusive. LIFETIME and DEADLINE seem very similar, though the sender sets the former by marking packets, and the latter is determined by the receiving application. OBSOLETE packets are those whose information has been invalidated by more recent packets. EXPENDABLE packets carry information that can be discarded in the first place and need not have any relation to other packets. The CATCH-UP semantic will let a receiver disregard all packets up to a specified packet sequence number, possibly a very large amount of information. The receiver takes this decision. When the receiver has caught up, this semantic is reset.

The packet sequence number, as single numbers, lists of numbers, or “all numbers less than”, is used to identify obsolete and redundant packets.

When the sender specifies a semantic it is included as information in the packets transmitted to the receiver. The receiving application informs the transport layer of the presence of the semantic. Properties that are conveyed in the packets are presented in Table 2.

An application can always override a sender’s recommendation. Not activating a semantic constitutes a more strict behaviour. If a sender specifies ABSOLUTE, a receiver may take a more relaxed position, and specify BEST-EFFORT or in some cases, CATCH-UP, e.g. when a whiteboard page is being updated and the receiver knows that it already has a consistent view.

A Semantic Reliability Framework

There is a trade-off between efficiency of handling errors in the application according to the ALF principle, and efficiency of handling errors at a lower protocol layer, where the necessary information may not be available.

With support from the application layer, the transport layer can handle errors effectively. We propose a model by which the application defines an error-handling rule set based on received data and knowledge of application semantics. This information is then fed to the transport layer. Configurable parts of the transport layer protocol will not be engaged in interactions with other hosts. The transport layer may be differently configured at each host. Therefore, a failing host can refrain from disrupting the session for other participants.

The configuration of the transport layer is done dynamically. The sender supplies the semantics. It can also be constructed by the receiving application, depending on local conditions. When conditions for the receiving application change, the transport layer will immediately be reconfigured.

Receivers who join a session late and wish to update their dynamic transport layer configuration do so by multicasting a request message to its neighbours. The configuration is then multicast to the same scope. Receivers who do not have the semantic parameters can still receive from and transmit to the group. The configuration is not a prerequisite, but rather an enhancement, a protocol booster, which can be used with any protocol.

If we extend this framework to let the application configure the nearest upstream router’s transport layer it will be even more versatile. With wireless connectivity this is more important, since we must not send unnecessary data over a wireless link.

Semantics distributed by the application will be interpreted directly by the transport layer, which is then configured accordingly. The application will configure the transport layer using an API. This API can also be used by the application to query for sender-initiated semantics. The implementation is divided into two parts. The application level uses the first part to set up the semantics.

```
Semantic wbSemantic = new
Semantic("channelId",
SemanticProperty.BEST_EFFORT);
```

The transport level executes the semantic rules, using the other part

```
theSemantic =
theSemantic.getInstance("channelId");
if (theSemantic.isBestEffort) {
  // Do not handle lost packets
  return;
}
```

The `channelId` associates the data flow with the relevant semantics.

Some of the semantics will work properly only when the sender adapts the packet payload size to payload characteristics. In a video application, each packet should hold one or more full frames, depending on the frame type. In a stock exchange update application, each packet should convey one record of information.

Fair Navigator

The Fair Navigator scenario uses a centralized information service. People carry Fair Navigators (PDA's) with suitable software. Since these have limited storage and computing capacity, and are connected through a wireless interface, it is important that there is no excess traffic.

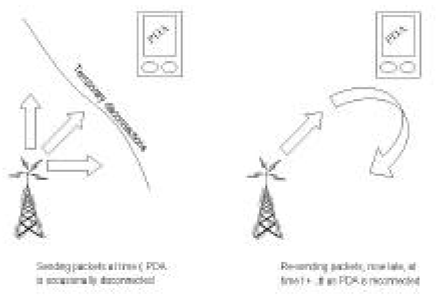
PDA
Temporary disconnection
PDA
Sending packets at time t, PDA is occasionally disconnected
Re-sending packets, now late, at time t + ..t as PDA is reconnected

Figure 2. PDA transport layer drops outdated packets.

When the information regarding a meeting is transmitted, the sender will apply the LIFETIME semantic. When the meeting is over, it makes no sense to accept the packets (Figure 2). A PDA coming on-line after a break detects expired packets and can use the CATCH-UP semantic to discard these.

If the connection degenerates, the application can use the BEST-EFFORT semantic in order to minimize the network impact and to make the application perform as good as possible.

Shared Whiteboard

In a shared whiteboard packets must not be lost. A user draws a circle, followed by other figures. Another user has lost the circle-packet. Before this error is corrected, the first user deletes the circle. Now, there is no meaning in re-sending the original circle-packet.

The sender can detect this and remove the original circle-packet from its buffer. The receiver, still missing both packets, can upon reception of the delete-packet remove the request for the originally lost packet and drop the delete-packet since it will point the circle packet out as being obsolete.

A complete update of a page can be detected by the receiver and if the page is already up to date, the receiver can CATCH-UP and drop all update packets.

CONCLUSIONS AND FURTHER WORK

In wireless connections it is important not to send data that the application can manage without. If absolute reliability is required large delays can be introduced because of the greater chance for disconnections and varying bandwidth. If a protocol must adapt to a worst-case receiver without semantic reliability in a wireless network, the better nodes will suffer badly.

Current solutions explore semantic reliability but do not devise a complete solution to the problem of reducing the number of messages, which are due to the error-handling process itself. The overall number of messages can be reduced using hierarchic models, but in the end, the wireless receiver failing to receive packets will cause problems where it hurts the most, in the wireless link.

In order to be able to use application specific semantics, we explore the application level framing principle in the sense that the application is involved in the process. The transport layer will handle packets more effectively, but the application will decide which packets can be dropped with the least concern. Our approach will enable applications to adapt to varying resources such as in the Fair Navigator example, where a number of wireless connected, "network friendly", PDA's located at weak spots will act in both in the best interest of the user and of the wireless network.

We believe that this technique can be applied also to congestion control. It will give the application some control over which packets are dropped from a queue that is full or almost full, based on application semantics.

The first implementations of this framework are planned in the Fair Navigator and is under way in the shared whiteboard of the distributed collaboration software MarratechPro (Marratech - The E-Meeting Company, 2001).

ACKNOWLEDGEMENTS

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

Stefan Elf is also with Ericsson Erisoft AB, SE-931 87 Skellefteå, Sweden. Views expressed in this paper are his own and not necessarily shared by his employer.

REFERENCE LIST

1. (Marratech - The e-meeting company [Web Page]. URL http://www.marratech.com/ [2001, October 15].

2. Baldoni R., Prakash R., & Raynal M./Singhal M. (1998). Efficient ? -causal Broadcasting. Journal of Computer System Science and Engineering .

3. Byers J W, Luby M, Mitzenmacher M, & Rege A. (1998). A digital fountain approach to reliable distribution of bulk data. Computer-Communication-Review, 28(4), 56-67.

4. Byung Won On, Haesun Shin, Miae Choi, & Myong Soon Park. (2000). A hierarchical ack-based protocol for reliable multicast in mobile networks. Proceedings of IEEE ICON International Conference on Networks (pp. 359-62).

5. Chawathe Y, McCanne S, & Brewer EA. (2000). RMX: reliable multicast for heterogeneous networks. Proceedings IEEE INFOCOM 2000 (pp. 795-804).

6. Clark DD, & Tennenhouse DL. (1990). Architectural considerations for a new generation of protocols. Computer-Communication-Review. Vol.20, No.4; Sept. 1990; P.200-8.

7. Floyd S, Jacobson V, McCanne S, Ching Gung Liu, & Lixia Zhang. (1995). A reliable multicast framework for light-weight sessions and application level framing. Computer-Communication-Review. Vol.25, No.4; Oct. 1995; P.342-56.

8. Lin JC, & Paul S. (1996). RMTP: a reliable multicast transport protocol. Proceedings of IEEE INFOCOM '96 (pp. 1414-24).

9. Mauve M, & Hilt V. (2000). An application developer's perspective on reliable multicast for distributed interactive media. Computer-Communication-Review. Vol.30, No.3; July 2000; P.28-38.

10. Pereira J, Rodrigues L, & Oliveira R. (2000). Semantically reliable multicast protocols. Proceedings of 19th IEEE Symposium on Reliable Distributed Systems (pp. 60-9).

11. Rizzo L. (2000). pgmcc: a TCP-friendly single-rate multicast congestion control scheme. Computer-Communication-Review. Vol.30, No.4; Oct. 2000; P.17-28.

12. Rodrigues L, Baldoni R, Anceaume E, & Raynal M. (2000). Deadline-constrained causal order. Proceedings Third IEEE International Symposium on Object- Oriented Real-Time Distributed Computing (ISORC 2000) (pp. 234-41).