Reliable Multicast in Mobile Networks

Pasi Tiihonen and Petri Hiirsalmi

Lappeenranta University of Technology P.O. Box 20 FIN-53851 Lappeenranta, Finland,

{Pasi Tiihonen, Petri Hiirsalmi}@lut.fi

Key words: Reliable multicast, mobile network, simulation, protocol behaviour

Abstract: IP- Multicast will be a valuable feature in the mobile network environment. Providing a reliable multicast for mobile hosts proves to be a very challenging problem. In this paper the cost of reliability is studied using the SRM (Scaleable Reliable Multicast)- protocol and the NS (Network Simulator)- simulator. Simulations are performed using different delays and protocol timer settings to examine the protocol-related issues. The SRM- protocol is not well suited for mobile network purposes. It seems that, in particular, protocol timer settings and environmental parameters have a significant effect.

1. INTRODUCTION

The world of telecommunications is continually evolving as a reaction to an increasing demand for more sophisticated, wireless and also location-independent services. Former wireless systems have been mainly designed to offer speech services as efficiently as possible. An example of this trend is one of the most successful mobile systems, GSM (Global System for Mobile telecommunication). The dynamic and location-independent multicast model of the Internet, where members of a multicast group may join or leave the group dynamically any time and anywhere, is complex even in a fixed network. The mobile computing environment itself also adds complexity to the problem. For example, in most wireless implementations, network bandwidth is scarce, error rates are high,

movements can be frequent, and the point of network attachment for a mobile user changes so that it might not always be possible to directly access a multicast router.

In the future many of the services in the mobile networks may be implemented using multicast as the communication method. When using multicast, the service provider could send the service information to certain multicast groups, and end-users who would like to obtain that service could just join that particular group. But the provision of multicast services to the end-users seems to be a very challenging problem for several reasons. Firstly, even unicast routing for mobile hosts is a difficult problem because the routing of datagrams intended for the mobile host changes whenever the mobile host changes location. Secondly, most of the existing multicast routing protocols assume stationary hosts when configuring the multicast delivery tree. Because the delivery trees established for a static multicast could not be changed efficiently, movements of hosts after the tree is constructed create problems [1].

2. RELIABLE PROTOCOLS IN A MOBILE NETWORK ENVIRONMENT

In any reliable protocol some party must be responsible for loss detection and recovery. In unicast communication either the sender or the receiver can take this role, while in TCP the sender times transmissions and retransmits the same datagram until an acknowledge is received. This method is shown to work well in unicast. But if a TCP-style sender-based approach is applied to multicast, problems will arise. Since datagrams trigger acknowledgements from all the receivers, the sender could become congested because of the vast amount of acknowledgements received. Also, if the sender is responsible for a reliable delivery, it should keep a track on the changing group of active receivers and their states. Since the IP multicast model states that data is sent to the multicast group not to a set of receivers, this task can become impossible. Finally, the algorithms used to adapt to a changing network environment tend to be useless in the case of a multicast. For example, how is it possible to compute the round-trip time estimate to retransmit timer, when the propagation time can vary among different receivers?

The problems illustrated above denote that a point-to-point, sender-based control method does not scale up well for multicast communication. Because members of a multicast group have different transmission paths and may join or leave a group at any time, the coupling of a sender-receiver pair as performed in unicast cannot be generalised to multicast. So it is obvious that receiver-based reliability assurance is a much better method for multicast [2], [3].

When trying to apply unicast methods for multicast communication we also face the problem of conversation procedures between sender and receiver used to describe their progress of communication [4]. A receiver can request retransmission either in application data units or in terms of the shared communication state. Since, in multicast communication, the state synchronisation is much weaker and more diverse, using a shared communication state, *no name* data does not work well. For example, if a receiver joins a group later, it receives some sequence of datagrams. Because it does not know what it has missed, it cannot do anything with the data received nor make an intelligent request for retransmission [5].

2.1 Scalable Reliable Multicast

In SRM [3], when a member of a multicast group transmits data, it is always seen as a multicast by the rest of the group. Each member is also individually responsible for detecting loss and requesting *retransmission*. Because it is possible that the last datagram of a sequence is dropped, each member generates multicast periodic session messages that announce the highest sequence number received. Session messages also contain timestamps, which are used to estimate the distance from each member to every other.

Figure 1. SRM message sequence chart

When a host notices that it has lost some information, for instance data-packet number 13 in Figure 1, it starts the repair request timer and, when the timer fires, sends *repair request*. In a fixed network this would not cause any problems. The nearest node to the point of failure which has the missing data would receive the request first, start the repair timer and probably would not get *repair packets* from further nodes before the repair timer went off, and thus would be the only node sending the *repair packet* needed. But in a mobile - or in better words a radio - network the *repair request* travels through the air and reaches all hosts in that multicast group simultaneously. Although timers should be varied randomly, it seems that all hosts would send *repair packets* almost simultaneously and the idea of SRM is lost.

2.2 Simulation environment

Simulations were carried out using Network Simulator (NS) [6]. NS is a freeware simulator developed at the University of Berkeley as a variant of the REAL network simulator, which was developed at Cornell University. REAL is a network simulator originally intended for studying the dynamic behaviour of flow and congestion control schemes in packet-switched data networks. NS is a hybrid of C++ and OTcl (Object Tcl) programming languages. OTcl works mainly as a

configuration and command interface but is also a part of an implementation of NS.

The simulation model used can be seen in Figure 2, which is a screen grab from the Network AniMator (NAM). Nam is a visualization tool for NS. The model consists of nine nodes, node 0 acts as a CBR (Constant Bit Rate) traffic source and nodes 1 to 9 act as mobile hosts. The layout is star-like, where all mobile hosts are connected directly to the source node "in center".

Figure 2. Simulation model used as a NAM view [7]

2.3 Simulation results

In the simulations the total number of data packets sent is approximately 480. There are 9 mobiles entering the node with certain intervals and delays. The arrival process used is deterministic with inter-arrival times of one second or 0.8 of a second. The delay experienced simulates delays in the cell change, and in multicast group forming. This model is illustrated in Figure 3.

Each mobile host stays at the cell area for 8 seconds. The traffic source is a constant bit rate type and sends a packet every 0.0333 second. When the mobile host enters the cell and experiences delay (i.e. 600ms) it loses some information (approximately 18 data packets with 600ms delay). The SRM protocol uses timer dependent *repair request - repair* packets to ensure reliability and tries to retransmit the missing data. Part of the simulation concentrates on studying the effect of these timers on the efficiency of the protocol.

Figure 3. Simulation model illustrated using a time line

In Figure 4, different timer settings for 600ms delay and an inter-arrival time of 1.0 second are examined. In the Figure the probabilities of there being certain numbers of *repair request packets* (which all generate at least one retransmission) in the network at time T are given. Timer settings of the protocol are, by default, 1.0 second for a *repair-request timer* and 2.0 seconds for sending a *repair- packet*.

Figure 4. “The probabilities for the different numbers of repair request packets there are in the network at any time.” The delay is 600ms, mobiles arrive at 1 second intervals, using three different type of timer seeds

If compared to the total number of data packets transmitted during the simulation (not including the retransmissions), *repair requests* give us a idea as to how much reliability will cost and denote the importance of the protocol timer settings. Timer settings compared in the following tables are the best and the worst cases concerning the situation delineated in Figure 4. The percentage value denotes the proportion of *repair-request* packets of the total number of data packets.

Table 1. Probabilities of repair-requests with delay of 600ms, interarrival time 1.0 second and timers set for 0.5 second and 1.5 seconds

Probability (timers 0.5 and 1.5 sec)	Repair requests	Percentage
P (0.5)	30	6.25 %
P (0.7)	53	9.05 %
P (0.9)	94	19.58 %

Table 2. Probabilities of repair-requests with delay of 600ms, interarrival time 1.0 second and timers set for 1.5 seconds and 2.5 seconds

Probability (timers 1.5 and 2.5 sec)	Repair requests	Percentage
P (0.5)	10	2.08 %
P (0.7)	20	4.16 %
P (0.9)	29	6.04 %

If we want to be sure that “almost all” cases are covered (i.e. with probability $P(0.9)$), the number of *repair requests* in the worst case can be significant. Although *repair requests* are much smaller packets - if measured in bytes - than actual data packets, they all need processing and routing, and thus consume network resources.

Figure 5 demonstrates the same kind of situation with different parameters. Interarrival time of the mobiles is now 800ms (i.e. mobiles arrive at the cell using a deterministic arrival process and the interval between events is 800ms) and the delay experienced is 80ms. Also the timer values are altered.

Figure 4. “The probabilities for numbers of repair request packets in the network at time T ”. The delay is 80ms, mobiles arrive in 0.8 second intervals, using four different type of timer seeds

If we make same kind of a comparison as earlier on the total number of data packets transmitted (Tables III and IV), it is clear that the worst case in this second scenario is still a much better performer than the worst case in the first set-up. This leads to conclusion that timer settings and protocol performance are also dependent on other parameters, such as interarrival time and delay.

Table 3. Probabilities of repair-requests with delay of 80ms, interarrival time 0.8 second and timers set for 1.0 second and 1.7 seconds

Probability (timers 1.0 and 1.7 sec)	Repair requests	Percentage
P (0.5)	21	4.37 %
P (0.7)	37	7.77 %
P (0.9)	58	12.08 %

Table 4. Probabilities of repair-requests with delay of 80ms, interarrival time 0.8 second and timers set for 1.5 second and 2.5 second

Probability (timers 1.5 and 2.5 sec)	Repair requests	Percentage
P (0.5)	10	2.08 %
P (0.7)	16	3.33 %
P (0.9)	28	5.83 %

Although the delay seems to have less meaning than the timer values, the combined effect of the interarrival time and delay can cause a significant difference. In Figure 6 the results for four different parameter combinations are shown: 60ms delay with an interarrival rate of 1.0 second using modified timers, 80ms delay with an interarrival rate of 0.8 second using default timers, 100ms delay with an interarrival rate of 0.8 second using modified timers, and 150ms delay with an interarrival rate of 1.0 second using default timers. For instance, in Figure 6 it can be seen that the difference in the numbers of request packets with between 60ms delay with interarrival time of 1.0 second and 100ms delay with interarrival time of 0.8 second is over 40 percent at P(0.6). The situation is about the same when considering the worst performers but there are some differences. While the best performers using default timers achieved about similar results, the case is a little bit different with the worst performers; when the interarrival time is 0.8 second, the performance is worse, with greater delays [7].

Figure 6. The best values with different parameters. “def” denotes that default timer settings (1.0 second and 2.0 seconds) are used, “0.8s” indicates the interarrival time (otherwise 1.0 second)

It is clear that when demanding reliability, we have to pay. SRM should be quite efficient and scalable but numbers show that in some cases the result seems to be almost disastrous. The number of *repair request packets* in the network at a certain time T can be 20% of the total number of data packets transmitted. As mentioned earlier, this may not be any problem for transmission media; but how about network components?

In the mobile environment, parameters such as delays and interarrival times cannot be accurately predicted. This adds supplementary demands to the protocol used. Somehow the protocol should be initialised so that situations, where the number of *repair requests* might increase four times - and even more - could be avoided.

It has to be noted that the transmission errors on radio link and the effects of the timer values on the quality of service are ignored in these simulations, and thus repair requests are generated only because of the *group forming delay* experienced when arriving at the cell.

Minimising the number of the repair requests leads to the situation where clients lose information while the protocol does not manage to send repair-packets in time. Therefore, if we consider only the

number of *repair-request* packets in these conditions, "the best" solution is to increase timer values so that the mobile host has left the cell before the timer goes off, and while there will be transmission errors, there are no *repair -requests*. This example indicates that the proportions of requests initiated by timer values, error rates of the transmission path and group-forming delays are have complex causes and they all effect reliability in different circumstances.

3. CONCLUSIONS

SRM protocol implements reliability by using *repair request* and *repair packets*. When the host notices that it has lost some information, it starts the repair request timer and, when the timer goes off, sends a *repair request*. In a fixed network, this would not cause any problems, and the nearest node from the point of failure would receive the *request* first, start the repair timer and probably would not get *repair packets* from further nodes before its repair timer went off. Thus this is the only node needed to send the *repair packet*. Other hosts which received the *repair request* notice *repair packet*, transmitted by some other host, cancel their own repair timers. But in the cellular radio network, propagation delays are some microseconds and, in practice, all hosts belonging to the same multicast group in that area will get the *repair request* almost at the same time. They start their repair timers and, while protocol dependent timers and transmission delays are milliseconds, even hundreds of milliseconds, all hosts have time to send their *repair packets* before they receive a cancelling *repair packet* from some other host.

SRM should be quite efficient and scalable but numbers show that in some cases the result seems to be something else. The number of *repair request* packets in the network at a certain time T can be up to 20% of the total number of data packets transmitted. While *repair-requests* are much smaller -if measured in kilobytes- than actual *data-packets*, this may not be a bandwidth problem but it has to be remembered that they also need processing and routing. When transmission media evolve and become faster, processing times have significant meanings; the difference in transmission speed is obtained by reducing the processing needed, and the propagation on the radio link has much less meaning.

In the mobile environment parameters such as delays and interarrival times cannot be accurately predicted. This adds supplementary demands to the protocol used. Somehow, the protocol should be initialised so that situations where the number of *repair requests* can increase four times - and even more - can be avoided.

REFERENCES

- [1] Amp Acharya and B. R. Badrinath: "Delivering Multicast Messages in Network with Mobile Hosts", Intl. Conf. on Distributed Computing Systems, May 1993
- [2] Brian Whetten and Grusel Taskale: "An Overview of Reliable Multicast Transport Protocol II". IEEE Network, Volume: 14 Issue: 1, Jan.-Feb. 2000, Page(s): 37-47
- [3] Floyd, Sally and Jacobson, Van and Liu, Ching-Gung and McCanne, Steven and Zhang, Lixia: "Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", Sigcomm'95, Cambridge, Massachusetts, 1995
- [4] K. Brown and S. Singh: "The problem of multicast in mobile networks", IEEE 5th International Conference on Computer Communications and Networks, October 1996
- [5] Arup Acharya, Ajay Bakre and B. R. Badnnath: "IP Multicast Extensions for Mobile Internetworking", Infocom96, 1996
- [6] Berkeley University of California, the Computer Science Division, Network Simulator <http://Nmesh.cs.berkeley.edu/ns1>
- [7] Tiihonen, Hiirsalmi and Jormakka: "SRM in Mobile Multicast", 9th summer school on Telecommunications, Lappeenranta University of Technology, 2000