

A Protocol for Reliable SSM

By Ramakrishnan Venkitaraman

(rxv024000@utdallas.edu)

March 2003

1 Introduction

In the previous draft of this paper (www.utdallas.edu/~rxv024000/papers/) we had discussed about SSM(Source Specific Multicast) and the various issues and semantics that are to be taken into consideration when designing a reliable multicast protocol for SSM. In this paper we that address the various issues and propose a protocol for “Reliable Multicast delivery”.

In this paper we propose two versions of our Reliable Multicast (RM) protocol. The *first version* of the protocol is based on the “End to End argument design” which stresses on the objective that any protocol that’s designed to be deployed in the Public Internet must be scalable and key to scalability is to keep the network simple and let the necessary and complex functionality required by the protocol be handled by the end systems [1]. This version of the protocol needs no help from the intermediate nodes and routers and all the necessary functionality is realized by the end systems. The *second version* is an optimization of the first version in which we customize the protocol for cases in which protocol is enhanced by using the services that the intermediate routers and nodes provide.

This paper is organized as follows

1	Introduction.....	1
2	Reliable SSM Protocol - “End to End” System Design.....	2
2.1	Protocol Family.....	2
2.2	Protocol Building Blocks and Rationale.....	3
2.2.1	Target Application	3
2.2.2	Scalability	3
2.2.3	Delivery Service model.....	4
2.2.4	Group Membership Dynamics.....	4
2.2.5	Network Topologies.....	5
2.2.6	Router/Intermediate System Assistance	5
2.2.7	Manual vs. Automatic Configuration	5
2.3	Protocol Components and Rationale.....	5
2.3.1	Data Reliability	5
2.3.2	Loss Detection	6
2.3.3	Loss notification.....	7
2.3.4	Loss Recovery.....	9
2.3.5	Protocol in short.....	10
2.4	Building Block Advantages	11
2.5	Building Block Disadvantages.....	12
3	Reliable SSM Protocol – Optimized.....	12
3.1	Why need this?.....	12

3.2	Protocol Family.....	12
3.3	Protocol Building blocks and rationale.....	13
3.3.1	Target Application	13
3.3.2	Scalability	13
3.3.3	Choice and positioning of the Intermediate nodes.....	14
3.3.4	Functionality provided by the “Helpful Intermediate” (HI) nodes	14
3.3.5	FEC Based Approaches	15
3.3.6	Controlled Repair Forwarding	15
3.3.7	Delivery Service Model	15
3.3.8	Group Membership Dynamics.....	15
3.3.9	Network Topologies.....	15
3.4	Protocol Components and Rationale.....	16
3.4.1	Source/Sender discovery.....	16
3.4.2	Data Reliability	16
3.4.3	Loss Detection	16
3.4.4	Loss Notification.....	16
3.4.5	Loss Recovery.....	16
3.4.6	Fault tolerance.....	16
3.4.7	Protocol in short.....	17
3.5	Building Block Advantages	18
3.6	Building Block Disadvantages.....	18
3.7	Conclusion	18
3.8	References.....	18

2 Reliable SSM Protocol - “End to End” System Design

In this section we propose the version of the RM for SSM protocol which is based on the idea of End to End System design [1]. All the necessary functionality that are required for the implementation of the protocol is implemented at the end systems running the application and the network is kept simple. This design decision is based on the rationale based on the view that the best way to meet diverse application requirements is to leave as much functionality as possible to the application [4].

2.1 Protocol Family

As has been suggested in [2], two of the primary reliability requirements for a reliable transport protocols are confirming delivery to the sender and that of achieving good throughput.

The first requirement that confirms the delivery to the sender is the most basic one. This can be accomplished in a number ways and some of the ways can be found in the initial draft of this paper. Reference [2] gives some of the popularly followed approaches and classifies the protocols into protocol families based on how the protocol confirms the delivery of the message to the sender.

Our protocol uses the NACK (Negative ACKnowledgement) strategy in which case the receiver sends a NACK to the sender when it does not receive a packet that it was expecting. How the receiver gets to know that he had indeed missed a packet is an issue in itself and is handled in the following sections.

Based on the categorization of protocols in to families as followed in [2] this End to End version of our protocol belongs to the “*NACK ONLY*” category wherein we try to reduce the number of acknowledgements (i.e. the feedback traffic from the receiver to the sender) by sending only Negative Acknowledgement’s to the sender and thus avoiding the well known problem of NACK implosion. Again how NACK implosion is handled is an issue in itself and is handled in the following sections.

2.2 Protocol Building Blocks and Rationale

In this section we discuss about the important design decisions made as a part of the protocol development process and the rationale behind the same.

2.2.1 Target Application

This protocol was designed and developed mainly for applications such as reliable file transfer or its equivalents like live software updates where the in order delivery of data is not a requirement for the application to work correctly. Moreover we assume that the sender application only needs to know whether all the receivers got the packets or not and is not interested in knowing which of the receivers got it and who did not? We also make the following assumptions a) Data reception characteristics like the time of reception are not constrained to fall in a particular range. b) The protocol should support a wide variety of network types and must be scalable. c) The order in which the data is sent at the sender side may be different from the order in which the receiver will receive it. d) We are developing a protocol to work in the public Internet. e) Network has a feedback path from the receiver to the sender. f) Newly joined receivers to the session will get the data only from the time of their join. For more information regarding how these assumptions affect protocol design please refer to [12].

This protocol can also be extended to applications where in order delivery of the data is important (i.e. applications like real time transmission of a multimedia content to a set of receivers wherein new data always supercedes the old data). This extension is made possible with the help of a set of added functionality to the protocol and a set of constraints posed on the receiver end (like the presence of a Jitter Buffer) of the application. This assumes that the receiver has enough memory to buffer data so that even if a loss does occur, it gets repaired by the time the application asks for it.

2.2.2 Scalability

The primary reason for making the protocol to rely on End to End mechanisms [1] was to make sure that it is scalable to be deployed in the Public Internet. By Scalability we mean

that the protocol must support and work under a wide variety of conditions that include multiple network topologies, receiver set size and like. By making the protocol to be End to End we have made sure that the protocol is scalable as it does not make any assumptions about the underlying network and does not ask for any help from the intermediate nodes and the routers.

2.2.3 Delivery Service model

The service model that's supported by this protocol will range from long lived transfer sessions of bulk quantities of data (file transfer) to other kinds of applications like real time transmissions.

Our service model also assumes that the sender has the ability to cache the data it has already sent. This ability is required as the sender is the only one who can transmit data in the case of SSM and in our case (End to End) we rely only on the sender to do the retransmission if the data which he had already sent gets lost or does not reach the receiver with out error. Then, in order to retransmit the data on request by a receiver or a set of receivers (on the reception of a NACK), he must maintain copies of data (in his buffer) that he had already sent and for which he is yet to confirm the reception. How long does he hold the buffer and how does he exactly make sure that all the receivers have got the data that has been sent are issues which will be addressed in the following sessions.

2.2.4 Group Membership Dynamics

Our protocol supports the ability to continue communication among the existing group members even if group membership changes dynamically over time. Group membership can change over time due to two reasons. One due to the addition of new members to the group when the session is still on and the second reason is when some of the group members choose to leave the group due to some reason.

In the first case that is during addition of new members to the group, the receivers will certainly be able to receive all the data that's transmitted by the sender from the time that it joined the group. Whether it will be able to receive the packets that have already been exchanged depends on factors like the degree to which the sender will be able to buffer previous transmissions and its ability to send those packets only to this receiver (say like unicast). It cannot multicast as it has only one recipient and multicasting will result in the waste of bandwidth and network resources. As of now we do not address these issues.

In case one or more of the existing receivers decide to leave the group then their group membership will be done away with and they will not receive the packets that are subsequently transmitted. It can also be noted that the existing multicast session will in no way be affected as this node decided to go away from the group because this version our protocol for its working, does not rely on any of the functionality provided by the leaving node. In cases where security of the data does matter (i.e. we need to ensure that

the leaving members are not allowed to access the transmitted data) some group key management mechanism must also be implemented.

2.2.5 Network Topologies

The process of designing a protocol for the Public Internet has traditionally assumed that service will be provided to a wide variety of network topologies. Since our protocol which provides reliability on top of SSM is based on the End to End design and since it does not make any assumptions about the underlying network, our protocol can be deployed anywhere SSM can be deployed.

2.2.6 Router/Intermediate System Assistance

The efficiency of a reliable multicast protocol can be improved if we get help from the intermediate systems and/or routers in the network. But this may very well compromise on the scalability of the protocol and its deployment in the Public Internet.

This version of our protocol does not ask for any assistance from the routers or intermediate nodes and is only based on End to End design. We also propose another version of this protocol which is an efficient way of providing reliability when compared to the first version but it does make some assumptions about the ability of the intermediate nodes and routers and get assistance from them.

2.2.7 Manual vs. Automatic Configuration

Our protocol for providing reliability for SSM is a fully automated process that's running at both at the sender and the receiver nodes and provides reliable delivery on top of SSM. The process of providing reliability needs no manual intervention. As discussed in [6] it's always recommended that a protocol that's designed to work in the public Internet be always automatic due to the heterogeneity of the public Internet.

2.3 Protocol Components and Rationale

In this section we will look at the design details of our Reliable SSM protocol. This section will give us more insight about how does our algorithm really works and accomplishes its task of providing reliability on top of SSM.

2.3.1 Data Reliability

As we have already discussed in the previous sections, one of the most basic functionality that's expected off a reliable protocol is reliable transport. That's all the interested receivers of the SSM channel must be able to receive all the data that's sent to that channel and in case they do not get the data they were expecting, they will request a retransfer from the sender and the sender will retransmit the same.

This loss of data can be due to various reasons that are common in the Internet. It should also be noted that the receiver might have got the data but when it checked the data and found that the data has been corrupted (maybe due to transmission errors) the receiver sent a retransmission request.

Providing data reliability consists of three basic components. They are

- i. Loss detection,
- ii. Loss Notification and
- iii. Loss Recovery.

We will discuss it in the following sessions.

2.3.2 Loss Detection

Well, one of the fundamental questions to be answered when designing a reliable protocol is how does the receiver get to know that it has indeed lost a packet that it should have received. There are many approaches to handle this. It is to be noted that the receiver after discovering the loss unicasts the retransmission request to the sender because in SSM only the sender is allowed to send to the group. Now let's try to understand the various ways using which loss could be detected.

If the data from the sender is a set of packets with sequence numbers then the loss of a packet will be detected when the receiver receives a packet whose sequence numbers are not contiguous. Say the receiver received a packet with sequence number 5 and another with the sequence number of 7, it's pretty clear that the receiver has missed packet with number 6. But one thing to note here is that we are assuming in order delivery of packets. In the case of out of order delivery of packets, it could very well be the case that the packet with the sequence number 6 was not lost but is on its way to the receiver. So in cases like these the receiver cannot simply ask the sender to retransmit just because the packet was late. So the receiver should wait before asking the sender to retransmit.

How long should the receiver wait before it asks the sender for the retransmission of data? This is based on the RTT (Round Trip Value). The receiver should wait for at least $RTT/2$ time duration before asking for retransmission. How does the receiver get to know what the RTT value is? This is done by configuring the sender to send the RTT value to the receivers as soon as any receiver joins the session. The next question is how does the sender get to know what the RTT value to the receiver is? Calculating the appropriate/reasonable RTT is in itself a big issue and as of now we do not propose a way of how to do the same. Let's as of now assume that the RTT value is known. There are many papers in the literature that deal with the process of how to calculate the RTT. Some information regarding the same can be found in [11].

Now the problem with using sequence numbers to detect packet loss is how do we get to know that the receiver has lost the last packet sent by the sender? The RTT value can again be used for this. But in this case, the receiver should also have knowledge about how many packets it is supposed to receive from the sender. One straight forward way

that this can be done is to ask the sender to transmit the same to the receivers when they join. That is the sender will tell the receivers how many packets its going to receive in this SSM session. After knowing the total packet count and the RTT value, if the receiver has received all the packets but for the last packet and the timer(at least $RTT/2$) has expired then the receiver can ask the sender to retransmit the lost packet.

For applications in which the sender will continuously be sending the packets to the receiver, there is another way to detect the loss of packets given the RTT value. Here we don't need the packets to have distinct sequence numbers. In this case, the receiver can wait for at least $RTT/2$ interval and if he does not receive the packet that it was expecting, then it can ask for retransmission. Note that here we don't need the packets to have sequence numbers to detect a loss. If the sender does not transmit for quite a while then the receiver can assume that the transmission session is over. In cases where the sender needs to take a break and again retransmit the sender can send a control packet to the SSM group intimating the message that he is taking a break and then come back after the time duration. The exact nature of these will depend upon the application specific needs.

Now that we have seen how the receiver gets to know that it has indeed lost a packet, lets consider the issue of Loss Notification. The next section discusses the same.

2.3.3 Loss notification

Loss notification answers the following question. How does the receiver communicate that it has lost a packet to the sender? There are many approaches to solving this problem. It is to be noted that the receiver sends a unicast packet to the sender to notify about the loss of the packet. The reason is that in SSM only the sender can transmit to the group. How loss notification is done is one of the areas in which Reliable protocols for SSM will differ from the well known ASM model. Protocols for the ASM model like PGM [3] use the ability of any receiver to send to the group to avoid NACK implosion.

NACK implosion is the term that's used to refer to the case in which we have many receivers who have missed a packet and all of them send a NACK to the sender asking for retransmission of the same packet and the sender is now faced with a new problem of handling all those NACKS and processing the same and this is an unnecessary overhead to the sender. It would be sufficient if the sender is to receive only one NACK per lost packet since in any case it is going to multicast the packet retransmission.

One straight forward approach to notify the sender of the loss is to ask the receiver to send a NACK to the sender as soon as it detects the loss. This approach does work but has its own set of problems. The most important disadvantage of using such a naïve approach is the following. If there is only one receiver who experiences loss and notifies the sender of the loss then its fine but in cases where there are a many receivers who have experienced loss and all of the receivers send a NACK to the sender then the sender will face the problem of NACK implosion. The following approach can be used to avoid the problem of NACK implosion.

In order to avoid the problem of NACK implosion timer based approaches can be applied. We herein propose timer and probability based approaches similar to the one discussed in [5], [8] and [9]. The basic idea in timer based approaches is to wait for a certain interval of time before requesting for retransmission of the packet. To decide on how long should the receiver wait before sending the notification to the sender saying that it has lost a packet and wants a retransmission of that packet is the problem in question.

In our protocol, the *time duration* for which the receiver is going to wait before sending the retransmission request to the sender is a *function of 2 components*. One is the distance of the receiver from the sender and the other factor is the size of the receiver set which denotes the number of receivers who belong to this SSM group.

Distance can be estimated based on the RTT value. A higher RTT value from the sender to the receiver means that the receiver is at a greater distance from the sender when compared to another receiver with a smaller RTT value to the sender. A receiver who is at a greater distance from the sender will on an average have a higher wait time for the timer when compared to the one which is near the sender. The reason for the same is that there might have been a receiver closer to the sender who has also missed the packet and who has after waiting for a smaller timer value sent a request to the sender for retransmission. In this case any other retransmission request for the same packet is redundant. What exactly is the wait time and the algorithm to calculate the same are issues which are not handled by this paper. Some details regarding the same can be found in [9].

The other factor to be considered as mentioned above in the calculation of the delay or wait timer before transmitting the retransmission request is the number of receivers that belong to this SSM group. The rationale is that if the receiver set size is small then the wait time range for the receivers must be small. If the wait time is large for a small set then there will be an unnecessary high delay between the time that loss is detected and the time at which the receiver gets the retransmitted packet. In case the receiver set is large then the range of time over which the receiver waits before requesting for retransmission must be large else a smaller wait time range would result in many receivers sending retransmissions at the same time and could potentially result in NACK implosion. Some more details regarding the same can be found in [2] and [3]. Each receiver may also be given some probability values which when chosen from a random range will tell the node the probability with which it should respond with a retransmission request and the probability with which it should defer such a request. Some more information regarding the same can be found in [2].

“Research suggests that NORM (Nack Oriented Reliable Multicast) group sizes on the order of tens of thousands of receivers may operate with modest feedback to the sender using probabilistic, timer- based suppression techniques” [5] and [10]

So our idea of using random timers and probabilistic mechanism is very realistic and can be done but how exactly are the values chosen and what’s the nature of the distribution are questions that are to be given more thought into. For these types of timers it is

important that the receivers in addition to initiating the NACK timers record the current position in the senders transmission sequence at which they initiate this cycle. When the timer expires, the receivers should consider only repairs needed with respect to the recorded position. The importance of this can be found in [5].

To minimize NACK loss we may want (in cases) a receiver to repeatedly (may be periodically) retransmit a NACK until it hears from back from the sender in the form of a NACK acknowledgement [3]. Simply stated the above steps help in *NACK elimination* (suppression of duplicate NACK's).

So as of now we know how the receivers detect that they have lost a packet and also how they communicate the same to the sender. In the next section we see how the senders respond to such requests for retransmission.

2.3.4 Loss Recovery

As the name suggests this function deals with how the sender responds to the loss notification that he gets from the receiver. We note again that the receiver actually unicasts the NACK to the sender. When the sender receives the packet, by observing the packet, he comes to know what the packets that he needs to send back to the receiver as a retransmission for recovery. Now before sending the packet the sender does the following.

One of the first things to keep in mind is that since our protocol employs random timers and probabilistic estimates in most of the cases we expect only to receive one NACK though potentially the number of receivers who might have lost the packet will be more than one and all but the one which indeed sent the NACK will be waiting on timers before deciding on to or not to send the NACK.

In the case of the ASM model and the protocols that have been designed for the ASM model like PGM [3], since the NACK is multicast other receivers will get to know that the sender has got the NACK. But in the SSM model since the NACK is unicast, we must have a mechanism that will enable the receivers who are waiting on the timers to know that the sender has indeed received the NACK and retransmission will follow soon. This design decision will depend on the application for which we are trying to provide reliability.

Note that one of the main reasons why the receiver might have lost the message from the sender and the reason why it is asking for a retransmission is because the network is congested. So it is very likely that the sending window in the sending node and/or the intermediate nodes is already full and might have also got a recent reduce in size due to congestion. So how do we retransmit and ensure that all the receivers get the retransmission as soon as possible and also ensure that we do not get duplicate NACK's that is how do we make sure that our retransmission reaches the receivers waiting for it in time (i.e. before receivers timeout).

If the application for which we are building reliability deals with messages of large sizes, then the amount of time that it will take for the application to retransmit the message will be large and in that case it is very likely that loss is due to congestion and the retransmission might not serve the purpose (as discussed in the previous paragraph). So what is being suggested is that the sender will send a Multicast NACK acknowledgment to all the receivers in the receiver group [2]. The size of this message will be very very small when compared to the original packet and it may be sent in an “out of band” way if possible. By doing this we make sure that the sender is not subjected to NACK implosion. This is achieved by making the receivers to reset their timers as soon as they hear the NACK acknowledgement. The reset value should be designed in such a way that there is enough time for the retransmission to reach the receivers. More thought must be given into the specification of the exact values of the same and we do not get into it as of now. Simply this process helps in *NACK elimination*.

In cases where the size of the message is comparable with that of the size of the NACK acknowledgement, the sender can consider sending the original message (retransmission as out of band) of the packet itself instead of trying to send the NACK acknowledgement. This will reduce the overhead involved in sending the NACK acknowledgement followed by message (retransmission).

It should also be noted that due to the nature of the algorithm discussed, it is possible that the sender receives more than one request for a dropped packet. When the sender detects the same, the sender can transmit a control packet to the receivers to increase their spread in retransmission distribution which is in most cases implemented as exponentially distributed random timers. In order to prevent these duplicate requests from making the sender to retransmit duplicate repairs, a source should ignore retransmission requests for a period of time after it has sent a retransmission for a packet. [9] gives some details regarding how long this timer should be.

Receivers may also want to consider exponentially backing off their timers in case they receive a missing data or the NACK acknowledgement before its own request timer for that data expires. A similar approach has been suggested in [9] along with the issues in such an approach.

2.3.5 Protocol in short

2.3.5.1 Sender

The job of the sender is basically to keep on sending the data to the set of receivers belonging to the SSM group. The sender will implicitly assume that the receivers are properly receiving the packets unless informed otherwise. The receivers inform the sender that they have not received a data by sending a NACK to the sender in which case the sender retransmits the requested packet.

The sender has a sending or transmit window that it advances when it is sure that all the receivers have received the transmitted data. How the sender gets to know the same is

based with the use of times. The value of this timer is the maximum RTT value from the sender to any receiver in the receiver set. If the sender does not hear a NACK from the receivers for the maximum RTT time, then it will assume that all the receivers have received the data and he can move the trailing edge of the transmit window.

The random delay used by the sender to wait before moving the trailing edge of transmit must be adaptive/vary as the receiver set scaled i.e. increase as the receiver set scales up and decrease it scales down. Another way of implementing the same is suggested in [3] wherein the trailing edge is advanced arbitrarily.

2.3.5.2 Receiver

The job of the receiver is basically to receive the packets that are sent to it by the sender and to inform the sender about a loss in case it detects a loss of the data. The exact way of how does it detect a loss and when does it send the retransmission requests is as discussed in the preceding sessions. Moreover, the random delay used by the receiver to wait for before sending the NACK must be adaptive/vary as the size of the receiver set scales i.e. increase as the receiver set scales up and decrease as the set scales down.

2.3.5.3 Intermediate nodes

This version of the protocol assumes no help from the intermediate nodes or routers. If SSM can run on it then using this protocol ensures that reliability is guaranteed. Well that's what "End to End" design is about; implement the functionality at the end systems.

2.3.5.4 Group Membership

How do the receivers get to know about the sender and the SSM group is not handled in this paper and we assume that the receivers know what SSM group they are interested in being a member of. Group membership notification is sent to the sender and the group is controlled and managed by the sender of the group.

2.3.5.5 Session Management

The Reliable SSM session starts when the receivers convey their interest in being part of the group and the sender starts transmitting data to the group. At the start of the session the sender sends the receivers control packets that are essential to the protocol working. New receivers who join in the middle of the session are also sent similar control packets by the sender when they join. The session ends when the sender has completed the transmission and is sure that all the receivers have got the packets.

2.4 Building Block Advantages

We have stressed the advantages through out the previous sessions. As can be clearly seen, this version of our protocol relies completely on the "End to End" capabilities of the

systems and thus is simple to deploy in a setting like the public Internet where we cannot expect the intermediate routers and nodes to provide you any support to your protocol. One of the main reasons is that the public Internet is not owned by any single authority and is composed of a mix of separately administered domains. Moreover since our protocol is “End to End” its clearly very scalable and has all the advantages that are pointed out in [1].

2.5 Building Block Disadvantages

Well the obvious risk associated with designing a protocol truly on an “End to End” basis is that its performance will not be as good as the one which gets assistance from the intermediate systems. The intermediate systems can help in the retransmission of packets, NACK suppression etc.

Moreover in our case the sender will be able to move the trailing edge of the transmit window only after waiting for a time interval equal to the maximum RTT time. The delay can be objectionably high for some applications and may not be acceptable. For cases like these we need a protocol which gives out a performance even if its to demand for some extra functionality from the intermediate routers.

For example for the ASM model PGM [3] proposes a hybrid scheme that gets help from the intermediate nodes for NACK suppression, NACK elimination and constrained forwarding. Seeking help from the intermediate routers greatly helps PGM achieve good performance.

3 Reliable SSM Protocol – Optimized

3.1 Why need this?

In the previous section we had discussed our version of the Reliable SSM protocol which relies on the “End to End argument” design principle. In this session we will see a version of this protocol that uses some services provided by the intermediate nodes to improve the overall performance of the protocol. The original design of the public Internet was based on the “End to End” argument design principle. But the new range of emerging set of requirements could compromise with the original design goals [13]. So, for want of performance, protocols do compromise on “End to End” system design.

3.2 Protocol Family

This version of protocol belongs to the set of protocols that can be categorized to be based on the TREE topology [2]. The set of senders, intermediate nodes and receivers can be considered to form a tree like structure with the receivers forming the leaf nodes and

the sender forming the root of the tree and the intermediate routers and nodes supporting the protocol and aiding in its functionality forming the non leaf nodes of the logical tree. We use the term source to refer to the original *source* of the SSM session and the term *senders* to basically mean the intermediate nodes which act on behalf of the source as senders of packets belonging to the same session.

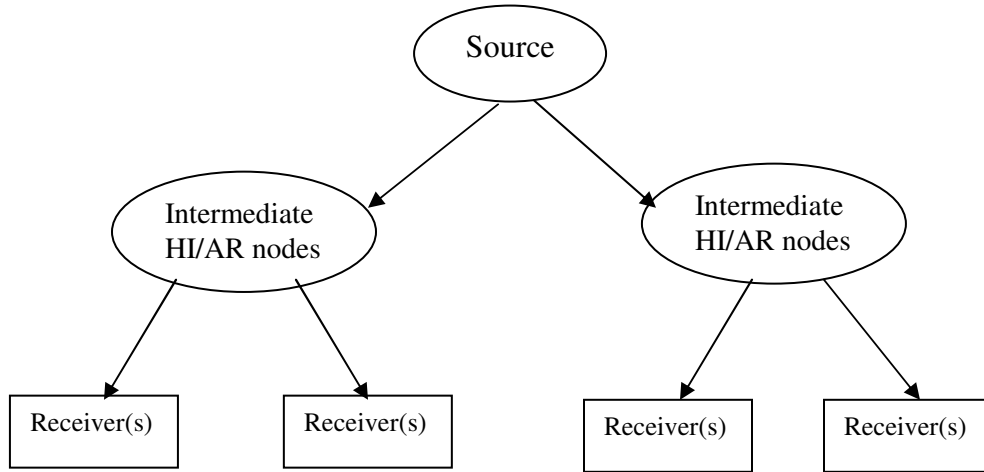


Figure: Logical Tree Structure

3.3 Protocol Building blocks and rationale

As we have already pointed out, this version of the protocol is designed to make the protocol more efficient in its function of providing reliability by getting help from the intermediate nodes when compared to the initial version of this protocol.

3.3.1 Target Application

This version of the protocol is designed targeting the same set of applications for which the previous version of the protocol was designed for.

3.3.2 Scalability

In the previous version of the protocol, our main concern regarding scalability was to enable the protocol to support a wide variety of network topologies, hosts and to make it work any where SSM would work. But as was mentioned in the previous sections this approach uses timers and maintaining timers when the receiver set is too large is tough. A well known approach to handle this type of scalability is by using *hierarchy*. A similar hierarchy based approach is also used in PGM [3].

In this version of the protocol we have a set of intermediate nodes that help the sender in reliably transmitting the packets to the destination receivers. Moreover, the presence of

hierarchy in the network will make the network to be more scalable when compared to the original version of the protocol. The number of intermediate nodes to have in the network and the exact functionality provided by these nodes is application specific.

3.3.3 Choice and positioning of the Intermediate nodes

Now that we have decided to get help from the intermediate nodes, we have to decide on how do we select a node to be an intermediate node supporting our protocol and where do we need to position the selected intermediate nodes. There are a few approaches that are possible to achieve this.

The set of nodes that are selected must be able to store the messages that are sent from the source and be able to retransmit them in case some of its receivers report a loss and request for retransmission. The selection will depend on the functionality that we are expecting these nodes to provide.

The positioning of the receivers must be in such a fashion that the given set of say N intermediate nodes helping our protocol (let's call them *HI Nodes*) will improve the performance of our protocol to the maximum possible extent. How this is done is in itself an area of research and we do not discuss the details in this paper.

3.3.4 Functionality provided by the “Helpful Intermediate” (HI) nodes

The functionality that we expect the HI nodes to provide will help us in making the correct choice of the HI nodes. The approaches followed here are very much like a “divide and conquer” approach to handle a large set of receivers by implementing a hierarchy of nodes with a logical tree like topology.

One approach is to allow some of the receivers (call them as *Active Receivers-AR*) of the original parent of the SSM session to be themselves the sources for another set of receivers. In this case what we have is a set of multicast sessions. One from the original source to the set of AR's and another set from the AR's to the set of receivers of the multicast session. We can consider the set of AR's to be managing its own domain of receivers. Note that in case we have a very large receiver set then we may have a hierarchy in which AR's may in turn have all or some of its children to in turn be AR's.

Another approach is to just allow only the root to be the sender to the group and all the intermediate nodes are in the tree just to provide the sender support in NACK suppression and NACK forwarding. Here the intermediate nodes do not buffer packets and neither do they retransmit that. The type of approach to be followed will depend upon the application in question and interested readers are referred to [7] wherein similar approaches have been suggested. The IR nodes can also aid in the process of congestion control as has been suggested in [2].

3.3.5 FEC Based Approaches

FEC stands for Forward Error Correction. In cases where loss is too high and the receiver set is too large and any sort of feedback mechanism for loss notification is prohibitive, reliability can be provided by using FEC. In the case of FEC, we use additional bits in the transmitted data that will aid in the process of reconstructing the lost packets or the lost information in the packets. FEC based approaches are also called as proactive approaches in which we send more information along with the original set of packets which will aid in the process of reconstruction in the event of loss even before we detect any loss. For more information refer to [3] and [5].

3.3.6 Controlled Repair Forwarding

In cases wherein we have the set of intermediate nodes (HI) aiding the functionality of the protocol to be AR's, what is done is; in the event of a loss report from one of the receivers of the AR, it will always try to satisfy the repair requests using its resources and will get help from the original source of the SSM session only if it's not able to do the repair. By doing this we are reducing some load in the sender and we call this strategy as *controlled repair forwarding* and similar approaches are employed in [3].

3.3.7 Delivery Service Model

As was discussed in the previous sections the delivery service model will depend on the type of functionality that we are expecting from the HI nodes and the ability of the AR's. For maximum performance it's recommended to design the system in such a way that hierarchy is followed both during the process of sending the messages from the source/senders to the receivers and also when sending the repair requests from the receivers to the senders or to the original source.

3.3.8 Group Membership Dynamics

This version of our protocol has also been designed to support a wide variety of groups and group sizes. The issue of scalability is addressed by following the hierarchy based approaches suggested in the previous sections. Group membership can change dynamically and requests are sent to either the original source or to the HI nodes.

3.3.9 Network Topologies

The question of whether a particular topology supports our model will depend upon whether the nodes in it can or cannot act as HI nodes. In case we find a network that does not support the extra functionality as requested by this version of the protocol, then the receivers over that particular network can use the End to End version of our protocol and other networks which do support may implement the optimized version of the protocol.

3.4 Protocol Components and Rationale

3.4.1 Source/Sender discovery

In this version of the protocol, we may consider the AR's to be also like senders it may be the case that some receivers may potentially have an option of choosing between the source and the sender or between the senders and this can be done by taking in to consideration a set of non functional requirements like cost. This process can be also automated by making the DNS servers to map the join requests of the receivers to an SSM group to the nearest (or any other non functional requirement) available sender.

3.4.2 Data Reliability

As with the previous version of the protocol, the main objective of this version of this protocol is to provide reliable transmission of data from the sender to the receiver. But as has been discussed in the previous section, the way it is done is by getting help from the intermediate nodes.

3.4.3 Loss Detection

The mechanism using which a receiver will get to know that it has not received a packet that it should have received is by using the same algorithm that was suggested in the "End to End" version of the protocol.

3.4.4 Loss Notification

After detecting a loss, the loss notification process uses the same algorithm that was suggested for the "End to End" version of the protocol but for one change. The change deals with to whom the receiver informs about the loss. In this case the receiver sends the loss to its immediate sender that is the one from whom it is receiving the data for the multicast session irrespective of whether that node is the original SSM source or an IR node acting as the sender.

3.4.5 Loss Recovery

The recovery process will be local recovery in cases where the AR nodes buffer the packets and send them as repair packets when requested for. If this is not possible then assistance can be got from the original sender of the SSM session and the recovery process may proceed. The other details regarding the recovery process are similar to the one discussed for the "End to End" version of the protocol.

3.4.6 Fault tolerance

Given that there are a set of nodes which could be potentially considered to be mirrors of the original source based on need, fault tolerance can be provided wherein the failure of

an HI node acting as a source may be rectified by making another HI as the new sender and this process must be transparent to the receiver. How this is accomplished is an issue in itself and is not addressed in this paper. A similar approach for treating these nodes as backup nodes has been suggested in [7]. Well, no fault tolerance can be provided if the node that fails is the original source unless we have source replication and transparent SSM session control transfer.

3.4.7 Protocol in short

3.4.7.1 Source/Sender

As in the “End to End” version of the protocol the job of the sender is to send the packets to the receivers and to provide with retransmission of packets when it receives some repair requests of the packet. The difference is that the source may retransmit to an AR or an ordinary receiver. In the case on an AR, the AR acting on behalf of the source retransmits it to the receivers.

3.4.7.2 Receivers

The functionality of the receivers here is the same as was for the “End to End” version of the protocol with the NACK’s being sent to the nodes from which it is receiving the packet. The receivers can also prefer to receive only from the original source or from an AR node. The tradeoff in the choice of the same is similar to the ones discussed in PIM [14] where the argument is on choosing between the source specific tree and a shared tree.

3.4.7.3 Intermediate nodes

The intermediate nodes can act as AR or HI nodes helping the source by employing NACK elimination, controlled forwarding and providing with retransmissions. These functionalities provided by the intermediate nodes are the ones that help the protocol in achieving a higher degree of performance.

3.4.7.4 Group membership

The task of maintaining group member ship is handled both by the HI nodes and the original source. They follow a divide and conquer strategy for handling the join, leave and retransmission/repair requests.

3.4.7.5 Session Management

Global session management (SSM) is in the hands of the original source of the SSM group while the local session management is handled by the IR and AR nodes respectively. Again, they follow a divide and conquer strategy for accomplishing session management.

3.5 Building Block Advantages

The main advantage of this version of the protocol is its improved performance when compared to the original version of the protocol. Please refer to the previous sections to get to know how this extra performance is gained.

3.6 Building Block Disadvantages

The trade off that has been made to achieve increased performance in this version of the protocol is by making the intermediate nodes help the SSM source. So obviously this protocol cannot be applied in networks in which such help from intermediate nodes cannot be expected and in these cases we have to use the “End to End” version of the protocol to provide with reliable transmission of data. More over special care should be taken in the selection and positioning of the designated nodes in order to achieve good performance. For example if the HI node is an AR then care should be taken that it is willing to stay throughout the session and will not leave in the middle of the session.

3.7 Conclusion

In this paper we have discussed two versions of RM protocol. One based on the “End to End arguments in system design [1]” and the other an optimized version of the same in which we get help from intermediate nodes in the form of HI nodes and AR nodes. The performance of the second version is better when compared to the first one while the first version is considered to be more applicable in the context of the public Internet. During the course of the discussion we have left out a lot of details like how RTT is calculated, how the random timer’s values are set, positioning and selection of the HI nodes etc. These are issues which need more thought in to it and are research areas. In many cases they depend on the nature of the SSM application for which we are trying to provide reliability.

3.8 References

1. "End to End Arguments in System design" by J.H.Saltzer, D.P.Reed and D.D.Clark - 2nd International Conference on Distributed Systems(April-1981, France) IEEE1981
2. "Reliable Multicast Transport Building Blocks for One-to-Many Bulk Data Transfer" by B.Whetten et.al -Networking Working Group, RFC 3048
3. "The Reliable PGM Multicast protocol" by Jim Gemmell et.al –IEEE network January/February 2003.

4. "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing" by Sally Floyd et.al -(earlier version of paper appeared in ACM SIGCOMM 95)
5. "NACK-Oriented Reliable Multicast (NORM) Building Blocks" by B.Adamson et.al - Internet Draft, RMT working Group.
6. "Reliable Multicast Transport Building Block for Track" by Brian Whetten et.al. - RMT Working Group IETF Draft
7. "IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications" by Hugh W. Holbrook and David R. Cheriton -1999 ACM SIGCOMM
8. "Reliable Multicast Transport Protocol(RMTP)" by Sanjoy Paul et.al - IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communications.
9. "Reliable Multicast Framework for Light-weight Sessions and Application Level Framing (SRM)" by Sally Floyd et.al -IEEE/ACM Transactions on Networking, December 1997
10. "Quantitative Prediction of Nack Oriented Reliable Multicast (NORM) Feedback" by J. Macker et.al, Proc. IEEE MILCOM 2002, October 2002.
11. "Strawman Specification for TCP friendly (Reliable) Multicast Congestion control" by Mark Handley et.al
12. "A Protocol for Reliable SSM-Draft" by Ramakrishnan Venkitaraman –The University of Texas at Dallas, February 2003
13. "Rethinking the design of the Internet: The end to end arguments vs. the brave new world" by Marjory S. Blumenthal et.al -ACM Transactions on Internet Technology, Vol 1, No 1, August 2001.
14. "Protocol Independent Multicast by Andrew Adams et.al" -IETF working group PIM
15. "Multicast Routing in Datagram Networks and Extended LANs" by Stephen Deering et al. ACM transactions on computer systems, May 1990
16. "The Evolution of Multicast: From the MBone to the Interdomain Multicast to Internet2 Deployment" by Kevin C.Alemorth
17. "An Overview of SSM" by Bhattacharya et al. –IETF Draft
18. "An Architecture for Wide Area Multicast routing" by Stephen Deering –ACM 1994

19. “SSM for IP” by Holbrook et al. –IETF Draft
20. “PIM-SM (Revised)” by Bill Fenner et al –IETF Draft
21. “The Reliable Multicast Design Space for Bulk Transfer” by H.Handley et al. –
Networking Working Group.