Asian Journal of Information Technology 6 (5): 622-630, 2007

A Novel Approach to Make Stealth Multicast Paradigm Steadfast

[1]M. Raja Ram and [2]E.P. Srinivasan
[1]Department of EEE, Thanthai Periyar Government Institute of Technology, Vellore
[2]Department of Information and Communication, Government College of Engineering, Salem, India

Abstract: Stealth Multicast is a novel concept that allows for practical adoption of network-level multicast on a domain-wise basis rather than a global scale. In the stealth multicast framework, redundant unicast packets are dynamically assembled into virtual groups for multicast transmission across the domain. At the edge of the domain, the packets are converted back to unicast, thus hiding the existence of stealth multicast from the external Internet. True to its namesake, stealth multicast operates in complete stealth, providing seamless interoperability without requiring modifications to end-user applications nor requiring inter-domain support. Although stealth multicast can offer a significant mechanism for multicast deployment, it is not ideal for applications which require reliable delivery.This is because Stealth multicast is created for connectionless UDP streams. This study proposes how to achieve reliability in Stealth Multicast paradigm by incorporating NACK based protocol.

Key words: Stealth multicast, NACK, packet loss recovery, local repeater, end-end delay, reliability, ns2

INTRODUCTION

Figure 1 illustrates the paradigm where our approach is applied and its fundamental concepts wherein a server dispatches information to four separate clients using separate unicasts. The key component of the model is the Stealth Multicast (StMc) module which assembles candidate packets into virtual groups for multicast transmission across the network.

The VGDM is placed at the edge of the domain and queues packets for assembly into virtual groups for multicast transport across the domain. The stealth multicast process begins as a group oriented application transmits packets via separate unicasts to multiple clients. The packets travel via the uplink to the domain and arrive at the edge router. The packets are then transferred to the VGDM for virtual group consideration. A filter may be applied at the edge router to remove packets from consideration that should not or would never become part of a virtual group.

Stealth multicast (Striegel, 2004) can be used for a wide range of applications such as on-line games where end-to-end network-level multicast support is not available and the strict delay requirements do not permit application-level multicast. Stealth multicast is a domain-wise solution rather than an end-to-end solution. Thus, stealth multicast is interested only edge-to-edge transport rather than end-to-end transport, which improves deployability.

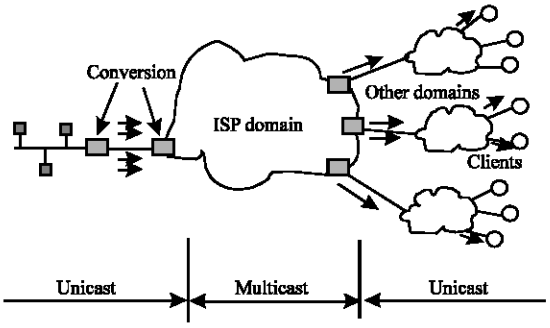

Fig 1: Stealth multicast overview

In order to provide reliability, the StMc architecture can be divided into two domains, namely Sender and ISP domain, In Sender domain, Sender sends only the unicast packets to the VGDM module. So the reliability can be given here only in the multicast domain not in unicast.So this Steadfast Stealth Multicast (SStMc) approach only concentrates on multicast part of the Stealth architecture.

Let's take a brief view of Reliablity in Multicast. Reliable multicast, also known as asynchronous Byzantine Agreement (Bracha and Toueg, 1985) is a fundamental communication protocol that underlies many forms of secure distributed computation. A reliable multicast protocol enables a node to multicast a message to a group of receivers in a way that ensures that all honest destination group members receive the same message, despite the contrary efforts of potentially

Corresponding Author: E.P. Srinivasan, Department of Information and Communication, Government College of Engineering, Salem, India

malicious group members and even a malicious multicast initiator. Even though feedback from receiver to the sender in the form of Acknowledgements (ACKs) or Negative Acknowledgements (NACKs) helps in achieving reliability, they are primary barrier for scalability as they may cause "Feedback Implosion".

A NACK-based protocol is preferable than an ACK-based protocol because only receivers detecting a packet loss send feedback to the sender. However, even with NACK-based protocol, "NACK implosion" is the serious problem for scalability. To reduce or eliminate the implosion of NACKs at the sender, some of current reliable multicast protocols implement NACK suppression mechanisms. When a receiver detects loss of a packet, it delays the transmission of its NACK for randomly chosen time. When a receiver's random timer expires, a corresponding receiver multicast a NACK to the sender and all receivers. When a receiver receives a NACK of a corresponding lost packet sent by another receiver, it quits sending a NACK. This mechanism is called "NACK suppression" which reduces total number of generated NACKs.

Motivation: Although stealth multicast can offer a significant mechanism for multicast deployment, stealth multicast is not ideal for applications which require reliable delivery. Stealth multicast model is created for connectionless UDP streams which follows send and forget mechanism. To support reliability concerns it should operate on connection oriented TCP. But present complexities in TCP, such as overheads in retransmission, adaptiveness and additional state maintenance makes it not suitable for multicast environments. In order to make Stealth Multicast as a reliable one, a layer of reliability protocol has to be added on top of it. So we have incorporated NACK feedback mechanism to achieve reliability in StMc architecture. Also the argument that routers should be kept as simple as possible and they should not take part in the end-end issues like loss detection and recovery is one other motivation factor that made in our approach end hosts to perform loss recovery tasks, finding that a loss has occurred and recovering from the loss in the most effective way in terms of recovery latency without the implosion and exposure problems.

PACKET LOSS RECOVERY PROCESS

In this study we have discussed about the overview of packet loss recovery process that could help you to easily perceive our forecoming implementation section. When a receiver detects the loss of a packet, it waits for a random time period (upper bounded by a timeout value) to listen for NACKs corresponding to the lost packet initiated by some other receiver and then multicasts a NACK to the multicast group (including the sender) for the packet if it does not hear a NACK in the meantime. Any receiver that hears the NACK and has lost the same packet suppresses its own NACK. When the sender receives a NACK, it knows that at least one receiver did not receive the packet and retransmits the packet.

Similar to NACK suppression, it can achieve loss notification suppression. Any receiver whose loss notification is not greater than the advertised value suppresses its own loss notification. A receiver whose loss notification is greater than the advertised value backs off for a random time (upper bounded by a timeout) and then multicasts its own loss notification if all the advertised notifications it has heard in the meantime are smaller than its own value.

SStMc approach tries to reduce the overhead on routers and tries to avoid the routers to participate in the end host issues. For reducing the overhead on the routers, the receivers can benefit from having some topology information that can be used during the loss recovery process. Routers that know about their neighbor receivers participating in the multicast session communicate with each other to recover from packet losses. To provide information to the routers about its neighbors an efficient route tracing mechanism with minimum overhead on routers which can be selected from the available pool of Multicast protocols based upon the underlying topology.

SStMc approach tries to develop a mechanism wherein not all the packets need to be recovered, as this could still result in acceptable quality. Whenever the receivers record more loss than the decided session loss threshold, the receivers start the loss recovery process. Similarly, the decision to send the NACK and SStMc packets are decided based upon the loss percentage as well as the recovery latency expected and the buffer size of the receivers.

Replier receiver selection: Finding the best replier for the recovery of lost packets in a multicast session has the following goals:

- Low recovery latency to the replier.
- Directing the request only to the receiver that has successfully received the requested packet.
- In case of global Loss, finding the root of the loss sub-tree for the starting point from where the repair packets will be sub-casted, so that only the receivers in the loss sub-tree receive the repair packet.

- Avoiding the unnecessary NACKS in the loss sub-tree, as these NACKs result in duplicate repairs being sent in the loss subtree.
- Allow only one NACK to escape from the loss sub-tree, to avoid an explosion of repair packets in the loss sub-tree leading to duplicate repairs.

The receiver closest to the multicast capable router (in terms of RTT to the router) which is selected by the network administrator acts as the group head for the other receivers attached to the same multicast capable router and is responsible for the recovery of lost packets experienced by the other receivers attached to same multicast router. This replier acts as the local replier (Rubenstein *et al*., 1998) for the other receivers attached to the same router. And the same receiver asks for the retransmission of any packets lost on trunk links in the tree as described later.

Loss detection and loss recovery: Each receiver keeps track of the sequence number of the packets that the sender sends to the multicast session. When the receiver receives a packet sequence number greater than the next expected packet sequence number it concludes that the packet is lost and starts the loss recovery process. If the loss occurs on the trunk link (Global loss), all the receivers down the link experience that loss. Each receiver starts its recovery mechanism to recover from the loss. When a replier receiver receives a NACK sent by the next downstream replier receiver, it checks for the availability of the packet with it. If it finds that the requested packet is available with it, then it encapsulates the requested packet into another packet addressed to the router which is assumed to be the root of loss subtree. The sender of the NACK to the replier receiver informs this router address.

The repair packet is then sent to the designated router. The router with its address as the destination address of the packet then strips off the outer packet and then sub-casts it down the tree. The receiver (receiver(s) other than the replier receiver(s)) experiencing the loss first waits for its local replier receiver to send a SStMc packet. When the receiver finds that the local replier has not sent any SStMc packet for the same loss after waiting for one RTT to the local replier, it concludes that the loss has occurred on its link to the router and also concludes that the lost packet is available with the local replier receiver and sends a NACK to the local replier asking for the lost packet. If the loss has occurred on leaf link(local loss) the above approach should not be carried out to avoid unnecessary repair packet instead repair should be unicasted.

SstMc approach uses three strategies (selection of repliers, defining forwarding points and directed multicast for retransmission) to closely approximate an optimal recovery scenario. In the optimal recovery scenario, the router directly below a data loss sends a request to the router immediately above the loss and the router immediately above the loss multicasts the lost packet to the affected branch. In this approach, each router selects one of the downstream links as a replier link. A receiver that detects a loss sends a request to the local router. Routers forward all incoming requests to the replier link, except the request that arrives on the replier link, which is forwarded upstream. If the router has no downstream replier links, the request is also forwarded upstream. If a request arrives on a downstream link other than the replier link, the router forwards to the Local Replier. The router inserts into the request its address and the identifier for the link on which the request arrived. The request is then forwarded downstream toward the replier. A replier with the requested data extracts from the request the address of the turning point and unicasts a reply directly to the turning point which then performs a directed multicast to the subtree rooted at the turning point.

In SStMc , DLR(Designated Local Replier/Repeater) advertisements are forwarded to the upstream router up by one level. In that level it compares with the existing list of DLR's. The DLR advertisement which contains the minimum RTT value will be advertised in the next level. This mechanism helps to increase the overall efficiency of the approach. It reduces the packet recovery latency and improves throughput of the link caused by the NACK forwarding mechanism in the earlier approaches.

SStMc IMPLEMENTATION

SStMc implementation introduces a new SStMc packet type and three new agents namely; sender agent, receiver agent and protocol agent. SStMc packet header and each of the agents are implemented as a C++ class and operate with some associative classes like timer classes, State information classes and Tcl linkage classes.

SStMc packet type: An SStMc packet is inherited from ns2 packet class, so it has all ordinary packet headers of ns2, like "IP header" or "common header". Besides, it has an SStMc header indicating the subtype of the packet. There are five subtypes, which are defined by the protocol. Some subtypes have an extra header for storing the required information for the protocol. Table 1 gives SStMc packet subtypes and headers of each.

Table 1: SStMc packet type

Packet type	Header	Definition
SStMc _PING	SStMc Header + upstream node + start time	SStMc Ping Packet
SStMc _DATA	COMMON Header + TYPE + session ID + DATA SeqNO.	SStMc DATA Packet
SStMc_MEND	COMMON Header + TYPE + session ID + REPAIR SeqNO.	SstMc Retransmitted DATA Packet
SStMc_NACK	SStMc Header + Sender ID + Group ID	SStMc NACK Packet
SStMc_N_CONF	SStMc Header + Negative CONF Seq.No.	SStMcNegative CONF packet

SStMc agents: In Network simulator, agents are defined as data structures that represent endpoints where packets are constructed or consumed. In our implementation, three kinds of Agents: SStMc -Sender, SStMc -Receiver and SStMc -Agent were developed by deriving from the base class Agent in NS2 to fulfill the whole requirement of SStMc. A new SstMc Error Model for controlled simulation of loss is also developed. For example, a sender agent creates the data packets and multicast them to the whole group. When a receiver receives a data packet, it reads the data ID of the packet and updates its receiver window. Each agent has a timer class for scheduling agent-specific tasks. A timer may start by a Tcl command or an incoming packet may trig a timer. For example the sender timer for sending periodic NACK packets starts when the Tcl command "start" is executed, while the ACK timer of a receiver starts when the first data packet receives. Each timer has a timer interval, which is bounded by an agent parameter or dynamically calculated during the simulation. All agent parameters like timer intervals, packet size or number of packets can be set to a value through the Tcl scripts by the users. A default value for each parameter should be defined in the necessary ns2 files.

SStMc sender agent: An SStMc sender agent works at the sender node of the multicast group. A SStMc-Sender agent sends application DATA packets as well as the PING packets in the multicast tree. Before sending the DATA packets this agent adds own protocol specific headers to the packets. Each time a DATA packet is sent at some constant interval of time, it adds to its header a sequence number for the data packet, which is used by the SStMc-Receiver agent to detect loss using gap-detection mechanism.

SStMc -NACK packet: When a SStMc Sender receives a NACK packet sent by the some other receiver, it sends the Negative CONF packet to the downstream router and it stores the NACK request in Pending Repair Data list. If the timeout of particular REPAIR data happens, sender sends the multicast packet to the multicast tree.

The parameters for sender agent can be initiated in Tcl code and the parameters used in our approach are:

- Heartbeat_interval: Time interval for sending PING packets
- Repair_delay: Time interval for sending retransmission packets 2 other classes that support the SStMc Sender class are
- Repair_data_timer: Timer for sending retransmissions.
- Heartbeat_timer : Time to send the Heart beat signals, it used to send periodic PING packets

Sender Agent also has 2 other objects ReplyItem and RepairdataItem . These 2 objects are used to buffer the NACK requests and repair data retransmissions.

SStMc-receiver agent: The core functionality of the working of the SStMc approach is designed and implemented in this agent. To add the functionality of a client in this design, implemented a Receiver buffer mechanism in this agent. This buffer is used to keep track of the packets received, packets lost, packets repaired as well as the maximum delay that the receiver should tolerate for the recovery of the lost packet. This buffer is also used to collect the statistics after the simulation run. The Receiver agent performs the following functions depending upon the type of packet it has received:

SStMc -dATA packet: When a SStMc -Receiver receives a DATA packet, it simply hands the DATA packet to the Receiver buffer to check for any lost packet. The receiver buffer informs the receiver that a loss has occurred. It starts a NACK timer to send the NACK for the loss of packet(s) depending upon the delay between the receiver and its replier receiver.

SStMc-PING packet: When a SStMc -Receiver receives a PING packet, it records the nearest multicast router and its interface through which it has received the packet and it updates the session's parameters. Then the SStMc -Receiver sends information packet containing information about itself if it is a Designated Local repeater, to its upstream router.

SstMc -NACK packet: When a SStMc Designated Receiver receives a NACK packet sent by the some other receiver, it queries the buffer to check for the availability of the packet in the buffer. If the buffer informs about the availability of the packet it then either encapsulates the repair packet or sends it directly to the requester through upstream router.

Compare to Sender Mechanism, here after receiving the NACK packet the Local repeater will not wait for the random amount of time to receive NACK packet from other nodes. This reduces the time taken for recovery process and it reduces the space overhead for storing NACK packets. If the NACK is previously available in the transmitted NACK list then backoff the NACK and reschedule the NACK timer. And the parameters for the agent used in our approach are:

- Max_NACK_conf_retries : Retransmission count for Negative Confirmation Packets
- Max_NACK_data_retries_: Retransmission count for Negative ACK Packets
- DLR_ : Used to set the Receiver as a DLR
- NACK_bo_ivl_: Sending NACK Interval
- NACK_rpt_iv:_Sending NACK Retransmission Interval
- NACK_repairdata_ivl : Time to wait for REPAIR Data

For a receiver a multicast session starts with the first incoming PING and finishes when the last packet has been received and the receiver buffer is empty. Receiver agent has one timer object

NACK_timer : Time to send the NACK packets to the upstream router.

SStMc agent: An SStMc agent works at all nodes of the multicast group. The SStMc -Agent agent mimics the behavior of a multicast capable router attached to a node. It works on top of the other multicast routing protocols being implemented in NS2. When a SStMc-Agent agent receives a packet of type, which is SStMc specific it carries out following functionalities:

SStMc -PING packet: When the SStMc -Agent receives a PING packet sent by the upstream router, it checks for the start of new session and it updates the session variables. If the Agent receives from any of the Local Repeater in downstream router it records the location of downstream local repeater and it sorts the Local repeaters based on the RTT. And it advertises the shortest RTT Local repeater to the upstream router.

SStMc -MEND packet: When a SStMc-Agent receives a MEND packet, it checks the legitimacy of the packet and if not so it discards the packet. Then the MEND Packet is forwarded to the links leading to the other downstream participants in the multicast session using the forwarding mechanism of the node to which it is attached.

SStMc -NACK packet: When a SStMc-Agent receives a NACK packet, it replies with the negative CONF packet. And it checks in the NACK list for transmitted NACK to the upstream, if the NACK is in the list then the NACK packet will be suppressed. Otherwise, it checks for the available Local repeaters in the list for the best link to transmit the NACK packet compare with the Upstream Router. If the Agent is associated with the Local Repeater then the NACK packet will be transmitted to the Local repeater for retransmission. and the parameters for the agent are:

- NACK_retrans_ival: Interval for Retransmission OF NACK Packets
- NACK_repairdata_ival: Interval to wait for the Repairdata Packet.

The agent also has 2 associative classes, State Info, Repair State. These 2 classes are used to Keep track of States of NACK Packets and REPAIR data. Apart from these classes SStMc Agent also has 2 timer objects.

- NACK_retrans_timer: Timer to keep track of retransmission of NACK packets NACK_ repairdata_timer:Timer to keep track of Reception of REPAIR data.

SStMc error model: In NS-2 (UCB), losses can be simulated by using different implementations of the error-models (list error-model, Select error-model and Periodic error-model to name a few). But none of the error-model provides the necessary functionality to simulate losses of some predefined loss percentage. SStMc ErrorModel, which drops packets on the links depending upon the loss percentage, set by the simulation setup on the sequence which is mentioned.

PERFORMANCE EVALUATION OF SStMc APPROACH THRO' SIMULATION STUDY

In simulation study, SStMc under different scenarios SStMc with Local Repeater, SStMc without Local Repeater, SStMc with forwarding mechanism, SStMc without forwarding mechanism have been simulated in Network Simulator-2(UCB) under different test cases, for comparing some characteristics of this protocol. For this purpose, several network topologies were created and the experiments designed for measuring different characteristics were performed on the same topology for each protocol. In order to obtain some quantitative results related to protocols, different evaluation metrics were defined. Details of the network topology, definition of the

evaluation metrics, experiment design, results obtained from these experiments and the comments on these results are presented henceforth.

Network and application model: In real networks, most of the packet losses are due to the buffer overflows in routers. In order to simulate these events in a simulation, background traffic, representing the normal traffic flow of the network, should be generated under the multicast traffic. But it is very difficult to generate proper background traffic and it brings considerably large processing load to the simulation. Instead of this, in Ns-2 (UCB), a user-defined packet loss rate can be assigned to each link. The simulator arbitrarily drops some packets passing over the link on an average rate that is defined by the user. In this simulation, all test cases are repeated under small and large loss rates, representing light and heavy background traffic, respectively. Furthermore, in order to observe the operation of SStMc on different network conditions, all evaluation metrics were measured with respect to varying loss rates.

Evaluation metrics: Evaluation metrics are measurable parameters about the protocols that provide quantitative results from simulation executions allowing commenting on different characteristics. The metrics used in this study have been defined as follows:

End-End Delay (EED): Is the average time elapsed since a packet is sent from the sender until the whole group has correctly received it.

Packet Recovery Latency (PRL): Is the average time between a packet drop being detected by a receiver and its repair packet reaching the receiver.

Packet Request Overhead (PRO): Is the additional load on intermediate nodes (routers) generated by the protocol. In order to measure this parameter, the number of request packets processed by each router is counted and the average value is calculated over all routers who participate in the multicast distribution tree. In SStMc request packets are the NACK packets, which are sent in case of a packet loss.

Source load: The ratio of the number of NACKs received by a source to the total number of NACKs generated in a simulation run.

Replier load: The replier load is defined as the ratio of the number of NACKs received by a replier to the total number of NACKs generated in a simulation run.

Both the source load and the replier load serve as indices of how well NACKs are distributed among group members. To evaluate the capability of controlling NACK implosion at the source, all three schemes in terms of the source load and the maximum replier load in a tree topology of depth 3 (in which the leftmost router is the source, the leaf routers are the receivers, the root is the core and each-leaf router has maximum of 2 children routers) was compared.

Experiment design: In this study we have defined about the selected N/W parameters and their range of variation. The selected parameters used in this study are as follows:

Group Volume (GV): It is the number of group members on the underlying topology. So that, group volume gives an opinion about the density of the multicast

Group Diameter (GD): In a network, the diameter can be defined as the distance between the end nodes. So it is directly related to the propagation delays of the links. In this study, delays were assigned by Tiers, while creating the topology. All values were multiplied by a coefficient, in order to obtain varying group diameters.

Packet Loss Rate(PLR): In order to simulate the background traffic on the network, a packet loss rate was assigned to each link. For observing the operation of the approach in different network conditions, varying loss rates were used.

Simulation results: In this study, results obtained from the simulations are presented in graphics showing the variation of the evaluation metrics against the network parameters.

End-end delay: In Fig. 2 and 3 the results obtained for end-end delay (in ms) against various volume of tree

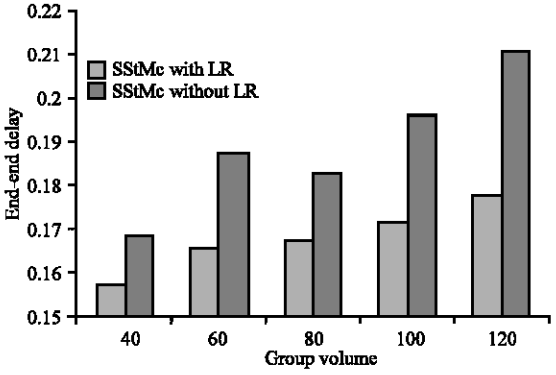

Fig 2: End-end delay vs GV (.01% PLR)

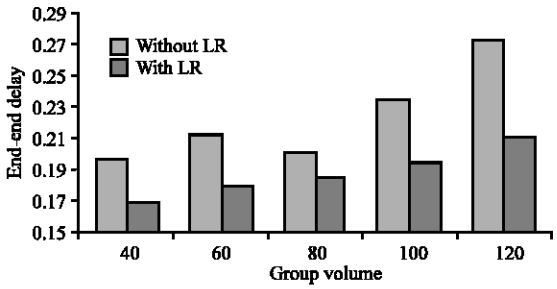

Fig 3: End-end delay vs GV(1% PLR)

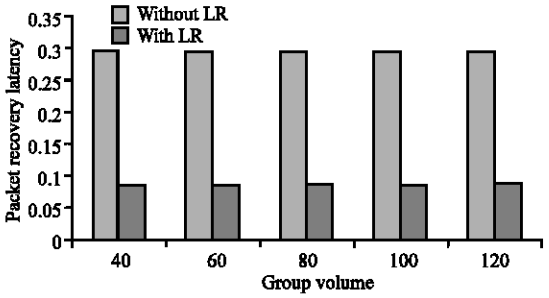

Fig 5: PRL vs GV (1% PLR)

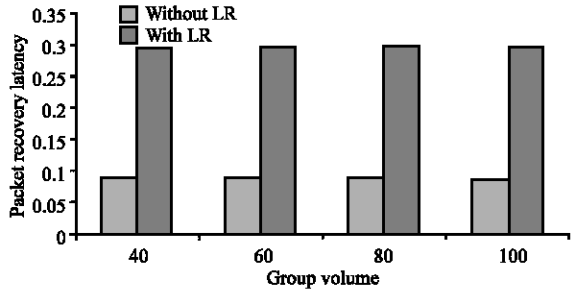

Fig 4: PRL vs GV (.01% PLR)

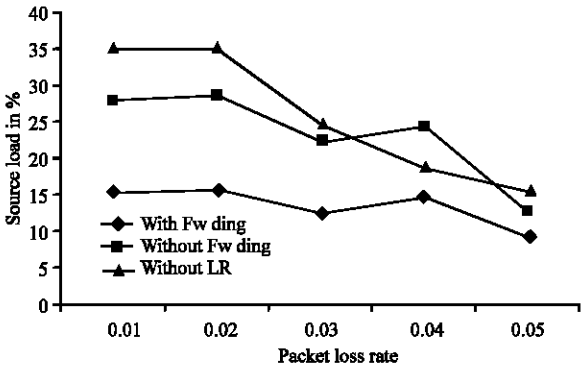

Fig 6: Source load % vs PLR in tree topology

topology (in nodes) are presented. As it is seen from Fig. 1 and 2, in both the cases, increasing group size causes a smooth increase on End-End Delay. It is expected, because as the number of member increases, the number of the members far from the sender will also increase and the number of packet drops will increase since these packets spread more on the links. In this case, results are very similar to each other. But in the case of High group volume the EED increases abruptly. In the high loss rate increase in percentage of EED is high compare to the low loss rate.

Packet recovery latency: Graphics given in Fig. 3 and 4, illustrate the variation of recovery latencies against different network parameters. As it is seen in Fig. 3 and 4, PRL for both the approaches remains almost constant as the group volume increases. Since the use of local retransmission, not being effected by growing group size is an expected result. In both small and large loss rate cases, the characteristics of SStMc graphics are similar, which means recovery mechanism of the protocol is not affected from the network condition.

As shown in Fig. 5 and 6, the sender load incurred in SStMc with forwading (SStMcFwd) is much smaller than other two approaches in all the chosen Error rates. As expected, SstMcFwd gives good results in case of high error rate as like other two's, but the decrease is in smoother flow. On the other hand, SStMcFwd has a higher maximum replier load than without forwarding approach. This is due to the fact that some of the requests that are directed to the source in that approach are now distributed among repliers in SStMcFwd. Note that the increase in the maximum replier load is smaller than the decrease in the sender load in SStMcFwd. This implies that the sender load is distributed among more than one replier. This demonstrates the capability of NACK implosion control in SStMc. Figure 7 shows the average per message Requests comparison among all three approaches. In this the graph clears that increase in error rate increases the number of requests. But the increase is smaller in SStMcFwd approach compare to other two approaches.

Figure 8 gives the performance comparison in terms of recovery latency among all three approaches. SStMc with the use of forwarding mechanism achieves better performance than other two approaches in the case of tree topology. This is because all intermediate routers select (under the assumption of equal link error probabilities) the shortest downstream path as the replier path, thus leading to better performance. In the case of random topology, the performance depends on the relative positions of the source, the receiver that suffers from data loss and the replier.

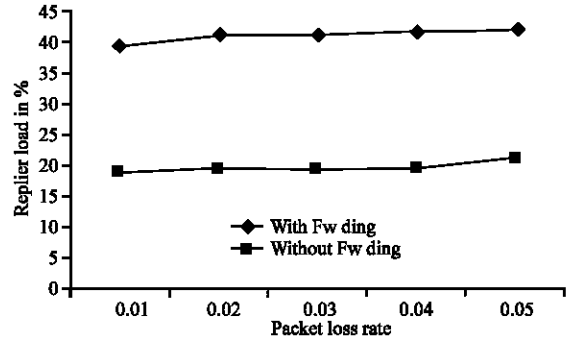

Fig. 7: Replier load % vs PLR in tree topology

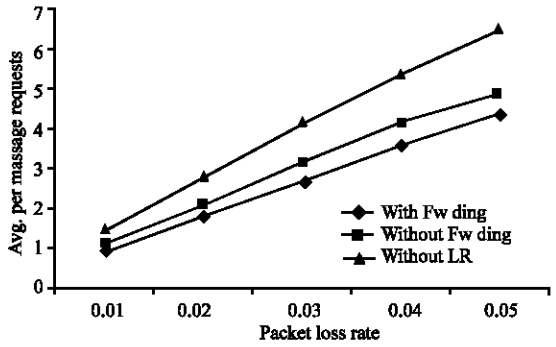

Fig. 8: Average per message requests arrival at sender vs PLR in tree topology

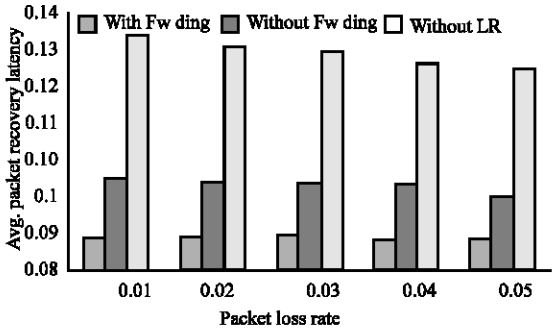

Fig. 9: Average PRL Vs PLR in tree topology

RELATED WORK

We had investigated about three main groups of reliable multicast approaches such as Sender-Initiated Protocols (Towsley *et al*., 1997) Receiver-Initiated Protocols(that includes error recovery mechanisms like Receiver-initiated error recovery (Pejhan *et al*., 1996) Hierarchical error recovery (Pejhan *et al*., 1996) and Forward Error Correction (FEC) mechanism (Rubenstein, 1997) (that includes Proactive FEC (Rubenstein *et al*., 1998) and Reactive FEC (Rubenstein *et al*., 1988). Moreover we had examined about Reliable protocols such as PGM (Speakman *et al*., 1998) SRM (Floyd *et al*., 1997) TRAM (Kadan sky *et al*., 2000; Nonnenmachr *et al*., 1998) and found relative to the above proposed research.

CONCLUSION

SStMc is designed to provide the reliability to the existing Stealth Multicast in the stealthy way i.e., only the intermediate Routers are modified such that it handles the stealth packet as an ordinary multicast packet. For this purpose a modified UDP is used which supports multicast in this Implementation. It differs from other reliable multicast protocols in its hybrid combination of unicast and multicast transmission. This practical approach has led to an implementation that has been evaluated over various sizes of topologies. These experiments have demonstrated its efficient use of network resources when compared to IP Multicast.

Local Recovery mechanism, Advertisement of Local Repeater to the upstream router, Selection of Local Repeater in Routers and Queuing decision in various Network elements complements the whole recovery mechanism in terms of Bandwidth utilization as well as Recovery latency.

REFERENCES

Bracha, G. and S. Toueg. Asynchronous consensus and broadcast protocols. J. ACM., 32: 824-840.

Floyd, S., V. Jacobson, C. Liu, S. McCanne and L. Zhang, 1997. A reliable multicast framework for light-weight sessions and application level framing. In: IEEE. ACM. Trans. Networking, 5: 784-803.

Kadansky, M., D. Chiu, J. Wesley and J. Provino. 2000. Tree-based Reliable Multicast (TRAM). Internet-Draft: Draft-kadansky-tram-02.txt. Work in progress.

Nonnenmacher, J., M. Lacher, M. Jung, G. Carl and E. Biersack, 1998. How bad is reliable multicast without local recovery. In: Proc. IEEE. INFOCOM'. New York, 98: 972-979.

Pejhan, S., M. Schwartz and D. Anastassiou, 1996. Error control using retransmission schemes in multicast transport protocols for real-time media, IEEE. ACM. Trans. Networking, 4: 413-427.

Rubenstein, D., 1997. Using packet-level FEC with real-time data delivery, Univ. Massachusetts, Amherst.

Rubenstein, D., J. Kurose and D. Towsley, 1998. Real-time reliable multicast using proactive forward error correction, In: Proceeding 8th International Workshop NOSSDAV.

Speakman, T., D. Farinacci, S. Lin and A. Tweedly, 1998. PGM reliable transport protocol specification, Internet Draft draft- peakman-pgmspec- 01.txt.

Striegel, A., 2004. Stealth multicast: A catalyst for multicast deployment, In: Proceeding of IFIP Networking, Athens, Greece.

Towsley, D., J. Kurose and S. Pingali, 1997. A comparison of sender-initiated and receiver-initiated reliable multicast protocols, IEEE. J. Select. Areas Commun., 15: 398-406.

UCB/LBNL/VINT Network Simulator-ns (version 2), available at:www.mash.cs.berkeley.edu/ ns/.