Performance Evaluation of RMTP Using NS & RMTP Integration in Current Networks

T. Asfourl and A. Serhrouchnil

ENST Ecole Nationale Supérieure des Telecommunications, Paris, France
{asfour, ahmed}@enst.fr

Abstract In this paper we focus on reliable multicast services. We present our work concerning the integration of RMTP in the network simulator NS. The resulting model allows us to simulate the performance of RMTP-based reliable multicast services. The simulation results of eight different scenarios presented in this paper show the poor performance of RMTP when used to multicast data to receivers in "heterogeneous" environments. Another question that we address in this paper is the scalability limitation of RMTP in spite of its hierarchical design. This limitation is due to several reasons like the use of the hierarchical acknowledgments (HACKS), the unlimited number of receivers that can be associated to a designated receiver, and the buffering space required in the sender and in the DRs when some or many receivers experience high loss rates. We discuss as well the challenges that face the integration of RMTP in current networks and its future coexistence with other multicast as well as unicast protocols.

1 Motivation

In nowadays networked, computerized world the word "reliability" is associated to almost every product and service. In protocol engineering terms the word "reliability" has more than one interpretation according to the requested service. We define the service in our context as a function of three main factors: time, data, and number of customers that request it. Reliability in time sensitive service is mainly a function of some or many temporal factors like delay and synchronization signals, while it is mainly a function of data factors like data size and loss rate in data sensitive services.

Services with two involved end points are called unicast services. In multicast more than two end points are involved in the communication. File transfer between a sender and a receiver is an example of a data sensitive unicast service. Videoconferencing and multi-party games are examples of time sensitive multicast services. In our paper we focus on reliable multicast services only.

Multicast services can be built directly using IP-multicast [6] at the network level. The main point of the architecture of this protocol is that senders and receivers don't need to know each other explicitly. It is sufficient to know one group address, which makes the distribution of data simple and scalable. In contrast, this "anonymity"

S. Rao and K.I. Sletta (Eds.): INTERWORKING 2000, LNCS 1938, pp. 147-159, 2000.

prevents sources from necessary feedback information about sent data, which means that services built directly on IP-Multicast are not reliable.

Transport layer multicast protocols have been proposed to achieve reliability, like RMTP 10,161, LGMP [9], MFTP 151, SRMTP [3,4), MDP [12] SRM [8] OTERS [5], etc. In our paper we focus on RMTP as an example of a tree-based reliable multicast protocol. A multicast service based on RMTP is defined as a function of three factors: throughput, delay, and number of receivers. The quality of this service is evaluated. RMTP is simulated using the network simulator NS, and the source code will soon be available for public use. The integration of RMTP in current networks is discussed from both commercial and technical points of view. The rest of this paper is organized as follows: In the first section we introduce RMTP and in the second section we present a simplified RMTP model that we have integrated in the network simulator NS, and we describe as well the simulation environments and the five resulting agents and their conceptual protocol flow. The third section describes the simulation scenarios and the fourth section shows the results of the simulation. In the fifth section we discuss the integration of RMTP in current networks and its coexistence with other transport protocols in the network. Finally the last section will conclude this paper and give some perspectives.

2 A Tree-Based Multicast Transport Protocol: RMTP

RMTP [10,16] is an example of a tree-based multicast transport protocol. Receivers are divided into subgroups depending on their geographical location and organized in a tree-like hierarchy. In each subgroup a designated receiver (DR) is responsible for processing receivers status messages and for local recovery of lost packets. The sender multicasts data to all the receivers, which share the same IP-Multicast address. Receivers send their status periodically to the associated DR, who in turn plays the role of the sender in its local region and multicasts lost packets, if possible, to the receivers in its local region. The DRs send their status periodically to the sender. The sender processes these status and performs global recovery. Figure 1 shows the topology of this protocol.

All receivers share one IP-Multicast address, and this is the address used by the sender for multicasting data. In each local subgroup, receivers share an IP-Multicast address that will be used by the subgroup associated DR for local recovery. Receivers send their status to the associated DR by unicast, and DRs in turn send their status also by unicast back to the sender. Figure 1 shows a one level hierarchy of DRs. It is quite possible to have multiple hierarchical levels, in other words a subgroup can be split into several subgroups, and in consequence we obtain a recursive hierarchical structure. In Figure 1 we see that the DRs send their status to the "Top Node" and not directly to the sender. The top node has a network control function and is mainly used to provide network managers with tools for monitoring and controlling the performance and network utilization of large applications.

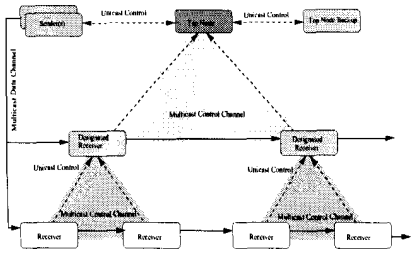

Fig. 1. RMTP Architecture

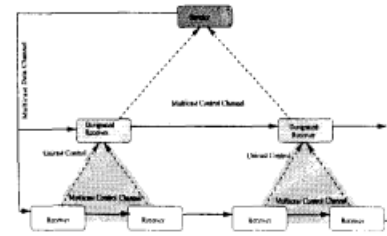

Fig. 2. RMTP Simulation Prototype Architecture

In previous version of RMTP this top node does not exist and status messages are directly sent to the sender. Any way the sender and the top node have a duplex unicast connection, and status messages will be aggregated or forwarded from the top node to the sender, and this is why we allow ourselves to say that DRs send status messages to the sender. RMTP receivers send HACKs (Hierarchical ACKS) to notify the reception and the loss of data packets. Additional options can be used like, NACKs and FEC for increasing scalability and average latency. In our paper we don't consider these options, and our simulation also does only consider the main RMTP features.

3 Simulation Model

In this section we begin by presenting the main features of RMTP that will be simulated and integrated in NS, and then we show the simulation model entities, as well as the relation between these entities.

3.1 RMTP Model

In the previous section, Figure 1 presents the architecture of RMTP as described in [11]. Our model, shown in Figure 2, excludes the top node and therefore the DRs send their status directly to the sender. The following main features will be supported by our model:

- The sender multicasts data to all receivers and DRs using the multicast Data channel.
- Receivers in each subgroup send HACKs periodically by unicast to the associated DR.
- DRs send HACKS periodically by unicast to the sender.
- In each subgroup the DR multicasts loss packets to the subgroup using the multicast control channel.
- The Sender achieves global recovery.

- A congestion control and avoidance algorithms,that will be described later, is used by the sender to prevent overloading the network.

3.2 Simulation Environment

The network simulator NS [7] is used to simulate RMTP. NS is an object oriented simulator from U.C. Berkeley/LBNL. It is written in C++, with an OTcl interpreter as a front end. The simulator is event driven and runs in a non real time fashion. A rich set of protocols in different layers have already been integrated in the simulator. Some examples for available protocols are FTP, TELNET, HTTP, TCP, SRM, and UDP. NS is a free software for network simulations, and it is possible for users to integrate new protocols and new classes into. Some Reliable multicast protocols like SRM [8), MFTP [15], and OTERS [5] have already been integrated in the simulator.

3.3 RMTP Agents

To integrate RMTP into NS we defined a set of classes with a class hierarchy shown in Figure 3.

The *RMTP* class contains common functions and variables for both RMTP sender and receiver. The *Snd* class presents the sender, and the *Rev class* presents the receiver. The designated receiver is the combination of the two classes *SndDr* that inherits from the *Snd* class and *RcvDr* that inherits from the *Rev class.* This combination is due to the dual role of the DR as a sender, for local recovery in its local region, and as a receiver at the same time. The same class hierarchy in C++ is mirrored in OTcl.

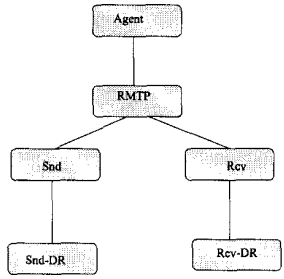

Fig. 3. RMTP: Class hierarchy within NS

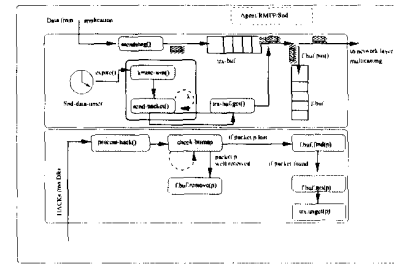

Fig. 4. Sender Block Diagram

RMTP class contains common functions and variables for both RMTP sender and receiver. *Snd* class presents the sender, and *Rev* class presents the receiver. The designated receiver is the combination of two classes *SndDr* that inherit the *Snd* class and *RcvDr* that inherit the *Rev* class. This combination is due to the binary role of the DR *as* a sender, for local recovery in its local region, and as a receiver at the same time. The same class hierarchy in C++ is mirrored in OTcl.

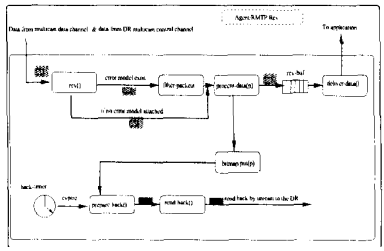

Fig. 5. Receiver Block Diagram

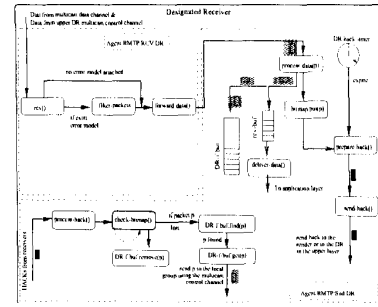

Fig. 6. Designated Receiver Block Diagram

3.4 RMTP Sender Functional Block Diagram

Figure 4 shows the conceptual protocol flow of the RMTP sender. The actual simulation of RMTP with NS is based on these diagrams. The sender is responsible for receiving data from the application layer and for splitting it into RMTP data packets. Packets are sent for buffering in what we call "transmission buffer", and there it waits for a send signal. The sender multicasts data packets at regular intervals defined by the parameter *interval-*. Regulating the send intervals is done by the use of a timer that expires after the delay *interval-,* and upon expiration the timer is rescheduled. The sender uses three variables to control the number of transmitted packets in each interval: Ws-, *cwnd_* and *use-win-* The maximum number of packets transmitted in *interval- is Ws-*. At the beginning of each interval the sender computes the number of lost packets *nb_lost_* in the previous interval by checking the bitmap of the last HACK processed. The following algorithm will be then applied for the calculation of *use-win-:*

```
if        nb_lost_ >  CongC_

    closecwnd();

else if nb_lost_ > CongA_

    mincwnd();

else

    opencwnd();

use _win_= window();
```

The *window()* function returns the minimum of *cwnd-* and Ws-, while *closecwnd()* closes the congestion window i. *e. cwnd_ = 1* when *nb_lost_* exceeds a congestion

control threshold *CongC_ , and mincwnd()* minimizes the congestion window to Ws-/2 when *nb_lost_* exceeds a congestion avoidance threshold *CongA_*, and finally *openewnd()* open the congestion window exponentially, i. e. *cwnd_ = cwnd- + 1*. The above algorithm is a TCP-like exponential slow-start algorithm for congestion control and avoidance.

After the number *k = use* -win- of packets to be sent is determined, the sender calls *send-packet() k* times, at each call the *send-packet()* gets a packet from the transmission buffer and sends it to the underlying network layer to be multicast using the multicast data channel. A copy of this packet is sent as well for buffering, in what we will call "final buffer" for possible retransmission requests. In our current study we don't consider the buffer size limitation question for both transmission and final buffers, we suppose that both buffers are big enough and no loss due to their size can be occur.

The RMTP sender receives DRs status messages, which arrive periodically, and aggregates these messages. Based on the bitmap included in the aggregated message the sender checks for well received and lost packets. If a packet is declared lost from any of the DRs the sender will look for this packets in its final buffer and reinsert it in the head of the transmission buffer. The sender will remove all well received packets from its final buffer.

3.5 RMTP Receiver Functional Block Diagram

Figure 5 shows the conceptual protocol flow of the RMTP receiver. The RMTP receiver listens to both the multicast data channel and the associated DR control channel. If an error model is attached to the receiver, packets will be filtered using this error model, and packets that are not dropped will be forwarded to the *process-data()* function. If no error model is attached to the receiver, received packets will be forwarded directly to the *process-data()* function. This function sends the received packet for buffering before it can be delivered to the application. It is as well responsible for setting the bit corresponding to the received packet to 1 in the bitmap. Hacks will be sent in fixed intervals controlled by a timer. At the expiration of this timer a HACK packet is prepared and sent to the associated DR, and the timer is rescheduled.

In our simulation model we choose to attach error models to receivers and not to the sender, which seems more logical, since errors can occur anywhere: in the sender, in network or in receivers, but finally it is up to the receiver to detect these errors. In other words our simulation model considers an accumulated error model attached to receiver.

3.6 RMTP DR Functional Block Diagram

Figure 6 shows the conceptual protocol flow of a RMTP designated receiver. The DR has a sender's and a receiver's role, and this is why we split it into two blocks. The first block represents the DR receiver block which has the main data reception function like end-point receivers. In contrast, the DR receiver block is not responsible

for HACK reception and sending, this function is moved to the DR sender block. The DR sender block completes the reception data process and keeps a copy of received packets in a DR final buffer for possible retransmissions. The DR sender block is responsible for sending HACKS periodically to the sender or to upper DRs. It receives HACKS from end point receivers and aggregates these hacks. Then it checks the bitmap in the aggregated HACK and removes well received packets from its final buffer. The DR sender block looks for lost packets in its final buffer, and found packets are retransmitted by multicast to the local group using the DR multicast control channel. In contrast to the main sender, the DR sender block has a limited final buffer size, by consequence packets may not be saved for possible retransmission in the final buffer.

4 Simulation Topologies

In order to create topologies for multicast scenarios we wrote a generic program that takes two parameters: the number of DRs and the number of receivers attached to each DR. Using NS facilities, bandwidth, delay and link characteristics are defined very easily which allows us to generate very different scenarios in order to simulate the behavior of RMTP in different network conditions. Eight different scenarios shown in Figure 7 will be presented in the paper. Each of the first six scenarios is a tree with the sender as the root, one internal node as the DR and 50 receivers. Different link characteristics are associated to each scenario, with error models attached to some or all receivers. Performance criteria are: Number of sender retransmissions *(Snd_ret),* number of DRs retransmissions *(DR-rett)),* average throughput for each receiver and DR (TH), and average delay per packet for each receiver and DR *(D).*

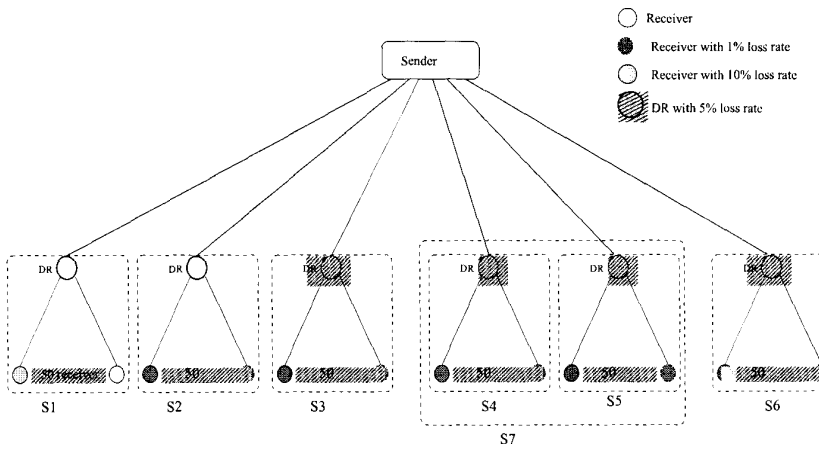

Figure 7. The topologies of all the nine scenarios

The first six topologies have the same multicast tree structure of one sender, one DR, and 50 receivers but with different link characteristics and attached error models. Table 1 give a summary of the different characteristics of the five scenarios.

Table 1. Bandwidth, loss rate and delay characteristics of the first six scenarios

Topo#	BW Snd-DR	D Snd-DR	BW DR-Rcv	D DR-Rcv	Error modell
1	512kb/s	40ms	64kb/s	10ms	10% first receiver
2	512kb/s	40ms	64kb/s	10ms	1% all receiver
3	512kb/s	40ms	64kb/s	10ms	1% all receivers, 5% DR
4	1Mb/s	40ms	512kb/s	10ms	1% all receivers, 5% DR
5	128kb/	40ms	28kb/s	10ms	1% all receivers, 5% DR
6	512 kb/s	200ms	64kb/s	30ms	11% all receivers, 5% DR

In the seventh scenario the sender will open a multicast session simultaneously with the fourth and fifth subgroups presented respectively in the fourth and fifth scenarios. While in the eighth scenario the sender opens a multicast session with two identical subgroups, each one has the topology of the subgroup in the fourth scenario. `All` scenarios have the same window and congestion control parameters as well as send data and `HACK` intervals. `A` summary of these parameters are given in Table 2. Finally, the simulation time of all simple topologies, i. e. which have only one subgroup, is 500s and for each combined topologies, i. e. which have more than one subgroup, is 250s. This is due to the heavy NS runtime load caused by memory and CPU consumption during simulation, which increases with the number of active agents.

Table 2. Send data and congestion control parameters

interval-	2 *s*	*CongC_*	30 packets
Rcvhack_int-	0.3 s	Ws-	10 packets
*Dr_hack_int_*	0.8 *s*	*CongA_*	20 packets
f -bu, f -7R	500 packets	*f -bu f _snd_*	not limited

5 Simulation Results

The retransmission simulation results in Table 3 show that the local recovery role of DRs becomes limited when the network conditions become "worse", and in consequence in these conditions it is up to the sender to retransmit packets. In the first topology the DR is very efficient and it performs 327 retransmissions and protects the sender that has to retransmit only 9 packets. The 9 retransmissions does not mean that there were exactly 9 lost packets which the DR was enable to recover. Since `HACKs` take time to arrive at the sender, during this time the sender keeps sending data. When a `HACK` arrives to the sender, the sender considers that the packets sent and not acknowledged in the current HACK are lost, even if it is not the case. This is why it is important to send HACKS with a certain frequency to avoid this "misunderstanding". On the other hand if DRs send HACKS with a very high frequency, a lot of HACKs will be useless since they have the same information as older HACKs. This is why it is very important to choose the *DR_hack_in_* very carefully. This relation between sender-DR delay and the number of sender retransmissions illustrates the increase *of*

Table 3. Retransmission Results for all scenarios

Topo#	Snd ret	DR1 ret	DR2 ret	simulation time
1	9	327		*500s*
2	141	1472		*500s*
3	727	3120		*500s*
4	116	541		*500s*
5	2284	370		*500s*
6	1021	2709		*500s*
7	816	135	374	*250s*
8	31	465	444	*250s*

the number *of* retransmissions from 727 in the third scenario to 1021 in the sixth one. The third and the sixth scenario have the same attached error models and the same bandwidth characteristics but differs in delay criteria. In consequence two very important questions have to be addressed: How often HACKS must be sent from DR to sender and respectively from receivers to DR, and what is the relation between HACKS frequency and the round trip time (RTT). In our previous illustration we referred to the 300ms delay between the sender and the DR in the sixth scenario as the responsible cause for the big number of sender retransmissions. It is rather the RTT difference that illustrates this result, but since the two topologies have the same link speeds and no other flows than the RMTP flow exist, we can base on the delay instead of the RTT.

The results of the third scenario show that despite the important role of the DR in local recovery (3120 retransmissions), the number of senders' retransmissions (727) is high in comparison to the second scenario (141). This result is due to the fact that the final buffer itself suffers from loss so more loss indications will arrive at the sender. To illustrate the impact of the error model attached to the DR we suppose that : r is the probability of packet loss associated to each receiver, and *q is* the probability of packet loss associated to the DR. If we consider that the underlying routing tree is the same as the virtual multicast tree then the probability P that a packet is received well by the *n* receivers is: $P = (1 - q)(1 - r)^n$. In consequence the probability *P'* that at least one loss indication per sent packet will arrive to the DR is given by the formula

$$P'=I-P=I-[(I-q)(1-r)'1] \quad (1)$$

For the second scenario where *(q = 0%, r = 1%, n = 50)* the value of P' is 39%, while for the third scenario *(q = 5%, r = 1%, n = 50) P'* is equal to 42%. Unfortunately the final buffer size in the DR is limited, and the DR itself suffers from loss which makes the DR unable to reply to all retransmission requests, and it sends in turn more *loss* indications to the sender. In Equation 1 we notice that when the number *of* receivers *n -> oc, P -~* 1, which means that every sent packet has to be retransmitted.

The retransmission results *of* the fourth scenario, which enjoys good bandwidth criteria, show that the sender "misunderstanding" of received HACKS is decreased. This is due to the small delay that a HACK takes to reach the sender. In contrast the retransmission results of the fifth scenario where the sender "misunderstanding" of DR's hack is increased and the DR does not fulfill its role of local recovery due to bad adjustment of the *Rcv_hack_int_* parameter in function of the RTT. The impact of the bad adjustment of *Rcv_hack_int_* and *DR-hack_int_* illustrates as well the big number of senders' and DR' retransmission in the sixth scenario where propagation delay criteria are the "worst" among other scenarios.

In contrast to the first six scenarios, the simulation time of the seventh and the eighth scenario is *250s.* The retransmission results show that the DRS' role of protecting the sender from receivers' feedback become limited when the number of subgroups increases and especially when subgroups have different criteria.

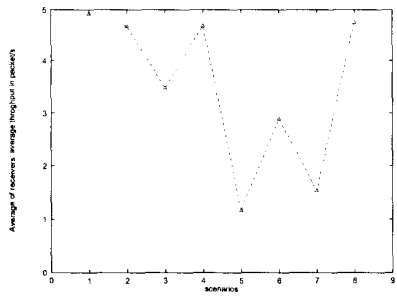

Fig. 8. Throughput results of all scenarios

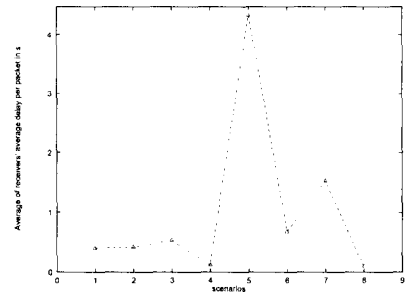

Fig. 9. Delay results of all scenarios

Figure 8 shows the average throughput results for all scenarios. We notice that the best average throughput is achieved in the first scenario *(4.9packet/s Ws-/2)* where only one receiver suffers from a 10% loss. The average throughput decreases to *4.6packet/s* due to the 1% loss rate in all receivers, and it decreases considerably to *3.5packet/s* with a 5% loss rate in the DR. Despite error models attached to the DR and to all receivers, good bandwidth criteria in the fourth scenario increase the throughput up to *4.67packet/s.* This is due to the quick recovery of lost packets. The impact of bandwidth limitation on the throughput average is very clear in the fifth scenario where the group achieves its worst throughput in comparison to other scenarios. In the six scenario where the multicast session is carried simultaneously to two bandwidth heterogeneous subgroups the average throughput is determined mainly by the worst subgroup. This result shows the limitation of RMTP performance when carrying one multicast session simultaneously to multiple subgroups with a big "intervariance" of link speed among subgroups. In the seventh scenario the sender carries one multicast conversation with two identical subgroups (identical to the subgroup of the fourth scenario). As we can see in Figure 8 the throughput of the multicast session of the seventh scenario is high and it is very close to the throughput of the fourth multicast session. In other words the average throughput of an RMTP multicast session carried to multiple subgroups with a small bandwidth "intervariance" does not decrease significantly with the increase of the number of subgroups.

Figure 9 shows the average of receivers' average delay per packet of the multicast session. The worst scenario in terms of delay results is the fifth scenario and this was expected due to the bandwidth limitation of the scenario's links. In contrast the average delay is minimal in the fourth and the eighth scenario due to good bandwidth criteria. We notice as well that the average delay increases when loss increases, and this is due to the increase of the number of necessary retransmission and in consequence lost packets experience more delay before arriving at receivers.

6 RMTP Integration in Current Networks

In the previous sections we noticed that RMTP faces critical problems when used to multicast data to heterogeneous and high big groups. The use of periodic HACK messages increases the network load as well as the processing load on both sender and receivers. We noticed as well the importance of adjusting the intervals at which receivers and DRs send their HACKS. The use of RTT in order to adjust these intervals becomes very critical in heterogeneous environments where receivers, respectively DRs, have different RTT values. In RTP/RTCP [13,14], where the RTP data protocol is responsible for data transmission and RTCP is responsible for control functions, the receivers send their status periodically to the sender. In contrast to RMTP and in order to improve the scalability, the rate of RTCP periodic state messages is adjusted in function of the number of receivers in the multicast session.

Another obstacle facing RMTP is the coexistence with other protocols in the network. Until today most applications are unicast and use TCP. So to be able to survive, RMTP must not affect TCP sessions. In other words there must be a fair share of network resources between the two protocols. A lot of reliable multicast protocols are proposed in order to have a good adaptation between services and protocols. Any way RMTP and any other multicast protocol have to take into consideration the interoperability with other protocols.

RMTP uses the strategy of recovery after loss was experienced. This strategy is a direct consequence of the current best-effort network. A lot of work in the QoS domain is achieved and we think that it is a good idea to give this protocol some extensions based on QoS criteria in order to deal with the reasons for loss and not with loss recovery. This is one of our objectives for further works in the multicast domain.

7 Conclusion

In this paper we presented our work concerning the integration of RMTP in the network simulator NS. The resulting agents are used for the performance evaluation of RMTP. The results of eight different scenarios show the poor performance of RMTP when used to multicast data to a heterogeneous set of receivers. The mechanism used by RMTP in order to split the set of receivers into subgroups is based on the geographical proximity of receivers. In addition to the problem of determining the geographical location of hosts in actual IP networks, this mechanism does not take into consideration the receivers' QoS, therefore the resulting subgroups might be very heterogeneous. The intrasubgroup heterogeneity affects the DR role in local recovery, and in consequence more retransmissions have to be done by the sender. The intersubgroup heterogeneity leads to a very poor throughput performance, since the throughput of "good" subgroups is controlled by the throughput of the "worst subgroup". On the other hand the hierarchical structure of RMTP is a good strategy for protecting the sender from receivers' feedback and avoiding the implosion problem. This protecting role becomes limited when the DRs become unable to fulfill their local recovery mission due to the heterogeneity of the set of receivers in the associated local group. We discussed as well some challenges that face the integration

of RMTP in current networks where TCP is still the dominant protocol. In a parallel work that we presented in [1, 2] we proposed a new mechanism for multicast group management that we called CGM. The main idea of this mechanism is to split the set of receivers in a multicast session into homogeneous subgroups based on their QoS criteria rather than their geographical distance like in RMTP. We used the CGM mechanism mainly with SRMTP [4] which stands for Scalable Reliable multicast Transport Protocols. The results that we have obtained encourage us to use the same mechanism with RMTP, and this is the subject of our current work in this discipline.

References

1. Taghrid Asfour, Stephan Block, Ahmed Serhrouchni, and Samir Tohme. Contractual group membership: a new mechanism for multicast group management. In *Proceedings of The Fifth IEEE Symposium on Computers and Communications* (ISCC'00), France, Antibes, July 2000.
2. Taghrid Asfour, Stephan Block, Ahmed Serhrouchni, and Samir Tohme. New *QoS* Based Mechanism for Heterogeneous Multicast Group Management. In *Proceedings of the 5th INFORMS Telecommunications conference, US,* Florida, March 2000.
3. Stephan Block. The Design of a Scalable Reliable Multicast Protocol. Technical report, Universität Stuttgart and Ecole Nationale Supérieure des Telecommunications (ENST), Paris, 1998
4. Stephan Block, Ken Chen, Philippe Godlewski, and Ahmed Serhrouchni. Some Design Issues of SRMTP, a Scalable Reliable Multicast Transport Protocol. In Helmut Leopold and Narciso Garcia, editors, *Multimedia Applications, Services and Techniques: Proc. 4th European Conference - ECMAST'99,* volume *1629 of Lecture Notes* in *Computer Science,* pages 423-440, Madrid, Spain, May 1999. Springer Verlag.
5. D. R. Cheriton and D. Li Cheriton. OTERS (On-Tree Efficient Recovery using Subcasting): A Reliable Multicast Protocol. In *Proceedings of 6th IEEE International Conference on Network Protocols (ICNP'98),* pages 237-245, Austin, Texas, October 1998.
6. S. Deering. Host Extensions for IP Multicasting. RFC 1112, August 1989.
7. Kevin Fall and Kannan Varadhan. NS Notes and Documentation. Technical report, UC Berkeley, LBL, USC/ISI and Xerox PARC, January 2000.
8. Sally Floyd, Van Jacobson, Ching-Guang Liu, Steven McCanne, and LiXia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking,* 5(6):784-803, December 1997.
9. Markus Hofmann. Impact of Virtual Group Structure on Multicast Performance. In C. Diot A. Danthine, editor, *From Multimedia Services to Network Services. Fourth International COST 237 Workshop,* volume 1356 of *Lecture Notes in Computer Science,* pages 165-180, Lisboa, Portugal, 1997. Springer Verlag.
10. Sanjoy Paul, John C. Lin, and Supratik Bhattacharyya. Reliable Multicast Transport Protocol (RMTP). *IEEE Journal ore Selected Areas in Communications, special issue on Network Support for Multipoint Communication,* 1996.
11. Sanjoy Paul, Brian Whetten, and Grusel Tascale. RMTP-11 Overview. Technical report, Talarian, Lucent Reuters, September 1999.
12. Luigi Rizzo and Lorenzo Vicisano. A Reliable Multicast data Distribution Protocol based on software FEC techniques. Technical report, Dipartimento di Ingegneria dell'Informazione, Università di Pisa and Department of Computer Science, University College, London, April 1997.
13. H. Schulzrinne. RTP Profile for Audio and Video Conferences with Minimal Control. RFC 1890, January 1996.

14.H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 1889, January 1996.
15.StarBurst Software. StarBurst MFTP-An Efficient, Scalable Method for Distributing Information Using IP Multicast. http://www.starburstcom.com/white.htm, 1998.
16.B. Whetten, M. Basavaiah, S. Paul, T. Montgomery, N. Rastogi, J. Conlan, and T. Yeh. RMTP-11 Specification. Internet Draft, Internet Engineering Task Force, April 1998.