Pathological Behaviors for RLM and RLC

A. Legout and E. W. Biersack
Institut EURECOM
B.P. 193, 06904 Sophia Antipolis, FRANCE
{legout,erbi}@eurecom.fr

Abstract

RLM [4] and RLC [7] are two well known receiver-driven cumulative layered multicast congestion control protocols. They both represent an indisputable advance in the area of congestion control for multimedia applications. However, there are very few studies that evaluate these protocols, and most of the time, these studies conclude that RLM and RLC perform reasonably well over a broad range of conditions.

In this paper, we evaluate both RLM and RLC and show that they exhibit fundamental pathological behaviors. We explain in which context these pathological behaviors happen, why they are harmful, and why they are inherent to the protocols themselves and cannot be easily corrected. Our aim is to shed some light on the fundamental problems with these protocols.

Keywords: RLM, RLC, Pathological behaviors, Congestion Control, Multimedia, Multicast, Cumulative layers.

1 Introduction

Multimedia applications will probably become some of the most popular applications in the Internet. One fundamental problem when introducing a new application in the Internet is to find an efficient way (for both the application and the network) to do congestion control. Cumulative layered multicast congestion control protocols are presented as the best solution for the dissemination of multimedia content to a heterogeneous set of receivers (see for instance [4, 7, 6, 3]). Therefore, these applications are the subject of active research.

Steven McCanne et al. introduced the first receiver-driven cumulative layered multicast congestion control protocol called RLM [4]. The behavior of RLM is determined by a state machine where transitions among the states are triggered by the expiration of timers (the join-timer and the detection-timer) or the detection of losses. The maintenance of the timers and the loss estimator are fundamental parts of the RLM protocol. In order to scale with the number of receivers, RLM needs an additional mechanism called *shared learning*. McCanne evaluated RLM for simple scenarios and only considered inter-RLM interaction. He found that RLM can result in high inter-RLM unfairness. Bajaj et al. [1] explored the relative merits of uniform versus priority dropping for the transmission of layered video. They found that RLM performs reasonably well over a broad range of conditions, but performs poorly in extreme conditions like bursty traffic. Gopalakrishnan et al. [2] studied the behavior of RLM for VBR traffic and show that RLM exhibits high instability for VBR traffic, has very poor fairness properties in most of the cases, and achieves a low link utilization with VBR traffic.

A TCP-friendly version of RLM, called RLC, was introduced by Vicisano et al. [7]. RLC is based on the generation of periodic bursts that are used for bandwidth inference and on synchronization points (SP) that indicate when a receiver can join a layer. The TCP-friendly behavior is mainly due to the exponential distribution of the layers that results in an exponential decrease of the bandwidth consumed (like TCP) in case of losses. While the exponential distribution of the layers is not a requirement for the TCP-like behavior if the protocol drops the layers in an exponential way, it considerably simplifies the protocol. We are not aware of any study considering another layer distribution. Vicisano found that RLC can be unfair with TCP for large packet sizes.

According to these previous studies, RLM and RLC seem to perform reasonably well in a broad range of cases. However, in this paper, we evaluate both RLM and RLC with very simple scenarios and show that they exhibit pathological behaviors. We explain in which context these pathological behaviors happen, why they are harmful, and why they are inherent to the protocols themselves and cannot be easily corrected. Our aim is to shed some light on the fundamental problems with RLM or RLC.

The paper is organized as follows. In section 2 we present the scenarios considered for the simulations. We discuss the results of the simulation for RLM in section 3, and for RLC in section 4. We conclude the paper in section 5.

2 Simulation Topologies

Fig. 1 shows the three topologies used to evaluate the behavior of RLM and RLC. A source and a receiver, when not

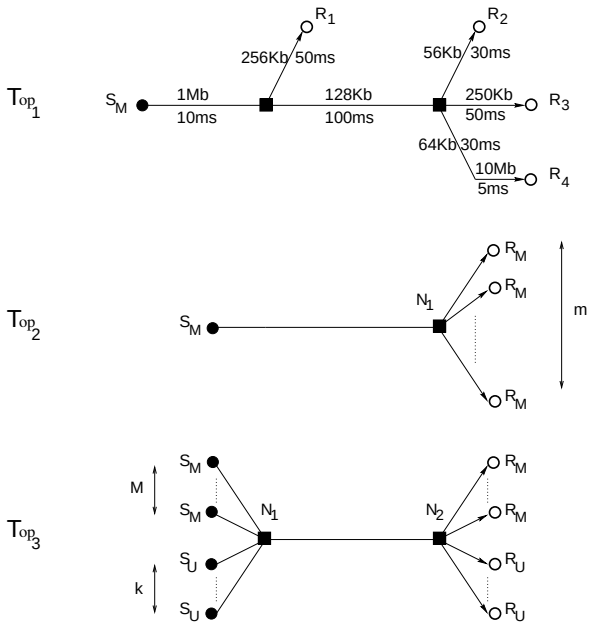


Figure 1: Simulation Topologies.

specified, refer to a RLM (or RLC) source and receiver, respectively. The first topology, Top_1, consists of one source and four receivers. We evaluate the speed, the accuracy, and the stability of the convergence in the context of a large heterogeneity of link bandwidths and link delays. The second topology, Top_2, consists of one source and m receivers. For all the simulations, the links (N_1, R_M) have a bandwidth uniformly chosen in $[500, 1000]$ Kbit/s and a delay uniformly chosen in $[5, 150]$ ms. We evaluate the scalability with respect to session size. The last topology, Top_3, consists of M multicast sources (with one receiver), and k unicast sources. For all the simulations, the links (S_M, N_1), (S_U, N_1), (N_2, R_M), and (N_2, R_U) have a bandwidth of 10 Mbit/s and a delay of 5 ms. We evaluate the scalability of the multicast protocol with an increasing number of multicast sessions and with an increasing number of unicast sessions. Also, we evaluate the fairness of the multicast protocol towards the unicast sessions.

We evaluate RLM and RLC using the ns [5] simulator. We use the following default parameters for our simulations: The multicast routing protocol is DVMRP (in particular graft and prune messages are simulated). We chose the packet size for all the flows (RLM, RLC, CBR, and TCP) to be 500 bytes.

RLM and RLC are designed for FIFO scheduling. However, we made all the simulations for both FIFO and FQ scheduling; in a given simulation, all the queues are either Fair Queuing (FQ) queues with a shared buffer or FIFO queues. The main reason for considering FQ scheduling is to evaluate how FQ impacts the behavior of RLM and RLC[1].

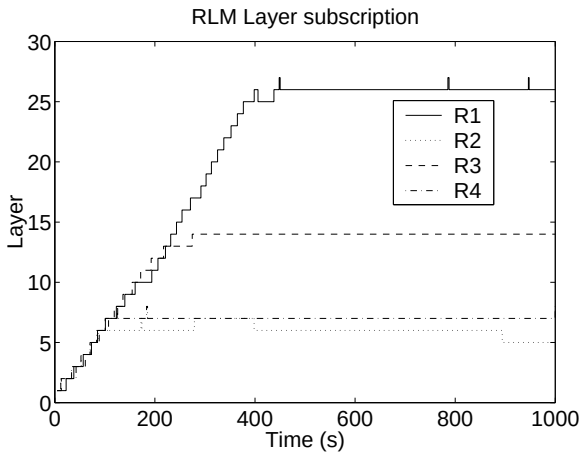
3 Pathological behaviors of RLM

We use the ns implementation of RLM with the parameters as chosen by McCanne in [4]. For all the simulations, the buffer size (or shared buffer size for FQ) is 20 packets. We run all the simulations for RLM for a duration of 1000 seconds.

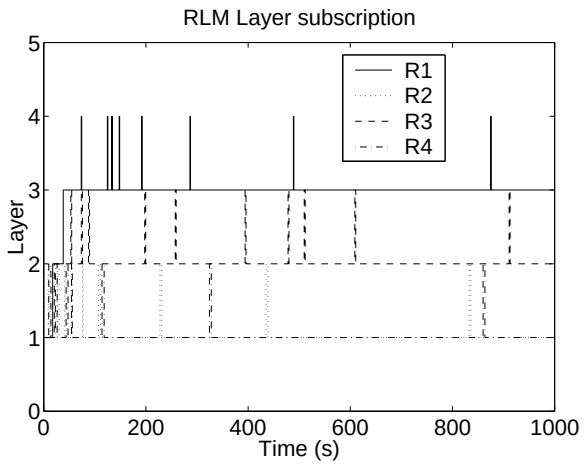
In several places, in this section, we consider thin layers (typically 10 Kbit/s or 20 Kbit/s layers granularity). We do not argue that thin layers are reasonable, practically applicable, etc. (Linda Wu et al. [8] study an architecture exploiting thin layers/streams). In fact, we use thin layers as a diagnosis tool; thin layers clearly exhibit pathological behaviors that still hold with coarse layers. However, directly using coarse layers does not allow to easily find if there is a pathological behavior and what is the reason of this pathological behavior.

The first simulation evaluates the speed, the accuracy, and the stability of RLM convergence on Top_1. We consider 10 Kbit/s layer granularity. We only present the results for FIFO scheduling (FQ scheduling gives the same result as, in this experiment, we have only one source). We see in Fig. 2(a) the very slow convergence time of RLM. Receiver R_1 needs more than 400 seconds to converge to the optimal rate. Moreover, the mean loss rate for this simulation is 3.2%. The 10 Kbit/s layers granularity is a tough test for RLM, and shows a pathological behavior of RLM in extreme cases. The slow convergence time is explained by the value of the minimum join-timer of RLM that is fixed to 5 seconds. The smaller the layer granularity, the slower the convergence. The significant loss rate is explained by the loss threshold of RLM set to 25%. With such small layers, we never enter in a congestion period where a receiver experiences a loss of more than 25% of the packets. Each receiver sees a persistent loss rate for the whole simulation that results in a mean loss rate of 3.2%. As a receiver can only do a join experiment if he does not see losses during a given period of time, there is very low number of join experiments in this simulation. We made another simulation with exponential layer sizes starting at 32 Kbit/s (the layer bandwidth distribution is {32,64,128,256,512,1024} Kbit/s) and give the results in Fig. 2(b). In this case RLM performs significantly better than in the previous case. The convergence time is reasonably fast. We clearly see the join experiments

[1] Another reason is the following: In [3] we introduce a new cumulative layered multicast congestion control protocol called PLM. This protocol requires a Fair Queuing network (i.e. a network where every queue is a FQ queue). In order to compare PLM with RLM and RLC, we must consider the same scenarios (the scenarios in this paper are a subset of the scenarios in [3]) and in particular, the same scheduling discipline. Moreover, as FQ improves the performance of RLM and RLC, it is fair to consider FQ for the comparison between these protocols and PLM. We find that PLM outperforms in all the cases RLM and RLC.



(a) Layer subscription for each RLM receivers, 10 Kbit/s layers.



(b) Layer subscription for each RLM receivers, exponential layers ($2^i \cdot 32Kbit/s, i = 0, 1, 2, 3, \cdots$).

Figure 2: Speed, accuracy, and stability of RLM convergence for a single session, Top_1.

that are, in this case, the main reason for a mean loss rate of 0.81%.

The second experiment evaluates the scaling of a single RLM session with respect to the number of receivers on topology Top_2. We consider 50 Kbit/s layer granularity. For this simulation, we consider the link (S_M, N_1) with a bandwidth of 280 Kbit/s and a delay of 20 ms. We start 20 RLM receivers at time $t = 5$ s then we add one receiver every five seconds from $t = 205$ s to $t = 225$ s, and at $t = 400$ s we add 5 more RLM receivers. The aim of this experiment is to evaluate the impact of the number of receivers on the convergence time and on the stability, and to

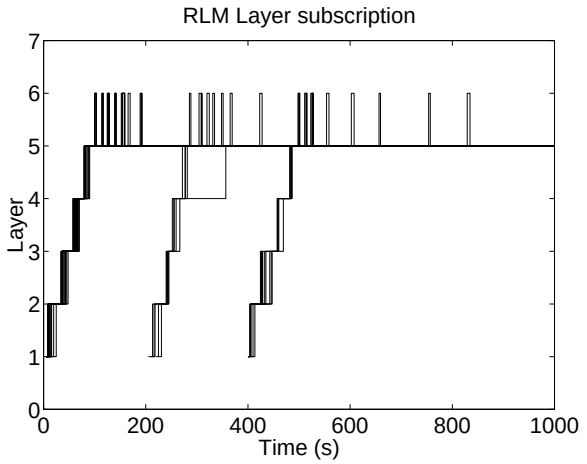


Figure 3: Scaling of a RLM session with respect to the number of receivers, Top_2.

evaluate the impact of late joins. We only present the results for FIFO scheduling (FQ scheduling gives the same result as, in this experiment, we have only one source). The most interesting event in Fig. 3 is the *receiver synchronization*. Due to the shared learning, receivers cannot join upper layers while there are some receivers subscribed only to lower layers. Indeed, the shared learning precludes a receiver to do a join experiment if there is a pending join experiment for a lower layer. Late joins can slow down the convergence time for RLM receivers. We did the same experiment with exponential layers and observed a similar behavior.

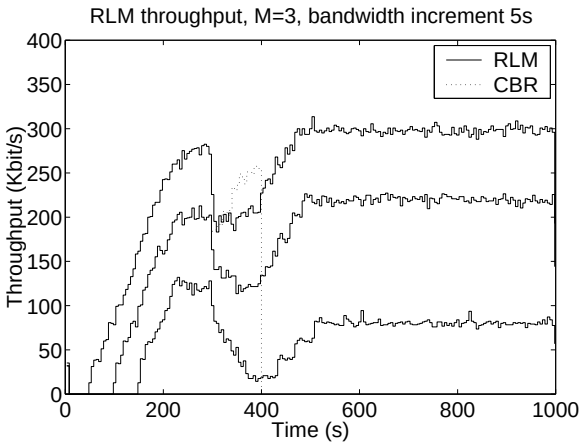
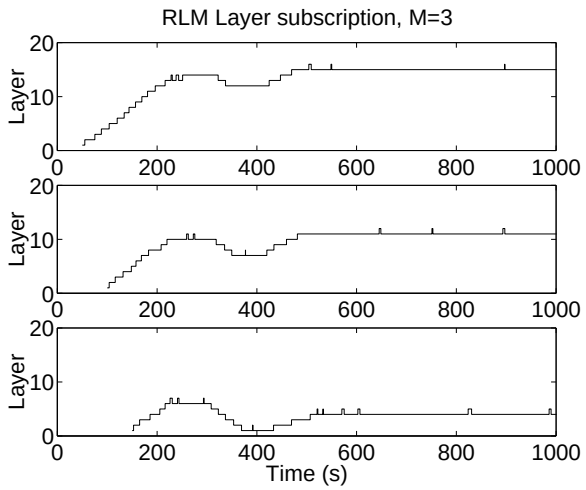
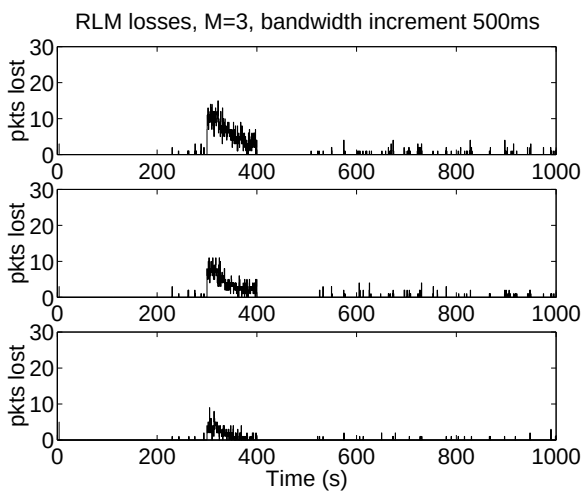


Figure 4: Mean throughput of RLM and CBR flows sharing the same bottleneck, FIFO scheduling, Top_3.

The third experiment considers a mix of RLM and CBR flows on Top_3. We consider a layer granularity of 20 Kbit/s. We comment this experiment for both FIFO and FQ scheduling. For FIFO scheduling, we consider $M = 3$ RLM sessions and $k = 1$ CBR flow. The bandwidth of



(a) Layer subscription of each RLM session.



(b) Loss rate of each RLM session.

Figure 5: RLM and CBR flows sharing the same bottleneck, FIFO scheduling, Top_3.

link (N_1, N_2) is $200 * M = 600$ Kbit/s and the delay is 20 ms. We start each of the three RLM receivers at times $t = 50, 100, 150$ s and the CBR source at time $t = 300$ s; we stop the CBR source at $t = 400$ s. The CBR source rate is 300 Kbit/s, half the bottleneck bandwidth. The aim of this scenario is to study in the first part (before starting the CBR source) the behavior of RLM with an increasing number of RLM sessions, and in the second part (after starting the CBR source) the behavior of RLM in case of severe congestion. When the CBR source stops we observe how fast RLM grabs the available bandwidth.

Fig. 4 shows the mean throughput of the three RLM sessions and Fig. 5(a) shows the layer subscription for the three RLM receivers. There is a slow convergence due to the small layer granularity. We see also a high unfairness among the sessions during the whole simulation. Moreover, the high period of congestion (when the CBR source sends packets) results in a large number a losses for the RLM sessions (see Fig. 5(b)). When the CBR source starts and creates congestion, the RLM sessions start dropping layers. However, the process of dropping layers with RLM is very conservative (sluggish) and induces significant transitory losses (see Fig. 5(b)). Indeed, a receiver can only drop one layer per detection-timer period. The mean loss rate is 2.3% in this experiment. We note the same effect as in experiment one: The small layers result in losses that never exceed the loss threshold (see Fig. 5(b)), therefore never result in a layer drop, and result in a very low number of join experiments (see Fig. 5(a)). We did the same simulation with exponential layers. As expected, the large layer granularity results in a higher reactivity for RLM. When the CBR source starts, RLM reacts fast to the congestion by dropping one layer (dropping one layer is enough in this case to avoid congestion). The resulting mean loss rate is reduced to 1.4%. However, RLM results in a very high unfairness in case of exponential layers as well. The first session gets roughly 500 Kbit/s, the second gets roughly 100 Kbit/s, and the third session must drop all the layers.

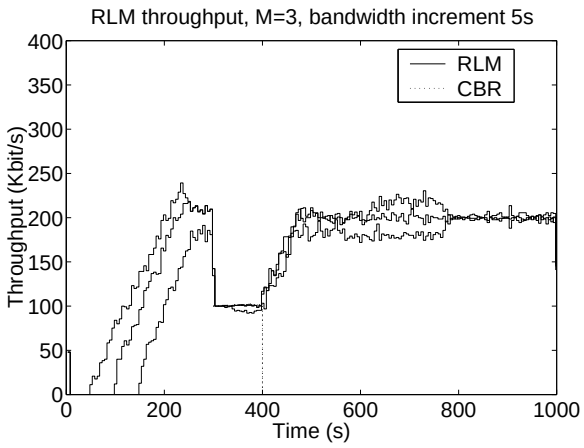


Figure 6: Mean throughput averaged over 5s intervals, FQ scheduling, Top_3.

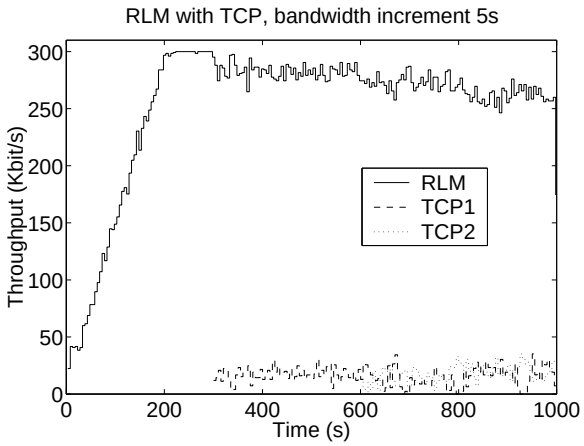
For FQ scheduling, we consider $M = 3$ RLM sessions and $k = 3$ CBR flows. The bandwidth of link (N_1, N_2) is $200 * M = 600$ Kbit/s and the delay is 20 ms. We start each of the three RLM receivers respectively at time $t = 50, 100, 150$ s. We start the CBR sources at time $t = 300$ s and stop the CBR sources at $t = 400$ s. The rate of each CBR source is 500 Kbit/s. We choose as many CBR sources as RLM sessions to simulate severe congestion. Indeed, with FQ, the only way to create congestion is to sig-

nificantly increase the number of sessions. In this case, the three CBR sources grab half of the bottleneck bandwidth.

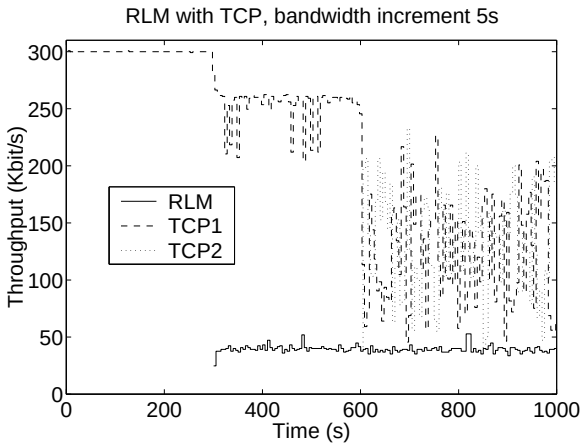
Fig. 6 shows the mean throughput for the three RLM sessions. The most noticeable point, compared to the FIFO scheduling case, is the good fairness among the RLM sessions. However, even with FQ scheduling, the fairness is not ideal (see Fig. 6 between $t = 400$ s and $t = 800$ s). The mean loss rate for this simulation is 4.6%. As FQ enforces fairness among all the flows, the RLM flows can not grab more bandwidth than their fair share. While, with FIFO scheduling a RLM flow can grab more bandwidth than its fair share from the CBR flow. Therefore, the RLM receivers experience more losses with FQ than with FIFO. We do not notice any other significant difference compared to the FIFO scheduling case. We did the same simulation with exponential layers and observed a good fairness among the RLM flows (according to the layer granularity). RLM reacts fast to the congestion and the resulting mean loss rate is lower than 1%.

The fourth experiment considers a mix of one RLM session and TCP flows on Top_3. We consider $M = 1$ RLM session and $k = 2$ TCP flows and a layer granularity of 20 Kbit/s. The bandwidth of link (N_1, N_2) is $100 * (M + k) = 300$ Kbit/s and the delay is 20 ms. We do all the simulations for FIFO and FQ scheduling. In a first set of simulations, we start RLM first at $t = 0$ s, then TCP1 at $t = 300$ s, and TCP2 at $t = 600$ s. In a second set of simulations, we start TCP1 first at $t = 0$ s, then RLM at $t = 300$ s, and TCP2 at $t = 600$ s. For FQ scheduling, the simulations do not bring any new results compared to the previous experiment. In summary, with FQ scheduling, RLM shares fairly the bandwidth with TCP (according to the layer granularity), and experience a transitory period of congestion when a new TCP flow starts. This period of congestion results in a significant loss rate (from to 2% to 8% according to the simulation scenario) with 20 Kbit/s layer granularity, and in a low loss rate (around 0.5% for all the scenario) with exponential layers.

In the following we consider FIFO scheduling. Fig. 7 shows the mean throughput averaged over 5 seconds intervals of the RLM and TCP flows for FIFO scheduling. When RLM starts first it grabs all the available bandwidth. TCP can only achieve a very small throughput (see Fig. 7(a)) due to the large RLM loss threshold of 25%. Indeed, when RLM is in the steady state, a receiver must experience a loss rate higher than the loss threshold to drop a layer. However, TCP is not able to create a large enough congestion and therefore fails to grab bandwidth from RLM. When RLM starts after TCP1, RLM is not able to grab bandwidth from TCP. This is due to the join experiment process of RLM. When a RLM receiver does a join experiment and experiences losses during this join experiment, he infers that it can not join this layer. Moreover, in order to do a join experiment, a receiver must not see any loss during a given period of time. The



(a) RLM session starts first.



(b) RLM session starts after TCP1.

Figure 7: Mean throughput of RLM and TCP flows sharing the same bottleneck, FIFO scheduling, Top_3.

key point is: whereas a RLM receiver in steady state needs a 25% loss rate to drop a layer, a RLM receiver needs only one loss to infer than he cannot join a layer or to preclude a join experiment (the reader can refer to [4] for all the details about the RLM protocol).

In conclusion, we found several pathological behaviors of RLM: i) The minimum join timer gives a large lower bound to the speed of convergence; ii) The high loss threshold can result in a high mean loss rate. Moreover, it results in a very aggressive behavior when competing with TCP. iii) The shared learning results in receiver synchronization; iv) The join experiment process results in a very conservative behavior when competing with TCP flows; v) The conservative drop process (one layer dropped per detection-timer)

results in extended transient periods of losses in case of congestion.

Each of these pathological behaviors is very hard to correct as the parameters involved are the result of complex tradeoff. The minimum join timer is a tradeoff between the speed of convergence of the frequency of the join experiments. The loss threshold is a tradeoff between a conservative and a reactive behavior in case of loss. One solution is for both, the join timer and the loss threshold, to dynamically adjust these parameters according to the network conditions. However, that requires complex network inference mechanisms: an additional (large time scale) bandwidth inference mechanism to infer if a receiver needs to add several or only few layers to reach the equilibrium; an additional congestion inference mechanism to determine if the congestion is heavy (one needs to drop several layers to reach the equilibrium) or light (one needs to drop only one layer to reach the equilibrium). These questions need further research. The shared learning and the join experiment process are foundations of the RLM protocols and cannot be changed without redesigning the whole protocol. Finally, the conservative drop process is necessary for RLM to avoid over-reaction to losses and is, therefore, very hard to tune.

4 Pathological behaviors of RLC

We use the ns implementation of RLC with the parameters as chosen by Vicisano in [7]. We identify behaviors in the ns version of RLC that are not conform with the description of RLC in [7]. We do not correct these behaviors as we do not know if they are intended by the authors or if they are the result of a bug. We always take into account these behaviors in our simulations and discuss them when they impact the results. The main *peculiar* behavior is that RLC drops the current layer when it experiences losses during a burst, whereas, according to [7], RLC should stay at the current layer and just infer that it cannot join an upper layer.

RLC can be considered a TCP-friendly version of RLM with the improvement of the synchronization points (data packets with a special flag) and a new bandwidth inference mechanism based on periodic bursts. In fact, we show that both the synchronization points and the periodic bursts lead to pathological behaviors, and that the RLC behavior is very sensitive to the queue size.

For all the simulations with RLC, we just indicate the rate B_0 of the base layer L_0. The rate of layer L_i is $B_i = 2^i \cdot B_0$. If not specified, the default buffer size (or shared buffer size for FQ) is 20 packets. The first simulation evaluates the speed, the accuracy, and the stability of RLC convergence for Top_1. The rate of the base layer is 32 Kbit/s. We only present the results for FIFO scheduling (FQ scheduling gives the same result as, in this experiment, we have only one source). The queue size is 15 packets. Fig. 8

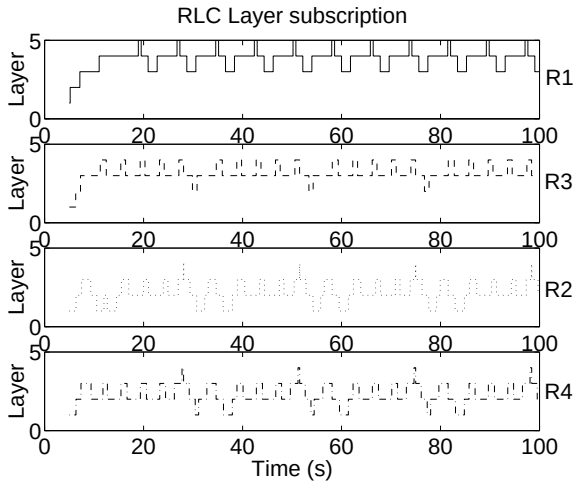


Figure 8: Layer subscriptions for a single session, 4 receivers, Top_1.

shows the layer subscription for the RLC receivers. The solid line is for R_1, the dashed line is for R_3, the dotted line is for R_2, and the dashed-dotted line is for R_4. This simple experiment shows one of the most fundamental problem with RLC. For instance, when R_1 subscribes to layer 4, he receives 256 Kbit/s. As his bottleneck bandwidth is 256 Kbit/s, he experiences no loss. The source sends periodically a burst that doubles, over a short period of time, the sending rate to allow the receiver to infer if he can join a higher layer. However, the burst does not make the queue overflow, and R_1 infers that he can join layer 5. After a short period of time, R_1 will experience a large number of losses and will drop the layer. For receiver R_1, we observe a cascade drop from layer 5 to layer 3. However, this cascade drop is due to the peculiar behavior pointed out at the beginning of the section. Indeed, just after dropping layer 5, the queue will remain full (as the bottleneck bandwidth is equal to the layer 4 rate), the source will generate a burst that makes the queue overflow as the queue is already full before the burst. The receiver will experience losses during the burst and due to the peculiar behavior will drop the layer 4. We can explain the behavior of the other receivers in the same way. The periodic erroneous bandwidth inference leads to a mean loss rate up to 13%.

This experiment shows a fundamental pathological behavior of RLC. RLC's bandwidth inference is based on the generation of periodic bursts that aim to reduce the transitory period of congestion due to join experiments (see [7] for more details). To succeed, the burst must make the queue overflow when there is not enough bandwidth to accommodate a new layer. However, queue overflow happens in our simulations *only* for a very judicious choice of the queue sizes, which is impossible to do in a real network. As the bandwidth inference does not succeed, the receivers pe-

riodically join a layer when there is not enough bandwidth available to add this layer. That leads to periodic congestion and periodic losses.

To avoid cascade drop, RLC uses a deaf period of fixed length after dropping a layer during which it does not drop layers. However, this deaf period reflects the delay between the time the receiver sends a leave request and the time the receiver sees the effect of the leave request on the bottleneck router. This value varies highly over time and for different receivers. As the join experiments are sender-based in RLC, there is no way for a receiver to infer the appropriate duration for the deaf period without adding a complex protocol. This is a significant weakness of RLC as a correct static choice of the deaf period can be very difficult. If RLC must drop several layers to react to a severe period of congestion, the deaf period will significantly slow down the drop process. However, we note that with exponentially distributed layers, dropping one layer is most of the time sufficient to react to congestion.

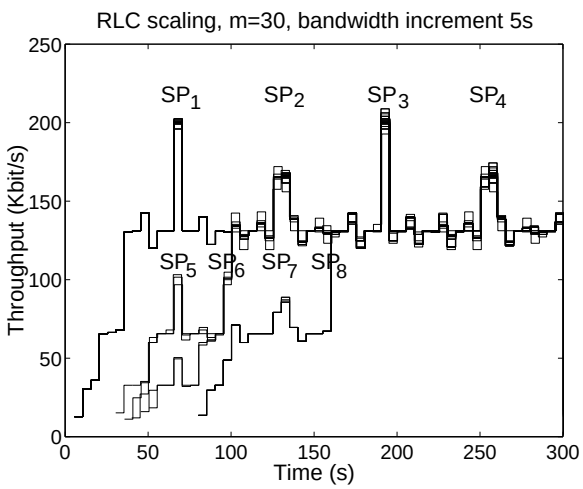


Figure 9: Scaling of a RLC session with respect to the number of receivers, Top_2.

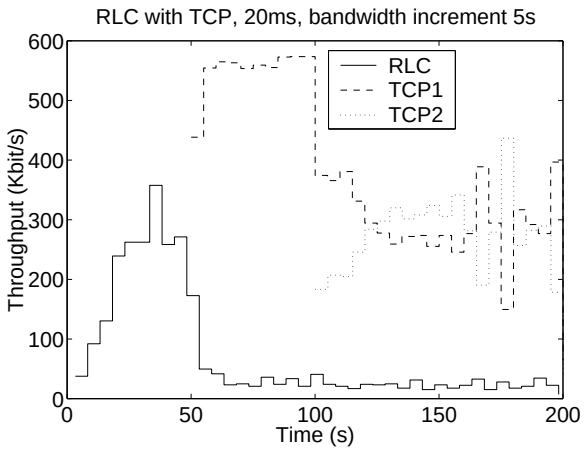
The second experiment evaluates the scaling of a single RLC session with respect to the number of receivers on topology Top_2. For this simulation we consider the link (S_M, N_1) with a bandwidth of 250 Kbit/s and a delay of 20 ms. The queue size is 10 packets. We start 20 RLC receivers at time $t = 5$ s then we add one receiver every five seconds from $t = 30$ s to $t = 50$ s, and at $t = 80$ s we add 5 RLC receivers. The rate of the base layer is 8 Kbit/s. The aim of this experiment is to evaluate the impact of the number of receivers on the convergence time and on the stability, and to evaluate the impact of late joins. We only present the results for FIFO scheduling (FQ scheduling gives the same result as, in this experiment, we have only one source). A receiver can only increase his number of layers at synchronization points (SP) if no losses are experienced during the burst preceding that SP. The distance between two SPs doubles at each layer, and the SPs at layer L_i are a subset of the SPs at layer L_{i-1} (see [7] for more details). Fig. 9 shows the mean throughput for all the receivers. We first note that the small throughput oscillations around the mean throughput are due to the succession of periodic burst and silent period that slightly increases or decreases the mean throughput averaged over 5 seconds intervals. The annotations SP_i indicate the occurrence of some relevant SPs. In this simulation, the bandwidth inference using bursts never succeeds, i.e. the bursts never make the queue overflow, and the receivers join an additional layer that the network cannot support. We observe a new pathological behavior of RLC. Between $t = 30$ s and $t = 50$ s late joiners start. Around $t = 60$ s, at the synchronization point SP_5, some late joiners join layer 5[2] and the others join layer 4. But, as the synchronization point SP_1 is synchronized with SP_5, the first receivers (that join at $t = 5$ s) join layer 6 that cannot be supported. This results in a period of congestion that is misinterpreted by the late joiners who drop a layer. The late joiners can only subscribe to the highest layer supported at SP_6, which is not synchronized with an upper layer SP. We observe the same pathological behavior with the late joiners that start at $t = 80$ s. This pathological behavior significantly slows down the convergence speed. We note that, even if the burst succeeds in inferring the available bandwidth, the same problem persists. Indeed, if the burst (to join layer 6) makes the queue overflow, the first receivers will infer that they cannot join layer 6 and they will stay at the current layer at SP_1. However, the late joiners cannot join an upper layer at SP_5 as they will see losses, shared among all the layers, due to the burst on layer 5.

With the parameters choice in [7], the SPs are exponentially spaced. At layer i, the distance between the SPs is $2^i \cdot 8 \cdot \frac{s}{B_0}$, where s is the packet size and B_0 is the throughput of the base layer. For $B_0 = 16$ Kbit/s and $s = 256$ bytes, the distance between the SPs at layer i is roughly 2^i seconds. For instance, a receiver can only join layer 6 every 64 seconds. The exponentially spaced SPs can significantly slow down the convergence of the receivers to the highest layers.

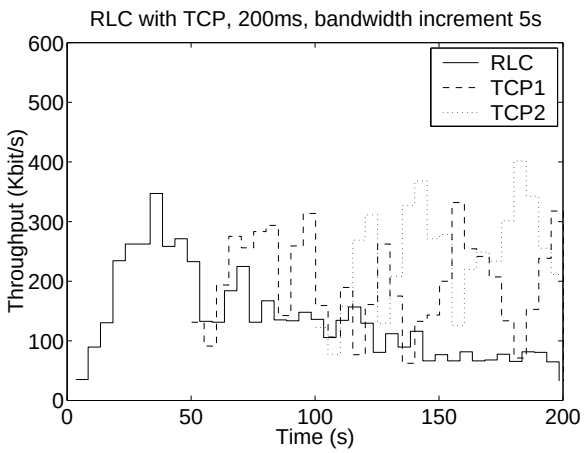
We did a third experiment that considers the same scenarios than the third experiment for RLM. We do not give plots for this experiment as it does not exhibit pathological behaviors. For this experiment, RLC performs reasonably well. The RLC sessions share fairly the bandwidth among each other and adapt reasonably fast to the transitory period of congestion produced by the CBR source(s). The mean loss rate for all the scenarios range from 0.6% to 2.9%.

The fourth experiment on Top_3 considers a mix of RLC and TCP flows. We consider $M = 1$ RLC session and $k = 2$ TCP flows. The bandwidth of link (N_1, N_2) is $200 * (M +$

[2]In this simulation layer 4 corresponds to a 64 Kbit/s, layer 5 corresponds to 128 Kbit/s, and layer 6 corresponds to 256 Kbit/s.



(a) The delay of the link (N_1, N_2) is 20 ms.



(b) The delay of the link (N_1, N_2) is 200 ms.

Figure 10: Mean throughput of RLC and TCP flows sharing the same bottleneck, Top_3.

$k) = 300$ Kbit/s and the delay varies from 20 ms to 400 ms. The rate of the base layer is 16 Kbit/s. We start RLC at $t = 0$ s, TCP1 at $t = 50$ s, and TCP2 at $t = 100$ s. We did all the simulations for both FIFO and FQ scheduling. For FQ scheduling, we do not see any pathological behavior and do not present the plots. In this case, RLC shares fairly (according to the layer granularity) the bandwidth with the TCP flows. For these scenarios, the mean loss rate range from 0.7% to 1.6%.

Now we comment the simulations for the fourth experiment with FIFO scheduling. Fig. 10(a) shows the mean throughput averaged over 5 seconds intervals for the RLC and TCP flows when the delay of the link (N_1, N_2) is 20 ms. When TCP1 starts, RLC drops to layer 1 and then oscillates between layer 1 and layer 2. When TCP2 starts, we do not notice any particular behavior for RLC. This experiment shows that RLC can be very conservative compared to TCP. Fig. 10(b) shows the same experiment than previously except that the delay of the link (N_1, N_2) is 200 ms. We see that when TCP1 starts, RLC shares fairly the bandwidth with TCP1. When TCP2 starts, RLC gets a lower bandwidth than the two TCP flows. In a last experiment (we do not give the plot), we increase the delay of the link (N_1, N_2) to 400 ms. For this experiment, RLC fairly shares the bandwidth with TCP1 and TCP2. The explanation of this behavior is simple. The TCP cycle (i.e. the time between two losses) is shorter with a small RTT than with a large RTT. As a consequence, the smaller the RTT is, the larger the number of losses RLC experiences in a given time interval. As the RLC throughput is function of the number of losses, the higher the number of losses, the lower the RLC throughput.

In conclusion, we observed several pathological behaviors of RLC: i) The bandwidth inference mechanism based on burst leads to a high number of losses and does not succeed to make the queue overflow. ii) The synchronization points, as distributed in RLC, can significantly reduce the speed of convergence of late joiners. iii) The claimed TCP-friendly behavior of RLC results in a very conservative behavior of RLC compared to TCP.

Moreover, we cannot easily correct any of these pathological behaviors. For the periodic bursts to succeed, we must know how long the burst should persist in order to make the queue overflow. That requires a mechanism close to a bandwidth inference mechanism that renders to periodic burst useless. Moreover, the static choice of the burst length is a very difficult tradeoff between the probability to make the queue overflow and the amount of periodic congestion (and losses) generated. The pathological behaviors ii) and iii) raise new questions: Does RLC still achieve its claimed TCP-like behavior with non exponentially distributed layers? What is the influence of the placement of the SPs on the RLC behavior? These questions are for future research.

5 Conclusion

In this paper, we have evaluated RLM and RLC on simple scenarios. We show that both protocols exhibit pathological behaviors. We discuss which part of the protocol leads to a given pathological behavior and explain that most of the time these pathological behaviors are difficult to correct. We note that most of the problems come from the bandwidth inference mechanism used that is responsible for transient periods of congestion, instability, and periodic losses.

In [3] we present a new cumulative layered multicast congestion control protocol, called PLM, based on the generation of packet pairs (PP) to infer the available bandwidth. Bandwidth inference using PPs does not have any of

the weaknesses of the bandwidth inference mechanisms of RLM and RLC, and PLM outperforms in all the cases RLM and RLC. However, PLM requires a Fair Queuing network. With a FIFO network, traditional solutions like RLM and RLC are still necessary, but require improvements of the bandwidth inference mechanism. We hope that this paper contributes to identify the fundamental problems of these protocols, and will stimulate research to improve these protocols.

References

[1] S. Bajaj, L. Breslau, and S. Shenker, "Uniform versus Priority Dropping for Layered Video", In *SIGCOMM'98*, Vancouver, British Columbia, CANADA, September 1998.

[2] R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson, and C. J. Sreenan, "Stability and Fairness Issues in Layered Multicast", In *NOSSDAV'99*, 1999.

[3] A. Legout and E. W. Biersack, "PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes", In *Proceedings of ACM SIGMETRICS'2000*, Santa Clara, California, USA, June 2000.

[4] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven Layered Multicast", In *SIGCOMM 96*, pp. 117–130, August 1996.

[5] NS, UCB/LBNL/VINT Network Simulator - ns (version 2), http://www-mash.cs.berkeley.edu/ns/.

[6] T. Turletti, S. Fosse-Parisis, and J. Bolot, "Experiments with a Layered Transmission Scheme over the Internet", Research report, INRIA, B.P.93, Sophia-Antipolis Cedex, France, November 1997.

[7] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like Congestion Control for Layered Multicast Data Transfer", In *Proceedings of IEEE INFOCOM*, San Francisco, CA, USA, March 1998.

[8] L. Wu, R. Sharma, and B. Smith, "Thin Streams: An Architecture for Multicasting Layered Video", In *NOSSDAV'97*, pp. 173–182, St Louis, Missouri, USA, May 1997.