PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes (extended version)[*]

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

March 15, 2000

Eurecom Technical Report

Abstract

A major challenge in the Internet is to deliver live audio/video content with a good quality and to transfer files to large number of heterogeneous receivers. Multicast and cumulative layered transmission are two mechanisms of interest to accomplish this task efficiently. However, protocols using these mechanisms suffer from slow convergence time, lack of inter-protocol fairness or TCP-fairness, and loss induced by the join experiments.

In this paper we define and investigate the properties of a new multicast congestion control protocol (called PLM) for audio/video and file transfer applications based on a cumulative layered multicast transmission. A fundamental contribution of this paper is the introduction and evaluation of a *new* and *efficient* technique based on packet pair to infer which layers to join. We evaluated PLM for a large variety of scenarios and show that it converges fast to the optimal link utilization, induces no loss to track the available bandwidth, has inter-protocol fairness and TCP-fairness, and scales with the number of receivers and the number of sessions. Moreover, all these properties hold in self similar and multifractal environment.

Keywords: Congestion Control, Multicast, Capacity inference, Cumulative layers, Packet Pair, FS-paradigm.

1 Introduction

Multimedia applications (audio and video) take a growing place in the Internet. If multiple users want to receive the same audio/video data at the same time, multicast distribution is the most efficient way of transmission. To accommodate heterogeneity, one can use a layered source coding where each layer is sent to a different multicast address and the receivers subscribe to as many layers as their bottleneck bandwidth permits. The multimedia applications can easily be transmitted using cumulative layers: each higher layer contains a refinement of the signal transmitted in the lower layers. File transfer to a large number of receivers will probably become an important application for software updates or electronic newspaper posting. Multicast distribution with a cumulative layer coding based on FEC (see [21]) is an efficient solution to this problem.

A receiver-driven cumulative layered multicast congestion control protocol (RLM) was first introduced by Steven McCanne [11] for video transmission over the Internet. RLM has several benefits: First, the cumulative layered transmission uses a natural striping of the multimedia streams and achieves a very efficient use of the bandwidth as the different layers do not contain redundant information but refinements. Second, the receiver-driven approach allows each receiver to obtain as much bandwidth as the path between the source and this receiver allows. However, RLM has also some fundamental weaknesses. RLM is not fair (neither inter-RLM fair nor TCP-fair), RLM converges slowly to

[*] A shorter version of this paper appears in the Proceedings of ACM SIGMETRICS 2000, 17th - 21st June 2000, Santa Clara, California, USA.

the optimal rate and tracks this optimal rate slowly (after a long equilibrium period, RLM can take several minutes to do a join experiment and so to discover bandwidth that became recently available), finally RLM induces losses.

A TCP-friendly version of a cumulative layered receiver-driven congestion control protocol was introduced by Vicisano [22]. Whereas this protocol solves some fairness issues it does not solve issues related to the convergence time (the subscription to the higher layers is longer than the subscription to the lower layers), and does not solve the issues related to the losses induced.

We want a congestion control protocol for multimedia and file transfer applications that guarantees fast convergence and does not induce losses. We introduce in [8] a *paradigm* to devise end-to-end congestion control protocols only by taking into account the requirements of the application (congestion control protocols tailor-made to the application needs). Our paradigm is based on the assumption of a Fair Scheduler network i.e. a network where every router implements of PGPS-like [13] scheduling with longest queue drop buffer management. We show that this assumption is practically feasible. Moreover this paradigm only assumes selfish and non-collaborative end users, and guarantees under these assumptions nearly ideal congestion control protocols.

To practically validate the theoretical claims of our paradigm, we devise a new multicast congestion control protocol for multimedia (audio and video) and file transfer applications. We devise a receiver-driven cumulative layered multicast congestion control protocol that converges fast to the optimal rate and tracks this optimal rate without inducing any loss. The cornerstone of our congestion control protocol is the use of packet pair (PP) to discover the available bandwidth (see [7]). We call the protocol *packet Pair receiver-driven cumulative Layered Multicast* (PLM).

In section 2 we introduce the FS-paradigm. Section 3 presents the PLM protocol. We evaluate PLM in simple environments to understand its major features in section 4 and in a realistic environment in section 5. We compare PLM with other layered multicast congestion control protocols in section 6. Section 7 explores the practical validation of the theoretical claims of the FS-paradigm, section 8 presents the related work, and we conclude the paper with section 9.

2 The FS Paradigm and Its Application

A paradigm for congestion control is a model used to devise new congestion control protocols. A paradigm makes assumptions and under these assumptions we can devise compatible congestion control protocols; compatible means that the protocols have the same set of properties. Therefore, to define a new paradigm, we must clearly express the *assumptions* made and the *properties* enforced by the paradigm.

In the context of a formal study of the congestion control problem as a whole, we defined the Fair Scheduler (FS) paradigm (see [8]). We define a Fair Scheduler to be a Packet Generalized Processor Sharing scheduler with longest queue drop buffer management(see [13], [19], [4], and [3] for some examples). For clarity, we make a distinction between the assumption that involves the network support – we call this the *Network Part* of the paradigm (NP) – and the assumptions that involve the end systems – we call this the *End System Part* of the paradigm (ESP).

The assumptions required for the FS paradigm are:

- For the NP of the paradigm we assume a *Fair Scheduler* network, i.e. a network where every router implements a Fair Scheduler.

- For the ESP, the end users are assumed to be selfish and non-collaborative.

The strength of this paradigm is that under these assumptions we can devise nearly ideal end-to-end congestion control protocols (in particular fair with TCP), i.e. different protocols that have the following set of properties: stability, efficiency, fairness, robustness, scalability, and feasibility. The main constraint of the FS-paradigm is the deployment of FS routers. However, we explained in [8] how and why this deployment is feasible per ISP. The only assumption that the paradigm makes on the end-user is its selfish and non-collaborative behavior (we do not require these properties, we just do not need anything else to achieve the properties of an ideal congestion control protocol).

We consider for the PLM congestion control protocol multimedia (audio and video) and file transfer applications. The requirements of multimedia applications are very specific. We must identify how to increase the satisfaction of a user of a multimedia application: (i) A user wants to receive the highest quality (high throughput, low number of losses) and (ii) wants to avoid frequent modifications in the quality perceived. The requirement of a file transfer application is a small transfer time (high throughput, low loss rate). In the next section we define mechanisms that allow to meet these requirements. We devise the PLM protocol with the FS-paradigm. We assume a Fair Scheduler network and all the mechanisms at the end-system try to maximize the satisfaction of the users (selfish behavior). What is remarkable with this paradigm is that whereas the end-users are selfish, we achieve the properties of an ideal end-to-end congestion control protocol.

To understand why the FS-paradigm is of great benefit to devise congestion control protocols we take a simple example (examples specific to PLM are presented in section 7). First we have to identify the requirements of a user (i.e. how to increase his satisfaction). For our purpose we suppose that the user wants to converge fast to an optimal rate and to be stable at this optimal rate. The FS-paradigm guarantees that even a *simple* congestion control algorithm will converge and be stable at this optimal rate. This is the cornerstone of the practical application of the FS-paradigm. We do not have to devise complicated congestion control protocols to converge to the optimal rate and to stay at this optimal rate. Of course, the FS-paradigm does not give this simple algorithm, but if one finds a simple algorithm that converges to the optimal rate, this algorithm leads to a congestion control protocol that will converge fast and will be stable.

PLM is a demonstration of the practical application of the FS-paradigm. We have a simple mechanism, Packet Pair, and do not introduce any complicated mechanism to improve the convergence nor the stability. We discuss in section 7 some implications of the FS-paradigm on the design of PLM.

3 Packet Pair Receiver-Driven Layered Multicast (PLM)

Our protocol PLM is based on a cumulative layered scheme and on the use of packet pair to infer the bandwidth available at the bottleneck to decide which are the appropriate layers to join. PLM assumes that the routers are multicast capable but does not make any assumption on the multicast routing protocol used. PLM is receiver driven, so all the burden of the congestion control mechanism is at the receivers side. The only assumption we make on the sender is the ability to send data via cumulative layers and to emit for each layer packets in pairs (two packets are sent back-to-back). We devise PLM with the FS-paradigm, in particular we assume a Fair Scheduler network. In the next two sections we define the two basic mechanisms of PLM: The receiver-driven cumulative layered multicast principle and the packet pair mechanism.

3.1 Introduction to the Receiver-Driven Cumulative Layered Multicast Principle

Coding and striping multimedia data onto a set of n cumulative layers $L_1, \cdots, L_n$ simply means that each subset $\{L_1, \cdots, L_i\}_{i \leq n}$ has the same content but with an increase in the quality as i increases. This kind of coding is well suited for audio or video applications. For instance, a video codec can encode the signal in a base layer and several enhancement layers. In this case, each subset $\{L_1, \cdots, L_i\}$ has the same content and the higher number of layers we have, the higher quality video signal we obtain. For audio and video applications, the cumulative layered organization is highly dependent of the codec used. Vicisano in [21] studies two cumulative layered organizations of data, based on FEC, for file transfer. In this case the increase in the quality perceived is related to the transfer time.

Once we have a cumulative layer organization it is easy for a source to send each layer on a different multicast group. In the following, we use indifferently the terminology *multicast group* and *layer* for a multicast group that carries a single layer. To reap full benefits of the cumulative layered multicast approach for congestion control, a receiver-driven congestion control protocol is needed. When congestion control is receiver-driven, it is up to the receivers to add and drop layers (i.e. to join and leave multicast group) according to the congestion seen. The source has only a passive role, consisting in sending data in multiple layers. Such a receiver-driven approach is highly scalable and does not need any kind of feedback, consequently solves the feedback implosion problem.

One fundamental requirement with cumulative layered congestion control is that all the layers must follow the same multicast routing tree. In Fig. 1 we have one multicast source and two receivers. The source sends data on

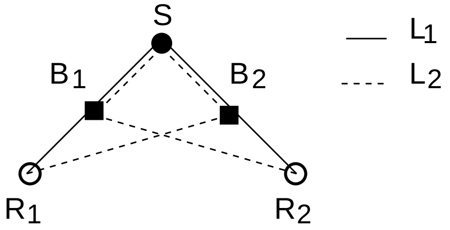


Figure 1: Example of two layers following two different multicast trees.

two layers, each layer following a different multicast tree. Imagine congestion at the bottleneck B_1, receiver R_1 will infer that it should reduce its number of layers. As we use cumulative layers we can only drop the highest layer: L_2. However this layer drop will not reduce congestion at bottleneck B_1. When the layers do not follow the same multicast routing tree, the receivers can not react properly to congestion.

One of the weakness of the cumulative layered congestion control protocols is the layer granularity. In fact this granularity is not a weakness for audio and video applications. Indeed, it makes no sense to adjust a rate with a granularity of, for instance, 10 Kbyte/s, if this adjustment does not improve the satisfaction of the users. Moreover a user may not perceive fine-grain quality adjustments. We strongly believe that a standardization effort should be made on the characteristics of the perceived quality compared to the bandwidth used. These characteristics are codec dependent. Imagine, for the purpose of illustration, the following classification for audio broadcast: quality 1: 10 Kbit/s (GSM quality); quality 2: 32 Kbit/s (LW radio quality); quality 3: 64 Kbit/s (quality 2 stereo); quality 4: 128 Kbit/s (FM radio quality); quality 5: 256 Kbit/s (quality 4 stereo). It is clear, in this example, that there is no benefit in creating an intermediate layer as this layer will not create a significant modification in the perceived quality. If a user does not have the minimum bandwidth required, he can not connect to the session. Who can have satisfaction in listening a sonata of J.S. Bach with a lower quality than GSM quality? Therefore, we do not believe that the layer granularity is a weakness for congestion control for audio and video applications.

For file transfer applications, the layer granularity leads to a higher transfer time (dependent on the layer distribution) than rate/window based solutions in case of small homogeneous groups. However, a sender rate/window based multicast congestion control protocol must adapt to the slowest receiver. In case of high heterogeneity of bandwidth, the layered scheme is clearly the most efficient. It is not the purpose of this paper to study how much bandwidth should be given to each layer.

All the previous multicast layered congestion control schemes are based on CBR layers. The protocols like RLM heavily rely on the accurate knowledge of the throughput of each layer to infer the available bandwidth. If a layer is added whereas this layer uses less than its regular throughput, the join experiment becomes meaningless. In the case of PLM, as the bandwidth inference is not based on join experiments but on PP estimate, even if a layer uses less than its regular throughput, the bandwidth inference is still accurate. Therefore, PLM can accommodate VBR layers if we can define an upper bound of the bandwidth required by reaches by each layer. PLM will simply assume that each layer is CBR with a bandwidth defined by the upper bound of the VBR layers. However, this solution can be very inefficient in case of VBR layers with a large standard deviation and a small mean throughput. One solution is to have a protocol that dynamically adapt its layers to the VBR layers. Managing dynamic layers is very complex and is an area for future research. In fact, we do not see any strong argument in favor of VBR encoding compared to CBR encoding. Even if CBR encoding can result in a slight decrease in quality, the ease of exploitation of CBR layers is a strong argument in favor of CBR codec. The study of a codec to get the appropriate layer distribution is beyond the scope of this paper.

In the following, we simply assume that we have a given set of CBR layers, without making any assumptions on the layer granularity.

3.2 Receiver-Driven Packet Pair Bandwidth Inference

The packet pair (PP) mechanism was first introduced by Keshav [7] to allow a *source* to infer the available bandwidth. We define a *receiver-driven* version of packet pair. Let the *bottleneck bandwidth* be the bandwidth of the slowest link on the path between the source and a receiver. Let the *available bandwidth* be the maximum bandwidth a flow can obtain. We assume a network where every router implements a Fair Scheduler. If a source sends two packets back to back (i.e. a packet pair), the receiver can infer the available bandwidth for that flow from the spacing of the packet pair and the packet size. By periodically sending packet pairs, the receiver can track the available bandwidth. The main feature of the PP bandwidth inference mechanism, unlike TCP, is that it *does not require losses*. Indeed, the bandwidth inference mechanism is based on measuring the spacing of the PPs and not on measuring loss.

For the packet pair bandwidth inference mechanism to succeed, the Fair Scheduler must be a fine approximation of the fluid Generalized Processor Sharing (GPS). Bennet and Zhang show that the Packet Generalized Processor Sharing (PGPS) is not a fine enough approximation of the GPS system for the packet pair mechanism to succeed. However, they propose a new packet approximation algorithm called WF^{2}Q that perfectly suits the packet pair bandwidth inference mechanism (see [2] for a discussion on the impact of the packet approximation of GPS system and for the details of the WF^{2}Q algorithm). In the following, we assume an algorithm for the Fair Scheduler that is a fine approximation (in the sense of the packet pair mechanism) of the GPS system like the WF^{2}Q algorithm.

The great interest of a receiver-based version of packet pair is twofold. First, we have considerably less noise in the measurement (see [14]). In the sender-based version, the packet pair generates two acknowledgments at the receiver and it is the spacing of these Acks that is evaluated at the sender to derive the available bandwidth. However, if we have bottleneck on the back-channel, the Acks will be spaced by the back-channel bottleneck and not by the data channel bottleneck. Second, the receiver can detect congestion before the bottleneck queue starts to build and far before the bottleneck queue overflows. A signal of congestion is a packet pair estimate[1] of the available bandwidth lower than the current source throughput. In the simplest case where an estimate is given by a PP measurement, the first PP that leaves the queue after congestion occurs is a signal of this congestion. The delay between the congestion event at the bottleneck and the receiver action (to this congestion) is the delay for the PP to go from the bottleneck to the receiver (in the order of $\frac{RTT}{4}$ if we assume the bottleneck at the middle of the path from the source to the receiver). The PP bandwidth inference mechanism does not need losses to discover the available bandwidth and its receiver-driven version allows a receiver to react to congestion before the bottleneck queue overflows. We say that the receiver driven PP bandwidth inference mechanism *does not induce losses* when discovering the available bandwidth. An original consequence (unlike all the congestion control protocols that consider losses as signals of congestion) is that PLM can work without modification and no loss of performance on a lossy medium like a wireless link.

It is commonly argued that PP is very sensible to network conditions. We identify two major components that can adversely impact PP. First, the physical network characteristics (load balancing, MAC layer, etc.). Second, the traffic pattern (PP estimate in a self similar and multifractal environment).

The physical network characteristics can indeed adversely impact PP. However, they can aversely impact all the congestion control protocols. For instance, load balancing on a packet basis clearly renders the PP bandwidth inference mechanisms meaningless but the TCP bandwidth inference mechanisms as well. How can you estimate the RTT if one can not assume that all the packets take the same path (or at least if one can not identify which packet takes which path). Most of the physical network noise can be filtered with appropriate estimators (see [7]). We leave this question as a future research.

The traffic pattern *does not adversely impact* PP measurements. A PP leaving the bottleneck queue will be spaced by the available bandwidth for the relevant flow. As real traffic in the Internet is self similar and multifractal [5], the PP estimate of the available bandwidth will highly fluctuate. The oscillations can be misinterpreted as instability of the PP estimation method. In fact, as the background traffic is highly variable, it is natural that the available bandwidth at the bottleneck is highly variable. The oscillations in the available bandwidth estimation are not due to instability but to the high efficiency of the PP method. It is the task of the congestion control protocol to filter the PP estimates in order to react with a reasonable latency (i.e. the congestion control protocol must not overreact to PP estimates).

[1] The appropriate estimator must be defined in the congestion control protocol. We define the (simple) estimator for PLM in section 3.3.

3.3 PLM Protocol

We assume the source sends via cumulative layers and emits the packets as packet pairs on each of the layers, i.e. *all the packets on all the layers are sent in pairs* (we thus maximize the number of estimates). Moreover, we assume that the set of layers of a same session is considered as a *single* flow at the level of a Fair Scheduler. Now we describe the basic mechanisms of PLM that takes place at the receiver side. When a receiver just joined a session, it needs to know the bandwidth used by each layer. How to obtain this information is not the purpose of this paper. However, a simple way that avoids (source) implosion is to consider a multicast announcement session where all the sources send informations about their streams (for instance the name of the movie, the summary, etc.) and in particular the layer distribution used. A receiver who wants to join a session, first joins the session announcement and then joins the session. In the following, we assume that the receivers who want to join the session know the bandwidth distribution of the layers.

Let PP_t be the bandwidth inferred with the packet pair received a time t, and let B_n be the current bandwidth obtained with n cumulative layers: $B_n = \sum_{i=1}^{n} L_i$ where layer i carries data at a bandwidth L_i. Let $\hat{B}_e$ be the estimate of the available bandwidth.

At the beginning of the session, the receiver just joins the base layer and waits for its first packet pair. If after a predefined timeout the receiver does not receive any packet we infer that the receiver does not have enough bandwidth available to receive the base layer, and therefore can not join the session. At the reception of the first packet pair, at time t, the receiver sets the check-timer $T_c := t + C$, where C is the check value (we find in our simulations that a check value C of 1 second is a very good compromise between stability and fast convergence). We use the terminology check (for both T_c and C) because when T_c expires after a period of C seconds the receiver checks whether he must add or drop layers. When the receiver sees a packet pair a time t_i:

```
• if $PP_{t_i} < B_n$ then /*drop layers*/
    – $T_c := t_i + C$
    – until $B_n < PP_{t_i}$ do    (1)
        ∗ drop layer $n$
        ∗ $n$ := $n - 1$
• elseif $PP_{t_i} \geq B_n$ and $T_c < t_i$ /*have received PPs during at least $C$ units of time*/
  then /*add layers*/
    – $\hat{B}_e := \min_{T_c - C < t \leq t_i} PP_t$ /*take the minimum bandwidth estimate*/
    – $T_c := t_i + C$
    – if $\hat{B}_e \geq B_n$ then while $B_{n+1} < \hat{B}_e$ do    (2)
        ∗ add layer $n + 1$
        ∗ $n$ := $n + 1$
```

In summary, we drop a layer each time we have a sample PP_t value lower than the current layer subscription B_n, but we add layers according to the minimum PP_t value received during a period C (if all the samples are $PP_t \geq B_n$ during this period). The algorithm is conservative by using strict inequalities ($<$) in (1) and (2). We can consider a less conservative version of the same algorithm by using ($\leq$) in (1) and (2). For all the simulations we take the conservative version with strict inequalities.

In case of high congestion (with a FS network, high congestion can only happen when a large number of new flows appear in a period of less than C seconds) packet pairs could never reach the receiver (one pathological case is when the second packet in a PP is always lost). So we need a mechanism to reduce congestion enough to receive a packet pair. If after a predefined timeout period we do not receive any packet, we drop a layer; if the layer dropped is the lowest, we exit the session. We identify losses using the packets sequence number, and use a one bit field that marks the first packet of a PP burst (the other packets of the burst must have the following sequence numbers). So each packet contains a layer id, a sequence number, and a one bit field. If we receive some packets, however no PP, we evaluate the loss rate with a short term estimator and drop a layer if this estimator exceeds a predefined threshold (we fix empirically the threshold to 10% of loss). After dropping a layer we wait for a period called a blind period (we fix empirically the blind period to 500ms) before re-estimating the loss rate. This blind period helps to not over react to loss. We call that the aim of PLM is to allow a PP to reach the receiver in order to obtain an accurate estimate of the available bandwidth rather than dropping layer on loss.

Unlike RLM (see section 6.1) that produces losses with join experiments and unlike Vicisano's TCP-friendly protocol RLC (see section 6.2) that produces losses at join attempts (periodic bursts), PLM has the fundamental property that it does not induce any loss to discover the available bandwidth.

When a receiver joins a layer, all the other receivers downstream of the same bottleneck must join the same layer otherwise there is link underutilization (other benefits of the join synchronization are demonstrated in [18]). When a receiver drops a layer due to congestion, all the other receivers downstream of the same bottleneck must drop the layer otherwise the congestion persists. PLM achieves both, join and drop synchronization. Drop synchronization is obvious as the signal of drop (without high congestion) is a PP: Every receiver downstream of the same bottleneck will receive the same PP at the same time (roughly the same time according to the distance between the receivers and the bottleneck) and therefore will decide to drop layers at the same time. The join synchronization is less obvious at first sight. If the receivers do not start at the same time (the more probable situation) their timer T_c will not be synchronized and therefore they will join layers at different times. This is not a drawback as late joiners can not be synchronized before reaching the optimal rate. However, at the first drop inferred (due to $PP_t < B_n$) all the check-timers will be resynchronized and so all the following joins. This re-synchronization avoids the problems due to clock drift as well.

One problem for PLM can be the slow IGMP response in case of drop. IGMP can take several seconds before ”pruning” a group. This is a problem related to IGMP (and not to PLM). Rizzo in [17] introduces predictive techniques for fast IGMP leave. Moreover the leave synchronization of PLM tends to attenuate the drawback of the large leave delay in IGMP.

We have defined a multicast congestion control protocol that achieves and tracks the available bandwidth without loss induced to discover the available bandwidth. Moreover, a significant contribution of PLM is the introduction of a simple and efficient technique (based on PP) to infer which layer to join. As PLM is devised in the context of the FS-paradigm, all the properties of an ideal congestion control protocol – stability, efficiency, fairness, robustness, scalability, and feasibility – are theoretically guaranteed by the FS paradigm. We check via simulations if all these appealing properties really hold.

4 Initial Simulations

The initial set of simulations does not aim to evaluate PLM on realistic Internet scenarios but rather provides insights on how PLM behaves for specific and relevant configurations.

4.1 Evaluation Criteria

Three PLM parameters influence the behavior of PLM.

- The granularity of the layers: As PLM infers the bandwidth available and tracks the appropriate layers with a PP estimate, the less bandwidth per layer there is, the less stable the layer subscription will be. Moreover, an inaccurate estimate leads to a modification of the layer subscription, small layers emphasize this behavior.

- The check value C: A larger C leads to a higher stability as we can add layers only once each C interval, but leads to a lower speed of convergence.

- The burst size: In section 3.3 we only consider PP, packet bursts of *size two*. However, PLM is not limited to a burst of size two. Increasing the burst size increases the accuracy of the estimate, but increases the probability of loss due to the bursty nature of each layers.

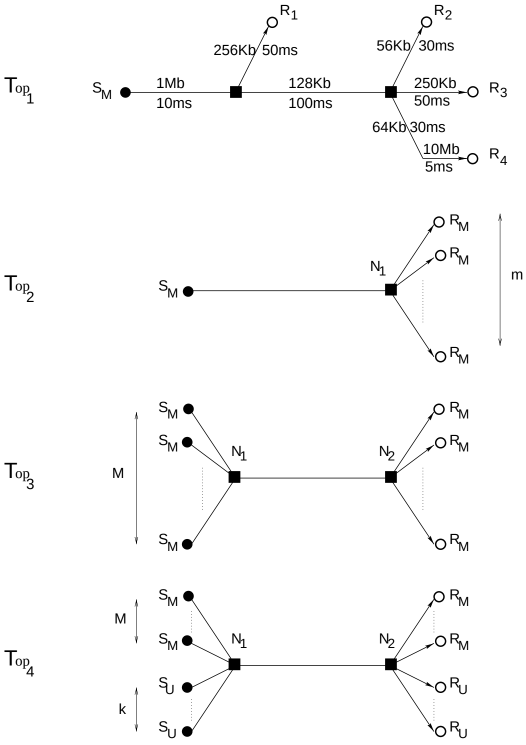


Figure 2: Simulation Topologies.

The behavior of PLM can be influenced by other flows (PLM sessions, TCP flows, etc.). To evaluate the behavior of PLM under these various internal and external parameters we consider three evaluation criteria. The first criterion is the bandwidth seen by each receiver. We want the mean bandwidth achieved by each receiver to be close to the available bandwidth. Moreover, we want the receivers to converge fast to the optimal rate and to be stable at this optimal rate. Our second criterion is therefore the number of layers subscribed. However, a bandwidth close to the optimal rate and a stable layer subscription are not sufficient. Therefore, our third evaluation criterion is the loss rate.

4.2 Initial Simulation Topologies

Fig. 2 shows the four topologies simulated to evaluate the PLM behavior. The first topology Top_1 consists of one PLM source and four PLM receivers. We evaluate the speed and the accuracy of the convergence in the context of a large heterogeneity of link bandwidths and link delays. The second topology Top_2 consists of one PLM source and m PLM receivers. We evaluate the scalability of PLM with respect to session size. The third topology Top_3 consists in M PLM sources and one PLM receiver per source. We evaluate the scalability of PLM with respect to the number of sessions and the inter-PLM session fairness. The last topology Top_4 consists of M PLM sessions (with one receiver), and k unicast sessions. We evaluate the scalability of PLM with an increasing number of unicast sessions, and the fairness of PLM with respect to the unicast sessions.

We have implemented the PLM protocol in the ns [12] simulator. We use the following default parameters for our simulations. If we do not explicitly specify a parameter, the default parameters are used. The routing protocol is DVMRP (in particular graft and prune messages are simulated). All the queues are Fair Queuing (FQ) queues and each flow (PLM session or unicast flow) has a queue size of 20 packets. We chose the packet size of all the flows (PLM and TCP) to be 500 bytes. Each layer of a PLM source is simulated with a CBR source sending packets using PP. To avoid synchronization of the CBR sources we add randomness in the generation time of the PP. The check value is $C = 1$ second. For the experiment that simulates TCP flows, we use the ns implementation of TCP Reno. To exclude the influence of the max-window, we choose a max-window of 4000 packets.

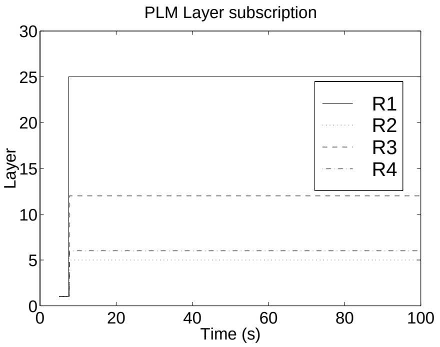


Figure 3: Speed, accuracy, and stability of PLM convergence for a single session, Top_1.

4.3 Initial PLM Simulations Results

4.3.1 Basic Scenarios

We start our simulations evaluating the speed and the accuracy of the PLM convergence on topology Top_1. We choose 10 Kbit/s per layer. This very fine grain choice of bandwidth increment is a tough test for PLM as pointed out in section 4.1. Fig.3 shows the results of this simulation. All receivers join the session at $t = 5$s. Before $t = 7$s (a check value plus the time to graft the appropriate groups) all the receivers converge to the optimal layer (i.e. the highest number of layers possible to use the available bandwidth). Moreover, the receivers stay at the optimal layer during the whole simulation and without loss induced (there is no loss during the whole simulation). In conclusion of this first experiment, PLM converges to the optimal link utilization in the order of a check value C and stays at this rate we no loss induced.

Our second simulation on topology Top_2 considers the scaling of PLM with respect to the number of receivers. We chose 50 Kbit/s bandwidth per layer. For this simulation we consider the link (S_M, N_1) with a bandwidth of 280Kb/s and a delay of 20ms. The links (N_1, R_M) have a bandwidth uniformly chosen in $[500, 1000]$ Kb/s and a delay uniformly chosen in $[5, 150]$ ms. The first set of experiments considers m receivers, $m = 1, ..., 100$ all started at $t = 5$s. For this set of experiments, all the receivers converge to the optimal layer at the same time and stay at this layer without loss induced, whatever the number of receivers is. We do not give the plot of these experiments as Fig.4 shows the same behavior. In Fig.4 we show the results of an experiment where we start 20 PLM 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 PLM receivers. The aim of this last 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. When a receiver joins the session he starts at layer one and after a check value C he joins the optimal layer five (the vertical lines in Fig.4). We see in Fig.4 that neither the number of receivers nor the late joins have an influence on the convergence time and on the stability. This is not trivial for a receiver-driven protocol. Indeed, RLM for instance uses shared learning that leads to join synchronization (see section 6.1). Therefore, for RLM late joins have an influence on the convergence time of the other receivers. Once again, for all the experiments on topology Top_2 we do not observe any packet loss, and once the optimal layer is reached, the receivers stay at this layer for the whole simulation. We can easily explain these results as the layers subscribed only depend on the PP received; the receivers joining the session do not have an impact on the PP received by the others receivers. Multiple receivers joining the session at the same time, late joins, multiple late joins do not have any influence on the PLM session. In conclusion PLM perfectly scales with the number of receivers.

As we see in the previous experiments the number of receivers does not have any influence on the dynamics of a PLM session. Therefore, for all the other experiments we always consider PLM sessions with only one receiver.

Up to know we see a perfectly stable behavior of PLM, however to really evaluate the stability of PLM we must consider dynamic scenarios. The simulations on topology Top_4 consider dynamic scenarios. 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 consider a layer granularity of 20 Kbit/s.

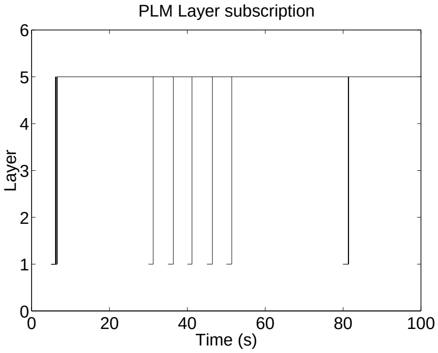
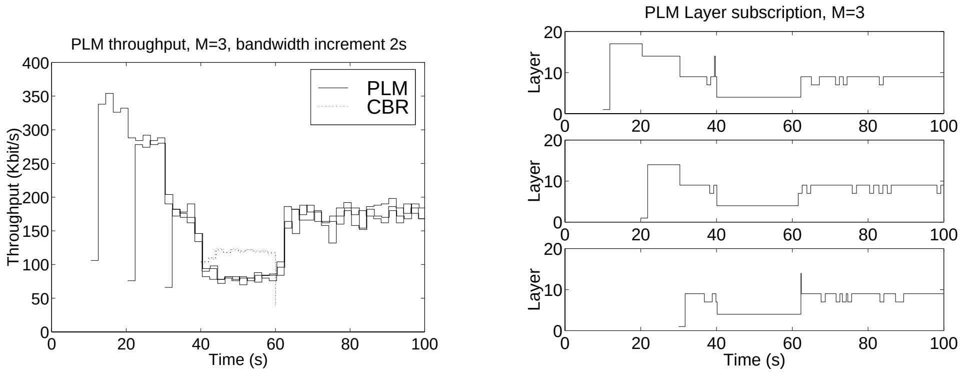


Figure 4: Scaling of a PLM session with respect to the number of receivers, Top_2.

The first experiment on topology Top_4 considers a mix of PLM and CBR flows. We consider $M = 3$ PLM sessions and $k = 3$ CBR flows for the experiment plotted in Fig. 5. 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 PLM receivers respectively at time $t = 10, 20, 30$s. We start the three CBR sources at time $t = 40$s and stop the CBR sources at $t = 60$s. The aim of this scenario is to study in the first part (before starting the CBR sources) the behavior of PLM with an increasing number of PLM sessions, and in the second part (after starting the CBR sources) the behavior of PLM in case of severe congestion. When the CBR sources stop we observe the speed of PLM to grab the available bandwidth. We choose as many CBR sources as PLM sessions to simulate severe congestion. Indeed, with FQ, the only way to create congestion is to significantly increase the number of sessions. We have a bottleneck link at 600 Kbit/s, but the first PLM receiver can only get 340 Kbit/s (see Fig. 5(a)). This is only due to the limited number of layers, in our simulation we have 17 layers, so $17 * 20 = 340$ Kbit/s if the receivers subscribe all the layers. When a new PLM session starts, the PLM sessions share equally the bottleneck at roughly (due to the layer granularity) $600/M$ Kbit/s where M is the number of PLM sessions. When the CBR sources start, the PLM sessions share the bottleneck at roughly $600/(M + k)$ Kbit/s where k is the number of CBR flows. We note that the CBR flows have a slightly higher share than the PLM flows. This is simply due to the layer granularity. The theoretical share is $600/6 = 100$ Kbit/s, however as we have chosen the conservative algorithm for PLM (see section 3.3), PLM infers an available bandwidth of 80 Kbit/s and therefore joins 4 layers. All the three PLM sessions have the same behavior. The CBR flows grab has much available bandwidth they can, in this case 120 Kbit/s each. When the CBR sources stop, the PLM sessions increase their subscriptions to layers until reaching the highest link utilization (according to the layer granularity). PLM is very reactive and takes all the bandwidth available. In Fig. 5(b) we see that changes in layer subscription to adapt exactly to the available bandwidth are within the time of one check value C (here $C = 1$ second). Moreover during the whole simulation (and even during the period of severe congestion) the PLM sessions experience no loss. PLM does not need large buffer sizes to absorb transient period of congestion (from the time the congestion starts to the time the reaction of PLM has an effect on the buffer). Indeed, the congestion is detected by a PP already when the congestion starts (we do not need a buffer to fill or to overflow to detect congestion) and the period of transient congestion (until the prune sent by the PLM receiver reaches the bottleneck) lasts less than a RTT. We repeated this experiment with different number of PLM sessions ($M = k = 1, 5, 10$) and different burst sizes (from 2 to 4 packets in a burst) and did not see any significant modification in the behavior of PLM. We observed some losses for the PLM sessions for a burst of 4 packets. However, these losses are infrequent and appear principally when a new session starts.

The second experiment on topology Top_4 considers a mix of a PLM sessions and TCP flows. We consider $M = 1$ PLM session and $k = 2$ TCP flows for the experiment plotted in Fig. 6. The bandwidth of link (N_1, N_2) is $100 * (M + k) = 300$ Kbit/s and the delay is 20 ms. We start the first TCP flow at $t = 0$s, then we start the PLM session at $t = 20$s and finally start the second TCP flow at $t = 60$s. The aim of this experiment is to study the behavior of PLM with TCP flows in a simple experiment. We see that a single PLM session perfectly adapts to the available bandwidth in presence of TCP flows. Here again this adaptation is in the order of one check value and induces no loss for the PLM sessions.



(a) Bandwidth of each PLM and CBR receiver.

(b) Layer subscription of the PLM receivers.

Figure 5: PLM and CBR flows sharing the same bottleneck, Top_4.

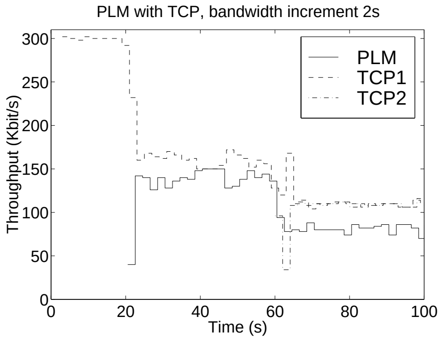


Figure 6: PLM and TCP flows sharing the same bottleneck, Top_4.

We can draw a partial conclusion at this point of our study: PLM reaches the optimal layer in the order of one check value C and tracks the optimal layer in a dynamic environment without loss induced for the PLM sessions.

4.3.2 Multiple PLM Sessions

To further explore the properties of PLM we do experiments to study the scalability of PLM for an increasing number of PLM sessions. This experiment is on topology Top_3. The links (S_M, N_1), and (N_2, R_M) have a bandwidth of 10 Mbit/s and a delay of 5 ms. The link (N_1, N_2) has a bandwidth of $200 * M$ Kbit/s where M is the number of PLM sessions. All the sessions start randomly in $[5, 10]$ seconds. We do a measurement from $t = 20$s to $t = 100$s. For this experiment we vary the value of C (1 and 5 seconds), the value of the PP $Burst$ size (2, 3, and 4 packets), and the bandwidth distribution of the layers (20 Kbit/s and 50 Kbit/s). For each simulation we vary the number of PLM sessions. We only show the individual plots for $C = 1$, $Burst = 2$, and a layer granularity 50 Kbit/s, as this experiment is pathological and shows interesting behaviors (see Fig. 7 and Fig. 8). In Fig. 7 a 'x' shows the mean bandwidth for a simple receiver. The solid line shows the mean bandwidth for all the receivers. We see that the mean bandwidth increases from 150 Kbit/s for a small number of sessions, to 195 Kbit/s for a large number of sessions. The layer granularity is 50 Kbit/s and the available bandwidth for each session is 200 Kbit/s. Due to the conservative algorithm (see section 3.3) the PLM session infers a bandwidth at 150 Kbit/s and so joins 3 layers. However, the number of sessions increases, the number of sessions that do not join the fourth layer increases, and so the number of sessions that can get this available bandwidth. Therefore, even with a large layer granularity the PLM sessions reach the available bandwidth in the case of a high degree of multiplexing. We obtained the same results for the mean bandwidth

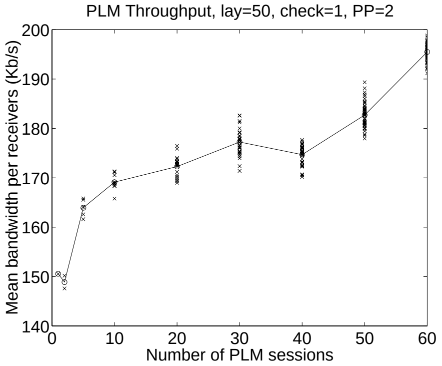


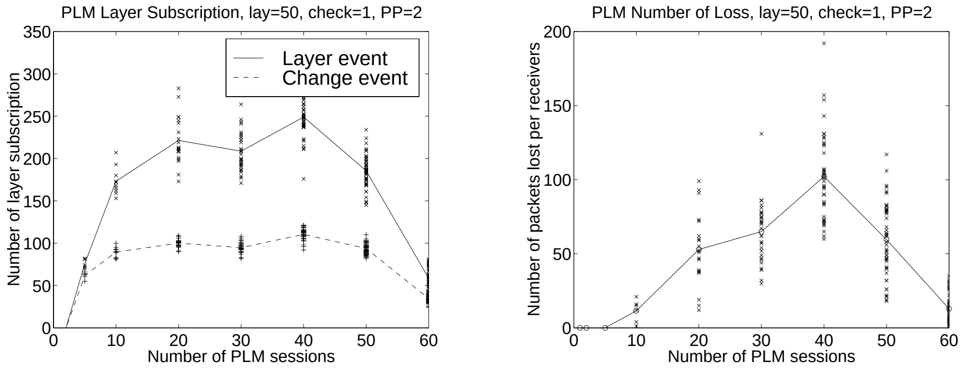
Figure 7: PLM throughput, C=1, layer granularity 50 Kbit/s, Burst of 2 packets, Top_3.

with the other parameters studied. We note that PLM achieves good inter-PLM fairness as the mean bandwidth for the individual receivers is close ($\pm 5\%$)to the mean bandwidth for all the receivers (small standard deviation).

Concerning the number of layers subscribed, we distinguish in Fig. 8(a) between the total number of individual layers a receiver joins during the whole simulation experiment (*layer event* in Fig. 8(a)) and the total number of events to change layers (*change event* in Fig. 8(a)). For instance, if a receiver is at layer 2 and at time t he infers enough available bandwidth to join layer 5, we say that at t there are 3 layer events but 1 change event.

This scenario, with PLM sessions only, is pathological and is not realistic in the Internet where we have a mix of fine grain adjustment (1 packet per RTT like TCP) protocols and large grain adjustment (one layer like PLM) protocols. In Fig. 8(a) we see the total number of layer subscriptions during the simulation. Several layers are added or dropped in a single event, roughly two layers per event because the number of layer events is roughly two times the number of change events. In fact, if the routing protocol allows cumulative graft, the change event curve has to be considered (as all the layers change in a single event can be send in one cumulative graft or prune), otherwise the layer event curve is considered (this curve shows the number of graft or prune sent in case of non cumulative graft or prune). We note the striking similarity between Fig. 8(a) and Fig. 8(b). The shape of both plots is due to the same phenomenon. The available bandwidth for each session is the bottleneck bandwidth divided by the number session that share the same bottleneck. A PP can only infer the right available bandwidth if, at the moment the PP reaches the bottleneck, all the sessions are back-logged in the queue. If a session use less than its fair share, another session can get the bandwidth. However, even if some sessions use more than their fair share (because other sessions use less), a PP infers the bandwidth available only dependent on the number of sessions back-logged but independent on the bandwidth really used by these sessions. Therefore, several sessions can infer in parallel a higher available bandwidth than their fair share. That leads to oscillation (see Fig. 8(a)) and losses (see Fig. 8(b)). For a large number of sessions there is such a high multiplexing that the sessions use, in the mean, a bandwidth close to the available bandwidth. There is less free bandwidth to grab, the oscillations and the losses significantly diminished (see Fig. 8).

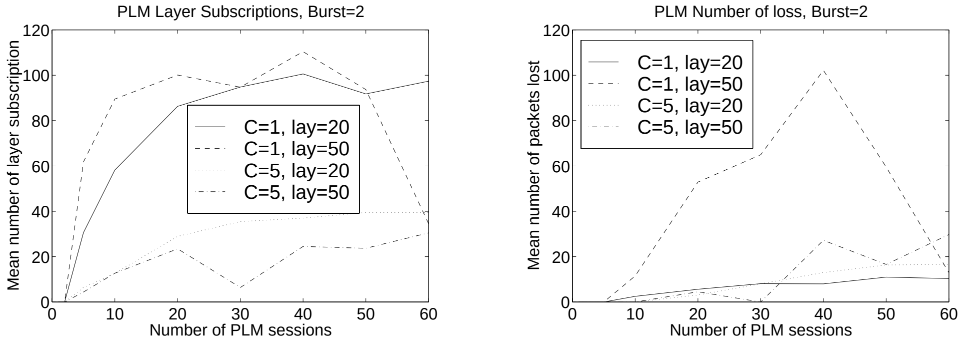
This simulation scenario studied represents a worst case result for the layer subscription oscillation and the loss rate for all our simulations. This is pathological since we only have PLM sessions with a large layer granularity compared to the available bandwidth for each flow. That results in a wrong estimation of the available bandwidth causing oscillation and, due to the large layer granularity, losses. Even in this pathological case PLM performs reasonably well, as the oscillations do not lead to a decrease in the mean bandwidth. In fact, a session can never infer an available bandwidth lower than the real bandwidth available when all the session are back-logged. Moreover the losses induced by these oscillations are reasonably low. The mean loss is, even in the worst case, less than 2% of packets lost. However this pathological case can be avoided by adjusting correctly the PLM parameters (check value, bandwidth granularity), and is improbable in a real network. In Fig. 9 we have the layer subscription (change event) and the number of losses for various scenarios. We see that increasing the check value C significantly reduces the number of losses (see Fig. 9(b)) and the number of oscillations (see Fig. 9(a)). In a real network we would have a mix of adaptation granularity: Large granularity in the case of PLM and fine granularity in the case of TCP. Such a multiplexing of PLM and TCP flows will avoid the presence of free available bandwidth, and so reduce the oscillations and the number of losses. This assertion



(a) PLM layer subscription.

(b) PLM losses.

Figure 8: PLM layer subscription and losses, C=1, layer granularity 50 Kbit/s, Burst of 2 packets, Top_3.



(a) PLM layer subscriptions.

(b) PLM losses.

Figure 9: PLM layer subscription and losses, Burst of 2 packets, Top_3.

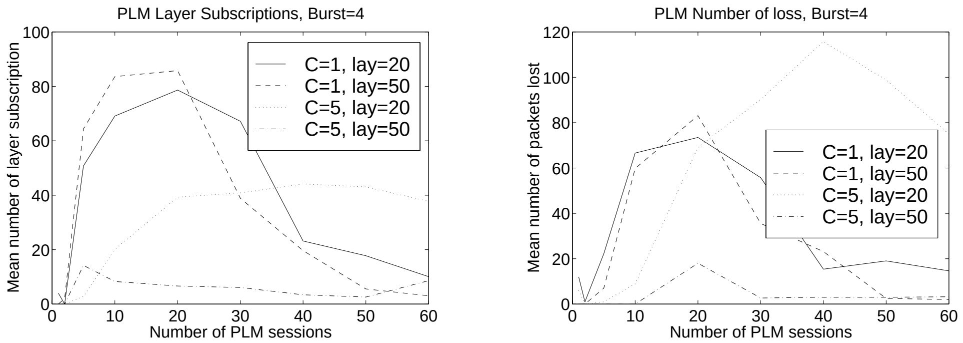
is evaluated in the following with a mix of PLM and TCP flows.

Finally, increasing the size of the burst does not significantly change the mean bandwidth (we do not give the plot). Fig. 10(a) shows the layer subscriptions for a burst of 4 packets. We see a slightly lower number of oscillations compared to the case of a burst of size of 2 packets (see Fig. 9(a)), and we see that the curve has its maximum for a lower number of sessions. However, the large burst size significantly increases the number of losses (see Fig. 10(b)).

In conclusion, PLM scales well with the number of sessions, achieves good inter-PLM fairness, reaches the optimal rate in the order of one check value C, and leads to a low number of losses (even in the pathological case). The oscillations of PLM should not be considered as instability, they rather reflect the fact that PLM does not want to leave bandwidth unused (when there is bandwidth available PLM sessions try to get it). We saw that oscillations can be significantly reduced by increasing the check value. However, we will see that these oscillations disappear when there is a mix of TCP and PLM flows.

4.3.3 Multiple PLM Sessions and TCP Flows

Our next experiment on topology Top_4 considers a mix of PLM sessions and TCP flows. We consider M PLM and $k = M$ TCP sessions. The bandwidth of link (N_1, N_2) is $100 * (M + k)$ Kbit/s and the delay is 20 ms. The links (S_M, N_1), and (N_2, R_M) have a bandwidth of 10 Mbit/s and a delay of 5 ms. We start the TCP flows randomly in $[5, 10]$s and the PLM sessions randomly in $[20, 25]$s. We do the measurement between $t = 30$s and $t = 100$s. The



(a) PLM layer subscription.

(b) PLM losses.

Figure 10: PLM layer subscription and losses, Burst of 4 packets, Top_3.

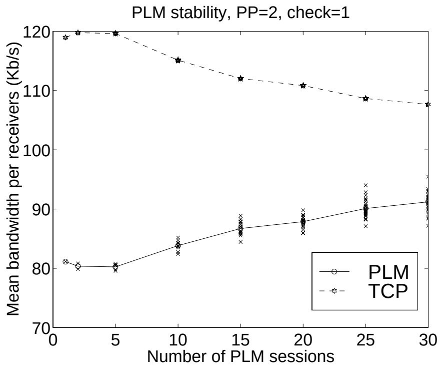


Figure 11: Throughput for a mix of PLM and TCP flows, C=1, burst of 2 packets, 20 Kbit/s layer granularity, Top_4.

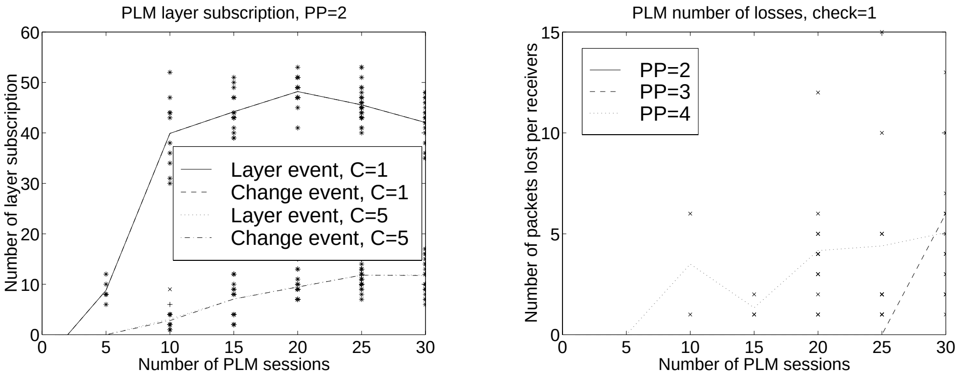
aim of this experiment is to study the scaling properties of PLM with concurrent TCP flows. We repeat this simulation for various check values ($C = 1, 5$), burst sizes (2, 3, and 4 packets), and bandwidth granularities (20 Kbit/s and 50 Kbit/s). We plot the results for $C = 1$, a burst of 2 packets, and a layer granularity of 20 Kbit/s, as this experiment is very representative.

Fig. 11 shows the throughput for each receiver for the PLM and the TCP flows. Due to the layer granularity, for a small number of sessions, PLM sessions can only get 80 Kbit/s (compared to the 100 Kbit/s available for each flows). When the number of sessions increases, the PLM sessions get more bandwidth due to multiplexing. The TCP flows have exactly the opposite behavior of the PLM sessions. PLM achieves a good inter-PLM fairness, indeed the mean bandwidth for the individual receivers remains close to the mean bandwidth for all the receivers.

Fig. 12(a) shows the layer subscription. The change event curve and the layer event curve are identical. That means there is never a change of more than 1 layer at the same time. Moreover, the number of oscillations is low with $C = 1$ and can be significantly reduced with $C = 5$. We have few oscillations, and these oscillations lead to *no loss* for a burst of 2 packets (see Fig. 12(b)) and a very low number of losses for a burst of 4 packets. We see that with concurrent TCP flows, PLM behaves remarkably well. We have an optimal link utilization, inter-PLM fairness, a very low number of oscillations, and no loss induced.

4.3.4 Variable Packet Size

For all the previous simulations we always considered packets of the same size. We know that FQ guarantees an equal share of the bottleneck to each flow. Therefore, if flows have different packet sizes, the PP may lead to wrong estimate of the available bandwidth.



(a) PLM layer subscription, C=1,5, Burst of 2 packets.

(b) PLM losses, C=1, Burst of 2, 3, and 4 packets.

Figure 12: Layer subscription and losses for the PLM sessions for a mix of PLM and TCP flows, 20 Kbit/s layer granularity, Top_4.

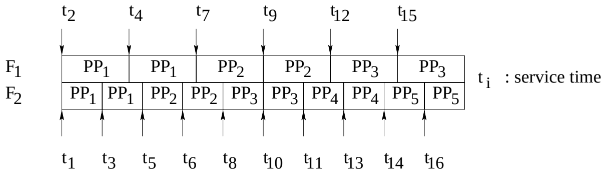
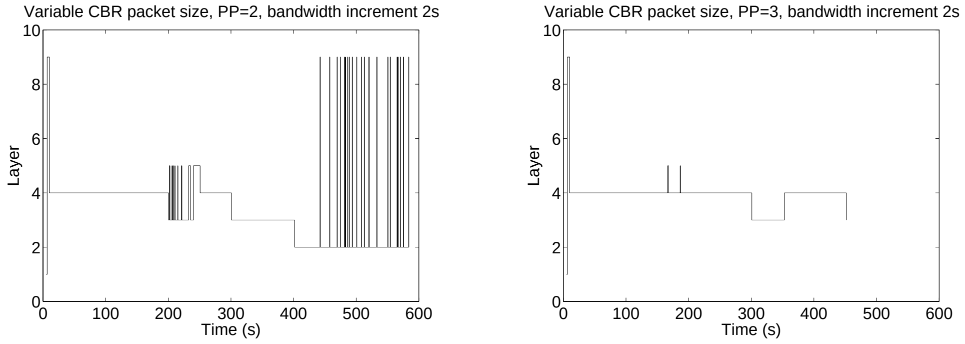


Figure 13: Service time of packets of variable size in a single FQ queue.

Imagine a bottleneck queue with a capacity of B and two concurrent flows F_1 and F_2 with a packet size of 500 and 300 Bytes respectively. Fig. 13 shows the packet service time of the buffered packets of each flow. The available bandwidth inferred by PP_1 of F_2 is $\frac{300}{t_3-t_1} = \frac{300}{\frac{300}{B}+\frac{500}{B}} = \frac{300}{800}B < \frac{B}{2}$, the available bandwidth inferred by PP_2 of F_2 is $\frac{300}{t_6-t_5} = \frac{300}{\frac{300}{B}} = B > \frac{B}{2}$, the available bandwidth inferred by PP_3 of F_2 is $\frac{300}{t_{10}-t_8} = \frac{300}{\frac{300}{B}+\frac{500}{B}} = \frac{300}{800}B < \frac{B}{2}$, etc. The available bandwidth inferred by PP_1 of F_1 is $\frac{300}{t_4-t_2} = \frac{300}{\frac{300}{B}+\frac{500}{B}} = \frac{300}{800}B < \frac{B}{2}$, the available bandwidth inferred by PP_2 of F_1 is $\frac{300}{t_9-t_7} = \frac{300}{\frac{300}{B}+\frac{500}{B}} = \frac{300}{800}B < \frac{B}{2}$, the available bandwidth inferred by PP_3 of F_1 is $\frac{300}{t_{15}-t_{12}} = \frac{300}{\frac{300}{B}+\frac{300}{B}+\frac{500}{B}} = \frac{300}{1100}B < \frac{B}{2}$. In this simple example, when the packet size of the flows is different, the PP bandwidth inference is not accurate. We see that the most significant problem comes from small packets compared to the size of the packets of all the other flows crossing the same FQ queue. Indeed, this is the only case where a PP can be served in a single round and so infers an available bandwidth equal to the bottleneck bandwidth (see PP_2 for flow F_2). If we are not in the case of small packets, the PP can never overestimate the available bandwidth.

However, there are ways to avoid this problem. First, it is often possible to take large packets for multimedia streams. If, however, the packets must be small, we can increase the size of the burst and so avoid the overestimating effect due to the small packets. Second, multiplexing of flows significantly reduces the problem due to the packet size. Indeed, with large number of flows crossing a single FS router, the number of flows becomes more important than the packet size of each flow. These two remarks are confirmed in the following simulations.

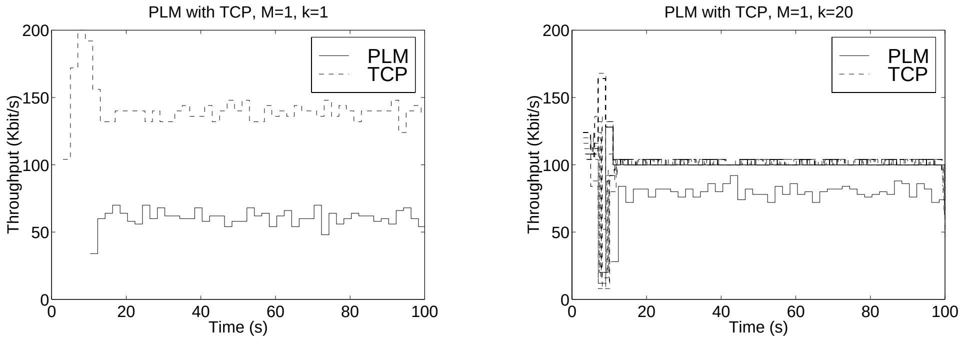
In Fig. 14 we study the impact of the packet size on the PP bandwidth inference. We use topology Top_4, the links (S_M, N_1), (S_U, N_1), (N_2, R_M), and (N_2, R_U) have a bandwidth of 10 Mbit/s and a delay of 5 ms. The link (N_1, N_2) has a bandwidth of 400 Kbit/s and a delay of 20ms. We consider 20 Kbit/s layers. We have $M = 1$ PLM session, and $k = 1$ CBR flow. We start the PLM session at $t = 5$s. We vary the packet size of the CBR flow during the time while fixing the same size for the PLM packet to 500 Bytes. We start at $t = 10$s a CBR flow with packets of 50 Bytes. At $t = 50$s and every 50 seconds the packet size of the CBR flow changes to the following values: 100, 200, 300, 400,



(a) Burst=2.

(b) Burst=3.

Figure 14: Mix of PLM and CBR flows. Influence of the Burst size on the bandwidth inference for variable packet size, Top_4.



(a) One PLM session and one TCP flow.

(b) One PLM session and twenty TCP flows.

Figure 15: Mix of PLM and TCP flows. Influence of the multiplexing on bandwidth inference. PLM packet size: 500 Bytes, CBR packet size 1000 Bytes, Top_4.

500, 750, 1000, 1250, 1500, 1750, and 2000 bytes. We see that packet sizes lower than the PLM packets can lead to a wrong estimate, however the worst estimates are for packet size larger than the PLM packet size (see Fig. 14(a)). Increasing the burst size to three significantly reduce the error in the estimate (see Fig. 14(b)). The reason is that the number of bytes in a burst is larger than the largest packet size of concurrent CBR flow. Therefore, a FQ queue can never serve a whole burst in a single round.

Fig. 15 presents the effect of the flow multiplexing for the PP inference when each flow has a different packet size. We use topology Top_4, 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 consider 20 Kbit/s layers. We have $M = 1$ PLM session, and $k = 1, 20$ TCP flows. The link (N_1, N_2) has a bandwidth of $100 * (k + M)$ Kbit/s and a delay of 20ms. We start the PLM session at $t = 5$s and randomly start the TCP flow(s) in $[0, 5]$s. The PLM packet size is 500 Bytes, the TCP packet size is 1000 Bytes. Fig. 15(a) shows the plot for $k = 1$ TCP flow. The PLM flow gets only 60 Kbit/s instead of 80 Kbit/s due to the wrong estimates. However when we increase the number of TCP flows, the wrong estimates disappear and PLM gets its available bandwidth at 80 Kbit/s.

In conclusion, whereas different packet size can lead to a wrong estimate, choosing larger packet sizes for the PLM flows, the natural multiplexing of the flows, and increasing the burst size significantly reduce the problems related to

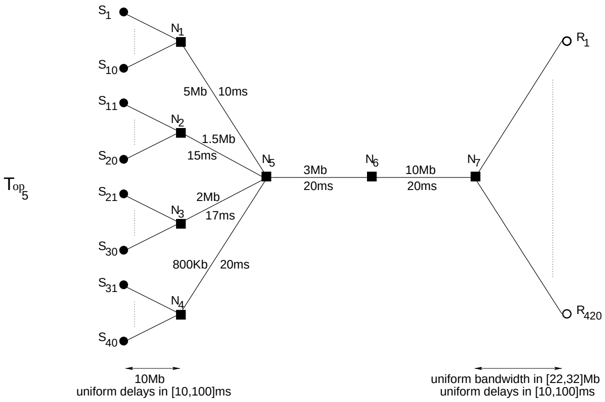


Figure 16: Simulation topology Top_5 for the realistic background traffic.

the correct bandwidth estimation in case of packets of different size.

5 Simulations with a Realistic Background Traffic

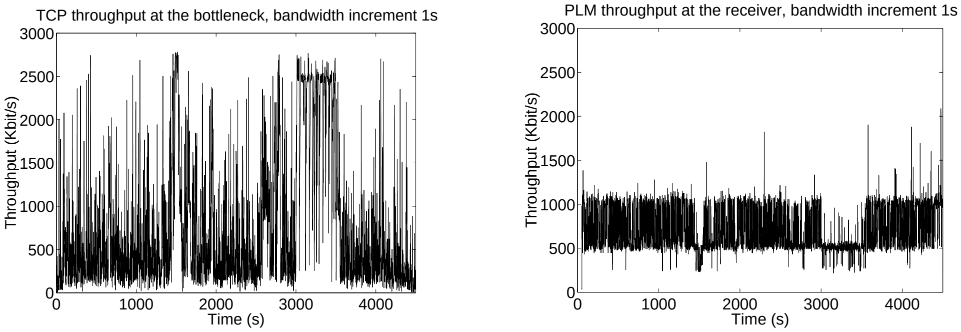
Recent works show the evidence of self similar [10] and even multifractal [5] traffic in the Internet. The burstiness over many time scales of such a traffic can adversely impact congestion control protocols. Moreover, it is commonly argued that the PP bandwidth inference mechanism is highly sensible to the traffic pattern. These arguments motivate this set of simulation that aims to study the performances of PLM with self similar and multifractal background traffic.

5.1 Simulation Scenario

To simulate self similar and multifractal traffic we reuse a large part of the ns scripts and of the scenarios used in [5]. In the following, we explain the scenario we considered in this paper. Fig. 16 shows the topology considered for our simulations. Let $\{S_i, i = 1, \ldots, 40\}$ be the set of web servers and $\{R_i, i = 1, \ldots, 420\}$ be the set of web clients. Client and server communicate with a ns version of HTTP 1.0 and of TCP Reno. Let N_S be the number of sessions, each session contains 300 pages, each page contains one object. Each session is defined for a client randomly chosen among the set of clients. For a given session, the client requests each page on a server randomly chosen among the set of servers. All the objects of a same page are requested on the same server. The inter-session starting time is exponentially distributed with a mean of 1s, the inter-page starting time is exponentially distributed with a mean of 15s. The inter-page starting time is the same that the inter-object starting time as there is only one object per page. The object size is pareto distributed (pareto II) with a mean of 12 packets/object and a shape of 1.2. The TCP packet size is 1000 Bytes. We run all our simulations for 4500 simulated seconds.

Feldmann et al. shown that this scenario produces a traffic close to a real Internet traffic. In particular, the traffic is self similar at large and medium time scales and multifractal at small time scales (see [5] for details).

Topology Top_5 has 4 bottlenecks, the links (N_2, N_5), (N_3, N_5), (N_4, N_5), and (N_5, N_6). All the queues are fair queuing with a shared buffer of 100000 Bytes. A flow for the FQ scheduler is defined as one (source,receiver) pair for the TCP flows, a PLM session is considered as one flow. We place a PLM source at node S_{10} and a PLM receiver at node R_{100}. The bottleneck link for the PLM flow is the link (N_5, N_6). If we do not specify, the bottleneck link always refers to the PLM bottleneck link. For all the simulations we start the PLM session at $t = 50$s. Remember that PLM scales perfectly with the number of receivers. Increasing the number of receivers in the simulations does not change our results. In the following we take only one PLM receiver for the PLM session.



(a) Mean throughput averaged on 1s intervals for the background traffic at the bottleneck (N_5, N_6).

(b) Mean throughput averaged on 1s intervals for the PLM flow at the PLM receiver.

Figure 17: $N_S = 100, C = 1$, 1000 bytes PLM packet size, exponential layers.

5.2 PLM Simulations Results with Realistic Background Traffic

We call the self similar and multifractal web traffic the background traffic. We consider two kinds of background traffic: a lightly loaded with $N_S = 100$ sessions; a heavy loaded with $N_S = 400$ sessions. The lightly loaded scenario is characterized by a mean throughput for the background traffic of about 600 Kbit/s and a loss rate around 0.4%. The heavy loaded scenario is characterized by a mean throughput for the background traffic of about 2200 Kbit/s and a loss rate around 1.5%. All the other parameters varied during the simulations are related to PLM. We consider various check values ($C \in \{0.2, 0.5, 1, 5\}$), two layer distributions (100 Kbit/s layers, exponentially distributed layers: 32 Kbit/s, 64 Kbit/s, 128 Kbit/s, etc.), and two PLM packet sizes (500 Bytes and 1000 Bytes).

In order to get a benchmark for the efficiency of PLM, we replace for the two kinds of background traffic the PLM session with a TCP flow where the TCP sender always has data to send (the TCP source is at node S_{10} and the TCP receiver at node R_{100}). We call this TCP flow the benchmark TCP flow, or TCP$_b$. For the lightly loaded scenario, the mean throughput for the background traffic is 569 Kbit/s. The mean throughput achieved by the benchmark TCP connection is 1453 Kbit/s with a loss rate of 0.185%. For the heavy loaded scenario, the mean throughput for the background traffic is 2318 Kbit/s. The mean throughput achieved by the benchmark TCP connection is 479 Kbit/s with a loss rate of 0.809%. Remember that the bottleneck bandwidth for TCP$_b$ (and PLM) is 3 Mbit/s.

Fig. 17 shows the results of a simulation with a lightly loaded background traffic ($N_S = 100$). The PLM check value is $C = 1$, the PLM packet size is 1000 Bytes, and the layers are exponentially distributed. We first note that the PLM session closely follows the evolutions of the throughput of the background traffic (for instance look at $t = 1500$s). Moreover, PLM does not experience any loss during the whole simulation. PLM is able to track the available bandwidth without loss induced even in complex environments. The mean throughput for the whole simulation for the background traffic is 737 Kbit/s, and is 733 Kbit/s for the PLM session.

Fig. 18 shows the layer subscription of the PLM receiver during the simulation. Most of the layer subscription oscillations are between layer 5 (512 Kbit/s) and layer 6 (1024 Kbit/s). There are 2090 layer changes during the simulation (roughly 1 layer change every 2 seconds). The layer subscription oscillation is not a weakness of PLM, but a consequence of the high efficiency of PLM. As the background traffic fluctuates in a large range of time scales, PLM must join and drop layers in a large range of time scales in order to get the available bandwidth. PLM is not unstable, it is the background traffic that is “unstable” and PLM simply follows its variations. We see in all the simulations a tradeoff between the layer subscription oscillation and the efficiency of PLM. Increasing the layer granularity (from 100 Kbit/s to exponentially distributed layers) or increasing the check value reduces the layer subscription oscillation but reduces the efficiency of PLM as well (because the more the layer subscription oscillates, the more efficient PLM is).

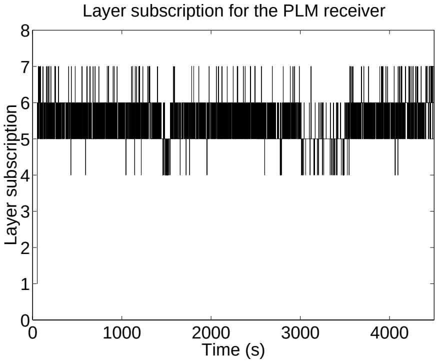


Figure 18: Layer subscription for the PLM receiver.

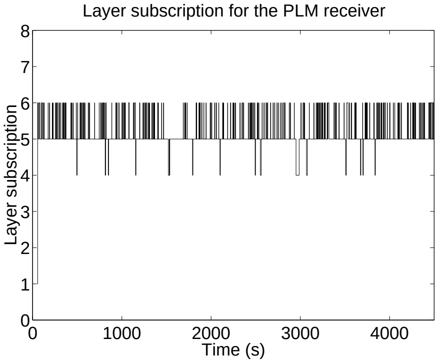


Figure 19: $N_S = 100$, $C = 5$, 1000 bytes PLM packet size, exponential layers. Layer subscription of the PLM receiver.

For a check value $C = 0.2$s and 100 Kbit/s layers, we obtain a surprising result. The mean throughput for the PLM session is 1507 Kbit/s, higher than the TCP benchmark. It is the only one experiment for the lightly loaded background traffic where observe some losses for the PLM flow. In fact, we get 22 losses corresponding to a loss rate lower than 3.10^{-5}. The PLM receiver generates 14010 change events and 124328 layer changes. We explain the big difference between layer change and the change event because the large variations in the available bandwidth enforce the PLM receiver to join and drop multiple layers in a single event. As explained in section 4.3.2 with cumulative graft and prune the value of interest is the number of change events.

This result is fundamental for the evaluation of the efficiency of PLM. For the case of a single PLM multicast session from sender S to receivers $R_1, \ldots, R_n$ we observed that PLM could achieve for each receiver R_i a throughput B_i^{PLM} that is higher than the throughput B_i^{TCP} obtained by a single unicast TCP connection between S and that receiver R_i. This was not case for any previously proposed multicast congestion control protocol MCC where there was always, for at least one receiver R_i, $B_i^{MCC} < B_i^{TCP}$. However, the high number of layer changes is not possible with the actual multicast routing protocols. This experiment simply aims to show the high efficiency of PLM even in complex environments and the high efficiency of the packet pair bandwidth inference mechanism as we use it in PLM.

Stability in the layer subscription does not mean low efficiency. For a check value $C = 5$s and exponentially distributed layers, the mean throughput for the PLM receiver is 561 Kbit/s with only 417 layer changes (roughly 1 layer change every 10 seconds) and no loss. Fig. 19 shows the evolution of the layer subscription for this scenario. We see that layer subscription oscillate around layer 5 (512 Kbit/s).

For file transfer applications, layer subscription oscillation is not a drawback. These applications want to achieve the highest throughput. The limit in the number of layer subscription oscillations is only imposed by the routing protocol. For multimedia applications, frequent variations in the throughput (and thus in the quality perceived) can be a serious drawback. However, we believe it is the task of the application to filter the oscillations (for instance see

[16]). If the application is not smart enough to filter the oscillations, we shown that an appropriate choice in the check value parameter can significantly reduce layer subscription oscillations with a reasonable decrease in the efficiency. In [9] we studied a bandwidth allocation policy that significantly rewards multicast sessions with respect to the number of receivers. Thus, a lower efficiency than TCP does not necessarily mean a lower throughput for a multicast session.

The set of experiments for the heavy loaded scenario ($N_S = 400$) confirms our conclusions and does not show other significant new results. The set of experiments with a PLM packet size of 500 Bytes simply shows a slight decrease in the mean throughput for PLM and a slight increase in the layer subscription oscillations. This is due to the error in the estimate with packets of different sizes (see section 4.3.4). However, the multiplexing of the flows is sufficient to avoid large estimation errors. We do not notice any instability or significant inefficiency of PLM in these simulations.

In conclusion, we tested PLM in a large variety of situations and with different values of parameters and always found that PLM is efficient, stable, and induces no loss (or a negligible loss rate in some extreme situations) even with a realistic background traffic.

6 Comparison with other Cumulative Layered Protocols

In this section we simulate RLM (see [11]) and RLC (see [22]) for the basic scenarios of section 4.3.1 to compare their performances to PLM. If not specified, the simulation configuration is the same as in section 4.3.1.

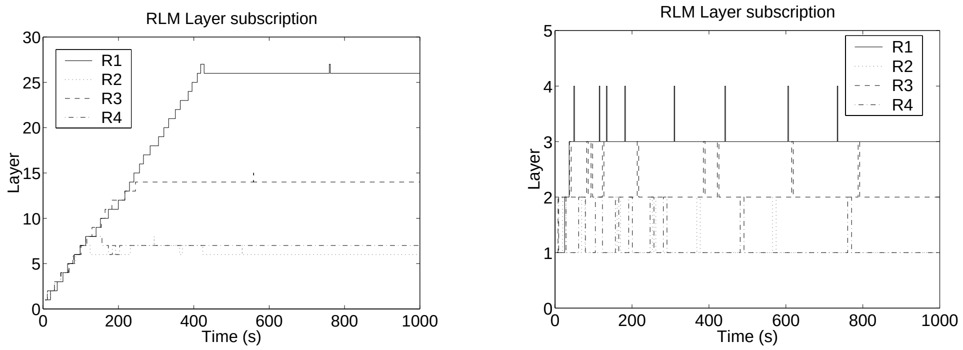
6.1 RLM simulations

We use the ns implementation of RLM with the parameters choice of McCanne in [11]. RLM was designed with FIFO scheduling. However, we only present results with FQ scheduling. Indeed, all the simulations done with FQ scheduling outperform the simulations done with FIFO. Therefore, in order to compare honestly RLM and PLM we compare both with FQ scheduling. We run all the simulations of this section for 1000 seconds.

The first simulation evaluates the speed and the accuracy of RLM convergence for Top_1. We see in Fig. 20(a) the very slow convergence time of RLM compared to PLM (compare Fig. 20(a) and Fig. 3) for exactly the same scenario. Receiver $R1$ needs more than 400 seconds to converge to the optimal rate. Moreover, the mean loss rate for this simulation is 3.2%. As pointed out in section 4.3.1 the 10 Kbit/s layers is a tough test for RLM. and shows weaknesses 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 lower the convergence time. The significant loss rate is explained by the loss threshold of RLM that is set to 25%. With such small layers we never enter in a congestion period where we lose more than 25% of the packets. For each receiver the last layer causes a low loss rate that result in the 3.2% mean loss rate. This is the reason why we have a very low number of join experiments. We make 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. 20(b). In this case RLM performs significantly better than in the previous case. Even if the convergence time remains larger than with PLM, it is reasonably fast. We clearly see the join experiments that are, in this case, the main reason for a mean loss rate of 1.6%.

The second experiment on topology Top_2 evaluates the scaling of a single RLM session with respect to the number of receivers. The most interesting event on Fig. 21 is the join synchronization. Due to the shared learning, receivers can not join upper layers while there are some receivers subscribed only to lower layers. Late joins can slow down the convergence time for RLM (unlike PLM). We did the same experiment with exponential layers and observed no noticeable changes.

The third experiment on Top_4 considers a mix of RLM and CBR flows. Fig. 22(a) shows the layer subscription of the three RLM sessions. The first session starts at $t = 50$s, the second at $t = 100$s, and the third at $t = 150$s. The three CBR sources start at $t = 300$s. We see, due to the small layer size of 10 Kbit/s, slow convergence. When the CBR sources start and create 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. 22(b)). The mean loss rate is 7.5% in this experiment. We note the same effect as in experiment one: The small layers result in losses



(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 20: Speed, accuracy, and stability of RLM convergence for a single session, Top_1.

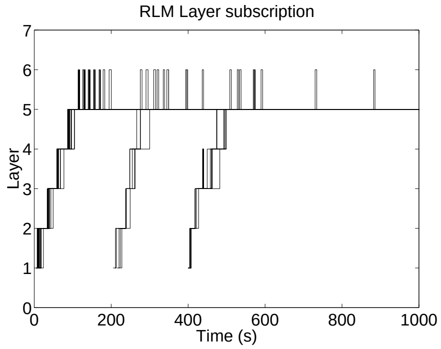


Figure 21: Scaling of a RLM session with respect to the number of receivers, Top_2.

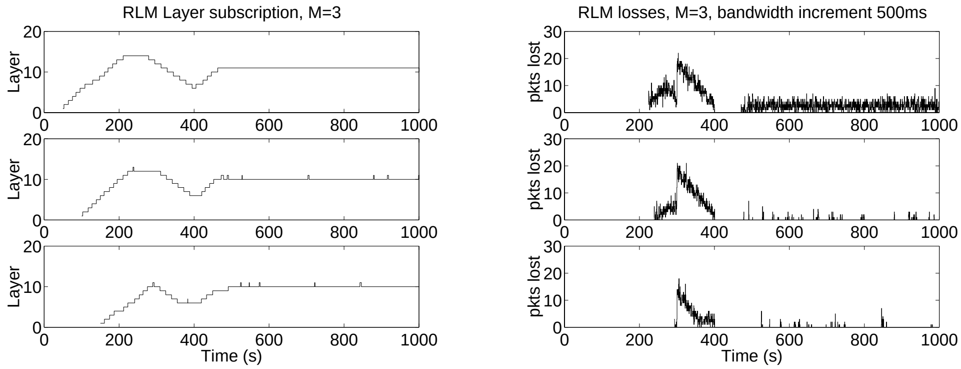
that never exceed the loss threshold (see upper plot in Fig. 22(b)), never result in a layer drop, and result in no join experiment (see upper plot in Fig. 22(a)). We do the same simulation with exponential layers. As expected the large layer granularity results in higher reactivity for RLM. When the CBR sources start, 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 drops to 1.2%.

The fourth experiment on Top_4 considers a mix of one RLM session and TCP flows. We observe roughly the same behavior as in the third experiment. RLM converges slowly to the optimal rate (see Fig. 23(a)). When TCP2 starts, a transitory period of congestion appears until RLM drops the right number of layers to reach its fair share of the bottleneck bandwidth (see Fig. 23(b)).

In conclusion of these experiments on the basic scenarios, RLM achieves reasonable performances only with large layer sizes. Nevertheless, PLM outperforms in all the cases RLM.

6.2 RLC Simulations

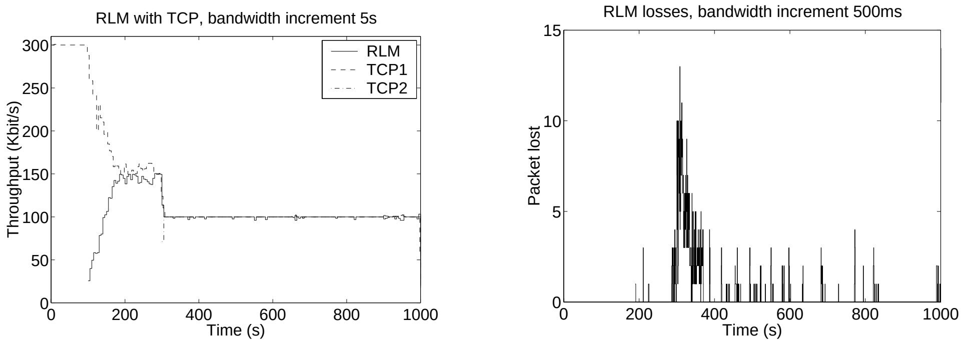
In this section, for space limitations, we do not give the plots for the simulations, just general results. RLC can be considered as a TCP friendly version of RLM with the improvement of the synchronization points (data packets with a special flag). Receivers can only increase his number of layers at synchronization points (SP) if no losses are experienced during the burst preceding that SP (see [22] for more details). We expected RLC to perform better than RLM, but our experiments show significant weaknesses of RLC. RLC bandwidth inference is based on the generation



(a) Layer subscription of each RLM sessions.

(b) Loss rate of each RLM sessions.

Figure 22: RLM and CBR flows sharing the same bottleneck, Top_4.



(a) Throughput of the RLM session and of the TCP flows.

(b) Loss rate of the RLM session.

Figure 23: RLM and TCP flows sharing the same bottleneck, Top_4.

of a periodic burst that aims to reduce to transitory period of congestion due to join experiments (see [22] for more details). To succeed, the burst must make the queue overflow. However, this is the case in our simulations *only* with a very judicious choice of the queue size, which is impossible to do in a real network. As the bandwidth inference does not succeed, the receivers periodically join layers when there is not enough bandwidth available to add these layers. That leads to periodic congestion and periodic losses. In our experiment we have seen losses from 2% up to 15% according to the choice of the buffer size.

As the deaf period (which aims to avoid cascade drop) is constant, unlike RLM, we have cases (for instance large buffer size and low bandwidth link) where the initial value of 1.1 second is not enough and leads to cascade drop.

In summary, RLC addresses some issues of RLM but raises new issues related to the inefficient bandwidth inference method.

7 Validation of the FS-paradigm

We have devised PLM using the FS paradigm. So theoretically PLM must have the properties of an ideal congestion control protocol. In this section we check which properties of an ideal congestion control protocol PLM meets.

Stability In some pathological cases (only PLM sessions) PLM oscillates, however these oscillations are around the optimal rate and are fairly distributed among the receivers. Even in complex environments, PLM reaches and tracks the available bandwidth without loss induced.

Efficiency PLM is clearly efficient in the sense of satisfying the multimedia and file transfer applications: PLM allows for each receiver to get the maximum bandwidth available on the path between the source and this receiver.

Fairness PLM achieves inter-PLM fairness and TCP-fairness[2].

Robustness PLM robustness is guaranteed by the FS network.

Scalability PLM is highly scalable due to the receiver-driven cumulative layered scheme. PLM does not require any signaling nor feedback. The burden of PLM is distributed among the receivers.

Feasibility PLM requires a FS network, this issue is discussed in [8] and it is shown to be feasible per ISP (Internet Service Provider). PLM simply requires an IP-multicast network.

In conclusion, PLM meets all the properties of an ideal congestion control protocol. However the fundamental point (and the success) of the FS-paradigm is that we do not add any specific mechanism *in PLM* to meet these properties. For instance, to be satisfied, the user of a multimedia application requires: (i) to receive the highest quality (high throughput, low number of losses) and (ii) to avoid frequent modification in the quality perceived. Of course these two requirements are related with the properties of an ideal congestion control protocol, but we do not specifically address the properties of an ideal congestion control protocol in the design of PLM. The benefits of the FS-paradigm become obvious when we look at the number and the influence of the PLM parameters, which are the check value C, the burst size, and the layer granularity. We have only three parameters whose exact value is not very important for the correct behavior of PLM. The fine tuning of the check value, for instance, does not fundamentally change the properties of PLM but defines a slightly different operating point that can improve the satisfaction of a user. In the contrary to PLM, RLM has numerous parameters, where each parameter was introduced to improve the stability and the efficiency of RLM. A wrong choice of any such parameter can significantly decrease the performance of RLM. Its TCP-friendly version needs fine tuning of the parameters to effectively achieve TCP-friendliness.

PLM is significantly simpler than the previous cumulative layered receiver-driven protocols and yet outperforms these protocols. We therefore consider PLM as a practical proof of the validity of the FS-paradigm.

8 Related Work

Steven McCanne [11] first presented a cumulative layered scheme combined with a receiver-driven driven protocol to join and leave layers. His protocol, RLM, is the basis of most of the studies on cumulative layered protocols. These studies explored the properties of RLM ([6], [1]), or tried to improve the performance of RLM ([22], [23], [20]). Bajaj et al. explore the relative merits of uniform versus priority dropping for the transmission of layered video. As pointed out by the authors their results are “ambiguous” and can not be easily summarized. We refer the interested reader to [1]. Gopalakrishnan et al. study the behavior of RLM in various scenarios and find that RLM shows high instability for VBR traffic, has very poor fairness properties in most of the cases, and achieves a low link utilization with VBR traffic. Vicisano et al. devise a TCP-friendly receiver-driven protocol and propose how to achieve synchronization of the layer subscription. A TCP-friendly version of RLM was introduced in [20] where the layer subscription depends on a TCP-friendly formula. Since RLM leads to large oscillations under network congestion, Wu et al. propose to use thin streams [23].

The use of Packet Pair for bandwidth inference was introduced by Keshav [7] and used by Paxson (he introduces a more refined technique, PBM see[14]) and Ratnasamy [15] in the context of a FIFO network to discover the *bottleneck* bandwidth. The bottleneck bandwidth gives at most an upper bound for the *available* bandwidth. While the bottleneck bandwidth is interesting for many problems, it does not have a significant interest for congestion control.

[2] We talk about TCP-fairness that is different from TCP friendliness. TCP-fairness means that PLM does not significantly affect the performance (throughput, delay, etc.) of TCP flows when sharing the same bottleneck.

9 Conclusion

We started our study with an introduction to the FS-paradigm. This paradigm allows to theoretically devise nearly ideal congestion control protocols. Then, we devised the PLM protocol according to the FS-paradigm. PLM is based on a cumulative layered multicast scheme and on the use of Packet Pair to discover the available bandwidth without doing any join experiment unlike RLM. We investigate the properties of PLM under various conditions and see that PLM converges in the order of one check value to the optimal link utilization and stays at this optimal layer subscription during the whole simulation. Then we investigated the scalability of PLM with respect to the number of receivers and to the number of sessions. PLM behavior is fully independent of the number of receivers. However, concurrent PLM sessions can slightly decrease the performance of PLM. But adjusting correctly the PLM parameters significantly reduces the number of losses and the layer subscription oscillations. In a mix of TCP flows and PLM sessions, PLM performs well. PLM has a very low number of layer subscription oscillations and no loss is induced. We investigated the effect of variable packet size and saw that the multiplexing of the sessions significantly reduces the problems of convergence due to the packet size. Finally, we explore the impact of realistic background traffic on PLM. We obtain a confirmation of our results. PLM converges fast to the available bandwidth and track this available bandwidth without loss induced.

The introduction of our available bandwidth estimator based on packet pair solves the problems (efficiency, stability, loss induced, simplicity, fairness, etc.) of the previous layered multicast congestion control protocols. This is therefore a significant contribution to the field of multicast congestion control that makes PLM the new benchmark for the layered multicast protocols.

Another implication of the good performance of PLM is the practical validation of the FS-paradigm. The simulation results show that the simple mechanisms introduced in PLM lead to a very efficient congestion control protocol, as predicted by the FS-paradigm.

In conclusion, we have devised a layered multicast congestion control protocol that outperforms all the previous congestion control protocols like RLM and its TCP-friendly variants. Indeed PLM converges fast to the available bandwidth, induces no loss to discover and track the available bandwidth, requires no signaling, and scales with both the number of receivers and the number of sessions. All these properties hold in complex environments such as self similar and multifractal traffic.

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] J. Bennett and H. Zhang. Wf2q: Worst-case fair weighted fair queueing. In *Proceedings of IEEE INFOCOM'96*, San Francisco, CA, USA, March 1996.

[3] J. C. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. *IEEE/ACM Transactions on Networking*, 5(5):675–689, October 1997.

[4] A. Demers, S. Keshav, and S. Shenkar. Analysis and simulation of a fair queueing algorithm. In *Proc. SIGCOMM'89*, pages 1–12, Austin, Texas, September 1989.

[5] A. Feldmann, A. C. Gilbert, P. Huang, and W. Willinger. Dynamics of ip traffic: A study of the role of variability and the impact of control. In *ACM SIGCOMM'99*, volume 29, Massachusetts, USA, September 1999.

[6] R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson, and C. J. Sreenan. Stability and fairness issues in layered multicast. In *NOSSDAV'99*, 1999.

[7] S. Keshav. *Congestion Control in Computer Networks*. PhD thesis, EECS, University of Berkeley, CA 94720, USA, September 1991.

[8] A. Legout and E. W. Biersack. Beyond tcp-friendliness: A new paradigm for end-to-end congestion control. Technical report, Institut Eurecom, November 1999. http://www.eurecom.fr/ legout/Research/research.html.

[9] A. Legout, J. Nonnenmacher, and E. W. Biersack. Bandwidth allocation policies for unicast and multicast flows. In *Proceedings of IEEE INFOCOM'99*, New York, USA, March 1999.

[10] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the Self-Similar Nature of Ethernet Traffic. In *Proceedings of ACM SIGComm*. ACM, Sept. 1993.

[11] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *SIGCOMM 96*, pages 117–130, Aug. 1996.

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

[13] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks. In *Proc. IEEE INFOCOM'93*, pages 521–530, 1993.

[14] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California, Berkeley, April 1997.

[15] S. Ratnasamy and S. McCanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In *INFOCOM'99*, New York, USA, March 1999.

[16] R. Rejaie, M. Handley, and D. Estrin. Quality adaptation for congestion controlled video playback over the internet. In *ACM SIGCOMM'99*, Cambridge, MA, September 1999.

[17] L. Rizzo. Fast group management in igmp. In *Hipparc'98*, 1998.

[18] D. Rubenstein, J. Kurose, and D. Towsley. The impact of multicast layering on network fairness. In *SIGCOMM'99*, September 1999.

[19] D. Stiliadis and A. Varma. A general methodology for designing efficient traffic scheduling and shaping algorithms. In *Proceedings of IEEE INFOCOM '97*, April 1997.

[20] 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.

[21] L. Vicisano. Notes on a cumulative layered organization of data packets accross multiple streams with different rates. Technical report, UCL London, Jan. 1997.

[22] 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.

[23] L. Wu, R. Sharma, and B. Smith. Thin streams: An architecture for multicasting layered video. In *NOSSDAV'97*, pages 173–182, St Louis, Missouri, USA, May 1997.