HAL
archives-ouvertes.fr

TUF: a Tag-based UDP Multicast Flow Control Protocol

Antoine Clerget

▶ To cite this version:

Antoine Clerget. TUF: a Tag-based UDP Multicast Flow Control Protocol. RR-3728, INRIA. 1999. <inria-00072936>

HAL Id: inria-00072936
https://hal.inria.fr/inria-00072936

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

TUF: A Tag-based UDP Multicast Flow Control Protocol

Antoine Clerget

N° 3728

July 1999

THÈME 1

ISSN 0249-6399

$\mathcal{T}$UF: A Tag-based UDP Multicast Flow Control Protocol

Antoine Clerget

Thème 1 — Réseaux et systèmes
Projet Rodeo

Rapport de recherche n° 3728 — July 1999 — 24 pages

Abstract: Considerable efforts have focused on congestion control protocols for large-scale multicast applications, but in most cases, these consider only single rate sessions. In heterogeneous environments, it is interesting to offer each receiver its maximum fair rate, regardless of the presence of slower receivers in the group. In this paper, we propose an alternative to layered encoding/transmission schemes. $\mathcal{T}$UF, our Tag-based UDP multicast Flow control protocol, uses very simple router filters, that, as RED or diffserv, are not aware of the transport layer, do not add flow state, nor intensive computation. Filtering is done based on packet "tags", computed at the source, and provides fair bandwidth sharing. We believe that this scheme can coexist with TCP, and can be useful for reliable multicast applications.

Key-words: Multicast, congestion control, heterogeneity, max-min fairness, router filters, multi-rate sessions, Internet, UDP, Tags

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Utilisation de Labels pour le Contrôle de Congestion Multipoint Multidébit

Résumé : Les travaux sur le contrôle de congestion pour les environnements multipoints étudient principalement le cas des sessions mono-débit. Or, dans un contexte fortement hétérogène, on souhaiterait pouvoir proposer à chaque récepteur un débit adapté à sa capacité, indépendamment de la présence d'autres membres dans le groupe. Nous proposons ici une alternative aux mécanismes de transmission en couches. $\mathcal{T}$UF, un protocole de contrôle de congestion basé sur l'utilisation de labels, nécessite l'ajout de filtres simples dans les routeurs. Comme pour RED ou diffserv, ces filtres n'interviennent pas au niveau transport, n'ajoutent pas d'état par flot dans les routeurs, et n'effectuent que des opérations très simples. Le filtrage, basé sur la valeur d'une étiquette contenue dans le paquet et calculé par la source, permet de partager équitablement la bande passante. Nous pensons que ce protocole peut cohabiter avec TCP, et être utilisé pour le multipoint fiable.

Mots-clés : Multipoint, codage hierarchique, contrôle de congestion, hétérogénéité, équité, filtres, protocoles, Internet, multidébit, UDP

1 Introduction

With the advent of new multicast services, such as audio-conferencing or software distribution, controlling the greed of multicast applications has become critical. The multicast environment raises new issues in the development of congestion control protocols. In unicast end to end congestion control protocols, a source adapts its rate in response to congestion signals sent back by the receiver. In large multicast environments, sources may be overwhelmed by feedback from all the members of the group. Heterogeneity among the receivers is also a major issue and will tend to appear more frequently in the future, with the extent of the Internet and the arrival of new types of links and devices. In a context of high bandwidth, delay, and jitter variability, it is difficult to send to every receiver its *fair share*, and to avoid being limited by the worst receiver in the group.

Defining this "*fair share*" in a heterogeneous multicast environment is not straightforward. To adopt a common viewpoint on fairness, we must first define a set of users and resources, as well as a satisfaction metric. In the networking context, this metric can for example be response time, power[1], throughput, throughtput per downstream receiver [1]. We can then either define a "fairness index" [2], which quantitatively measures the "equality" of resource allocation, or define qualitative properties such as max-min fairness [3] or proportional fairness [4, 5]. We say that a resource allocation is **max-min fair** if it is feasible and if it is not possible to increase the satisfaction of a user without simultaneously causing a decrease in the satisfaction of a more constrained user.

Discussions on fairness for unicast compare flows throughput over a path or over a common bottleneck link. As outlined in [6], it is difficult to have the same consensus on fairness for multicast. A multicast session can have independent bottleneck links that it shares with different unicast or multicast flows. Defining a fair allocation policy, thus the "satisfaction" of a multicast flow, is not intuitive. Does a multicast session deserve more bandwidth than a single TCP connection if it serves multiple receivers? Most articles on multicast fairness consider the case of single-rate sessions sharing bandwidth with unicast sessions, and focus on the *inter-flow fairness*, based on the data path to the worst receiver in the group [6, 7, 8]. They do not consider the *inter-receiver fairness*, among receivers of a same session.

It seems however more natural to see fairness from a receiver standpoint. The receiver's satisfaction can simply be defined as the bandwidth received. It does not depend on whether it receives information from a unicast or a multicast flow. In [9], the notion of global max-min fairness is defined from this perspective : *users* are receivers and not flows. This definition applies to unicast sessions as well as multicast sessions, and even to multi-rate sessions[2]. Weaker fairness properties from a flow standpoint can then be inferred, such as **same-path-receiver fairness** [9], that states that the rates of two flows over a same data path are identical.

[1] Ratio of throughput and response time

[2] Receivers of a multi-rate multicast sessions receive data at different rates

Our aim here is a protocol that achieves a bandwidth allocation policy appropriate for large heterogeneous environments, in which a receiver is not penalized by the presence of other slow or distant members in the group. In other words, we would like each receiver to achieve the rate it would get if unconstrained by other receivers in the group, called the isolated rate in [10, 11]. With max-min fairness, a user's bandwidth allocation is at least as large as that of all users whose data-path contain its bottleneck. Receivers are then constrained only by their bottleneck link and the presence of other flows.

A bandwidth allocation policy for multi-rate multicast sessions must rely on mechanisms in the network. Layered transmission schemes use multicast routing functionalities to provide different rates to members of the group. Section 2 explains the drawback of this approach. However, adding complex flow management in the routers seems unrealistic in the Internet today. Section 3 presents $\mathcal{T}$UF, a Tag-based UDP Flow control protocol based on very simple packet filters in the routers that are not aware of the transport layers, do not add flow state, nor intensive computation. Section 4 comments on the use of $\mathcal{T}$UF with reliable multicast. Section 5 shows that it exhibits interesting fairness properties, and that it can coexist with TCP in a friendly manner. Section 6 proposes simulation results and Section 7 concludes the paper.

2 Related Work

Most of the congestion control proposals for multicast consider the case of single-rate sessions, mostly because their target application is reliable multicast. They try to adapt the source's rate in a scalable and TCP-friendly manner to the slowest receiver. The TCP friendliness is achieved using either window-based algorithms [6, 7, 12], rate-based algorithms based on TCP models [8, 13, 14, 15, 16], or additive increase, multiplicative decrease schemes [17, 18, 19]. Section 4 shows however that reliable multicast applications can benefit from multi-rate data distribution.

The main proposal for multi-rate multicast sessions today consists in exploiting routing functionalities in the network through layering. In a layered encoding/transmission scheme, the source hierarchically encodes the data in a number of layers that incrementally combined, provide an increasing quality to the user. Each layer is then sent on a separate multicast group. Receivers take the initiative of joining and leaving these groups to maximize their satisfaction, while limiting network congestion. In Receiver-driven Layered Multicast (RLM) [20], "probing" experiments are used to decide when to join additional layers, and layers are dropped when congestion is detected. Coordination between receivers is necessary to avoid misinterpretation of congestion signals due to other receivers' experiments. Concurrent RLM sessions can also be the source of misinterpretation. [21] proposes the use of sender initiated probes to synchronize these experiments. In [22], receivers compute their subscription level from loss rate and round trip time estimations using a model of TCP's throughput.

In layered transmission schemes, the bandwidth occupied by the session at a bottleneck link is determined by the most demanding downstream receiver. Reacting to congestion therefore requires cooperation. This cooperation is difficult to realize, especially in very

heterogeneous environments, where some receivers are far downstream the bottleneck, others just behind it. In case of congestion, downstream receivers must jointly leave a layer. The reaction time to congestion is therefore determined by the time it takes for the most distant receiver to detect the congestion and initiate a prune message propagation up the distribution tree. The aggressiveness of the flow must be adjusted consequently, to the detriment of closer receivers in the distribution tree.

Moreover, increasing the number of layers used for the rate control has a cost :

- Multicast addresses are scarce resources
- Applications have to encode these layers
- State is added in the routers for each multicast group
- Frequent join and leave messages generate an IGMP traffic overhead

An application will therefore use a small number of layers, and the granularity in available rates may be very important. This has an impact on fairness and efficiency.

Finally, basing congestion control functions on particularities of multicast routing protocols should be avoided. These layered schemes implicitly suppose that :

- The different layers are routed along the same multicast tree. Future possible multicast routing schemes that try for example to do load sharing, or frequent re-routing, will cause these layered schemes to fail.
- Reaction time to congestion and join and leave latencies should be of the same order of magnitude. The fact that join and leave latencies are critical to the performance of RLM is revealing. This may be problematic for example, when access control is performed before the right is granted to join a group. This high join latency will impact the performance of congestion control.

In [23], Brian Levine identifies the fact that a source can send packets only to the entire group as a weakness of today's IP-multicast architecture. The Addressable Internet Multicast architecture (AIM) proposed adds the ability to deliver packets to subsets of a multicast group on a per-packet basis.

What we propose here is to deliver a packet to a subset of the multicast group based on the packet's *tag*. The tag identifies the "priority level" of the packet, that will be filtered and delivered only to the members who should receive the corresponding data rate.

Documents recently sent to the RM[3] mailing list [24, 25] also advocate for the use of filters in the routers for multicast congestion control. We believe that the protocol $\mathcal{T}$UF, by its simplicity and scalability, is a nice way of implementing such filters.

[3]The Reliable Multicast Research Group (IRTF). http://www.east.isi.edu/RMRG

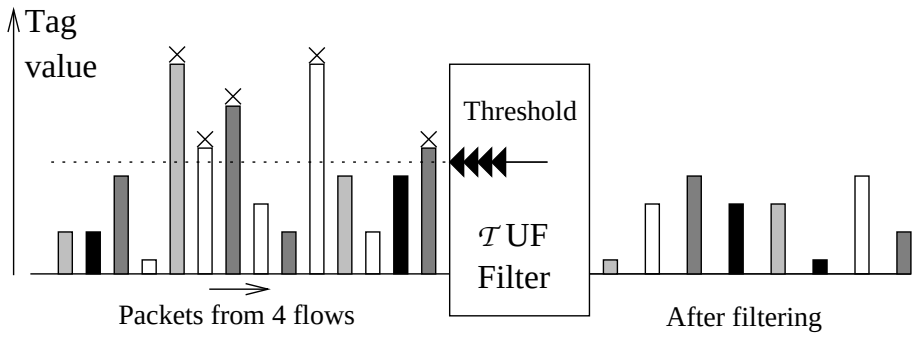

Figure 1: A $\mathcal{T}$UF filter

3 $\mathcal{T}$UF: Tag-based UDP Flow control

To discard packets from the flows whose bandwidth exceed their fair share, filters have to be installed in the routers. There are basically two approaches to discard this surplus of packets : either the router randomly discards packets, such as RED, or it uses information in the packet. With the second approach, we keep a control over the packets that are forwarded, and it enables us to define an algorithm which does not add flow state in the routers. Figure 1 describes the behavior of a $\mathcal{T}$UF filter.

Suppose that a router filters out packets whose tag value is above a threshold K. Since we want max-min fairness and therefore at least same-path-receiver fairness, it must forward packets to an outgoing interface at an equal rate for each flow. This constraints the rate $r(K)$ at which a source can send packets tagged with a value below K. r is an increasing function, shared by all the sources.

For its simplicity, we chose the linear function $r(K) = B_0.K$. As long as sources follow rule 1, they can tag packets in the way they find the most appropriate.

Rule 1 *$\forall K$, a source should not send packets tagged with values below K at a rate higher than $B_0.K$*

B_0 defines a rate "quantum" in packets per second, i.e. the lowest achievable rate for the congestion control protocol. The highest achievable rate is $B_0.K_{max}$, where K_{max} is the highest possible tag value. Typically, $K_{max} = 2^{32} - 1$.

The $\mathcal{T}$UF protocol now lies on two separate functionalities:

- Packet tagging at the source
- Determining the threshold.

3.1 Packet Tagging

A source tries to tag packets with the lowest possible values, under the constraint defined by rule 1.

A naive way of tagging packets is to use the sequence number as a tag, that sequence number begin reset to 0 every $T_0 = 1/B_0$. Such a strategy follows rule 1, but will result in tremendous burstiness since the K first packet of a T_0 period will be forwarded, followed by a blank period, where all packets are discarded. It is therefore necessary to do **tag interleaving**.

A source sending packets at time $T_0.x/2^n$, with $x = 0, 1, \ldots, 2^n - 1$, can tag these packets with $I(x/2^n)$, where I is defined as follows ([26]) :

$$\begin{array}{rccc} I: & [0,1[& \longrightarrow & \mathbb{N} \cup \infty \\ & \sum_{i=1}^{\infty} b_i.2^{-i} & \mapsto & \sum_{i=1}^{\infty} b_i.2^{i-1} \end{array}$$

For example, for $n = 3$, packets sent will be tagged $0, 4, 2, 6, 1, 5, 3, 7$. If the threshold set by the router is $K = 3$, half of the packets will be discarded, without burstiness. Since packets are not precisely sent by the application at times $x/2^n.T_0$, the T_0 time period is split into 2^n time slots, during which only one packet can be sent.

This would be acceptable if sources were infinitely greedy, but this will not be the case in practice. Suppose that $n = 3$ as in the example above. An application sending at a constant bit rate of 4 packets per T_0 can have its packets tagged $0, 2, 1, 3$ if it starts sending at the beginning of the time period, or $4, 6, 5, 7$ if it starts transmitting one time slot later. In the second case, for a threshold of $K = 3$, all the packets will be discarded. Hence, the lowest available tag since the last transmission should be used. This is realized by the following tagging function:[4]

$$\mathcal{T}ag(\hat{t}, t) = Inf\{I(x\ [T_0]/T_0) \mid x \in]\hat{t}, t]\}$$

where $\hat{t}$ is the time of the previous transmission, and t is the current time. $x\ [T_0]$ is the remainder of x divided by T_0. Appendix A explains how this value can easily be computed. To avoid synchronization effects, a time offset can be randomly added between 0 and T_0 at the beginning of the UDP session.

3.1.1 Taking into account the datagram size

There is little limitation on the UDP datagram size (typically 64KB). Giving a tag per datagram, and thus setting the rate in datagrams per second does not make sense, as it will favor bursty applications that send very large datagrams.

If we consider that the transmission cost of a datagram is proportional to the number of IP fragments sent, then we would rather set the bandwidth in IP packets per second.

[4]Note that this definition does not require the partitioning of the time period into time slots

A tag could then be given to each IP fragment. However, since forwarding only part of the IP fragments is useless, we can give all fragments the "worst" tag, hereby defining the global UDP datagram tag :

$$\mathcal{T}ag(\hat{t}, t, size) = Inf^{\lceil size/MTU \rceil}\{I(x\ [T_0]/T_0) \mid x \in]\hat{t}, t]\}$$

$Inf^k(E)$ is the k^{th} smallest element in E. Appendix A explains how this value can easily be computed. MTU is set to 536 bytes to make the tagging algorithm independent from the path used by the flow.

3.1.2 Tagging and hierarchical encoding

To work with existing applications, we propose to have the system tag the datagrams. However, applications may want to explicitly define drop precedence among the data units, in particular, layered multimedia applications such as video-conferencing tools.

In $\mathcal{T}$UF, a layered application reserves bandwidths $b_1, b_2, \ldots, b_n$ (in packets/s) for each of its layers $1, \ldots, n$. Let's define the upper tag of each layer :

$$\begin{aligned} \mathcal{T}ag_0 &= 0 \\ \mathcal{T}ag_i &= \mathcal{T}ag_{i-1} + b_i/B_0 \end{aligned}$$

When a datagram is sent in layer i, the system computes the corresponding tag :

$$\mathcal{T}ag(\hat{t}_i, t, size, i) = \mathcal{T}ag(\hat{t}_i, t, size) + \begin{cases} \mathcal{T}ag_{i-1} & if\ \mathcal{T}ag(\hat{t}_i, t, size) + \mathcal{T}ag_{i-1} < \mathcal{T}ag_i \\ \mathcal{T}ag_n & \text{otherwise} \end{cases}$$

where $\hat{t}_i$ is the last transmission time in layer i.

Datagrams are tagged independently in the different layers. If the applications exceeds the reserved bandwidth in one of the layers, the datagram is sent with the lowest priority (in layer $n+1$). Datagrams from the higher layers have the highest tags, and will be dropped first. Depending on its fair share, a receiver will receive layer $1, 2, \ldots, i-1$ and part of layer i.

3.2 Determining threshold

By choosing a threshold on one of its interfaces, a $\mathcal{T}$UF router determines the rate of each constrained flow on that interface, i.e. $K.B_0$. A first objective is to try to find the highest threshold that will fully utilize the link's capacity. The second objective, is to have a threshold dynamics that is friendly with other types of flows, such as TCP connections.

This TCP-friendly rate control problem has been extensively studied in the literature. In general, the rate is adapted to the congestion state of the link, estimated using loss rate measurement, and/or round trip time.

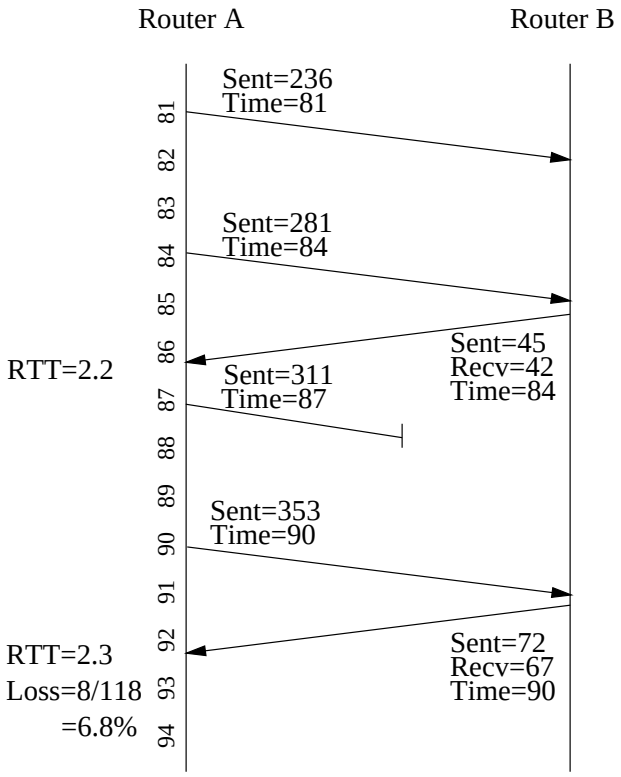

Figure 2: Router A probing link towards router B

These parameters are measured periodically by exchanging control messages between neighbor routers. As explained in [15], it is preferable to have a fixed re-computation interval rather than a multiple of the round trip time, since this latter increases significantly in case of congestion. The control messages announce the number of packets sent so far on the interface. The other router replies by informing the probing router of the corresponding number of received packets, as described in Figure 2. Once a $\mathcal{T}$UF router has collected a significant amount of reports, it can compute the loss rate and round trip time. These estimations can be smoothed using an exponential sliding window as for the RTT estimation in TCP ([27]). To deal with extreme congestion conditions, the sending rate is divided by 2 when the probing router receives no answer after a certain time period.

Here are possible strategies for evaluating the interface's threshold :

1. Linear increase - Multiplicative decrease. If the loss rate is zero, $K_i = K_{i-1} + C$, else $K_i = \gamma.K_{i-1}$, with $\gamma < 1$. The difficulty is to find appropriate constants C and γ.
2. TCP-friendly based formulae. For example, using the TCP model presented in [15],

$$K_i = 2.K_{i-1} \text{ if } p_i = 0 \text{ or :}$$

$$K_i = \frac{1}{B_0} min\left(\begin{array}{l} W_{max}/\overline{RTT}, \\ \dfrac{1}{\overline{RTT}.\sqrt{\frac{2bp}{3}} + t_0.min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)} \end{array} \right)$$

$\overline{RTT} = max(R_{UDP}, RTT_{estimated})$. R_{UDP} slackens the rate of UDP traffic on links with very short delays (see Section 5).

3. Other rate adaptation schemes, for example :

$$K_i = \omega^- * K_{i-1} * (1-p) \text{ for } p > 0 \text{ with } \omega^- < 1$$
$$K_i = \omega^+ * K_{i-1} \text{ for } p = 0, \text{ with } \omega^+ > 1$$

The threshold adaptation algorithm tries to estimate the available bandwidth on the link. If we have only UDP controlled traffic on the link, the unique objective is to find a good and stable estimation of this fixed bandwidth. Strategy 3 (with $\omega^+ = 1.07, \omega^- = 0.95$) seems to provide a good average bandwidth utilization. However, friendliness becomes an issue when other flows, such as TCP connections, use the link. We have chosen strategy 2 to estimate the threshold in a TCP-friendly manner. Future work should be done on the evaluation and comparison of other TCP-friendly rate control algorithms.

4 Reliable multicast

Congestion control is traditionally a concern for reliable multicast protocols since error recovery and congestion control are often mixed. The purpose of this section is to show that reliable multicast can also benefit from the use of $\mathcal{T}$UF and multi-rate sessions, de-correlating congestion control and error recovery.

When considering reliable multicast data transfers, we should make a distinction between finite and the infinite data transfers. In the first case, providing different bandwidths to the receivers increases their satisfaction, since it is inversely proportional to the duration needed to retrieve the data. In the second case (continuous data streams), applications are intrinsically single-rate oriented, and the information rate is limited by the worst channel capacity[5]. We can however also benefit from the use of $\mathcal{T}$UF.

Forward Error Correction (FEC) is particularly well adapted to the multicast context and can be combined with ARQ (retransmission requests) to achieve reliable transmission [28]. Processing power is not such an important issue for Forward Error Correction, since powerful FEC packet encoders and decoders are now proposed, based on Turbo codes [29], or Vandermonde matrices [30]. The schemes proposed below therefore rely on Forward Error Correction for error recovery.

[5] Note that we use the word "capacity" and not "rate" to refer to Shannon's capacity, or goodput for the binary erasure channel. A receiver with a very high bandwidth and a high loss rate (e.g. on a wireless link) can have a low capacity.

4.1 Finite data transfers

Layered encoding schemes have been proposed [26, 31] for the organization of data and redundancy (FEC) packets. These organization schemes enable receivers to retrieve the data in an almost optimal time, regardless of their receiving rate, i.e. of the number of layers joined. Using the tagging algorithm proposed in Section 3.1.2, this hierarchically encoded data can be sent on a single multicast group. Congestion is automatically controlled in the network by the $\mathcal{T}$UF routers. Receivers listen to the multicast group until they have enough information (original data+redundancy packets) to rebuild the original file. The only feedback needed is at the end of the transfer, to inform the source that it can stop transmitting. The data organization proposed in [26] combined with the tagging algorithm (3.1.2), enables us to have a fine granularity on the reception rate, and thus accomplish near-to-optimal reception time.

4.2 Infinite data transfers

Since we have finite buffers in our hosts, and since we tolerate no loss, the rate at which information is sent from the source is limited by the worst channel capacity between the source and the members of the group. Even though this means that the session is intrinsically single-rate oriented, from an informational standpoint, this does not mean that all receivers must receive the same amount of redundancy.

Supposing that a source can estimate the worst channel capacity, it can send the data in two layers :

- The original data layer, that has the highest priority. The bandwidth reserved for that layer is slightly below the worst channel capacity.
- A redundancy layer, which will enable fast receivers to deal with higher loss rates, reducing retransmission requests.

With this scheme, heterogeneous receivers that have the same capacity, some with high loss rates but high bandwidth, others with lower bandwidth but lower loss rates, can all achieve their capacity thanks to the multi-rate session.

Feedback is necessary because of limited buffer size at the source. However, it is not related to congestion control, and acknowledgments or negative-acknowledgments can be sent on a much larger time-scale. Moreover, the use of exponential timers can make this feedback scalable [32].

Feedback is also used by some receivers to inform the source of their receiving rate. The source is particularly interested in the worst receiving capacity, for the determination of the lowest layer's bandwidth, and the best receiving rate, to avoid generating useless FEC packets (and reduce redundancy in the network - see Section 5.1).

5 Discussion on fairness

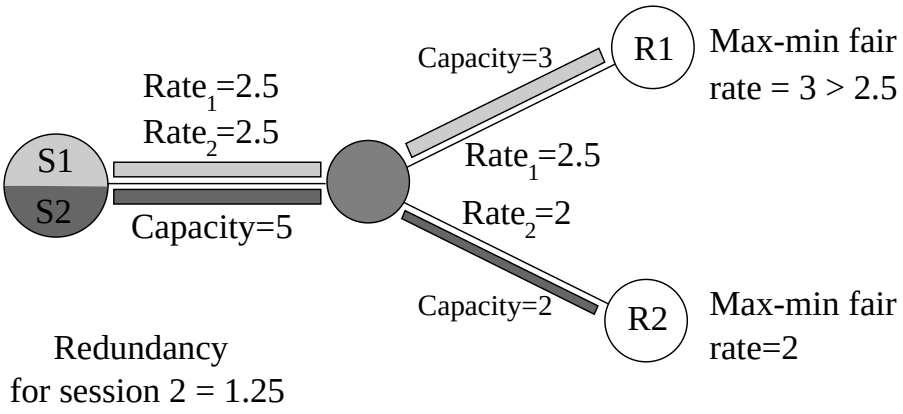

Figure 3: Inefficient bottleneck link utilization

5.1 Intra-protocol fairness

The basic principle of $\mathcal{T}$UF filters is that they provide to a flow either all the bandwidth received from upstream (i.e. packets from this flow are all tagged below the threshold), or the maximum bandwidth allocated to all flows on this link (i.e. $K.B_0$, where K is the threshold). This leads to interesting fairness properties, as defined in [9], such as same-path-receiver-fairness. Receivers are not constrained by other members of the group. However, we do not always have global max-min fairness as shown on Figure 3.

Under some circumstances, where multiple links are congested on a same data path, resources on a link may be used inefficiently. Bandwidth is wasted to transmit data unused by downstream receivers, and could have been used by another flow. This is due to the "local" nature of our congestion control scheme, and the absence of flow state in the routers. No flow information concerning downstream receivers is collected at the link. This inefficiency is measured by the **redundancy** defined in [9] as the ratio between the bandwidth used by a flow on a link and the maximum bandwidth received by downstream receivers of the flow. A link is efficiently used if the redundancies of all the flows equal 1. Example of globally efficient network configurations are those where receivers are present at all nodes of the multicast tree. Rubenstein shows in [9] that redundancy has a negative impact on max-min fair bandwidth allocations. Redundancy also exists in layered congestion control protocols due to the difficulty of coordinating join and leaves. Future work should be done on our scheme to evaluate and limit the effects of redundancy.

5.2 Fairness with TCP

Achieving intra-protocol max-min fairness and being fair with TCP, even if we consider only same-path-receiver fairness, are incompatible, since TCP does not provide such fairness properties with itself. TCP has biases against connections with multiple congested gateways

or with longer round trip times [33]. Our objective is therefore to achieve bounded fairness introduced in [6] :

$$\exists\ \alpha, \beta \mid \forall \text{ receiver r},\ \ \alpha\lambda_r^{UDP} \leq \lambda_r^{TCP} \leq \beta\lambda_r^{UDP}$$

This guarantees that for every receiver, nor the TCP connection, nor the UDP multicast connection collapses. Appendix B suggests the following values for α and β : $\alpha = \frac{R_{UDP}}{R_{max}\, n(n+1)}$ and $\beta = \frac{R_{UDP}}{R_{min}}$, where R_{min} and R_{max} are bounds on the round trip time, R_{UDP} is the minimum round trip time considered for the computation of the threshold (Section 3.2) and n is the maximum number of congested links.

We here clearly see the bias of TCP again long round trip times and multiple congested gateways, since the low bound α rapidly decreases as R_{max} and n increase.

If we consider $R_{min} = 5\ ms$, $R_{max} = 500\ ms$, with only one bottleneck on each path ($n = 1$), and $R_{UDP} = 50\ ms$, we can give the following bounds on fairness :

$$\alpha = 1/20, \beta = 10$$

If these bounds are not satisfactory, another way of enforcing fairness with TCP is to use class based queuing between TCP, UDP controlled traffic, and UDP-non-controlled traffic. $\mathcal{T}$UF routers will estimate the available bandwidth for UDP-controlled traffic. Intra-protocol fairness is achieved with the tagging mechanism.

6 Simulations

We simulated $\mathcal{T}$UF with the NS simulator [34]. In the chosen topology, there are 4 types of receivers[6] A-1, A-2, B-1, B-2. Receivers A-2 and B-2 share a common and unique bottleneck of respectively 1 MB/s and 500 KB/s. In our simulations, there is an equal number of nodes for each type. Receivers A-1 and B-1 represent the slow receivers, behind a second congested link, as shown on Figure 4. This topology enables us to evaluate the max-min fairness between :

- receivers behind a same bottleneck,
- receivers behind different bottlenecks,
- and receivers behind multiple congested links.

$\mathcal{T}$UF filters are installed on each outgoing interface of the source node and routers A and B. The bottleneck bandwidths of the A-1 and B-1 receivers vary between 40KB/s and 450 KB/s. All nodes subscribe to all multicast flows, which originate from the same source. Simulations are run for 1000 seconds. TCP and UDP packets are both 1000 bytes long. All the links have a 50 ms delay.

[6]Remember that receivers are considered at transport layer, i.e. a node embodies multiple receivers, one for each multicast group.

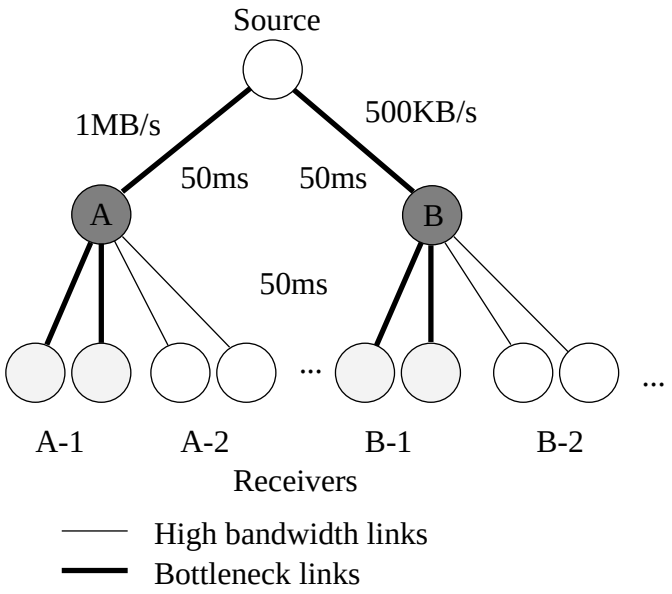

Figure 4: Simulation Topology

In the first series of simulation, we evaluate the intra-protocol fairness of $\mathcal{T}$UF. Sources are greedy and generate CBR traffic at 10 MB/s. We compare the throughput at a receiver with its max-min fair share for a varying number of multicast groups and members. For our topology, the max-min fair share of a receiver is equal to the bandwidth of its bottleneck link divided by the number of flows. Figure 5 and 6 show the average bandwidth/fair-share ratios, and the vertical lines shows the minimum and maximum values for this ratio. The average ratio is 95% and its worst value is 80% (for a simulation with 76 receivers). $\mathcal{T}$UF therefore meets our max-min fairness objective. The threshold adaptation mechanism (Strategy 3) results in a slight network underutilization (approximately 5 to 10%), which explains the value of 95%, but this impacts equally on all the receivers who get approximately their fair share.

In the second simulation, we show that $\mathcal{T}$UF also gives its fair share to receivers of non greedy sources. In this simulation, 15 multicast sessions were started, generating CBR traffic at rates varying between 15 KB/s and 250 KB/s. Figure 7 shows the rates received for receivers with various demands. Receivers of non greedy sources, i.e. that ask less than their fair share, are given all the requested bandwidth, and filtering concerns only the most greedy flows. We note that the sum of received rates is approximately 1 MB/s.

The third simulation presents the behavior of the $\mathcal{T}$UF tagging and filtering mechanisms. Packet tags are traced over the 1MB link, and Figure 8 presents their distribution. First, we notice that the tagging algorithm chooses tags uniformly in the interval $[0, 200/B_0]$. Since

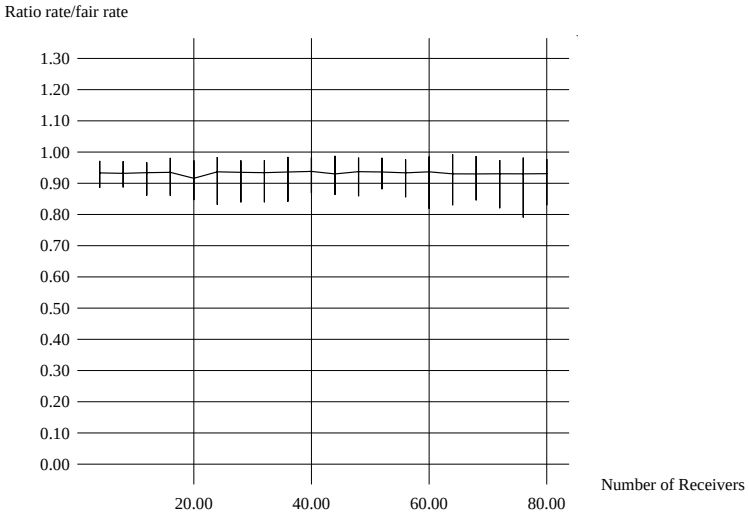

Figure 5: Receiver's rate / max-min fair rate for a varying number of receivers and 5 multicast groups

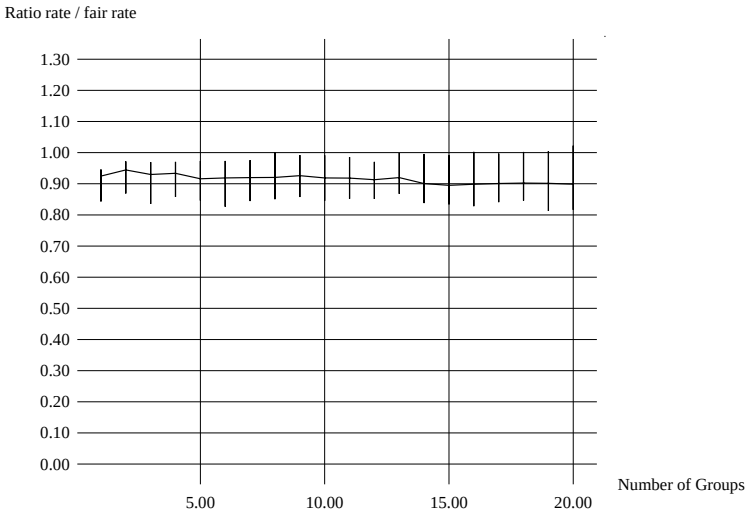

Figure 6: Receiver's rate / max-min fair rate for a varying number of multicast groups and 5 nodes of each type

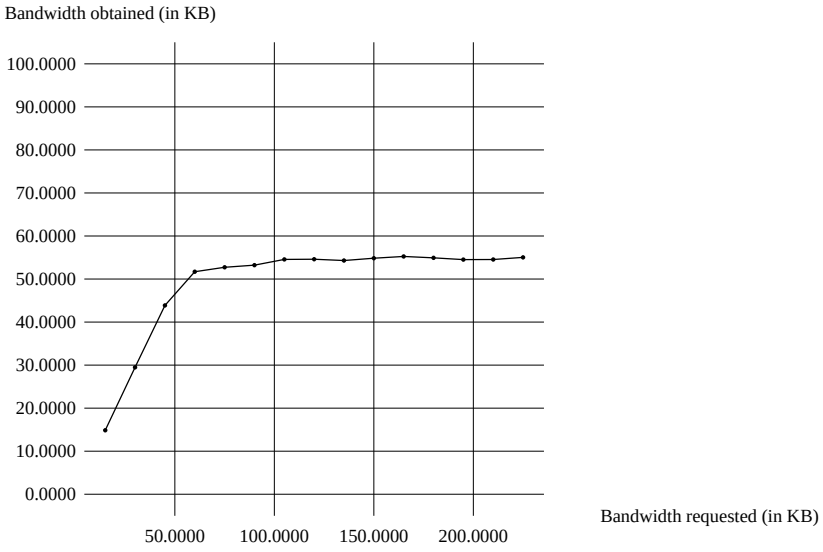

Figure 7: Bandwidth share among 15 non-greedy groups over the 1MB link

200 packets/seconds corresponds to the bandwidth allocated per flow on the 1MB link (the simulation was run with 5 groups), all tags below the threshold are used. Our tagging algorithm thus provides correct, uniform and efficient tagging.

Figure 8 also shows that packet forwarding rapidly decreases as tags reach the threshold. This proves that the bandwidth estimation algorithm is quite stable. The loss probability, due to the threshold adaptation mechanism, is approximately 2 to 3 %.

Finally, our last set of simulations shows that we achieve bounded fairness between UDP controlled traffic and TCP connections. In each simulation, we had an equal amount of UDP flows and TCP connections. Half of the TCP connections end in A, the other half extend to an A-2 node. The UDP flow achieves the same throughput in both cases, whereas the TCP connection achieves a smaller throughput for the A-2 receiver, due to the bias against long round trip times. We see, however, that neither the UDP connection, nor the TCP connection collapses, and that the ratio of their throughput stays within the predicted bounds (R_{min}=50ms, R_{max}=450ms, R_{UDP}=150ms, n=1):

$$\alpha = 1/6, \beta = 3$$

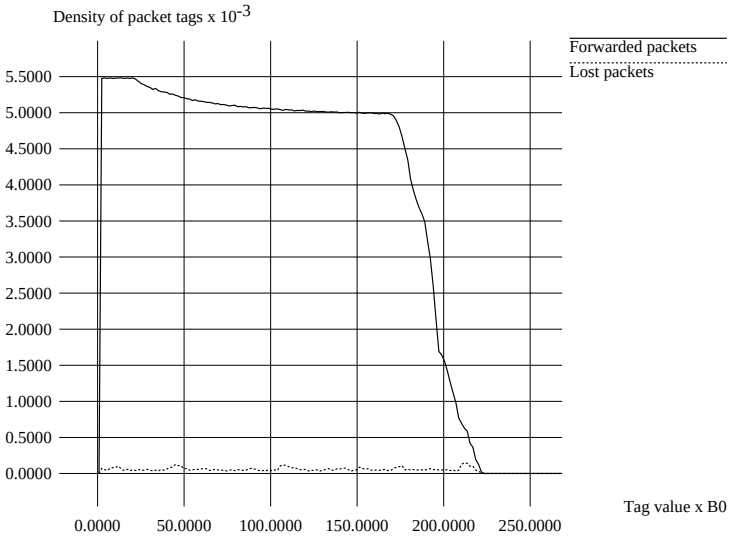

Figure 8: Density of packets tags over the 1MB link. $B_0 = 10^{-4} pkt/s$

7 Conclusion and future work

Congestion control for multi-rate multicast sessions must rely on mechanisms in the network. The protocol $\mathcal{T}$UF proposed here is a simple alternative to layered encoding/transmission schemes, that requires adding simple router filters in the network. We think it is particularly appropriate for congestion control in heterogeneous multicast environments, although it can also be used for unicast flows.

We are currently working on a real implementation. The TOS field being too small, we propose to put the tag in a 4-bytes IP option, with the *copied* flag set, to copy the option in each IP fragment. Packet tagging is done in the source's system, thus working with existing applications. Hierarchical applications that wish to define drop precedence among packets can inform the system of the layer to use on a per-packet basis.

To account for cheaters and non-controlled UDP flows, UDP packets can be tagged or re-tagged on a per-flow basis at the boundaries of a network. The rate of UDP traffic can also be controlled only over some links of the Internet, allowing for incremental deployment. Our approach requires modifications in the routers, and adds some (limited) overhead for bandwidth estimation and tag comparison. However, we believe that by its simplicity, $\mathcal{T}$UF is an interesting approach to congestion control for multi-rate multicast sessions.

References

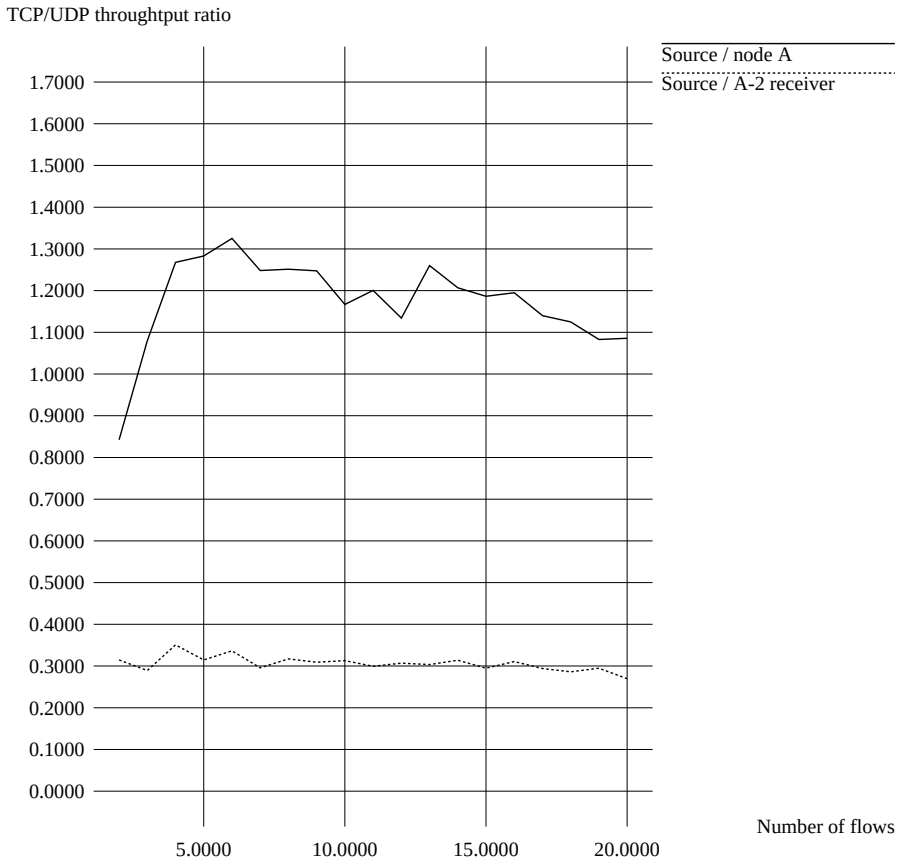

Figure 9: $\lambda_{TCP}/\lambda_{UDP}$ for a varying number of flows

[1] A. Legout, J. Nonnenmacher, and E. W. Biersack, "Bandwidth allocation policies for unicast and multicast," in *Proceedings of IEEE Infocom'99*, March 1999.

[2] Rajendra K. Jain, Dah-Ming W. Chiu, and William R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," Tech. Rep. TR-301, DEC, Littleton, MA., September 1984.

[3] Dimitri Bertsekas and Robert Gallager, *Data Networks*, chapter 6, pp. 524–529, Prentice-Hall, 1987.

[4] Frank Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, 1997.

[5] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan, "Rate control for communication networks : shadow prices, proportionnal fairness and stability," *Journal of the Operationnal Research Society*, vol. 49, pp. 237–252, 1998.

[6] Huayan Amy Wang and Mischa Schwartz, "Achieving bounded fairness for multicast and tcp traffic in the internet," in *Proceedings of ACM Sigcomm'98*, September 1998.

[7] Injong Rhee, Nallathambi Ballaguru, and George N. Rouskas, "Mtcp : Scalable tcp-like congestion control for reliable multicast," Tech. Rep. TR-98-01, Department of Computer Science, North Carolina State University, January 1998.

[8] Supratik Bhattacharyya, Don Towsley, and Jim Kurose, "The loss path multiplicity problem in multicast congestion control," in *Proceedings of IEEE Infocom'99*, New York, March 1999.

[9] Dan Rubenstein, Jim Kurose, and Don Towsley, "The impact of multicast layering on network fairness," in *Proceedings of ACM SIGCOMM'99*, 1999.

[10] Tianji Jiang, Ellen W. Zegura, and Mostafa Ammar, "Inter-receiver fair multicast communication over the internet," in *Proceedings of NOSSDAV'99*, 1999.

[11] Tianji Jiang, Mostafa H. Ammar, and Ellen W. Zegura, "Inter-receiver fairness : A novel performance measure for multicast abr sessions," in *Proceedings of ACM Sigmetrics'98*, Madison, Wisconsin, June 1998.

[12] S. Jamaloddin Golestani and Krishan K. Sabnani, "Fundamental observations on multicast congestion control in the internet," in *Proceedings of IEEE Infocom'99*, March 1999.

[13] Todd Montgomery, "Loss tolerant rate controller (ltrc)," Tech. Rep. NASA-IVV-97-011, NASA, August 1997.

[14] Mark Handley and Sally Floyd, "Strawman specification for tcp friendly (reliable) multicast congestion control (tfmcc)," proposed to the RM mailing list, November 1998.

[15] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose, "Modeling tcp throughput: A simple model and its empirical validation," in *Proceedings of ACM SIGCOMM'98*, 1998.

[16] Dorgham Sisalem and Henning Schulzrinne, "The loss-delay based adjustment algorithm: A tcp-friendly adaptation," in *Proceedings of NOSSDAV'98*, July 1998.

[17] Dorgham Sisalem, Frank Emanuel, and Henning Schulzrinne, "The direct adjustment algorithm: A tcp-friendly adaptation scheme," Tech. Rep., GMD Fokus, August 1997.

[18] Yuan Gao and Jennifer C. Hou, "Racoom: A rate-based congestion control scheme for multicasts," Tech. Rep., The Ohio State University, Dept. of Electrical Engineering., 1999.

[19] Reza Rejaie, Mark Handley, and Deborah Estrin, "Rap: An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *Proceedings of IEEE Infocom'99*, March 1999.

[20] Steven McCanne, Van Jacobson, and Martin Vetterli, "Receiver-driven layered multicast," in *Proceedings of ACM SIGCOMM'96*, 1996.

[21] Lorenzo Vicisano, Luigi Rizzo, and Jon Crowcroft, "Tcp-like congestion control for layered multicast data transfer," in *Proceedings of IEEE Infocom'98*, 1998.

[22] Thierry Turletti, Sacha Fosse Parisis, and Jean Bolot, "Experiments with a layered transmission scheme over the internet," Tech. Rep. RR-3296, INRIA, November 1997, http://www.inria.fr/RRRT/RR-3296.html.

[23] B. N. Levine and J.J. Garcia-Luna-Aceves, "Improving internet multicast with routing labels," in *Proc. IEEE International Conference on Network Protocols*, October 1997, pp. 241–50.

[24] Michael Luby and Lorenzo Vicisano, "Heterogeneous multicast congestion control based on router packet filtering," Document sent to the RM mailing list, June 1999.

[25] Shu Liu, "Multicast congestion control," Document sent to the RM mailing list, June 1999.

[26] Antoine Clerget and Walid Dabbous, "Organizing data transmission for reliable multicast over satellite links," in *Proceedings of Africom CCDC'98, Internet and Global Networking*, Tunis, October 1998, pp. 71–80.

[27] W. Richard Stevens, *TCP/IP Illustrated*, vol. 1, pp. 299–301, Addison Wesley, 1994.

[28] Jorg Nonnenmacher, Ernst Biersack, and Don Towsley, "Parity-based loss recovery for reliable multicast transmission," in *Proceedings of ACM SIGCOMM'97*, 1997, pp. 289–300.

[29] John W. Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proceedings of ACM SIGCOMM'98*, September 1998.

[30] Luigi Rizzo, "Effective erasure codes for reliable computer communication protocols," *Computer Communication Review*, April 1997.

[31] Lorenzo Vicisano, "Notes on a cumulative layered organization of data packets across multiple streams with different rates," University College London, Computer Science, Research Note RN/98/25, 1998, http://www.cs.ucl.ac.uk/external/L.Vicisano/pubb/layers.ps.gz.

[32] Jorg Nonnenmacher and Ernst Biersack, "Scalable feedback for satellite broadcasts," in *ACM/IEEE MobiCom'97 Workshop on Satellite-based Information Services (WOSBIS'97)*, October 1997, pp. 15–21.

[33] Sally Floyd and Van Jacobson, "On traffic phase effects in packet-switched gateways," *Internetworking: Research and Experience*, vol. 3, no. 3, pp. 115–156, September 1992.

[34] Steve McCanne and Sally Floyd, "Ucb/lbnl/vint network simulator (ns) version 2.1b5," http://www-mash.cs.berkeley.edu/ns/, 1999.

8 Appendix A : Computing $\mathcal{T}ag(\hat{t}, t, size)$

Here is an algorithm to compute $\mathcal{T}ag(\hat{t}, t, size)$:[7]

- Let $\hat{x} = (\hat{t}\ [T_0])/T_0$
- Let $x_1 = (t\ [T_0])/T_0$
- Let $x = \begin{cases} x_1 & \text{if } x_1 > \hat{x} \\ x_1 + 1 & \text{if } x_1 \leq \hat{x} \end{cases}$
- Let $n = Inf\{i \in \mathbb{N}\ |\lfloor 2^i.x \rfloor - \lceil 2^i.\hat{x} \rceil \geq size - 1\}$
- Let f be the following recursive function : (a, b, k, n are integers)

$$f(a, b, k, n) =$$

$$\begin{cases} 0 & \text{if } n = 0 \\ f(\lfloor \frac{a+1}{2} \rfloor\ [2^{n-1}], \lfloor \frac{b}{2} \rfloor, k, n-1) & \text{if } k < d/2 \\ f(\lfloor \frac{a}{2} \rfloor, \lfloor \frac{b-1}{2} \rfloor\ [2^{n-1}], k-d, n-1) + 2^{n-1} & \text{if } k \geq d/2 \end{cases}$$

$$\text{and } d = (b-a)\ [2^n] + \begin{cases} 1 & \text{if a and b are even} \\ 0 & \text{otherwise} \end{cases}$$

- $\mathcal{T}ag(\hat{t}, t, size) =$

$$\begin{cases} \text{if } t \leq \hat{t} & \text{Not defined} \\ \text{if } t \geq \hat{t} + T_0 & size \\ otherwise & f(\lceil 2^n.\hat{x} \rceil, \lfloor 2^n.x \rfloor\ [2^n], size - 1, n) \end{cases}$$

- $\mathcal{T}ag(\hat{t}, t) = \mathcal{T}ag(\hat{t}, t, 1)$

In practice, the tag value is limited by its field size s in the packet. The size of the tag computed in bits is given by the integer n. Therefore, the value ∞ can be returned by the tagging algorithm when n exceeds s. The number of iterations to compute the tag is then bounded by $2s$:

- s to compute the value of n
- $n \leq s$ to compute the value of f

Since these iterations consist in very simple operations (binary shift, and, or, add, subtract), tag computation requires very little computing power.

[7] $\lfloor x \rfloor$ refers to the largest integer smaller than x. $\lceil x \rceil$ refers to the smallest integer larger than x.

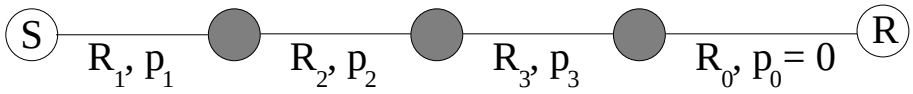

Figure 10: Model of the source-receiver path

9 Appendix B : Bounds on fairness with TCP

We can roughly model the path between the source and a receiver as the succession of $n \geq 1$ congested links (i.e. on which the loss rate is not zero), followed by a pure delay link (i.e. loss rate = 0), as shown on Figure 10.

$$\lambda_r^{TCP} \approx \frac{C}{R\sqrt{p}}, \text{ and } \lambda_r^{UDP} \approx \frac{C}{Max_{i=0}^{n}(Max(R_{UDP}, R_i)\sqrt{p_i})}$$

$$\begin{aligned}
\frac{\lambda_r^{TCP}}{\lambda_r^{UDP}} &\approx \frac{Max_{i=0}^{n}(Max(R_{UDP}, R_i)\sqrt{p_i})}{R\sqrt{p}} \\
&= \frac{Max_{i=0}^{n}(Max(R_{UDP}, R_i)\sqrt{p_i})}{(\sum_{i=0}^{n} R_i)\sqrt{1 - \prod_{i=1}^{n} 1 - p_i}} \\
&\leq \frac{Max_{i=0}^{n} \frac{R_{UDP}.R_i}{R_{min}}\sqrt{p_i}}{(\sum_{i=0}^{n} R_i).Max_{i=1}^{n}\sqrt{p_i}} \\
&\leq \frac{R_{UDP}}{R_{min}} = \beta
\end{aligned}$$

$$\begin{aligned}
\frac{\lambda_r^{TCP}}{\lambda_r^{UDP}} &\approx \frac{Max_{i=0}^{n}(Max(R_{UDP}, R_i)\sqrt{p_i})}{(\sum_{i=0}^{n} R_i)\sqrt{1 - \prod_{i=1}^{n} 1 - p_i}} \\
&\geq \frac{Max(R_{UDP}, R_B).\sqrt{p_B}}{(\sum_{i=0}^{n} R_i)\sqrt{\sum_{i=1}^{n} p_i}} \\
&\geq \frac{Max(R_{UDP}, R_B).\sqrt{p_B}}{(\sum_{i=0}^{n} R_i)\sum_{i=1}^{n}\sqrt{p_i}} \\
&\geq \frac{1}{(\sum_{i=0}^{n} R_i)\sum_{i=1}^{n} 1/Max(R_{UDP}, R_i)} \\
&\geq \frac{R_{UDP}}{R_{max}.n(n+1)} = \alpha
\end{aligned}$$

where R_{min} and R_{max} are bounds on the round trip time, R_{UDP} is the minimum round trip time considered for the computation of UDP's bandwidth (Section 3.2) and n is the maximum number of congested links.

Contents

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399