A study of a simple preventive transport protocol

Fabien Chatté, Bertrand Ducourthial, Silviu-Iulian Niculescu

A study of a simple preventive transport protocol

Fabien Chatté, Bertrand Ducourthial, Silviu-Iulian Niculescu
Lab. HEUDIASYC (UMR CNRS 6599)
Université de Technologie de Compiègne,
Centre de Recherche de Royallieu, BP 20529, F-60205 Compiègne CEDEX, France.

***Abstract*— The main qualities of a protocol for multimedia flows transportation are related to the way congestions are handled. This paper addresses the problem of end-to-end congestion control performed in the Internet transport layer. We present a simple protocol called Primo, that determines the appropriate sending rate in order to maximize network resources usage and minimize packets loss. Comparison with existing transport protocols (TCP Reno, Sack, Vegas and TFRC) are considered, regarding various efficiency criteria such as sending and reception rates stability, loss rate, resources occupancy rate and fairness.**

***Abstract*— Les principales qualités d'un protocole pour les flots multimédia concernent la gestion des congestions. Cet article s'intéresse au problème du contrôle de congestion de bout en bout effectué dans la couche transport d'Internet. Nous présentons un protocole simple appelé Primo, qui détermine le taux d'émission approprié dans le but de maximiser l'utilisation des ressources réseaux et de minimiser la perte des paquets. Des comparaisons avec des protocoles de transport existants (TCP Reno, Sack, Vegas et TFRC) sont analysées en prenant en compte différents critères tels que la stabilité des débits d'émission et de réception, le taux de perte, l'occupation des ressources et l'équité.**

I. Introduction

A. Congestions

Congestion analysis in the Internet started by the end of the 80s. Roughly speaking, a *congestion* is caused by the incapacity of the routers to temporarily store some packets before re-sending them. In the reliable transport protocol TCP (more than 90% of the Internet flows), these packets have to be resent, fact that consumes not only network resources, but also decreases the throughput of network applications. Furthermore, congestions may appear in any IP network, and concern both network users and administrators. Indeed, when the network is congested, the quality of reception is reduced mainly due to the throughput reduction, transport delay augmentation, packets loss *etc.*

In the TCP/IP standard architecture, the transport layer is in charge of congestions. When a flow encounters a congestion, the sending rate of the source should be adapted to avoid congestion aggravation. The adjustment of the sending rate is one of the main characteristics of transport protocols. Multimedia flows imply specific adjustment algorithms to guarantee good receptions (eg. avoiding reception freezing).

B. Related work

In this paper, we focus on the end-to-end congestion control performed in the Internet transport layer. A lot of work has been done in this field, such as, among others, TCP Vegas [1], RAP [16], TFRC [9] (see, for instance, [4] for a state of the art).

A congestion control mechanism is either preventive or corrective. A *corrective* policy (*e.g.*, TCP) reduces the sending rate after a congestion has been detected. A *preventive* policy attempts to reduce the

Part of this work was funded by the French Ministry of Reasearch (ACI program, 2001-2004) as well as the French innovation agency (ANVAR, 2001-2003).

F. Chatté is now at Neopost Industry.

Corresponding author: Bertrand.Ducourthial@hds.utc.fr. Phone: +33.3.44.23.46.46, Fax: +33.3.44.23.44.77

sending rate before a congestion appears. These two behaviors can not be used concurrently. Indeed, when the preventive protocol detects a risk of congestion, it reduces its sending rate, whereas the corrective one continues to increase its sending rate, until a packet loss is observed. Hence, the corrective protocol will obtain all the available bandwidth, at the expense of the preventive one. The so-called TCP-friendly protocols are able to fairly share the bandwidth with TCP while offering a congestion control mechanism more suitable for real-time flows.

C. Objectives

Our study deals with the impact of some combined advances on congestion control, in the aim of maximizing network resources usage during multimedia flows transportation, while minimizing packets loss. To maximize available bandwidth utilization, the sending rate should *rapidly* reach its optimal rate. When this steady state behavior is reached, the sending rate should be *smoothly* adjusted. Indeed, strong variations of the sending rate affect the network stability and the reception rate, which represents a major drawback for multimedia flow. When network conditions are changing, the sending rate should be *rapidly* and *moderately* adapted in order to avoid network saturation. Indeed, when the network is saturated, the packets loss increases.

As already mentioned, congestion control schemes can either be *preventive* or *corrective*. We deliberately investigate *preventive* schemes, since such an approach seems to lead to better results. The drawback is the bad cohabitation with current transport protocols in the Internet. But some recent studies showed that preventive protocols could be modified on the fly to become more "aggressive" when a corrective protocol is suspected to share the same router queues [10]. Moreover, preventive protocols could be used in small private networks, such as networks of communicating mobile devices. Such ad hoc networks are in rapid expansion.

To summarize, in order to maximize network resources and minimize packets loss during multimedia flows transports, we study a transport protocol that rapidly reaches the optimal rate, smoothly adapts the sending rate, and reacts rapidly, moderately and preventively to congestion formation. Its simple congestion control scheme relies in part on a combination of several independent advances observed in recent works.

D. Contributions

In this paper, we present a preventive transport protocol called Primo, that achieves the objectives mentioned above: it rapidly reaches the optimal rate, smoothly adapts the sending rate, and rapidly but moderately reacts to congestion formation. Moreover, the protocol is very simple, that means implementation facility. Primo does not present the drawbacks of the reactive schemes, the congestion window or the congestion controls based on RTT. Among others, Primo integrates several ideas from [1], [9], [16], [18], and our contribution mainly consists in emphasizing the powerful of an original and simple combination of such advances.

Extensive simulations under *network simulator* [13] have been done, first for studying the main proposed protocols in order to design Primo, and then for evaluating our protocol. We developed a rigorous methodology, allowing the comparison of protocols with seven *efficiency criteria*, while varying many parameters [2]. In this paper, we present a summary using five efficiency criteria. Primo is compared with TCP Reno [17] (one of the most used version on Internet), TCP Sack [7] (the most powerful version used on Internet), TCP Vegas [1] (the only version of TCP which implements a preventive behavior) and TFRC (one of the best known alternative to TCP, which is TCP-friendly).

Based on the results obtained, Primo protocol is very encouraging. We end this paper by a discussion about future work.

II. Primo basics

In this section, we justify the main choice we made to design Primo in order to reach our objectives.

A. Controller definition, and characteristics

Primo uses a simple linear controller, which simplifies its implementation; it means PRoportional Integral MOdified. The proportional component allows to adjust "moderately" the sending rate, that is, proportionally to the network perturbation. The integral component takes care of the past sending rates values, in order to improve stability by avoiding short and strong variations due to transient failures.

Note that a derivative component could help to anticipate variations. However, it is sensitive to the instability induced by input delay uncertainty. Inputs are obtained and derived from the network or the destination to manage the sending rate of the source, and the delay uncertainties become more important when the network approaches the saturation. In conclusion, it seems clear that a derivative component is not well suited for congestion control.

B. Input variables

Input variables allow to give indirect information on the network state. The literature reports the use of *three common variables*: Explicit Congestion Notification (ECN) [15], [14], Round Trip Time (RTT), and reception rate.

The ECN option gives mainly a binary information about the network congestion state by marking some packets; this is not sufficient for our purpose.

Next, the RTT is equal to the delay between the arrival of an acknowledgement and the departure of the related packet. It becomes clear that a network congestion leads to large values of the RTT (packets spend more time in the routers queues). TCP congestion mechanism is mainly based on the variations of the RTT (more precisely on the mean value of the RTT combined with the loss rate). Unfortunately, the RTT does not allow to differentiate in which way the congestion appeared: forward way (from the source to the destination) or backward way (from the destination to the source).

Since the main influence of a source is on the forward way, Primo uses the Forward Trip Time (FTT), in order to avoid unnecessary rate reduction. The FTT, similar to the ROTT in [18], is equal to the delay between the arrival of a packet at the destination and its departure from the source. It cannot be exactly measured since the source and destination clocks are not necessarily synchronized. However, such a mismatching does not disturb the control task since it uses the FTT variations and not the exact values. If the source and destination clocks have not the same precision, the FTT will increase or decrease monotonically. The clock deviation can be generally neglected. Nevertheless, note that it could be computed and compensated using an algorithm like ESRS [19].

The gap measured between the smallest FTT observed in the past and the current FTT represents a good estimation of the network congestion state. Primo adjusts the source rate *proportionally* to this *measured error*.

However, our simulations showed that, when using the FTT, the source can prevent congestion formations, but has some difficulties to stabilize its sending rate. As explained above, this cannot be acceptable for multimedia flows.

Assume now that there exists a congestion on the forward way. Then some packets will spend more time in a router queue before reaching the destination, and other packets are destroyed. The bandwidth allocated for the flow corresponds to the reception rate, that is, the rate measured by the destination. Hence, when the gap between the FTT reference value and the current FTT is positive (*i.e.*, when a congestion is forming), the sending rate of the source will be computed using the reception rate of the destination. Note that the slow start of TFRC uses the reception rate in order to avoid destination saturation [9].

To summarize, *Primo uses the variations of the FTT to estimate the degree of congestion, and the reception rate to precisely adjust the sending rate in case of congestion*.

C. Output variable

For a source, the only way to avoid a network congestion is to adjust its sending rate. In practice, this rate can be adjusted

by varying the size of a sending window (as TCP does), or the duration of an Inter Packet Gap (IPG) (as RAP [16] or TFRC [9] do).

Our simulations showed that, when the congestion control mechanisms are based on a sending window, they are more vulnerable to packet loss [2]. Moreover, they have more difficulties than others to maintain a constant sending rate when propagation delays are important. Finally, they lead to unfair bandwidth sharing in presence of heterogeneous propagation delays.

Based on these simple remarks, it seems reasonable to use the *Inter Packet Gap* for *adjusting the sending rate*. We have chosen this option for Primo.

III. "PRIMO INTERNALS" AND RELATED ALGORITHM

In this section, we describe the main characteristics of Primo.

A. Computing variables

The FTT is computed by the destination node at each packet arrival, by subtracting the departure date (contained in the header of the packet) to the arrival date. The (instantaneous) reception rate is updated at each packet arrival by dividing the amount of data received by the time elapsed from the last reception. Both the FTT and the instantaneous reception rate are written in the header of the packet acknowledgment sent back to the source.

At each acknowledgment arrival, the source checks if the reference value of FTT has to be updated to ensure that it is always equal to the smallest measured FTT. Moreover, the source computes the difference ΔFTT between the instantaneous FTT and its corresponding reference value, and stores the result in ARRAYΔFTT, an array able to contain all the ΔFTT from a fixed (for a given connection) delay D (in milliseconds). This array is used for defining the integral component. The source also computes the reference RTT ($\mathtt{RTT}_{\mathrm{ref}}$) in order to set the retransmission time out TO ($\mathtt{TO} = 2 \times \mathtt{RTT}_{\mathrm{ref}}$).

B. Starting phase

In order to estimate the initial available bandwidth, the source begins by sending a burst of four packets with a null inter packet gap. The available bandwidth is computed by the destination node by dividing the size of three packets by the delay between the first arrival and the fourth one.

The first acknowledgement is sent after the reception of the first four packets. It contains the FTT of the first received packet and the estimated available bandwidth. At the reception of the first acknowledgment, the source will set the reference value FTT with the received FTT, the sending rate with the available bandwidth estimated by the destination, and the retransmission time out TO with the first RTT. All the variables are now initialized and the sending rate adjustment begins.

Note that if one of the four first packets or the first acknowledgment are lost, the whole initialization phase is restarted since the estimation of the available bandwidth requires a rifle of four packets.

C. Estimating the network congestion state

Before sending each packet, the source computes the new sending rate (see the Algorithm 1 below). It first estimates the network congestion state by using the FTT variation stored in ARRAYΔFTT D milliseconds ago (where, D is a fixed constant, see above). This value, denoted by $\Delta\mathrm{FTT}_{\mathtt{D}}$, is compared to three thresholds:

(1) The first one, denoted $\mathtt{TH}_{\mathrm{str}}$, corresponds to a large going beyond of the reference value FTT. When $\Delta\mathrm{FTT}_{\mathtt{D}}$ is larger than $\mathtt{TH}_{\mathrm{str}}$, the sending rate should strongly decrease to avoid network saturation (*strong congestion case*).

(2) The second, denoted $\mathtt{TH}_{\mathrm{mod}}$, corresponds to a congestion beginning. When $\Delta\mathrm{FTT}_{\mathtt{D}}$ is between $\mathtt{TH}_{\mathrm{mod}}$ and $\mathtt{TH}_{\mathrm{str}}$, the sending rate should decrease to prevent congestion formation (*moderate congestion case*).

(3) The third one, denoted $\mathtt{TH}_{\mathrm{rel}}$, represents a very small going beyond of the reference value FTT. If $\Delta\mathrm{FTT}_{\mathtt{D}}$ is smaller than $\mathtt{TH}_{\mathrm{rel}}$, the sending rate should increase. Indeed, the sending

rate should always be greater than the available bandwidth to ensure that the queue of the bottleneck is never empty. This is necessary to detect any bandwidth release (*release case*).

The sending rate is unchanged if $\Delta\text{FTT}_{\text{D}}$ is between TH_{rel} and TH_{mod}. This is the range of acceptable sending rate, regarding the network congestion state.

It is important to note that TCP Vegas [1] uses also some fixed thresholds to adjust its congestion window size. Moreover, it also maintains the expected sending rate slightly too great, in order to discover any bandwidth release. Note however that Primo does not rely on a congestion window, as Vegas does.

D. Adjusting the sending rate

The value of the reception rate is multiplied by a factor that controls the rate increase or decrease, depending on the network congestion state: F_{str} (strong congestion), F_{mod} (congestion beginning) and F_{rel} (bandwidth release), with $0 < \text{F}_{\text{str}} < \text{F}_{\text{mod}} < 1 < \text{F}_{\text{rel}}$. To obtain the correction, this value is then subtracted from the value of the sending rate one RTT ago (old_rate). This correction of the old_rate is weighted by a reactive parameter W_1. Note that the sending rate variation is more important for large values of W_1. Finally, a weighted average value is computed between the current sending rate value (rate) and the new computed value (new_rate), using a second weight W_2. This leads to a more or less smoothness of the sending rate (see Algorithm 1).

Note that such weighted averages are very common; they are, for instance, used to guarantee the smoothness of the RTT in TCP [17].

As in TCP, in the case of network saturation, the rate is halved and the TO period is doubled if no acknowledgment is received during a period of TO seconds (heavy congestion state).

E. Parameters

Algorithm 1 summarizes the computation at date t of the new sending rate, based on the current sending rate value, and the last reception rate sent by the destination node (see [21] for the ns-2 code).

Algorithm 1: **Primo(t, old_rate, rate, rcv_rate)**

new_rate = rate
$\Delta\text{FTT}_{\text{D}}$ = ARRAYΔFTT[t-D]
if $\Delta\text{FTT} \geq \text{TH}_{\text{str}}$ **then**
  ▷ *strong congestion case*
  new_rate = old_rate $- \text{W}_1 \times$ (old_rate $- (\text{F}_{\text{str}} \times$ rcv_rate))
else if $\Delta\text{FTT}_{\text{D}} \geq \text{TH}_{\text{mod}}$
  ▷ *moderate congestion case*
  new_rate = old_rate $- \text{W}_1 \times$ (old_rate $- (\text{F}_{\text{mod}} \times$ rcv_rate))
else if $\Delta\text{FTT}_{\text{D}} < \text{TH}_{\text{rel}}$
  ▷ *release bandwidth case*
  new_rate = old_rate $- \text{W}_1 \times$ (old_rate $- (\text{F}_{\text{rel}} \times$ rcv_rate))
end if
new_rate $= \text{W}_2 \times$ rate $+ (1 - \text{W}_2) \times$ new_rate

The definition of the new sending rate as a function of the current sending rate value, and of the last reception rate is strongly related to the definition of proportional, and integral laws in control design (see, for instance, the discrete design, analysis tools, and feedback properties in [8]).

In the sequel, we considered the following set of parameters: $\tau = 1$ (the corresponding discrete delay value), $\text{D} = \text{RTT}_{\text{ref}} \times \tau$, $\text{TH}_{\text{str}} = 2 \times \text{TH}_{\text{mod}} = 0.34 \times \text{FTT}_{\text{ref}}$, $\text{TH}_{\text{mod}} = 0.17 \times \text{FTT}_{\text{ref}}$, $\text{TH}_{\text{rel}} = 0.15 \times \text{FTT}_{\text{ref}}$, $\text{F}_{\text{str}} = 0.95$, $\text{F}_{\text{mod}} = 0.99$, $\text{F}_{\text{rel}} = 1.05$, $\text{W}_1 = 0.5$ (gain), and $\text{W}_2 = 0.9$. In order to determine these values, a study has been done under matlab (see for instance some previous work on the equivalence between continuous and discrete modeling of packets switched networks [3]).

As discussed in conclusion, the next step is to design an adaptive version of this algorithm, where the parameters are adjusted on the fly depending on network conditions. However, it is first necessary to validate the algorithm. Section IV shows that, while the parameters of Primo are fixed, it achieves good performances in various situations, encompassing very large network parameters ranges.

IV. Simulations methodology

In this section, we present the simulation conditions.

A. Topology

Two topologies are commonly used in order to evaluate the congestion control

schemes. The topology in Figure 1-top is composed of two sources (S_1 and S_2), two routers (R_1 and R_2) and two destinations (D_1 and D_2). Such a topology is used to study the network's parameters influence (see eg. [5], [12]). For the results presented in this paper (Section V), the routers queues use a Drop Tail policy and admits at most 30 packets. All links are symmetric: delays and bandwidths are identical in the two way. The two sources use the same protocol.

The topology in Figure 1-bottom allows to study the protocol's behavior in a network with multiple congestions (see eg. [6], [5]). Three families of flows are in competition here: flows a and b pass through two routers and could experience one congestion; flows c and d could experience two congestions, and, finally, flow e three congestions, respectively. In Section VI, we study the capacity of different protocols to stabilize and smooth the sending rates of the five sources.

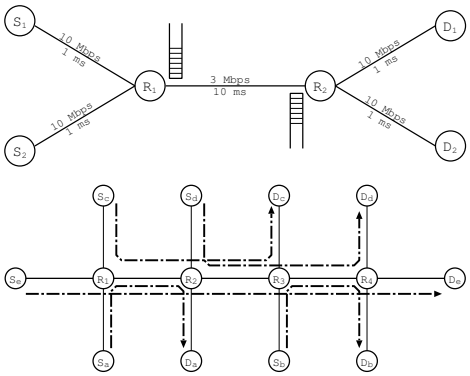

Fig. 1. Topologies for simulations.

B. Parameters' variations

In order to limit the combinatorial number of simulations needed to test all the combinations of parameters, we focus on the case when a network parameter is varying while the others are fixed to their default value (corresponding to a reasonable case). Such a study is not restrictive, and our intention is more to see the way the parameters may affect the behavior of the network. In this paper (Section V), we report the influence of four parameters (see [2] for more analysis):

- Bandwidth. The ratio between the input and output of the first router varies from 5% to 80%: the shared link bandwidth varies between 1 Mbps to 16 Mbps while the others remain equal to 10 Mbps.
- Queue size. The queue size of the routers varies from 5 to 50 packets.
- Homogeneous propagation delays. The propagation delay of the shared link varies between 5 ms to 200 ms.
- Heterogeneous propagation delays. The propagation delay of the second connection access link varies between 1 ms to 50 ms, leading to some RTT value without congestion varying from 24 to 122 ms for the second source, and fixed to 24 ms for the first source.

C. Efficiency criteria

The protocol performances are compared in Section V on the basis of the following five criteria, expressed in percentages. They concern both users' and operators' point of view. A protocol with good performances obtains high percentages.

- Sending rate stability. The variations of the sending rate should be as smooth as possible to avoid network instabilities. This criterion is evaluated using the standard variation of the rate; it is then normalized by division by the mean rate.
- Reception rate stability. The variations of the reception rate should be as smooth as possible to ensure the quality of multimedia receptions. Same evaluation as above.
- "Goodput" rate. This represents the rate of data correctly transmitted (*i.e.*, without any lost and then without retransmission). This criterion is equal to one minus the packets loss rate, multiplied by 1000. It still allows to assimilate high values to better cases.
- Occupancy rate. It represents the utilization of the network capacities. Indeed, all the previous criteria could be high, while the network is not fully used, leading to a bad throughput for the flow.
- Fairness. This criterion shows the capacity of a protocol to share equitably

a link between several connections. Regarding the topology of Figure 1-top, all the connections should have the same fraction of bandwidth, even if they have different propagation delay. This criterion is evaluated by comparison of the lower mean fraction of allocated bandwidth to 50% (ideal case).

D. Presentation of the results

Our comparison methodology allows to study a protocol regarding complementary criteria, depending on several parameters. However, it produces many values, and a protocol could be efficient regarding a given criterion and less regarding another one. Then, for simplifying results reading and comparisons, the values of the five criteria of a given protocol are aggregated on some *pentagon diagram* (Figures 2 to 5). Based on the criteria above, it follows that the protocol admits good properties if the pentagon diagram of a protocol is large. This helps the comparison along several criteria. Note that the diagrams display only average values. So, when necessary, the comment emphasizes on some specific values.

V. Results related to network's parameters influence

In this section, we analyze the results for the four sets of simulations described in Section IV-Parameters variations, and devoted to the network's parameters influence study. These simulations have been performed on the topology of Fig. 1-top.

A. Bandwidth influence

Simulation's results are presented in Figure 2. In these simulations, the five efficiency criteria have been measured while the bandwidth was varying on the shared link (R_1, R_2).

All the five protocols obtain relatively close results for the fairness, occupancy rate and goodput criteria. However, TCP Reno and Sack have lower constancy rates than the other protocols. This is due to the lack of sending burst avoidance mechanism: congestion window is sent in a single burst, and the source then waits for acknowledgments, leading to alternative sending and idle phases.

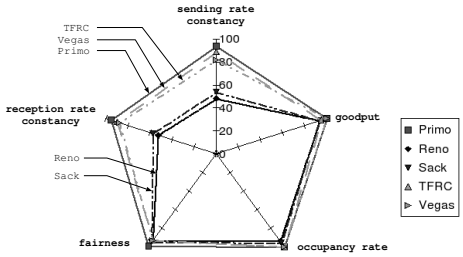

Fig. 2. Performances of TCP Reno, Sack, Vegas, TFRC and Primo with a bandwidth variation.

The sending and reception rates constancy of TFRC is lower than TCP Vegas and Primo. This is due to the fact that TFRC is not preventive. We observed that the queue on router R_1 is generally 75% full with TFRC. To the contrary, TCP Vegas and Primo react more rapidly when a congestion appear: the sending rate decreases before the queue becomes full (the queue is about 10% full with Primo, and 15% with Vegas).

Note that Primo obtains some slightly better results than Vegas, especially for rates constancy. Indeed, when the bandwidth ratio is low (between 10% and 30%), TCP Vegas has some difficulties to estimate the available bandwidth. These difficulties lead to some variations in the sending rate, and the constancy rates criteria decrease.

B. Queue size influence

Simulations results are presented on the pentagonal diagram of Figure 3.

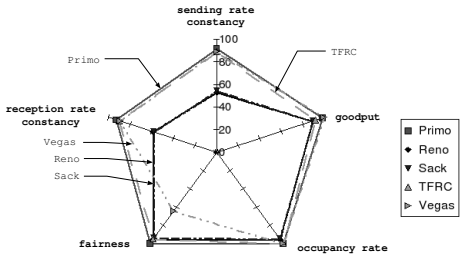

Fig. 3. Comparison of TCP Reno, Sack, Vegas, TFRC and Primo with a queue size variation.

As in previous simulations, TCP Reno and Sack have very close results, and some poor rates stability. For the simulations

of this test, the fairness criterion of TCP Vegas is at most equal to 75%. Note that when the queue is small (5 packets for two flows), TCP Reno and Sack have better results for the fairness (around 90%) than Primo (around 75%) and Vegas which obtains only 2%. But when the queue size is larger than 10 packets (a more general case), Primo offers excellent results for the fairness (near 100%) and for the rates stability (around 93%). TCP Vegas needs to maintain a sending rate slightly larger than the bandwidth, leading to some packets in the queue. If the queue size cannot allow to store enough packets per flow, TCP Vegas performances are poor. Finally, Primo and TFRC obtain both good results for this test.

Note that similar results have been obtained with a RED queue on router R_1 when it can contain at least 10 packets.

C. Propagation delay with homogeneous RTT influence

Figure 4 shows results for the set of simulations with variation of the propagation delay.

As previously, TCP Reno and Sack cannot stabilize the sending and reception rates. But we observed also that TCP Vegas has some difficulties in this set of simulations. When the RTT is larger than 100 ms, the sending rate stability of TCP Reno, Sack and Vegas falls under 20%. With a RTT equals to 200 ms, the sending rate stability reaches only 2% for TCP Sack, and 5% for TCP Vegas. It seems that the sending burst avoidance of TCP Vegas is less efficient when the RTT is large.

Both TFRC and Primo do not rely on a congestion window. Hence, they are less sensitive to long propagation delays, and they obtain good sending and reception rate constancy (these efficiency criteria are around 86% for TFRC and 93% for Primo).

D. Propagation delay with heterogeneous RTT influence

In this set of simulations, the propagation delay of the link (S_2, R_1) in the Figure 1-top is varying, leading to heterogeneous RTT between the flows of sources S_1 and S_2. Results are given in Figure 5.

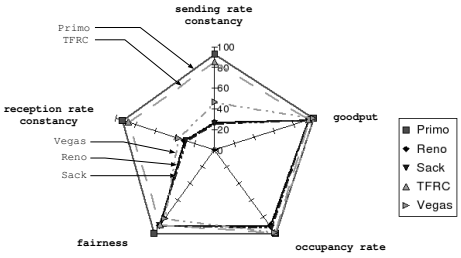

Fig. 4. Comparison of Primo, TCP Reno, Sack and Vegas with homogeneous RTT variation.

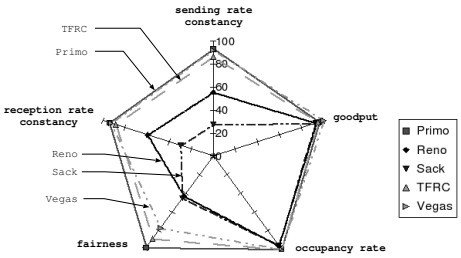

Fig. 5. Comparison of Primo, TCP Reno, Sack and Vegas with heterogeneous RTT variation.

The main difference with the previous set of simulations concerns the fairness. As already known [11], TCP Sack and Reno have fairness problems in presence of heterogeneous RTT. When the difference between the two RTT is larger than 60 ms, TCP Reno fairness is lower than 27%, while TCP Sack fairness is varying between 42% and 0% (meaning that the second connection cannot send any data). Note also that their sending and reception rate stability is also low (lower than 30% for Reno and lower than 55% for Sack).

As previously, TCP Reno and Sack obtain poor results concerning rate constancy. For these efficiency criteria, TCP Vegas has slightly better results than Primo, which is slightly better than TFRC.

TCP Vegas has slightly better results than the other protocols for the goodput criterion, but its fairness criterion is about 77%. Fairness is about 89% for TFRC and 98% for Primo. These protocols, that do not rely on a congestion window, obtain good results in presence of heterogeneous propagation delays.

E. Complementary remarks

One can note that the occupancy and goodput rates are generally high for the

four protocols, and the four set of simulations. However, TCP Vegas and Primo have generally better results for these two criteria. Indeed, the preventive protocols avoid the routers queues saturation, and thus the packets loss.

Concerning the occupancy rate, TCP Reno and TCP Sack decrease brutally their sending rate when they detect a congestion. By reducing more slowly their sending rate, Primo and Vegas better use the network resources, leading to a better occupancy.

When the number of sources increases in Figure 1-top, we observed that the performances of Primo are not significantly affected. However, the fairness of TCP Vegas is at most 75%. The other protocols, which have a corrective behavior, are affected by the increase of the number of sources (notable decreasing of sending and reception rates constancy).

VI. Results related to sending rates stability in a multi-congestion network

In this section, we present and analyze the results obtained on the second topology (Fig. 1-bottom), and devoted to the multi-congestion influence study. Note that, while each router-to-router link is shared by three flows, one could refer to several kind of fairness in order to evaluate the bandwidth sharing by the different transport protocols. Here, we only focus on the sending rate stability. For multimedia flow transportation as well as stability of the whole network, it is important that the sources are able to stabilize and smooth their sending rate.

Several scenarios have been simulated; we report two of them in Figures 6 and 7. In Fig. 6, all the flows `a`, `b`, `c`, `d` and `e` start at the same time. In Fig. 7, first the flow `a` and `b` (that encounter a single congestion) start, then the flows `c` and `d` (that encounter two congestions) start, and finally the flow `e` (that encounter three congestions) starts. While it is sometimes impossible to distinguish all the flows in the figures, these results allow to give the following conclusions.

First, TCP Reno never stabilize nor smooth the sending rates of the sources. Note that TCP Sack has the same behavior. The sending rates can fall down to 0 and then abruptly increase. As already known, these protocols are not well suited for multimedia flows.

Second, TFRC is able to smooth the sending rates, due to its inter-packet delay mechanism. However, they are not perfectly stabilized. Indeed, one can observe smooth variations. Note that, since the sending rate varies smoothly, TFRC can be used for multimedia flows transportation.

Third, TCP Vegas and Primo both stabilize and smooth the sending rate of the sources. In some cases, TCP Vegas obtains slightly better results than Primo. Note that, in most of our scenarios, Primo is always very close to a bandwidth sharing of about one third of the bandwidth per flow.

VII. Conclusions and future work

In this section, we summarize and discuss our contribution, before outline future work.

A. Contribution analysis

Transport protocols define their congestion control on empirical rules, and most of them use packet loss as information of the network state. This method induces a corrective behavior and so the creation of congestions instead of avoiding them.

Based on an analysis of the literature [4], we studied a preventive congestion control scheme that combine both the forward trip time (FTT) and the reception rate (Section II). The variations of the FTT are used to estimate the degree of congestion, and the reception rate is used to precisely adjust the sending rate in the case of congestion. We used an appropriate proportional integral modified control mechanism in the aim of reaching some pre-defined objectives: rapidly reach the optimal rate, smoothly adapt the sending rate, rapidly, moderately and preventively react to congestion formation. As other transport protocols, the simple congestion control scheme we obtained (Section III) relies on fixed thresholds and parameters, that we calibrated using a fluid modeling under matlab (see [3]).

We implemented Primo under network simulator [13] and developed a complete evaluation methodology that encompass all

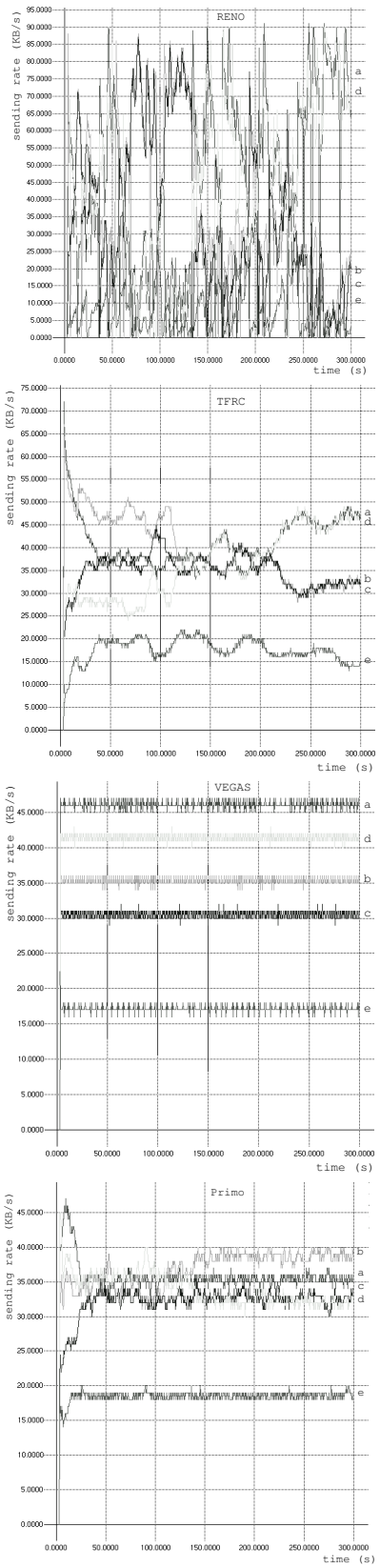

Fig. 6. Sending rates stability in a multi-congestion environment (see Fig. 1-bottom), when all the flows start simultaneously. From top to bottom: Reno, TFRC, Vegas, Primo.

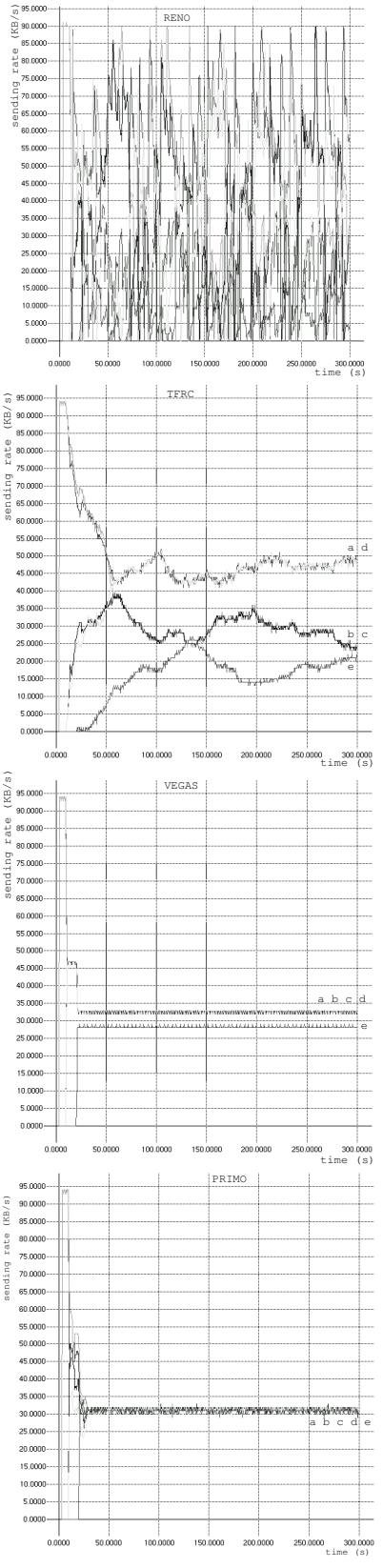

Fig. 7. Sending rates stability in a multi-congestion environment (see Fig. 1-bottom) when flows a and b start first, then the flows c and d and finally the flow e. From top to bottom: Reno, TFRC, Vegas, Primo.

the main parameters [2]. Many measures have been done while several network's parameters were varying (note that combined influence of several parameters has not been extensively simulated since this requires a combinatorial number of simulations, and our aim was mainly to see the way in which stand-alone parameters could impact the performances). In this paper, we reported a summary of results (Section IV) concerning the influence of the available bandwidth, the queue size and the propagation delays on the sending rate stability, the reception rate stability, the fairness, the occupancy rate of the shared link and the good transmission rate. We also reported results concerning the stability of the sending rate in a multi-congestions environment. More complete results can be found in [2].

Results presented in Sections V and VI indicate that the *fairness objective* is completely respected. More, the *rapidity* is an inherent characteristic of the protocol, and the *initialization phase* has a duration of about one RTT (very less than other transport protocols). The *stability of the sending and reception rates* is very good, and improve Vegas results. The full use of the *network capacities*, and the *preventive behavior* are equally respected.

Obviously, Primo has better results than TCP Reno and Sack. But, it also obtains better results than TFRC, a TCP-friendly alternative to TCP. This can be explained by the fact that Primo is preventive while TFRC is corrective. As a drawback, Primo is not TCP-friendly; we discuss this issue in Section VII-B. We also observed that, while Primo obtains very good results in multi-congestions environments, it seems that TCP Vegas gives slightly more constant sending rate than Primo. However, in other tests, Primo has always better results than Vegas, especially in presence of long or heterogeneous propagation delays. This is mainly due to the fact that Primo does not rely on a congestion window but on an inter packets delay mechanism. As a conclusion, Primo results are very encouraging since they are better than those of TFRC and TCP Vegas.

B. Future work

Our main contribution has been to show that a simple preventive congestion control scheme, based on a proportional integral modified controller and feed by pertinent inputs from the destination, can be very efficient for congestion control. However, Primo is not adaptive, meaning that it uses fixed parameters, and is not TCP-friendly, meaning that it could not be used simultaneously with TCP (as in the Internet). We now discuss these issues.

Our study puts some light on control-feedback techniques (proportional-integral controllers) in connection with the available information in the network. In our opinion, this represents a first step in defining a *dynamical* (preventive) congestion control mechanism with respect to the (congestion) status of the network. More explicitly, we think that the definition of some *adaptive* version of Primo would represent a better solution to the congestion problem under consideration. As suggested by Tsypkin in [20], by *adaptation*, we understand the process of modifying the parameters (and eventually the structure) of the system, and the control actions, such that the current information is used to obtain a *definite* (usually *optimal* or *sub-optimal*) state of the system when the operating conditions are uncertain (or not completely known), and time-varying. Without entering in details, we think that one of the possibilities to be considered is to adapt the new rate on-line in the sense that the sending rate would rely on a network congestion state parameter explicitly defined as a function of $\Delta\text{FTT}_{\text{D}}$. The optimality criterion should be defined function of the efficiency criterion defined in the Section IV. Note that the analysis presented in this paper was more than necessary in order to see the potential improvement that we may obtain via such a construction.

To our opinion, progress in congestion control would mainly concern preventive protocols. But this implies that they could not be used in the Internet, or more generally when some flows are in competition with corrective protocols in shared router's queues. A first approach is to modify online the preventive congestion control scheme

(as explained above) in such a way that it becomes more aggressive when the presence of a corrective protocol is suspected [10]. This may be combined with some quality of services policies to avoid as far as possible the presence of incompatible flows in the same queue. Another approach is to use the preventive control schemes in dedicated networks, where the kind of transport protocols used can be managed. This is the case of some spontaneous ad hoc networks based on embedded communicating devices: laptops, mobile phones, personal assistants, communicating vehicles[1]... In such potentially highly dynamic networks, TCP-like protocols experience strong difficulties. Embedded systems could exclusively use preventive schemes for the applications inside the local ad hoc network. This may represent an important domain for preventive protocols.

ACKNOWLEDGEMENTS

The authors wish thank the anonymous reviewers for their comments, and suggestions to improve the overall quality of the paper. Special thanks are addressed to Farid Sayah, M.Sc. student at Université de Technologie de Compiègne, France, who worked on Primo during the last year, and prepared a distributable ns-2 transport agent [21].

REFERENCES

[1] L.S. Brakmo, S.W. O'Malley, and L.L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. In *ACM SIGCOMM*, pages 24–35, 1994.

[2] F. Chatté. Contribution au contrôle de congestion dans les protocoles de transports (in French). PhD Thesis. Université de Technologie de Compiègne, France, July 2004 (available on demand).

[3] F. Chatté, B. Ducourthial, D. Nace, and S.-I. Niculescu. Fluid modeling of packet switching networks: perspectives for congestion control. *International Journal of Systems Science*, Vol. 34, No. 10-11, pp. 585–597, 2003.

[4] F. Chatté and B. Ducourthial. Contrôle de congestion dans les protocoles de transport internet : état de l'art et perspectives (in French). *Technique et Science Informatique*, Hermès, Paris, France, volume 23, number 8, pages 1057–1084, 2004.

[1]Our current researches deal with such ad hoc networks, where some applications require multimedia flows (eg. images transfer in a convoy of vehicles).

[5] A. De Vendictis, L. Bonnacci, and A. Baiocchi. TCP New Vegas: providing good TCP performance in both homogeneous and heterogeneous environments. In *Proc. of 15th ITC Specialist Seminar*, Wuerzburg (Germany), July 2002.

[6] S. Floyd. Connections with multiple congested gateways in packet-switched networks. ACM SIGCOMM Computer Communication Review, volume 21, number 5, pages 30-47, October 1991.

[7] S. Floyd and A. Romanow. TCP selective acknowledgement option. RFC 2018, October 1996.

[8] G.F. Franklin, J.D. Powell, A. Emami-Naeini. *Feedback control of dynamic systems* Addison-Wesley: Reading, USA, 1991.

[9] M. Handley, S. Floyd, J.D. Padhye, and Widmer J. TCP friendly rate control (TFRC): Protocol specification. RFC 3448, January 2003.

[10] U. Hengartner, J. Bolliger, and T. Gross. TCP Vegas revisited. In *Proc. of the IEEE INFOCOM'2000*, volume 3, pages 1546–1555, 2000.

[11] F.P. Kelly. Mathematical modelling of the Internet. In *Proc. of 4th Int. Congress on Industrial and Applied Mathematics*, Edinburgh, July 1999.

[12] J. Mo, R.J. La, V. Anantharam and J. Walrand. Analysis and comparison of TCP Reno and Vegas. In *Proc. of IEEE INFOCOM'99*, Mars 1999.

[13] The network simulator, `http://www.isi.edu/nsnam`

[14] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ECN) to IP. Request For Comments 2481, January 1999.

[15] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. Request For Comments 3168, Sept. 2001.

[16] R. Rejaie, J. Handely, and D. Estrin. RAP : An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *Proc. of IEEE INFOCOM'99, NY, USA*, March 1999.

[17] R. Stevens. TCP/IP Illustrated. Addison Wesley, 1994.

[18] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda. Achieving moderate fairness for UDP flows by path-status classification. In *Proc. of the IEEE LNC'2000, Tampa, FL*, November 2000.

[19] Y. Tobe, Y. Tamura, and H. Tokuda. Detection of change in one-way delay for analyzing the path status. First Passive and Active Measurement Workshop, April 2000.

[20] Tsypkin, Ya. Z.: *Adaptation and learning in automatic systems* (Academic Press: New York, 1971).

[21] `http://www.hds.utc.fr/~ducourth/dwl/primo`