Coherent Charging of Differentiated Services in the Internet Depending on Congestion Control Aggressiveness

Philippe Owezarski[*], Nicolas Larrieu

LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse cedex 4
France
e-mail : {owe, nlarrieu}@laas.fr

Abstract. This paper deals with both enforcing and charging end-to-end Quality of Service (QoS) and differentiated services in the Internet. Today, much work for optimizing QoS and defining charging mechanisms accordingly is related to low network layer (up to IP), and the current proposals under the spotlights are VPN, CDN, (over)-provisioning, and of course DiffServ. However, the service an user can get can be quite different from the one provided by the network layer (IP in the Internet). Transport protocols (TCP most of the time) induce oscillations (often seen as self-similarity) in the traffic, in particular because of their congestion control mechanisms. Based on this result, it is obvious that it is impossible to coherently charge such kinds of services at layer 3. QoS has also to be managed and enforced by transport protocols, that also have to strongly impact the way pricing is done. As a consequence, a new approach for services differentiation and charging (in addition to existing layer 3 approaches) at transport level is proposed. This approach relies on the aggressiveness of congestion control mechanisms because the more aggressive a protocol, the better the QoS it provides in case of a well provisioned network. And of course charging has to evolve accordingly. This idea is demonstrated by taking examples as UDP and several versions of TCP. In particular it is proved that service differentiation can be made that way in the Internet. This paper also gives a quantitative study of the QoS got by users depending on the amount of resources consumed, and it is observed that this function is linear. The charging of each service can then be quantified that way.

Keywords. Charging Internet services, Internet QoS, Services Differentiation, Congestion Control, TCP

1 Introduction

Common issues of the Internet currently deals with QoS and charging such QoS. Even if the term QoS can have several meanings depending on the point of view, it exists two main points of view:

1. Applications or users point of view: each application has specific requirements in terms of throughput, delay, jitter, reliability, etc, and each application should take

[*] Corresponding author

advantage of a dedicated communication service. And of course, users want to pay only for the service they need, and not more.

2. Carriers point of view whose goals are to optimize the use of communication resources (and then maximize their benefits), to limit loss and delay, and to charge users in a coherent way.

It is then very difficult to make users and carriers requirements converge[1]. As well it is difficult to find pricing mechanisms that fit both points of view: carriers want to charge users for all consumed resources (or allocated ones in some cases) while users want to pay only for usable ones. In particular, they do not want to pay for retransmissions due for example to some congestion points in the network, as it is for the smart market model. All solutions for QoS and related charging mechanisms mainly deal with the definition of a limited set of more or less guaranteed service classes as ATM, IntServ [1] that proved to be unsuited for large scale networks, and now DiffServ [2] and/or MPLS, etc. DiffServ, for instance, aims to enforce several classes of services that can be handled and enforced on every sub-network or Autonomous System (AS) of the Internet (including peering links). In fact, it consists in provisioning, in theory everywhere in the network, some amount of resources that will be assigned, in priority, to best traffic classes, to make them get a better priority than other lower classes. However, it seems that DiffServ lacks on some point:

1. How to make every class of service to be defined the same way on every AS, peering link, etc?
2. How to avoid higher classes to disturb the quality of lower ones?
3. Are the defined classes (18 for DiffServ) enough according to the huge number of application classes?

It is clear here that it is quite impossible to charge such kinds of services as users that are interested by the QoS they get from end to end will get only the worst QoS provided by the worst AS, and they will have to pay for higher services uselessly on other AS providing better services.

The other approaches for providing QoS aim to optimize the current best effort service to make it a “premium” or “gold” service that can handle all kinds of applications classes (then solving points 2. and 3.), from the real-time to the ones that requires only best effort. For these later, the QoS they will get will be improved a lot compared to current best effort service they use. But it is impossible to find charging mechanisms fitting the wish of users to pay only for the service they need. Here they will always pay the price of the best service. Relying on this idea, the CDN (Content Delivery Network) approach solves the first point by removing peering links and considering a single limited AS on which traffic can be precisely controlled. The general over-provisioning approach aims to improve the CDN solution for the whole Internet by over-provisioning all AS and all access and peering links, and optimize the way flows and packets are computed. The over-provisioning approach seems to be the most promising for carriers. The process to make the Internet evolve that way has been started by carriers, but nobody knows if this process will be completed one day, in particular because it is costly, and users do not seem to be ready to pay for more than they need?

In addition, ensuring QoS at the network is not enough. As well charging mechanisms at this level does not respect actual users needs. Section 2 shows that the QoS provided by IP can (will?) be disturb by congestion control mechanisms of TCP that infer self-similarity properties, and then a large variability of the QoS users will get. Charging mechanisms at level 3 are then not coherent because of the transport protocols behaviors just above and between users and network layer. As a consequence, QoS enforcement and charging

[1] It is the case because there is almost no real and accurate reservation mechanism in the current Internet.

mechanisms have to be extended and considered up to layer 4 (at least). Section 2 also proposes a new approach for end-to-end service differentiation that takes into account the amount of resources consumed by each flow. Assignment of resources between concurrent flows in the Internet is performed according to congestion control mechanisms of transport layer [3]. This new approach for service differentiation then relies on congestion control mechanisms that are more or less aggressive and provides a related QoS. "Aggressiveness" is defined in this paper as the ability for a protocol to consume resources. As well, as the choice of the network solution for enforcing QoS at IP level is not defined yet, the approach for service differentiation and differentiated charging in this paper is independent from network strategies and architecture. Section 3 then shows some experiments showing, on some examples, that service differentiation at level 4 according to congestion control classes is feasible and very coherent. Then, section 4 proposes a quantitative study of services provided by several transport protocols. It evaluates the QoS they provide, their aggressiveness, related to the amount of resources they consume, and demonstrates how charging has to be made to be coherent for both users and carriers. In particular, section 4 gives the charging factors to apply between different classes of services, i.e. different classes of congestion control mechanisms. Finally, section 5 concludes the paper.

2 A Congestion Control Based Approach

It has been said before that it is obvious that transport layer (most of the time TCP in the Internet) modifies services got by applications and their users. Monitoring studies showed that TCP infers complex traffic models – very far from the classical Poisson model for flows or packets arrivals [4]. For instance, [5, 6, 7, 8, 9] showed that internet traffic has self-similarity characteristics, in particular because of TCP congestion control mechanisms. As a result, traffic contains multi-scales oscillations inferring a huge variability that dominates the mean traffic[2] [10], and then a final service, seen by applications and users, changing a lot, and obviously not guaranteed. QoS provided by network at IP level is then completely disturbed by TCP mechanisms.

A transport protocol remains mandatory to recover from IP losses, even if their number is very low. As well, as transport is also in charge of enforcing end-to-end QoS, acting at this level for guaranteeing QoS is coherent according to TCP/IP architecture. In addition, it is obvious that some kinds of services differentiation has to be made at transport level (at least), and more precisely at the level of congestion control mechanisms, as they are responsible of the variability of QoS provided by TCP to applications and users.

Our solution for guaranteeing QoS is then really different from the ones that have been already proposed as IntServ or DiffServ because it seems quite difficult (maybe impossible?) to define, provision and reserve resources for global classes of services that are supposed to provide the same service on all the domains of the global Internet. Our approach is strongly inspired by recent results on traffic modeling (based on traffic monitoring) that show that because of the oscillations of the traffic it is quite impossible to guarantee a stationary service: the impact of the oscillations of one flow[3] provokes strong oscillations on the other flows transported in parallel on the same link, path, or using the same amount of resources allocated for a dedicated class of service. Such oscillations are responsible of a strong decrease of the average performance level of the network, because of the periods when the network is almost

[2] This can make network provisioning very difficult.

[3] A flow here is defined as a mono-directional TCP connection. A TCP connection then contains 2 flows, one in each direction.

not loaded, and the ones when the network is overloaded, provoking then some losses. Given these results, our approach proposes first to stabilize the amount of traffic of each flow, to make it as regular as possible. Therefore, if each flow is stationary, then the whole traffic built with the aggregation of all flows (that are all stationary) will be stationary. Our approach then recommends to make traffic sources as regular as possible to be able to guarantee QoS in the network[4]. That is why some congestion control mechanisms as TFRC [11] have been designed to provide regular throughput services, in particular for multimedia streaming applications.

Congestion control mechanisms also define the process of communication resources allocation and consumption by flows. Besides, the differences between versions of TCP (Tahoe [12], Reno [13], New Reno [14], SACK [15]) only rely on the congestion control mechanisms, especially "fast retransmit" and "fast recovery" [3, 16]. The evolution process of TCP, to improve its mean performance, dealt with increasing its aggressiveness according to available resources (of course trying to be as fair as possible) [13]. Recall that "aggressiveness" is defined in this paper as the ability for a protocol to consume resources. The idea developed in this paper is then to propose transport service classes based on their congestion control mechanisms. Indeed, we can think that the more resources a protocol consumes, the better the QoS it provides. The following shows that this concept works and that different aggressiveness of congestion control mechanisms lead to differentiated QoS. In addition to some "anti-oscillations" congestion control mechanisms, guaranteeing relative services based on transport protocols aggressiveness should be enough for providing guaranteed Internet QoS.

As mentioned previously, this approach is completely opposed to some usual approaches as DiffServ. Our approach is mainly based on recent traffic monitoring studies and their conclusions on traffic characteristics, and then proposes some solutions to change Internet traffic nature to make it easy to provide guaranteed QoS. As well, it has been observed that QoS degradation in the Internet is mainly due to some under-provisioned links (peering links between competing carriers or ISP, or carriers involved in a commercial agreement, where peering links are provisioned according to average traffic and not its variability). Qualifying a link of being under-provisioned means that this link is not able to compute high intensity oscillations. Indeed, the oscillating nature of Internet traffic implies to strongly over-provision Internet links to avoid short scale congestions[5]. But this is a waste of resources. By regulating traffic it is then possible to avoid such kind of short scale congestions.

Based on these principles for enforcing guaranteed QoS in the Internet, pricing mechanisms we proposed are extensions of the "smart market" model, meaning that congesting user traffic will be charged more. Our proposal for charging relies on the principle of charging flows depending on the risk of congestion they create. Indeed, the more aggressive a transport protocol, the more oscillating the traffic, because an aggressive protocol is able to use a lot of resources very fast, then inducing high intensity oscillations. Because of the bad impact of strong oscillations on QoS, our charging mechanisms proposed to charge more more aggressive protocols. The remainder of this paper will show that such approach is coherent and that service differentiation can be made based on transport congestion control aggressiveness.

[4] Of course Internet flows will never be as regular as the telephone network or the ATM CBR service that manage transmissions bit per bit or byte per byte. Because of the packet oriented nature of the Internet, there will always be some bursts of bytes, but our approach claims that a quite regular service can be guaranteed if we are able to limit the amplitude of bursts.

[5] Of course, such kind of over-provisioning is enforced on a carrier or ISP network, but not necessarily on peering links

3 Experiments and Demonstration with Examples

This study has been made using many examples. To illustrate this work, this paper will show what we get, on one example based on 3 different transport protocols significantly chosen:

1. UDP that has no congestion control mechanism and then is very aggressive. It is chosen here as it is often used by real-time applications, and is almost the only alternative solution to TCP. That is why, in the early 80s with the arrival of multimedia capabilities, multimedia applications took advantage of UDP because TCP was not providing enough throughput and performances. UDP is then nowadays considered as the "premium" service in the Internet[6].
2. TCP Tahoe, an old version of TCP, supposed to have low aggressiveness. This version of TCP almost disappeared, but it has been chosen here to maximize differences between the selected protocols.
3. TCP New Reno that is currently the most used version of TCP (even if it should be TCP SACK pretty soon)

To prove that the service differentiation on congestion control aggressiveness is possible, it is shown that:

1. QoS evolves in the same way as the order of aggressiveness of protocols.
2. Flows of a same aggressiveness class fairly share resources.

3.1 Experimental platform

This study has been performed using the NS 2 Network Simulator. Based on the selected transport protocol for the example presented in this paper, some QoS classes have been arbitrarily (only based on our belief and without any previous quantitative study) assigned to each protocol. TCP Tahoe is considered as the best effort class because of its congestion control algorithm that should be the less aggressive among the 3 protocols. UDP is considered as the premium service, and TCP New Reno is an intermediate service, its congestion control algorithm being more aggressive than the one of TCP Tahoe, and less than UDP that does not perform any congestion control.

FCFS (First Come First Served), similar to FIFO (First In First Out), and Tail Drop policies for managing queues in routers have been selected because they are the most frequent strategies applied currently on routers.

The network topology used for simulations is depicted on Figure 1, that also gives links characteristics. This test topology consists of 3 senders UDP, TCP New Reno, and TCP Tahoe that can send one or more flows each, an intermediate node for concentration and a receiver. The intermediate node represents a router of the network. The link between the router and the receiver is considered as a backbone network. By modifying the characteristics of this link, it is possible to change the load of the link and so demonstrate the aggressiveness effects on QoS. The simulation have been made in 3 cases:

1. Unloaded network (Capacity = 5 Mbit/s, Delay = 2 ms).
2. Loaded network (Capacity = 1 Mbit/s, Delay = 10 ms).

[6] However, UDP is not a solution for the Internet. UDP was initially designed for LAN as Ethernet, and the fact that it does not provide any congestion control mechanism is a huge danger for the liveness and integrity of the Internet. UDP, even if it was an easy and fast solution for multimedia applications up to now has to be replaced by a more suited transport protocol. TFRC newly designed is a good candidate.

3. Congested network (Capacity = 0.5 Mbps, Delay = 20 ms).

The other links characteristics are: Capacity = 5 Mbit/s, Delay = 2 ms. Applications using TCP New Reno and TCP Tahoe are FTP applications. The one using UDP is a Constant frame rate application sending packets with a constant throughput of 448 Kbit/s. All applications send 200 packets of 1000 bytes each (which is enough to reach TCP stationarity), and start at the same time.

This study has been made only on long flows – called elephants – because of the classification of flows depicted on Figure 2. This figure shows that short flows (mice) represent the huge majority of flows in the Internet, but only a small part of the whole traffic. At the opposite, there are only few elephants, but they generate the huge majority of traffic. In addition, because of the congestion control mechanisms of TCP ("slow start"), only elephants are able to generate high throughput traffic and are then responsible of the high amplitude oscillations, that are the ones that are traced in this paper because of their bad impact on QoS.

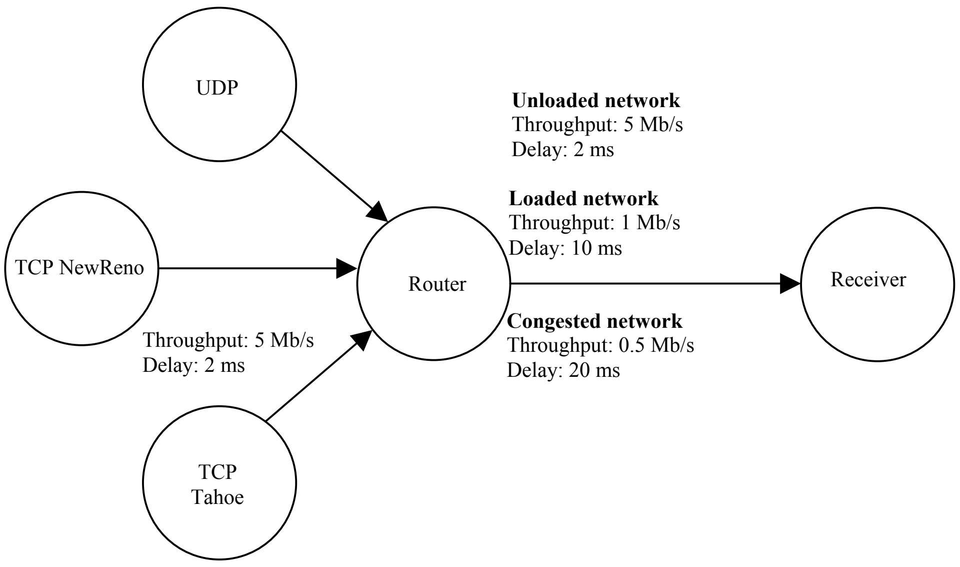

Figure 1. Simulation topology

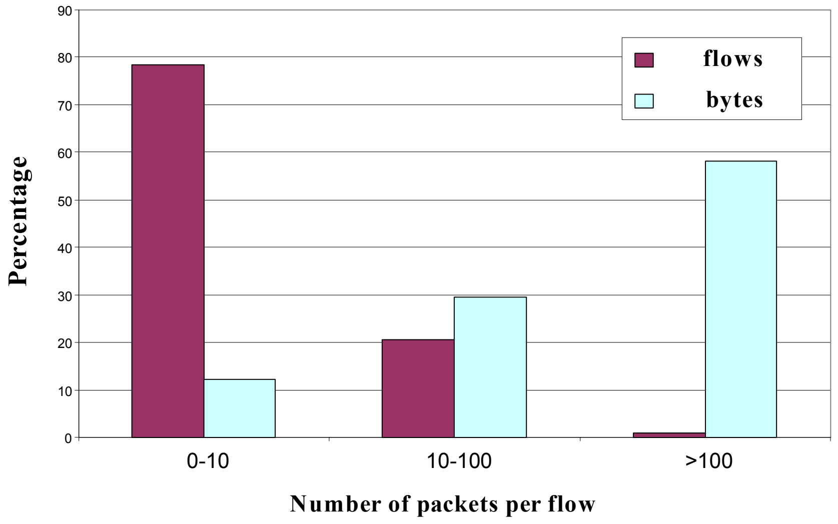

Figure 2. Mice vs. elephants traffic

Mice traffic can be modeled by a gaussian noise, thus having very low oscillations. Then, the impact of mice on QoS degradation and oscillations is limited. We will then consider only elephant traffic, and we will just make some adjustment at the end for integrating mice behavior.

Note also that all following experiments have been made several times, and we just compute the average on all these experiments.

3.2 Respect of differentiated classes

The parameters that have been analyzed to put forward the differences between protocols in terms of aggressiveness and QoS are:

1. *Sending time for all packets by the application*: This is the main parameter to quantify QoS. This parameter is inversely proportional to the goodput of the application. The shorter this time, the higher the goodput, and the higher the QoS got by users.
2. *Average waiting time in router queues*: This is an important parameter to quantify protocol aggressiveness. If a protocol is aggressive, it is going to send packets without taking care of the congestion state of links and router queues, and there is a risk for packets to remain queued for a long time if the network load is high. The higher this time, the more aggressive the protocol.
3. *Throughput consumed by the connection*: Compared to the goodput, throughput takes into account traffic related to loss recovery. It is then more representative of protocol aggressiveness than goodput, because losses are mainly created by queue overflows, that are generally due to a too high aggressiveness of protocols.
4. *Loss ratio*: it is also a significant parameter to quantify aggressiveness, for the same reasons as previously, because losses are most of the time the result of a too high aggressiveness.

Table 1 presents all the measured values:

		Unloaded network	**Loaded network**	**Congested network**
Sending time of all packets	UDP	3.55 s	3.88 s	4.24 s
	TCP New Reno	0.69 s	4.28 s	9.83 s
	TCP Tahoe	0.71 s	4.78 s	7.25 s
Average waiting time in FIFOs	UDP	8 ms	277 ms	644 ms
	TCP New Reno	43 ms	258 ms	224 ms
	TCP Tahoe	44 ms	198 ms	250 ms
Throughput	UDP	448 Kb/s	413 Kb/s	377 Kb/s
	TCP New Reno	2.3 Mb/s	374 Kb/s	163 Kb/s
	TCP Tahoe	2.25 Mb/s	335 Kb/s	221 Kb/s
Loss ratio	UDP	1 %	8.5 %	10 %
	TCP New Reno	1 %	2 %	12 %
	TCP Tahoe	0.5 %	6.5 %	10.5 %

Table 1. Simulation results

To better visualize results, figures 3 and 4 show the results of NS simulations in the 3 cases: unloaded, loaded and congested network. Note that the throughput does not appear not to overload the paper. It can be easily inferred from goodput and loss ratio.

The following presents some conclusion from result analysis.

3.2.1 Unloaded network (Figure 3)

In this case, and it is logical, loss ratio is very low. Besides, there is no significant difference between these 3 protocols. They are all, in these conditions, equivalent in term of efficiency, even if TCP New Reno is a bit better than TCP Tahoe – as predicted by our proposal. UDP, in this case, cannot be compared to the other protocols because of the fixed constant rate of its source that is much lower than network capacity[7]. Nevertheless, it can be noticed that the minimum transmission time of the UDP flow is enforced with an unloaded network; 200 packets sent with a 448 kbit/s rate are theoretically sent in 5.55 s, what is the case here.

Finally, it is very difficult to see any difference in protocol aggressiveness with these results[8] because when protocols require communication resources, these resources are available and allocated without delay. This behavior then perfectly respects the "smart market" principle.

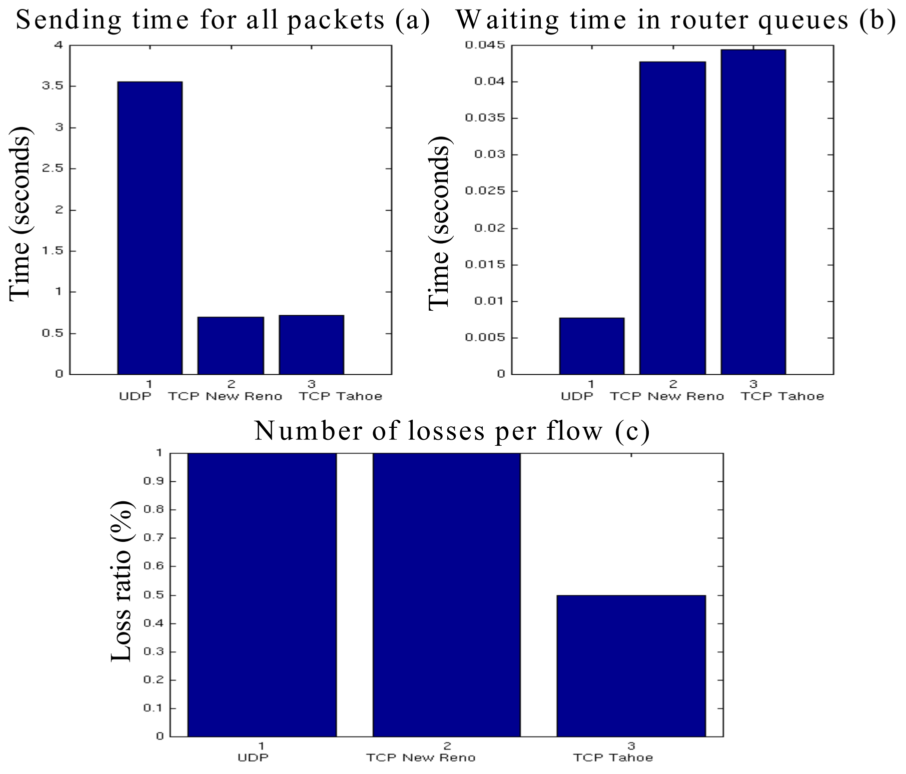

Figure 3. Service differentiation in unloaded networks

3.2.2 Loaded network (Figure 4)

If load increases on the network, loss ratio also increases. However Figure 4 shows a total respect of our proposal, and QoS and aggressiveness are actually evolving in the same way. Then UDP performances are better than TCP New Reno's (sending time is lower and goodput higher), that are themselves better than TCP Tahoe's – as predicted by our proposal. Note

[7] Note however that UDP loss ratio is not zero even if average UDP traffic is less than the link capacity. These losses are created in the router because of the impact of TCP bursts (oscillations) on the router scheduling, and UDP packets are not computed frequently enough, with the selected FIFO size, to avoid FIFO overflow.

[8] Even if deviations on Figure 2(b) seem graphically large, the measurement scale has to be considered. These values are so low that the differences between protocols are negligible. It only demonstrates that there are no contention between flows thanks to resource over-provisioning.

however that on Figure 4(c), TCP Tahoe's loss ratio is higher than it should be. The analysis showed that in case of loss, TCP Tahoe only detects loss after sender timeout and runs in congestion avoidance mode, while TCP New Reno starts earlier a "fast recovery" phase. So, TCP New Reno re-sends packet(s) earlier than TCP Tahoe and these packets are then put before TCP Tahoe's in router queues.

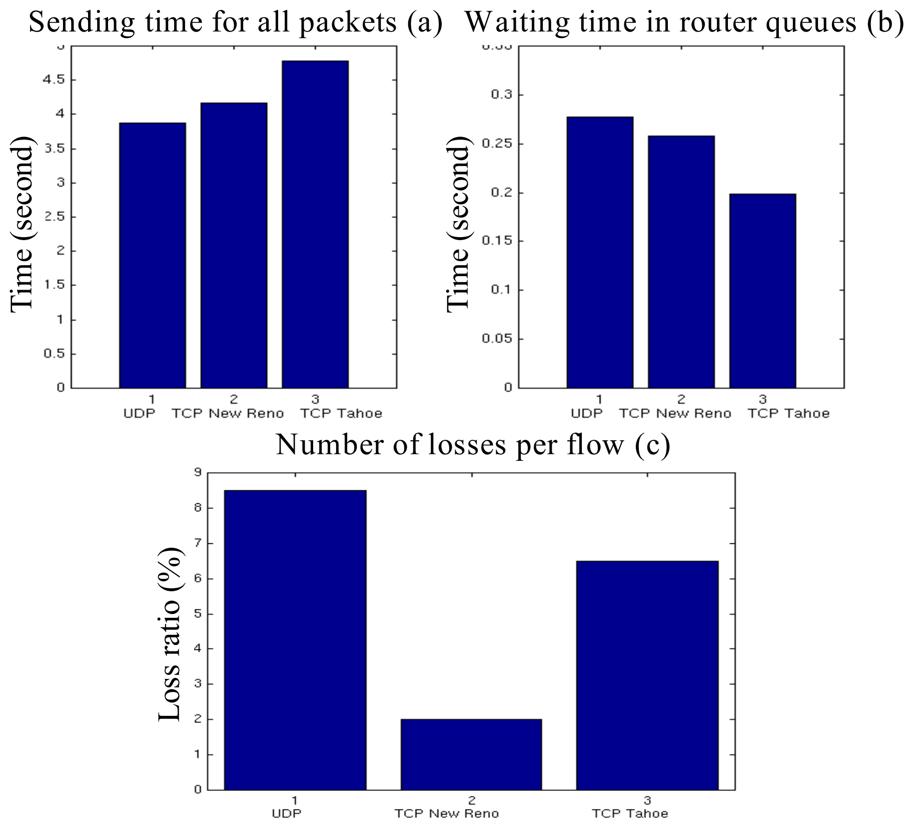

Figure 4. Service differentiation in loaded networks

3.2.3 Congested network (Figure 5)

In the case of a congested network, figure 5 shows some mismatches with our theory on the relation between QoS and aggressiveness, but it is canceled by results shown on Figure 8. Figure 5 and 8 show that UDP traffic is transmitted without difficulties (as planned by our theory). In fact, in case of congestion, the different TCP versions reducing their amount of data sent, they let most of the communication space to UDP flows that do not care about congestion. Finally, UDP takes advantage of all the resources it requires and provides a very good service for users. Of course very important congestion issues can arise because of UDP if its load increases too much, and network integrity may be no more ensured, justifying that way the use of TCP friendly congestion control mechanisms for application taking advantage of UDP services.

Dealing with TCP Tahoe and New Reno, figure 5 shows that the less aggressive protocol finally provides the better service. This result is due to the UDP flow that let only few amount of resources available for the two TCP flows[9]. In this case, TCP New Reno aggressiveness that sends packets with less certitude on network state than TCP Tahoe creates more loss and,

[9] We are in the context of an under-provisioned network.

at the end, a lower QoS. But Figure 8[10], that presents the results of the same experiment with a larger number of flows, then reducing the statistical effects of specific (chaotic) protocol behaviors, and in a more realistic context than the one of Figure 5, shows that TCP New Reno actually provides a better QoS than TCP Tahoe.

Finally, our theory on QoS / aggressiveness works.

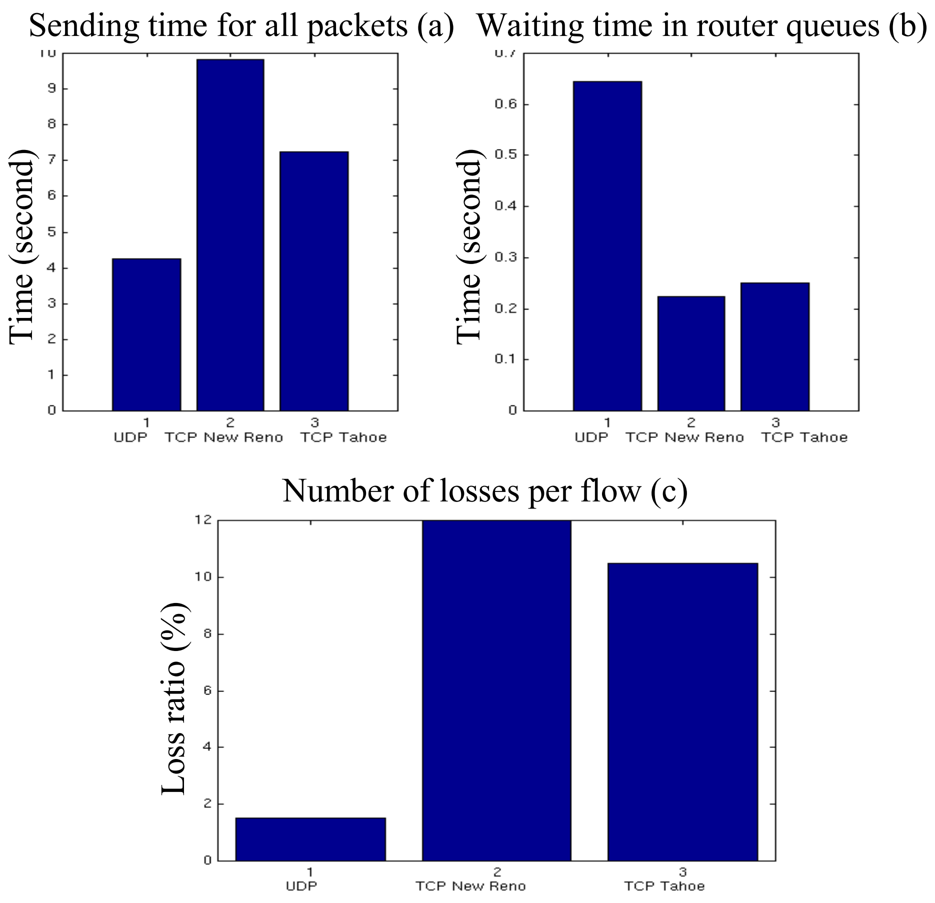

Figure 5. Service differentiation in congested networks

3.3 Fairness between flows of the same class

This new simulation aims to prove that if several flows are using the same QoS (or aggressiveness) class, they will get almost the same amount of resources and the same service. For that, 3 flows (instead of one) of each class are sent in parallel. UDP source sends data at a 150 kbit/s rate. All applications send 200 packets of 1000 bytes each and start at the same time. The minimum theoretical sending time of 200 packets of 1000 bytes is 10.61 s. Figures 6, 7 and 8 show the results in the 3 cases: unloaded, loaded and congested network.

Given these results, fairness inside each class is ensured. In all cases, values for the different parameters are very close for a same protocol, i.e. for a same QoS class. However, fairness between different TCP versions is not ensured.

[10] Figure 8 depicts the results of the same experiment as Figure 5, but instead of having a single source of each kind (UDP, TCP New Reno, TCP Tahoe) there are 3 flows of each kind. It helps to regulate the traffic by averaging the results obtained. Results of Figure 8 are then more realistic than Figure 5's.

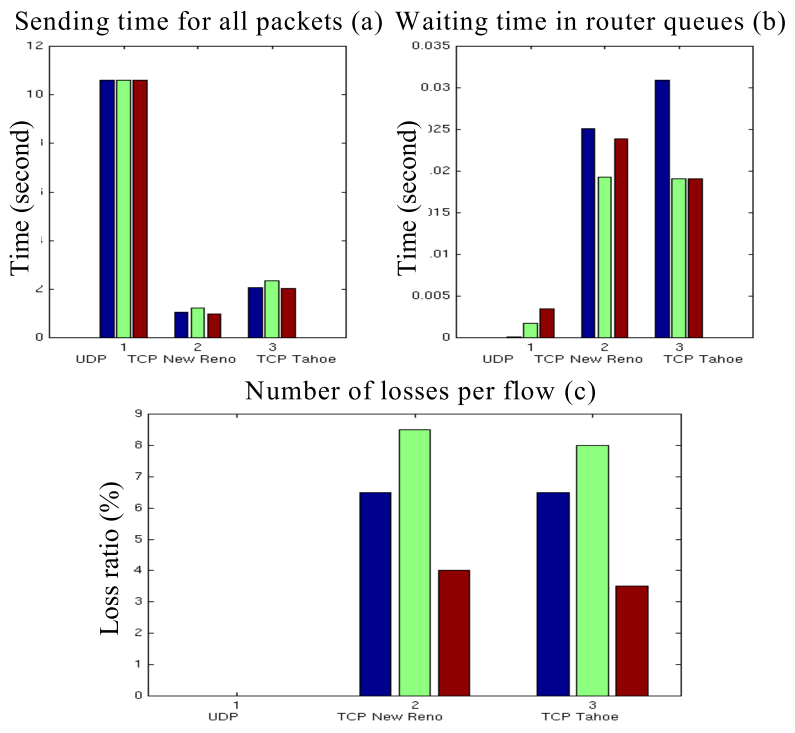

Figure 6. Fairness in unloaded networks

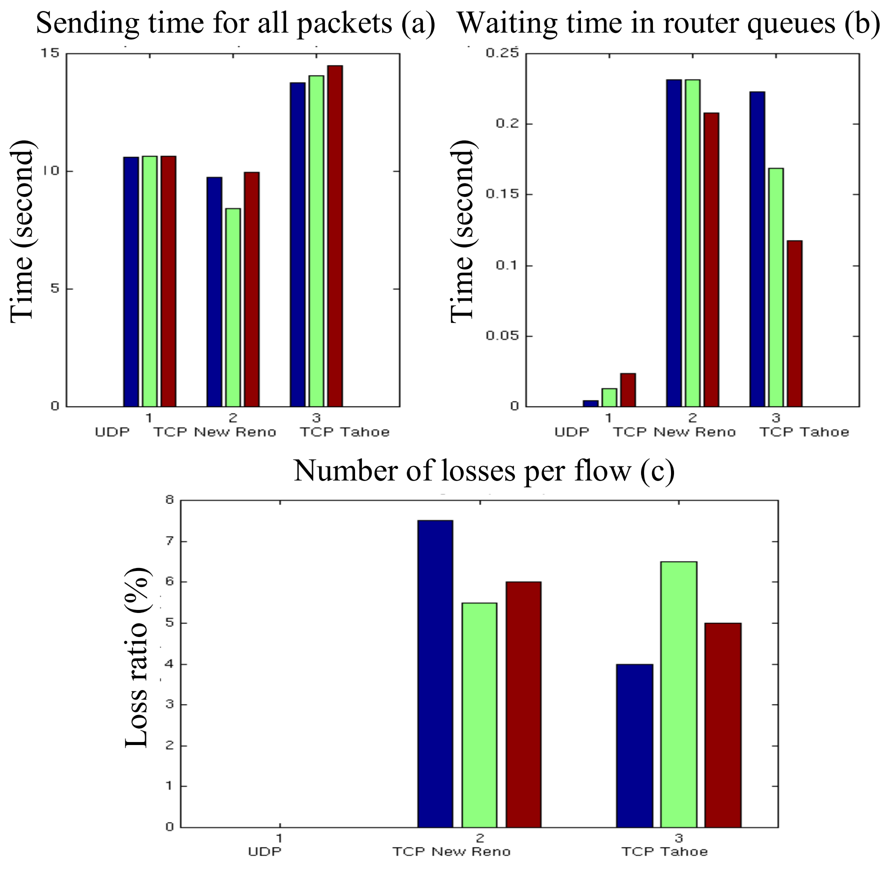

Figure 7. Fairness in loaded networks

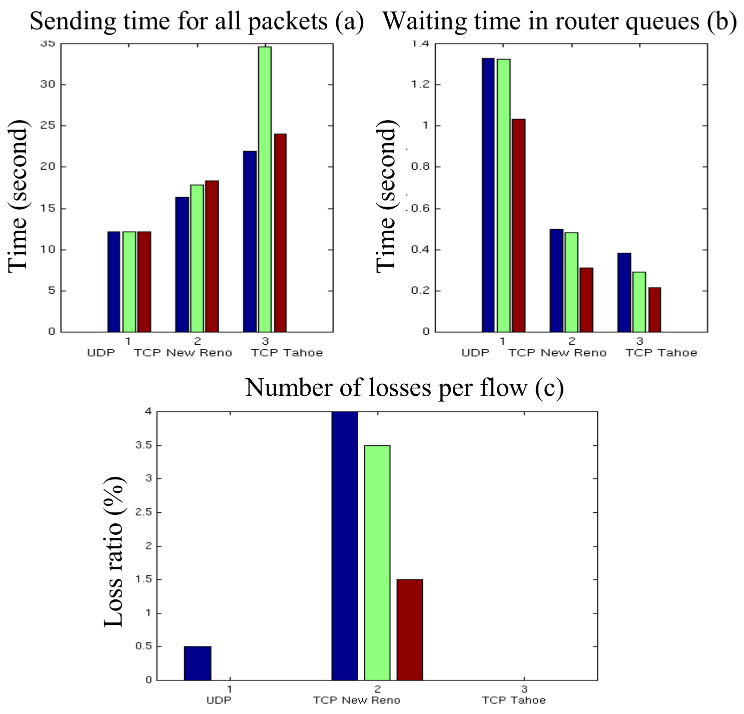

Figure 8. Fairness in congested networks

4 Charging differentiated congestion control classes

This section proposes a quantitative study to find metrics for the QoS provided by different versions of TCP, their aggressiveness, and the relation existing between these two metrics. Compared to section 3, the transport protocol considered are all ancient and current TCP versions as well as new transport protocols that are very promising and that should be used in a short future in the Internet[11]. These protocols are:

- TCP Tahoe;
- TCP New Reno;
- TCP SACK (Selective ACKnowledgements) [13, 15];
- TCP Vegas [18], a version of TCP, not deployed and up to our knowledge not used, that proposes a new congestion control mechanism. This congestion control mechanisms is based on the evaluation of the variation of the RTT on each packet. The evolutions of RTT are interpreted as evolutions of the congestion level on the path: if the RTT increases, it is supposed that the congestion level increases, and then the sending rate is reduced;
- TFRC (TCP Friendly Rate Control) [11], that calculates on the receiving entity the loss rate, and adapt sending rate accordingly. The advantage of such mechanisms is its increased regularity compared to other versions of TCP, and then it is more suited for

[11] In this part we do not considered UDP that does not provide any congestion control and that causes too much troubles in the network, if applications using UDP are not implementing a congestion control. If UDP is used in too important proportions, there is a threat on the liveness of the network. That is why it is recommended to use TCP friendly congestion control even for applications using UDP [17]. All applications and all transport protocols should be TCP friendly in a short future. Besides, to avoid having troubles on the network because of UDP, UDP traffic is separated in routers from the remainder of the traffic, with strict admission control, making its amount remaining very low.

streaming services. Note that the TFRC congestion control mechanisms is one of the mechanism that can be used in the DCP protocol (recently called DCCP), currently studied at the IETF [19].

Then, some NS simulations are lead on the same principle as what was done in section 3, except that we are now taking a larger number of capacities for the backbone network, in order to make this time a quantitative study of the relations between aggressiveness of congestion control mechanisms and QoS provided, and then define charging mechanisms for this service differentiation architecture, proposed in section 2 and demonstrated in section 3.

Figure 9 represents the aggressiveness of the 5 versions of TCP. It appears, as planned by our differentiation architecture, that for the aggressiveness (Ag), i.e. the amount of resource each protocol is able to consume:

Ag(TCP New Reno) > Ag(TCP SACK) > Ag(TCP Tahoe) > Ag(TFRC) > Ag(TCP Vegas)

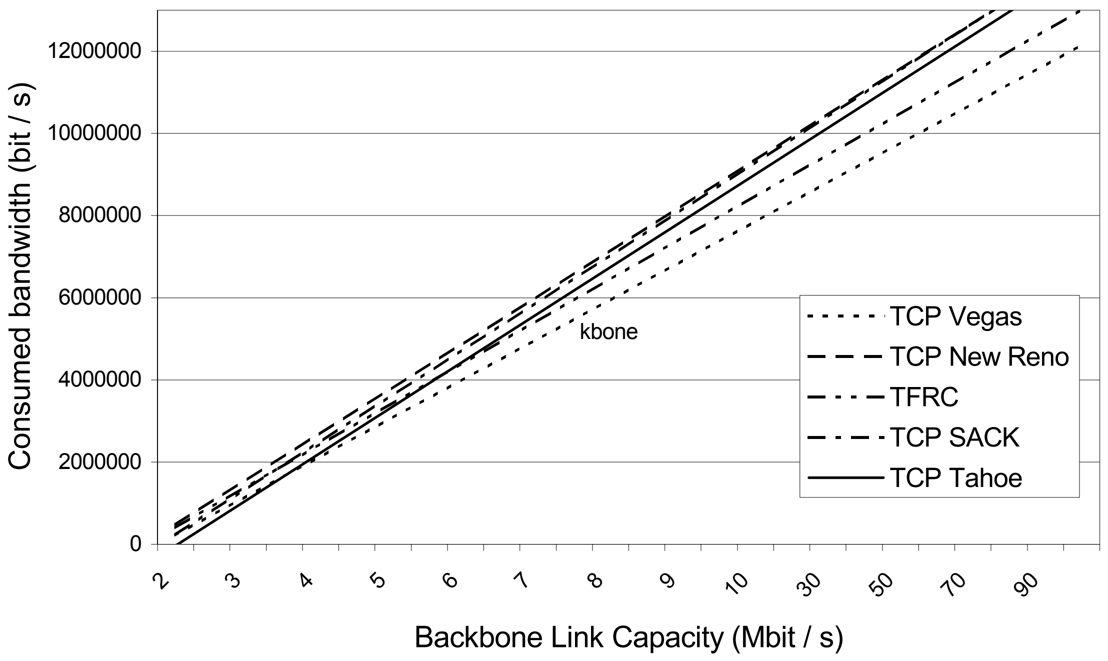

Figure 9. Consumed bandwidth depending on Link Capacity (linear interpolation)

Figure 10 then represents the relation between goodput (i.e. the main QoS parameter with the definition of QoS taken in this paper) and aggressiveness (i.e. consumed bandwidth). The result is pretty much linear and quite impressive. The curve is following almost exactly the curve QoS(TCP X) = Ag(TCP X). This means that charging according to the aggressiveness of congestion control mechanisms matches both users and carriers interests, as there is no divergence between the evolution of the two metrics. Figure 9, for every capacity of the backbone link (meaning in the Internet, the minimum link capacity on the way from the source to the destination host), gives the (mean) value of the charging to apply.

So, for charging user traffic, it is enough to classify packets depending on the congestion control mechanism used at the admission in the network, and to apply the right price corresponding to the protocol class. This class information should be put in the ‘protocol’ field of the IP header. The only difference with the way it is working today is that it is needed to give a different protocol number to each version of TCP. It is then very easy to deploy. No network equipment has to be modified, just new TCP stacks (or other protocols stacks) on end hosts have to be added depending on users needs. As well, this architecture is compatible with the ALF[12] approach [20] that was designed years ago in which each application was supposed to design and implement its own congestion and flow control mechanisms. Finally, this means that such architecture for enforcing and charging QoS can support an infinite number of different classes of services.

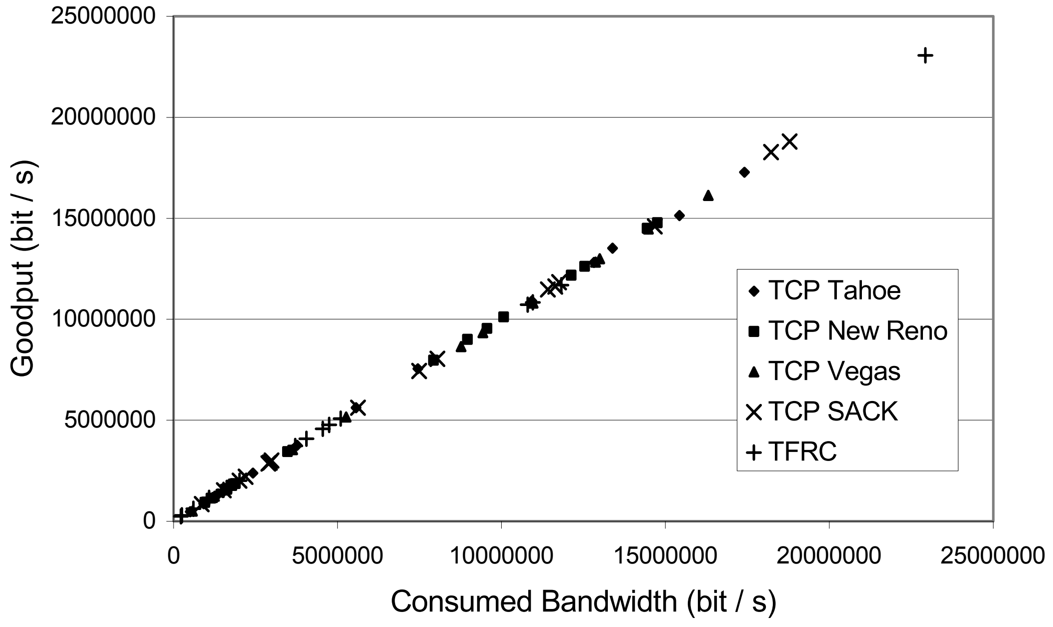

Figure 10. Relation between goodput and consumed bandwidth

5 Conclusion

This paper addressed the issues of enforcing QoS and charging all different services accurately and coherently in the Internet. Much approaches dealing with enforcing QoS and charging provided services are concerned by low layers (from physical to IP) and deal with network architectures and IP traffic management. Main approaches under the spotlights are CDN, DiffServ, over-provisioning, etc. However, based on previous Internet monitoring results, this paper showed that enforcing QoS at the network level only is not enough. Transport layer and its congestion control mechanisms modifies the characteristics of services provided by IP to users, makes it self-similar, with a high variability, and it is then not possible to guarantee any end-to-end temporal parameter to users. Charging such services then cannot be made in a coherent way. The approach described in this paper proposes to define QoS classes according to the aggressiveness of congestion control mechanisms (relative QoS), to regulate the traffic of every sources, and charge services accordingly. This approach has been illustrated with examples: two of them are described in the paper.

This approach then proposes as a part of the solution to the Internet QoS issue a new architecture for services differentiation and differentiated charging mechanisms based on the

[12] ALF: Application Layer Framing

aggressiveness of congestion control mechanisms, at transport level. It means that every application, every flow, will be able to take advantage of the best suited (eventually its dedicated) transport protocol (similar to the ALF approach, but at transport level). Nevertheless, this architecture does not guarantee QoS parameters as delay, throughput, goodput, loss ratio, etc. that are related more specifically to layer 3. It just guarantees end-to-end-service differentiation in the current Internet, i.e. two flows of different classes will get QoS accordingly, and flows of the same class will get almost the same service[13,14]. On the other side, as aggressiveness (i.e. QoS) is dependent from the amount of consumed resources, it has been demonstrated that it is possible to charge services according to the transport classes proposed in this paper. Such charging mechanisms are coherent with users as well as carriers strategies. Besides, this paper gives a QoS metrics based on a metric of the aggressiveness that can be used to quantitatively predict resources that will be consumed by a flow, and charge the service that will be provided.

In addition to the service differentiation, and because of the oscillating nature of current flows, as well as link traffic, our architecture recommends to promote the use of congestion control mechanisms that make traffic smoother. It will help to improve performances, QoS and to stabilize the offered service. Our approach also proves to be very coherent as the more aggressive a protocol, the more oscillating the generated traffic. Then, the service that will be charged more is the one that is the more unstable and thus the one that is the more able to provoke (short term)congestions on the network. Our proposal is in fact an extension of the smart market model adapted to the current nature of Internet traffic. These results have been inspired by recent work on network monitoring and its results on Internet traffic modeling. Note also that in this paper, we just consider the TFRC congestion control mechanism as an example of stable transport protocol. Of course, the description of other work on that topic is not in the scope of this paper, but designing new transport protocols that infer less oscillations or self-similarity in the traffic is one of the major research topics for next generation Internet.

Another important aspects of charging is related to non cooperative users that may spoof the protocol number of the IP packet to avoid paying the higher prices. Up to now, this work is at its early stage, but we will give the principles of the solution whose evaluation is in process. The proposed solution for controlling that users are not spoofing their real “protocol” field consist in using monitoring tools. In fact, monitoring equipments are now available in every router of the Internet (for instance with the NetFlow tool of CISCO [22]). Such tool is able to provide network operator with any kind of information on traffic, and in particular on users flows in the network. With such equipment, it is then easy to develop a small piece of code to calculate the real aggressiveness[15] of the protocol used by one flow and to check if it is really matching the behavior of the protocol indicated in the “protocol” field. In fact, such

[13] This solution for transport level service differentiation does not require any change of the current TCP/IP or network architecture. It just requires to assign a new protocol number to any transport congestion control mechanism and to use this field of the IP header to classify packets and perform service differentiation according to the associated aggressiveness metrics.

[14] It is to note that this approach is completely independent from the underlying technology. This approach does not care if the underlying network is a CDN, an over-provisioned network to provide 100% of premium IP service, or DiffServ. Our approach can be put on top of any network architecture and perfectly works. It is to note however that if an IP service differentiation mechanism is used, then a good choice of scheduling and discarding strategies for router queues, with also a right mapping with aggressiveness classes of transport protocols, optimizes the final end-to-end QoS provided to users. This point is demonstrated in [21].

[15] Measuring aggressiveness, here, can be as simple as measuring the slope of the curve giving the throughput of the flow depending on time.

an approach for detecting non cooperative users is directly inspired by some kinds of Intrusion detection systems (IDS) that are detecting intrusions just by comparing actual traffic with regular signatures [23]. Finally, it is also to note that the monitoring tool that is hosted in routers is also the key tool for billing as it is able to extract flows from the global traffic, and then able to generate the invoice for customers.

6 References

[1] Braden, R., Clark, D., Shenker, S., « Integrated Services in the Internet Architecture : An overview », RFC 1633, June 1994

[2] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W., « An Architecture for Differentiated Services », RFC 2475, December 1998

[3] Allman, M., Paxson, V., Stevens, W., « TCP Congestion Control », RFC 2581, april 1999

[4] Paxson, V., Floyd, S., « Wide-Area Traffic: The Failure of Poisson Modeling », IEEE/ACM Transactions on Networking, vol. 3, no. 3, June 1995

[5] Uhlig, S., Bonaventure, O., « Understanding the long-term self-similarity of Internet traffic », in Proc. Quality of Future Internet Services workshop (QoFIS'2001), 2001

[6] Venkatachalam, M., Kaur, J., Vin, H., « End-to-End Model for a Flow in the Internet », To appear in proceedings of INFOCOM, 2002

[7] Crovella, M., Bestavros, A., « Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes », IEEE/ATM Transactions on Networking, vol. 5, no. 6, December 1997

[8] Leland, W., Taqqu, M., Willinger, W., Wilson, D., « On the self-similar nature of Ethernet traffic (extended version) », IEEE/ACM Transactions on Networking, 1994

[9] Erramilli, A., Narayan, O., Willinger, W., « Experimental queueing analysis with long-range dependent traffic », IEEE/ACM Transactions on Networking, April 1996

[10] Wolf, R., « Stochastic modeling and the theory of queues », Prentice Hall, 1989

[11] Floyd, S., Handley, M., Padhye, J., Widmer, J., "Equation-based congestion control for unicast applications", ACM SIGCOMM, 2000

[12] Stevens, W.R., « TCP/IP Illustrated, Volume 1: The Protocols », Addison-Wesley, 1994

[13] Fall, K., Floyd, S., « Simulation-based Comparisons of Tahoe, Reno, and SACK TCP », Computer Communication Review, Vol. 36, n°3, July 1996

[14] Floyd, S., Henderson, T., « The New Reno Modification to TCP's Fast Recovery Algorithm », RFC 2582, April 1999

[15] Mathis, M., Mahdavi, J., Floyd, S., Romanov, A., « TCP Selective Acknowledgment Options », RFC 2018, October 1999

[16] Jacobson, V., « Congestion avoidance and control », proceedings of SIGCOMM'98, august 1998

[17] Floyd, S., Fall, K., "Promoting the use of end-to-end congestion control in the Internet", Transactions on Networking, February 1998

[18] Brakhmo, L.S., O'Malley, S.W., Peterson, L., « TCP Vegas: New Technique for Congestion Detection and Avoidance », Proceedings Of ACM SIGCOMM'94, London, October 1994

[19] Floyd S., Kohler E. and Padhye J., "Profile for DCP congestion control ID 3 : TFRC congestion control", Internet Draft, IETF, November 2001.

[20] Clark, D.D., Tennenhouse, D.L. "Architectural Considerations for a New Generation of Protocols," in SIGCOMM Symposium on Communications Architectures and Protocols, (Philadelphia, Pennsylvania), Sep. 1990

[21] Owezarski, P., Martinie, C., « Conception et simulation d'une architecture à différenciation de services alternative à DiffServ », Rapport LAAS No. 01447, Octobre 2001, in French

[22] "NetFlow Services Solutions Guide", http://www.cisco.com/univercd/cc/td/doc/-cisintwk/intsolns/netflsol/

[23] S. Axelson, "Intrusion detection systems: a survey and taxonomy", Technical report N° 99-15, Department of computer enfineering, Chalmers university of technology, Sweden, March 2000