End-to-end delay constrained protocol over the EDS service differentiation

Benjamin Gaidioz, Pascale Primet

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

End-to-end delay constrained protocol over the EDS *service differentiation*

Benjamin Gaidioz, Pascale Primet

N° 5030

December 2003

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--5030--FR+ENG

INRIA
RHÔNE-ALPES

End-to-end delay constrained protocol over the EDS service differentiation

Benjamin Gaidioz*, Pascale Primet†

Thème 1 — Réseaux et systèmes
Projet RESO

Rapport de recherche n° 5030 — December 2003 — 16 pages

Abstract: The TCP/IP stack has been mainly designed for elastic traffic (file transfers). It is nowadays recognized that it is not able to efficiently support traffic patterns with completely differing requirements (e.g. applications with delay requirements). Service differentiation at the flow agregate level (DiffServ) is a promising way to implement some form of IP QoS because it is robust and scalable. The EDS PHB is a Diffserv PHB based on both loss rate and delay proportional differentiation. In this article, we start from a network layer implementing the EDS service differentiation and present a specific transport protocol which aims at providing an end-to-end delay guarantee to applications with delay requirements.

Key-words: IP quality of service, service differentiation, Equivalent Differentiated Services, guaranteed delay over IP

* Université Lyon 1, LIP RESO (UMR CNRS-INRIA-ENS-UCBL N°5668)
† LIP RESO (UMR CNRS-INRIA-ENS-UCBL N°5668), INRIA

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Protocole à contrainte de délai de bout-en-bout sur la différenciation de service EDS

Résumé : La pile de protocoles TCP/IP a été conçue principalement pour transporter du trafic « élastique » (transferts de fichier). Aujourd'hui, il est admis qu'elle ne convient pas pour de nouvelles catégories de trafic aux besoins très différents (par exemple des flux temps-réel avec des contraintes de délais). La différenciation de services au niveau d'agrégats de flux (DiffServ) est une architecture prometteuse pour mettre en oeuvre une forme de qualité de service IP car elle est robuste et s'étend bien à un grand nombre de noeuds. EDS est un PHB Diffserv qui s'appuie sur la différenciation proportionnelle en délai et taux de perte. Dans cet article, en nous appuyant sur le modèle de différenciation de service EDS, nous présentons un protocole de transport qui garanti un délai de bout-en-bout du réseau, ceci pour répondre aux besoins d'applications qui ont des contraintes de délai.

Mots-clés : qualité de service sur IP, différenciation de services, Equivalent Differentiated Services, délai garanti sur IP

1 Introduction

The network layer of Internet (IP) provides a simple, robust and effective packet forwarding service. The TCP/IP stack has been mainly designed for elastic traffic (e.g. FTP). It is nowadays recognized that it is not able to efficiently support new traffic patterns with different requirements in terms of speed, latency and reliability (e.g. real-time applications, interactive applications, bulk transfers, etc.). It is commonly accepted that IP needs to be extended with a form of quality of service (QoS). In the case of IP, adding QoS has to be done carefully in order to maintain the efficiency and robustness of the network.

The DiffServ architecture [1] is convincing as a QoS approach on a large scaled network. Its principles keep the network layer simple and robust. In the core network, IP packets are expected to be marked with a specific *class identifier*. This identifier selects a forwarding treatment met by the packet each time it crosses a router. From the core network point of view, there is a very small number of classes, there is no resource reservation between hops. Marking is done at the edge routers, where the number of flows is sufficiently low to apply marking rules without losing much performance.

A characteristic that makes TCP/IP so appealing is its ease of use: a host can plug into the network and use it immediately. According to its design philosophy [2], IP attempted to provide a basic *building block* out of which a variety of types of service could be built. The decision was an extremely successful one, which allowed the Internet to meet its most important goals.

In this report, we rely on the implementation of the EDS service differentiation [8, 7] at the IP level (see in the technical report [7] for details). This architecture provides a best-effort service differentiation which is not sufficient to satisfy most of the applications needs. Thus, it needs to be extended with end-to-end protocols which use the services it provides in order to ensure some higher level guarantees to applications.

We start the article with a quick reminder on EDS (sect. 2). Then we present an adaptive transport protocol which aims at providing an end-to-end delay guarantee to applications with delay requirements in sect. 3. In sect. 4, we present related work. Finally, in sect. 5, we give conclusions and future work.

2 Quick overview of EDS

Fig. 1 gives the intuition of the concept of Equivalent Differentiated Services. It shows both the service provided by a best-effort router and an EDS router. Roughly, the EDS router provides a set of N classes, each one experiencing a different level of performance in both queuing delay and loss rate. Considering a single performance criteria (delay or loss rate), the range of performance is centered around an average performance, which is the performance all packets would have obtained through a plain best-effort router. Thus, the performance is directly linked to the router load. Moreover, there is an asymmetry between delay and loss rate performance: The class which obtains the i^{th} best performance in delay obtains the i^{th} worst performance in loss rate.

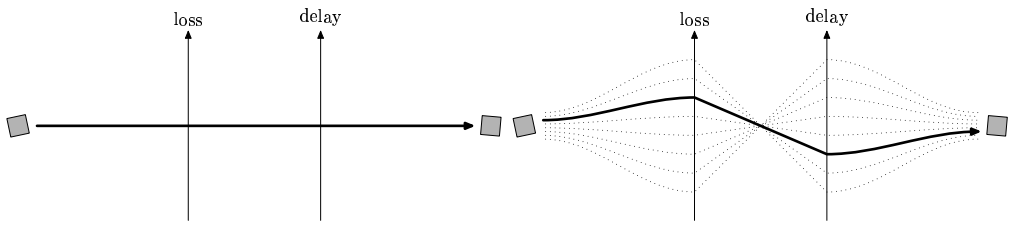

Figure 1: A packet (gray square) is crossing a best-effort router (left) or an EDS router (right). Through the best-effort router, the packet experiences a given queuing delay and a given loss probability. Through an EDS router, the packet is member of a service class among N classes, where it experiences a specific queuing delay and a specific loss probability, both being relatively better or worst than the performance through a best-effort router.

The EDS service differentiation proposal has been designed by starting from the Internet protocol design principles which gave to IP its robustness, ease of deployment and ease of use. Thus, EDS provides *best-effort* service differentiation.

Stronger guarantees on the plain best-effort IP are implemented in end-to-end protocols. For example, TCP guarantees reliability. These protocols use the building block provided by IP to provide a specific service matching the need of specific applications. Since we have defined a best effort service differentiation system from the same design rules, the natural way to implement stronger QoS guarantees has to be done in the protocol layer.

3 Delay constrained protocol

In this section, we describe a transport protocol which aims to match the needs of an application in terms of end-to-end delay and loss rate. Considering an application sending packets at a given rate with the expectation that the packets are received within a delay shorter than a known delay bound. Moreover, the end-to-end reliability has to be relatively high. Delay bound and maximum loss rate are both given as parameters to the protocol.

3.1 End-to-end delay constrained transport protocol

On a plain best-effort network, there is of course no way to control neither the end-to-end delay nor the loss rate. Packets are forwarded in a delay and with a loss probability depending on the network load.

Assuming the replacement of the best-effort service with EDS service classes, a protocol has the possibility to use a specific best-effort class with a delay which is proportionally higher or lower than with the plain best-effort service. The end-to-end delay still depends on routers load. However, by switching from a class to an other, the end protocol can have a control over the end-to-end delay anyway.

Since the application described has delay constraints, it may move from the center classes (which experience an average performance) to quicker classes. However, it is inherent to EDS that quicker classes gets a higher drop probability. Thus, the application has to choose a good compromise.

We assume that the *receiver* can compute the end-to-end delay in order to decide if the packet is valid or not. We do not address the synchronization question here. The algorithm is given in algorithm 1.

```
nb_paq ← nb_paq + 1
if temps ≤ Δ then
  nb_ok ← nb_ok + 1
end if
seq_max ← max(seqno, seq_max)
seq_min ← min(seqno, seq_min)
nb_sent ← seq_max − seq_min)
if nb_paq = N then
  s_n ← nb_paq ÷ nb_sent
  s_d ← nb_ok ÷ nb_paq
  s ← s_d × s_n
  if s < S then
    if s_n < 1 ∧ s_d = 1 then
      c ← c − 1
    else if s_d < 1 then
      c ← c + 1
    end if
  end if
end if
```

Algorithm 1: Packet reception

- Periodically, the sender sends a given number of packets, each one identified by a sequence number. Depending on the network load, some of them are dropped while being forwarded to the receiver. We note s_n the share of the packets which reach the receiver ($0 \leq s_n \leq 1$).
- The receiver keeps and uses packets only when they crossed the network in a delay shorter than Δ. We note s_d (share due to the delay) the share of packets which are kept because they are in time.

Thus, the share of packets kept by the application is $s = s_d \times s_n$. If s is sufficiently high, the application receives a sufficiently large number of valid packets and runs correctly. If s is too low, the application does not work correctly because it receives not enough valid packets, either because they are late or because they are dropped by EDS. The receiver

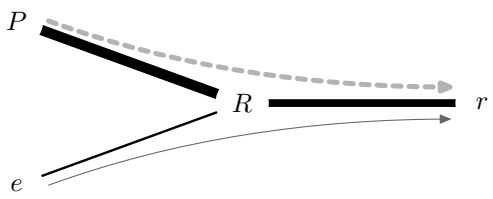

Figure 2: The simulated topology. The router is node R, receiver is node r, random Pareto sources are locate on node P and the application with delay requirements is located on host e.

warns the sender of this situation by sending a control message to it, so that it uses a new class identifier.

1. A value of $s_d < 1$ means that packets carried from the sender to the receiver are sometimes to slow. Their average delay is probably close to Δ and queuing delay in routers could be increasing. Using a slower class may result in $s_d \sim 0$ then $s \sim 0$. The receiver asks the sender to use a quicker class where s_d should be equal to 1. However, s_n may become too low. In the implementation, the sender to chooses the quickest class (and not the class just quicker than the current one).

2. If $s_d = 1$ but $s_n < 1$, the protocol is using a class with a network loss rate which is too high. The class may be quick enough but provides an insufficient reliability. The receiver asks for a slower class where the loss rate is lower, in the expense of a larger delay. In the implementation, the sender selects the slowest class near the current one.

It can happen that using a quicker class make s_n getting too high, or using a slower class makes s_d getting too high and the performance remains bad. In such case, the protocol may oscillate from a class to an other while the application does not work. It means that the QoS requirements of the application are too strong. The application would not have worked on a best-effort network anyway.

3.2 Simulation of the protocol

The protocol has been implemented in NS [9] and runs on a topology (see fig. 2) where it meets concurrent random traffic generated according to a Pareto distribution. Random traffic and application packets come from two different links and goes through a router (plain RED or EDS).

- The output link has a bandwidth of 10 Mb/s.
- There are eight EDS classes. Delay and loss rate are differentiated so that classes obtain a relative delay ranging from 1 to 8 and a loss rate ranging from 8 to 1. Thus, class 1

is to slowest and class 8 is the quickest. Class 8 has the highest loss probability and class 1 the lowest.

- There are eight Pareto sources (one per class). The average load generated by Pareto sources is a bit higher than what the output link can handle, so that packets experiences various queuing delays and random losses probability. The random sources link has a highest bandwidth so that they can send bursts. Their average rate is equal to the output link bandwidth.
- In the experiments, links latency is the same everywhere and is equal to δ where δ vary from an experiment to an other.
- The router queue is set to K where $K = 300$ most of time except in one experiment where $K = 100$. The RED mechanism starts dropping packets when queue length is 20 packets long with a maximum threshold of K where the maximum average drop probability is 25%.

In the next four experiments, we vary the link latency, queue capacity and network load and see how the protocol behaves on best-effort and EDS. We show three types of graph.

delay distribution graph For each packet, we compute the end-to-end delay. From these statistics, we draw a graph showing the share of *sent* packets which where received in a delay shorter than t, for all t between zero and the maximum experienced delay. Since only a specific share σ of all sent packets is received, the curve ranges from zero to σ.

class ID evolution Each time the receiver asks for a new class identifier, we show it on a graph, so that it is possible to follow the evolution of the class identifier during the experiment.

share s of valid packets This is the evolution of s during the experiment.

The value of s is computed each the receiver receives 32 packets.

3.2.1 In the case where the expected end-to-end delay is too low compared to end-to-end latency

In the configuration we take here, there is almost no hope for the application to run well because the end-to-end link latency is large. Combined with queuing delay, even the quick class cannot ensure a good end-to-end delay. Links propagation delays are set to 90 ms. Thus, the end-to-end minimum delay is 180 ms while the limit is 200 ms. Results are shown on fig. 3.

end-to-end delay Because of the link latency, only packets sent when concurrent traffic is low (at the beginning) are received below the 200 ms threshold. In the entire experiment, whatever the underlying network layer (EDS or best-effort), most of the packets

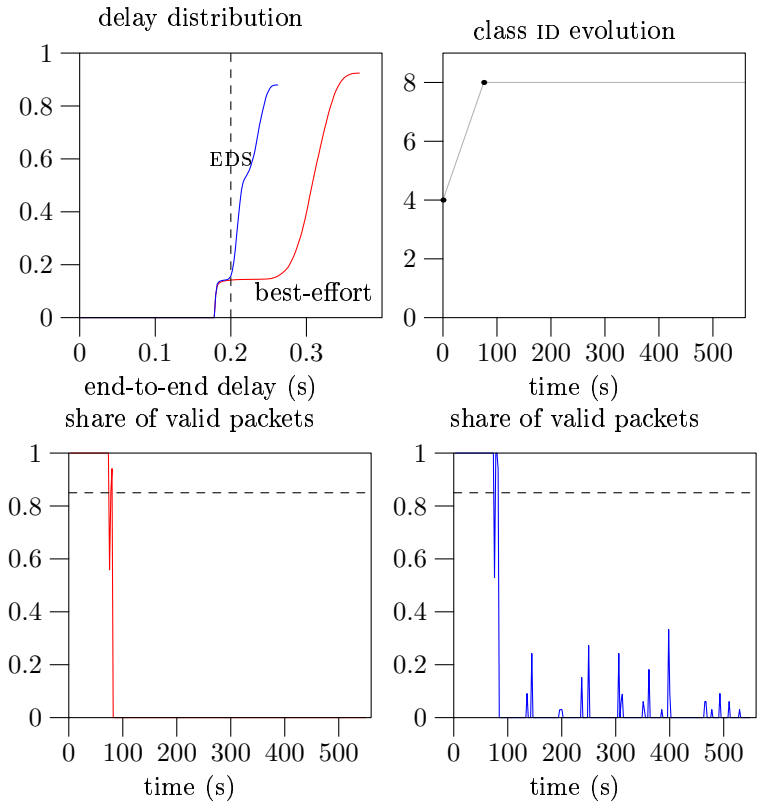

Figure 3: Experimental results with a link latency of 90 ms, a queue size of 300 packets and a moderate network load.

are late. The delay distribution graph shows that, when running over EDS, the protocol choosed quick classes since most of the packets are received in a delay shorter than the average delay met on best-effort.

evolution of the class identifier The protocol moves to class 8 because of the delay that becomes too high. It does not move from there until the end. Indeed, since most of the loss is due to the delay and it cannot choose a quicker class, it stays in class 8.

share s of valid packets When load increases, since packets are received late, the share s of valid packets is equal to zero most of time. There are some peaks with EDS where s grows but is anyway below the threshold of 85%.

3.2.2 In the case where expected end-to-end delay is large compared to end-to-end latency

In the configuration we take here, the end-to-end latency is low compared to the application needs. Combined with queuing delay, even the slow class ensures a good end-to-end delay. Links propagation delays are set to 10 ms. Thus, the end-to-end minimum delay is 20 ms while the limit is 200 ms. Results are shown on fig. 4.

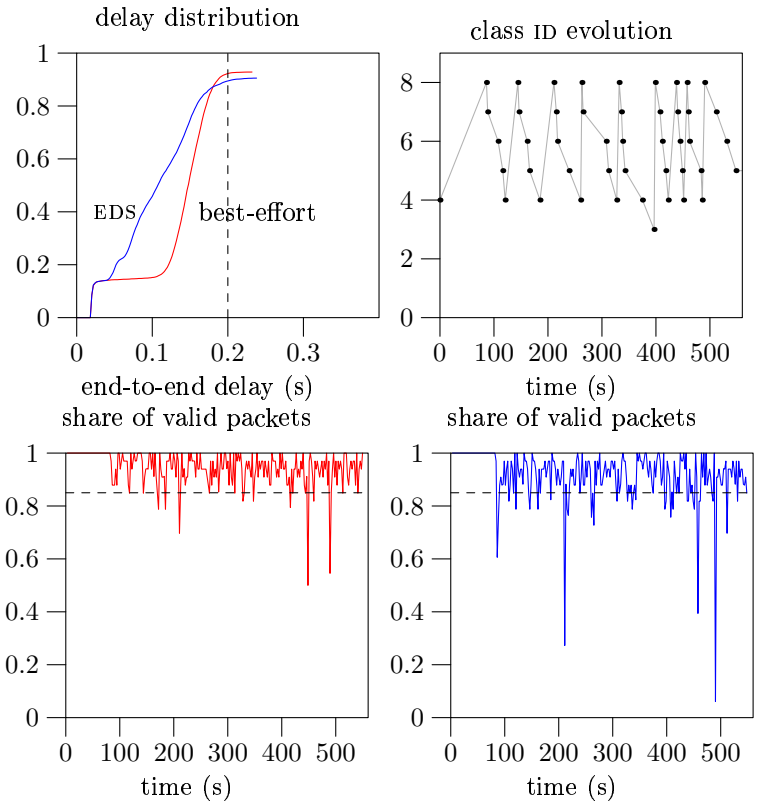

Figure 4: Experimental results with a link latency of 10 ms, a queue size of 300 packets and a moderate network load.

end-to-end delay Since the link delay is low, most of the packets are received below the 200 ms threshold, with both EDS and best-effort.

evolution of the class identifier The class identifier oscillates between class 4 and class 8. Because of high delay in low classes, it periodically jumps to class 8 in order to obtain

a low delay. There, it meets sometimes a high loss rate, due to loss rate differentiation and moves to a slower class.

share s of valid packets Since most of the packets are received below the 200 ms threshold, with both EDS and best-effort, the share s of valid packets is good (above the 85% threshold). Using EDS, there are some places where s is very low. This is due to delay differentiation when the protocol moves to a slow class. Usually, it jumps right after in the quickest class.

3.2.3 In the case where expected end-to-end delay can be reached but loss rate is too high

In the configuration we take here, the end-to-end latency is low compared to the application needs. Combined with queuing delay, even the slow class ensures a good end-to-end delay. Links propagation delays are set to 10 ms. Thus, the end-to-end minimum delay is 20 ms while the limit is 200 ms. However, load is high and loss rate is high. Results are shown on fig. 5.

end-to-end delay In this experiment, the average loss rate is high. Because of the high loss rate and thanks to the low delay (links delay are low), the protocol, when running over EDS moves to a *slow* class where it meets as best as possible its delay requirements and loss rate requirements. This explains why the delay distribution with EDS is shifted to the right, the protocol uses a slow class. The delay distribution curve is ends on a highest value with EDS because it uses a class with a low loss rate.

evolution of the class identifier The class identifier moves from class 4 to class 1 because of the loss rate. Then, it does not move from there.

share s of valid packets Since the link latency is low, the share s of valid packets when running over best-effort, is close to the threshold of 85%. However, because of the high load, it is often below the threshold. In the case of EDS, since the protocol uses a slow class, it obtains a low network loss rate and a sufficient end-to-end delay. Thus, its share s is much better.

3.2.4 In the case where expected end-to-end delay can be reached without too much loss rate

The configuration we take here is a good one where it is possible for the protocol to ensure the end-to-end delay and a sufficiently low loss rate to the application. Links propagation delays are set to 40 ms. Results are shown on fig. 6.

end-to-end delay A much larger part of packets are received in time when the protocol runs over EDS. This is due to the quick classes where the loss rate is sufficiently low.

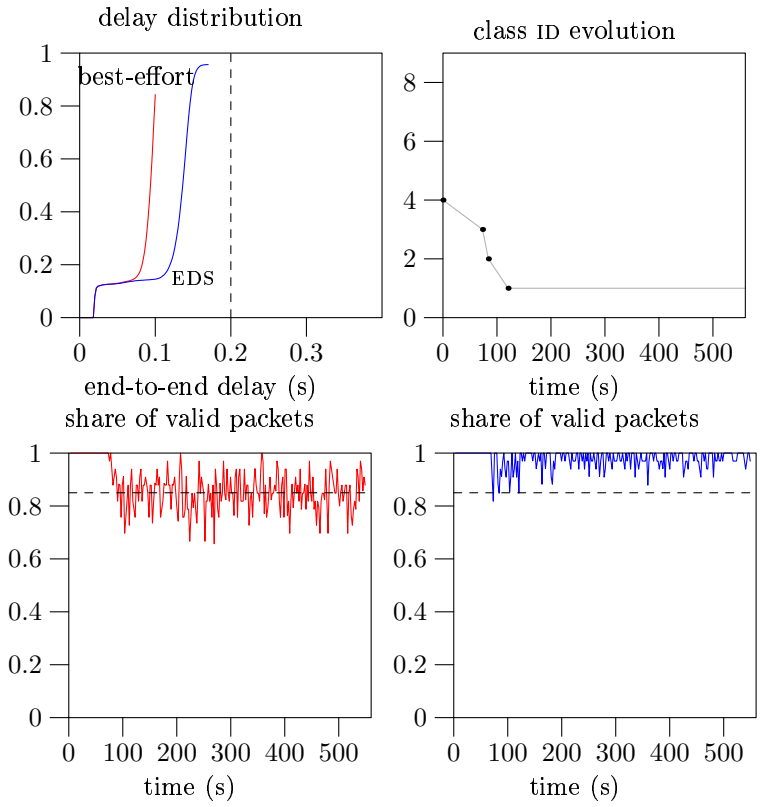

Figure 5: Experimental results with a link latency of 10 ms, a queue size of 100 packets and a high network load.

evolution of the class identifier The protocol moves between class 8 and class 5. Each time it meets a high loss rate (low s), it moves either to class 8 if it is due to delay or to a slower class if it is due to the network loss rate.

share s of valid packets The share s of valid packets is much better when the protocol runs over EDS because most of its packets arrive in time. Moreover, the threshold of 85% is met most of time.

3.3 Overall performance of the protocol

In this section, we conduct the experiment a very large number of times by making some parameters vary a bit each time. This permit to have an idea of the overall performance of the protocol under various network conditions.

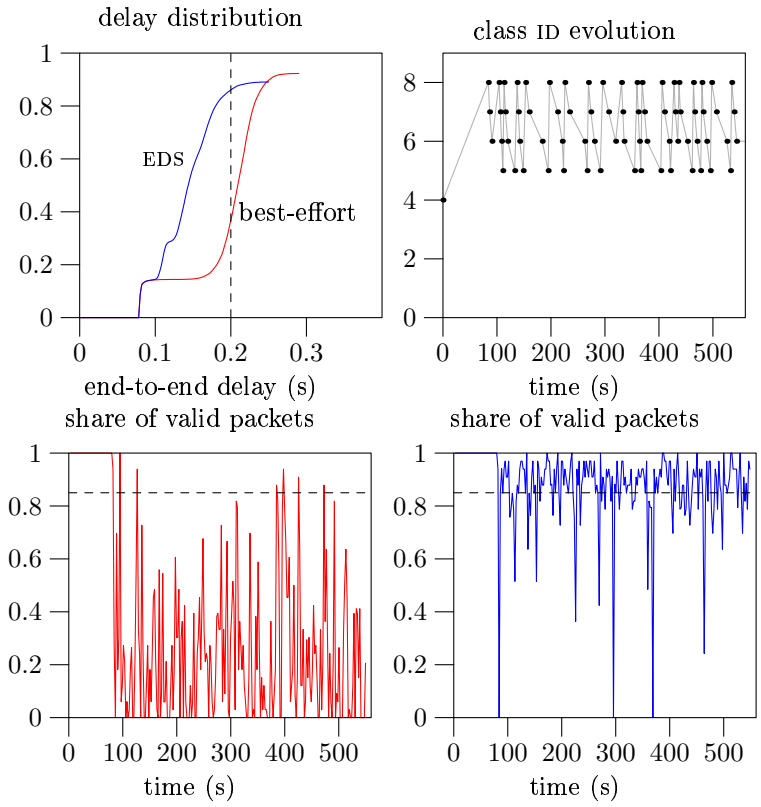

Figure 6: Experimental results with a link latency of 40 ms, a queue size of 300 packets and a moderate network load.

Because of the very large number of experiments, we do not show the absolute values of the protocol performance. All performance (s) are represented by a dot in a rectangle where their coordinates (x, y) represents the network conditions (x stands for the application delay requirements, y stands for intensity of random traffic). The little dot is more or less dark depending of the share of time the applications ran under good conditions: white when no valid packet is received, black if all packets are received and more or less gray otherwise.

3.3.1 Example

As an example, we show on fig. 7 the shape of the graphs when considering only one constraints ($s = s_d$ or $s = s_n$). The more the dot is on the left, the less the delay contrainst, the more it is on the top, the higher the concurrent trafic intensity.

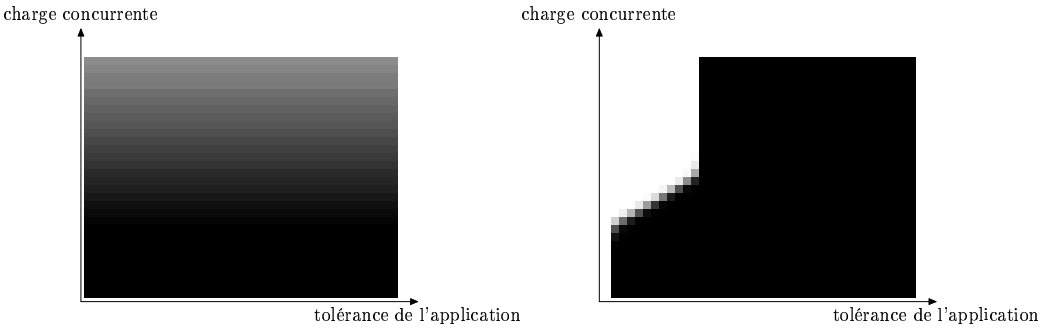

Figure 7: Performance of the application in a best-effort network (share of the time the application ran with good conditions) when $s = s_d$ (left) or $s = s_n$ (right).

Thus, considering the s_d share, when the application has a low delay constraints, most of the dots are white, but when the intensity of the random trafic decreases, there are some network conditions that are fine for the application because the queue length is sufficiently short. When the delay constraint is high, all packets which arrive (not all packets are received, it depends on the loss rate in the network) arrive in time, so that all dots are black.

In the case of the s_n share, it does not depend of the delay constraints. All dots become more and more white when the intensity of the trafic increases.

3.3.2 Best effort vs. EDS

We show on fig. 8 the application performance when running in a best-effort network and an EDS network. The graphs show how EDS can improve the performance of the application.

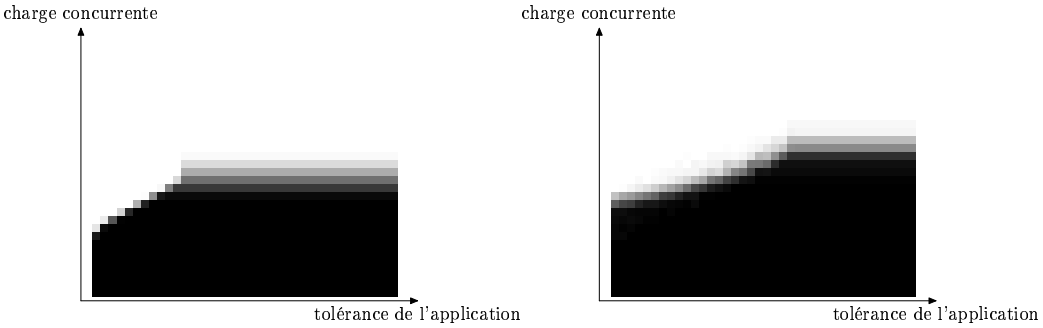

Figure 8: Performance of the application in a best-effort network and an EDS network (share of the time the application ran with good conditions) with $s = s_d \times s_n$.

- When it has strong delay requirements, the application can work better with a high load of concurrent traffic. The dark dots spread over a larger range using EDS than best-effort.
- In average network condition, the application runs fine with both systems,
- Under a large load but with lower delay requirements, the protocol switches to a class with a lower loss rate with EDS and the application work better than with best-effort.

3.4 Conclusion on the protocol

When the network load or network topology makes it impossible to obtain the expected performance, the protocol does not obtain a worst performance on EDS than with best-effort. Experiments shows that thanks to EDS, it is possible to obtain a better quality of service than on a plain best-effort network when the network topology, load and differentiation allows it.

The delay constrained protocol presented above combined with EDS is not expected to provide a hard quality of service to the application using it. With *proportionally* differentiated services, a protocol cannot guarantee an end-to-end delay if the network load is too high. Moreover, even if a sufficiently short delay can be obtained thanks to delay differentiation, there is no guarantee that the associated loss rate will permit the application to work well. Using quicker classes oblige the protocol to lose packet more frequently than slower classes.

4 Related Work

Similar work is conducted on top of different service differentiation systems.

- The authors of proportional differentiated services [3] have seen the need to match stronger QoS requirements from the end-to-end viewpoint since proportional differentiation does not provide absolute guarantees. An adaptive protocol [4] has been designed to dynamically select the best class and then provide an absolute guarantee in delay.
- The TCP-friendly differentiated services marking [5] is based on the same way to consider service classes as a building block protocols can use with a fine grained adaptation algorithm. The system is based on the standard *assured* service where classes get different drop level probability (no delay differentiation). Their objective is similar to us since they do not aim at providing strong guarantees but improving the overall performance (they prove that they obtain less timeouts).

In both cases, the underlying service differentiation layer provides privileged classes. The architectures must be deployed with an access control system (or marking has to be done by a trustworthy border entity) to ensure a cordial use of services.

5 Conclusion and future work

We have presented both the EDS best-effort service differentiation system and a transport protocol built on top of it.

The EDS system has been designed by mapping the IP design philosophy to service differentiation. The same way TCP has been designed to provides reliability on top of the unreliable network layer IP, we have presented a protocol which provides specific soft quality of service properties to applications. It ensures *as best as possible* an end-to-end delay and a relative reliability to a real-time application.

We want to increase the range of QoS needs that this architecture could solve. Specifically, we are working on other protocols derived from TCP that provide a better support to interactive applications like SSH flows or a support to long bulk transfers.

From a more practical point of view, since we implemented EDS with proportional differentiation schedulers in Linux [6], we are also going to start the implementation of the protocol presented in this article and see how it runs in real life.

References

[1] Steven Blake, David Black, Mark Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss. An architecture for differentiated services. Internet Request For Comments RFC 2475, Internet Engineering Task Force, December 1998.

[2] David D. Clark. The design philosophy of the DARPA internet protocols. In *Proceedings of Special Interest Group on data Communication (*SIGCOMM*)*, number 4 in Computer Communication Review, pages 106–114, Stanford, CA USA, August 1988. ACM, ACM Press.

[3] Constantinos Dovrolis and Parameswaran Ramanathan. A case for relative differentiated services and the proportional differentiation model. IEEE *Network*, 13(5):26–34, September 1999.

[4] Constantinos Dovrolis and Parameswaran Ramanathan. Dynamic class selection: From relative differentiation to absolute qos. In *Proceedings of the International Conference on Network Protocols (*ICNP*)*. IEEE, November 2001.

[5] Azeem Feroz, Amit Rao, and Shivkumar Kalyanaraman. A TCP-friendly traffic marker for IP differentiated services. In *Proceedings of International Workshop on Quality of Service (IWQoS)*, Pittsburgh, PA USA, June 2000. IEEE/IFIP.

[6] Benjamin Gaidioz, Mathieu Goutelle, and Pascale Primet. Implementation of IP proportional differentiation with waiting-time priority and proportional loss rate dropper in linux. Technical Report RR-4511, INRIA, August 2002.

[7] Benjamin Gaidioz and Pascale Primet. The equivalent differentiated services. Technical Report RR-4387, INRIA, February 2002.

[8] Benjamin Gaidioz and Pascale Primet. EDS: A new scalable service differentiation architecture for internet. In *Proceedings of International Symposium on Computer Communication (ISCC)*, pages 777–782, Taormina, Italy, July 2002. IEEE.

[9] Network simulator NS-2 web site. `http://www.isi.edu/nsnam/ns/`.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Futurs : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 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 Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

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