# Amélioration de la QoS du trafic de Cloud-Gaming : une comparaison entre HTB et L4S

Philippe Graff, Xavier Marchal, Thibault Cholez, Stéphane Tuffin, Bertrand Mathieu, Olivier Festor

# *Amélioration de la QoS du trafic de Cloud-Gaming : une comparaison entre HTB et L4S*

Philippe Graff[1] et Xavier Marchal[2] et Thibault Cholez[2] et Bertrand Mathieu[3] et Stéphane Tuffin[3] et Olivier Festor[2]

[1] *University of Waterloo, Canada*
[2] *Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France*
[3] *Orange Innovation, Lannion, France*

Le trafic applicatif généré par les services de Cloud-Gaming (CG) est particulièrement contraignant à transporter car il exige à la fois de très haut *débits* et de très faibles *délais*. La qualité d'expérience (QoE) se dégrade rapidement dès lors que les contraintes de qualité de service (QoS) du réseau ne sont pas satisfaites. En particulier, le partage du lien avec d'autres flux non sensibles à la latence peut aboutir à d'importants délais de file d'attente, plus communément appelé *phénomène du bufferbloat*. Nous évaluons ici deux solutions qui permettent chacune un certain degré d'isolation du trafic CG dans les files d'attente problématiques. La première solution se base sur une politique de file d'attente recourant à des classes de trafic (*Hierarchical Token Buckets*, HTB). La deuxième solution se base sur l'architecture L4S (*Low Latency, Low Loss & Scalable Throughput*) et le gestionnaire de file d'attente (AQM) *DualPI2*.

Nous avons réalisé pour cela une plateforme expérimentale de CG munie de *SCReAM*, un algorithme de contrôle de congestion (CCA) pour RTP compatible avec *L4S*. Nous montrons expérimentalement que ces deux approches permettent de préserver la QoS des flux CG et discutons de leurs contraintes opérationnelles respectives.

**Mots-clefs :** Faible Latence, Qualité de Service, Gestion Active de Files d'Attente, Bufferbloat, L4S, Cloud Gaming

## 1 Introduction

La plupart des équipements réseau mettent en œuvre des files d'attente afin de maximiser la bande passante tout en absorbant les *bursts* de paquets. La taille de ces files est d'autant plus grande dans les réseaux cellulaires qui sont soumis à de fortes variations de capacité. Le remplissage de ces files d'attente, plus communément appelé *bufferbloat* [GN11], induit des délais pouvant affecter certaines applications sensibles à la latence telles que le *Cloud Gaming*. Nous avons précédemment étudié les capacités d'adaptation des quatre principales plateformes de CG et montré que les seuls mécanismes de contrôle de congestion au niveau des applications ne sont pas suffisants pour garantir la QoS du trafic [MGK+23].

Les AQMs traditionnels appliquent une politique de gestion du trafic unique, indifféremment des objectifs de délai et débit propres à chaque flux. Sans moyen de découpler les différents types de trafic au niveau du réseau, la concurrence dans les files d'attente avec les flux régis par les pertes de paquets, plutôt que le délai, conduit inévitablement au *bufferbloat*. Notre objectif est d'évaluer des systèmes permettant la gestion différenciée de trafic entre les flux sensibles à la latence et ceux cherchant uniquement à maximiser le débit.

Les solutions considérées ont recours à des files d'attente distinctes de sorte à isoler les différents types de flux. La première se base sur la discipline de file d'attente *HTB*. Elle nécessite au préalable une classification des flux qui fut traitée dans nos précédents travaux[GMC+23, KGMC23]. La deuxième solution mobilise l'architecture *L4S* et l'AQM DualPI2. Seules deux classes de trafic sont ici considérées, mais elles ne nécessitent pas de classification préalable [ADSB+19]. En utilisant les 2 bits du champ *Explicit Congestion Notification* (ECN) de l'en tête IP, une application peut spécifier si son CCA est *classique* ou *scalable*. Les applications conformes aux exigences de L4S [SB23] entrent dans la seconde catégorie et bénéficient d'une file à faible latence. Bien que prometteuse, la technologie L4S manque d'évaluation basée sur des cas d'usage réalistes, les précédents travaux ayant recourt à la simulation [BJSOC21] ou à TCP Prague, qui

n'est pas adapté aux contraintes de temps réel. Nous avons donc mis au point notre propre plateforme de CG et l'avons munie de *SCReAM* [Joh14], un CCA conforme à L4S pour RTP.

Nous présentons brièvement cette plateforme de CG et notre protocole expérimental en section 2. Nous confirmons expérimentalement le problème du bufferbloat dans la section 3 avant d'évaluer l'apport d'HTB et L4S respectivement dans les sections 4 et 5. Enfin, la section 6 discute des contraintes opérationnelles des deux solutions et conclut cette étude.

## 2 Protocole expérimental

Il existe une précédente plateforme open-source de CG, *Gaming Anywhere* [HCC+14]. Malheureusement, celle-ci utilise TCP qui n'est pas adapté au transport de flux en temps réel, et ne supporte pas L4S. Nous avons donc développé notre propre plateforme de CG (eCGP) [†] pour nos besoins expérimentaux. Notre plateforme est open-source et repose sur la librairie `FFmpeg` pour gérer les flux audio et vidéo, ainsi que sur le CCA *SCReAM* (v2) développé par Ericsson pour RTP. Une description détaillée est disponible dans la version longue de l'article [GMC+24]. En pratique, SCReAM donne une consigne de débit à l'encodeur vidéo en fonction des signes de congestion du réseau qui sont de différentes natures. D'une part, les retours RTCP du client permettent d'estimer les délais et le taux de pertes. D'autre part, le feed-back ECN indique la taille de la file d'attente de l'équipement réseau saturé, ce qui est nécessaire au bon fonctionnement de *L4S*. Le flux CG est relativement élastique et peut varier entre 30Mbps et 10Mbps au prix d'une dégradation visuelle due à une plus forte compression de la vidéo.

Nos expériences consistent à mettre en concurrence deux flux : un flux CG généré par notre plateforme et un flux TCP. Ces flux sont soumis à un goulot d'étranglement au niveau d'un routeur intermédiaire dont nous changeons tour à tour le mécanisme de gestion de la file d'attente. Le flux TCP est généré par `iperf` et régit par l'algorithme *Cubic*, le CCA par défaut de la plupart des systèmes, qui considère uniquement les pertes de paquets. Les caractéristiques des liens entre le routeur et le client/serveur sont définis par des règles `tc-netem`. Nous limitons la capacité du lien descendant à 50 Mbps (celui véhiculant le flux vidéo sujet à congestion) et ajoutons un RTT de 30 ms entre le client et le serveur afin de simuler une latence réaliste pour un WAN. Au bout d'une minute de trafic CG, nous déclenchons le trafic concurrent TCP `iperf` à 10 Mbps. Nous incrémentons ensuite sa consigne de débit toutes les minutes, valant tour à tour 20, 30, 40 et 45 Mbps. Nous relevons trois métriques de QoS qui sont fortement corrélées à l'expérience utilisateur [JSSH11] : le débit, la latence induite par la file d'attente, et enfin le taux de pertes.

## 3 Évaluation d'une simple file droptail

Cette section constitue notre base de référence. Le goulot d'étranglement est alors muni d'une simple file d'attente de type *droptail* dont la taille est fixée à 250 paquets. Une fois la file remplie, les nouveaux paquets sont rejetés.

La Figure 1 trace l'évolution des trois métriques présentées précédemment et met en évidence un **problème de cohabitation entre les flux**. D'une part, le trafic CG s'efface devant le trafic TCP car il réagit plus tôt au signal de congestion basé sur la variation des délais. D'autre part, les délais d'attente sont identiques pour les deux trafics car les flux partagent une seule et même file. Les délais deviennent problématiques en cas de congestion (régulièrement >40 ms, avec des pointes à 100 ms), ce qui nuit à la réactivité du service et à sa QoE. Ceci confirme expérimentalement le **problème du bufferbloat**.

## 4 Évaluation de la discipline de file d'attente HTB

Nous considérons ici deux classes de trafic qui ont chacune leur propre file. La première correspond au trafic *Best Effort* (BE), tandis que la deuxième est associée au trafic CG. La classe CG est prioritaire et bénéficie d'une bande passante minimale garantie que nous fixons à 10 Mbps, correspondant à la recommandation minimale des plateformes commerciales de CG. En ce qui concerne la gestion des files, nous avons recours à deux mécanismes distincts. La classe BE suit une stratégie `pFIFO`. La classe CG utilise le

†. `https://github.com/mosaico-anr/eCGP`

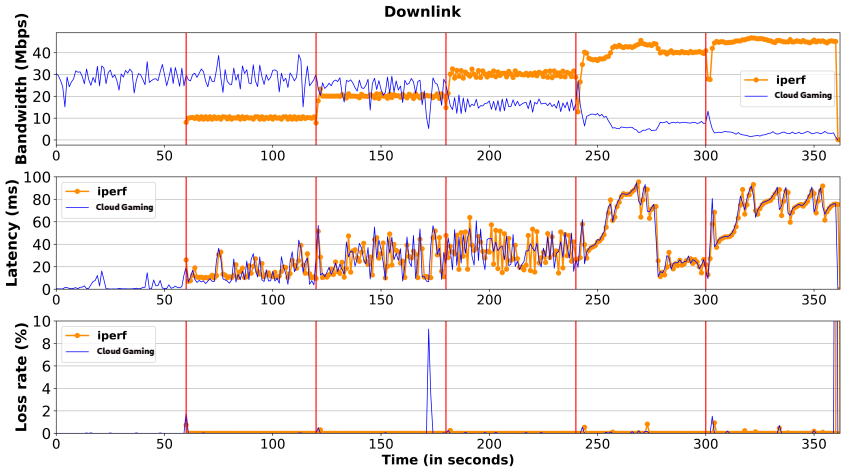


**FIGURE 1:** Trafic Cloud Gaming vs TCP Cubic sur file droptail


**FIGURE 2:** Trafic Cloud Gaming vs TCP Cubic sur HTB

mécanisme SFQ (*Stochastic Fairness Queuing*) garantissant un partage équitable des ressources entre flux. Nous avons limité la taille de la file BE à 250 paquets et celle de la file CG à 63 paquets, d'après [Jan17].

Les mesures rapportées dans la Figure 2 montrent que HTB permet d'améliorer la QoS du trafic CG. D'une part, le débit du trafic CG est préservé et maintenu à $\approx$ 30 Mbps, permettant ainsi une bonne qualité vidéo. La trafic TCP quant à lui plafonne à 20 Mbps, contraint par HTB. La répartition est néanmoins plus équitable qu'avec une file droptail. Surtout, la latence est désormais différenciée et propre à chaque file. Le délai moyen de la file CG se situe à 1.16 ms, ce qui permet un jeu très réactif. Celui de la file BE est bien supérieur à 100 ms, ponctuée de pics pouvant dépasser 150 ms. Ce délai n'est cependant pas préjudiciable car les services de cette classe, par exemple un transfert de fichier, ne sont pas sensibles à la latence.

## 5 Évaluation de l'architecture L4S

Nous déployons ici l'AQM *DualPI2* au niveau du routeur intermédiaire qui gère également deux files : une file *BE* classique et une file *L4S* à faible délai et destinée au trafic compatible L4S, dans notre cas au trafic CG. Toutes deux sont couplées de sorte à promouvoir une certaine équité entre les files. Nous avons gardé les valeurs par défaut des différents paramètres de l'AQM. Les résultats sont donnés dans la Figure 3.

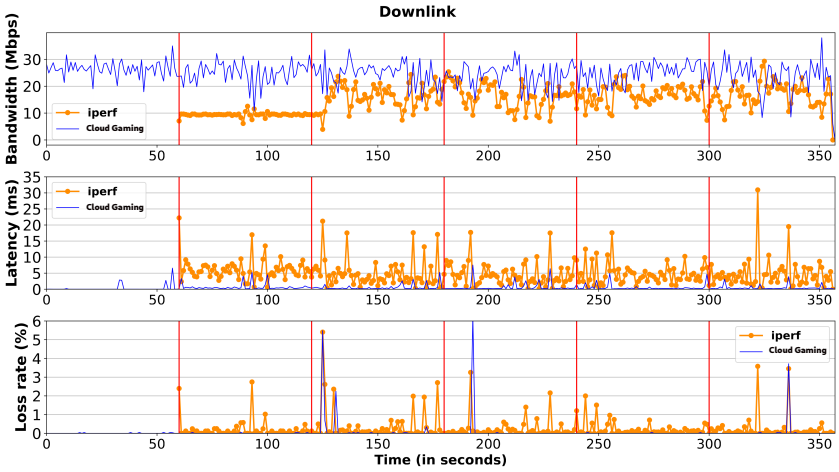


**FIGURE 3:** Trafic Cloud Gaming vs TCP Cubic sur L4S

Globalement, l'utilisation de deux files produit les mêmes effets bénéfiques que HTB sur la QoS du trafic CG, à savoir une faible latence et un débit élevé. On peut toutefois noter quelques différences. Pour commencer, le débit du trafic CG fluctue davantage, impacté par le trafic TCP concurrent car l'AQM *DualPI2* n'a pas de priorité stricte entre les files. Le débit moyen du trafic CG est plus faible, bien que satisfaisant, et se situe à $\approx$ 24.25 Mbps entre les minutes 3 et 6. Le débit moyen du trafic TCP est également inférieur à HTB et $\approx$ 16.4 Mbps. En ce qui concerne l'utilisation du lien, on mesure une baisse de 15% par rapport à HTB (81% contre 96%). Cette observation doit néanmoins être interprétée avec précaution, car dans le cas de HTB, le trafic CG utilise toutes les ressources nécessaires, tandis que le trafic concurrent se contente de la bande passante restante. Ainsi, les deux flux cessent de fluctuer dès qu'ils atteignent leur débit de croisière, ce qui profite à l'utilisation du lien. En revanche, l'absence de priorité dans L4S oblige chaque CCA à réagir aux fluctuations causées par les autres flux. Ces adaptations chroniques impactent le taux d'utilisation du lien. Enfin, le délai moyen du trafic CG est excellent à $\approx$ 0.53 ms et celui du trafic iperf est réduit d'un facteur 20 par rapport à HTB, passant de 99.78 ms à 4.68 ms. Ces résultats s'expliquent par l'utilisation de signaux de congestion proactifs de *L4S* privilégiant la latence au détriment du débit, mais pouvant se configurer.

## 6 Conclusion

Dans cet article, nous avons tout d'abord mis en évidence le problème du *bufferbloat*, c'est à dire les délais induits par les files d'attente dans le réseau, et le problème de cohabitation entre des flux temps réel sensibles au délai et d'autres cherchant uniquement à maximiser leur débit. Nous avons montré que des solutions à double files d'attente comme *HTB* et *L4S* répondent à ce problème et permettent d'améliorer grandement la QoS du trafic CG en cas de congestion.

Finalement, ce qui différencie ces deux solutions sont moins leurs performances que leurs contraintes opérationnelles respectives. En effet, HTB nécessite une classification préalable du trafic, qui est certes possible [GMC+23] mais très contraignante à maintenir dans le temps pour un opérateur afin de prendre en compte l'évolution du trafic Internet. Elle sera aussi sujette à des erreurs inévitables de classification. À l'inverse, l'AQM *DualPI2* est beaucoup plus simple à mettre en œuvre pour un opérateur car il n'y a pas de classe de trafic à configurer, mais elle nécessite en revanche le support des applications dont le CCA doit être compatible avec *L4S* pour bénéficier de la file à faible latence. L'annonce du support de L4S par différents acteurs de l'industrie, tant du côté des applications (Apple, Nvidia) que des télécommunications (Nokia, Ericsson), va néanmoins plutôt dans ce sens. Nos travaux futurs s'intéresseront à un autre mécanisme garantissant une faible latence au trafic, à savoir *Packet Wash* [DC21] qui fait partie du Big Packet Protocol.

## Références


[ADSB+19] Olga Albisser, Koen De Schepper, Bob Briscoe, Olivier Tilmans, and Henrik Steen. DUALPI2 - Low Latency, Low Loss and Scalable (L4S) AQM. In *Proc. Netdev 0x13*, March 2019.

[BJSOC21] Davide Brunello, Ingemar Johansson S, Mustafa Ozger, and Cicek Cavdar. Low latency low loss scalable throughput in 5g networks. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–7, 2021.

[CCG+17] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. BBR : congestion-based congestion control. *Commun. ACM*, 60(2) :58–66, 1 2017.

[DC21] Lijun Dong and Alexander Clemm. High-precision end-to-end latency guarantees using packet wash. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 259–267, 2021.

[GMC+23] Philippe Graff, Xavier Marchal, Thibault Cholez, Bertrand Mathieu, and Olivier Festor. Efficient identification of cloud gaming traffic at the edge. In *NOMS 2023, IEEE/IFIP Network Operations and Management Symposium, Miami, FL, USA, May 8-12, 2023*, pages 1–10. IEEE, 2023.

[GMC+24] Philippe Graff, Xavier Marchal, Thibault Cholez, Bertrand Mathieu, Stéphane Tuffin, and Olivier Festor. Improving Cloud Gaming traffic QoS : a comparison between class-based queuing policy and L4S. In IFIP, editor, *2024 8th Network Traffic Measurement and Analysis Conference (TMA)*, page 10, Dresden, Germany, May 2024. IEEE.

[GN11] Jim Gettys and Kathleen Nichols. Bufferbloat : Dark buffers in the internet : Networks without effective aqm may again be vulnerable to congestion collapse. *Queue*, 9(11) :40–54, 11 2011.

[HCC+14] Chun-Ying Huang, Kuan-Ta Chen, De-Yu Chen, Hwai-Jung Hsu, and Cheng-Hsin Hsu. Gaminganywhere : The first open source cloud gaming system. *ACM Trans. Multimedia Comput. Commun. Appl.*, 10(1s), 1 2014.

[Jan17] Ewa Czeslawa Janczukowicz. *QoS management for WebRTC : loose coupling strategies*. PhD thesis, 2017. Thèse de doctorat dirigée par Bonnin, Jean-Marie Informatique Ecole nationale supérieure Mines-Télécom Atlantique Bretagne Pays de la Loire 2017.

[Joh14] Ingemar Johansson. Self-Clocked Rate Adaptation for Conversational Video in LTE. In *Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop*, CSWS '14, pages 51–56, New York, NY, USA, 2014. Association for Computing Machinery. event-place : Chicago, Illinois, USA.

[JSSH11] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld. An evaluation of QoE in cloud gaming based on subjective tests. In *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 330–335, 2011.

[KGMC23] Joël Roman Ky, Philippe Graff, Bertrand Mathieu, and Thibault Cholez. A hybrid P4/NFV architecture for cloud gaming traffic detection with unsupervised ML. In *IEEE Symposium on Computers and Communications, ISCC 2023, Gammarth, Tunisia, July 9-12, 2023*, pages 733–738. IEEE, 2023.

[MGK+23] Xavier Marchal, Philippe Graff, Joël Roman Ky, Thibault Cholez, Stéphane Tuffin, Bertrand Mathieu, and Olivier Festor. An analysis of cloud gaming platforms behaviour under synthetic network constraints and real cellular networks conditions. *J. Netw. Syst. Manag.*, 31(2) :39, 2023.

[SB23] Koen De Schepper and Bob Briscoe. The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S). RFC 9331, January 2023.