An Analysis of Cloud Gaming Platforms Behaviour Under Synthetic Network Constraints and Real Cellular Networks Conditions

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

An Analysis of Cloud Gaming Platforms behaviour under Synthetic Network Constraints and Real Cellular Networks Conditions

Xavier Marchal[1], Philippe Graff[1], Joël Roman Ky[2], Thibault Cholez*[1], Stéphane Tuffin[2], Bertrand Mathieu[2], Olivier Festor[1]

[1] Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
firstname.lastname@loria.fr
[2] Orange Innovation, Lannion, France
{joelroman.ky; stephane.tuffin; bertrand2.mathieu}@orange.com

Abstract. With the recent technological evolutions in networks and increased deployment of multi-tier clouds, cloud gaming (CG) is gaining renewed interest and is expected to become a major Internet service in the upcoming years. Many companies have launched powerful platforms such as Google Stadia, Nvidia GeForce Now, Microsoft xCloud, Sony PlayStation Now, among others, to attract players. However, for all end-users to fully enjoy their gaming sessions over the wide range of network access qualities, CG platforms must adapt their traffic.
In this paper, we present the outcomes of a comprehensive measurement study performed on the four aforementioned CG platforms, configuring different synthetic network constraints like packet loss, throughput decrease, latency increase and jitter variation to observe the traffic of these CG platforms under degraded network conditions and infer their adaptive behaviour. We also present how the four CG platforms behave when used under real cellular network conditions, captured on the Orange network in January 2022.
Our findings show that the four platforms exhibit different adaptation behaviours. Moreover, many cases result in a degraded QoS, leaving room for further improvements at both application and/or network levels.

Keywords: Cloud Gaming · Traffic Analysis · Congestion Control · Mobile Network

Acknowledgment

This work is partially funded by the French National Research Agency (ANR) MOSAICO project, under grant No ANR-19-CE25-0012.

1 Introduction

The concept of cloud gaming is to run a game on a remote server. User commands pass through the network to reach the server which runs the game application

and streams back the resulting audio and video flows. Although already investigated a decade ago [1]–[4], cloud gaming, at that time, failed to reach its audience partly because of the low capabilities of devices, limited network capacity and sparse Cloud deployment. With the recent technological evolution in these fields and the increased deployment of multi-tier clouds, cloud gaming (CG) is gaining renewed interest and a lot of attention. CG is expected to become a core service in the coming years. Its global market is forecasted to rise over three billion gamers by 2023. This trend has also been caught by major providers who launched powerful platforms to attract game players. CG providers include those who enrich their existing game consoles platforms with CG platforms as well as Cloud providers moving into this new market. Big industry names in this space include Nvidia Geforce Now[3], Google Stadia[4], Sony PlayStation Now[5], Microsoft Xbox Cloud Gaming platform[6] (formerly known as xCloud), Amazon Luna[7], etc. There is a strong expectation about CG to deliver the quality allowing end-users to fully enjoy their gaming sessions. Not waiting for network operators to generalize high-throughput low-latency networks, CG providers implement their own application-level mechanisms to cope with varying network conditions, such as latency compensation techniques called "negative latency" by Google or "Outatime" [5] by Microsoft. These mechanisms together with congestion control techniques aspire to use available network capacities to jointly optimize the perceived network latency and the delivered video quality, which are tightly linked to players' QoE as demonstrated in [3] and [6]. Some works have focused on a specific platform [7]–[10] but ours is the first to compare several platforms altogether under the same degraded and real cellular network conditions to evaluate their capacity to adapt their traffic and maintain the service.

In this paper, we present the evaluation made from our measurements of the four main commercial CG platforms to date, namely Nvidia GeForce Now (GFN), Google Stadia (STD), Sony Playstation Now (PSN) and Microsoft Xbox Cloud Gaming (XC), using either native or web clients. A first set of measurements was performed between April and July 2021, testing the CG platforms' adaptation in front of synthetic network constraints. The considered types of network degradation cover packet loss, throughput decrease, latency increase and jitter variation. The constraints are applied in two ways: 1) before the start of the gaming session, or 2) in the course of a gaming session to observe how CG platforms mitigate and recover from sudden variations. A second set of measurements was performed in January 2022, evaluating the adaption of the four CG platforms when used in real cellular network conditions. Our evaluation reveals the limits of the four CG platforms to manage network degradation and highlights their numerous differences in managing them.

[3] `https://play.geforcenow.com/mall/#/layout/games`
[4] `https://stadia.google.com/`
[5] `https://www.playstation.com/en-us/ps-now/`
[6] `https://www.xbox.com/en-US/play`
[7] `https://www.amazon.com/luna/landing-page`

The remainder of this paper is organized as follows: Section 2 presents the related work and Section 3 gives more background on the four considered platforms. Section 4 and 5 respectively present our evaluation under initial degraded network conditions and with degradation occurring during the gaming session. Section 6 evaluates the four platforms' behaviour over cellular network conditions. Section 7 presents additional measurements to make the link between the measured bitrates and the QoE. Finally, section 8 concludes this paper.

2 Related Work

Our work is related to previous studies considering Cloud Gaming from the architectural, network, application or QoE perspectives, but also to studies considering other low-latency services transported over cellular networks.

2.1 Cloud Gaming related studies

The protocols used by CG platforms were studied in several works. The authors of [8] study the OnLive (OL) platform and identify several RTP (Real-time Transport Protocol) streams, multiplexed over a single UDP port. They also point out that video frame splitting is performed at the application layer. In [9], the authors study STD. They notice the presence of several groups of packets in downstream RTP traffic. Each of these groups appears every $\approx$ 16.67ms. This corresponds to the duration of one frame at 60 fps. The groups result from fragmentation caused by the maximum packet size allowed by the network. More recently, in [7], Di Domenico & al. performed a comprehensive protocol analysis of three recent CG platforms: Stadia, GeForce Now and PSNow. They discovered that STD and GFN rely on RTP to stream their audio and video, whereas PSN uses a proprietary protocol with several sub-channels. They discover that STD strictly applies the webRTC standard. While STD and PSN combine several flows on a single UDP port, GFN uses several ports. Among other interesting findings, they point out that some flows are dedicated to re-transmissions and that STD and GFN can stream data up to 44 Mbit/s.

Other studies like [11] and [12] investigated sub-delays making up the whole '*Response Delay*' defined as the delay between a game command and the display of the resulting frame on the player's screen. It would be decomposed in 4 sub-delays, namely: the '*Network Delay*', the '*Game Delay*', the '*Processing Delay*', and the '*Playout Delay*'. The last two are mainly about the frame encoding and decoding times. They proposed a methodology to estimate each sub-delay and showed that in 2013 both OL and StreamMyGame (SMG) CG platforms, suffered from prohibitive delays: over 100ms Processing Delay due to the lack of hardware encoding and over 100ms Game Delay for the most demanding games. [12] renames the '*Response Delay*' as '*Interactive Delay (ID)*'. Based on the RTT and the local ID, the authors estimated the streaming delay (*encoding, decoding, etc.*) to be over 100ms for OL.

Several papers [3], [13], [14] evaluated the impact of the types of games on the network and their sensitivity to network disturbances. The authors of [3] defined game types depending on their perspectives. They distinguished fast-paced games (*first person*), medium-paced games (*third person*), and slow-paced-games (*omnipresent*). They showed that fast-paced games are more sensitive to latency, but more resilient to packet loss. Moreover, the '*service to client*' direction is the most sensitive to disturbances. In this sense, the quality of experience (QoE) is the most impacted by downward packet losses and downward delays. In [13], the authors looked at the OL platform. The three types of games have roughly the same characteristics. However, *omnipresent* games have the lowest bit rate and *first-person* games have higher upload rate. In [14], the authors used frame metrics which allowed the extraction of spatial (SI) and temporal (TI) information reflecting respectively game scene complexity and the pace of changes on the screen. Not surprisingly, they noted that action games and shooters have the highest temporal information. However, at that time, they did not find a relationship between frame metrics and network metrics (*like the throughput*).

Several studies have considered the QoE and how certain QoS factors can impact it. In [3], Jarschel & al. use 58 participants to conduct their study. They are asked to play a game and then to evaluate their gaming experience, giving a score between 1 and 5. The authors then quantify the impact of QoS factors on QoE. The two most impactful factors are server-to-client packet loss and server-to-client delays. These results are consistent with what is shown in [15]. Here, Sviridov & al. assume that QoE is correlated with the gamer's playing capabilities. They then rely on Artificial Intelligence (AI) agents to automatically assess QoE. According to them, latency is the most impacting factor. On the other hand, they point out that client-to-server packet loss is not that important. The players repeat their unperformed actions, which in effect constitutes implicit Forward Error Correction (FEC). In [16], Ammar & al. want to identify performance indicators to detect a QoE decline. Their study applies to WebRTC-based videoconferencing. To achieve their goal, they rely on the statistics of the *WebRTC-internal* tool. They show that critical QoE phases correlate well with throughput decreases. In this sense, Zadtoota & al., want to model QoE based on bitrate and framerate [17]. Their study deals with Cloud Gaming QoE. When comparing the predicted QoE with what was reported by the study participants, the authors obtain a Root Mean Square Error of 0.58.

The adaptation of old CG platforms to network conditions were studied in [11] and [13], while [9] and [10] were respectively focused on STD and GFN. Chen & al. [11] quantified streaming quality under constrained network conditions (*on bandwidth, delay, loss*). They showed that OL and SMG handled well delays, but SMG could not handle the loss rate nor the bandwidth limit. Similarly, in [13], the authors proved that the frame rate of OL is decreased when losses were too high or the available bandwidth too low (from 60 to 25fps). They underlined the bit rate did not seem to be impacted by packet loss and latency. In [9], it appeared that STD immediately decreases its resolution when a constraint is applied during a game session. It then enters a transition phase, where it tries to

adjust its parameters to the network. This phase results in low QoE, as resolution and frame rate greatly fluctuate. According to [7], 720p resolution comes with an average bit rate of 11 Mbps (29 Mbps for 1080p and 44 Mbps for 2160p). In [10], GFN traffic was disturbed by adding latency, jitter, and loss. They noted GFN favors frame rate over resolution when network resources run out. This choice ensures streaming smoothness at the expense of quality.

2.2 Low latency services over cellular networks

Adding network disruptions in a synthetic way makes it easier to study the behavior of CG platforms. However, synthetic conditions are not representative of highly variable cellular network conditions. In [18], the authors point out that it is practically impossible to predict the capacity of a 3G cell. In addition, in [19], the authors observe that for a same radio access point (3G-DCH or LTE), there are large variations in the RTTs. When the nodes are in motion, the maximum RTTs can even reach a few seconds.
To cope with variations in network capacity, the buffers are oversized. In [20], the *bufferbloat* phenomenon is thus mentioned. It consists in a large accumulation of packets in the buffers. The problem is that filling these buffers induces latency, and that loss-based control congestion algorithms (CCAs) will only detect congestion when the buffer is full. CCA's reaction to congestion will thus be triggered very late. The study of CCAs for mobile networks is an area well covered in the literature. A survey [21] has been published on this subject. In [22], a TCP-based congestion control algorithm for cellular networks is presented. The algorithm takes into account the evolution of the RTT to adjust the TCP congestion window size. Moreover, the sensitivity of the algorithm can be adapted to the requirements of each application. In [23], Yan & al. present a distributed and collaborative system to evaluate Internet transport algorithms. Their contribution also allows to calibrate network emulators so that they simulate a certain network path as well as possible. In a second step, the authors use previously calibrated emulators to train a CCA based on neural networks (*Indigo*).
As far as we know, no work has yet been done on the adaptation of CG platforms to cellular networks. One contribution [24] studied video calling applications over WiFi and cellular networks. The authors have shown that QoE is undermined by bursty packet loss and long packet delays. Furthermore, they recommend having a stable video rate rather than Skype's overly aggressive FEC scheme increasing the bitrate as packet-loss rate increases which is counterproductive when losses are caused by self-inflicted congestion. Although video conferencing has a real time character, we cannot generalize these results to Cloud Gaming which has more stringent latency constraints, much higher bitrates in the downstream direction and has access to powerful Cloud resources.

When considering CG platforms studies on cellular networks or mobile devices, few works stand out. Wang et al. [25] present a model to measure Mobile Gaming User Experience (MGUE) with Game Mean Opinion Score (GMOS). This model depends on objective factors that are measured in real-time during

game session like response time (end-to-end delay), packet loss, video PSNR, initial video configuration and game genres. The model was used over a commercial LTE network in different conditions (indoor, outdoor, mobility) and showed that it was difficult to get high MGUE values due to the high response time in wireless networks. Kämäräinen et al. [26] aim to characterize the latency encountered in mobile Cloud Gaming and study each component of this overall latency to show if it is possible to reach an imperceptible latency for users. Their work, which pinpoints latency in mobile device as the main source of delays, also points out the impact of access network, in this case of LTE, on the overall latency. Other works focus on Virtual Reality (VR) gaming using cellular networks. Tan et al. [27] pinpoint the bottlenecks of medium quality mobile VR. Their work shows that cellular network bandwidth is not the main issue but LTE's signalling operations between the UE and the base station significantly affect mobile VR latency no matter the signal strength nor the competing traffic. These signalling operations involve error corrections, handover procedure and grant-based resource allocation in the uplink. They designed a client-side solution, LTE-VR, that reduces unnecessary latency due to LTE's signalling operations. Their solution reduces network latency by 5ms in average and up to 70ms. Zhang et al. [28] study the impact of congestion with different bearer settings and scheduling algorithms on VR video traffic over 4G/LTE networks. By setting up a testbed with a lightweight Cloud VR application, they show that network latency consists in 77% of total latency. Congestion and competing traffic scenarios increase latency up to 56% while using high-priority dedicated bearers in unacknowledged mode, can reduce latency in the radio segment but at the cost of sacrificing reliability, causing TCP retransmissions increasing the end-to-end latency.

Our work differs from the related work in that we compare the behaviour of 4 modern CG platforms under initial and sudden network constraints and real cellular network conditions. We will show that the results of studies focused on a single platform cannot be generalized because each one reacts differently. We propose the most comprehensive study to date regarding the number of CG platforms considered and the variety of network conditions applied, with a focus on highlighting the differences between platforms to infer their behaviour in adjusting their traffic.

3 Platforms' specificities

We consider four of the most famous CG platforms to date[8]. Other solutions where users must operate both client and server leverage similar technologies. They were initially proposed as "in home" game streaming solutions (i.e. to stream the video on a connected TV or smartphone from a game executed on a local computer). We can name for instance: Steam Remote Play, Parsec, Moonlight, Rainway, among others. While initially designed to be executed over a LAN, most of them now support streaming over the Internet. However, the lack

[8] Amazon Luna would have been another interesting candidate, but it is not yet officially released to date and not accessible in European countries.

of fixed infrastructure makes them hard to compare fairly against full-fleged platforms.

Despite providing a similar service, each platform has specificities that may impact its network traffic behaviour. Regarding the hardware, GFN and STD use x86 servers with high-end GPUs and execute a PC version of games, thus allowing more user control on input peripherals and graphic options, like enabling High Dynamic Range on compatible games. Nonetheless, their philosophy differs: GFN executes unmodified versions of games from the user own digital library while STD executes a specific optimized version that must be bought on the platform. XC and PSN execute the console version of games on console hardware, respectively Xbox One S, PS3 or PS4 which are currently being upgraded to Xbox Series X and PS5 hardware to support the new generation of games and to offer better performance for older games through retro-compatibility. Both CG platforms offer an always-changing game catalogue. Other specificities lie in supported video codecs and resolutions. All platforms use H264 except for STD that also supports VP9. STD is also the only platform to achieve 4K resolution but only with VP9. STD and GFN can run all games at 1080p60fps while XC and PSN were still limited at 720p30fps at the time of our first synthetic experiments (sections 4 and 5). They can now achieve 1080p60fps depending on games thanks to their hardware upgrade. Whatever the platform, their core algorithms being undocumented, our tests only allow us to make conjectures about their adaptation mechanisms.

4 Cloud Gaming traffic under initial network constraints

4.1 Experimental conditions

To conduct our study, we collected network captures between April and July 2021 using a commercial FTTH (Fiber To The Home) subscription of 10 Gbps. All the involved game servers were located in Europe according to latency measurements or the GeoIP database.

CG traffic goes through a computer acting as a *bridge* between the CG client and the Home Gateway. Network disruptions are created on this computer using '`tc-netem qdisc`' rules. Disruptions are only applied in the 'service to client' direction as it is known to be the most critical for the QoS [3] and demanding for the network. They are listed in Table 1. Traffic is captured on a *Windows10* CG client with *Wireshark* (*v 3.4.6*). PSNow (*v 11.7.0*) and GeForce Now (*v 2.0.32.95*) fat clients were used on this machine while Stadia and XCloud were played through the Chrome browser. For our measurements, we separated the "*service to client*" and "*client to service*" traffic directions. For each of them, we measured the bitrate, packet inter-arrival time (IAT) and payload size.

In this section, we study how the four cloud gaming services react on poor initial network conditions. More precisely, we study the effects of the following network characteristics that can affect user experience.

– The packet loss rate, ranging from 5 to 20%;

- The capacity of the link, ranging from 20 to 5Mbps;
- The latency between the service and the client, ranging from 20 to 100ms;
- The jitter, ranging from 5 to 20ms and following a normal distribution.

Packet loss and jitter values might appear very pessimistic or even unrealistic nowadays while bandwidth and latency values represent poor DSL connections. However, these values were chosen on purpose to push the platforms to their limits.

Table 1: Parameters used to degrade network quality

	Moderate	High	Extreme	Reference
Loss	5%	10%	20%	0%
Bandwidth	20 Mbps	10 Mbps	5 Mbps	10Gbps
Latency	20 ms	50 ms	100 ms	8 ms
Jitter	5 ms	10 ms	20 ms	0 ms

As shown in Table 1, each of these parameters are tiered based on their intensity (Moderate, High and Extreme) and their naming represents how much they impact the network quality. It is impossible to have the very same game executed on the four platforms because of irreconcilable catalogues but we chose the same type of racing game to have similar screen dynamics. The captures start at the very beginning of each gaming session. We consider the time interval between 200 and 500 seconds as we observed that each service has stabilized its throughput after 200s. For each network capture, we compute and analyse the evolution of three aforementioned metrics over time: bitrate, IAT and payload size. Given the very large number of curves produced by our measurement campaign[9] , all results cannot be presented due to space limitation. Instead, we focus on the most interesting behaviours and primarily on server-to-client bit rate variations. We discuss the other two metrics when necessary and at times we comment on the upstream direction. We released the full dataset for the community[10].

4.2 Overview of the results

On Figure 1, we represent the bit rate produced by each platform (from left to right : GFN, STD, PSN, XC) according to bandwidth limitations. In the legend, "*ratx*" stands for "available bandwidth limited to 'x' Mbps". We see that GFN reduces its bit rate by more than 50% (from 33Mbps to 16Mbps) when we limit the available bandwidth to 20Mbps. It successfully adapts to lower bandwidth capacities. We notice that it uses only around 80% of the maximum link capacity.

[9] We drew and analyzed 96 plots in total: 4 platforms * 4 network constraints * 3 reported metrics * 2 directions.

[10] Full dataset repository (120Go): `https://cloud-gaming-traces.lhs.loria.fr/index.html`

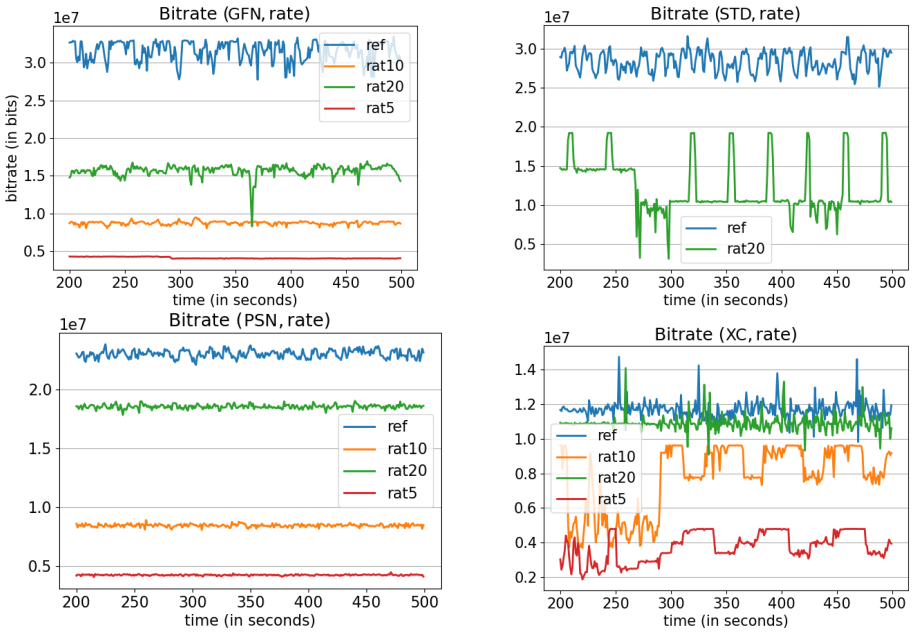


Fig. 1: GFN, STD, PSN and XC bitrate over time for different rate limitations (service⇒client)

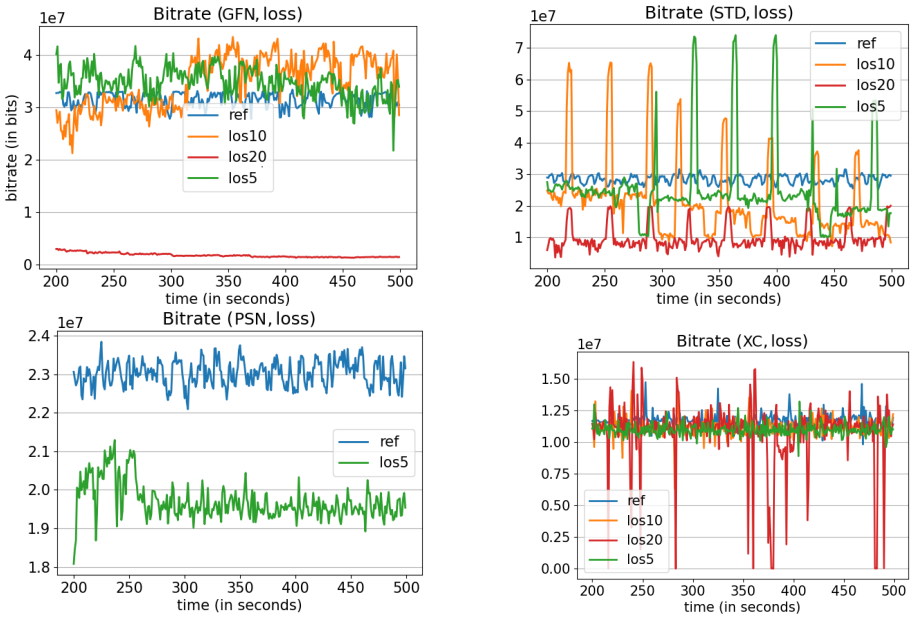


Fig. 2: GFN, STD, PSN and XC bitrate over time for different loss rates (service⇒client)

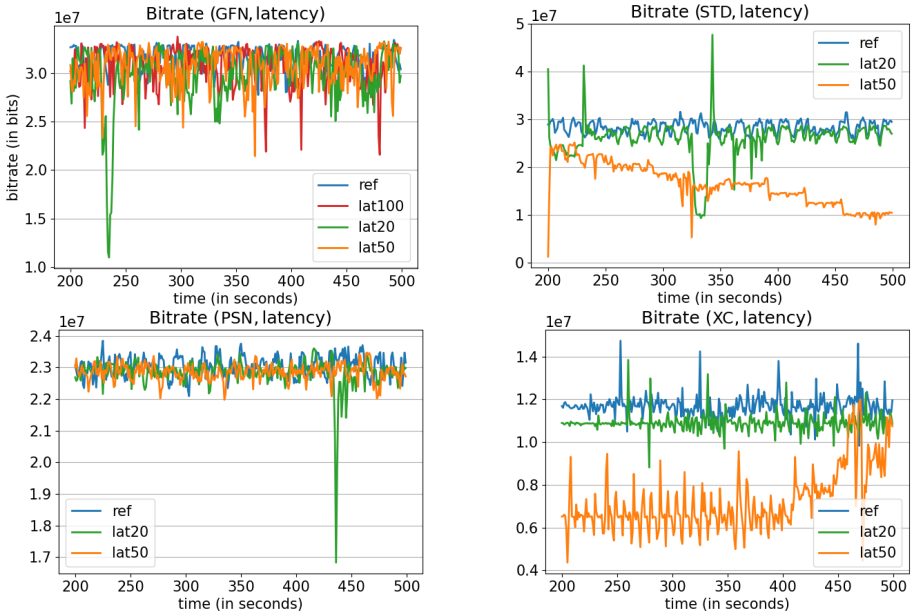


Fig. 3: GFN, STD, PSN and XC bitrate over time for different added latency values (service⇒client)

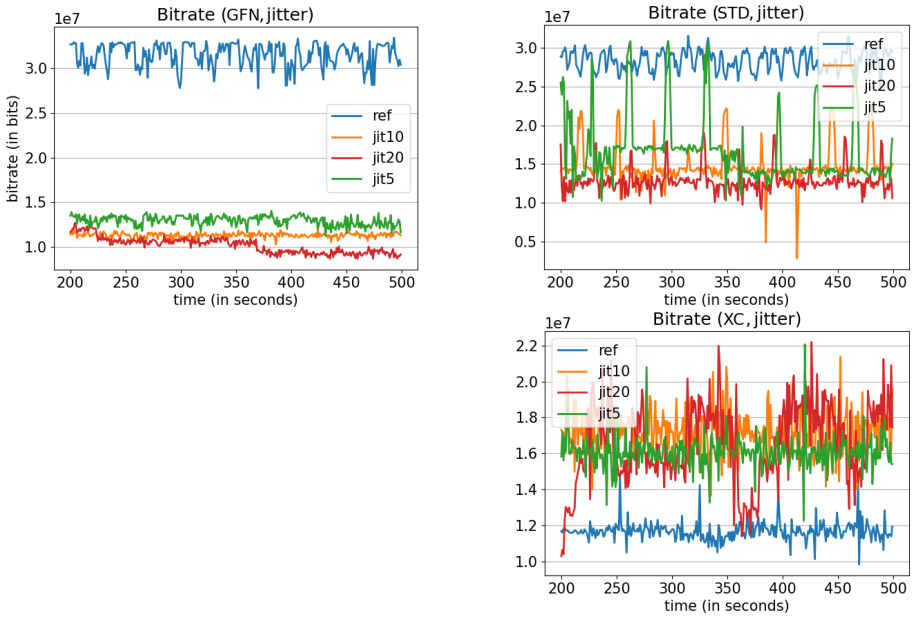


Fig. 4: GFN, STD, PSN and XC bitrate over time for different jitter values (service⇒client)

It seems to indicate that some margin is kept in case of bursts to avoid unneeded jitter and ensure a stable throughput. For STD, the first remarkable result is that it is not possible to complete a game session with a bit rate below 10Mbps at the time of the experiment. This makes it the most demanding platform regarding the bandwidth requirements. When the constraint is set to 20Mbps, STD does not choose to maximize the utilization of the link capacity. It stabilizes a bit above 10Mbps (around 50% of the link capacity) and exhibits periodical burst spikes to probe the bandwidth. Note that around the 260th second, the platform changes the resolution from 1080p to 720p. This is accompanied by a decrease in bit rate (from around 15 to 10Mbps). PSN shows a very stable bit rate adapted to each constraint, and uses around 90% of the available bandwidth.XC does not need to adapt to the 20Mbps constraint. Its throughput in normal conditions is indeed in the 12Mbps range. For the 10 and 5Mbps limitation, XC exhibits a specific behaviour. It repeatedly increases its bitrate until it reaches the limit, before lowering it abruptly after a few seconds. The bandwidth usage oscillates between 95% and 80%. This behaviour suggests that XC does not handle bandwidth constraints well. It must periodically readjust its throughput, affecting user experience.

Figure 2 shows the bit rate of the four platforms under several loss rates. In our convention, "*losx*" stands for "x" % of packet losses in addition to the reference "*ref*". Regarding GFN, the curves are roughly similar for 5 and 10% of packet loss. We see a slight bit rate increase (through the packet rate). We conjecture it is due to the addition of Forward Error Correction (FEC) which would be inhibited in normal conditions. The behaviour is totally different for 20% of packet loss as the bit rate is reduced by more than 90%. The QoE is highly impacted, as the video quality is at its lowest level (540p30fps). STD manages to adapt to loss by reducing the bit rate in proportion to the loss rate. We conjecture STD assumes loss are caused by self-inflicted congestion and reduces its bitrate in the aim of controlling congestion. The curves show periodic peaks occurring about every 40 seconds. That is the way STD probes for available bandwidth in case of network issue. With no bandwidth restriction on this experiment, the spikes are taller than in Figure 1, with observed burst reaching two or three times the average bitrate with more than 70Mbps. We can see peaks are more pronounced for smaller losses. PSN is far less resilient to losses. For any loss rate exceeding 5%, it refuses to launch a game instance. With a loss rate of 5% the average bit rate slightly drops to 20Mbps versus 23Mbps with no added loss. These bit rates similarities hide a reaction from PSN. Indeed, packet sizes and IAT significantly increase (+39.6% IAT on average) which is certainly due to the addition of error correction within packets. There is not much to say about XC except that it does not seem to (or need to) adapt to losses. The QoE is not impacted despite high loss rates. So we can conjecture XC constantly uses high redundancy to withstand high loss rates.

Figure 3 shows the bitrate of the four platforms when we add network delays. "latx" stands for an addition of "x" ms of latency in the server to client direction. GFN does not react much to latency. We observe a slight increase in bit rate

variance, but the mean is rather constant. The huge spike on the curve associated with 20ms of added latency is due to an artefact of gameplay. GFN is also the only platform to operate despite of 100ms of latency. STD's behaviour varies depending on the amount of added latency. 20 ms of added latency does not trigger any significant reaction. At 50ms, STD gradually reduces its bitrate from 30Mbps to 10Mbps, the same floor as in Figure 1. Strangely, STD does not show the spiky pattern like in the other experiments. When adding 100ms latency, STD agrees to launch a game instance, but threatens to shut it down if network conditions do not improve in the next 30 seconds. We conjecture STD assumes that high latencies are caused by self-inflicted congestion and reduces its bitrate to control it, excluding the eventuality of a high base latency. PSN does not take any action against latency but refuses to launch an instance with 100ms of added latency. XC slightly reduces the bitrate for 20ms of added latency. At 50ms the service shows an unstable bitrate with a heartbeat pattern around 7Mbps, mainly due to a high variance in packets IAT. The bit rate still slowly increases towards the end of the capture until it reaches its regular value. Like STD and PSN, 100ms is beyond its capacity to operate the service.

Finally, Figure 4 shows the bitrate of the four platforms with added jitter. The reference curve represents the evolution of the throughput without added jitter. A jitter of "x" ms is represented by the "*jitx*" curve. GFN's response to jitter is very important. For the lowest added jitter value, the bit rate is reduced by 20Mbps to reach 13Mbps. The bit rate is further reduced to 9Mbps for 20ms of added jitter thanks to a change of resolution to the most degraded mode at 375 seconds. This could be due to a small playout buffer translating high jitter to packet loss. STD also strongly reacts to jitter with a bit rate reduced by at least 10Mbps for 5ms of added jitter. The bit rate further drops by 5Mbps when we add 20ms jitter. It is then within 13Mbps. There is no data for PSN that failed to finish any capture even with the lowest jitter value. Since PSN has shown high sensitivity to packet loss, this may also be caused by a small playout buffer translating jitter into loss. XC totally differs from the others in that it increases its throughput as soon jitter is added. At the first jitter value, the platform increases its throughput by more than 30% by sending more packets per second. We assume that it is due to the retransmission of out of order packets caused by jitter. Same as for latency, a high jitter tends to increase the variance of the stream's bit rate.

Figure 5 displays our results in two dimensions with the average UDP payload size on the X-axis and the average IAT on the Y-axis. Each mark is associated with a certain platform, for a certain constraint. The goal is to highlight notable differences in the adaptation strategies of the different platforms. Three of the four services: GFN, STD and XC tend to drift to the upper left corner, but they have their own way to do it with different trend functions. GFN shows the wider span on the two dimensions which reflects a broader adaptation range. On the other hand, PSN is the sole to be in the upper right corner because of an increase of payload sizes in certain conditions. XC has a few points in the lower right corner because of decreased IAT under jitter.

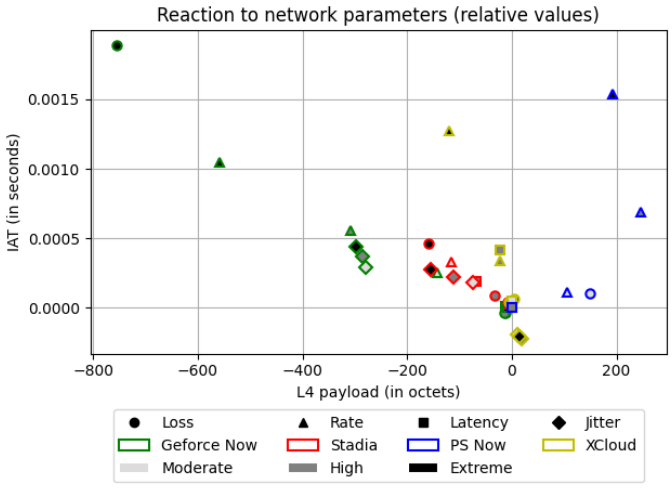


Fig. 5: Relative 2D space for mean UDP payload and IAT (service⇒client)

5 Cloud Gaming traffic under sudden network constraints

The previous section deals with network parameters applied before a session start, so the service has a chance to adapt its traffic from the start of the gaming session as a result of the different probing tests performed beforehand and documented in [7]. Here, we want to know how the four CG platforms react when we suddenly change the network conditions during a game session. Like before, we will artificially alter one of the four parameters: loss, rate, latency or jitter, with respectively 5%, 15Mbits, 50ms and 2ms. These limitations will be applied after 120 seconds of gameplay and will last 120 seconds. Then, another 120 seconds period is captured to see how the service recovers when the network conditions return to normal. Figure 6 shows the variations of the server-to-client bitrate of the CG platforms for the four parameters.

The first subfigure 6 shows the bit rate limitation to 15Mbps. XC nominal bit rate being inferior to the constraint, it does not need to adapt at all. But we couldn't choose a lower limitation since we previously noticed that Stadia explicitly refuses to maintain a session with a bit rate under 10Mbps. STD does not decrease much right after the constraint is set but the service has a hard time to stabilize with around 80 second of crenelated bit rate (with a ramping pattern on some of them). Afterward, the bit rate becomes more stable with a good usage of the available bandwidth, but it decreases a bit before the constraint is removed which hints the service was still trying to stabilize. Then, STD needs some time to recover (around 10s) with a huge spike followed by the exact same bit rate we recorded before the constraint was applied. This spike is a pattern regularly seen on STD and probably triggered by the congestion control algorithm to quickly evaluate the available bandwidth. For PSN, we can see a huge decrease of the

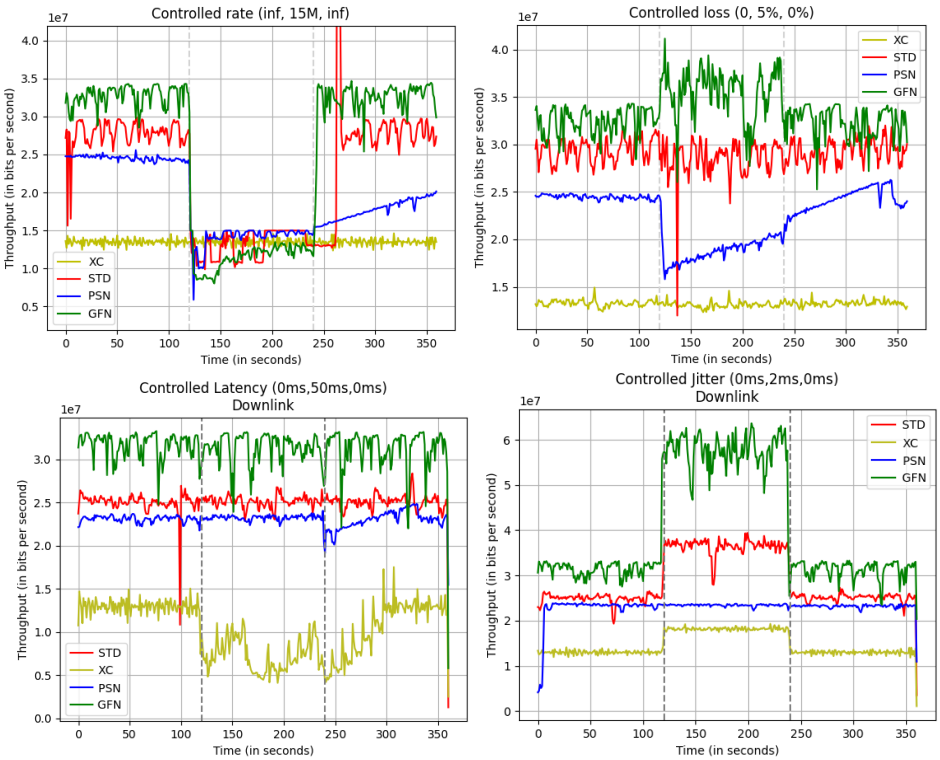


Fig. 6: Server-to-client throughput of cloud gaming platforms under sudden network constraints

bit rate the first second the constraint is applied (resulting in a stalled video) certainly due to packet drops in the queue of our computer and inefficiencies in PSN congestion control. Then, it recovers quite well in a single notch and a stable bit rate. Afterward, PSN exhibits a slope-like recovery reaching the same level as before the constraint was applied but it takes longer that the 120s period displayed on the graph. At last, GFN's initial reaction is like the other two with a quick reduction of the bitrate, then it keeps this bit rate some time before gradually improving but without fully using the available bandwidth like STD and PSN. After the constraint is removed, GFN exhibits a very fast recovery (between 1 and 2 seconds).

The second subfigure 6 shows the bitrate over time when a 5% loss rate is applied. We can see that each service has its own way to handle loss. GFN increases its bitrate probably due to additional FEC or retransmissions, PSN shows a sudden drop followed by a very slow ramping increase that grows in strength when the network conditions are back to normal. XC and STD do not adapt their traffic at all, probably meaning that their traffic already includes enough FEC to withstand a 5% loss rate.

The third subfigure 6 shows the reaction to 50ms of added latency. We can see that XC instantly decreases its bit rate up to almost three times at the lowest point, from 14Mbps to 5Mbps, and it never reaches a steady state as long as the perturbation lasts. Even worse, it takes one more minute after the constraint removal to restore the initial throughput. From a QoE perspective, the gaming experience was the most unpleasant on this platform under the above conditions, the game being unresponsive to the controls. Strangely, PSN seems to reduce its sending rate once the disturbance has stopped, and it takes about 60s to get back to its normal rate. GFN and STD, on the opposite, do not adapt their bit rate to the added latency.

The last subfigure 6 shows the reaction to 2ms of added jitter. Three of the four platforms, STD, GFN and XC significantly increase their bitrate under jitter. GFN reacts the most, doubling the bitrate under jitter up to 60Mps, STD goes from 25 to 37 Mbps (+66%). The most probable explanation is that jitter creates many out-of-order packets that must be retransmitted if the reception buffer is not large enough to allow packet reordering. Those three platforms also quickly come back to their initial rate as soon as jitter is suppressed. PSN, on the other hand, doesn't seem to react at all to the jitter. Regarding the client-to server traffic, similar behaviours can be observed for STD and PSN: STD increases its traffic in the same proportions in this direction while PSN does not react. GFN only exhibits a minor increase of its bit rate contrary to the major one in the opposite direction. XC is the most affected platform with client-to-server traffic multiplied by three under jitter, from 420Kbps to 1200Kbps. We conjecture these differences stem from the usage of different reliability modes at the transport layer for game controls. For example, it is possible to configure WebRTC's datachannels in reliable in-order delivery mode which is using retransmission or in unreliable mode which is limiting the number of retransmissions or the time during which retransmissions are allowed.

In conclusion, all platforms try to adapt their traffic at some point when one or another constraint is applied, reducing their bandwidth usage by adjusting a combination of video resolution, frames per second and compression level (which impact on QoE will be further analyzed in section 7.1), or increasing it to retransmit packets or include additional FEC. We can notice that all platforms do not react in the same way to the same constraints and all constraints do not have the same impact on the delivered bit rate. Please note that the lack of reaction does not necessarily mean that the quality of experience is not affected. For instance, PSN not reacting to latency or jitter leads to a degraded service. In several experiments the bit rate keeps being adjusted when a constraint is applied, and sometimes even after its release, further altering the QoE.

6 Cloud Gaming traffic over cellular networks

The two previous sections apply synthetic network constraints to study the behaviour of the different Cloud Gaming platforms. In this section, we study their behaviour when they are accessed through a mobile network. As we aim at con-

trasting the behaviours of STD, GFN, PSN and XC under the exact same mobile network conditions, we set up an emulated network environment fed with real traces from cellular networks thanks to Mahimahi[11], a network emulation tool developed by the Massachusetts Institute of Technology (MIT). Mahimahi allows to replay the timing and the amount of transmission opportunities provided by a mobile network as captured during a measurement campaign. Unfortunately, in Mahimahi's public project repository, captured mobile network transmission opportunities are old (2016[12]). To overcome this lack of recent and representative captures, we carried out several measurements over Orange France production network in January 2022 and captured the transmission opportunities (txops) under 6 distinct conditions. We released those traces on their own and in addition to the Cloud Gaming traffic we captured through them[13].

6.1 Experimental setup and data collection

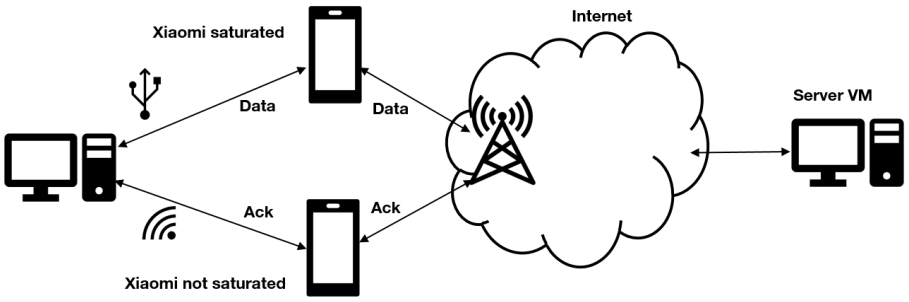


Fig. 7: Experiment setup for capturing txops

Collection of transmission opportunities on Orange's cellular network To capture the transmission opportunities (txops) allocated by the mobile network we used Saturator[14], a Mahimahi companion tool. As its name implies, Saturator saturates the radio link of a User Equipment (UE). As shown in Fig. 7, we relied on 2 UEs (Xiaomi MI 10 5G), 1 Linux Laptop (Ubuntu 20.04) and 1 Linux Server (Ubuntu 18.04) to perform our experiments. The laptop saturates the radio link of the USB-Tethered UE in the uplink while the server saturates this same link in the downlink direction. The laptop and server exchange Acks through the uncongested radio link of the Wi-Fi-tethered UE. The Linux Server is hosted in an overprovisioned Datacentre. During txops captures, the laptop sends timestamped data packets to the server which are used to compute the RTT upon reception of the corresponding Acks. This RTT value is used to adjust

[11] `http://mahimahi.mit.edu/`
[12] `https://github.com/ravinet/mahimahi/tree/master/traces`
[13] `https://cloud-gaming-traces.lhs.loria.fr/data.html`
[14] `https://github.com/keithw/multisend/`

the client congestion window. The same process is applied on the server-side, so that it can modify its congestion window accordingly and hence saturate the link radio of the base station toward the UE.

At the end of the experiment, raw Saturator data captured on the laptop side is aggregated and processed with the corresponding data on the server side in order to generate two txops file, one for downlink and one for uplink. Each line in the txops file represents a packet delivery opportunity i.e., the time at which an MTU-sized packet can be delivered. If several MTU-sized packets can be sent during the same millisecond, several lines are present in the file with the same timestamp. If no packet can be sent during few milliseconds, there is a corresponding time gap between the timestamps of two successive lines. Such txops files can then be used by Mahimahi's LinkShell tool to emulate a time varying cellular network exhibiting the same transmission opportunities as captured during the measurement.

The captures were performed following this setup on Orange's network in January 2022 and each capture lasted approximately 10 minutes. We made several captures in static conditions to get some diversity regarding mean bandwidth, bandwidth variability and time gaps in the allocation of transmission opportunities. Furthermore, we performed measurements inside a car traveling at the steady speed of 110km/h on a highway to capture txops in mobility. The coverage density varied during the journey (high in starting and ending urban areas and sparser in the rural area in-between). Six traces capturing cellular network conditions were produced in total. Table 2 lists metrics relative to the recorded traces. These metrics are relevant in the context of a CG service: the average bandwidth and its standard deviation, the probability to have at least 20 Mbps (all three with a resolution of 1 second), and the percentage of time when there is no transmission opportunity with a resolution of 33ms (enough time to loose a frame).

Table 2: Characterization of the captured cellular traces through network metrics

	Mean (Mbps)	Std (Mbps)	% of time >= 20Mbps	% of empty period of 33ms
Highway	43.7	31.5	70.5	17.7
Trace A	45.0	7.4	99.3	5.8
Trace B	78.8	20.3	99.5	8.0
Trace C	141.6	49.5	97.4	0.4
Trace D	173.3	43.4	99.8	3.8
Trace E	230.0	48.7	99.9	2.6

Collection of CG traffic over emulated cellular network conditions: In order to replay the captured transmission opportunities, we use the Linkshell tool of Mahimahi with droptail queues of 1875Ko for the downlink and 250Ko for the uplink, which is expected to match the buffer length of production net-

work base stations for the downlink and the buffer length of UEs in the uplink. We capture network traffic before and after Linkshell queues that allows us to measure queuing latencies and loss rates. A low queuing latency is critical to ensure a quick responsiveness and good QoE, while high loss rates can create unpleasant visual artifacts.

Table 3: Average bitrate, latency and jitter of each platform in normal conditions

	Geforce Now	Playstation Now	Stadia	Xcloud
Average bitrate (Mbps)	31.1	23.9	29.6	16.0
Latency (ms)	6.7±0.3	6.6±0.3	10.2±0.5	12.6±0.7

To evaluate the impact of cellular networks over CG traffic, Table 3 is given as a reference when a user plays with an FTTH connection. To measure end-to-end network latency, the method depends on the platforms: for STD and XC we use Chrome's WebRTC internal reports, for PSN we ping the IP address of the cloud gaming server and for GFN we ping the nearest node from the server that responds to ping requests. One noticeable difference compared to our previous capture campaign is the bitrate of XC. It increases from 12.6Mbps (in sections 4 and 5) to 16 Mbps as a consequence of XC now supporting 1080p resolution.

6.2 Bitrate Analysis of CG traffic adaptation to cellular network conditions

Table 4: Mean UDP throughput (left) and standard deviation (right) for each platform in the different cellular network conditions (in Mbps)

	Geforce Now		PS Now		Stadia		Xcloud	
	Mean thpt	Std thpt	Mean thpt	Std thpt	Mean thpt	Std thpt	Mean thpt	Std thpt
Highway	15.16	11.07	6.79	7.66	8.25	4.32	6.76	3.64
Trace A	28.84	4.53	21.69	2.00	12.21	2.19	10.77	2.37
Trace B	28.95	3.40	23.80	1.12	28.41	4.19	14.34	2.35
Trace C	30.71	2.53	23.72	0.29	30.01	3.93	13.36	0.53
Trace D	29.80	4.02	23.71	1.16	27.30	7.92	12.69	3.69
Trace E	30.06	2.44	22.17	2.09	27.69	6.23	11.81	3.12

Table 4 lists the average bitrates with their associated standard deviation for each CG platform in the different cellular network conditions we emulated with Mahimahi's Linkshell. Table 5 compares those bitrates to the available bandwidth offered by the network opportunities but also to the bitrates obtained in Table 3 to see how far they are from their nominal bitrate.

Table 5: Relative bandwidth usage compared to available bandwidth (left) and used bandwidth in reference condition (right)

	Geforce Now		PS Now		Stadia		Xcloud	
	% avlb	% ref	% avlb	% ref	% avlb	% ref	% avlb	% ref
Highway	34.5	48.8	15.4	28.4	18.8	27.9	15.4	42.3
Trace A	65.4	92.8	49.2	90.9	27.7	41.3	24.4	67.4
Trace B	38.4	93.1	31.6	99.7	37.7	96.1	19.0	89.7
Trace C	19.2	98.8	14.9	99.4	19.3	101.5	8.4	83.6
Trace D	17.3	95.9	13.8	99.4	15.8	92.3	7.4	79.4
Trace E	12.8	96.7	9.5	92.9	11.8	93.6	5.0	73.9

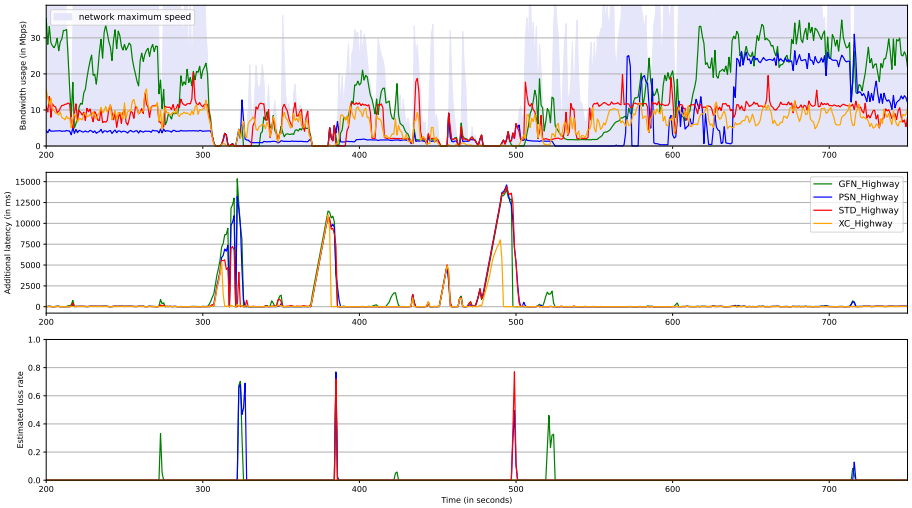


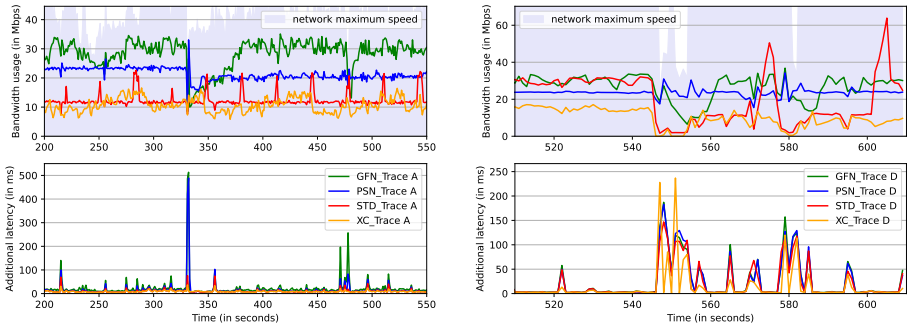
Fig. 8: Analysis of CG platforms' traffic on a cellular network under fast user mobility (highway)

First, the results obtained in highway conditions deserve a specific analysis. This is by far the most demanding trace for which no CG platform can provide good results. These conditions are extremely rough and result in low bitrates with high standard deviation. Figure 8 shows the bit rate of the four platforms for a portion of the trip, where the mobile network bandwidth reached zero on several occasions. We can see that STD and XC, the two WebRTC based platforms, achieve better results in these conditions: their congestion control algorithm adapts quickly to both bandwidth increase and decrease, resulting in a contained standard deviation. GFN is a little less aggressive when it needs to decrease its bitrate and stay cautious during its recovery phase, leading to slightly higher latencies when the available bandwidth suddenly drops and a slope-like shape during the recovery phases. It is also the platform maintaining the higher bitrates. PSN experiences great difficulties in these conditions. We have previ-

ously seen that it does not react well to latency. Here, this erratic bandwidth destroys the user experience during unstable phases. The only event that makes it decrease its bit rate is packet loss. Otherwise, it continues building outstanding latency until there is an opportunity to forward all the packets accumulated in the queue. Interestingly, the recovery phase around the 560 second mark starts with a chaotic episode followed by a rapid recovery at the end. The platforms remain globally stable after the 650 second mark.

In other mobile network conditions, platforms' behaviours are much more like what we have seen in Section 5. The two webRTC based platforms, STD and XC react more aggressively which results in a more degraded bitrate compared to the reference in a cellular network context. PSN does not adapt its nominal traffic at all for traces B,C, and D while GFN is in between, always adapting its bitrate but in a lesser extend than XC and STD. Most of the captures show local latency spikes when the available bandwidth suddenly drops but packet loss is almost fully avoided with an average loss rate of less than 0.1%. GFN and PSN can offer an experience similar to an FTTH connection with at least 90% of the average bitrate observed in FTTH condition in all non-highway mobile network conditions. PSN offers the stablest bitrate at the heavy cost of a lack of reaction to latency, as we have already seen. Contrarily to the highway conditions, we recognize the recovery behaviour observed in the controlled experiences: a slow and constant bitrate increase. PSN may have two kinds of recovery policy with one that triggers only in degraded conditions. Like in highway conditions, GFN maintains the highest average bitrate since it tries to achieve high bitrate whenever it can. Its slightly slower reactivity compared to WebRTC-based platforms helps containing bitrate variations. STD can offer a good experience starting from average network conditions. Trace A and B are a bit alike with the spiky bitrate pattern we identified in the previous section. It explains the slightly higher average bitrate than in FTTH conditions. The main difference between these two is that in trace A STD stays most of the time in 720p even if 1080p is largely feasible. At the end, it looks like STD overreacts. At last, XC offers a good experience starting from average. It reacts the same way as in the controlled latency experiment: a fast reaction to contain latency increase, but a very slow recovery time. It is worth noting that no strong correlation can be deducted by comparing the metrics in Table 2) and the distance to the reference bitrate of the 4 platforms in Table 3, meaning that the reactions are mainly triggered by specific events occurring in each trace and that become invisible at a larger scale.

Figures 9a and 9b show two cases when the platforms must adapt to their environment: respectively under low available bandwidth and sudden bandwidth drops. They are taken from traces A and D, respectively. The nearly horizontal lines representing PSN's bandwidth usage highlight its lack of adaptation. Only packet loss occurring after a huge spike of latency at 340s (Figure 9a) makes it change its bitrate. GFN behaves along the same line with a huge latency spike but less packet loss since it truly reduces its bitrate when the event occurs. It will then take some time to recover probably in case this problem can repeat.



(a) low available bandwidth (b) sudden bandwidth drops

Fig. 9: Behaviour of CG platforms during low available bandwidth or sudden bandwidth drops

STD and XC do not encounter any problem since they already adapt their bitrates in consequence. On Figure 9b, we can see the difference between GFN and WebRTC-based platforms when sudden bandwidth drops occur.GFN takes a smoother approach than STD and XC to adapt its throughput what does not lead to higher additional latency. Like we already know, their recovery is not the the same; their bit rate increases a bit right after this event. In the case of STD, we also see the introduction of the spiky pattern to check if it can go back to its former state.

Table 6: Sum of normalized standard deviation for each platforms in the different cellular network conditions

	Geforce Now	Playstation Now	Stadia	Xcloud
Highway	10.82	10.95	10.22	9.6
Trace A	1.54	0.77	2.47	2.80
Trace B	1.64	0.40	1.97	2.11
Trace C	1.45	0.67	1.55	0.77
Trace D	1.55	0.49	2.68	2.59
Trace E	1.16	0.81	1.82	2.25

Table 6 aims at comparing how the four platforms manage their bitrate stability. For this comparison, we compute a synthetic metric as follow: we split the gaming sessions in periods of 30s. For each period we compute 30 data points valued with the average bitrate over 1 second. We then compute the normalized standard deviation (the standard deviation divided by the mean) for each period. Finally, the synthetic bitrate stability metric is obtained by summing the per 30s period normalized standard deviation over the session duration, excluding the first 120s of the session as in previous calculations. Letting aside highway

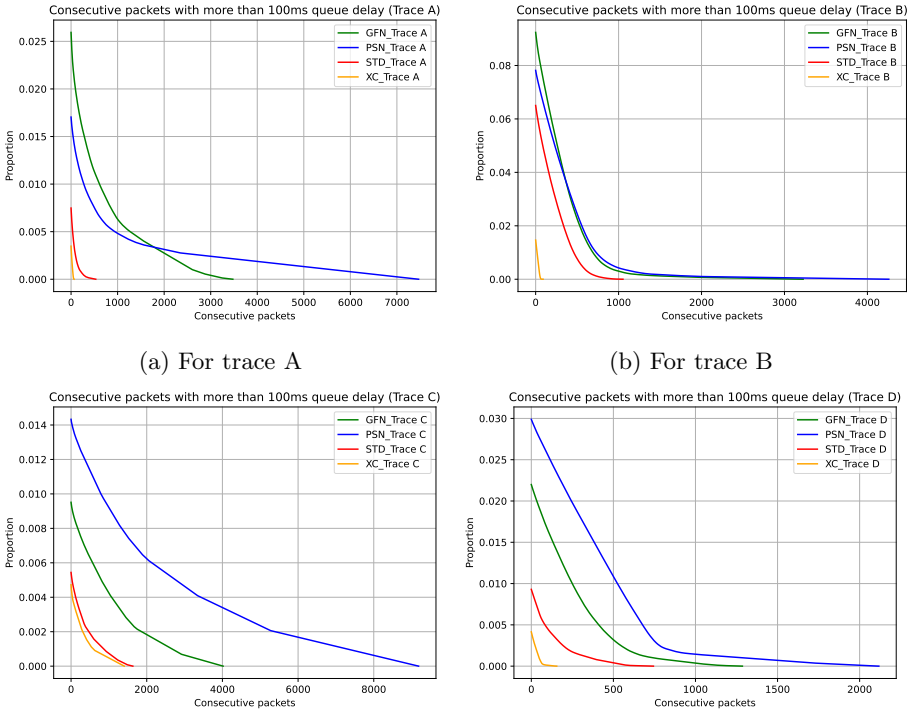
conditions, PSN is the platform exhibiting the more stable bitrate as we could already guess from the previous figures. GFN achieves consistent low scores on the five mobile network conditions. STD and XC have similar results probably because they are based on the same technology. They show a higher bitrate instability due to their high reactivity to network instabilities which, on the other hand, give the best gaming experience during high turbulence.

6.3 Latency Analysis of CG traffic adaptation to cellular network conditions

Table 7: Percentage of UDP packets with additional latency above respectively 10, 20, 50 and 100 ms for each platform in different cellular network conditions

	Geforce Now				PlayStation Now				Stadia				Xcloud			
Highway	51.6	42.8	29.3	14.9	45.5	38.4	27.0	15.0	37.2	28.6	18.4	8.7	27.7	17.9	9.7	4.0
Trace A	52.4	30.4	10.0	2.6	31.2	18.5	6.4	1.7	25.4	14.3	4.5	0.7	22.5	10.9	3.2	0.3
Trace B	26.3	17.4	13.6	9.2	18.5	15.1	11.9	7.8	18.3	12.6	9.7	6.5	7.2	2.8	2.0	1.5
Trace C	8.3	3.1	1.4	1.0	6.2	3.2	1.9	1.4	6.7	2.4	1.0	0.5	5.7	2.0	1.0	0.5
Trace D	8.9	4.9	3.1	2.2	8.4	5.7	4.1	3.0	6.1	2.8	1.4	0.9	4.0	1.5	0.6	0.4
Trace E	4.1	3.1	2.4	1.5	4.5	3.8	3.0	2.0	2.9	2.1	1.7	1.1	1.4	0.8	0.7	0.6

Bit rate reduction is meant to avoid packet loss and increased latency due to queuing. Here, we focus on the latter to evaluate how well latency is contained by the 4 platforms. Table 7 exhibits the percentage of packets experiencing an additional latency of at least 10, 20, 50 and 100ms In this paper, additional latencies refer to packet's sojourn time in Mahimahi's Linkshell downlink queue. We can see that STD and XC lead this metric in all cases, thanks to their fast and aggressive congestion control algorithm. XC achieves better results than STD, but the causes can be multiple. Since XC's bitrate is half of STD's, it is easier to get good results in poor network conditions. In addition, XC has been specially advertised to be used with smartphones. Thus, a particular attention has probably been taken on performance over cellular networks. PSN and GFN do not handle well poor network conditions. On the highway, more than a third of their packets arrive with at least 20ms of additional latency. STD and XC maintain it around a fourth. Similarly, packets that come with more than 100ms of additional latency have nearly twice the chance to occur. We can observe that PSN's results are not so bad on highway. This is because its bitrate is very low between seconds 300 and 500 (see Figure 8), compensating for its poor traffic management. GFN gets worst results than PSN because it seems to make some trade-off between bitrate and latency. Its congestion control algorithm is not as fast as WebRTC-based platforms, and sudden bandwidth drops produce high latency spikes, like in Figure 9a. The reason for this is that it maintains way higher bit rates than the other platforms in poor conditions (traces A and B). Starting



(a) For trace A

(b) For trace B

(c) For trace C

(d) For trace D

Fig. 10: Cumulative sum of packets that are delayed by more than 100ms for each CG platforms in conditions A, B, C and D

from good conditions (traces C, D, E), GFN achieves better queuing latency than PSN, with less packets experiencing at least 50ms of queuing latency.

In addition, Figure 10 gives the cumulative proportion of consecutive packets experiencing at least 100ms queuing delay. Unlike Table 7, which shows a summarized view of the queuing latency, we now consider the "duration" of the latency spikes in number of packets. The 100ms mark has been chosen because it is, most of the time, the acceptable limit for a good user experience. The curves are cumulative in a way that each value in the horizontal axis includes the following ones. XC remains unbeatable in this exercise because the intervals during which packets are more than 100ms late remain very short. The only exception is for trace C where it achieves similar results as STD. STD also exhibits a good reactivity, since it is the second most bandwidth-consuming platform and still manages to stay in the 1000 packets range. GFN and PSN encounter some difficulties compared to the others platforms. Except for trace A and small packet intervals, overall, GFN performs better than PSN but it is still not able to compete with STD and XC. PSN has by far the highest number of consecutive packets exceeding 100ms due to its inability to react to latency.

Overall, GFN is probably the platform that offer the best user experience because of a good trade-off between high bitrate, stability and adaptability. It is probably not designed for cellular networks, but still succeeds to adapt the traffic if there are not too many disturbances. Its congestion control algorithm is not the most reactive but seems to be quite well adapted in most of the cases. When network quality suddenly drops, users may suffer unresponsive periods. This is followed by a longer recovery phase, during which video quality is not the best possible. Its logic seems to react measurably and gently to small disturbances but to remain cautious in case other disturbances may still occur. STD and XC tend to maintain a low latency at any cost but may overreact and reduce the quality of user experience more than necessary. It is especially true for STD that always tries to offer the best video quality but this can be a bit counterproductive because of intermittent changes of resolution or quality under unstable conditions. Last, PSN is not meant to be used over cellular networks. Its congestion control algorithm doesn't take latency into account. Thus, the service continues like nothing happened. It can work on networks with small perturbations but in other cases, users will certainly find it difficult to have a good gaming experience. Our experiments show that CG services do not care and/or take advantage of high bandwidth allowed by 4G/5G cellular networks but can really suffer from network instabilities (sudden drops of bandwidth and temporary disconnections).

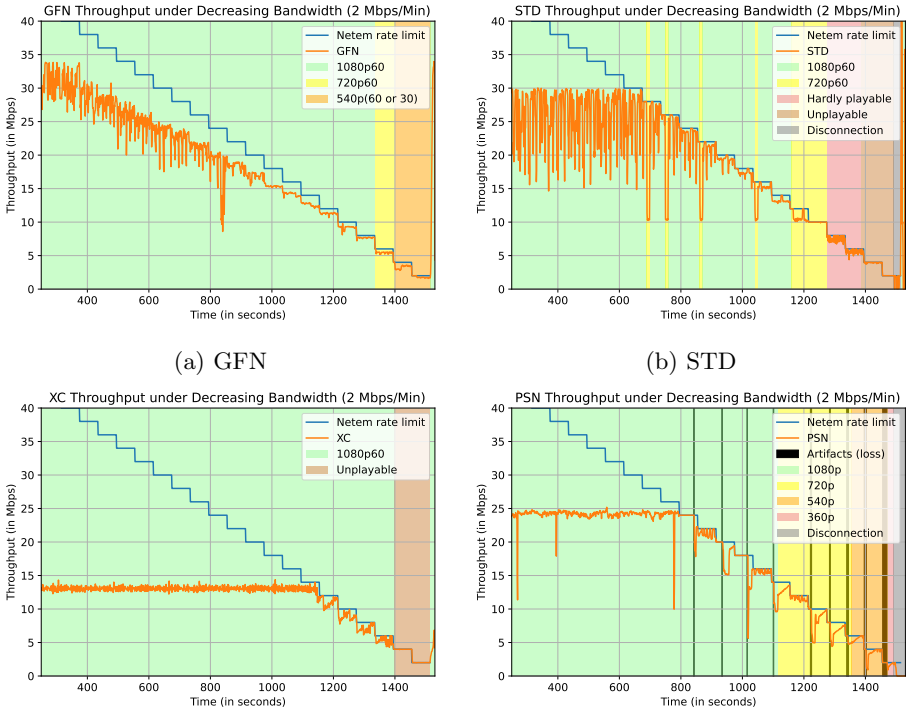
7 Discussion

7.1 Link between bitrate and QoE

As shown in Section 2.1, the QoE is mainly based on the quality of the video stream and the reactivity of player commands. The second criteria is mainly hindered by the added network latency that was analyzed in Section 6.3 while the first is tightly coupled to the bitrate that was deeply analyzed in Sections 4, 5, 6.2. In order to better link the video quality of the different platforms with their observed bitrate, we decided to conduct an additional experiment in which we progressively reduce the client's available bandwidth while observing the quality of the video stream.

During the first 15s, we do not apply any constraint. We then decrease the bandwidth from 50 Mbps to 2 Mbps, by a step of 2 Mbps every minute. Figure 11 shows the bitrate of each platform compared to the available bandwidth, while the colors give both objective and subjective quality of the video stream. Objective information about framerate or resolution changes can be found in the platforms statistics, except for PSN. For this one, we had to record the video feedback with a lossless codec at a high frame rate and to meticulously analyze the video in a second phase to get its properties.

CG platforms mostly rely on video compression to adapt their bitrate and always favor the frequency over the resolution. Each platform has his own strategy and limit that we discuss hereafter. To reduce its bitrate when less than 40



(a) GFN (b) STD

(c) XC (d) PSN

Fig. 11: Link between bitrate and the quality of the video stream provided by CG platforms

Mbps are available, GFN first reduces the quantization factor (video compression). The first resolution change (720p) occurs when only 6 Mbps is reached. The second one (540p) happens at 4 Mbps, with a framerate decrease (30fps). But it immediately switches back to 60 fps. That change is probably due to the lag spikes that occur at each step. For low bitrates, a 2 Mbps bandwidth drop is indeed a significant change. At the end of the session, with a 2 Mbps-bandwidth, GFN is at its lowest quality level; 540p at 30 fps. It is yet still responsive.

STD reacts well to bandwidth changes. There are some transient resolution drops to 720p60, but they usually last 10 seconds. The resolution is definitely set to 720p60 when 12Mbps of available bandwidth is reached. Note that the framerate is fixed at 60fps, regardless of the bitrate used. At 8Mbps, STD becomes unpleasant to play, with a jerky stream with periodic blurry frames. At 6Mbps, it starts to become unresponsive. We finally undergo a disconnection before the end of the experiment.

It is harder to see the adaptation of XC than for the other platforms because its nominal bit rate is lower. It will indeed perceive the bandwidth constraints only toward the end of the experiment. XC does not change its resolution nor its

framerate during the whole session only reducing bitrate through compression. A bit like STD, but in a more brutal way, XC becomes unresponsive at 4 Mbps. The game then becomes unplayable, with freezes that can last several seconds.

At last, PSN has the biggest drops when the bandwidth decreases. It can be noticed on the screen, with corrupt or ill-formed macroblocks due to packet loss. We can see on Figure 11d that PSN has more steps than the other platforms. It begins by switching to 720p at 14Mbps, then switches to 540p at 6Mbps and 360p at 2Mbps of available bandwidth. Shortly afer, PSN stops the connection due to bad network conditions. Concerning PSN framerate, our video analysis shows that it periodically changes between 30 and 60 fps during all the time of our experiment, whatever the current resolution or available bandwidth, with a period of around 75 seconds for the first and 90 seconds for the last.

7.2 Potential venues for improving CG traffic transport over wireless networks

ISPs often advertise their mobile network quality in terms of coverage and bandwidth, but never in terms of latency or stability. These last two criteria seem to be the most important for cloud gaming services.

Until now, optimizing the network for throughput and network capacity utilization has been made at the expense of latency, with the use of deep buffers. Indeed, Internet flows increase their throughput until a packet loss occurs. Large buffers allow to minimize packet loss rates caused by full buffers, thus they help in optimizing throughput. However, the link capacity of wireless networks can vary by several order of magnitude, and when link capacity quickly decreases, packets accumulate in the queue which creates delay. Additionally, the bandwidth delay product guides buffer sizing to achieve high utilization levels by having enough data in the buffer to keep the pipe busy as acknowledgments of connection-oriented transport protocols flow back from the destination to the source, triggering the transmission of next data. In this rule of thumb, the delay is the typical base RTT which is the end-to-end delay without queuing delay. Due to link capacity variations, full buffers can then produce delays several order of magnitude higher than the base RTT link. Transport protocols designed for truly latency sensitive applications are not connection oriented and tend to prefer packet loss over long delays due to retransmissions. Some latency sensitive flows are low rate, such as uplink Cloud Gaming flows, and the above engineering guidelines promoting large buffers meant for typical Internet flows, do not apply. Smaller buffer could be used to bound latency at the expense of packet loss. High-rate and latency sensitive flows, such as downlink CG flows, adapt their bitrates using latency-oriented congestion control algorithms, preferring to ensure the low latency over the high-capacity utilization. Again, the above buffer engineering guidelines do not apply, and smaller buffer could also be used.

An avenue for reconciliating incompatible objectives of typical flows and low latency flows, such as Cloud Gaming, consists in implementing distinct queues with distinct engineering guidelines. Recent work, such as the Low Latency,

Low Loss, Scalable Throughput (L4S) architecture, defined at the Internet Engineering Task Force (IETF), proposes a dual-coupled queues system. One active queuing mechanism, for low latency applications, is designed to be used in conjunction with congestion controls reacting to accurate congestion notifications. Another queuing mechanism is aimed at classic best-effort traffic. The two queues being coupled for a fair bandwidth sharing. However, it can not be deployed as it right now for wireless networks since the link capacity can largely vary and the known wireline Active Queue Management (AQM) do not consider this issue and AQM for wireless networks are still in their infancy. Another solution could consist in limiting capacity variations, also known as providing a Guaranteed Bitrate (GBR) service. But, such a network service is a bad fit for high bitrate flows as guaranteeing high bitrate involves exhausting the radio resources when in bad radio conditions, leaving other users with too few resources. A last venue could be to have the network itself being able to advertise in case of upcoming congestion, instead of the endpoints, in order to help to reduce the adaptation delay and avoid loss. But, if such a solution exists with the Explicit Congestion Notification (ECN) in the latest TCP implementation, no similar feature currently exists for RTP.

Beyond network-oriented solutions, CG platforms also have room to better compensate the network latency. This would consist of implementing algorithms present in the netcode of multiplayer games in any game that must be executed on a CG platform and activate it only when so. Without much detail, that is probably what Google expressed under the name of "negative latency" but this approach requires an important effort from game developers in the absence of a proper framework.

8 Conclusion

We presented in this paper the result of an extensive measurement campaign whose objective was to analyse the reaction of four CG platforms (GeForce Now, Stadia, Playstation Now, Xbox Cloud Gaming) to different synthetic network constraints (bandwidth, loss rate, latency and jitter) either applied initially or during a gaming session and to different mobile network conditions. Despite providing very similar cloud-gaming services, all platforms showed different behaviours due to their own application-level adaptation algorithms, one or another being able to better handle a specific constraint but not necessarily another. Overall, GFN seems to be the most capable platform to adapt its traffic to network constraints, while preserving a trade-off between stability and adaptability when necessary. The two webRTC platforms, Stadia and Xcloud, are very reactive to adapt their traffic to network conditions but tend to overreact. Lastly, PSN is unresponsive to any latency increase due to degraded network conditions, which highly affects user experience.

We conclude that the best-effort approach of current Internet protocols regarding latency exacerbates the differences between applications that must implement by themselves all adaptation mechanisms. But application-level mecha-

nisms often take too much time to detect and react to network issues, regularly leaving the service hardly playable with a lot of stuttering experienced at the user side. Leveraging new latency-oriented network technologies could produce an immediate response to preserve latency requirements and let the time for the application to adapt more smoothly and only in case of lasting disturbance. Our future work will consist in detecting CG traffic thanks to its specific features in order to apply latency-optimized network processing and thus enhance the QoE.

Declarations: Bertrand Mathieu declares to be a guest editor of the present journal but he will not take part in the review or editorial process for the present paper. The other authors declare to have no conflict of interest.

Author contributions: Xavier Marchal made most of the experiments, analysis and writing work in the paper. He notably produced all Tables and Figures. Philippe Graff contributed to the experiments and wrote most of the state of the art. Joël Roman Ky produced the traces from real cellular network and wrote the related section 6.1. Thibault Cholez is the corresponding author. He led the paper direction, supervised the work, wrote several paragraphs, and reviewed the manuscript. Stéphane Tuffin refined the analysis work, helped design the experiment and relevant metrics, wrote section 7.2 and part of the state of the art, and he reviewed the manuscript. Bertrand Mathieu wrote the abstract and introduction, part of the section 6, initiated the capture of cellular network traces and reviewed the manuscript. Olivier Festor reviewed the manuscript.

References

[1] M. Manzano, J. A. Hernández, M. Urueña, and E. Calle, "An empirical study of cloud gaming," in *2012 11th Annual Workshop on Network and Systems Support for Games (NetGames)*, 2012, pp. 1–2. DOI: 10.1109/NetGames.2012.6404021.

[2] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *2012 11th Annual Workshop on Network and Systems Support for Games (NetGames)*, 2012, pp. 1–6. DOI: 10.1109/NetGames.2012.6404024.

[3] 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*, 2011, pp. 330–335. DOI: 10.1109/IMIS.2011.92.

[4] W. Cai, R. Shea, C. Huang, K. Chen, J. Liu, V. C. M. Leung, and C. Hsu, "A survey on cloud gaming: Future of computer games," *IEEE Access*, vol. 4, pp. 7605–7620, 2016. DOI: 10.1109/ACCESS.2016.2590500.

[5] K. Lee, D. Chu, E. Cuervo, J. Kopf, Y. Degtyarev, S. Grizan, A. Wolman, and J. Flinn, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming.," in *MobiSys*, G. Borriello, G. Pau, M. Gruteser, and J. I. Hong, Eds., ACM, 2015, pp. 151–165, ISBN: 978-1-4503-3494-5.

[6] K. Chen, P. Huang, and C. Lei, "Effect of network quality on player departure behavior in online games," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 5, pp. 593–606, 2009. DOI: 10.1109/TPDS.2008.148.

[7] A. Di Domenico, G. Perna, M. Trevisan, L. Vassio, and D. Giordano, *A network analysis on cloud gaming: Stadia, geforce now and psnow*, 2020. arXiv: 2012.06774 [cs.NI].

[8] M. Manzano, M. Uruena, M. Suznjevic, E. Calle, J. Hernandez, and M. Matijasevic, "Dissecting the protocol and network traffic of the onlive cloud gaming platform," English, *Multimedia Systems*, vol. 20, no. 5, pp. 451–470, 2014, ISSN: 0942-4962.

[9] M. Carrascosa and B. Bellalta, *Cloud-gaming:analysis of google stadia traffic*, 2020. arXiv: 2009.09786 [cs.NI].

[10] M. Suznjevic, I. Slivar, and L. Skorin-Kapov, "Analysis and qoe evaluation of cloud gaming service adaptation under different network conditions: The case of nvidia geforce now," in *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, 2016, pp. 1–6. DOI: 10.1109/QoMEX.2016.7498968.

[11] K. Chen, Y. Chang, H. Hsu, D. Chen, C. Huang, and C. Hsu, "On the quality of service of cloud gaming systems," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 480–495, 2014. DOI: 10.1109/TMM.2013.2291532.

[12] R. Shea, J. Liu, E. C. .-. Ngai, and Y. Cui, "Cloud gaming: Architecture and performance," *IEEE Network*, vol. 27, no. 4, pp. 16–21, 2013. DOI: 10.1109/MNET.2013.6574660.

[13] M. Claypool, D. Finkel, A. Grant, and M. Solano, "Thin to win? network performance analysis of the onlive thin client game system," in *2012 11th Annual Workshop on Network and Systems Support for Games (NetGames)*, 2012, pp. 1–6. DOI: 10.1109/NetGames.2012.6404013.

[14] M. Suznjevic, J. Beyer, L. Skorin-Kapov, S. Moller, and N. Sorsa, "Towards understanding the relationship between game type and network traffic for cloud gaming," in *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2014, pp. 1–6. DOI: 10.1109/ICMEW.2014.6890690.

[15] G. Sviridov, C. Beliard, A. Bianco, P. Giaccone, and D. Rossi, "Removing human players from the loop: Ai-assisted assessment of gaming qoe," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 1160–1165. DOI: 10.1109/INFOCOMWKSHPS50562.2020.9162916.

[16] D. Ammar, K. De Moor, M. Xie, M. Fiedler, and P. Heegaard, "Video QoE killer and performance statistics in WebRTC-based video communication," in *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, 2016, pp. 429–436. DOI: 10.1109/CCE.2016.7562675.

[17] S. Zadtootaghaj, S. Schmidt, and S. Möller, "Modeling gaming qoe: Towards the impact of frame rate and bit rate on cloud gaming," in *2018*

Tenth International Conference on Quality of Multimedia Experience (QoMEX), 2018, pp. 1–6. DOI: 10.1109/QoMEX.2018.8463416.

[18] W. L. Tan, F. Lam, and W. C. Lau, "An empirical study on 3g network capacity and performance," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, 2007, pp. 1514–1522. DOI: 10.1109/INFCOM.2007.178.

[19] N. Larson, D. Baltrunas, A. Kvalbein, A. Dhamdhere, k. claffy kc, and A. Elmokashfi, "Investigating excessive delays in mobile broadband networks," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, ser. AllThingsCellular '15, London, United Kingdom: Association for Computing Machinery, 2015, pp. 51–56, ISBN: 9781450335386. DOI: 10.1145/2785971.2785980.

[20] J. Gettys, "Bufferbloat: Dark buffers in the internet," *IEEE Internet Computing*, vol. 15, no. 3, pp. 96–96, 2011. DOI: 10.1109/MIC.2011.56.

[21] H. Haile, K.-J. Grinnemo, S. Ferlin, P. Hurtig, and A. Brunstrom, "End-to-end congestion control approaches for high throughput and low delay in 4g/5g cellular networks," *Computer Networks*, vol. 186, p. 107 692, 2021, ISSN: 1389-1286. DOI: 10.1016/j.comnet.2020.107692.

[22] S. Abbasloo, Y. Xu, and H. J. Chao, "C2tcp: A flexible cellular tcp to meet stringent delay requirements," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 918–932, 2019. DOI: 10.1109/JSAC.2019.2898758.

[23] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein, "Pantheon: The training ground for internet congestion-control research," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, Boston, MA: USENIX Association, Jul. 2018, pp. 731–743, ISBN: 978-1-939133-01-4.

[24] C. Yu, Y. Xu, B. Liu, and Y. Liu, ""can you see me now?" a measurement study of mobile video calls," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014, pp. 1456–1464. DOI: 10.1109/INFOCOM.2014.6848080.

[25] S. Wang and S. Dey, "Cloud mobile gaming: Modeling and measuring user experience in mobile wireless networks," *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 16, pp. 10–21, 2012.

[26] T. Kämäräinen, M. Siekkinen, A. Ylä-Jääski, W. Zhang, and P. Hui, "A measurement study on achieving imperceptible latency in mobile cloud gaming," *Proceedings of the 8th ACM on Multimedia Systems Conference*, 2017.

[27] Z. Tan, Y. Li, Q. Li, Z. Zhang, Z. Li, and S. Lu, "Supporting mobile vr in lte networks: How close are we?" *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*, 2018.

[28] Z. Zhang, S. Shi, V. Gupta, and R. Jana, "Analysis of cellular network latency for edge-based remote rendering streaming applications," *Proceedings of the ACM SIGCOMM 2019 Workshop on Networking for Emerging Applications and Technologies*, 2019.