An Analysis of Cloud Gaming Platforms Behavior under Different Network Constraints

Philippe Graff*, Xavier Marchal*, Thibault Cholez*, Stéphane Tuffin†, Bertrand Mathieu†, Olivier Festor*
*Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France, {first.last}@loria.fr
†Orange Innovation, Lannion, France, {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 outcome of real-life measurements performed between April and July 2021 on the four aforementioned CG platforms, configuring different network constraints like packet loss, throughput decrease, latency increase and jitter variation to observe the behavior of these CG platforms under extreme network conditions.

Our findings show that the four platforms exhibit different adaptation behaviors. Moreover, many cases result in a degraded QoS, leaving room for further improvements at both application and/or network levels.

I. 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 applies the "*game logic*" and streams back the resulting audio and video. 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 catched 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 lucrative market. Big industry names in this space include Nvidia Geforce Now[1], Google Stadia[2], Sony PlayStation Now[3], Microsoft Xbox Cloud Gaming platform[4] (formerly known as xCloud), Amazon Luna[5], 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], [6]. Some works have focused on a specific platform [7]–[10]. Ours is the first to compare several platforms altogether under the same degraded network conditions to evaluate their capacity to adapt 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. The measurements were performed between April and July 2021. The considered types of network degradation cover packet loss, throughput decrease, latency increase and jitter variation. Two tests campaigns have been performed: 1) sessions started under already degraded network conditions, 2) network degradation introduced and removed during gaming sessions to observe how CG platforms mitigate and recover. Our evaluation reveals the limits of the four CG platforms to manage network degradation and highlights their numerous differences in managing them.

The remainder of this paper is organized as follows: section II presents related work, section III gives background on the four considered platforms. Section IV and V respectively present our evaluation under initial degraded network conditions and with degradation during the gaming session. Finally, section VI concludes this paper.

II. Related Work

The protocols used by CG platforms was studied in several works. The authors of [8] study the OnLive (OL) platform and identify several RTP (**R**eal-time **T**ransport **P**rotocol) 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

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

978-3-903176-36-2 ©2021 IFIP

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 fast-paced games to be more sensitive to latency, but more resilient to packet loss. Moreover, the '*service to client*' direction would be 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*).

The adaptation of old CG platform to network conditions were studied in [11], [13], while [9], [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 networks resources run out. This choice ensures streaming smoothness at the expense of quality.

Our work differs from others in that we compare the behavior of 4 modern CG platforms under initial and sudden network constraints. We will show the results of studies focused on a single platform [9], [10], cannot be generalized because each one reacts differently. Other studies compared up to three of them [7], but did not compare their behaviors in the face of network conditions. Moreover, [9] only applied a bandwidth constraint, while [10], dated from 2016, did not cover recent GFN improvements and some weaknesses revealed by the authors are not relevant anymore five years later.

III. Platforms' specificities

We consider four of the most famous CG platforms to date[6]. Other solutions where users have to 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 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 behavior. 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 graphic options and input peripherals. Nonetheless, their philosophy differ: 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 with the

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

exception of 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 experiments. They can now achieve 1080p60fps on some games thanks to their hardware upgrade. Whatever the platform, their core algorithms being unknown, our tests only allow us to make conjectures about their adaptation mechanisms.

As we are the first to consider XC in a research work, we also investigated its transport protocol. Like STD, it relies solely on WebRTC, but in an uncommon way as it uses data channels for audio and video tracks instead of media tracks. In total, XC uses six data channels labeled "video", "audio", "input", "control", "message" and "chat". Please note that today, GFN also has a web-client (officially in beta but nonetheless fully functional) using webRTC in addition to the legacy desktop client. GFN web client behaves a bit differently from the main desktop client, so we only used the latter in our experiments because it appeared to be more optimized.

IV. Cloud Gaming traffic under initial network constraints

A. Experimental conditions

To conduct our study, we collected network captures between April and July 2021 using a commercial FTTH (**F**iber **T**o **T**he **H**ome) 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. They are listed in Table I. 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 IAT (**I**nter **A**rrival **T**ime), the packet sizes, and the throughput.

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 are easy to change and 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 unrealistic nowadays while bandwidth and latency values represent poor DSL connections. However, these values were chosen to push the platforms to their limits.

As shown in Table I, 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 catalogs 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. Given the very large amount of curves produced by our measurement campaign[7], all results cannot be presented due to space limitation. Instead, we focus on the most interesting behaviors and primarily on server-to-client bit rate variations. We discuss the other two measurements when necessary and at times we comment on the upstream direction.

TABLE I: 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

B. 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. Here, "*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. 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. 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 (small margin). 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 behavior. 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 behavior suggests that XC does not handle bandwidth constraints well. It has to periodically readjust its throughput, affecting user experience.

[7]Full dataset repository: https://cloud-gaming-traces.lhs.loria.fr/index.html

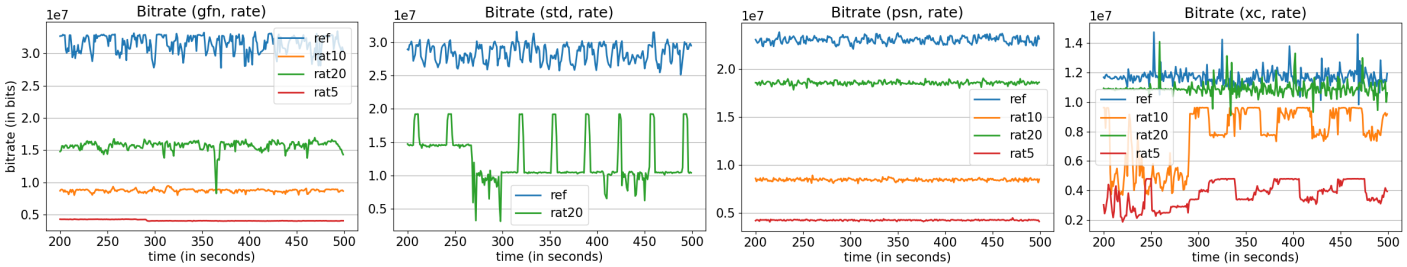

Fig. 1: GFN, STD, PSN and XC bitrate over time for different rate limitations (service⇒client)

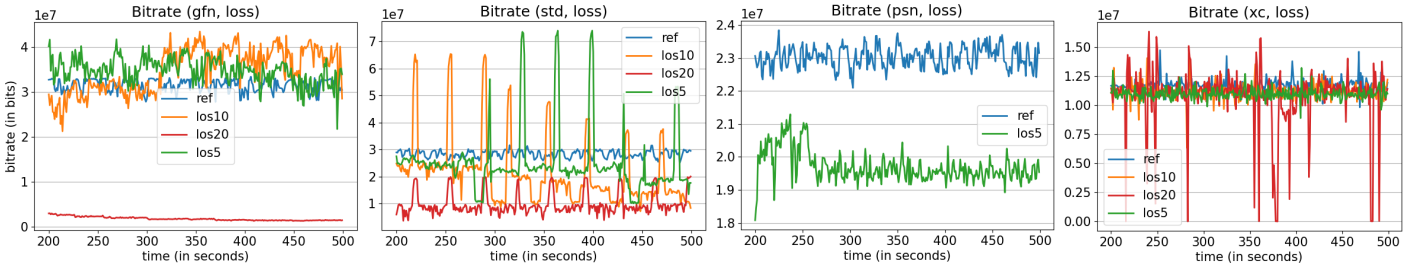

Fig. 2: GFN, STD, PSN and XC bitrate over time for different loss rates (service⇒client)

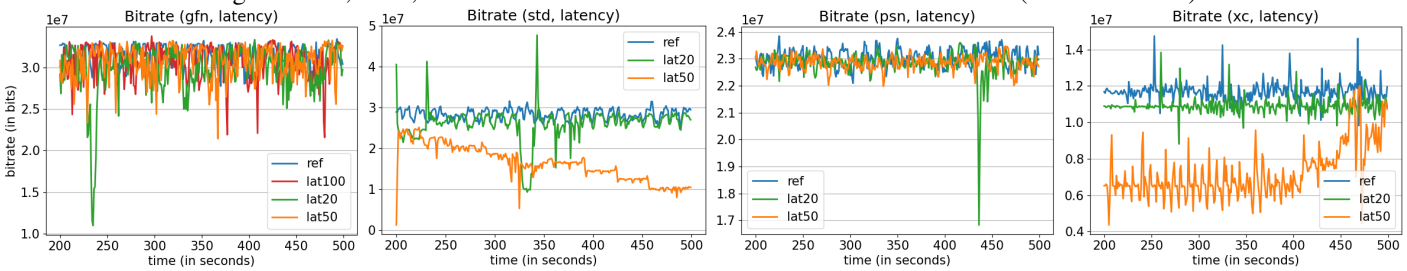

Fig. 3: GFN, STD, PSN and XC bitrate over time for different added latency values (service⇒client)

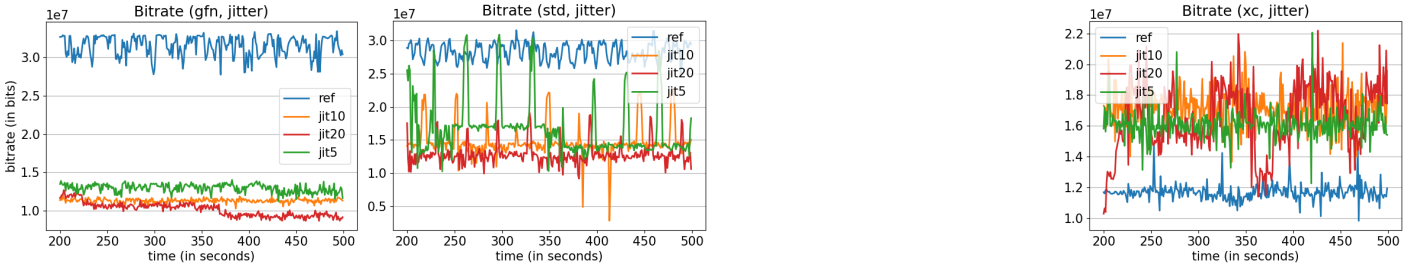

Fig. 4: GFN, STD, PSN and XC bitrate over time for different jitter values (service⇒client)

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 behavior 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. 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 behavior 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. 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 bitrate 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. 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. 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.

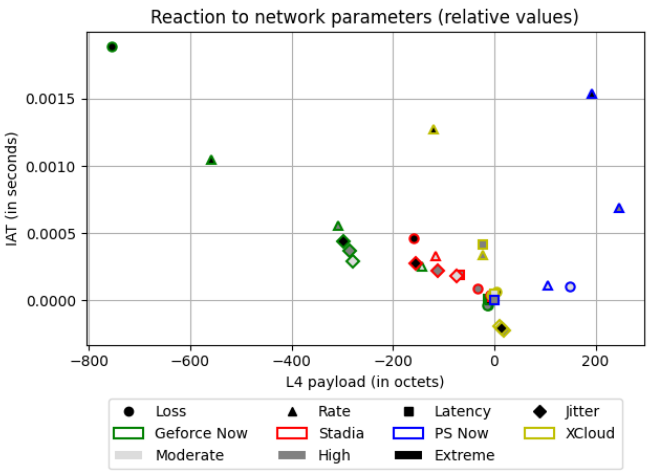

Fig. 5: Relative 2D space for mean UDP payload and IAT (service⇒client)

V. Cloud Gaming traffic under sudden network constraints

The previous section deals with network parameters applied before a session start, so the service can adapt its traffic before launching an instance thanks to different probing tests performed beforehand and documented in [7]. Here, we want to know how the different cloud gaming platforms react when we suddenly change the network conditions during a 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, an other 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 below the constraint it does not need to adapt. We couldn’t choose a lower limitation since we previously noticed that Stadia refuses to start with a bit rate of 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 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.

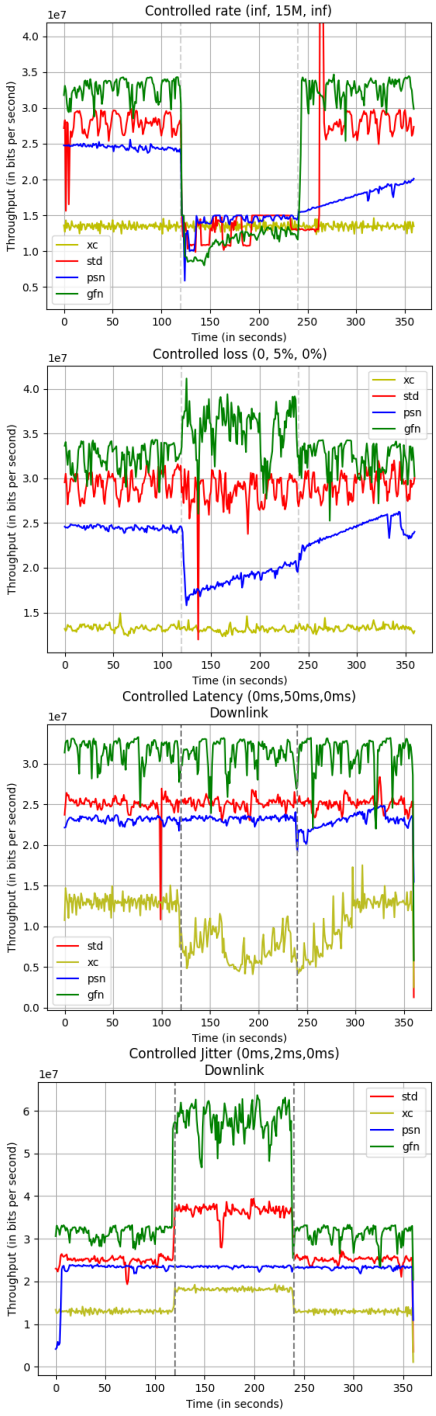

Fig. 6: Server-to-client throughput of cloud gaming platforms under sudden network constraints

Then it recovers quite well with a single notch and a pretty stable bit rate. Afterward, PSN exhibits a slope-like recovery reaching the same level as before the constraint was applied but after the 120s period displayed on the graph. At last, GFN has a reaction strength between the two others, 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 include 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 is on going. 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 60 seconds to get back to its normal rate. GFN and STD, on the opposite, do not adapt their bitrate 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 behaviors 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.

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, 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 all the time when a constraint is applied, and sometimes even after its release, further altering the QoE.

VI. 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 network constraints (bandwidth, loss rate, latency and jitter) either applied initially or in the course of a gaming session. Despite providing very similar cloud-gaming services, all platforms shown different behaviors due to their own application-level adaptation algorithms, one or another being able to handle better a specific constraint but not necessarily another. Overall, GFN seems to be the most capable platform to adapt its traffic to network constraints, all the others fail at some point to adapt and ensure the service continuity. The best-effort approach of current Internet protocols regarding latency also exacerbates the differences between applications that must implement by themselves all adaptation mechanisms.

But application-level mechanisms 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 cloud gaming traffic thanks to its specific features in order to apply a network processing optimized for low latency services to improve their QoS.

Acknowledgment

This work is partially funded by the French ANR MOSAICO project, No ANR-19-CE25-0012.

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. [Online]. Available: http://dblp.uni-trier.de/db/conf/mobisys/mobisys2015.html#LeeCCKDGWF15.

[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.