Network Traffic Characterization of Competitive Online Games

Kyle May

***Abstract*—Competitive online gaming is growing at an unprecedented rate. In 2020, a popular game, Fortnite, saw a concurrent 12.3 million users connect to the game servers. These games demand high performance from the network to provide a good experience to their users; for example, low latency is required to maintain consistency in distributed game state and scalable connection management is important to allow for the number of players on a single server to scale with users' expectations. In previous works, traffic generated by online games, across many genres, have been studied with packet level analyses. For example, inter-arrival time and packet size distributions have been studied in order to gain an understanding of the traffic patterns, with the intention of using this understanding to better the performance of the systems and supporting infrastructure.**

In this paper, we generate traffic traces for a set of competitive games, both isolated and in a realistic setting. The set of games studied are both from previous works and a set of games recently released. With these traces, we provide an analysis of the traffic, looking both at how traffic of games interact with typical accompanying applications and how they perform on their own, trying to build an understanding of both the client and server protocols. For example, we find evidence that clients reduce the frequency of sending updates to the server, which could be a mechanism to reduce server load. Another insight coming from studying non-isolated traffic traces is that clients might act as on/off processes at a larger time scale, as users play multiple sessions and take breaks between them.

I. Overview

A. Introduction

More than 214 million people in the United States play video games [1]. While single-player games are still popular, a majority of these games focus on multiplayer game play, which utilizes the Internet to allow players in separate physical locations to interact. These multiplayer online games have become very popular, particularly competitive games. Earlier this year, a competitive online game, Fortnite, set a record on 12.3 million concurrent players [7].

When thinking about multiplayer online games in the context of networking, it quickly becomes apparent that these are demanding applications. Most games follow a client-server architecture, in which the server maintains the state of the game, sending updates of the game state to each of the clients. From the other perspective, as each player takes actions, which change the game state, the client sends updates to the server, reflecting the actions of the player. The performance of the network will have a direct effect on the experience of the player because each player has the expectation that changes to the game state happen instantaneously and are consistent between all players.

One specific aspect of network performance that affects the experience of players is network and server latency. Because each player expects updates to happen instantaneously, the delay of the server updating local client game state can create some strange behavior. For example, player A moves into the line of sight of player B. Intuitively, the moment that player A moves into the vision of player B, player A would be rendered on player B's screen, and player B would be rendered on player A's screen However, because of the physical distance of the players and the client-server architecture, first player A moves and can see player B. However, in order for player B to see player A, first player A has to communicate to the server and then the server has to update player B's game state. High network latency can create these inconsistencies; however, when the latency is low, in practice, it is hard for players to notice.

Furthermore, within the last 10 years, games have scaled from 10s of players to 100s of players, with the same latency requirements for a good player experience. Therefore, there is a higher requirement on the server, managing more client connections. Imagining that the demand for the number of players in a single environment continues to scale, this could create a bottleneck for servers broadcasting updates to large number of players.

Ultimately, network performance impacts the experience of the players. As video game developers move from offering their games as a single purchase to providing it as a service, it is now even more important for devs to provide consistent, enjoyable experiences to retain customers. Therefore, understanding the network performance of video games and optimizing their performance should help the fiscal performance of a game.

In previous works, packet level analysis of traffic generated by these games have been studied to start understanding the characteristics of these applications. Statistics such as packet size distribution, inter-arrival time distributions, and more have been used to understand the load that the applications have on the network. Using these statistics, techniques for bettering performance have been suggested [9].

In this paper, we provide the methodology for collecting client traffic traces and analyzing these traces for a set of competitive online games released over the past decade. We build on prior works by (1) replicating results for a set of games previously studied, and (2) providing results for a set of games that have yet to be studied. On top of this analysis, we provide a more realistic traffic trace, not isolating a single game application, but rather observing a typical use case,

including concurrent music streaming and voice calls, and how the traffic interacts at the client. We provide many questions that could be answered in future works; for example, we pose questions about how clients can decrease server load by intelligently deciding when an update is redundant and how protocols have changed over the past decade.

B. Research Question

To study the traffic characteristics of a set of competitive online video games to develop an understanding of their networking behavior.

II. Related Work

The related work is separated into two sections. The first focuses on papers that emphasized modelling online game traffic, and the second section focuses on cloud gaming. The first section contains the most closely related works.

A. Traffic Modelling

The authors of [5] start to investigate the attributes of network traffic generated by a popular first-person shooter (FPS), Counter Strike. In order to do this, they studied a server that was connected to in a local-area network. They were able to analyze both client and server behavior because of the direct access to the server. With this, they describe qualitatively the actions of the client and the server, and provide some empirical data describing the communication.

The authors of [6] expand on this work by hosting a public server for Counter Strike, allowing users from the wider Internet to connect and play on this server. The traffic to and from the server was recorded and analyzed. From this trace, they found that the traffic was both highly predictable and highly periodic. Furthermore, they find that the packet sizes are small. With these insights, the authors start to explore the implications for networking infrastructure.

In [4], the authors explore a different genre of games, massively multiplayer online role playing games (MMORPG). They focus on establishing important characteristics of the traffic generated by the games, for example small packets and periodicity. These characteristics are similar to the properties found in the previous studies of FPS games. An interesting result of the paper is the affect of the use of TCP, particularly the large overhead of headers on small packets and acknowledgements.

The impact of genre on the traffic characteristics is explored in [3], in which the authors provide a survey of the results from a myriad of other works that look at many genres, beyond first-person shooters and MMORPGs. In this work, they provide a packet level analysis, including inter-arrival time distributions and packet size distributions, for each genre, and provide comparisons between the different genres. Furthermore, they identify gaps in the current research, particularly that online puzzle games and online fighting games have yet to be studied.

The largely popular game Fortnite is studied in [9]. In this study, the authors generate traffic traces, and try to correlate these traces with the in-game behavior of players. Their hope is that by taking advantage of domain- and game-specific information, networking protocols can adapt and provide a higher quality of service. One of their proposals is that gaming could take advantage of information-centric networking.

While the authors of [9] provide an analysis of traffic behavior of a modern competitive online game (Fortnite), new games are being constantly released and provides us with the opportunity to (1) recreate results and (2) provide analysis of newer competitive online games.

B. Cloud Gaming

The authors of [10] provide an architectural view of cloud gaming, with the goal of showing their uniqueness. On top of this, they start measuring performance of different games, focusing on interaction latency and streaming quality. Their results show that there are many challenges while moving towards the widespread deployment of cloud gaming.

A design effort to provide an open-source cloud gaming platform is recounted in [8], with the hope that providing an open-source platform to study will stimulate research into cloud gaming. After providing some detail about their design, they compare their service against other popular cloud gaming services at the time, showing that they achieve higher performance without creating larger network loads.

The authors provide a comprehensive study of a subset of available games on Google Stadia in [2], in particular gathering network traces and analyzing them. In their study, the authors study a variety of single player games provided on the Google's cloud gaming platform. Their reasoning in studying different games is that it will exercise the platform in different ways; however, they find that the traffic has similar properties between all games, and that a parameterized model would be a succinct way to account for game differences. Furthermore, they provide a description of the protocols used by Stadia in order to implement the game streaming service.

III. Data Collection and Analysis Methodology

In this section, we describe how traffic traces were collected and how each data set was generated from the traffic traces.

A. Data Collection Sites and Environment

All traces were collected on a Windows 10 machine connected to the Internet via a home router. While running each game, Wireshark 3.4.0 was run concurrently to collect all packets transferred on the Ethernet interface. In order to try to isolate the traffic of the game under test, all foreground and background processes not necessary for running the game, belonging to the current user, using the network (over a ten second period) were killed prior to collection. Automating this isolation would be an improvement over the current collection methodology. In Table I, the collection methodology is summarized.

B. Trace Contents

For each packet collected by Wireshark, the following information is collected: packet number (relative to the trace), time

TABLE I
SUMMARY OF KEY SYSTEM AND METHODOLOGY CHARACTERISTICS

Operating System	Windows 10
Network Protocol Analyzer	Wireshark 3.4.0
Data Analysis Tool	MATLAB R2020B
Trace Length	1 game
Trace Format	.pcapng
Internet Service Provider	Charter Spectrum

stamp (in microsecond resolution), IPv4 source address, IPv4 destination address, protocol type, and additional information (packet contents). Each trace is stored in a .pcapng file format.

C. Trace Collection

The following steps were used to collect traces for each game.

1) Start the game and navigate to the menu that allows you to queue yourself to play a game.
2) Start Wireshark collection, as described above.
3) Play a single session.
4) At the end of the session, stop the Wireshark capture.

In this methodology, there is not the possibility of connecting to multiple game servers, assuming that the infrastructure does not transfer running games between servers. From this, we can infer that the path between the client and server locations are stable.

D. Data Set Generation and Analysis

MATLAB R2020b was used to generate the analyses. Each MATLAB script takes in a .csv file of the dissected packet contents. The key data points for the dissected packet contents are the time stamp, protocol type, and packet size.

In order to only generate stats for the packets of interest, display filters were used; these filters can specify a variety of parameters, from IPv4 addresses for source and destination to protocol. The filters that were used are given with the generated data sets, but do not reflect the actual expressions used by Wireshark.

The following data sets were generated with the associated filters (LOCAL_HOST denotes the public IPv4 address of the desktop running the trace collection).

1) Host-generated packets: *ipv4.src == LOCAL_HOST*
2) Host-destination packets: *ipv4.dst == LOCAL_HOST*
3) All packets: *ipv4 == LOCAL_HOST*

After these data sets are generated, MATLAB scripts generate the analyses used in this paper.

The following sections highlight exactly how each of the statistics were calculated. For each histogram, the numbers were normalized in order to create a probability distribution.

1) Packet Size Distributions: For all of the data sets, a histogram was created for the packet sizes in bytes.

2) Inter-arrival Time Distributions: For the HDP data sets, the time between packets were calculated. Then, a histogram was created using these time deltas.

3) Inter-departure Time Distributions: For the HGP data sets, the time between packets were calculated. Then, a histogram was created using these time deltas.

4) Bandwidth Usage over Time: For the HGP and HDP data sets, the following procedure was used to generate bandwidth usage statistics for a given time window size. Each packet was assigned to a given time epoch. Within a time epoch, packet sizes in bytes were summed and divided by the time window size. This procedure generates a data point per epoch, and these data points were used to generate the visualizations.

E. Game Selection

A variety of games released in the past 10 years were selected to be studied. This was done in hopes to show an evolution in the networking behavior of games. All of the games selected include a competitive play list in which players compete for ranks. However, some of the games are first-person, while others are third-person. Some details about each game as well as details about the generated trace are given in Table II.

Many games run anti-cheat software while the game is running; this software can conflict with trace collection and ultimately limits which games can be studied. For example, recently released game *Call of Duty: Black Ops Cold War* would not run while trace collection was being performed.

F. Non-isolated trace generation

To this point, game traffic has been studied in isolation with respect to the other network traffic that would additionally be generated while playing a game. For example, players use other services to talk to teammates (i.e. Discord) because the quality provided by the in-game voice chat is not always the highest quality. Another example of an application that would typically accompany game traffic would be a music streaming application, ranging from Spotify to Youtube live streams. The final data set generated is a result of trace capture, following the methodology laid out above, of a realistic workload, using Counter Strike: Global Offensive as the prototypical game.

IV. RESULTS

In this section, we explore our results for each of the analysis; furthermore, we pose questions suggested by some of the data points that could be explored in future work.

A. Packet Size Distributions

In Figure 1, the packet size distributions for each game, split into each data set, are shown. There are three interesting components of these results: (1) in C, there is a spike at a small packet size, but also a non-trivial component spread from 500 to 1000, (2) in all of the traces, there were large packets sent from the server to the host, and (3) in V, R, and N, the distributions are more spread among smaller packet sizes compared to C, F, and A.

As seen in the C histogram of Figure 1, there is a narrow concentration in the histogram at X. However, the HDP data set contributes a large amount of packet sizes centered roughly

TABLE II
TARGET GAME SET AND TRACE INFORMATION

Name	Trace ID	Type	Num. Players	Release Year	Trace Capture Date (2020)	Number of Packets
Counter Strike: Global Offensive	C	First-person shooter	10	2012	Dec. 1, 7:40PM	151372
Fortnite	F	Third-person battle royale	100	2017	Dec. 12, 5:34PM	69354
Apex Legends	A	First-person battle royale	60	2019	Dec. 1, 7:30PM	137568
Valorant	V	First-person shooter	10	2020	Dec. 1, 6:12PM	97117
Rogue Company	R	Third-person shooter	8	2020	Dec. 1, 6:58PM	86617
Non-isolated	N	CS:GO	4	2012	Dec. 12, 5:14PM	694819

around 800 bytes. This is interesting because this means that the updates sent by the server to the client are not fixed size, and relatively large (from 500 to 1200 bytes). From the client, updates to the servers are small (around 100 bytes). We would have liked to explore exactly what the source of these packets are; for example, maybe these are used to connect the client to the server, but not for updating client game state. Recording larger traces would reduce the overall impact of the startup packets on the distribution.

In all of the traces, there are outliers in the packet size distributions, typically from 1200 to 1600 bytes. These are concentrated peaks in the distribution, so we imagine that these packets are for a specific purpose. Further, these packets are generated by the server. Because of bandwidth measurements, we believe that these packets are used to connect clients to their game servers.

The packet size distributions for C, F, and A are more concentrated on a single small packet size compared to V, R, and N. Since N is a non-isolated version of C, the other applications most likely dilute the original distribution, with a variety of small packet sizes. In order to compare C and N, we would need to figure out a way to separate the packets into the separate applications running concurrently in N. However, V and R are peculiar. The packet sizes are widely distributed, for both client- and server- generated packets. These are two games released in 2020. When this is compared to the games released a few years ago, this is a tangible difference. In order to see if there has been a shift in the industry standard protocols for games, we could increase our set of games to see if this trend continues.

B. Inter-arrival Time Distributions

In Figure 2, the inter-arrival time distributions for all games are shown. Intuitively, these are the probability density functions for the time between consecutive packets arriving at the host. There are three interesting observations from these distributions: (1) all isolated traces have a large peak at some interval less than 0.05 seconds, (2) some traces have a peak at almost 0 seconds, and (3) in the non-isolated version, there is no concentrated peak, but the majority of the inter-arrival times are below the original peak from the isolated version.

For each isolated isolated trace, there is a large peak at some small time interval (from roughly 0.01 to 0.05 seconds). This is the period of the updates sent by the server to all of the clients. The smaller the period, the more frequently the clients and server are synchronized, ensuring that the game state is consistent. This means that V should have the best synchronization, whereas A should have one of the worst. It does not look like there is a correlation between the release year and the update period of the servers. It would be interesting to explore this interaction more with user surveys to see exactly how this impacts quality of experience.

Another interesting aspect of some isolated traces (F, A) is that there is a large spike in the distribution near 0 seconds. We struggle to find an intuitive explanation for this; therefore, we would need to explore this more to ensure that this is not an error in how we calculate the inter-arrival times and its distribution.

In the non-isolated trace (N), the interarrival time distribution does not have a single peak, rather the distribution is spread over many values, from 0 to the original peak in the isolated version (C).

C. Inter-departure Time Distributions

In Figure 3, inter-departure time distributions are shown for each game. These are distributions for the time between sending consecutive packets from the host. There are two interesting aspects of these distributions: (1) most traces have a single peak and (2) F has 3 distinct peaks.

C, A, V, and R all have relative concentrated peaks in the distributions, showing that updates from the client to server are sent periodically. However, it is interesting the peaks from the inter-arrival time distributions do not match the peaks found in these distributions. For example, the peak in A's inter-arrival time distribution was at 0.05 seconds, whereas the peak in the inter-departure time is around 0.015 seconds. This would imply that the client updates the server at a higher rate than the server updates the client. This needs more investigation but this could be a way to manage traffic and processing at the server while the number of client connections increase.

In F, the distribution has multiple peaks in the inter-departure time distribution. This is interesting, as it implies that the client updates the server at different rates. There could be multiple reasons why this could be useful; for example, if the game state knows that the user does not need to update the server (i.e. in menu), then the client-side protocol can make the decision to not update the server to minimize traffic. As F has the highest player count per server (100) in the games studied, it would make sense that they would optimize for this. More investigation into the protocol would be needed to completely understand this mechanism.

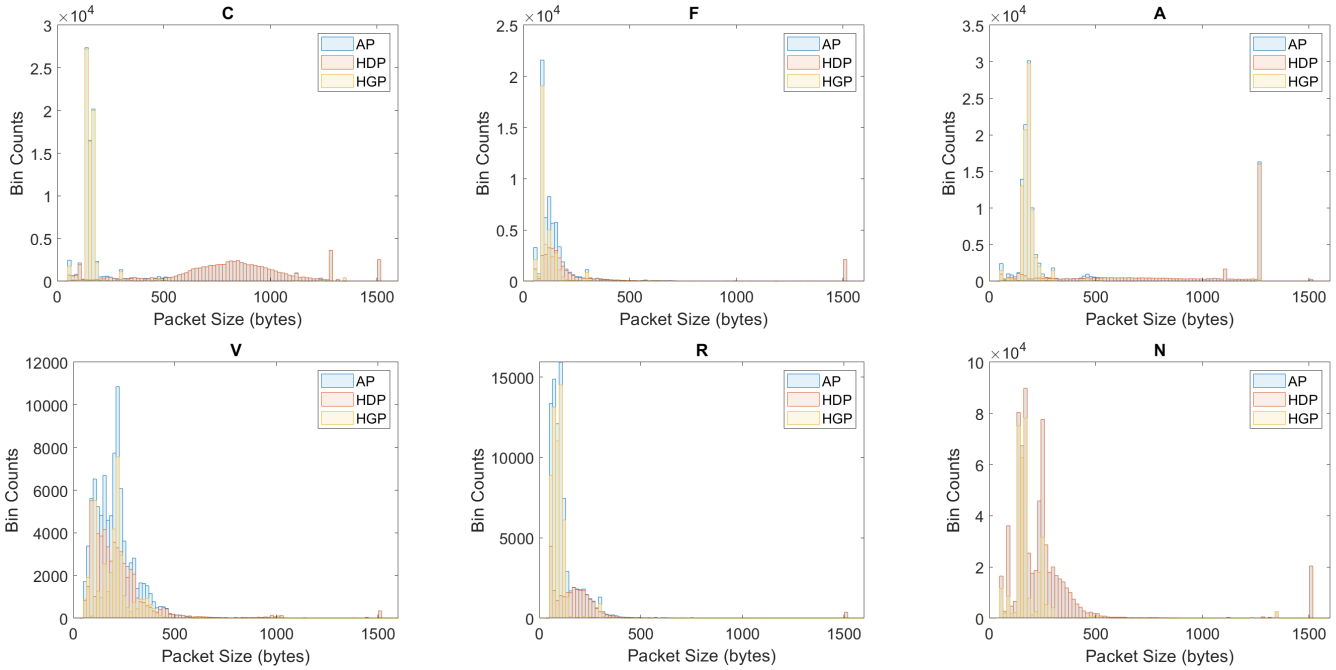

Fig. 1. Packet size distributions. For each game, three distributions are shown (for each dataset). The HGP and HDP distributions should add up to the AP distribution. As seen, the majority of the packets are relatively small, but there are some larger packets received by the client.

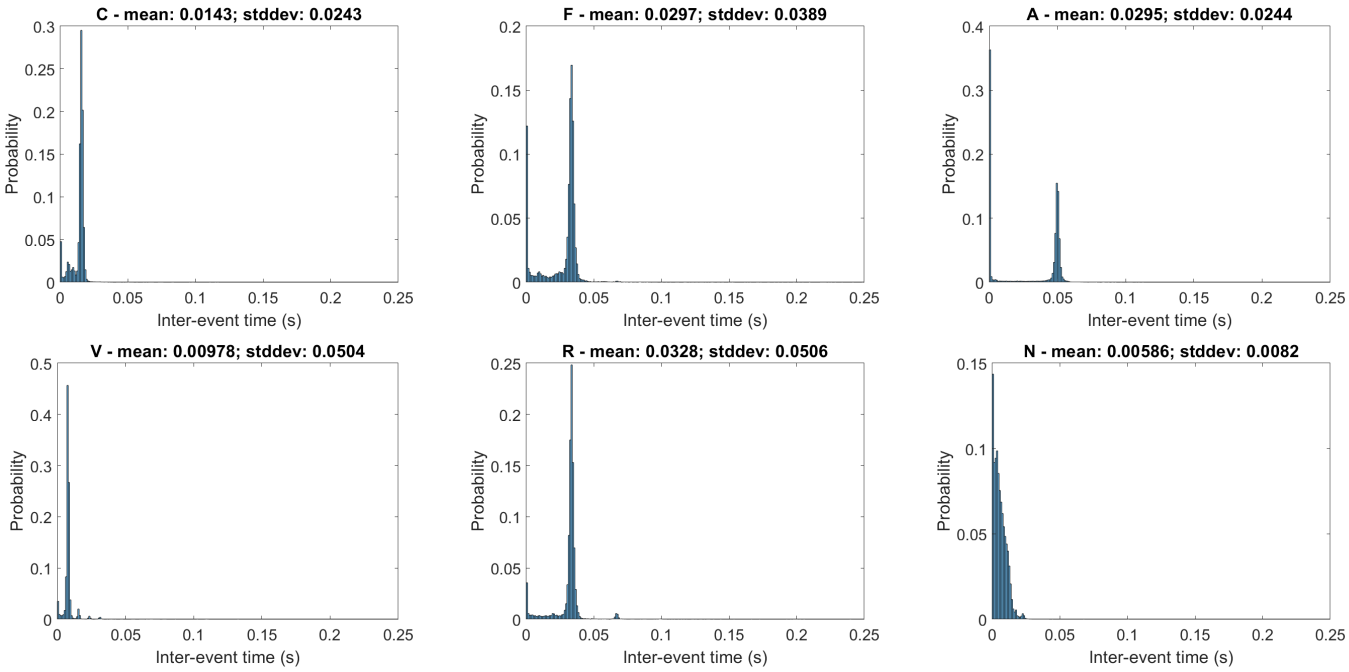

Fig. 2. Inter-arrival Time Distributions. This shows the time between packets arriving at the host. The distributions for the isolated games are concentrated around a single time, which should be the time between updates from the server. However, for the non-isolated trace, there are more arrivals at a higher rate. This might be due to the streaming nature of other applications like Discord.

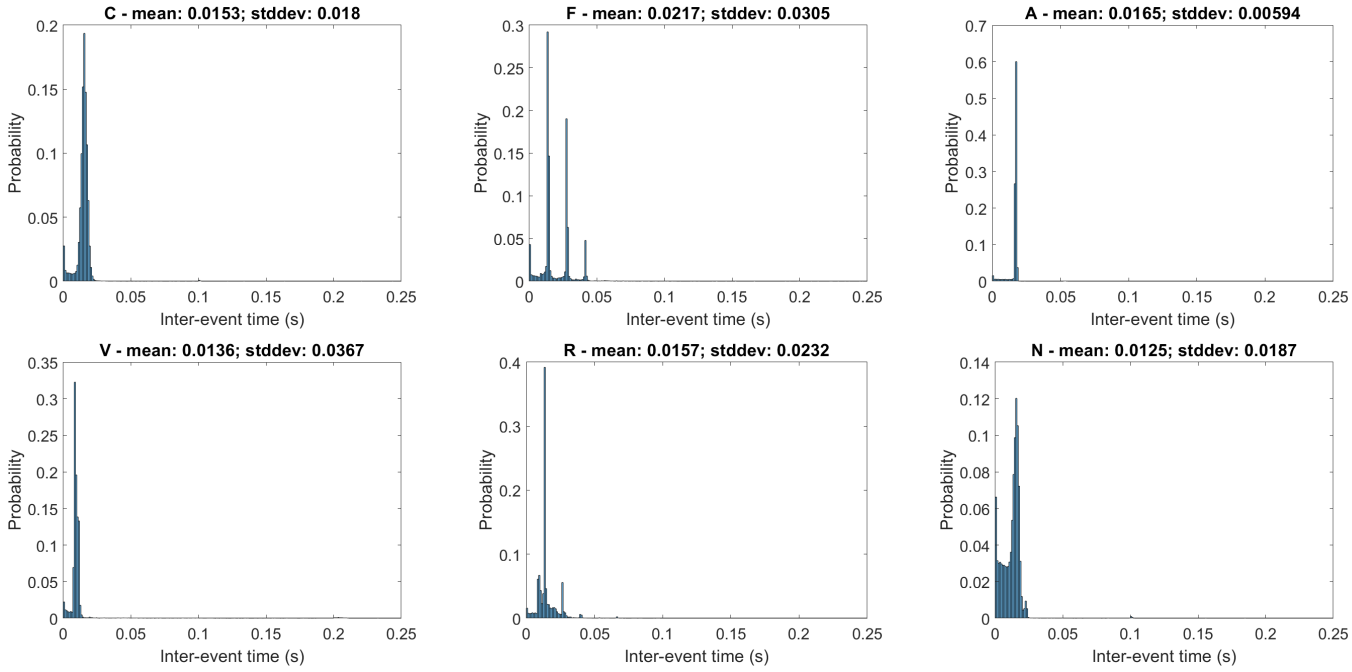

Fig. 3. Inter-departure Time Distributions. These distributions are concentrated around every 10 ms; however, there is some jitter. The most interesting plot from this set is the F data point. As seen in F, there are multiple peaks in the distribution, at distinct times.

D. Download Bandwidth Usage over Time

Figure 4 shows the download bandwidth usage over time, separate into 20 second windows. The isolated traffic traces are curious; some of them are relatively smooth (F) while others end up being bursty (R). The non-isolated trace emphasizes that upon ending and starting a game in C, there is a huge spike in the bandwidth consumption.

The trend seen in the non-isolated traces is that either upon starting or ending a game session, there is a spike in bandwidth consumption. During the game session, there is a somewhat consistent bandwidth consumption. A couple of behaviors that could be explored more would be the cause of the bandwidth consumption increases in F around the 25th and 40th epoch, as well as the cause of the burstiness in V and R.

The non-isolated trace is interesting. The distinct game sessions and lengths of the them are apparent. Further, there is a decreased bandwidth consumption directly after the peaks, which is the amount of time between starting a game session, while sitting either at the menu or in the queue (to find a game). This leads to more questions about higher level on/off behavior of the games. In order to start exploring this idea, more traces, from more users, would need to be collected and the on/off time characteristics would need to be explored.

E. Upload Bandwidth Usage over Time

In Figure 5, upload bandwidth usage over time is presented for each game. There isn't any particular data points that are interesting besides V, which is explored in the next section. For each data set, the update bandwidth usage is relatively constant during active game session, and dips off at the end of the game session. These dips can be seen to occur at the same time epochs in the non-isolated download bandwidth usage in Figure 4.

F. Bandwidth Usage analyzed over Different Time Scales

Figure 6 shows V's upload bandwidth usage over different time scales. This game's data was selected due to the dips in bandwidth used while the game session was active. By decreasing the time scale, it can be seen that there are certain points that use absolutely no upload bandwidth. This could be an optimization at the client to decrease redundant updates, which would decrease the load at the server side. To fully understand this, game state changes would need to be correlated with the changes to bandwidth consumption.

V. Discussion

This section will discuss key takeaways from this project, both from the data produced and from the experience of generating traces and analyzing them.

A. Pedagogical Takeaways

There were 3 aspects of this project that enabled us to start learning how to do networking research: (1) setting up Wireshark workflow, (2) performing simple analyses with MATLAB, and (3) inspecting data. Visualizing the data to be able to look for interesting trends was an important step, and being able to do this quickly with MATLAB was helpful.

B. Future Work

In this section, we explore potential avenues to continue exploring this research question, on top of the questions posed in the Results section.

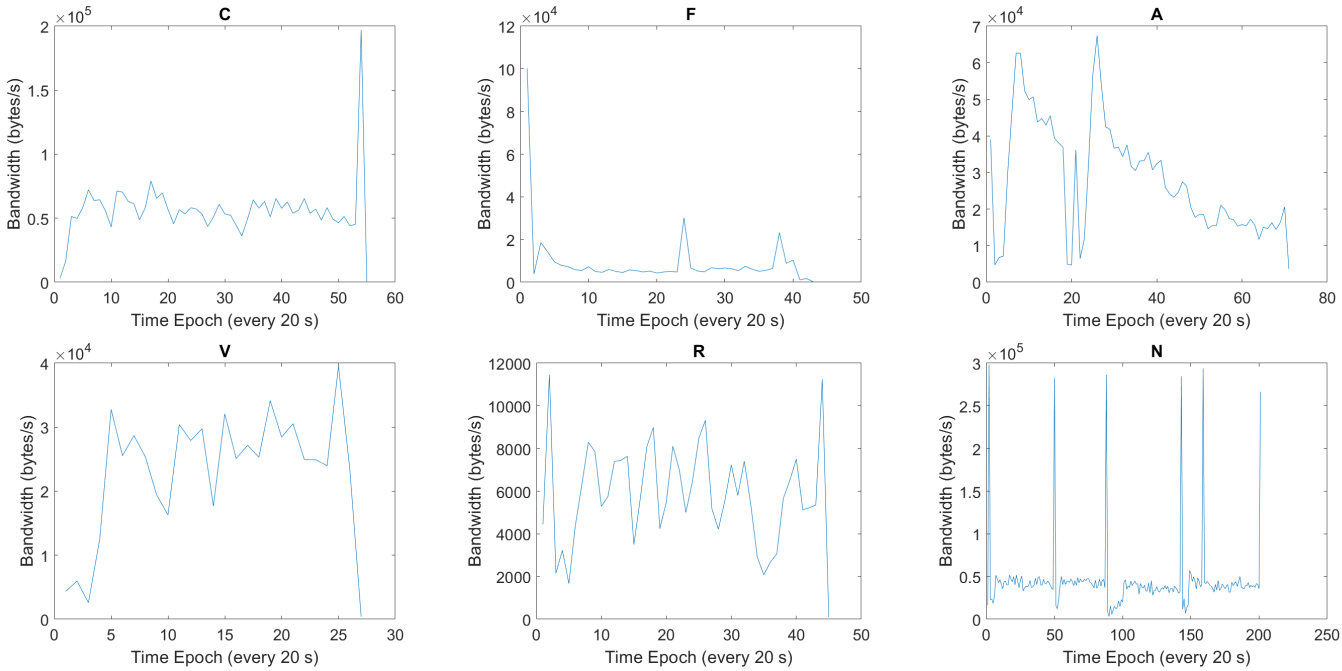

Fig. 4. Download Bandwidth Usage over Time. The bandwidth requirements, particularly the timing of the high bandwidth consumption are different for many games. For example, C has a relatively consistent bandwidth consumption until the client disconnects. However, in F, there is a large spike in bandwidth consumption while the client connects to the server and then it is relatively stable through out the game session.

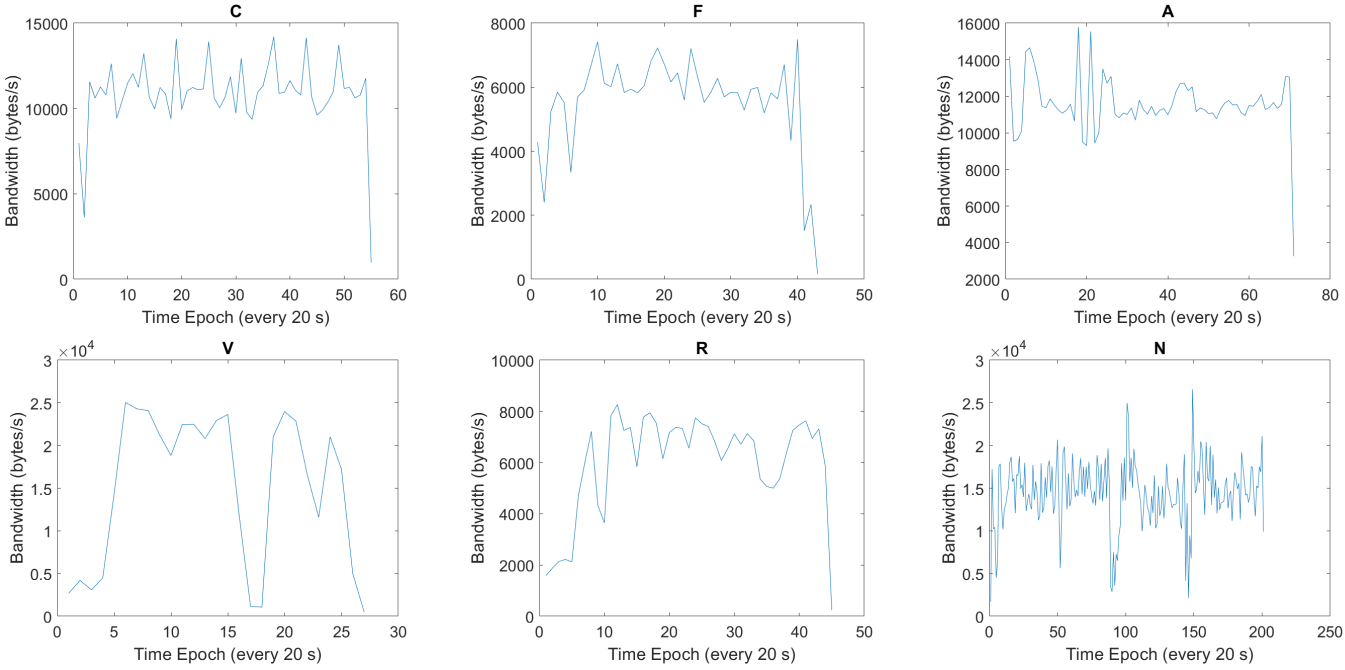

Fig. 5. Upload Bandwidth Usage over Time. For most of the generated traces, the bandwidth consumption is relatively stable, and then when the client disconnects, the bandwidth drops off completely. However, in V, bandwidth generated drops off periodically during the game session, which means that it could potentially be an optimization to limit client-to-server communication depending on the game state. V is explored in more depth in Figure 6.

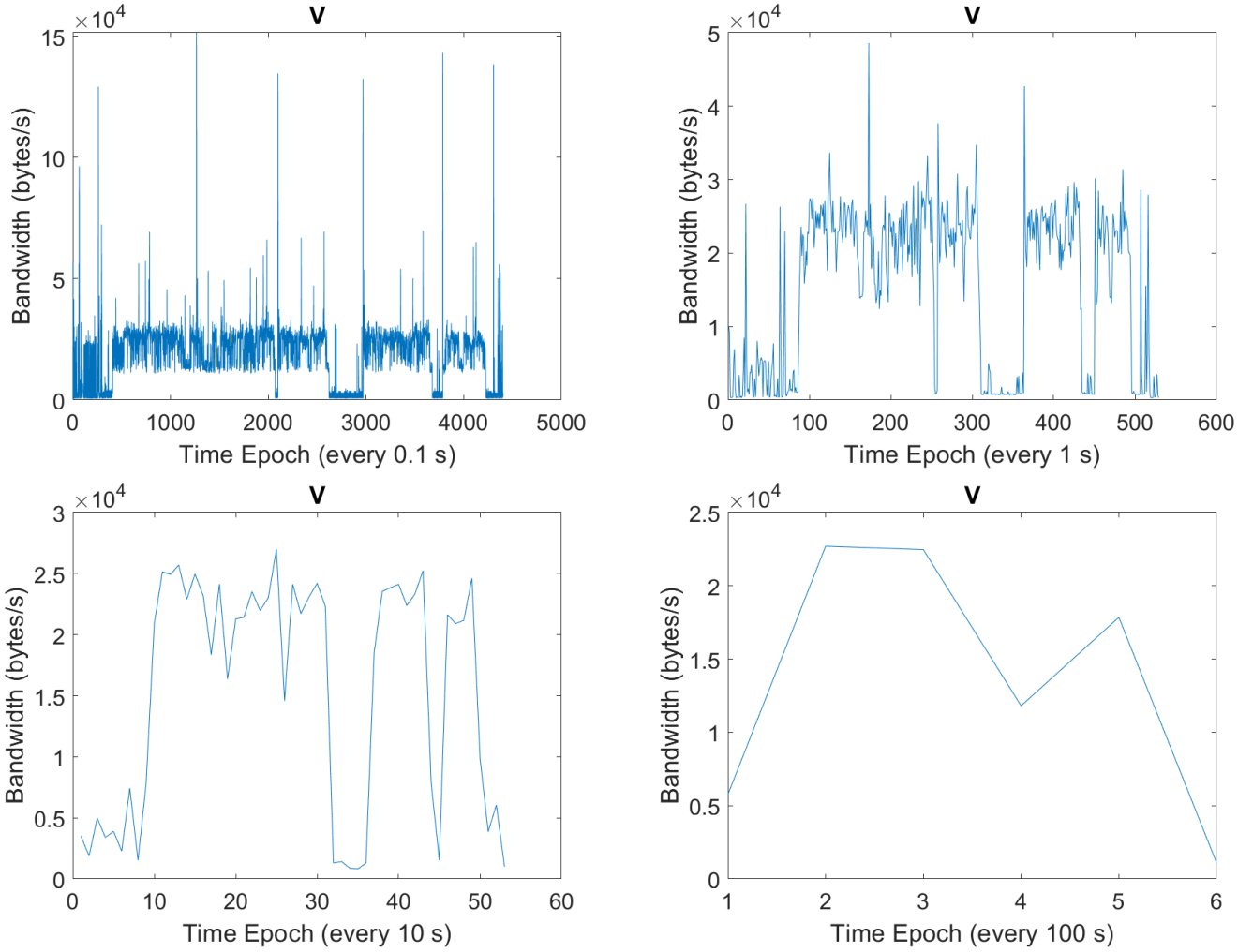

Fig. 6. Upload Bandwidth Usage analyzed over Different Scales for V-HGP Data Set. In this figure, different time epoch lengths are used, scaling exponentially from 0.1 seconds. As seen at the lower range, the bandwidth consumption is relatively stable, except when dropping to 0 at 3 points throughout the trace. These points of no bandwidth consumption can be seen affecting the plots of the longer epoch lengths.

1) Understanding Client-Server Paths: As discussed, low network latency is important for overall user experience. Currently, companies try to geographically distribute game servers such that they are relatively close to all of their users, and then, servers near the user are prioritized to host their games. Understanding how well this works would be informative. For example, we could imagine generating a sort of heat map geographically, denoting the latency to game servers, which could guide the placement of servers for a game in the future.

2) Studying More Non-isolated Traffic: While we provide a single non-isolated traffic trace and a corresponding analysis, more work could be done here. For example, we use a single composition of networked applications that we think is representative; in reality, this might not be uniform throughout the user base. Surveys could guide what application compositions could be studied. The study of non-isolated traffic would guide the development of the client side network protocols, ensuring that the developers understand exactly the use case of their application, potentially avoiding conflict with applications that the user would like to use concurrently.

3) Scaling Client Connections at Server: It would be informative to be able to study the traffic characteristics at/near the server side, particularly while scaling the number of client connections. We imagine that many of these servers would be co-located in a data center, and study the congestion characteristics of generating periodic traffic to many clients would be interesting. For example, moving all of this traffic from the internal network of the data center to the wider Internet could cause congestion within the data center, particularly if many servers end up synchronizing and sending updates to clients at the same time. Understanding this behavior as the number of clients scale and start the stress the network near the servers would be critical to continue to scale the games. There are some interesting properties of games that could exacerbate problems; for example, the fact that traffic cannot simply be routed to another server, around problem areas, because of the persistent connections between clients and a specific server.

4) Network Characteristics of Online Esport Tournaments: Traditionally, competitive tournaments are held in-person in a LAN environment, ensuring that many of the problems are solved (i.e. zero latency). However, during the COVID-19 pandemic, many of these tournaments have been held online. As these tournaments are particularly important, with teams competing for 100s of thousands of dollars, the network performance is critical for these particular games. Understanding if there is special infrastructure, not used for the normal user base, utilized during these events would be informative. If there is not special infrastructure or protocols used, studying these could guide the development of a special infrastructure to guarantee high performance during tournaments.

C. Improvements on Methodology

In this section, we point out some aspects of our methodology that should be improved.

1) Comparing Measurements with Previous Works: Being able to mathematically compare our results with the results of previous works would be a good start for building validity for our experimental environment. This could be done by using statistical tests to determine if the empirical distributions measured are coming from the same distribution. However, we would need access to the original data sets used in the previous works.

2) Isolating Game Traffic: While collecting traces, we used an ad-hoc method to ensure that there were not other applications generating traffic on the host. This leaves the possibility that the isolated traces were not perfectly isolated, which means that the data generated could contain traffic from other applications. Furthermore, we did not isolate the game traffic from protocols running concurrently on the host, like ARP. Being able to isolate the traffic more effectively would provide a better data set.

3) More In-Depth Analysis: We performed relatively simple analyses on the traces; however, it would be interesting to see what other tools we could use to analyze the data set. For example, we explored performing frequency analysis on the trace generated, but did not feel confident in how to interpret the results. Another example would be to study how the distributions used in the paper change over time.

VI. Conclusion

In this paper, we present analysis of competitive online game application traffic for a set of games, some of which that have not been studied, as well as analysis of a traffic trace of a realistic use case of one of the games, in which other networked applications are running concurrently. Specifically, we provide an analysis of inter-arrival and -departure time distributions of packets, packet size distributions, and bandwidth usage. While this is preliminary work, this project starts to build an understanding of the behavior of a large set of competitive online games, released in the past decade. The end goal of building this understanding would be to better the overall performance of online games; this is critical because the user experience hinges on a high quality of service from the network.

In future work, exploring more realistic workloads at the client would provide a better understanding of the traffic that each client would be injecting into the Internet. At the server side, exploring how scaling the number of concurrent clients changes the traffic patterns would be informative, particularly due to the expectations of the users that the number of players on a single server continues to scale past 100.

As online gaming continues to grow, it will become increasingly important for the developers to take a principled approach to the design, implementation, and evaluation of the networked infrastructure supporting their user base. Being able to support a massive number of concurrent users (already up to 12.3 million), using a demanding application in terms of network performance, will be critical for the continued success and growth of online gaming.

References

[1] "2020 essential facts about the video game industry," Jul 2020. [Online]. Available: https://www.theesa.com/esa-research/2020-essential-facts-about-the-video-game-industry/

[2] M. Carrascosa and B. Bellalta, "Cloud-gaming:analysis of google stadia traffic," 2020.

[3] X. Che and B. Ip, "Packet-level traffic analysis of online games from the genre characteristics perspective," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 240 – 252, 2012, collaborative Computing and Applications. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804511001731

[4] K.-T. Chen, P. Huang, and C.-L. Lei, "Game traffic analysis: An mmorpg perspective," *Computer Networks*, vol. 50, no. 16, pp. 3002 – 3023, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128605003956

[5] J. Färber, "Network game traffic modelling," in *Proceedings of the 1st Workshop on Network and System Support for Games*, ser. NetGames '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 53–57. [Online]. Available: https://doi.org/10.1145/566500.566508

[6] W.-c. Feng, F. Chang, W.-c. Feng, and J. Walpole, "A traffic characterization of popular on-line games," *Networking, IEEE/ACM Transactions on*, vol. 13, pp. 488 – 500, 07 2005.

[7] A. Gibson, "Most played games in 2020, ranked by peak concurrent players," Aug 2020. [Online]. Available: https://twinfinite.net/2020/08/most-played-games-in-2020-ranked-by-peak-concurrent-players/8/

[8] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen, "Gaminganywhere: An open cloud gaming system," in *Proceedings of the 4th ACM Multimedia Systems Conference*, ser. MMSys '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 36–47. [Online]. Available: https://doi.org/10.1145/2483977.2483981

[9] P. Moll, M. Lux, S. Theuermann, and H. Hellwagner, "A network traffic and player movement model to improve networking for competitive online games," in *2018 16th Annual Workshop on Network and Systems Support for Games (NetGames)*, 2018, pp. 1–6.

[10] 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.