A Network Traffic and Player Movement Model to Improve Networking for Competitive Online Games

Philipp Moll, Mathias Lux, Sebastian Theuermann, Hermann Hellwagner
Institute of Information Technology
Alpen-Adria-Universität Klagenfurt
{firstname}.{lastname}@aau.at

***Abstract*—The popularity of computer games is enormously high and is still growing every year. Despite the popularity of gaming, the networking part of computer games relies on decade old technologies, which have never been intended to be used for low latency communication and are often the cause for overloaded and crashing game servers during peak hours. In order to improve the current state-of-the-art technologies, research in the networking field has to be conducted, but is challenging due to the low availability of up-to-date datasets and network traces. Modern networking solutions of computer games try to take the players' activities as well as geographical closeness of different players in the virtual world into account, in order to achieve a high user satisfaction while keeping the network activity as low as possible. In this paper, we analyze the Battle Royale game mode of *Fortnite* as an example for a popular online game with demanding technical requirements with respect to networking. Based on the results of our analysis, we extrapolate player movement patterns as well as network traces, which can be used to study how to improve our current networking technology for online gaming, and to investigate possibilites to replace it by novel networking solutions, such as information-centric networking.**

I. Motivation

Computer games are a huge market and especially e-sports is on the rise. *Esport charts* (ESC) reported a peak in e-sports viewers of over 125 million in Oct. 2017[1]. The broad impact of gaming is also reflected by the progress that has been made in hardware and software. Real-time rendering hardware is affordable for many people and a broad range of tools and middleware like game engines allow game developers to focus on creating games rather than solving technical problems. However, regarding networking aspects, games still rely on decades-old techniques. Most games transmit their time critical information, like player location or world updates, via UDP and use TCP for matchmaking and information presented in context of the game itself [1]. Neither of these protocols was designed for games. That shows at every big game launch, where servers are crashing under the pressure of hundreds of thousands of players and network nodes struggle to keep up.

Novel information-centric networking (ICN) architectures utilize features such as inherent multicast support and name-based information retrieval, that seem to offer interesting support for multiplayer online games. In accordance with other authors (e.g. [2]–[4]), we hypothesize that ICN architectures may be better suited to fulfill the requirements of multiplayer online gaming. The motivation for this paper is to provide the basis for research on using the characteristics of Named Data Networking (NDN) [5] for multiplayer online games. NDN is one implementation of an ICN and will be investigated w.r.t. redundancy and network latency reduction for games.

The goal of our work is to provide means to test new networking approaches in the context of online gaming. In this paper we focus on a the popular game *Fortnite* and analyze it regarding its network traffic, game mechanics that influence networking, and the average player behavior. Based on our observations we extrapolate game network traffic in combination with simulated user behavior that can be used to simulate the progress of this game for the players involved as well as the server. In addition, we provide the scripts for researchers to simulate additional game rounds and adapt them to their needs.

II. Related Work

Network traffic produced by online gaming often shows the characteristic of a thin stream, which means that the packet sizes are far below the maximum segment size of TCP and the interarrival time between packets is high. Petlund et al. [6] discovered that the use of TCP for delivering thin streams can lead to delayed data delivery due to TCP's congestion control mechanism and proposed modifications that allow faster recovery from packet loss and thereby reduce network latency.

An increased interest in the NDN architecture as an information-centric alternative to current TCP- and UDP-based solutions has emerged in recent years. The online multiplayer games Matryoshka [4] and Egal Car [3] demonstrated the possibility to use ICN architectures for online gaming. G-COPSS [2] is a system for online gaming that is based on the novel NDN architecture. The authors utilize an information-centric subscription model to request game data of opponents, which are geographically close in the virtual world. G-COPSS outperforms IP-based approaches in terms of traffic reduction, achieved by the inherent multicast functionality of the underlying NDN architecture and in terms of latency, because client updates are sent in a peer-to-peer manner from client to client without making a detour over a central server.

As we can see in the case of G-COPSS, not only network traces, but also the influence of player positions can be used

[1]https://esc.watch/, last visited 2018-02-26

978-1-5386-6098-0/18/$31.00 © 2018 IEEE

to improve state-of-the-art network solutions. However, G-COPSS relied on randomized static player positions instead of modeled user behavior and player movements. Existing network traces, such as [7]–[9] are available, but do not provide information about user behavior.

III. Analysis of Game Data

Due to the broad market of video games, there is no *most popular* game or game genre. Instead, a lot of different genres and game types are popular in one way or another. For our experiment, the first obvious choice would be e-sports, where amateurs and professionals compete in different games and there is an organized structure of games, tournament and ranking systems. But in e-sports game matches and tournaments typically run in a controlled environment to reduce the influence of computer networks, so it's not a real life scenario. Moreover, these games typically let small groups compete with each other, like 4 vs. 4 or 5 vs. 5 players. Massive multiplayer online games (MMOGs) feature a lot of players, but are very hard to simulate in terms of communication behavior as they split the game world onto multiple servers. It is thus hard to simulate server traffic based on client observations, and there is no clear indication of start and end of a game round, rather a constant flow with temporary peaks. Our aim was to find a game that is (i) currently very popular, (ii) runs on a single server instance, (iii) has a very structured game progress with a clearly defined start and end, and (iv) is played by a large number of people within the same round.

A. The Battle Royale Game Genre

The *Battle Royale* genre is a good example satisfying the conditions mentioned above. Up to one hundred players compete in a king-of-the-hill like game, where players start in a large game world and continue to eliminate each other as the game world gets consecutively smaller and smaller. The classical first person shooter mechanics are extended by trap setting and multiple construction options to build fortifications and lookouts. The game ends when there is only one player left standing. There are also group based game modes, which work basically the same way, but two or more players team up to help each other and win the game together.

A game round starts in a lobby, where players are matched to build a group of up to one hundred. As soon as the clients are in sync the game starts with a flyover that delivers the players' avatars to the game map. Players decide individually when to jump, where to skydive and when to apply the parachute, and thereby select the spot where they land. Initially every player starts with just an axe, a hammer or a similar tool to tear down buildings and collect raw materials. Weapons, ammunition, traps, shields and other equipment are scarce resources distributed over the map and the type and strength of the weapons is randomly assigned for each round. After gearing up, players either actively head out to find and eliminate other players or try to hold out as long as possible by keeping a low profile.

The progress of the game is driven by a contracting storm. The players have to stay in the eye of the storm, otherwise they lose health points. Based on a pre-defined schedule, the storm contracts and leaves a smaller and smaller circular area for players to meet. At some point the world is so small that ultimately the remaining players have to confront each other. After players are eliminated, they cannot participate actively in the game, but they can be spectators seeing through the eyes of one of the remaining players to observe the rest of the game.

Currently, there are two major games in the battle royale genre. *PlayerUnknown's Battlegrounds* (PUBG), developed by PUBG Corporation, was the first battle royale game with a large audience and extensive media coverage. *Fortnite*, developed by Epic Games and People Can Fly, started out as a co-op survival game and later added the battle royale mode after realizing general interest in the battle royale genre. Fortnite has superseded PUBG[2] in terms of popularity with up to 3.4 M concurrent players[3].

With our aim to simulate the network traffic on both server and client sides for a game, we decided to investigate Fortnite Battle Royale further and monitored the network traffic as well as typical game progressions and players' behavior. From client-based network monitoring we could infer how the communication from a single client to the server and back looks like and from the game progression and player behavior monitoring we could infer the number of players in the game at a given time point as well as their probable position.

B. Analysis of Network Traffic

The source for our network traffic analysis was acquired by capturing the network traffic of Fortnite on a single client using WinDump[4]. WinDump is the Microsoft Windows version of the Unix tool tcpdump[5]. Overall, the traffic of ten game rounds was captured. While analyzing the captured network traffic streams, we saw that Fortnite only uses UDP traffic for delivering real-time game data. At the same time, we extracted game information, such as the current game phase as well as the number of active players from captured screencasts. The monitored network traffic along with game information from one game is visualized in Figure 1.

We define *outgoing* traffic as traffic which is produced on the client and sent to the server. The number of outgoing packets/second is visualized by the red line; the average outgoing payload/packet is visualized by the yellow line. The number of outgoing packets stays relatively stable during the whole game. The average payload size depends on the player's game state and is about 43.9 bytes/packet during active play and lower while spectating another player. We further observed large outgoing payload directly after the death of the avatar,

[2]More than 2.7 M concurrent players peak on PC according to http://store.steampowered.com/stats/, last visited 2018-02-26

[3]https://www.epicgames.com/fortnite/en-US/news/postmortem-of-service-outage-at-3-4m-ccu, last visited 2018-02-26

[4]https://www.winpcap.org/windump/, last visited 2018-02-26

[5]https://www.tcpdump.org/, last visited 2018-03-13

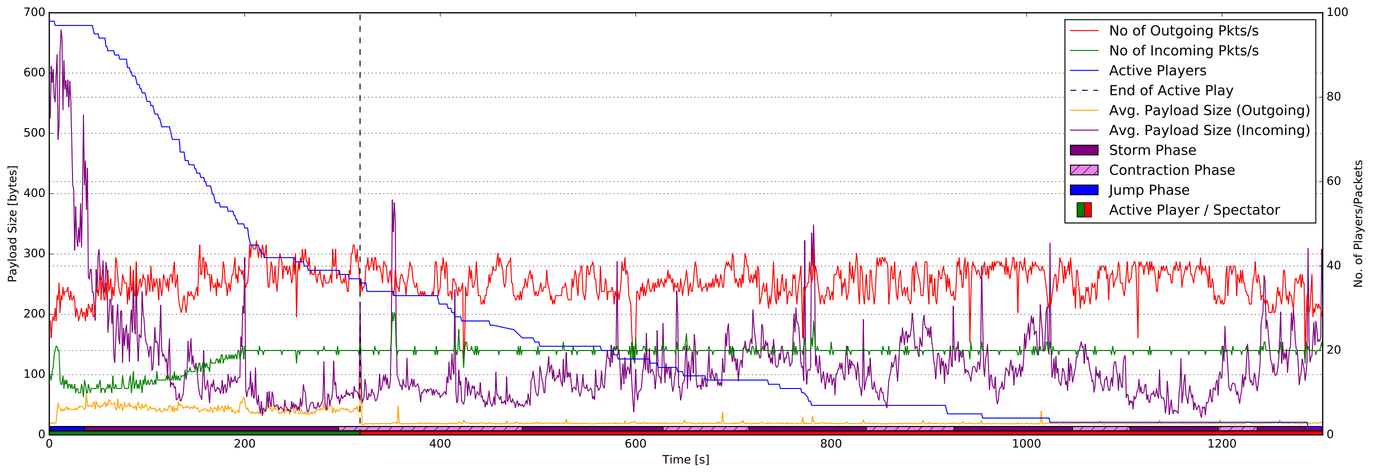

Fig. 1. Recorded network traffic of one game round on a single client, visualized together with the number of active players and game phases. The vertical dashed line shows the transition from active play to spectation of other players.

which may contain information about the player's items which are dropped into the game world immediately following the avatar's death. Another interesting observation is that even if the client acts as spectator, with no possibility to influence the game further, the average number of packets sent by the client stays the same as during the active play.

Incoming traffic is defined as traffic which is sent from the server to the client. The average number of incoming packets/second is visualized by the green line in Figure 1; the average payload size by the violet line. We noticed that the number of incoming packets/second is influenced by the number of active players. In the beginning of the game, only about 10 packets/second arrive. The number of packets is increasing until only 50 active players remain in the game, when an upper bound of 19.7 packets/second is reached. The average incoming payload size is dependent on the game activity. During the flyover phase, where players jump out of the flying bus, a lot of activity is concentrated on a relatively small area in the game world. This results in large packets in the beginning of the game, which get much smaller when more and more players have landed on the island, and stays relatively stable without encounter. During an encounter of two players, the average payload size increases significantly. Whether the player is actively playing or spectating another player makes no difference with respect to the characteristics of incoming traffic. If the spectated player encounters other players, the average incoming payload size increases in the same fashion as it does during active play. Details on the packet level network characteristics are given in Table I.

C. Analysis of Player Movements

Currently the gaming culture is obsessed with live streaming of games. There are people, called *streamers*, who play games, comment on their play and interact with the audience by reading chat messages and verbally responding to them. These live game streams can reach thousands of viewers and popular streamers make their living through sponsorship contracts and advertisements in the streams. Due to the high availability of live stream recordings, it was easy to obtain more than 36 hours of material to analyze the game progression and player behavior.

As in many first person shooter games, Fortnite offers a mini

TABLE I
OBSERVED PACKET LEVEL NETWORK CHARACTERISTICS OF FORTNITE

		Payload Size [bytes] avg. (std. dev.)	**Packets/Second** avg. (std. dev.)
Outgoing	**Normal Play**	43.9 (7.97)	35.9 (4.27)
	Spectation	19.4 (2.65)	35.9 (4.27)
Incoming	**Low Activity**	81.57 (34.47)	19.7 (1.6)
	Encounter	267.65 (57.96)	19.7 (1.6)

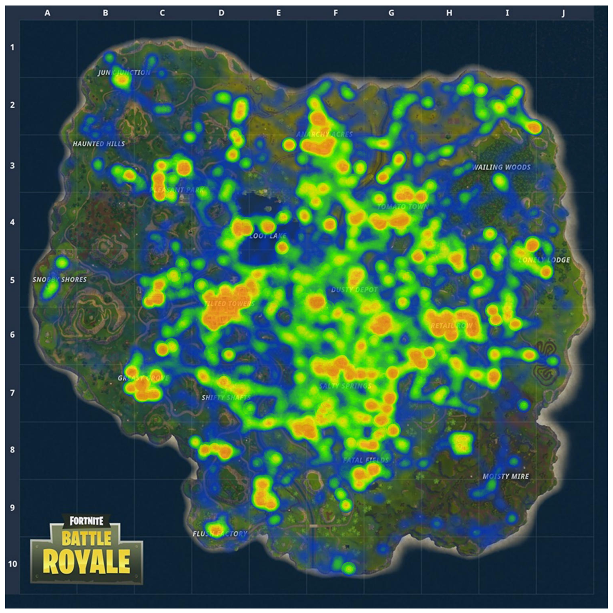

Fig. 2. Heat map from player movements in Fortnite Battle Royale generated automatically from more than 36 hours of game streams.

map in a corner of the screen for orientation of the player. The mini map is just a small portion of the overall game map centered on the player's position. The player's viewing direction and position are indicated by a triangle in the center of the mini map. By using local contrast enhancement and template matching algorithms, we can match the cutout of the mini map from game videos to an image of the overall game map and thereby find the current position of the player in the game world. We sampled the positions of players every second. With the positions of players we were able to create a heat map of player positions over different players and different game rounds. It can be seen from the heat map in Fig. 2 that players concentrate around special points of interest, e.g., locations called *Tilted Towers* or *Retail Row*, and that they are focusing rather on the center of the map than on the edges. This behavior comes from the paradigm of the contracting storm, as even with a random center of the storm the path to the center of the storm is more likely to be short if you are in the center of the map.

The storm itself follows the same pattern in every game. After an initial phase of jumping (around 30 seconds, depending on the route the bus takes), in which players are allowed to jump from the flying bus, the storm is forming for 60 seconds, where players are not influenced by the storm at all. After that the game is alternating between storm and contraction phases, which both get shorter as the game progresses. Within a storm phase, the players can see the next eye of the storm. Within a contraction phase, the storm is shrinking and players eventually have to move to stay ahead of the storm. In our observations the storm and contraction phases for the games were constant with the sequence of phases being in seconds: 200, $\underline{180}$, 150, $\underline{90}$, 120, $\underline{90}$, 120, $\underline{60}$, 90, $\underline{40}$, 90, $\underline{30}$, 60, $\underline{25}$, 60 and $\underline{25}$ (contraction phases indicated as underlined text).

IV. Extrapolation of Game Data

Our game data extrapolation results in various artifacts. We extrapolate player movements and encounters, which results in a decreasing number of active players as well as the network traffic generated during the game. The process of extrapolating game data can be structured in three phases. In the first phase, we generate the game world, including the successive positions of the eyes of the storm. Based on this information, we extrapolate movement patterns and encounters for 100 players. In the last phase, we use the movement patterns to extrapolate network traffic, which follows the characteristics of the real game traffic. In the following, each of the steps is explained in detail. We note that we provide additional insights into the process of extrapolation by making all our tools available as open-source software (https://github.com/phylib/FortniteTraces).

A. Extrapolation of Game World Properties

The game world in Fortnite Battle Royale has a fixed map with a single island in its center. In addition, the map is divided into 10×10 squares, which can be used for orientation. Our observations showed that the eye of the storm, which is used to push the game's progress, is randomly placed on the map in each game round. In the course of a game round, the eye is subject to slight movement and shrinks according to a fixed shrinking pattern. The initial eye of the storm is allowed to exceed the map borders. In our observations, a maximum of 15% of the first eye's diameter can be outside the map, which was considered when generating the eyes of the storm. The following diameters of the eyes of the storm were observed and implemented as 65%, 32%, 16.9%, 7.9%, 4%, 2% of the map size.

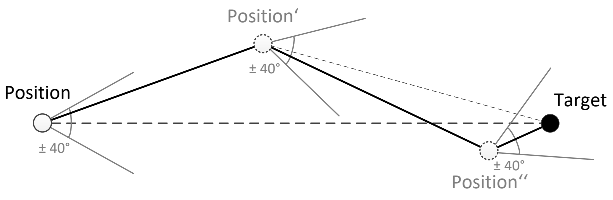

Fig. 3. Illustration of periodic course recalculation towards the current target

B. Extrapolation of Player Movements

To generate the players' movements, we used the data visualized in the heat map in Fig. 2. In a real game, players are able to land anywhere on the island, which also holds true for our extrapolated starting positions. We assume, that it is possible to land anywhere on the island, but more likely to land on a popular spot. The likelihood to land on a specific point on the island is dependent on its observed popularity. For instance, it is seven times more likely that an extrapolated player movement starts on a spot if it counts among the most popular one's, than on a spot where no movement was recorded. Furthermore, areas with high popularity were extracted as Points of Interests (PoI). For that purpose, locations with a popularity under a certain threshold were dropped in a first step. In a second step, the resulting areas were eroded and dilated in order to eliminate popular regions of insufficient size. The resulting regions are then used as PoIs.

Basically, there are three possibilities how a player is moving in the map. If it is outside the eye of the storm, it moves towards the eye. Otherwise it selects a geographically not too far away PoI inside the eye of the storm and moves towards the selected PoI. In the initial phase, where the location of the first eye of the storm is unknown, a PoI in the vicinity is selected. While moving towards a target, either the eye of the storm or the selected PoI, course deviations in the range of $\pm$ 40 degrees are allowed. Despite these deviations, we can ensure that the player reaches its destination, by periodically recalculating the walking direction, as illustrated in Fig. 3.

Another aspect that has to be considered when extrapolating player movements is the number of remaining players over time. The time between the death of two players can be modeled by means of exponentially distributed waiting times, where the corresponding parameter λ, which represents the expected number of events (deaths) per second, varies over time. In the beginning, the expected waiting time is short which results in a relatively high value for λ. The fewer players

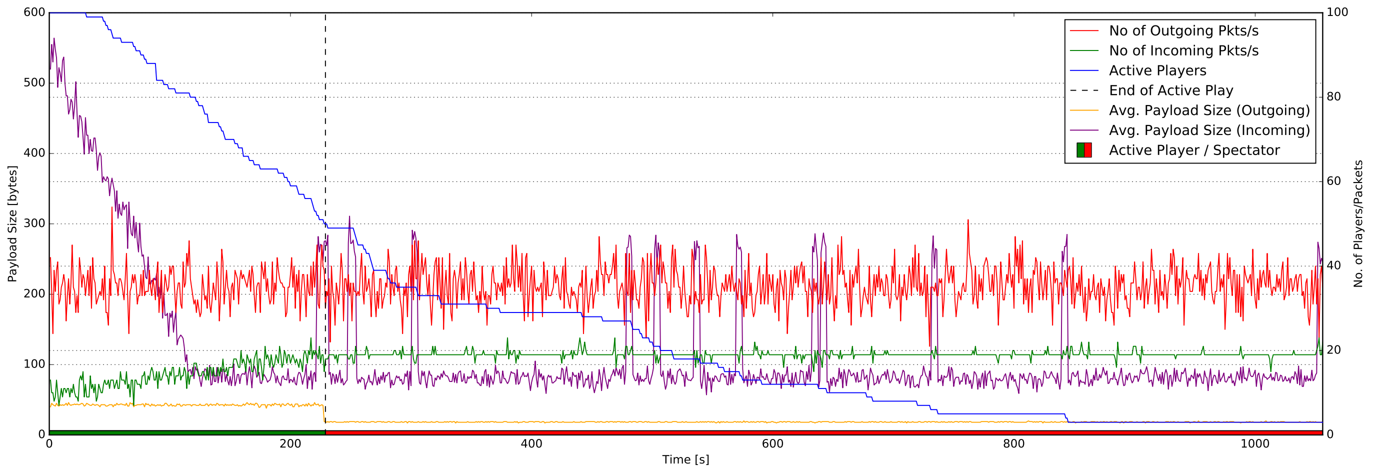

Fig. 4. Extrapolated network traffic for one player, including the number of active players during the game

remain, the longer the expected waiting time, which results in decreasing λ values. We estimated the expected number of deaths per second for different game phases based on our observations and verified the plausibility of our model by comparison of it's output with real game data. The player who dies is selected by searching the closest pair of points among all extrapolated player positions, because we assume a strong correlation between geographical closeness of players in the virtual world and their deaths.

C. Extrapolation of Network Traces

As described in Section III-B, game data is only sent between server and client, so there is no client to client communication, and only uses the UDP protocol. These facts ease the generation of network traces. Basically, for each client only two data streams need to be extrapolated, an incoming stream from the server to the client, and an outgoing stream from the client to the server. We generate data streams for 100 players per game, beginning with the start of the game and ending with the end of the game, which means that only one player is remaining. To determine the times of encounters and deaths, we use the extrapolated player movements from the previous section. After the player, for whom the current trace is generated, is eliminated, the player then acts as spectator and watches the game of his superior opponent until he dies, where the view is again switched to the view of the superior opponent, which is repeated until the end of the game.

The characteristics of a client's outgoing network traffic are modeled using normal distributions, with parameters matching the mean payload size, average number of packets/second and the given standard deviations of the analyzed network traces from Section III-B. As observed, the payload size of outgoing packets is higher during the active game. Modeling incoming traffic is more tricky. In the beginning of the game, the average number of incoming packets/second is low and steadily increasing until only 50 active players are left in the game. We modeled this ramp-up by linear interpolation, starting with 10 packets/second in the beginning of the game. As soon as the long term average of 19.7 packets/second is reached, the number of packets/second stays stable, because only a few anomalies were observed in the collected network traces. The appearance of the anomalies is modeled using an exponential distribution with 15 seconds as expected time between two anomalies. The number of packets during an anomaly is modeled using a normal distribution. For modeling the incoming payload size, we also used a normal distribution. We differentiated between normal play without encounter of other players, and encounters, which are based on the extrapolated player movements. The observed ramp-down in the beginning of the game was modeled using a normal distribution, with the mean payload size linearly interpolated between 550 bytes/packet in the beginning of the game down to the observed mean payload size for playing without encounters. A visualization of the extrapolated network traffic for one player is visualized in Fig. 4.

Fig. 5 shows the calculated server traffic load obtained from the extrapolated client network traces of a single game. Incoming traffic, sent from the clients to the server is visualized in green, outgoing traffic in red. Solid lines visualize traffic load when assuming that players leave the game after five seconds as spectators, i.e., after their deaths; dashed lines visualize the traffic load when each player stays in the game until the end of the game. The solid blue line visualizes the decreasing number of active players.

When focusing on the outgoing traffic including spectators (dashed red line), we can observe bandwidth peaks which get higher the longer the game lasts. These peaks arise during encounters of two players in the game. The longer the game lasts, the more spectators view the game of the same active player and basically receive the same game data, whereby traffic peaks arise at the same time for each spectator and sum up to bandwidth demand peaks. Efficient multicasting, as provided by novel network architectures such as NDN, could

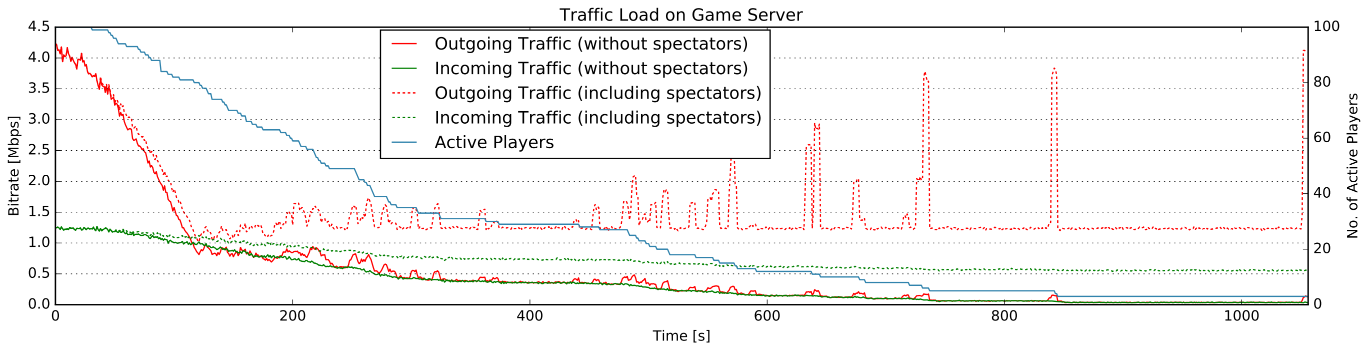

Fig. 5. Extrapolated traffic on the game server during a single game round, with players spectating until the end of the game (dashed lines), and leaving the game after five seconds as spectators/after their deaths (solid lines).

eliminate redundancies in traffic and thereby those peaks. This would lead to lower network overhead and to a lower server-side bandwidth demand. Bandwidth peaks of the solid red line are smaller, because fewer spectators are observing the game, which means that hardly any data has to be sent redundantly. Ultimately, we hypothesize that introducing an efficient multicasting system could eliminate the difference in bandwidth consumption of the dashed and solid red lines.

V. Conclusion

In this paper, we conducted a client-side network analysis of the online game Fortnite Battle Royale and built links to game events, such as encounters of two players in the virtual world, that change networking behavior. In addition, we analyzed the playing behavior of streamers and used the insights to extrapolate player movements for 100 players per game. Based on the insights of the network analysis and the generated player movements, we extrapolated network traces for all players in a game. Scripts to extrapolate movement and network traces are published as open source software along with sample traces on GitHub (https://github.com/phylib/FortniteTraces) in order to allow other researchers to use them for their research and to extend them for their needs. A refinement of the extrapolation by using stochastic processes instead of probability distributions is planned as future work to model time-dependent characteristics more accurately.

The game at hand, Fortnite, as well as many other games have strong formal constraints based on the game mechanics and game rules. We assume that many of these games can be analyzed and models can be derived in a similar way as we did in this contribution. Moreover, we saw that games of the first person shooter genre have similar network packet characteristics [1]. Thus, we believe that other games with game mechanics and rules similar to Fortnite behave similarly also from a general networking point of view. Our analysis indicated once more that player encounters influence the network traffic. While we are not able to simulate human behavior convincingly, the effects of such encounters on the network can be modeled. While we acknowledge that the introduction of new network protocols is a long term venture, we strongly believe that analyzing the use of the network in real life cases like online gaming and research on near real life network traffic are critical and necessary steps towards a more robust and efficient Internet architecture.

Acknowledgements

We would like to thank Pål Halvorsen and Carsten Griwodz for sharing their network traces and interesting insights into the network characteristics of *Anarchy Online*. Furthermore, we would like to thank Benjamin Hackl for important discussions about probability distributions used for the extrapolation of network traces.

This work was supported in part by the Austrian Science Fund (FWF) under the CHIST-ERA project CONCERT (project no. I1402).

References

[1] 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, Jan 2012.

[2] J. Chen, M. Arumaithurai, X. Fu, and K. Ramakrishnan, “G-COPSS: A Content Centric Communication Infrastructure for Gaming Applications,” in *2012 IEEE 32nd International Conference on Distributed Computing Systems*. IEEE, Jun 2012, pp. 355–365.

[3] Z. Qu and J. Burke, “Egal Car: A Peer-to-Peer Car Racing Game Synchronized Over Named Data Networking,” NDN Technical Report NDN-0010, October 2012, Tech. Rep.

[4] Z. Wang, Z. Qu, and J. Burke, “Matryoshka: Design of NDN Multiplayer Online Game,” in *Proceedings of the 1st International Conference on Information-Centric Networking - ICN ’14*. ACM Press, 2014, pp. 209–210.

[5] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, “Named Data Networking,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 66–73, July 2014.

[6] A. Petlund, K. Evensen, P. Halvorsen, and C. Griwodz, “Improving application layer latency for reliable thin-stream game traffic,” in *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games - NetGames ’08*. ACM Press, 2008, pp. 91–96.

[7] A. Petlund, P. Halvorsen, P. F. Hansen, T. Lindgren, R. Casais, and C. Griwodz, “Network traffic from Anarchy Online,” in *Proceedings of the 3rd Annual ACM Conference on Multimedia Systems - MMSys ’12*. ACM Press, 2012, pp. 95–100.

[8] Y. Wang, C.-H. Hsu, J. P. Singh, and X. Liu, “Network traces of virtual worlds,” in *Proceedings of the 2nd Annual ACM Conference on Multimedia Systems - MMSys ’11*. ACM Press, 2011, pp. 105–110.

[9] Y. Guo and A. Iosup, “The Game Trace Archive,” in *The 11th Annual Workshop on Network and Systems Support for Games - NetGames ’12*, 2012, p. 6.