A State Saturation Attack against Massively Multiplayer Online Videogames

Blake Bryant and Hossein Saiedian
Department of Electrical Engineering and Computer Science,
Information & Telecommunication Technology Center (ITTC),
The University of Kansas, Lawrence, Kansas, U.S.A.

Keywords: Videogames, Videogame Networking, Videogame Security, Performance Measurement, Videogame QoS, Videogame Cheating.

Abstract: Online videogames have enjoyed a recent surge in popularity due to increased work-from-home policies, the popularization of high-reward game tournaments, and the prospect of players earning decent wages from streaming online content. The viability of leveraging online gaming as a source of primary or supplemental income places higher stakes on the security and suitability of the network and underlying protocols used to transport game related data. A common technique used in videogames known as "animation canceling" has been used for decades to improve player performance in competitive game play. This paper reviews the potential impact of animation canceling in terms of network traffic generated and degradation to the player experience. The paper lays the conceptual groundwork for networked videogames by describing common network architectures that facilitates competitive videogame play. Finally, a AAA gaming title is selected as a case study, using the principles established within this paper, to evaluate the effects of animation canceling on competitive game play. This paper introduces a new term "state saturation" to describe a potential lag-based attack that may be implemented via animation cancelling to starve client-server based networked videogame command messages and game a competetive edge during game play.

1 INTRODUCTION

A thriving economy of computer-based videogames has existed for well over four decades. However, a drastic spike in the popularity of highly competitive multiplayer games has propelled the industry into a new economic tier, on par with that of traditional spectator sports. Notable gaming titles, such as Fortnite, Defense of the Ancients 2 (DotA 2) and Counterstrike GO (CSGO), now offer tournament pots in the tens of millions of dollars (Webb, 2019). Several content streamers on the videogame oriented Twitch steaming service now earn seven figure salaries, with some boasting net worth in the tens of millions of dollars (Hanson, 2019). The astronomical rewards offered in such tournaments, or stream viewership, will naturally motivate unscrupulous individuals to seek opportunities to exploit the growing esports ecosystem. Within this context, the networking protocols used to facilitate online multiplayer games should be evaluated for their feasibility in a semi-trusted or contested environment.

This paper provides a brief overview of the traditional TCP and UDP protocols used in networking and evaluates their suitability for use as a "gaming" protocol. Following this overview, current network frameworks within the videogaming industry are reviewed and evaluated for their suitability within a semi-trusted or contested environment. Finally, this paper concludes with a brief case study of a popular AAA videogame, "The Elder Scrolls Online," known to exhibit exploitable features within its netcode which may afford competetive players a distinct advantage over others.

2 GENERAL APPROACHES TO VIDEOGAME NETWORK DESIGN

Fiedler's work provides an overview of the three dominant approaches toward encoding videogame data to synchronize physics engines between hosts (Fiedler, 2015). The primary goal of each of these approaches is to synchronize representations of the virtual environment between remote systems participating in the game simulation. Each approach proposed by Fiedler makes use of the UDP protocol to avoid RTT delays and congestion management features of TCP, prefer-

Bryant, B. and Saiedian, H.
A State Saturation Attack against Massively Multiplayer Online Videogames.
DOI: 10.5220/0010302002170225
In *Proceedings of the 7th International Conference on Information Systems Security and Privacy (ICISSP 2021)*, pages 217-225
ISBN: 978-989-758-491-6

ring to implement reliability through the mechanisms inherent to the network approaches. The three approaches are: deterministic lockstep, snapshot interpolation and state synchronization. Each of these approaches impact the bandwidth requirements for synchronizing physics engines between hosts, as well as their ability to tolerate network latency.

2.1 Deterministic Lock-step Networking

The deterministic lockstep approach requires the least amount of data as it merely passes user input instructions to a remote host running an identical simulation synchronized with the sending hosts. However, this smaller packet size comes at the expense of an increase in both server and client-side processing. One critical limitation of this approach is that the extreme processing demands placed on the server drastically limits the number of clients that may participate in networked games. According to Fiedler, the recommended maximum number of connections is limited to four clients to one server (Fiedler, 2015). The Unreal Engine documentation also cites lack of persistence, lack of player scalability and lack of frame rate scalability as critical limitations of this approach (Epic Games, 2012). Additionally, this approach requires games to be constrained to identical platforms due to difficulty in maintaining a deterministic state between different operating systems, hardware and compilers.

The deterministic lock-step approach achieves reliable data transmission by requiring the client to send consecutive packets containing all unacknowledged user inputs. This effectively communicates the client packet buffer to the server until it is acknowledged, at which point, the client buffer is reduced to merely unacknowledged inputs. However, unlike the function of TCP acknowledgment packets, the client is not throttled by the lack of server acknowledgments. The advantage of this approach over TCP is obvious in that the client may continue to update its buffer and the server without incurring an additional round trip time (RTT) delay. It is also worth noting that simulations running on the server must wait until input instructions are received from the client before it may proceed with the simulation. This is one of the key differences between this approach and state synchronization discussed later.

2.2 Snapshot Interpolation

The snapshot interpolation technique adopts a slightly different approach and sends a representation of the current state of the simulation between hosts. Typically, the server sends only the objects a client must draw to depict a frame representing the game world, then the client processes user input commands and sends the newly rendered state of objects previously sent from the server. This exchange will undoubtedly require more information than simply replicating user inputs. As such, bandwidth requirements increase, and games are less tolerant of network latency. Furthermore, bandwidth requirements with snapshot interpolation are dependent upon the number of game objects that must be updated making this a poor choice for simulations with many object interactions. Snapshot interpolation overcomes the unreliable nature of UDP by assuming that some packets will occasionally be lost, but gaps in data may be addressed by mathematically inferring a trend line between the two states received. For instance, if the recipient receives snapshots for state 1 and state 3, but not state 2, the recipient can simulate a transition between the two states received and avoid waiting to receive state 2.

An unfortunate side effect of this approach is the need to buffer multiple state snapshots prior to sending packets. This buffering introduces additional delay on top of network delay and is dependent upon the simulation framerate in the form of the equation latency=R/B; where R represents the framerate and B represents the number of packets to be buffered. Typically, networked games operate on a standard 60 frames per second refresh rate, meaning a buffer of three snapshots would incur a delay of 50 milliseconds in addition to the network

Despite its limitations, snapshot interpolation overcomes the issues associated with the deterministic lockstep approach, namely head-of-line blocking waiting on packets to arrive and requirements for strict control over client and sender hardware/software configurations.

2.3 State Synchronization

The state synchronization approach effectively builds upon the advances made from the previous two techniques. Namely, simulations are run on both client and server systems which render simulations in response to inputs generated on the client and are sent to the server. However, unlike the deterministic lockstep approach, the server does not wait for inputs if they do not arrive, but rather predicts what would happen based on previous inputs and reconciles differences if/when client inputs arrive. Additionally, the client-side rendering is subject to modifications based on state updates sent from the server's representation of the game world. However, unlike snapshot inter-

polation, wherein the entire game world is sent, only a subset of object updates must be sent to the client for rendering. Which objects are sent in updates is based on a priority queue. Fiedler devised a technique known as a priority accumulator that tracks the priority of objects persistently between frames and accrues additional priority based on how stale the object is, meaning objects that have not been sent recently will naturally move to the front of the queue and ensure they are eventually sent (Fiedler, 2015).

The priority accumulator essentially maintains the status of objects that must be rendered in each frame whilst applying cumulative weights to objects that are most impactful on the gaming experience. For instance, player character objects will have a high priority, projectiles in shooting games will have the highest priority (even above player characters), and static objects will have the lowest priority. The priority accumulator is an effective bandwidth reduction tool in that it may selectively send only the most important objects and defer less critical objects for future updates. This also allows this technique to adhere to strict bandwidth limitations imposed by the game engine, potentially to ensure fairness amongst clients.

3 SECURITY CONCERNS WITH NETWORKED VIDEOGAMES

The background section of this paper described the unique networking challenges and approaches videogame developers apply to delivering simulated experiences. This section evaluates potential approaches attackers may use to exploit vulnerabilities in networked videogame implementations.

3.1 Client-side Exploits

Documentation associated with each of the example implementations of the deterministic lock-step and state synchronization aproaches cited concerns with unethical players attempting to modify client-side binaries to gain an unfair advantage during game play (Carmack, 1996; Valve, 2017; Valve, 2019; Van Waveren, 2006). One of the critical weaknesses of these two approaches is that they rely on broadcasting complete state data to all clients. As such, unethical players may modify their systems to remove intended restrictions on client interpretation of state data.

For example, the real time strategy game Starcraft II, and other similar titles, implement a "fog of war" effect, wherein players may only see a certain portion of the map that would be visible to units within their control. However, client-side modifications to the way in which the player's version of the game renders state data could result in disclosure of the entire map, giving one player a marked advantage over others. First-person-shooter client-side exploits could include aimbots or trigger bots that generate "hit" notifications on behalf of a player once they are within range of a competitor's character.

3.2 Timing Exploits

Snapshot interpolation and state synchronization approaches implement client-side prediction models and lag compensating techniques that allow for the queuing of player inputs and replaying of past server-side state models to detect "would be" collisions from delayed player input (Epic Games, 2012; Valve, 2019). This feature may be exploited by modifying the timestamp associated with player input updates, or merely adjusting the client clock time during play. This was specifically cited as a concern within the Epic Unreal Engine networking documentation (Epic Games, 2012). Unfortunately, as all of the networking approaches described in this paper rely on best effort delivery, precise timing is impossible due to the possibility of lost packets.

3.3 State Saturation

The state synchronization approach may be susceptible to a novel exploit presented in this paper known as "state saturation." State saturation refers to the process of intentionally stimulating an excessive amount of objects to be rendered within the scope of another player's simulation. As state synchronization approaches operate under the premise that only a portion of simulation world objects must be rendered by clients, and therefore consume bandwidth in periodic updates, and further more objects are determined to be in scope based on the visual perspective of a player's virtual character, an unethical player could attempt to force a disproportionately larger number of objects to be generated on a victims machine than on their own.

A notional scenario wherein this could occur would be something akin to causing an environmental avalanche, or detonating several graphically intensive explosions within a victim's view, while simultaneously out of the attackers view. In theory, the rapid introduction of new objects that must be rendered by the victim would adversely effect priority scheduling for the victim's state updates and either exhaust their bandwidth available to process moves, or inhibit their ability to render the attacker's actions in a timely manner, effectively reducing their available reaction time. This could give one player a marked advantage over

another during a competition, but would be highly dependent upon the game's state scoping model and potentially simulation specific environmental factors.

3.4 Volumetric Denial of Service

All network based game models are susceptible to bandwidth exhaustion associated with volumetric denial of service attacks. However, surprisingly, the oldest networking approach, using strict determinism, may be the best approach for thwarting such attacks. Deterministic lock-step networking approaches force the simulation to halt and await input from each client prior to progressing. An attacker attempting to exhaust a victim's bandwidth to introduce lag into their simulation could unintentionally introduce lag into their own experience as well. However, the effectiveness of this technique will vary between game implementations, depending on whether the networking model enforces strict determinism or allows for input loss.

4 CASE STUDY: THE ELDER SCROLLS ONLINE

The Elder Scrolls Online is a AAA massive multiplayer online role-playing game which claims to have accrued more than 15 million players between its launch in 2014 and January of 2020 (Talbot, 2020). The Elder Scrolls Online impacts multiple game-oriented revenue generation models. The game client software requires a one-time purchase with free access to online servers. In addition, players may optionally choose to pay a monthly subscription fee to access select desirable features of the game, which include unfettered access to periodic releases of discrete downloadable content (DLC) or quality of life improvements. Players may also choose to refrain from the subscription model and purchase select items through in game rewards. Furthermore, players may exchange real-world currency for virtual currency to purchase content.

4.1 Analysis of Social Media Data

The Elder Scrolls Online exhibits a healthy social community of players who host content on private websites or stream content via outlets such as Twitch or Mixer, all of which benefit from ad revenue associated with viewership. Actual subscriber numbers are not known to the public; however, active player counts may be extracted from statistics contained within the Steam content delivery service. Steamcharts.com reports concurrent player count typically fluctuates between 20,000 to 30,000 players online at a given time, with a peak volume of 49,000 concurrent players (Steamcharts, 2020).

A web scraping program was developed to gather data from the official user forums and support traditional SQL based relational queries to draw inferences from forum posts (The Elder Scrolls Online Forum, 2020). A total of 3.2 million messages, representing 164,000 distinct conversations, were analyzed. References to player action priority or mentions of a client-side exploit were of particular interest for this study.

Analysis of social media data indicates the Elder Scrolls Online fosters an active and vocal competitive player-versus-player (PVP) community. 6.75% of all forum topics analyzed were dedicated to discussion of PVP gameplay and 35.2% of all discussion topics containing at least one message referring to PVP activities. 34.03% of discussions contain a reference to game fairness, balance, or relative difficulty adjustments for player characters. The player community also comment on tactics or techniques used to improve performance within the game indicated by 15.6% of discussion topics containing at least one reference to the damage-per-second (DPS) statistic used to measure player performance within the game. The community also comments on the game's performance, with 3.2% of discussions referencing game performance or network lag.

DPS is often determined through skillful execution of player actions within an optimal sequence, referred to as a parse or rotation. References to this prioritized action sequence occurs in 6.2% of discussions, and a concept known as "animation cancelling," which will be expanded upon later in this paper, occurs in 3.98% of discussions.

Analysis of message composition for specific topics pertaining to DPS performance indicates that "rotation" or "parse" appears in 16.79% of messages that also contain the term "DPS". Furthermore, 10.96% of messages that reference "parse" or "rotation" also reference animation cancelling. Interestingly, 9.77% of topics pertaining to animation cancelling also contain at least one reference to "bugs" in the Elder Scrolls code. Forum discussions such as these ultimately exposed the presence of exploitable features within The Elder Scrolls Online game client that may be used by players to bypass player input throttling mechanisms covered in the following section of this paper.

4.2 Exploiting Client Side Validation- Animation Canceling and Weaving

The previous section of this paper introduced references to animation canceling. Many expert and competitive players in The Elder Scrolls Online community laud this technique as one of the " important elements of high-performance game play. One of the earliest references to the concept of animation canceling, within the official Elder Scrolls Online forums, dates back to May 2015 (Uberkull, 2015), which in itself references a quoted response from the game developer in 2014 stating that animation canceling was not intended, but isn't something that the developer intends to fix. YouTube personalities began referencing the technique as early as October of 2014, six months after the game was released to the public (Deltia, 2014). A Zenimax endorsed player representative known as "Alcast", published a player guide on his website pertaining to animation canceling in August of 2019, wherein he explains how to execute the technique as well as potential performance gains of a 50% improvement to player DPS (Naranarra, 2019).

Essentially, animation canceling relies upon players exploiting client-side validation of controller input prior to sending messages to the game server. This is made possible through The Elder Scrolls Online's "reactive" gameplay approach wherein players may input different types of actions with varying priorities. This priority-based system allows for interrupts to occur within a sequence of actions resulting in the highest priority actions being executed prior to the natural completion of the preceding action. The intended purpose is to allow players to reactively block or dodge actions of other characters within the simulation. However, player offensive actions are intended to be throttled by the game client to prevent players from merely rapidly pressing keys or clicking buttons and flooding servers with player input messages.

A control mechanism called a "cooldown" timer is implemented to prevent the same action from being executed more than once within a one-second update cycle. The cooldown timer is a hard-throttling mechanism that the player may not bypass. However, an additional "soft-throttling" mechanism is employed in the form of character animations associated with actions. For instance, a player swinging a sword typically takes longer to animate than the actual processing of the input message or the period of the action cooldown timer. The game client restricts player input from being accepted for the same type of input while an animation for the type is being executed. Figure 1 illustrates data flow for processing player input data on the game client.

Savvy players eventually learned that each player input belonged to one of the categories discussed above and could potentially be used to interrupt the animation sequence of a lower priority input. Note, this interruption did not cancel the previous input message, it merely removed the soft-throttling mechanism associated with the player waiting for an animation to conclude prior to submitting an additional input message to the server. The intended behavior of the client-side throttling mechanisms was for players to submit 1 attack action per second, with the possibility to interrupt said action with a defensive move in response to other characters. However, as multiple input types may be used as attacks, and multiple input types may be used as defenses, potential combinations of inputs could result in a continuous stream of alternating input types resulting in four times more attack messages than originally intended.

4.3 Animation Canceling Impact on Network Performance

The Wireshark protocol analyzer software was used to capture network packets during game play of the Elder Scrolls Online. Based upon the packet capture analysis, The Elder Scrolls Online makes use of TCP traffic exclusively. Overreliance on the TCP protocol to handle all client and server communication within the Elder Scrolls Online could be partially responsible for the challenges Zenimax Online Systems is facing in improving the network performance of their game, as reflected by player dissatisfaction with network performance in the official online forums.

Additional Wireshark packet captures were collected to measure variations in network traffic volume due to player actions in the game. Comparisons between player driven input were conducted within a player character "housing" instance, where no other player characters would be sending commands processed by the server. A Logitech G600 programmable mouse was used to store macros of input commands associated with various player input command sequences. 50 millisecond delays were interwoven between command inputs resulting in approximately 10 actions being sent to the game client per second.

The Elder Scrolls Online game client generates an average of 4.14 packets per second when sending state updates to the game server. Player movement, such as walking or running also exhibits this same traffic pattern. Rapid and persistent mouse clicks, indicating repetitive input commands for a "light attack" sequence, exhibit a slight increase of traffic sent to an average of 4.2 packets sent by the game client and

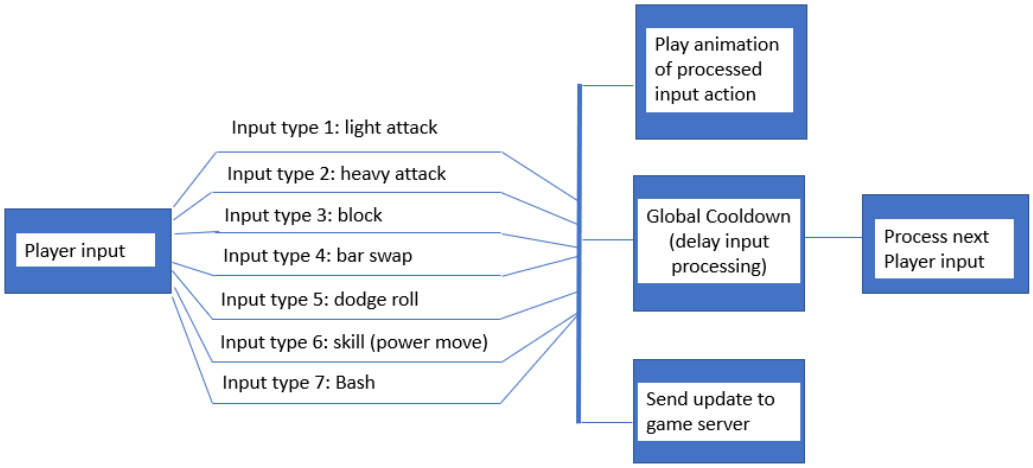

Figure 1: Elder Scrolls Online Player Input Processing.

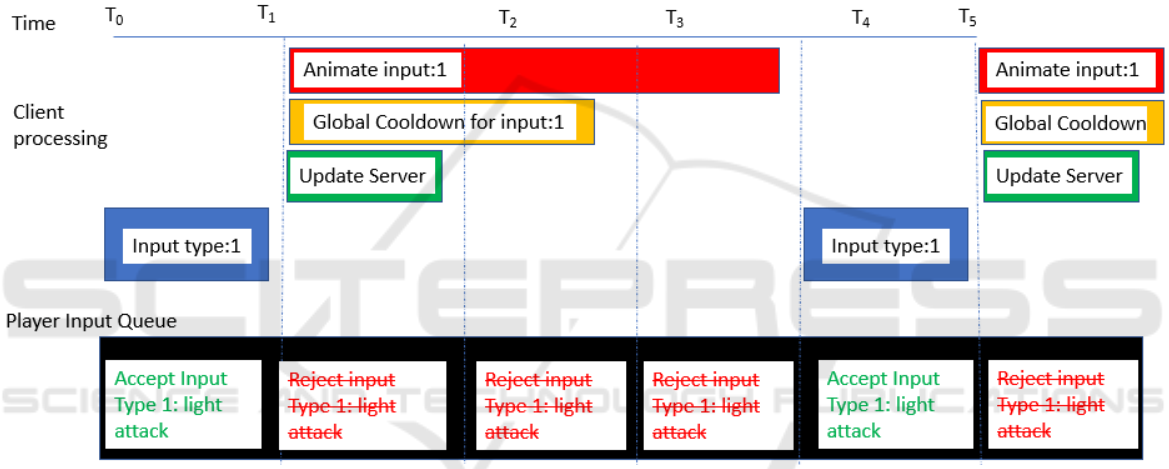

Figure 2: Elder Scrolls Online Intended Input Throttling Behavior.

4.89 packets received from the game server. This is a negligible 1.4% increase in packets sent and 18.1% increase in packets received. The increase in packets received from the server may be due to updates from the game server indicating random bonus damage inflicted by chance events associated with player actions. This mild increase in network traffic indicates that the intended game throttling mechanisms are functioning and limiting the 10 actions per second to a single player input per update cycle.

Scenarios where multiple types of offensive player inputs are alternated resulted in an average of 4.96 packets sent by the game client and 5.66 packets received from the game server. This is a 19.81% increase in packets sent and 36.71% increase in packets received. Scenarios where multiple types of offensive player inputs are alternated with defensive inputs generated an average of 11.4 packets sent by the game client and 10.9 packets received from the game server per second. This marks a 175.36% increase in packets sent and 163.29% increase in packets received over the update baseline. The latter scenario is most likely to occur in player-versus-player (PVP) type activity where players are reacting to unpredictable stimulus.

4.4 Animation Canceling Impact on Gameplay

As alluded to in the previous sections, implementing animation canceling techniques may increase the number of messages that are sent to, and therefore must be processed by, the game server. Analysis of official forum discussions indicates that player concerns pertaining to network performance are reflected within 35% of discussions about competitive player-versus-environment (PVE) gameplay and 60% of PVP game play discussions. PVP game play is expected to have a larger negative impact on network performance due to its use of alternating offensive

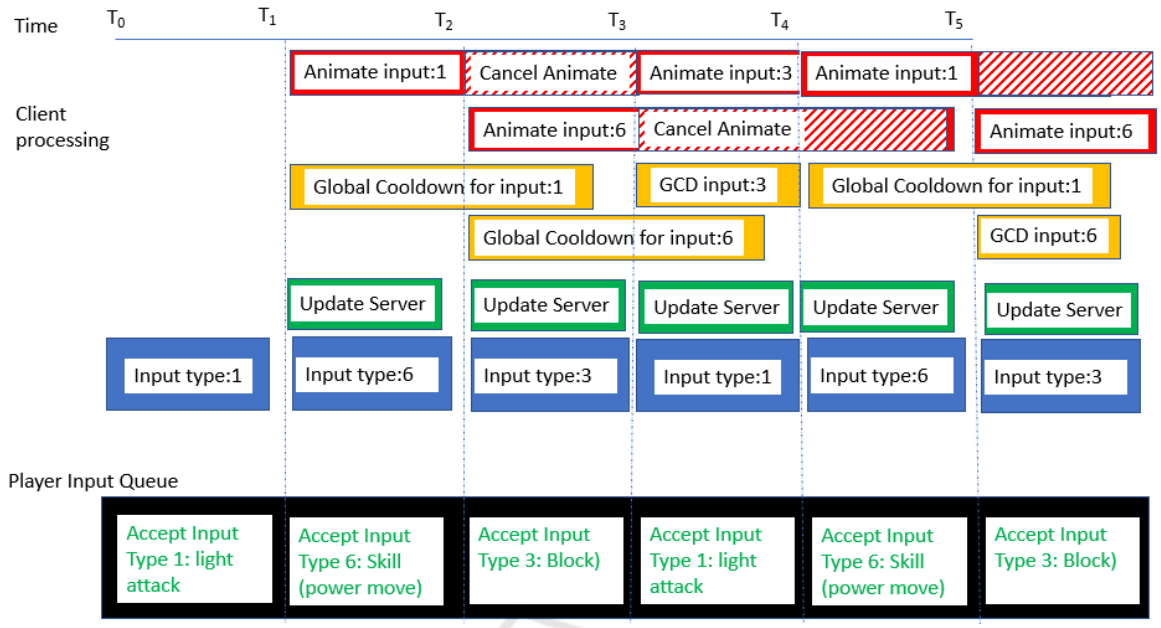

Figure 3: Elder Scrolls Online Unintended Throttling Behavior.

and defensive inputs, which appears to explain the increased occurrence of complaints by players within that sub community.

The Elder Scrolls Online also hosts a special player-versus-player (PVP) environment, named Cyrodiil, where hundreds of players may compete against one another. The official claim from Zenimax Online Systems regarding the number of players allowed within this environment is 1,800 players (roughly 600 for each of the three possible teams or alliances in the game) (Elder Scrolls Online Forum, 2015). However, it appears that the number of available simultaneous players have been adjusted over the years to account for complaints of network performance issues. It is currently estimated by the player community to be approximately 200 players per faction, for a total of 600 players at a time (Elder Scrolls Online Forum, 2016). Not surprisingly, 61% of player discussions pertaining to PVP activities on the Cyrodiil servers contain references to negative network performance.

4.5 Comparing the Case Study to Theoretical Network Models

Though the exact innerworkings of The Elder Scrolls Online network communication layer is unknown, Zenimax Online Systems publicly stated that early development of the game made use of the publicly available game engine known as the HeroEngine platform (Biessener 2012). The official HeroEngine documentation outlines the inter server communication model used to send updates between its component servers (HeroEngine (2012) . Game client information is replicated to area servers which serve as load balancers for communications to the master game server. As players' characters traverse the game world, their character model information is replicated between area servers and handoff is performed during a brief loading screen presented to the game client and player.

HeroEngine documentation indicates that bandwidth optimization techniques, similar to those described in Fiedler's state synchronization approach, are employed to decrease the size of each replication message sent between game clients as well as between area proxy servers and the master game server. Area servers track player character position information to be implemented as a factor in determining spatial awareness and determining object priority for replication. Objects that are further away from a player character object will have a lower priority in replication and may be throttled to decrease replication bandwidth. Furthermore, replication messages are optimized with bit packing so only byte streams are sent between the client and servers with special bit offsets mapped to data fields in lookup tables on game servers. This drastically decreases bandwidth consumption but makes reverse engineering of the protocol problematic.

The bandwidth optimization techniques appear to be effective in setting hard limits on player client communication rates. Client data rates are limited to the maximum amount of information the client may

communicate within a one second burst, not to exceed 40,960 bytes. Furthermore, the size of individual replication messages may not exceed 4,096 bytes. This indicates that only 10 maximum size replication messages may be sent between the game client and the game servers in a single second.

Zenimax Online Systems may have implemented a custom game engine between 2012 and 2014, the period between their leasing of the HeroEngine and the release of The Elder Scrolls Online game, however analysis of pcap data collected via Wireshark indicates there are several similarities between the Zenimax Online Systems game engine and the HeroEngine. The Zenimax Online Systems engine implemented in The Elder Scrolls Online appears to implement several different servers that the game client communicates with, as indicated by the variety of IP addresses contained within their communication. Furthermore, these IP addresses change when a player's character transitions between different areas of the game. Additionally, the number of updates messages sent between the client and the server increase in proportion to the number of dynamic game objects (such as other player characters) within proximity to the current game client-controlled character. The game engine used in The Elder Scrolls Online also makes exclusive use of the TCP protocol, like the HeroEngine.

5 CONCLUSIONS

This paper provided a brief overview of historical and contemporary approaches toward the implementation of networked multiplayer videogames. Three general approaches toward networked game development were presented: deterministic lockstep, snapshot interpolation, and state synchronization. Security concerns associated with potential player cheating were discussed within the context of these three general networking approaches. It is apparent that each game's exposure to security issues is dependent upon the chosen approach to implementing networking.

The AAA videogame title the Elder Scrolls Online was used as a case study for evaluating the potential impacts of a client-side videogame exploit known as "animation canceling." Experimental data generated from replicating common "animation cancelling" scenarios indicates that the developers of the Elder Scrolls Online appear to have optimized their network code for the most popular scenarios used in player-versus-environment (PVE) gameplay, but have not yet addressed drastic network performance impacts from a common "animation cancelling" scenario used in player-versus-player (PVP) gameplay. The combination of allowing client-side input interrupts, and the need to process defensive actions, such as player "block" commands by the game server, result in the possibility of increasing player generated network traffic by more than 175%.

Though analysis of social media data indicates "animation cancelling" resides in a small subset of discussions within official player forums, hardcore gaming personalities have demonstrated the ability to achieve up to 50% increased performance in damage-per-second (DPS) scores through skillful implementation of the technique. Furthermore, the subset of gameplay within the Elder Scrolls Online associated with PVP activity exhibits an increased frequency of complaints about network performance. It is likely that the increased occurrence of defensive interrupts within player input commands within the PVP environment, similar to actions described in section 4.3 of this paper, may be partially to blame for this suboptimal network performance.

However, the Elder Scrolls Online implementation of TCP based networking protocols may also be a key contributing factor in its degraded network performance within highly populated player areas. TCP based networking decreases the benefits of implementing some of the data loss and delay avoidance techniques described within section 2 of this paper. This may be especially relevant to the Elder Scrolls Online as it is most suited to the state synchronization approach. Furthermore, section 4.5 outlines data flow limitations implemented by the HeroEngine framework, which shaped the early development of the Elder Scrolls Online. These data flow limitations may be exploited through increased state synchronization size and velocity, referred to in this paper as "state saturation," ultimately starving game clients from sending player input information and contributing to player perceived lag spikes.

Ultimately, the authors assess that the use of "animation canceling" poses both a distinct competitive advantage to skilled players, and a considerable threat to network stability. The case study indicates that effective client validation may be used to minimize the impact "animation cancelling" may have on the network; however, failure to account for all potential user inputs and priority interrupts could result in drastic increases in network traffic for games based on the state synchronization framework.

REFERENCES

Arzyel (2019, June 22). ESO- How to Increase Your DPS — Animation Canceling Guide. Retrieved from YouTube Channel https://bit.ly/2XCAwgK

Biessener, A (2012, May 25). Why The Elder Scrolls Online Isn't Using HeroEngine. Retrieved from Gameinformer.com https://bit.ly/3gqA0dT.

Biplab, S., Kalyanaraman, S., & Vastola, K. S. (2003). Analytic Models for the Latency and Steady-State Throughput of TCP Tahoe, Reno, and SACK. *IEEE/ACM Transactions on Networking* (TON), Vol. 11 No. 6, pp. 959–971.

Carmack, J. (1996, August 16). QuakeWorld by John Carmack. Retrieved from fabiensanglard.net: http://fabiensanglard.net/quakeSource/johnc-log.aug.htm.

Claypool, M., LaPoint, D., & Winslow, J. (2003). Network analysis of Counter-strike and Starcraft. Conference Proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference, 2003. Phoenix: IEEE.

Dainotti, A., Pescape, A., & Ventre, G. (2005). A Packet-Level Traffic Model of Strarcraft. Second International Workshop on Hot Topics in Peer-to-Peer Systems. San Diego: IEEE.

Deltia (2014, October 16). ESO Animation Canceling Guide. Retrieved from Deltias gaming http://deltiasgaming.com/2014/10/16/eso-animation-canceling-guide/.

Decay2 (2020). Combat Metrics. Retrieved from https://bit.ly/33Bw1qu

Eckart, B., He, X., & Wu, Q. (2008). Performance Adaptive UDP for High-Speed Bulk Data Transfer. 2008 IEEE International Symposium on Parallel and Distributed Processing, pp 1–10.

Epic Games, Inc. (2012). Unreal Networking Architecture. Retrieved from UDK Networking Overview https://bit.ly/30BZ9Ml.

Elder Scrolls Online Forum (2015). How Will the Player Population be Limited? Retrieved from The Elder Scrolls Online Forum: https://help.elderscrollsonline.com/app/answers/detail/a_id/6533

Elder Scrolls Online Forum (2016). How Many Players are Allowed in a Campaign? Retrieved from The Elder Scrolls Online Forum: https://beth.games/2XEbL3H

Elder Scrolls Online Forum (2020). PC/Mac Patch Notes v5.3.4 - Harrowstorm & Update 25. Retrieved from The Elder Scrolls Online Forum: https://beth.games/3fE31AP

Fiedler, G. (2015). Physics for Game Programmers : Networking for Physics Programmers. Game Developer's Conference, (pp. Video Recording https://bit.ly/31z4nbj). San Francisco.

Frohnmayer, M., & Gift, T. (2000). The TRIBES Engine Networking Model. Proceedings of the Game Developers Conference. Retrieved from gamedevs.org: https://www.gamedevs.org/uploads/tribes-networking-model.pdf.

Gu, Y., & Grossman, R. L. (2007). UDT: UDP-based data transfer for high-speed. *Computer Networks*, Vol. 51, pp. 1777–1799.

Hanson, D. (2019). The 10 Richest Twitch Streamers in 2019: moneyinc.com https://moneyinc.com/richest-twitch-streamers-in-2019/.

HeroEngine (2012, October 31). HeroEngine Wiki. Retrieved from hewikiheroengine.com http://hewiki.heroengine.com/wiki/Replication_Tutorial.

Lee, C.-S. (2012). The Revolution of StarCraft Network Traffic. NetGames '12 Proceedings of the 11th Annual Workshop on Network and Systems Support for Games. Venice: IEEE.

Li, Y.-T., Leith, D., & Shorten, R. N. (2007). Experimental Evaluation of TCP Protocols for High-Speed Networks. *IEEE/ACM Transactions on Networking*, Vol. 15, No. 5, pp. 1109-1122.

Naranarra (2019). Weaving Beginner Guide Animation Canceling for Elder Scrolls Online. Retrieved from AltcastHQ https://alcasthq.com/eso-weaving-beginner-guide-animation-canceling/.

Steamcharts (2020). Elder Scrolls Online: Steamcharts.com https://steamcharts.com/app/306130.

Talbot, C. (2020). Elder Scrolls Online player count hits 15 million: PCGamesn.com https://www.pcgamesn.com/the-elder-scrolls-online/player-count.

The Elder Scrolls Online Forum. (2019) Does ESO Utilize TCP or UDP? The Elder Scrolls Online Forum. Retrieved from: https://forums.elderscrollsonline.com/en/discussion/472692/.

The Elder Scrolls Online Forum. (2020) PC/Mac Patch Notes v6.0.5 - Greymoor & Update 26: The Elder Scrolls Online Forum. Retrieved from : https://beth.games/3a7BWF9.

Thogardpvp. (2020). The Problem with Removing Animation Cancelling. Retriefed from Youtube: https://www.youtube.com/watch?v=CIzNq2exFHs

Uberkull. (2015). Animation Canceling Good for the Game? Retrieved from The Elder Scrolls Online Forum: https://beth.games/30FDeEb

Valve. (2017). Valve Anti-Cheat System (VAC). Retrieved from Steam Support: https://support.steampowered.com/kb/7849-RADZ-6869/#whatisvac.

Valve. (2019, January 8). Networking Entities. Retrieved from Valve Developer Community: https://developer.valvesoftware.com/wiki/Networking_Entities.

Valve. (2019, April 20). Source Multiplayer Networking. Retrieved from Valve Developer Community: https://bit.ly/3ks3UQS.

Van Waveren, J. (2006). The Doom III Network Architecture. Retrieved from mrelusive.com: https://bit.ly/304GBVv.

Webb, K. (2019). Business Insider. Retrieved from The Fortnite World Cup Finals start this Friday, and $30 million is on the line. Here's what you need to know about the competition: businessinsider.com https://bit.ly/36SDCAw.

Yue, Z., Ren, Y., & Li, J. (2011). Performance Evaluation of UDP-based High-speed Transport Protocols. 2011 IEEE 2nd International Conference on Software Engineering and Service Science (pp. 1–5). Beijing, China: IEEE.