Know Thy Lag: In-Network Game Detection and Latency Measurement

Sharat Chandra Madanapalli[0000−0003−0012−5295], Hassan Habibi Gharakheili[0000−0002−9333−7635], and Vijay Sivaraman[0000−0001−7985−6765]

UNSW Sydney, Australia
{sharat.madanapalli,h.habibi,vijay}@unsw.edu.au

Abstract. Online gaming generated $178 billion globally in 2020, with the popular shooter, action-adventure, role-playing, and sporting titles commanding hundreds of millions of players worldwide. Most online games require only a few hundred kbps of bandwidth, but are very sensitive to latency. Internet Service Providers (ISPs) keen to reduce "lag" by tuning their peering relationships and routing paths to game servers are hamstrung by lack of visibility on: (a) gaming patterns, which can change day-to-day as games rise and fall in popularity; and (b) locations of gaming servers, which can change from hour-to-hour across countries and cloud providers depending on player locations and matchmaking. In this paper, we develop methods that give ISPs visibility into online gaming activity and associated server latency. As our first contribution, we analyze packet traces of ten popular games and develop a method to automatically generate signatures and accurately detect game sessions by extracting key attributes from network traffic. Field deployment in a university campus identifies 31k game sessions representing 9,000 gaming hours over a month. As our second contribution, we perform BGP route and Geolocation lookups, coupled with active ICMP and TCP latency measurements, to map the AS-path and latency to the 4,500+ game servers identified. We show that the game servers span 31 Autonomous Systems, distributed across 14 countries and 165 routing prefixes, and routing decisions can significantly impact latencies for gamers in the same city. Our study gives ISPs much-needed visibility so they can optimize their peering relationships and routing paths to better serve their gaming customers.

1 Introduction

Online gaming is experiencing explosive growth: 2.9 billion players collectively contributed $178 billion to global revenues in 2020, representing a 23% growth over the year before [2]. Popular online games like Fortnite, Call-of-Duty, League of Legends and Counter-Strike account for hundreds of millions of online players. Interestingly, most of these games are free-to-play, and generate their whopping revenues from in-game purchases (in-game currency, emotes, skins, stickers, weapons, backblings, battle passes, and other such trinkets). Game publishers and platforms are therefore strongly motivated to give gamers the best possible experience to keep them engaged, and thus deploy their game servers on cloud platforms across multiple countries in an effort to minimize network latency for users.

Network latency (aka "lag") is indeed one of the largest sources of frustration for online gamers. A typical shooting game requires no more than a few hundred kbps of bandwidth, so a higher speed broadband connection does not by itself have a material impact on gaming experience. By contrast, a 100 ms higher latency can severely handicap the gamer [20], since their gunshots will be slower to take effect, and their movements lag behind others in the game. A whole industry of "game acceleration" is dedicated to address the latency issue, ranging from gaming VPNs/overlays (*e.g.,* WTFast [6] and ExitLag [1]) to gaming CDNs (*e.g.,* SubSpace [5]); indeed, one innovative eSport hosting company (OneQode [4]) has even gone to the extent of locating its servers in the island of Guam to provide equidistant latency to several Asian countries.

Internet Service Providers (ISPs), who have hitherto marketed their broadband offering based purely on speed, are now realizing that they are blind to latency. This is hurting their bottom line, since gamers are vocal in online forums comparing gaming latencies across ISPs, and quick to churn to get any latency advantage. With new game titles and seasons launching every week, and their popularity waxing and waning faster than the phases of the moon, ISPs are struggling to stay ahead to keep gamers happy, and consequently bearing reputational and financial damage.

ISPs have almost no tools today to give them visibility into gaming latencies. Traditional Deep Packet Inspection (DPI) appliances target a wide range of applications spanning streaming, social media, and downloads, and have evolved to largely rely on hostnames found in DNS records and/or the TLS security certificates of a TCP connection. Tracking modern games requires specialized machinery that can track UDP flows with no associated DNS or SNI signaling by matching on multiple flow attributes in a stateful manner. Further, game developers and publishers use different cloud operators in various countries to host their game servers, and use dynamic algorithms for game server selection depending on the availability of players and match making. These factors have made it very challenging for ISPs to get visibility into game play behaviors, limiting their ability to tune their networks to improve gaming latencies.

In this paper we develop a method to detect games, measure gaming latencies, and relate them to routing paths. Our first contribution in §2 analyzes ten popular games spanning genres, developers, and distributors. We identify key game-specific attributes from network traffic to automatically construct game signatures, and consolidate these into an efficient classification model that can identify gaming sessions with 99% accuracy within first few packets from commencement. Deployment of our classifier in a University network over a month identified 31k game sessions spanning 9,000 gaming hours, and we highlight interesting patterns of game popularity and engagement in terms of session lengths.

Our second contribution in §3 uses the servers identified using our classifier from the previous contribution to measure game servers location and latencies. We perform BGP route and Geolocation lookups, coupled with active ICMP and TCP latency measurements, to map the AS-path and latency to the 4,500+ game servers identified. We illustrate the spread of game servers across 31 ASes, 14 countries, and 165 routing prefixes, and the resulting impact on latency for each game title. We further show that different ISPs serving gamers in the same city can offer radically different gaming latency, influenced by their peering relationships and path selection preferences. Our study gives ISPs much-needed visibility into gaming behaviors and game server locations so they better optimize their networks to improve gaming latencies.

Table 1: List of games.

Game	Genre	Developer	Distributor/Publisher
Fortinite	Shooter	Epic Games	Epic Games
Call of Duty: Modern Warfare (CoD:MW)	Shooter	Infinity Ward	Blizzard Entertainment
World of Warcraft (WoW)	RTS	Blizzard Entertainment	Blizzard Entertainment
League of Legends (LoL)	MOBA	Riot Games	Riot Games
Counter Strike: Global Offensive (CS:GO)	Shooter	Valve Corp.	Steam
FIFA 20/21	Sports	Electronic Arts	Origin
Rocket League	Sports	Psyonix	Steam
Hearthstone	Card game	Blizzard Entertainment	Blizzard Entertainment
Escape From Tarkov	Shooter Survival	Battlestate Games	Battlestate Games
Genshin Impact	Action RPG	miHoYo	miHoYo

2 Game Detection

In this section, we begin by illustrating the network behavior of a representative online game (§2.1), followed by developing: (i) a method to automatically generate signatures of gaming flows (§2.2), and (ii) a deterministic classifier that combines the signatures to passively detect games using in-network attributes (§2.3). The classifier is evaluated (§2.3) and deployed (§2.4) to observe the gaming patterns in our university network.

We first collected and analyzed hundreds of *pcap* traces by playing ten popular online games (shown in Table 1) that represent a good mix across genres (*e.g.,* Shooting, Strategy, Sport), multiplayer modes (*e.g.,* Battle-Royale, Co-Operative, Player-vs-Player), and developers/distributors. These traces (labeled lab data[1]) were collected by playing games on a desktop computer in our university research lab. Next, we collected over 1000 hours of game-play packet traces selected from a full mirror (both inbound and outbound) of our university campus Internet traffic (on a 10 Gbps interface) to our data collection system from the campus border router[2]. Selected *pcaps* (labeled field data) were recorded by filtering the IP address of the game servers (to which our lab computer connected while playing). This helped us collect all game-play traffic to those "known" servers when someone on our campus played any game.

2.1 Anatomy of Multiplayer Games

Let us start with an illustrative example from a popular online game: Fortnite. It is a third person shooter (TPS) game developed by Epic Games which has risen in popularity with a game mode called Battle Royale wherein 100 players fight each other to be the last one standing. Fortnite is played by over 350 million players around the world [3]. In what follows, we outline the anatomy of a Fortnite game session by manually analyzing a packet capture (*pcap*) trace from our labeled lab data.

Gamer Interaction: A gamer first logs in to the Epic Games launcher and starts the Fortnite game client. The game starts in a lobby where users have access to their social network, collectibles, player stats, and game settings. When the user decides to

[1] Dataset available on request from the corresponding author.
[2] Ethics clearance (HC16712) obtained from UNSW Human Research Ethics Advisory Panel

play, the client contacts Fortnite's matchmaking server that groups players waiting in a queue and assigns a server on which the online game runs. Subsequently, the match starts, and its duration depends on how long the player lasts in the battle royale – the last one/team standing wins among 100 players. After the game, the user returns to the lobby area, where they may choose to start another game.

Network Behavior: From the *pcap* trace, we observe that the client communicates with various service endpoints (which can be identified by their unique domain name) for joining the lobby, matchmaking and social networking (as shown in Table 3 in Appendix A). These communications occur over encrypted TLS connections and constitute "foreplay" before game-play begins. Once the game starts, the actual game-play traffic is exchanged over a UDP stream between the client and a game server (which is usually different from the foreplay endpoints). However, the IP address of the gaming server is not resolved by DNS lookup – we, therefore, believe the server IP address is exchanged over the encrypted connection during the matchmaking process. The lack of the server identity/name (common across other game titles) makes the game-flow detection challenging. We note that the game server and other servers may or may not be co-located – *e.g.,* the game server may be very close to the user, but the matchmaking server could be operating from a different cloud in a different country.

The Fortnite game-play stream (identified using a five-tuple: SrcIP, DstIP, SrcPort, DstPort and Protocol) has a packet rate of 30-40 pkt/sec upstream and about 30 pkt/sec downstream throughout the game – fluctuations depend on player actions. However, this profile of flow rate (as used in some prior works to classify applications [12]) is insufficient to detect the game since we observed a similar pattern in other games. That being said, the UDP stream has some idiosyncratic characteristics: it connects to port 9017 on the server in our example trace; it starts with a few packets of payload size of 29 bytes; the first upstream packet contains 28 trailing `0x00`s; etc. These features, albeit simple, seem to be unique to Fortnite. The other competitive games we analyzed displayed similar patterns of user activity and interaction including contacting various services and having idiosyncratic patterns in the first few packets. We next describe methods to analyze multiple gaming flows to extract such signatures automatically.

2.2 Signature Generation

As briefly mentioned above, game-play servers typically lack DNS records, and the flow rate profile is quite similar across games. Therefore, identifying the game-play flows (among a mix of traffic) becomes challenging and requires us to inspect packets of flows for patterns. While signatures can be generated manually by playing the game to collect packet traces, we develop a method to automatically extract signatures from a collection of flows associated with game servers captured in our field dataset.

Dataset: From the lab and field packet traces (described above), we obtained over 20,000 labeled flows, with each game at least having 500 flows. We filtered and cleaned the field traces to remove non-game-play flows using simple heuristics such as flow duration (games typically tend to last for more than a minute at the very least) and protocols (excluding ICMP traffic). A flow record in our dataset contains: (i) game name, (ii) transport-layer protocol (UDP/TCP), (iii) server-side port number – *e.g.,* 9017 for the Fortnite example considered in §2.1, (iv) packet size (in bytes) arrays of upstream and downstream directions each for five intial packets – *e.g.,* up:[29,29,50,314,78] and down:[29,29,116,114,114], and (v) payload byte (in hex strings) arrays of upstream and downstream directions each for five initial packets – *e.g.,* [“`17aabb...`”,“`28a004...`”]. We note that while client-side port numbers can be useful, they are often obfuscated due to the presence of NAT and hence are not considered in this study. Further, we extract packet-level attributes from just the first five upstream and five downstream packets as they are enough to capture game-specific handshakes.

To extract game signatures from our dataset, we focused on extracting specific patterns, which could be a *static* value (consistent across all flows of a game title) or a range of *dynamic* values. To illustrate, Fortnite[3] comes with the following specific signatures: the server UDP port number is a *dynamic* value between 9000 and 9100; 1st upstream and downstream packets have a *static* size of 29 bytes ($u\_0\_len = d\_0\_len = 29$)[4]; second to tenth byte of 1st upstream packet are `0x00`. ($u\_0\_b\_1 = ... = u\_0\_b\_9 =$`0x00`)[5]

Static Signatures: We extract static signatures from *protocol*, *packet size* and *payload byte content* specific to each game title by checking if an attribute has the same value for more than α fraction of the flows. If so, the attribute and its value are added to that specific game's signature (*e.g.,* “$u\_0\_len = 29$” or “$u\_0\_b\_9 =$`0x00`”). Note that if α is set to a small value (say, 0.5), the game's signature becomes richer (containing more attributes to match) and more specific to that game. A rich signature demands more stringent requirements from a flow (*i.e.,* higher chance of rejecting a flow with minor deviation from expected attributes – resulting in false-negatives). Setting α to a value close to 1 makes the signature fairly generic, which would imply a chance of overlap with other games – resulting in false positives. We empirically tuned it at 0.90 to strike a balance and detect the games accurately. In addition, we use another parameter k to specify the depth of packet payload (in number of bytes) to be analyzed. We found that most of the static payload byte values can be captured by looking at just the first 10 bytes of each packet, meaning $k = 10$.

Dynamic Signatures: We extract dynamic signatures for *server-side port numbers* as they often do not have a fixed value but lie in a specific range of possible values (configured by their developers). Since we collected a rich set of flows in the field dataset, we use the *min* and *max* of the port numbers to identify an expected range. We further expand the range by rounding the *min* and *max* to the nearest 100 to capture those port numbers that might have missed out in our traces. Doing so gives us a signature like $port = [9000 - 9100]$ for Fortnite.

Thus, we obtain the static and dynamic signatures of each game title from a set of game flows along with parameters k and α as input. Note that signatures may overlap across games. For example, u_0_len is `29` for both games Fortnite and Call of Duty Modern Warfare (CoD:MW), shown in Fig. 9. Therefore, we need a model that can classify flows based on the attributes of packets as they arrive.

[3] A snippet of our signatures for three representative games is shown in Fig. 9 (in Appendix C)

[4] “d_0_len”: first letter denotes the direction (“d” for downstream and “u” for upstream), second letter (“0”) denotes the packet index, and third letter (“len”) denotes the packet size.

[5] “$u\_0\_b\_9$”: the letters “u” and “0” are same as above while third letter (“b”) denotes byte, and fourth letter (“9”) denotes the byte index.

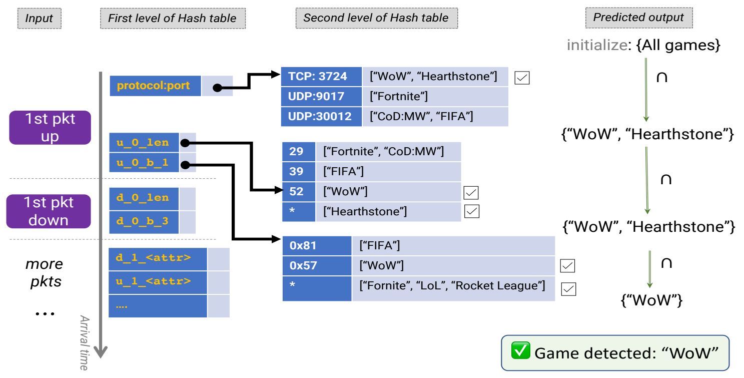


Fig. 1: The structure of our classifier, illustrating a progressive classification of a flow.

2.3 Game Classifier

We employ a *two-level hash table* (Fig. 1) that is constructed by combining all the game signatures extracted above, enabling us to rapidly detect game-play flows (and dismiss undesired traffic). The first level contains the packet attributes (*e.g.*, u_0_len, $u\_0\_b\_0$) as keys. The second level contains the possible values of the attribute as key, and possible game titles that have the same value as the entry of the hash table.

Flow of Events: Given the pre-populated hash table, we demonstrate our classification algorithm for an illustrative example in Fig. 1. We initialize the predicted output by the set of all possible games in our dataset (shown on the right side). For each incoming packet of a given flow (shown on the left side), the attributes are extracted and looked up in the hash table. For each attribute, a set of possible game classes is inferred. For an illustrative WoW (World of Warcraft) flow, upon arrival of the first packet, the protocol and port are identified as TCP:3724. Looking them up in the hash table followed by intersection with {`all games`} gives us the set {`‘‘WoW’’, ‘‘Hearthstone’’`} as output. We then proceed by looking up the packet size of 52 bytes. While 52 only yields WoW in our hash table, keep in mind that Hearthstone corresponds to a wildcard (* : indicating that attribute values were not static) meaning that the size of the first upstream packet in Hearthstone can be anything (including 52) and hence no change in the output game set. Upon extracting the second byte of the first upstream packet ($u\_0\_b\_1$) we narrow it down to WoW. When the set of games reduces to one game, we declare it as classified. Thus, the classifier rapidly eliminated other possibilities and detected a WoW game-play flow by analyzing the protocol, port, packet size, and the first few bytes of the upload packet. Note that packets' inter-arrival time in a game-play flow is in the order of milliseconds, giving sufficient room for hash table lookups (in the order of microseconds) in between packets.

We intentionally employ an algorithmic model rather than a machine learning model since the latter requires all the input attributes to be collected, stored and processed in memory to make a classification decision, which is more expensive in memory and compute. Our classifier model detects the game or rejects non-gaming flows progressively on a per-packet basis, without necessarily requiring the attributes of all ten initial packets. Whenever the possible games reduce to an empty set, we do not process packets of that flow further by classifying it as a non-gaming flow. This helps us quickly eliminate flows (often on the first packet) that do not form a part of our game set. For example, none of the games use HTTP(S), so a majority of the traffic using TCP:80 or TCP:443 is eliminated straight away and is never detected as a game. This avoids unnecessary per-flow state maintenance (no state is maintained for flows rejected on the first packet) and helps our detection method scale.

2.4 Evaluation

Our model (signatures and classifier algorithm) achieves an overall accuracy of **99.6**% (with a precision of 100% and a false negative rate of 0.36%) when it is applied to our field dataset. We found that flows of nine game titles receive a perfect accuracy 100%, while 4.5% of WoW flows are not detected as a game flow. Note that our game-specific signatures are generated based on traffic patterns found in $\alpha = 0.90$ fraction of labeled game flows; hence a minority of flows that do not conform to those signatures will not be detected as gaming flows. Our model may miss some game flows but indeed detects games correctly and confidently. We observe that the model is able to detect all games in our dataset within the first two packets (first upload and first download) as the signatures across the ten games are fairly unique, resulting in a rapid detection.

2.5 Field Deployment and Insights

The game detection system was deployed in our university campus network (with users from offices and student dormitories) during the month of Sep 2021 to obtain insights into the game playing patterns, as well as to determine corresponding gaming servers that clients connect to and their latency from our campus (discussed in §3). Our classifier (loaded with the signatures) is implemented as a DPDK [14] application running on a server which receives campus traffic mirror from optical taps (observed total traffic peak: 8Gbps). To reduce the rate of false positives in the wild *i.e.,* not detect non-gaming traffic as games, we made our algorithm more conservative to analyze all attributes of the initial ten packets of each flow before classifying the flow. Also, we monitored the activity of the flow for the first minute of its lifetime, ensuring packet rates match the expected rate of gaming flows (typically less than 100 pkts/sec).

The system detected over 31k game-play sessions, constituting nearly 9000 hours worth of game-play across the ten titles. We found that the top three games by the number of gaming sessions were CoD:MW (9545), Fortnite (7930), and League of Legends (6290). Interestingly, LoL dominated by the total number of gaming hours – LoL was played for 2611 hours, followed by CoD:MW for 1575 hours and Fortnite for 1562 hours. This highlights the games with which gamers generally engage most.

Fig. 2 shows the dynamics of daily game-play hours across the ten titles. Unfortunately there was a power outage in our lab on 14 Sep, causing data to be missed for that day. We make a couple of observations: (a) there is a slight decreasing trend of daily gaming hours during this period (more gaming hours in the first half than the second half) due to academic term starting on 13-Sep following a study break; and (b) gaming hours fluctuate across game titles – as an example, Genshin Impact (shown in brown) was more popular early in September ($\approx$ 87 hours daily), but then trended down to less than half that ($\approx$ 37 hours daily) towards the end of the month; Fortnite (shown in green) was played for 475 hours in the third week when Chapter 2 Season 8 was released, but this dropped to 325 hours in the fourth week once the excitement wore off –

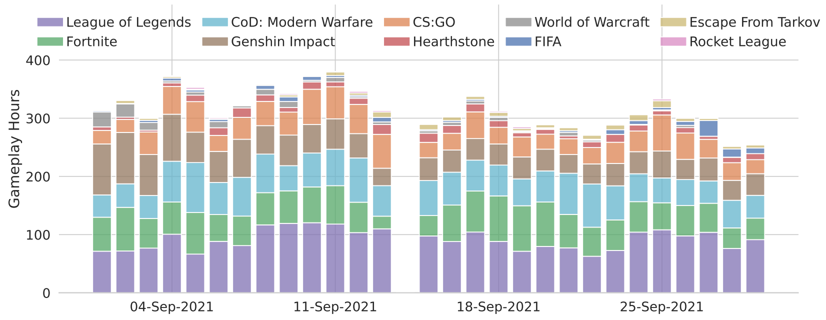


Fig. 2: Dynamics of daily game-play hours across ten titles during field trial.

such ebb and flow is the norm in gaming [17], requiring ISPs to have constant visibility so they can tune their networks accordingly.

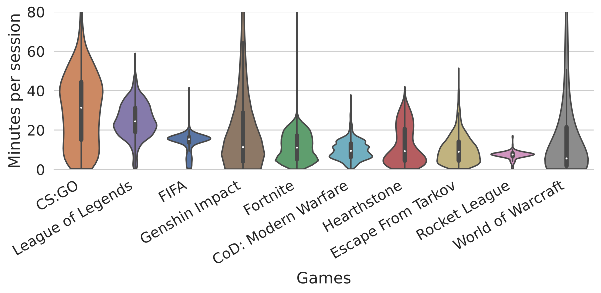


Fig. 3: Distribution of game-play session duration across the ten titles.

Fig. 3 shows the distribution of game-play session duration across the ten titles. We observe a few patterns of user engagement with various games: Several CS:GO, Genshin Impact, and WoW gamers spend more than an hour in each gaming session, with CS:GO being the most engaging game with median duration of 32 minutes. Rocket League is played for a relatively fixed duration of 10 minutes. Further, the impact of game modes is pronounced in games like CoD:MW with three bumps on its corresponding curve, highlighting three clusters of game modes, namely 5v5, GroundWar, and BattleRoyale offered by this game title.

Lastly, we analyzed short game flows (with duration less than 2 min), which can indicate game abandonment. While only 3.5% of the flows with local servers (within Australia) were short, it quadruples to more than 12% when the game is played on remote servers. Though correlation should not be interpreted as causation, it does indicate that gamers tend to abandon games more often when the latency to the server is high. The next section draws insights into game server locations and latencies.

Table 2: Summary of detected game-play sessions in our field trial.

# Game Session	**# Game Hour**	**# Game Server**	**# IP Prefix**	**# AS**	**# Country**
31673	8956	4523	165	31	14

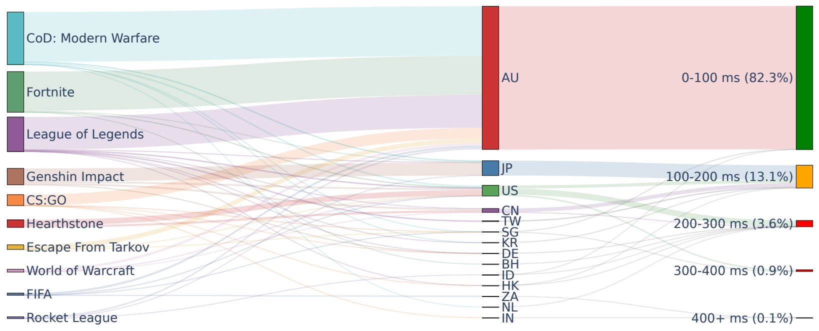


Fig. 4: Sankey diagram depicting game sessions, countries, and latency bands.

3 Mapping Game Server Locations and Latencies

Having measured gaming behaviors in the University campus over a one-month period, we now shift focus to the game servers, including their location and latency. This covers over 31k gaming sessions played against 4,500 unique game servers, spread across 14 countries and 165 routing prefixes and 31 ASes, as shown in Table 2.

3.1 Methods and Tools

We employed an IP Geolocation service [11] to tag the location of every server IP address. We also used the online Looking Glass tool exposed by the University's ISP, that offers `ping`, `traceroute`, and BGP queries to obtain routing prefix (*i.e.,* the subnet of the server IP address) and its AS path. Furthermore, we estimated the latency (we will use latency interchangeably with round-trip-time or RTT) by actively pinging the game servers ourselves. Since only 26% of the servers responded to ICMP pings, we used two additional tools – `HPing3` [18] was used to perform TCP ping using SYN packets to servers of TCP-based games (WoW and Hearthstone), and `fping` [19] to ping the entire subnet of the game server (since the entire prefix is housed in the same AS), yielding min, average, and max RTT to all servers in the subnet that respond. To corroborate the validity of (subnet) fping, we compared its average value to (endpoint) ping where available, and found the mean absolute percentage error (MAPE) to be less than 3%.

3.2 Mapping Game Servers from the University

A high level view of sessions of each of the ten game titles as they map to servers in various countries and at different latency bands is shown in Fig. 4. Most countries map to a single latency band (needless to say Australia (AU) is the home country), though some countries (like US) map to multiple latency bands, due to disparities in routing paths to multiple ASes in the same country, or to different subnets within the same AS. Specifically, 82.3% of the game-play sessions connected to servers within Australia with fairly low latency of 2-20ms, 13.1% of the sessions experienced 100-200ms, 3.6% had 200-300ms, and 1% had latency of 300+ ms.

Our measurements clearly reveal that game providers often use multiple CDNs (each identified by a unique AS number) to host their game servers – for example,

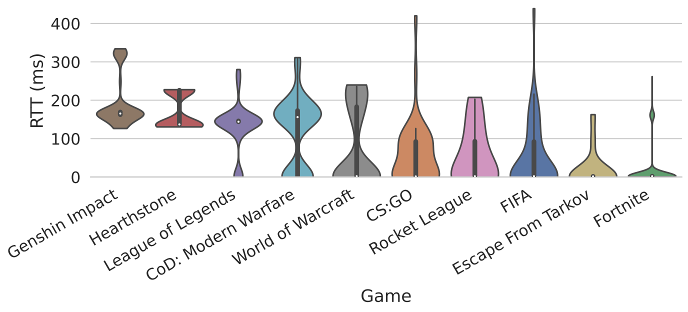


Fig. 5: Latency distribution of game servers from the campus field trial.

while Fortnite largely connects to Amazon cloud locally, some sessions connected to Google cloud in another country. There are several reasons why a gamer's session may be hosted at a server with high latency: (a) no nearer server availability; (b) there may not be enough local players available, and the player is therefore matched with players in other geographies; or (c) the player deciding to team with friends in another country, and the server is chosen in proximity to the majority of players.

To get a better understanding of gaming latency per title, we plot in Fig. 5 the latency distribution across the ten games. Fortnite and Escape from Tarkov predominantly use local servers (50ms or lower); League of Legends and CoD:MW use only a small number of local servers; while Hearthstone and Genshin Impact do not have any servers operational in the local country (the closest ones being 100+ ms away). It is also interesting to see that servers are clustered for some games (*e.g.,* Hearthstone, Genshin Impact, WoW), highlighting servers co-located in the same CDN. Curiously, though WoW and Hearthstone are from the same publisher (and share the AS owned by Blizzard), only WoW uses local in-country servers.

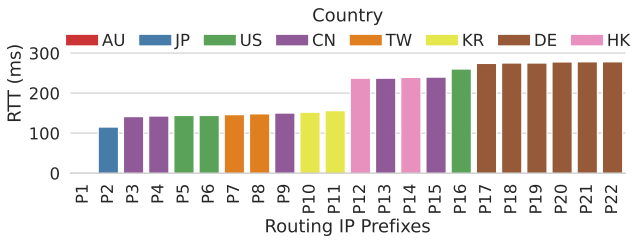


Fig. 6: Latency per IP prefix of the League of Legends servers.

To highlight the deeper dynamics of latency, we focus on League of Legends (LoL) and show in Fig. 6 the distribution of latency across various server prefixes, color-coded by their country of residence. The game connected to 293 servers located in 8 countries spanning 22 routing prefixes. We observe that it has only one routing prefix locally (P1) that offers a very low latency of under 5ms. Across other prefixes, we make a couple of observations. First, prefixes (P3, P4, P9) and (P13, P15), while located in China, belong to two different ASes and hence give very different latencies. In fact, P13 is geographically closer to P3 but the latter is one AS hop away while P13 is 3 AS hops away, leading to a latency differential of about 100ms. Second, prefixes (P5, P6, P16) belong to the same AS and are located in USA. They are all one AS away from

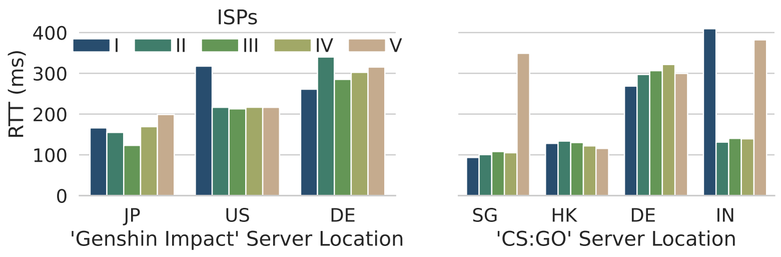


Fig. 7: Measured latency across ISPs to popular external (outside country) game server subnets of Genshin Impact (left) and CS:GO (right).

the source but P16 has a 120ms higher latency, illustrating that routing paths can vary for different subnets even within the same AS (in this case owned by Riot Games, the publisher of LoL). Further, counter-intuitively, prefix P16 is geographically closer (to game client) than P5 and P6. This analysis can help ISPs identify game server locations and routing prefixes so they can tune their peering relationships and path selections to improve latency for their gamers.

3.3 Comparing Gaming Latencies from Multiple ISPs

To better illustrate the impact of peering relationships and routing paths on latency, we performed active latency measurements (using an automated script) from several volunteers' home broadband connections in our local city to the game servers discovered in §2. The volunteers were spread across four residential ISPs (numbered *II*-*V*, with ISP-*I* representing the University), and we found that the average latency to game servers outside the country varied significantly across these ISPs, as illustrated in Fig. 7 for two representative games namely Genshin Impact and CS:GO.

Genshin Impact has no local servers, and a majority of its servers are in Japan (JP). It can be seen that ISP-*III* offers the lowest latency of 119ms while the latency is much higher (at around 198ms) with ISP-*V*. The USA serves the next higher number of sessions of Genshin Impact, and in this case ISPs *II*-*V* provide a latency of around 200ms, while the University's ISP-*I* has 300+ ms latency. For Denmark (DE), ISP *II* provides the highest latency at 322ms. Overall, a Genshin Impact gamer would get a better experience if they were with ISP-*III*. However, any ISP with this visibility into game server locations can optimise their routing paths to improve the gamer experience.

The difference across ISPs for CS:GO is even more stark, as shown in the right side of Fig. 7. In this case ISP-*V* offers a significantly worse latency to CS:GO servers in Singapore (SG) and India (IN). Given that CS:GO is a tournament-grade first-person shooting game, the latency handicap induced by ISP-*V* will be unacceptable to gamers, and likely to lead to complaints and churn. The situation is very avoidable – indeed we have reached out to this ISP, urging them to look into their peering relationships and routing path selections to address this issue.

4 Related Work

Signature Generation & Classification: Automatic signature extraction methods were first proposed a decade ago in the context of malware detection [16]. Work in [22] generates regular expressions from the payloads of un-encrypted protocols to detect application types. Bitcoding [13] proposes generating bit-level signatures for network applications by identifying bit positions that do not change in value. In a recent work [21], authors identify smart home devices and their events using signatures based on packet sizes and directions. Our methods build upon the prior work to generate signatures specific to online games. Our signature extraction is similar to [13]. However, it is faster since it looks for constant bytes in the payload instead of bits. Further, our algorithm incorporates port values in addition to byte patterns.

Gaming Applications: Several aspects of gaming applications have been studied in the past, including the impact of device-level attributes such as frame rates, and network-level parameters such as latency and loss across different games. The authors of [7] were among the first to analyze the effect of network parameters like delay, jitters, and packet loss on the game Unreal Tournament using real players. Subsequently, work in [10] analyzed multiple games using real players and [8] used bots to remove the skill bias of players. A common approach taken in these studies was to artificially induce delay/jitters/packet losses and observe the change in experience using MOS, win rates, game scores, etc. The authors in [9] studied the impact of latency on different player actions in games that have varying precision and deadline requirements. Their recent work [15] surveys the impact of different latencies on different game genres and users, concluding that gamers indeed feel the impact of high latency, especially in competitive multiplayer games. Work in [20] shows that latency is a more contributing factor than screen refresh rates in gamer performance. All prior works collectively highlight that reducing latency to gaming servers improves gaming performance and experience.

5 Conclusion and Future Work

The gaming industry is experiencing explosive growth, and ISPs are keen to offer a better gaming experience to their subscribers. However, they are hamstrung by the lack of visibility into gaming patterns, servers, and latencies. We collected and analyzed packet traces from ten popular games across various genres, extracted packet attributes, and developed a deterministic model to identify games based on automatically generated game-specific signatures. We deployed our system on live traffic of a university network, and over a 1-month period detected 31k game sessions to gain insights into game popularity and gaming engagement. We then related game latencies to routing paths by performing BGP/Geo lookups and active latency measurements to the 4,500+ game servers identified. We illustrated how the spread of games servers across ASes and countries impacts latency. Finally, we showed that ISPs serving gamers in the same city have varying latencies to these game servers, influenced by their peering relationships. While this paper studied ten popular games, we plan to evaluate the efficacy of the proposed method on a wider set of games. If conflicts arise amongst games, the classifier may require richer signatures extracted from more packets and/or deeper payload contents of individual packets. Another avenue for future work is analysis of public peering datasets to offer low-latency peering recommendations within cost budgets to ISPs.

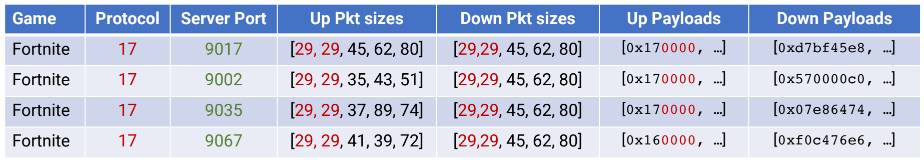
Acknowledgements We thank our reviewers and specifically our shephard, Anubhavnidhi Abhashkumar, for providing valuable feedback to improve our paper.

A Fortnite Services

Table 3: Fortnite Services, their name prefixes (suffix=`ol.epicgames.com`) and purpose.

Service	Domain Name Prefix	Purpose
Launcher	`launcher-public-service-prod06`	Epic games launcher for login
Waiting Room	`fortnitewaitingroom-public-service-prod`	The user decides the game mode
Party	`party-service-prod`	Lobby area to invite friends to play
Social Network	`friends-public-service-prod`	In-game social network
Matchmaking	`fortnite-matchmaking-public-service`	Creates matches among waiting players
Anti-cheat	`hydra.anticheat.com`	Third-party anti-cheat service
Data reporting	`data-router`	Anonymous stats reporting

B Fortnite Game Signature Generation



Game	Protocol	Server Port	Up Pkt sizes	Down Pkt sizes	Up Payloads	Down Payloads
Fortnite	17	9017	[29, 29, 45, 62, 80]	[29,29, 45, 62, 80]	[0x170000, …]	[0xd7bf45e8, …]
Fortnite	17	9002	[29, 29, 35, 43, 51]	[29,29, 45, 62, 80]	[0x170000, …]	[0x570000c0, …]
Fortnite	17	9035	[29, 29, 37, 89, 74]	[29,29, 45, 62, 80]	[0x170000, …]	[0x07e86474, …]
Fortnite	17	9067	[29, 29, 41, 39, 72]	[29,29, 45, 62, 80]	[0x160000, …]	[0xf0c476e6, …]

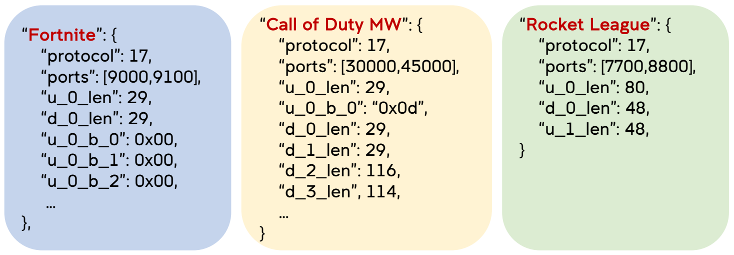
Fig. 8: An illustrative example of signature generation using Fortnite traffic traces

As shown in Fig. 8 above, each row corresponds to attributes extracted from the first few packets of Fortnite gaming flows from our dataset. The attributes include protocol, transport layer port numbers, packet sizes and payload bytes. In one flow (identified by the standard five-tuple), protocol and server port remain the same but the packet sizes and content vary as more packets arrive. For this illustration, the table shows 5 packet sizes in each direction and (stripped) payload content of the first packet.

Some attribute values (shown in red) are fixed/constant across all the flows (called *static signatures*) and other (shown in green) fall within a close range of values (called *dynamic signatures*). These signatures are same across the flows implying that they can detect a Fortnite game session. Using the static and dynamic signatures, a signature JSON is built as shown in the next section which is then used as an input to the game classifier algorithmic model.

C Example Game Signatures

Fig. 9 shows example signatures generated from our dataset. We can see that while all attributes have a key and a value, only *ports* has a range since it is a dynamic signature. We note that the complexity of signatures varies: some are primarily based on packet size (Rocket League) while others require payload bytes too (Fortnite and Call of Duty MW); some are based on attributes of first two packets (Fortnite and Rocket League) while others require more data (Call of Duty MW). These signatures need to be combined to predict the actual game being played as they may have some common attributes



```
“Fortnite”: {
    “protocol”: 17,
    “ports”: [9000,9100],
    “u_0_len”: 29,
    “d_0_len”: 29,
    “u_0_b_0”: 0x00,
    “u_0_b_1”: 0x00,
    “u_0_b_2”: 0x00,
     …
},
```

```
“Call of Duty MW”: {
    “protocol”: 17,
    “ports”: [30000,45000],
    “u_0_len”: 29,
    “u_0_b_0”: “0x0d”,
    “d_0_len”: 29,
    “d_1_len”: 29,
    “d_2_len”: 116,
    “d_3_len”, 114,
    …
}
```

```
“Rocket League”: {
    “protocol”: 17,
    “ports”: [7700,8800],
    “u_0_len”: 80,
    “d_0_len”: 48,
    “u_1_len”: 48,
}
```

Fig. 9: Signature of three representative game titles

for *e.g.,* both Fortnite and Call of Duty MW have the first upload packet length as 29 and thus require further inspection to classify the game. The classifier model takes into account all attributes and looks at the minimum number of packets to rapidly detect the game.

References

1. Exitlag (Oct 2021), `https://www.exitlag.com/en/`
2. Global Games Market to Generate $175.8 Billion in 2021 (2021), `https://newzoo.com/insights/articles/global-games-market-to-generate-175-8-billion-in-2021-despite-a-slight-decline-the-market-is-on-track-to-surpass-200-billion-in-2023/`
3. Here’s how many people play Fortnite (2021), `https://www.gamesradar.com/au/how-many-people-play-fortnite/`
4. Oneqode: The gaming infrastructure company (Oct 2021), `https://www.oneqode.com/`
5. Subspace: Dedicated network for real-time applications (Oct 2021), `https://subspace.com/`
6. Wtfast (Oct 2021), `https://www.wtfast.com/en/`
7. Beigbeder, T., Coughlan, R., Lusher, C., Plunkett, J., Agu, E., Claypool, M.: The effects of loss and latency on user performance in unreal tournament 2003. In: Proc. ACM SIGCOMM NetGames. pp. 144–151. Portland, Oregon, US (2004)
8. Bredel, M., Fidler, M.: A Measurement Study Regarding Quality of Service and Its Impact on Multiplayer Online Games. In: Proc. IEEE NETGAMES Workshop. pp. 1–6. IEEE, Taipei, Taiwan (2010)
9. Claypool, M., Claypool, K.: Latency can kill: Precision and deadline in online games. In: Proc. ACM MMSys. p. 215–222. Phoenix, Arizona, USA (Feb 2010)
10. Dick, M., Wellnitz, O., Wolf, L.: Analysis of factors affecting players’ performance and perception in multiplayer games. In: Proc. ACM SIGCOMM NetGames. pp. 1–7. Hawthorne, NY, USA (2005)
11. Dowling, B.: The Trusted Source for IP Address Data (2021), `https://ipinfo.io/`
12. Habibi Gharakheili, H., Lyu, M., Wang, Y., Kumar, H., Sivaraman, V.: iTeleScope: Softwarized Network Middle-box for Real-Time Video Telemetry and Classification. IEEE Transactions on Network and Service Management **16**(3), 1071–1085 (2019)
13. Hubballi, N., Swarnkar, M.: BitCoding: Network Traffic Classification Through Encoded Bit Level Signatures. IEEE/ACM Transactions on Networking **26**(5), 2334–2346 (2018)
14. Intel: Data plane development kit (dpdk) (2021), `https://www.dpdk.org/`

15. Jiang, C., Kundu, A., Liu, S., Salay, R., Xu, X., Claypool, M.: A survey of player opinions of network latency in online games (2020), https://ftp.cs.wpi.edu/pub/techreports/pdf/20-02.pdf
16. Kaur, S., Singh, M.: Automatic attack signature generation systems: A review. IEEE Security & Privacy **11**(6), 54–61 (2013)
17. Quwaider, M., Alabed, A., Duwairi, R.: The impact of video games on the players behaviors: A survey. Proc. of 10th International Conference ANT **151**, 575–582 (2019)
18. Sanfilippo, S.: Active network security tool (2021), http://www.hping.org/
19. Schweikert, D.: fping homepage, https://fping.org/
20. Spjut, J., Boudaoud, B., Binaee, K., Kim, J., Majercik, A., McGuire, M., Luebke, D., Kim, J.: Latency of 30 Ms Benefits First Person Targeting Tasks More Than Refresh Rate Above 60 Hz. In: Proc. ACM SIGGRAPH Asia 2019 Technical Briefs. p. 110–113. Brisbane, QLD, Australia (2019)
21. Trimananda, R., Varmarken, J., Markopoulou, A., Demsky, B.: Packet-level signatures for smart home devices. In: Proc. NDSS. San Diego, California (2020)
22. Wang, Y., Xiang, Y., Zhou, W., Yu, S.: Generating regular expression signatures for network traffic classification in trusted network management. Journal of Network and Computer Applications **35**(3), 992–1000 (2012)