Identifying MMORPG Bots: A Traffic Analysis Approach

Kuan-Ta Chen[13], Jhih-Wei Jiang[2], Polly Huang[1], Hao-Hua Chu[2], Chin-Laung Lei[1], and Wen-Chin Chen[2]
jethro@fractal.ee.ntu.edu.tw, {aslook,wcchen}@cmlab.csie.ntu.edu.tw, hchu@csie.ntu.edu.tw, {phuang,lei}@cc.ee.ntu.edu.tw

[1]Department of EE
National Taiwan University

[2]Department of CSIE
National Taiwan University

[3]Institute of Information Science
Academia Sinica

***Abstract*— MMORPGs (Massive MultiPlayer Online Role Playing Games) have become extremely popular among network gamers. Despite their success, one of MMORPG's greatest challenges is the increasing use of game bots, i.e., auto-playing game clients. The use of game bots is considered unsportsmanlike and is therefore forbidden. To keep games in order, game police, played by actual human players, often patrol game zones and question suspicious players. This practice, however, is labor-intensive and ineffective. To address this problem, we analyze the traffic generated by human players vs. game bots and propose solutions to automatically identify game bots.**

Taking *Ragnarok Online*, one of the most popular MMORPGs in the world, as our subject, we study the traffic generated by mainstream game bots and human players. We find that their traffic is distinguishable by: 1) the regularity in the release time of client commands, 2) the trend and magnitude of traffic burstiness in different time scales, and 3) the sensitivity to network conditions. Based on these findings, we propose four decision schemes and two integrated schemes to identify bots. For our data sets, the conservative scheme reduces the false positive rate to zero and achieves 90% accuracy. Meanwhile, the progressive scheme yields a false negative rate of less than 1% and maintains 95% accuracy.

***Index Terms*— Game Bot, Internet Measurement, Network Games, Traffic Analysis**

I. Introduction

MMORPGs (Massive Multiplayer Online Role Playing Games) have become extremely popular among network gamers, and now attract millions of users to play in an evolving virtual world simultaneously over the Internet. The number of active player subscriptions [17] doubled between July 2004 and June 2005 to a 500-million player base. Despite their success, one of MMORPG's greatest challenges to maintain the subscription base is the use of game bots[1].

A game bot, usually game-specific, is an automated program that can perform many tasks in place of gamers. Since bots never get tired, bot users can improperly reap rewards with less time investment than legitimate players. As this undermines the delicate balance of the game world, bots are usually forbidden in games. However, identifying whether or not a character is controlled by a bot is difficult, since a bot-controlled character *obeys the game rules completely*. The state of the practice of bot identification is to *manually* launch a dialogue with a suspect character, as a bot cannot talk like a human. But this method leads to a significant administrative burden. To the best of our knowledge, this work is the first to address automatic bot identification techniques using traffic analysis.

[1]http://en.wikipedia.org/wiki/MMORPG#Bots

Taking *Ragnarok Online*, one of the most popular MMORPGs in the world, as the subject of our analysis, we analyze the traffic of mainstream game bots and human players with different network settings. We find that traffic due to bots vs. human players is distinguishable in various respects, such as the regularity and patterns in client response times, i.e., the release time of client commands relative to the arrival time of the most recent server packet, the trend and magnitude of traffic burstiness in different time scales, and the sensitivity to network conditions.

Based on the above findings, we derive four strategies *to determine whether or not a given traffic stream corresponds to a game session played by a game bot.* Using proper combinations, we propose two integrated schemes, one conservative and one progressive, to automatically identify game bots. Our evaluation shows that the conservative scheme reduces the false positive rate to zero and achieves 90% accuracy in identifying bots. Meanwhile, the progressive scheme yields a false negative rate of less than 1% and maintains 95% accuracy. The former completely avoids making false accusations against bona fide players, while the latter tracks game bots down more aggressively.

The remainder of this paper is organized as follows. Section II describes related works. Section III provides a brief introduction of the game *Ragnarok Online* and an assessment on the current status of game bots. Section IV discusses the trace collection. Section V analyzes the discrepancies between traces for bots and human players. Then, based on our findings, we propose four bot identification schemes in Section VI. Section VII evaluates the performance of the proposed schemes, and discusses their practical use in real business operations. Finally, Section VIII draws our conclusion.

II. Related Work

As cheating is regarded as a crucial challenge to the design of online games, a great deal of effort has been devoted to cheat prevention [3, 6, 7]. While game cheats exploit loopholes in game rules or specific implementations, researchers attempt to guarantee the correctness of game systems by, for example, runtime verification of transaction atomicity [7]. Although the use of game bots is often considered as a type of "cheating," the practice is not the same as exploiting implementation bugs, because game bots *obey the game rules completely*. In other words, the integrity or correctness proof approach does not apply to bot detection problems.

Game traffic analysis, which forms the basis of our bot detection schemes, has drawn increasing attention from network researchers in recent years [1, 2, 5, 8, 9]. Among these works, Feng et al. [9] analyzed an extensive packet trace of a busy *Counter-Strike* server. Their analysis revealed that although game traffic is highly predictable, it contains bursts of tiny packets. Chen et al. [5] analyzed a 1,356-million-packet trace from a sizable MMORPG called *ShenZhou Online*. They identified some features, namely, strong periodicity, temporal locality, and diversity of user behavior, which—taken together—make MMORPG traffic distinct. This work is similar to these earlier works in the sense that we perform a thorough analysis of MMORPG game bot traffic. Instead of pursuing traffic models, however, our analysis focuses on the *distinction* between traffic patterns exhibited by human players and game bots, which is the key to designing bot identification schemes using a traffic analysis approach.

III. Ragnarok Online and The Game Bots

Ragnarok Online [12], one of the most popular MMORPGs worldwide, was created by the Gravity Corporation of South Korea in 2002. According to MMOGCHART [17], "*Ragnarok Online is supposedly the second biggest MMOG in South Korea, with well over 2 million subscribers. ... Recently they claimed 17 million worldwide with over 700,000 in North America, ...*" By any standard, *Ragnarok Online* is one of the most popular and well-known MMORPGs in the world. One reason for its popularity is the well-rendered comic-style graphics that perfectly integrate 2D and 3D graphics. The screen shot of *Ragnarok Online* in Fig. 1 shows eight characters holding a sorcery ritual. The game design encourages players to get involved with other characters and the community, which is another reason for *Ragnarok Online*'s popularity.

The core aspects of most MMORPGs are more or less standard, e.g., training characters, obtaining better equipment, and completing various quests, which usually involve fighting with monsters. Characters gradually become stronger and better equipped by gaining experience points and by accumulating loot from combat. However, repeated combat is time-consuming and can become somewhat routine and boring; thus, some players seek to set up scripts (also known as macros or bots) that can automatically and repeatedly perform assigned tasks without human involvement. Given that bots never get tired, bot users can reap huge rewards without the time investment made by other honest players.

While *Ragnarok Online* is considered as one of the most well-known MMORPGs, it is also notorious for the prevalence of the use of game bots. Since a bot can train characters and earn rewards 24 hours a day, characters trained using bots tend to be stronger than those trained *by hands*, and

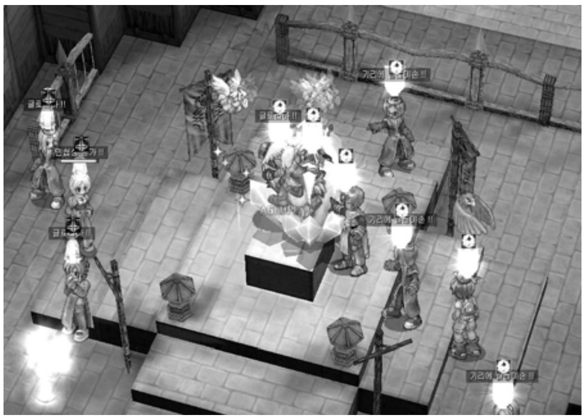


Fig. 1. A screen shot of the game *Ragnarok Online*. On the screen about eight characters are holding a sorcery ritual. Figure courtesy of http://www.Ragnarok.co.kr.

highly-prized equipments are more likely to be held by bot users. Consequently, there is always heated debate between bot advocates and opponents in *Ragnarok Online* related forums.

From the view point of game operating companies, bots erode the balance and order of the game world, as bot users can monopolize scarce resources by unleashing the indefatigable power of bots. Although companies try to prevent the use of game bots, automatic bot detection mechanisms are not currently available; thus, bot-controlled characters can only be identified *manually through human intelligence*. That is, game masters try to open online dialogues with suspicious characters; based on their responses or lack of responses, the game masters can decide if they are actually bot-controlled or human-controlled. However, this method is very inefficient given millions of online players, and leads to a significant administrative burden. The biggest restriction is that the method can only be applied if certain characters are suspected first. As the problem of the widespread use of game bots is becoming more serious, we believe the development of automatic bot identification techniques for *Ragnarok Online* and similar MMORPGs is urgent.

We surveyed publicly-available game bots for *Ragnarok Online*, and found that though more than a dozen bots are available, most of them derive from the well-known *Kore* project[2]. *Kore* is a console-based, platform-independent, and open-source bot program written in Perl and C. It could be described as the ancestor of *Ragnarok Online* bots, since many popular bots, e.g., *KoreC, X-Kore, modKore, Solos Kore, wasu, Erok, iKore*, and *VisualKore*, have been developed from it. As well as Kore, we found that another bot series, DreamRO[3] and its derivatives, is also very popular in China and Taiwan.

Both *Kore* and *DreamRO* are standalone bots, i.e., they can communicate directly with game servers without the *Ragnarok Online* game clients. Their actions are script-based, and almost every action available in the game client is covered. When a bot program is running, it continuously reports the latest information and the current status of the game, e.g., a character's

[2]Kore, http://sourceforge.net/projects/kore/

[3]DreamRO, http://www.ayxz.com/soft/1805.htm

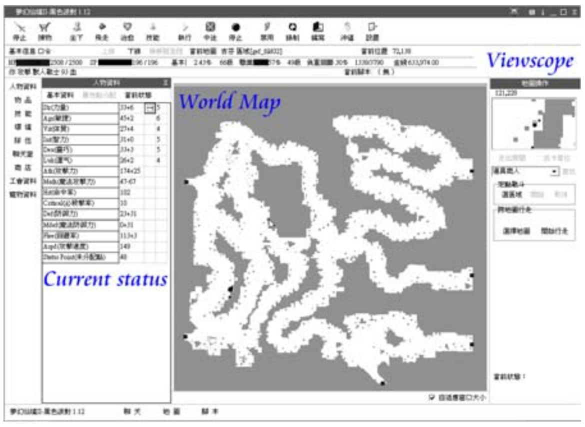


Fig. 2. A screen shot of the *DreamRO* bot.

location, the last action, the "hit" point, and information about nearby monsters. Both *Kore* and *DreamRO* allow users to give commands anytime, regardless of the prearranged actions (by scripts), i.e., the bots are both script-based and interactive. Fig. 2 shows the main window of a running *DreamRO* bot, in which the character is chasing a monster. In the picture, the world map (large map) is the current map where the character is located, and the view scope (small map) indicates there is a monster nearby. The status panel lists the current status of the character.

IV. Trace Collection

To develop bot identification techniques based on discrepancies in traffic patterns, we acquired a number of *Ragnarok Online* game traces for both popular bot series, *Kore* and *DreamRO*, and for human players. For brevity, we use "players" to denote human players hereafter. To make trace collection tractable, we chose a bot program to represent each series. *KoreC*, the Chinese edition of *Kore*, was selected as the representative of the *Kore* series, and *DreamRO* was chosen to represent the *DreamRO* series. Although only one bot was analyzed for each series, we believe bot programs in the same series behave similarly based on the same kernel implementation.

We collected a total of 19 game traces at the client side, i.e., the traffic monitor was attached to the same LAN as the game client. In order to ensure heterogeneity among the limited number of traces, we intentionally incorporated combinations of controllable factors into the trace collection. From a networking perspective, both bot and player traces contained fast (campus network) and slow (residential connection) access links, and the network media ranged from Fast Ethernet to ADSL and cable-modem. In terms of user behavior, the human players were diverse in their choice of characters and game playing proficiency: among the four players, Gino and Kiya were experienced—both of them had played *Ragnarok Online* for more than one year—and their characters were high-level ($>$ level 60), well-equipped, and highly-skilled. On the other hand, Kuan-Ta and Jhih-Wei were newcomers to *Ragnarok Online*, and their characters were low- to middle-level (level 5 and level 40 respectively) without advanced skills or powerful weapons.

The scripts we used for the two bots are commonly available among the *Ragnarok Online* community. Their actions are set to the most common "kill, loot, and trade" cycles. In other words, at the start, the bot will go to a selected area, where there are many monsters, and proactively pursue and attack the nearest monster. After the monster is killed, the bot will pick up the loot, and turn to another monster. The process will not terminate until the backpack is full of loot. At that time the bot will return to a marketplace to sell the gathered loot, and then restart the cycle. As in the human player case, we purposely ran the bots with characters of different levels of proficiency and professions.

The collected game traces are summarized in Table I. For brevity, we denote the four players as *A*–*D*, Kore as *K*, and DreamRO as *R*. Traces from the same bot/player are coded by a unique digit following the bot/player's code. The period of a game trace indicates the continuous gaming time; that is, the human players did not intentionally leave their characters idle during the game. Each game trace is composed of a number of TCP connections, where each connection corresponds to the activity within the same map. The game world of *Ragnarok Online* is partitioned into a number of maps, provided by several map servers. When a character moves across map boundaries (via walking, transport, or teleporting), the game client will disconnect from the original map server and establish a connection with the new map server. Therefore, the number of connections implies the number of map switches, which indicates how frequently a character moves across maps. The packet rate column lists the average data packet rate sent by game clients and servers. The average client packet rate is an indicator of *player activity*, since each player command is conveyed by a client data packet. On the other hand, the average server packet rate is an indicator of *interaction*, i.e., how popular and active the area where the character resides is, as server packets convey nearby characters' activities [5]. Note that the average packet rate is roughly the same for the same bot/player with the same network setting, which may be seen as a "signature" of the game playing behavior of a certain bot/player. The behavior of our selected human players can be shown to be heterogeneous by these two metrics. Also, the average round trip times (RTT) and packet loss rate statistics manifest the heterogeneity of the network QoS the traced sessions experienced.

V. Traffic Analysis: The Search for Patterns

From a traffic analysis perspective, the most intuitive discrimination between bots and players is the *release timing of client commands*. For human players, client commands, e.g., approach another character, attack a nearby monster, or cast healing magic, are triggered by keyboard strokes or mouse clicks. On the other hand, for game bots, triggering client commands is decided by the decision engine in the bot program. Thus, the design of *when to issue* the next command is important to us because it leads to major discrepancies in traffic patterns between different bot series, and between bots and human players.

TABLE I
GAME TRAFFIC TRACES (8 HUMAN PLAYER TRACES AND 11 BOT TRACES)

Category	Player	ID	Network*	Period	# Conn	Pkt	Bytes	Pkt Rate‡	Avg RTT	Loss
Human Player	Gino	*A1*	HiNet 2M/512Kbps	1.8 hr	12	51,823	3.5 MB	0.9 / 3.9 pkt/s	82.0 ms	0.03%
		A2		5.6 hr	14	147,814	10.5 MB	0.8 / 3.4 pkt/s	95.4 ms	0.03%
	Kiya	*B1*	APOL 2M/512Kbps	0.4 hr	45	15,228	1.0 MB	1.2 / 4.5 pkt/s	81.6 ms	0.01%
		B2		2.3 hr	108	59,247	3.8 MB	1.1 / 3.3 pkt/s	108.8 ms	0.12%
		B3		2.1 hr	189	47,721	3.2 MB	0.9 / 2.8 pkt/s	125.5 ms	0.23%
		B4		5.0 hr	326	129,177	8.4 MB	1.1 / 3.3 pkt/s	109.8 ms	0.09%
	Kuan-Ta	*C1*	ASNET†	0.8 hr	2	9,681	0.6 MB	0.8 / 1.4 pkt/s	191.8 ms	1.73%
	Jhih-Wei	*D1*	TANET	2.4 hr	28	48,617	3.2 MB	0.8 / 2.6 pkt/s	45.1 ms	0.01%
Bot	Kore	*K1*	TANET	13.4 hr	104	245,709	13.6 MB	0.7 / 2.3 pkt/s	33.0 ms	0.01%
		K2		26.5 hr	306	479,374	30.4 MB	1.0 / 2.1 pkt/s	45.6 ms	0.04%
		K3		32.7 hr	37	271,416	13.3 MB	0.6 / 0.7 pkt/s	96.5 ms	0.004%
		K4	ETWEBS-TW	13.0 hr	38	225,528	11.5 MB	0.9 / 2.0 pkt/s	65.7 ms	0.01%
		K5		5.7 hr	31	110,883	6.0 MB	1.1 / 2.1 pkt/s	90.6 ms	0.20%
	DreamRO	*R1*	TANET	3.0 hr	7	46,381	2.6 MB	0.9 / 1.7 pkt/s	83.4 ms	0.03%
		R2		4.8 hr	21	77,675	4.4 MB	0.9 / 1.9 pkt/s	65.2 ms	0.02%
		R3		42.3 hr	42	652,877	34.1 MB	0.8 / 1.8 pkt/s	85.3 ms	0.05%
		R4	ETWEBS-TW	11.2 hr	77	320,686	25.1 MB	1.7 / 3.5 pkt/s	85.2 ms	0.05%
		R5		23.1 hr	176	672,325	53.3 MB	1.7 / 3.6 pkt/s	79.4 ms	0.16%
		R6		10.5 hr	36	209,347	13.1 MB	1.0 / 2.4 pkt/s	87.7 ms	0.05%
Total	2 B / 4 P	19		206.6 hr	1,599	3,821,509	241.6 MB			

* This column lists network names looked up using WHOIS service.
† Access link bandwidth: ASNET (2M/512Kbps), ETWEBS-TW (2M/256Kbps), and TANET (100Mbps)
‡ Packet rate column format is "client data packet rate / server data packet rate," i.e., pure TCP ack packets do not count.

Following an analysis of the release timing of client commands from bots ("bot command" for short), we find that the release of commands, for both *Kore* and *DreamRO*, depends on the following events: 1) *timer expiration*, and 2) *server data packet arrivals*. The use of periodic timers is intuitive and reasonable, since many actions in a game are iterative in nature, e.g., continuous slash until an enemy is defeated. A bunch of successive commands are also usually implemented with timers, e.g., when the life point is lower than a certain threshold, a character must immediately drink a healing potion, and then cast protective magic on himself and offensive magic on the most threatening enemy. Using a timer to schedule the above commands sequentially with certain intervals is the most common design. On the other hand, since server data packets carry the latest status of the character and environment, e.g., the current life point of the character, the movement of nearby monsters, and whether the last slash hit the enemy, bots often respond to server data packets by issuing new commands. For example, to pursue a fleeing enemy, a bot can continuously issue movement commands whenever it receives the latest location information about the enemy from the server data packets.

In the following, we analyze the traffic traces of game bots and human players, and search for distinctive traffic patterns exhibited by bots, but not by players, and vice versa. The analysis of traffic patterns comprises four steps. First, based on intuition, we inspect the regularity embedded in the traffic. Second, we examine the timing of client commands relative to the arrival time of the most recent server data packet. We then observe the traffic burstiness of the packet traces. Lastly, we identify particular patterns of human behavior caused by sensitivity to network conditions, which, of course, is not possessed by game bots.

Since our analysis focuses on application-level interaction between game clients and servers, pure TCP acknowledgement packets, i.e., packets without a data payload, are excluded from the analysis. For the sake of brevity, we use "client packets" to denote data packets sent out by game clients, excluding pure TCP acknowledgement packets; "server packets" are similarly defined. In addition, since we concentrate primarily on client packets, the term "packets" denotes "client packets," unless otherwise specified.

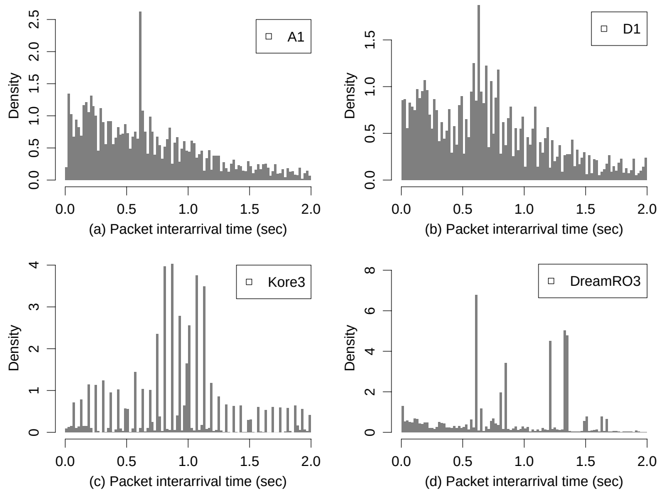


Fig. 3. Histogram of packet interarrival times

A. Regularity in Client Traffic

1) Packet Interarrivals: Fig. 3 shows the histograms of packet interarrival times shorter than 2 seconds. While player traces in the upper two plots show randomness in packet interarrival times, the bot traces, shown in the lower two plots, suggest the existence of a timer triggering mechanism and the absence of randomness that characterizes human actions.

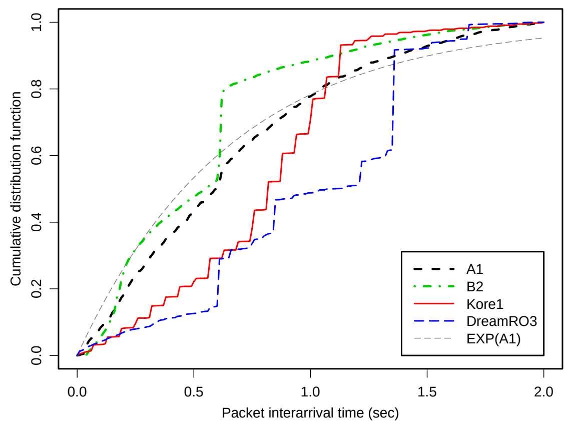


Fig. 4. CDF of packet interarrival times

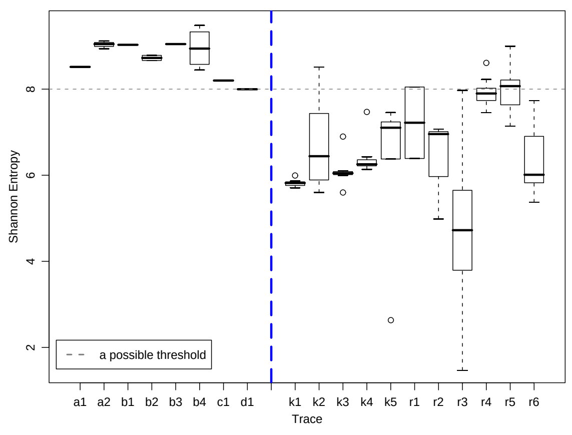


Fig. 5. Entropy computed from packet interarrival times for each trace. Each group of $5,000$ inter-packet times is computed separately. The threshold is chosen arbitrarily to reveal that the entropy of player traffic is almost always higher than that of bot traffic.

Specifically, *Kore3* reveals a periodic timer of 16 Hz, i.e., most of the packet interarrival times are multiples of $1/16$ second, while *DreamRO3* displays more regular timing, as most of the inter-packet times concentrate on certain values.

An empirical cumulative distribution function (CDF) plot of packet interarrival times manifests the above statements more clearly. In Fig. 4, the CDF curves of player traces, *A1* and *B2*, increase smoothly, except for a sudden rise around 0.6 seconds, which is a frequency component inherent in game clients. We also provide the CDF curve of an exponential random variable fitted to *A1* by maximum likelihood estimation (MLE). Though the exponential curve does not fit the empirical CDF of *A1* very closely, it can be seen an approximation. We will exploit this property to detect excessively fast packet bursts in Section V-C.2. On the other hand, the curves of *Kore1* and *DreamRO3* show *zigzag* patterns, which strongly suggests that packet interarrivals in both bot traces are concentrated around certain times.

2) Entropy of Packet Interarrival Times: From the above observation, we find that the distribution of packet interarrival times is more regular for bot traffic than player traffic. We now attempt to exploit this property to distinguish bot traces from player traces.

A well-known metric for judging the degree of randomness of a random variable is its degree of entropy. We start with the definition of Shannon entropy, a traditional measure of the uncertainty in a random variable. The Shannon entropy, $H(x)$, of a discrete random variable, x, that takes on the value v_i with probability p_i is defined as:

$$H(x) = -\sum_i p_i \log_2 p_i .$$

We compute the entropy of each trace by segments, i.e., the inter-packet times of each trace are first divided into several segments, and the entropy is computed for each segment separately. The computed entropy with segment size $5,000$ for all traces is depicted in Fig. 5. The result conforms to our observation that the entropy of player traces is mostly higher than that of bot traces. We provide a possible threshold on the plot so that the entropy of player traces is higher than the threshold, while the entropy of bot traces is lower. However, choosing an appropriate threshold is difficult, because we cannot decide how large is "large" for the computed entropy. Furthermore, if we compute the entropy with larger segments, the entropy between bot and player traces will be less distinguishable. This is because with more packets, and therefore more interarrival times, there is more chance of randomness in the packet interarrival times of bot traces. For these reasons, we do not use the entropy of packet interarrival times to distinguish between bots and human players.

3) Frequency Components: When considering the time factor, we take successive packet interarrival times in the same trace as a time series. We find that in some bot traces, packet interarrival times occur periodically in a statistical sense. For example, Fig. 6(a) depicts 100 successive inter-packet times in the trace *DreamRO3*. On the graph, there is a pronounced pattern, wherein one or two large packet gaps of approximately 2.5 seconds occur about every 10 packets. Such a regular pattern is a consequence of bot behavior for certain tasks; for example, in its monster-hunting process, a bot will move in a randomly chosen direction for certain steps, and repeat the steps until a monster is within its view scope. This time series can be viewed in the frequency domain by a transform to the corresponding power spectral density function, as shown in Fig. 6(b). On the graph, frequency components of 0.1 Hz and 0.2 Hz are clearly present, where the 0.1 Hz frequency corresponds to the 10-packet-period in Fig. 6(a). Note this phenomenon appears in both the *Kore* and *DreamRO* traces, but it is not present in player traces. However, not all bot traces exhibit pronounced frequency components, since the proportion of regular behavior is small compared to the whole trace. Therefore, we do not consider that periodicity in packet interarrival times is an effective method for recognizing game bots.

We can use another metric to check the periodicity in traffic, i.e., the frequencies embedded in packet arrival processes. For each trace, we obtain the corresponding packet arrival process by counting the number of client packets released every 0.1 seconds. Since for each trace, at every instant exactly one

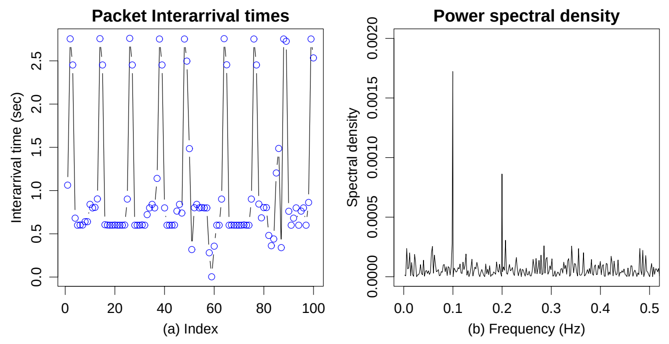


Fig. 6. (a) *DreamRO3* exhibits regular packet interarrival times, such that, on average, one or two large packet-gaps occur for every 10 packets. (b) Power spectral density of packet interarrival times

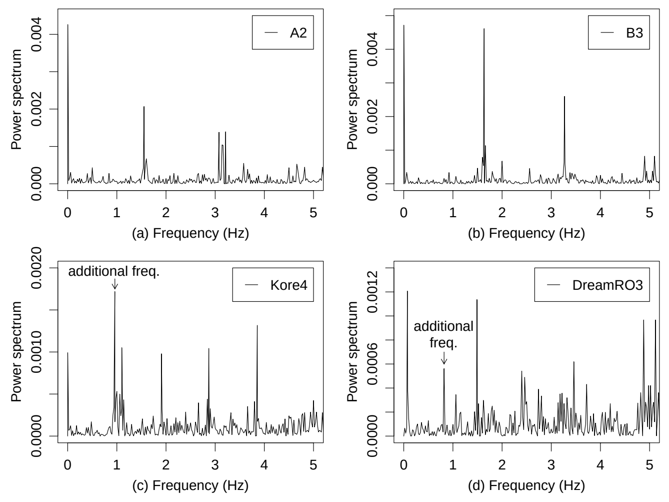


Fig. 7. Power spectral density of packet arrival processes. Frequencies embedded in player traces are alike; however, bot traces display additional frequency components that do not appear in player traces.

connection is active, we argue that the corresponding packet arrival process is just stationary, regardless of the rate variation during game playing. A preliminary check of player traces shows that at least three strong frequencies are inherent in the game design, namely, $1/12$ Hz, 1.67 Hz, and 60 Hz. This behavior echoes the study of another MMORPG [5], *ShenZhou Online*, which shows that significant frequency components exist in game traffic in either direction. Comparing the power spectrum of bot traces with that of player traces, we find that bots induce additional frequency components not present in player traces, as shown in Fig. 7. However, since the frequencies in game traffic may be adjusted based on a character's attributes [5], e.g., race, skill, or equipment, and since we do not have complete knowledge of all possible frequencies built into the design of *Ragnarok Online*, we cannot decide whether a frequency component is *inherent* in the game design or *induced* by a bot program. For this reason, we do not use frequency components to identify game bots.

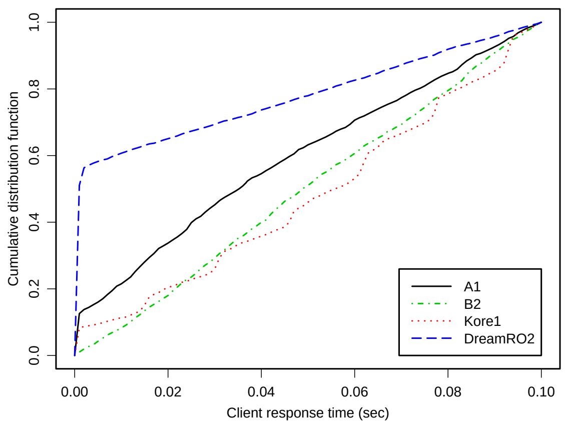


Fig. 8. CDF of client response times

B. *Command Timing*

Having investigated the regularity in client traffic due to timers, we now examine the regularity of client commands that respond to server packets. We start with the definition of "client response time" as the *time difference between a client packet departure time and the most recent server packet arrival time*, if no other client packets intervene; otherwise, the metric is undefined. Since we do not consider the corresponding server response time, for brevity, we use "response time" to denote client response time hereafter.

By the above definition, for each trace, we compute the client response times for those client packets that immediately follow a server packet. As an initial assessment of whether the response times of bot traces differ significantly from those of player traces, cumulative distribution functions of response times of less than 0.1 seconds for selected traces are plotted in Fig. 8. In the figure, except for an initial rise for *A1*, the two player traces, *A1* and *B2*, are similar in that their response times less than 0.1 seconds increase smoothly, i.e., they are almost uniformly distributed. On the other hand, bot traces reveal different patterns: the CDF of *Kore1* is a zigzag-type, i.e., the response times are clustered around certain intervals, while that of *DreamRO2* has a strong mode with very small response times. In the following, we discuss these two properties of bot traces, i.e., strong modes and zigzag CDF, in more depth.

1) Prompt Response of DreamRO*:* Among all game traces, only those of *DreamRO* possess a considerable number of short response times, e.g., ≤ 1 ms, which we call *prompt responses*. These responses are frequent enough and clustered so that more than one peak is formed in the corresponding histogram, as shown in Fig. 9. Note that to distinguish peaks clearly we take a logarithm of the response time. The prompt response manifests that the design of *DreamRO* often issues client commands immediately upon the receipt of server packets, while *Kore* employs a more sophisticated command timing mechanism.

2) Regularity: Although prompt responses are not present in the traces of *Kore*, it still relies on server packet arrival events to schedule the release of client commands. In Fig. 10,

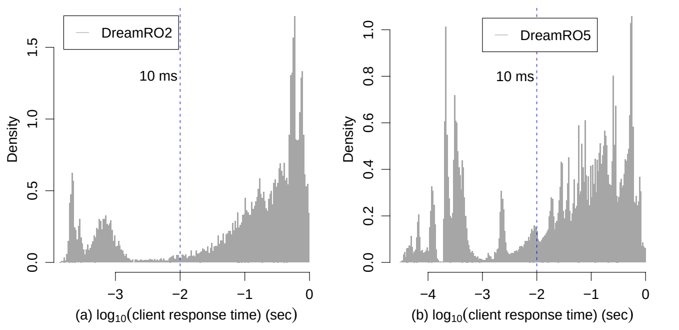


Fig. 9. Histograms of client response times in two *DreamRO* traces. More than one peak is formed at time scales smaller than 10 ms. Peaks do not occur in player traces or *Kore* traces.

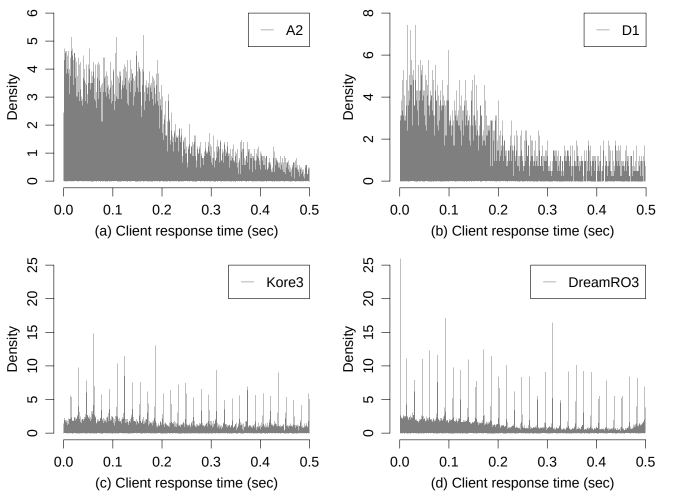


Fig. 10. Histogram of client response times shorter than 0.5 seconds

which depicts histograms of response times shorter than 0.5 seconds, both bot traces show *spiky* densities, while player traces do not present any visible patterns. These plots indicate that both *Kore* and *DreamRO* schedule their client commands by an intentional delay time following receipt of a server packet. In the histogram, if the bin width is small enough, the distance between *spikes* will reflect the smallest scheduling unit of the command departure times; according to our traces, the value is set to 15 ms for both *Kore* and *DreamRO*. In Section VI-A, we develop a bot detection scheme based on the prompt responses and the regularity in response times identified above.

C. Traffic Burstiness

Traffic burstiness, i.e., the variability of traffic volume, or the packet count sent in successive periods, is an indicator of how traffic fluctuates over time. While traffic burstiness is commonly related to the scaling property of a traffic stream or used to detect how *bursty* a traffic stream is, we use it to assess how *smooth* the bot traffic is. Our hypothesis is that a bot, by virtue of its periodicity (as identified in Section V-A), should exhibit smoother traffic compared with player traffic.

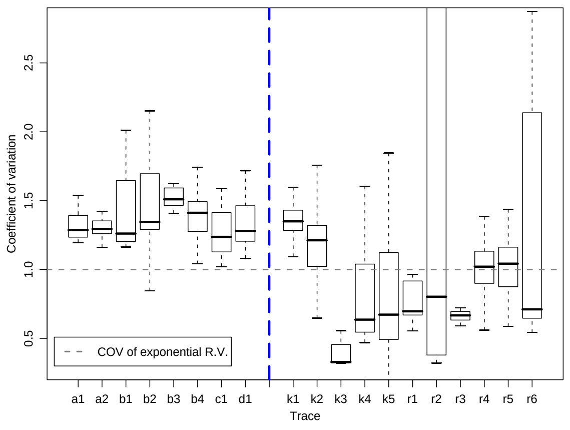


Fig. 11. Coefficient of variation computed from packet interarrival times for each trace. Each group of 500 inter-packet times is computed separately.

There are several commonly used metrics of traffic burstiness. In the following, we first evaluate the coefficient of variation (COV) of packet interarrival times, and then use the index of dispersion for counts (IDC) to capture the variability of traffic over different time scales.

1) Coefficient of Variation: The coefficient of variation (COV) of packet interarrival times is defined as the ratio of the standard deviation of the interarrival times to the expected number of interarrival times. The COVs of selected game traces, computed with a segment size of 500, are plotted in Fig. 11. By definition, the COV of any exponential random variable is equal to 1, as its standard deviation is always equal to its mean. From the graph, the average COVs of player traces are all higher than 1, while most bot traces have COVs lower than 1. However, we do not consider that the COV is an effective indicator for differentiating bots and players because of randomness, as in the entropy method (Section V-A.2): for larger segments, all game traces tend to have COVs higher than 1, so the boundary between bot traces and player traces is difficult to determine. Thus, in the following, we use a more sophisticated method to measure traffic burstiness: by characterizing it in various time scales.

2) Index of Dispersion for Counts: Like all other programs, a bot program must have a *main loop*, where an iteration of the loop corresponds to a minimum unit of operation, e.g., issuing a command for a character, or processing a server packet. The rationale behind multi-time scale burstiness analysis is that, assuming each iteration (of the main loop) takes approximately the same amount of time, and in each iteration the game bot sends out roughly the same number of packets, then *traffic burstiness will be lowest at the time scale equal to the amount of time needed for each iteration of the main loop*.

To measure traffic burstiness in different time scales, we use the index of dispersion for counts (IDC). The IDC at time scale t is defined as the variance in the number of arrivals in an interval of time t divided by the mean number of arrivals in t [13], that is,

$$I_t = \frac{\mathrm{Var}(N_t)}{E(N_t)}, \qquad (1)$$

where N_t indicates the number of arrivals in an interval of

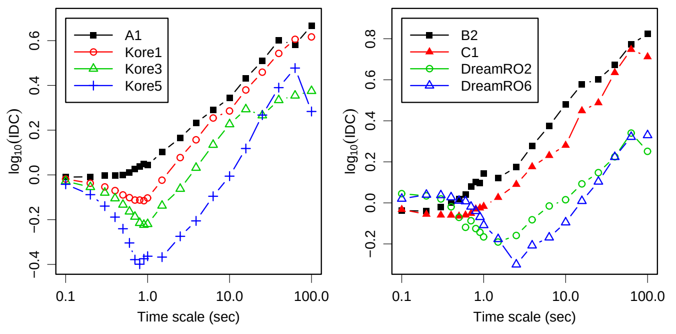


Fig. 12. IDC plots for different traces. Player traces are represented by filled symbols; bot traces are represented by unfilled symbols. Note the trend of IDC for bot traces has a "dip" around 1 second.

time t. The IDC is thus defined so that, for a Poisson process, the value of the IDC is 1 for all t.

When we first compute the IDC for the game traces, we find that *DreamRO* traces exhibit surprisingly high burstiness in all time scales. Our analysis shows that, occasionally, the packet rate of *DreamRO* is extremely high, e.g., sending 8 packets in 0.1 seconds and 15 packets in 1 second, while a human player sends a maximum of 3 packets in 0.1 seconds and 8 packets in 1 second in our traces. The unusually high rates cause unreasonably high traffic burstiness, which prevents us from determining the true burstiness of *DreamRO* traffic. To overcome this problem, we propose a simple heuristic to restrict the packet rate in different time scales. Since, in Section V-A.1, we observed that packet interarrivals of a human player trace can be approximated by exponential distributions, we can compute a 95% confidence band of the expected number of packets by an exponential random variable given the mean rate λ. Specifically, we first infer λ from the whole trace, then use the Poisson assumption to compute a confidence band of the packet count at time scale t as $(high_t, low_t)$, and modify Eq. 1 by computing $N_t = min(N_t, high_t)$.

The IDCs for selected game traces, with the above rate regulation formula applied, are plotted in Fig. 12. We make two observations from the plots: 1) bot traffic is smoother than player traffic, but it is hard to define a threshold for the burstiness magnitude; and 2) all bot traces support our hypothesis that they have the lowest burstiness at time scales around 0.5–2 seconds. In other words, the burstiness initially exhibits a "falling trend" when the time scales are small, but after a certain time scale with the lowest burstiness, a "rising trend" will appear. In contrast, the burstiness trends of most player traces monotonically increase in time scales > 1 second. We will exploit these patterns to develop a bot identification scheme in Section VI-B.

Another aspect we investigate is the *magnitude of traffic burstiness*. Though we cannot judge how "smooth" a traffic process is simply by the absolute value of IDC measures, in our case, we can take the IDC of server packet arrivals as the baseline, and obtain the *relative smoothness of client packet arrivals*. The rationale behind the comparison is that, even if the client traffic is extremely different, game servers still treat all clients *equally*, i.e., the burstiness of server traffic processes, especially in larger time scales, should be similar regardless of the client type. For comparison, we normalize the server packet arrival process so that it has the same mean rate as client packet arrivals, and define the *cross-point* as *the minimum time scale where the burstiness of the client traffic is lower than that of the corresponding server traffic* to determine the relative smoothness of the client traffic. Fig. 13 shows the burstiness comparison for selected traces, where the dashed vertical line denotes the *cross-point*. According to the plots, while both client types have server traffic of similar burstiness trend and magnitude, bot traces have cross-points at lower time scales (< 1 second) than player traces due to their relatively smoother client traffic. This property will be further exploited to identify bots in Section VI-C.

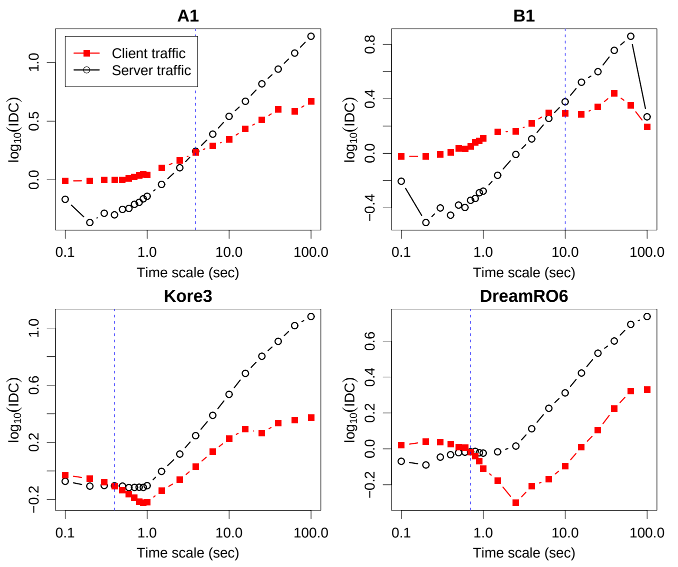


Fig. 13. IDC magnitude comparison: The IDC of client packet arrivals versus the IDC of server packet arrivals.

D. Network Condition Sensitivity

The last aspect we consider is the *subconscious human reactions to network conditions* embedded in traffic traces, which is considerably different from previous approaches.

Our finding is that, *human players adapt to the game pace involuntarily*. While game pace is built into network game clients, it is inevitably affected by network conditions as the latest information about other characters and the environment is conveyed by server packets. In short, the transit delay of server packets influences the pace of game playing and therefore a player's pace. To evaluate how network delay affects a player's pace, samples of round trip times (RTT) as well as the average packet rate within the next second following each RTT sample are computed. The plots describing the relationship between average packet rates and RTTs, where the RTTs are grouped in units of 10 ms, are depicted in Fig. 14.

First we analyze the player traces shown in Figures 14(a) and (b). The trend is clearly *downward*. For slower game pace, which is caused by severely delayed server packets, human players unconsciously slow down their keyboard and

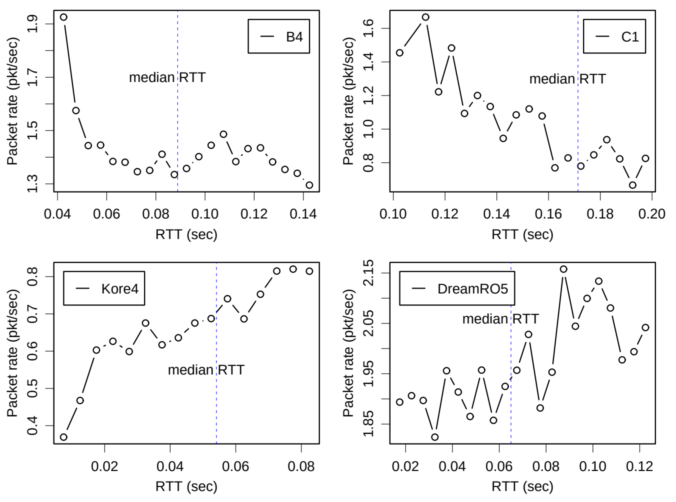


Fig. 14. Average packet rates vs. round trip times plot. The figures exhibit a downward trend for player traces, and a upward trend for bot traces.

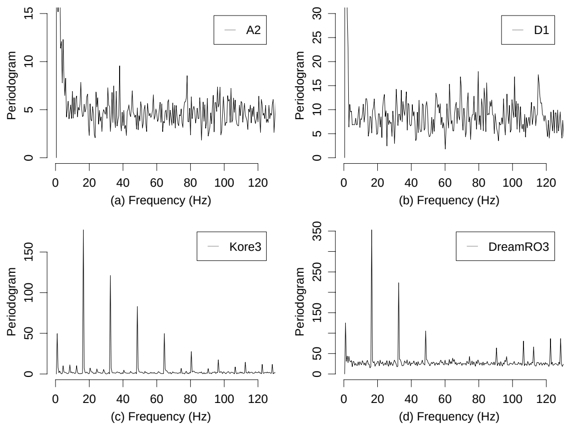


Fig. 15. Periodograms of corresponding histograms (of client response times) in Fig. 10. Note strong frequency components exist in bot traces.

mouse actions so that the corresponding packet rate is lower. Figures 14(c) and (d) indicate that the same phenomenon does not appear in bot traffic: both the *Kore* and *DreamRO* traces show a *upward* trend in the relationship of the packet rate versus RTT. Since bots have their own pacing schemes (certain frequencies by timers), they are not *paced* by server packets like human players. One possible explanation of the bots' upward trend (instead of no trend) is that, for a server packet that arrives late, bots issue more commands which are accumulated before the arrival of that server packet.

The dashed vertical line on the graph denotes the median of the RTT samples. We find that our traces conform to the above observations for RTT samples smaller than the median RTT. One explanation is that higher RTTs are spread more diversely so that sample numbers in each group are not large enough to provide a robust statistic. Another possibility is that higher RTT samples involve packet loss and retransmission, which could affect the TCP congestion window size. This in turn regulates the maximum packet rate. For these reasons, we restrict our analysis to RTT samples lower than the median. The pacing property exhibited by human players will be further exploited in Section VI-D as a means of distinguishing bots from human players.

VI. Proposed Identification Methods

In this section, we propose four decision schemes for the bot identification problem. A decision scheme for a given packet trace will output a dichotomous answer—true or false—to indicate whether or not the trace corresponds to a game session played by a game bot. In the following, we present our methods and the preliminary results. A complete performance evaluation of these methods is provided in the next section.

A. Patterns in Command Timing

Our first method, *command timing*, relies on two properties related to the client response time following receipt of server packets, which was described in Section V-B. In this scheme, we simultaneously apply two tests: 1) whether *multiple peaks* exist in the histogram of client response times less than 10 ms; and 2) whether *regularity* exists in response times less than one second. The scheme returns true if either test is true, and false otherwise.

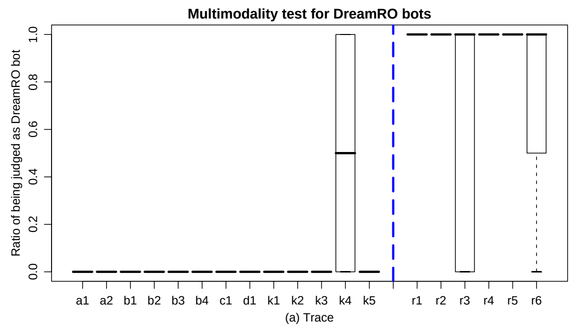
1) Multimodality Test for DreamRO*:* To detect the multi-modality of response times less than 10 ms, we use the Dip test [14, 15], which is designed to test unimodality. On the response time histogram, we first find all local peaks and valleys; then, for each candidate mode, which is determined by two valleys with at least one peak in between, we apply the unimodality test for the candidate, i.e., if the Dip statistic is significant at significance level 0.05. The multimodality test is passed if and only if we can successfully identify two or more modes for response times smaller than 10 ms. Using this test with a segment size of $10,000$, we can correctly distinguish *DreamRO* bots from all other client types in most cases, as shown in Fig. 16(a).

2) Regularity Test: As discussed in Section V-B.2, client response times in bot traces show highly regular patterns in the form of response times clustered in multiples of a certain value (cf. Fig. 10). To check the existence of such regularity, we take the histogram of response times as a spatial series, and then apply methodologies from frequency domain analysis. For a histogram with n bins, we apply a Fourier transform on its ordinates by:

$$I(f) = n\,|d(f)|^2\,,$$

where $d(f)$ is the discrete Fourier transform of that series at frequency f, and $I(f)$ is known as the periodogram. In order to exclude client packets that were not issued due to the arrival of server packets, only response times shorter than one second are considered. In Fig. 15, the corresponding periodograms of histograms in Fig. 10 are depicted. The strong spikes in the periodograms for bot traces show clear evidence of regularity in the response times.

To judge whether periodicity exists in periodograms, which is equivalent to the existence of regularity in the response times, we adopt Fisher's test for periodicity. Fisher [10] proposed a test of the significance of the largest peak in the



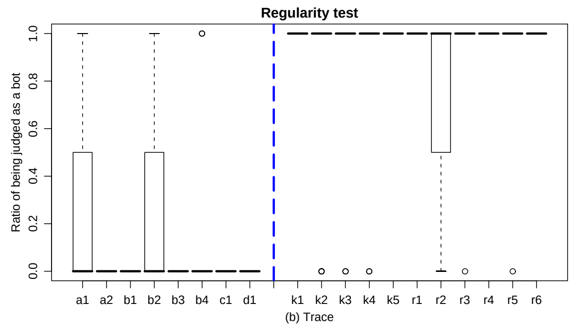


Fig. 16. Bot identification results using the *command timing* method, which comprises the multi-modality test and the regularity test.

periodogram, which is used to determine if the prominent frequency component is "strong enough." The test statistic is the ratio of the largest periodogram ordinate at the Fourier frequencies to the sum of the ordinates. Fuller [11] proposed an equivalent statistic, which is the ratio of the largest ordinate to the average of the ordinates. While the null hypothesis is that the data consists of white noise, Section 6.8 in [4] suggests that, even if the Gaussian assumption is not satisfied, the theory should continue to provide a useful approximation. Suppose $I_1, I_2, \ldots, I_m$ are periodogram ordinates, then by the null hypothesis, $I_1, I_2, \ldots, I_m$ are independent and exponentially distributed with mean σ^2; that is,

$$P(I_j/\sigma^2) = 1 - e^{-x}, x \geq 0, j = 1, 2, \ldots, m.$$

Let $X_m = max(I_1, I_2, \ldots, I_m)$, $Y_m = \sum_{i=1}^{m} I_i$, and Fuller's test statistic be defined as $\xi_m = X_m/(Y_m/m)$. The significance value for Fuller's test is obtained by

$$P(\xi_m \leq \xi) \approx \exp(-me^{-\xi}).$$

Using this method, we test the regularity in response times. A trace is declared as corresponding to a bot if Fuller's statistic is significant at 0.01. The identification outcome, computed with segment size $2,000$, as shown in Fig. 16(b), indicates that the result is correct in most cases, although there are some misjudged cases.

B. *Trend of Traffic Burstiness*

We now turn to the second identification method. In this method, we use the property that bot traffic will exhibit the lowest burstiness at a time scale approximately equal to the iteration time of its main loop. (cf. Section V-C.2)

To check whether the burstiness initially exhibits a falling trend followed by a rising trend, we use the Mann-Kendall

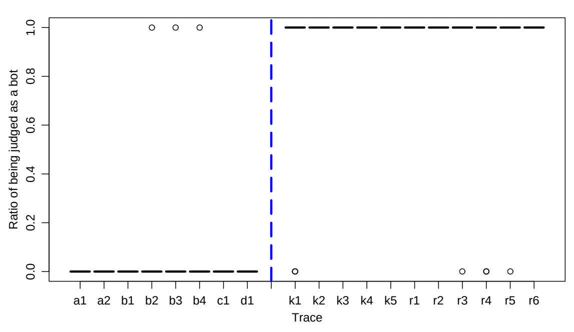


Fig. 17. Bot identification results using the *burstiness trend* method

correlation test [16] to detect the trend of a pair of series. The nonparametric Mann-Kendall test is expected to be robust to outliers, because its statistics are based on the ranks of variables, not directly on their values. Given IDC ordinates, $\{I_t\}$, where $t > 0.1$ is the corresponding time scale, this scheme comprises two sub-tests: 1) Whether (t, I_t) exhibits a significant falling trend followed by a significant rising trend (both at a significance level 0.05), and whether both trends can be detected in time scales smaller than 10 seconds. 2) Whether any time scale $t' > 10$ exists so that $\{(t, I_t), t < t'\}$ exhibits no significant trend, or a significantly negative trend. The scheme outputs true if either test is true; otherwise, it outputs false. The result, as shown in Fig. 17, shows that the decisions of this scheme are correct, except for a few outliers.

C. *Magnitude of Traffic Burstiness*

In this subsection, we propose a bot detection method based on the relative traffic burstiness of client packet arrivals between server packet arrivals. As described in Section V-C.2 and exemplified in Fig. 13, the burstiness of client traffic is comparable to that of normalized server traffic. Also, recall that we define the "cross-point" as a metric of how smooth the client traffic is.

The method based on the magnitude of traffic burstiness is as follows. For a given packet trace, we compute the IDCs for client and server traffic, and search for the cross-point. If the cross-point does not exist, i.e., the client traffic is always more bursty than the server traffic, it is set to the maximum time scale we use, which is 100 seconds. In Fig. 18, we plot the cross-points for all game traces using segment size $10,000$. By observation, we set a threshold at 10 seconds so that a trace is said to correspond to a game bot if the cross-point is smaller than 10 seconds, and to a human player otherwise. The decisions are exact in most cases for player traces; however, some bot traces are classified as human players, especially for *Kore* traces. This suggests that the burstiness of server traffic may not be a good baseline, since it depends on the region where the character resides and the activity of nearby characters. Nevertheless, this method is still worth mentioning as it yields the minimum false positive rate, i.e., the number of times of a player is misjudged as a bot. Details of the correctness of all four schemes will be given in Section VII.

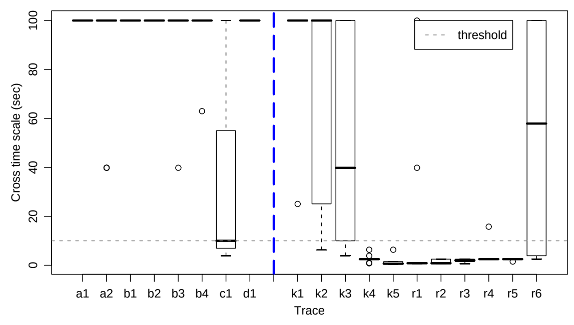


Fig. 18. Bot identification results using the *burstiness magnitude* method

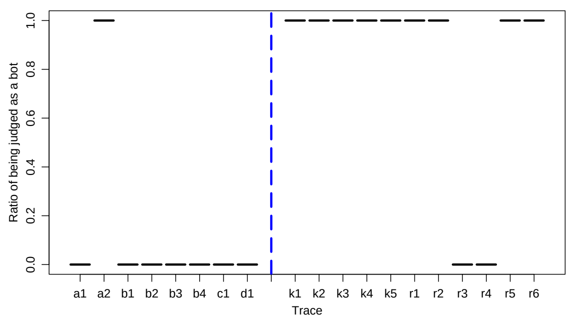


Fig. 19. Bot identification results using the *pacing* method

D. *Reaction to Network Conditions*

In Section V-D, we investigated the relationship between round trip times and the corresponding packet rate; that is, human players subconsciously adapt to network delay, and therefore a *negative* correlation exists between the RTT and the packet rate. In contrast, the RTT and the corresponding packet rate are *not correlated or positively correlated* for bot traces.

The last scheme is based on the above properties. For a given trace, we first take the RTT samples and the corresponding packet rates within the next second of the RTT sampling times. Then we group these samples by RTT with group size 10 ms such that a series, $\{(RTT_i, N_i), i \geq 1\}$, is formed, where RTT_i is the center point of group i and N_i is the average packet rate of samples in group i. We use the Mann-Kendall test to detect the trend of packet rates versus RTT. The detection rule is as follows: first, τ_1 and τ_2, statistics of the Mann-Kendall test, for $\{(RTT_i, N_i), i \geq 1\}$ and $\{(RTT_i, N_i), i \geq 2\}$ respectively, are computed. If $sign(\tau_1) * sign(\tau_2) > 0$, i.e., the trend remains the same regardless of the first RTT group, the identification is completed by $is.bot \leftarrow \tau_1 \geq 0$. Otherwise, we compute τ_3 for $\{(RTT_i, N_i), i \geq 3\}$, and conclude the test by the sign of τ_3, i.e., $is.bot \leftarrow \tau_3 \geq 0$.

The classification result of the above procedures is plotted in Fig. 19. Although most player traces are correctly judged, the scheme seems to have problems correctly identifying *DreamRO* bots. We find that *DreamRO* traces sometimes exhibit a negative correlation between the RTT and the packet rate, which we characterize as human behavior. The reasons for this behavior are still under investigation. However, we consider such methods based on human behavior rather interesting and potentially useful. Further investigation into the relationship between game traffic patterns and user behavior will be part of our future work.

VII. Performance Evaluation and Discussion

In this section, we evaluate proposed bot identification schemes. For each scheme, three metrics are evaluated: the *correct rate*, the ratio the client type of a trace is correctly determined; the *false positive rate*, the ratio a player is judged as a bot; and the *false negative rate*, the ratio a bot is judged as a human player. In addition, we are concerned about the sensitivity of the input size, i.e., *how long a traffic stream must be to enable correct identification*. Thus, the performance metrics are computed on a segment basis by dividing the traces into segments of a given size. The results are presented in Fig. 20.

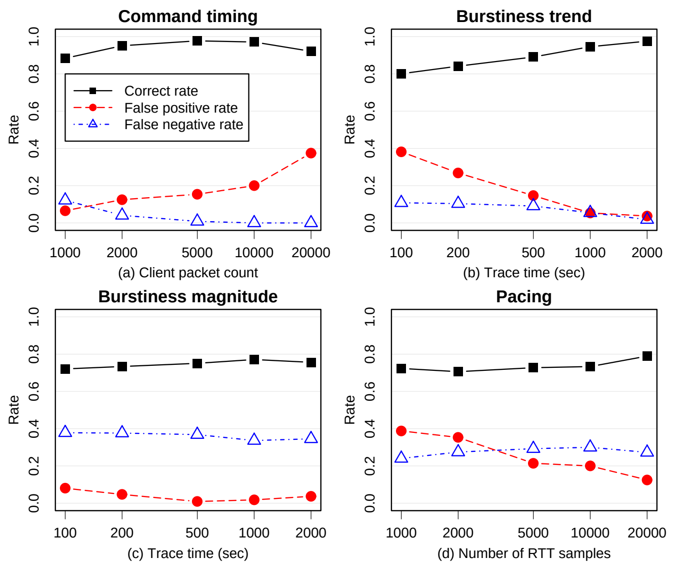


Fig. 20. Evaluation results for the proposed decision schemes with different input size.

The evaluation results reveal that the first two methods, *command timing* and *burstiness trend*, work rather well. Specifically, both methods can achieve correct decision rates higher than 95% and false negative rates lower than 5%, given an input size larger than $10,000$ packets. From the viewpoint of business operations, a good bot detection method should minimize the false positive rate while yielding a high correct decision rate. This is because misjudging a human player as a bot would annoy legitimate players, while misjudging a bot (as a human player) would be relatively acceptable. By this rule, the *burstiness magnitude* method is good, since it always achieves low false positive rates ($< 5\%$), and yields a moderate correct decision rate ($\approx 75\%$). Although the *pacing* method does not perform well, it is still proposed because of its unique relation to human behavior.

In practice, we can detect game bots using an integrated approach, i.e., by applying multiple schemes simultaneously and combining their results according to preference. For exam-

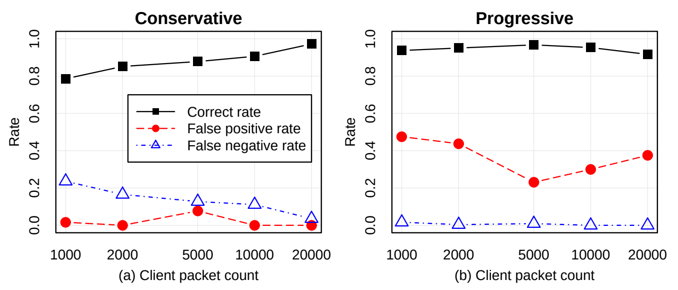


Fig. 21. Evaluation results for the integrated schemes with different input size.

ple, if a conservative judgement is preferred, the final decision could be integrated by a logical "AND"; that is, a traffic stream would only be declared as corresponding to a game bot if all schemes agree with that declaration. By this reasoning, we propose two integrated schemes, a conservative approach and a progressive approach. Combining the *command timing* and *burstiness trend* methods, the conservative approach is achieved by a logical "AND" operation and the progressive approach by an "OR" operation. The performance of these integrated schemes, as presented in Fig. 21, is rather good in terms of reducing the occurrence of certain kinds of false alarm. The conservative approach reduces the false positive rate to zero and achieves a 90% correct decision rate, given an input size of $10,000$ packets. Meanwhile, the progressive approach produces a false negative rate of less than 1% and achieves a 95% correct decision rate, given an input size of $2,000$ packets.

VIII. Conclusion

In this paper, we have addressed the game bot problem in MMORPGs, and used traffic analysis to identify game bots. Taking *Ragnarok Online* as the subject of our analysis, we traced and analyzed packet traces for mainstream game bots and human players with different network settings. We have shown that the traffic corresponding to bots and human players is distinguishable in various respects, such as the regularity and patterns in client response times, the trend and magnitude of traffic burstiness in different time scales, and the sensitivity to network conditions. We consider that the tendency of human players to unconsciously adapt to the game pace is rather important and worth further investigation.

Based on the findings of traffic analysis, we have proposed four decision schemes and two integrated schemes for the bot identification problem. For our collected traces, the conservative approach of our proposed integrated schemes reduces the false positive rate to zero and produces a 90% correct decision rate, given an input size of $10,000$ packets. The progressive approach, on the other hand, yields a false negative rate of less than 1% and achieves a 95% correct decision rate, given an input size of $2,000$ packets. The former completely avoids making false accusations against bona fide players, while the latter tracks game bots down more aggressively.

Acknowledgments

This work would not have been possible without valuable traffic traces. The authors are indebted to Kiya Hsu, Gino Liu, and Chia-Chun Hung, who continuously played *Ragnarok Online* for more than five hours during the trace collection.

References

[1] R. A. Bangun and E. Dutkiewicz, "Modelling multi-player games traffic," in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC)*, Las Vegas, NV, Mar. 2000, pp. 228–233.

[2] R. A. Bangun, E. Dutkiewicz, and G. J. Anido, "An analysis of multi-player network games traffic," in *Proceedings of the 1999 International Workshop on Multimedia Signal Processing*, Copenhagen, Denmark, Sept. 1999, pp. 3–8.

[3] N. E. Baughman and B. N. Levine, "Cheat-proof playout for centralized and distributed online games," in *Proceedings of the Twentieth IEEE Computer and Communication Society INFOCOM Conference*, Anchorage, AK, Apr. 2001.

[4] P. Bloomfield, *Fourier Analysis of Time Series: An Introduction*. New York: John Wiley & Sons, 2000.

[5] K.-T. Chen, P. Huang, C.-Y. Huang, and C.-L. Lei, "Game traffic analysis: An MMORPG perspective," in *NOSSDAV'05: Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM Press, 2005, pp. 19–24.

[6] E. Cronin, B. Filstrup, and S. Jamin, "Cheat-proofing dead reckoned multiplayer games," in *Proc. of 2^{nd} International Conference on Application and Development of Computer Games*, Jan 2003.

[7] M. DeLap, B. Knutsson, H. Lu, O. Sokolsky, U. Sammapun, I. Lee, and C. Tsarouchis, "Is runtime verification applicable to cheat detection?" in *SIGCOMM 2004 Workshops: Proceedings of ACM SIGCOMM 2004 workshops on NetGames '04*. ACM Press, 2004, pp. 134–138.

[8] J. Färber, "Network game traffic modelling," in *NetGames '02: Proceedings of the 1st Workshop on Network and System Support for Games*. ACM Press, 2002, pp. 53–57.

[9] W. C. Feng, F. Chang, W. C. Feng, and J. Walpole, "A traffic characterization of popular on-line games," *IEEE/ACM Transactions on Networking*, June 2005.

[10] R. A. Fisher, "Tests of significance in harmonic analysis," *J. Roy. Stat. Soc., Ser. A*, vol. 125, pp. 54–49, 1929.

[11] W. A. Fuller, *Introduction to statistical time series*. John Wiley & Sons, 1996.

[12] "Ragnarok Online," Gravity Corp. [Online]. Available: http://iro.ragnarokonline.com/

[13] R. Gusella, "Characterizing the variability of arrival processes with indexes of dispersion," *IEEE J. Select. Areas Commun.*, vol. 9, no. 2, Feb. 1991.

[14] J. Hartigan and P. Hartigan, "The dip test of unimodality," *Ann. Stat.*, vol. 13, pp. 70–84, 1985.

[15] P. M. Hartigan, "Computation of the dip statistic to test for unimodality," *Appl. Stat.*, vol. 34, no. 3, pp. 320–325, Sept. 1985.

[16] M. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, pp. 81–93, 1938.

[17] B. S. Woodcock, "An analysis of MMOG subscription growth - version 18.0." [Online]. Available: http://www.mmogchart.com/