Online Gaming Traffic Generator for Reproducing Gamer Behavior

Kwangsik Shin, Jinhyuk Kim, Kangmin Sohn, Changjoon Park, Sangbang Choi

Online Gaming Traffic Generator for Reproducing Gamer Behavior

Kwangsik Shin[1], Jinhyuk Kim[2], Kangmin Sohn[1], Changjoon Park[1], Sangbang Choi[2],

[1] Smart Game Research Team, Content Research Division
Electronics and Telecommunications Research Institute,
138 Gajeongno, Yuseong-Daejeon, 305-700, Korea
{kwangsik, sogarian, chjpark}@etri.re.kr

[2] Electronic Engineering Department
Inha University
253 Yonghyun-dong, Nam-gu Incheon 402-751, Korea
graybaily@inha.edu, sangbang@inha.ac.kr

Abstract. In this paper, we proposed an online gaming traffic generator reflecting user behavior patterns. We analyzed the packet size and inter departure time distributions of a popular FPS game (Left4Dead) and MMORPG (World of Warcraft) for regenerating gaming traffic. The proposed traffic generator generates an inter departure time and gaming packet based on analytical model of the gamer behaviors, then transmits the packet according to the inter departure time. Packet generation results show that generated packets of World of Warcraft is much different with analytical model, unlike Left4Dead. It is caused by Nagle algorithm and Delayed Acknowledgments of TCP. Thus, we disabled the Nagle algorithm in the proposed traffic generator. The generation results show that the revised proposed traffic generator guarantees goodness of fit in the generated traffic distribution.

Keywords: Traffic generator, online gaming traffic, traffic analysis.

1 Introduction

With the recent global explosion of on-line multiplayer gaming, it is becoming more important to understand its network behavior and usage. Those are expected to be major contributors to World Wide Web traffic. On-line gaming is an increasingly popular form of entertainment on the Internet, with the on-line market predicted to be worth over $19.7 billion dollars in 2011 [1]. As an example of a popular, money-making game, World of Warcraft (WOW) has over 11,500,000 subscribers each paying a monthly fee in December 2008 [2]. Nowadays, in proposition to their success, the stability and performance of game servers become major issues in online games.

With the advances in computing and communication technologies, the issue of server testing using virtual user has been attracting interests from the researchers.

These researches help reduce the costs and time required to test and deploy new applications and systems. The server testing technology using virtual clients has been studied very actively on file server, web server and etc [3-4]. The industry standard load testing solutions such as LoadRunner[5], QALoad[6] and e-Load[7], can emulate hundreds or thousands of concurrent users and prevent costly performance problems. Recently, online game server testing introduces the new research issues into the server testing technology [8-9]. These solutions enable to measure end-to-end performance, diagnose application, system bottlenecks and tune for better performance. One of the most important steps for these systems is the creation of tools for scalable, tunable, representative network traffic generation. Such tools are especially critical in emulation and simulation environments where representative background traffic is needed.

There are a number of works on the general topics of traffic characterization and modeling. Current studies for traffic generation tend to focus on either simple packet streams or recreation of a single application specific behavior [10-14]. However, interactive game traffic is much different with web or conventional internet traffic prevailing on the Internet today. It tends to employ small, highly periodic packets. Packets are small since the application requires extremely low latencies which make message aggregation and message retransmissions impractical. While conventional traffic generation tools are quite useful for evaluating the behavior and performance of host systems for traditional internet service, they are often inappropriate for use in online game tests. Therefore, in this paper, we propose an online gaming traffic generator reflecting user behavior patterns.

The rest of this paper is organized as follows. Section 2 summarizes related works for traffic generation. Section 3 provides a brief account of the online gaming traffic analysis results. Section 4 describes architecture of the proposed traffic generator and operation flow. Section 5 presents the result of traffic generation. And finally, section 6 draws some conclusions and future works.

2 Related Work

There are a number of works on the general topics of Internet traffic characterization and modeling. Current studies for traffic generation tend to focus on either simple packet streams or recreation of a single application specific behavior.

Packet streaming methods such as those used in tools like iperf [10] consist of sequences of packets separated by a constant interval. These methods form the basis for standard router performance tests such as those recommended in RFC 2544 and RFC 2889. Another example is an infinite FTP source, a commonly used traffic generator in simulations. While these approaches provide some insight into network system capabilities, they lack nearly all of the richness and diversity of packet streams observed in the live Internet [11]. Several successful application-specific workload generators have been developed as following. These generate application-level request sequences based on network traffic that has the same statistical properties as live traffic from the modeled application.

SURGE (Scalable URL Reference Generator) [12] is the tool for generating HTTP workload. One role for HTTP workload generation is as a means for understanding how servers and networks respond to variation in load. Empirical studies of operating Web servers have shown that they experience highly variable demands, which is exhibited as variability in CPU loads and number of open connections SURGE applies a number of observations like server file size distribution, request size distribution, relative file popularity, embedded file references, and idle periods of individual users of Web server usage to create a realistic Web workload generation tool which mimics a set of real users accessing a server.

Tmix [13] is a traffic generation system for the widely used ns-2 simulator. As the web have evolved, there remains no existing model of HTTP workloads that accounts for routine uses of the web for applications such as peer-to-peer file sharing and remote email access. In order to perform realistic network simulations, one needs a traffic generator that is capable of generating realistic synthetic traffic in a closed-loop fashion that "looks like" traffic found on an actual network. Tmix takes as input a packet header trace taken from a network link of interest. The trace is "reverse compiled" into a source-level characterization of each TCP connection present in the trace. The characterization, called a connection vector, is then used as input to an ns module called Tmix that emulates the socket-level behavior of the source application that created the corresponding connection in the trace.

D-ITG(Distributed Internet Traffic Generator) [14] is a platform capable to produce traffic that emulate sources of various protocols: TCP, UDP, ICMP, DNS, Telnet and VoIP (G.711, G.723, G.729, Voice Activity Detection, Compressed RTP). D-ITG produces traffic accurately adheres to patterns defined by the inter departure time between packets (IDT) and the packet size (PS) stochastic processes. Also, it can perform both one-way-delay (OWD) and round-trip-time (RTT) measurement, packet loss evaluation, jitter and throughput measurement. D-ITG improves log-server using distributed architecture especially for performance enhancement achieved by the sender (in terms of generated data rate) and the receiver (in terms of received data rate). Log-server is used by senders and receivers to maintain the information needed to compute statistics about the experiment made.

These typically focus on generating sequences of conventional Internet applications that result in network traffic that has the same statistical properties as live traffic from the modeled application. However, interactive game traffic will be much different with web or conventional internet traffic prevailing on the Internet today. It tends to employ small, highly periodic packets. Packets are highly periodic as a result of the game's dynamic requirement of frequent, predictable state updates amongst clients and servers. While these tools are quite useful for evaluating the behavior and performance of host systems for traditional internet service, they are often inappropriate for use in online game tests. Therefore, in this paper, we propose a method suitable for modeling and generating MMOG's traffic.

3 Traffic Analysis

In this section, we analyzed the packet size and inter departure time (IDT) for various games of different genres, since online gaming traffic has different characteristics according to the type of game. Among them, we chose two recent popular online games, Left4Dead (L4D) [15] and WOW. The L4D is a cooperative first-person shooter (FPS) game by Valve Corporation and the WOW is a massively multiplayer online role playing game (MMORPG) by Blizzard Entertainment. FPS games provide large-scale gaming, and sometimes team-based combat in a real-time virtual environment. This is made through games utilizing the character's point of view, transmitting data from the client to the server, and then processing immediately the data received at that time. MMORPGs are a genre of computer role-playing games in which a very large number of players interact with one another within a virtual world. They focus on an accurate execution of client inputs, which has an impact on the transport protocol used. In the WOW, TCP serves as the transport protocol. TCP is connection oriented and offers reliability. This attribute is well suited for MMORPGs, preventing error propagation during long sessions.

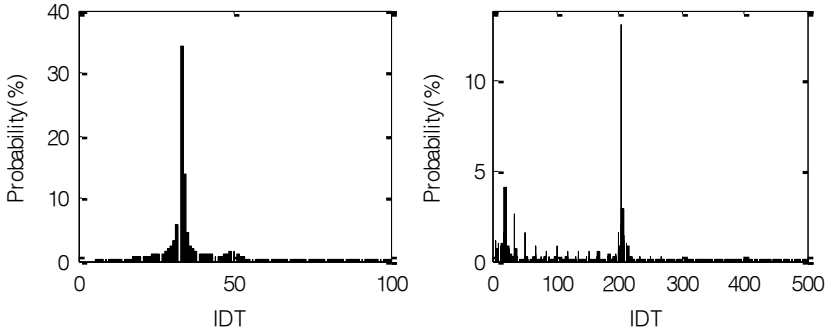

Fig. 1. IDT histograms for (ⓐ left) Left4Dead and (ⓑ right) WOW gaming packets.

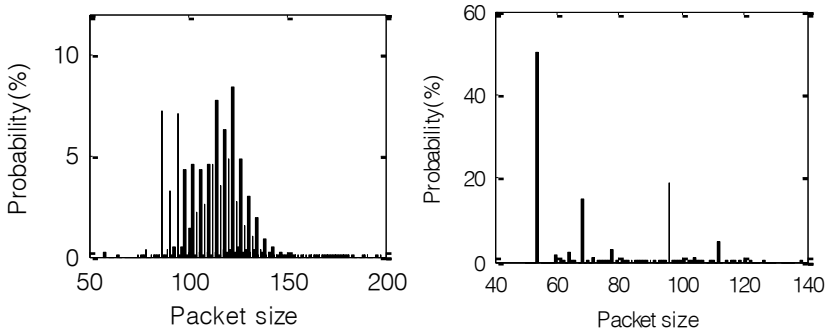

Fig. 2. Packet size histograms for (ⓐ left) Left4Dead and (ⓑ right) WOW gaming packets.

Fig. 1 and 2 show distributions of IDT and packet size for the L4D and WOW, respectively. From the figures, we know that there are clear distinctions between two

genres. In the fig. 1, mostly IDTs of the L4D are less than 50ms, while there are lots of packets with long IDT for the WOW. As a FPS game, the L4D imposes the hardest real-time requirements on a network. Since it is very sensitive to interactivity, this type of game requires low-latency point-to-point communication as well as directed broadcast channels to facilitate its real-time game logic. Therefore, packets are sent via UDP since clients should send packets at an interval that is much shorter than the time it would take to retransmit lost packets. In the other hand, fig. 2 shows that packet sizes of the WOW are less than ones of the L4D. Even though the MMORPG uses a TCP connection with long latency caused by error recovery mechanism, it reduces the round trip time (RTT) by using small packets.

While there are some differences according to game genre, online gaming traffic generally tends to employ small, highly-periodic packets in common. An in-depth knowledge of their traffic behavior will certainly assist network game developers and publishers to provide services and design their game servers better. In this paper, we propose an online gaming traffic generator in order to make available traffic characterizations of popular network games.

4 Architecture

In this section, we introduce the architecture of the proposed online gaming traffic generator shown in fig. 3. In the figure, *Analytical Model* is a result of analysis for each online game. Since online gaming traffic is typically messy data that is very difficult to analyze, we already studied an analytical model designing method for gaming traffic in a preliminary work [16].

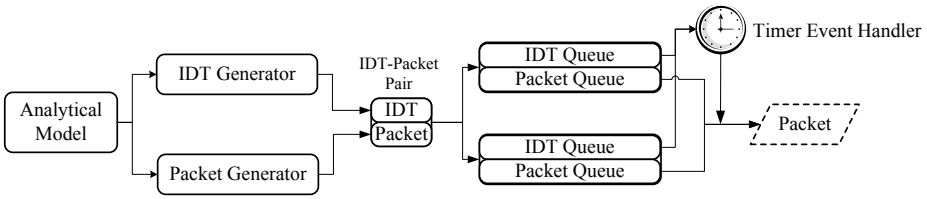

Fig. 3. The architecture of the proposed traffic generator.

Packet Generator spawns a specified gaming packet classified by packet size and stores it on *Packet Queue*. The *Packet Queue* is implemented by double buffer to hold a block of data, so that *Timer Event Handler* will see a complete version of the data, rather than a partially-updated version of the data being created by *Packet Generator*. *IDT Generator* produces an interval, *IDT*, for triggering a *Timer Event Handler* that pulls out a generated packet. A *Timer Event Handler* is activated when an *IDT* is expired, and it takes out an *IDT* from the *IDT Queue* whenever it terminates execution. Since the *IDT Generator* is a producer of the *IDT Queue* and the *Timer Event Handler* is a consumer of it, we also implemented the *IDT Queue* as a double buffer. Because transmission of each generated packet is controlled by corresponding *IDT*, *Packet Queue* is synchronized with *IDT Queue* by *IDT-Packet Pair*.

Fig. 4 shows a flow chart of the proposed traffic generator. There are two threads; *Generator* and *Transmitter*. The former is a producer thread and the other is a consumer thread of *IDT & Packet Queue*. The producer initially finds a *Write_Q* and then verifies whether it is empty or not. If not, this process is repeated until detecting empty queue. In the other case, it generates an *IDT* and *Packet* based on *Analytical Model* of the gamer behaviors, and then groups them together prior to enqueue. After buffering *IDT-Packet Pair*, generating, pairing, and enqueueing processes are reiterated until corresponding queue is full. The consumer firstly gets a *Read_Q* and then checks state of queue. If the queue is not empty, it dequeues an *IDT-Packet Pair* and activates *Timer Event Handler*. The *Handler* sets timer as the *IDT* of dequeued *IDT-Packet Pair* and transmits the *Packet* of it after timer expiration. These procedures are continuously performed again until the queue is empty. When detecting unfilled queue, the *transmitter* defines corresponding queue as *Write_Q* and the other queue as *Read_Q*, respectively.

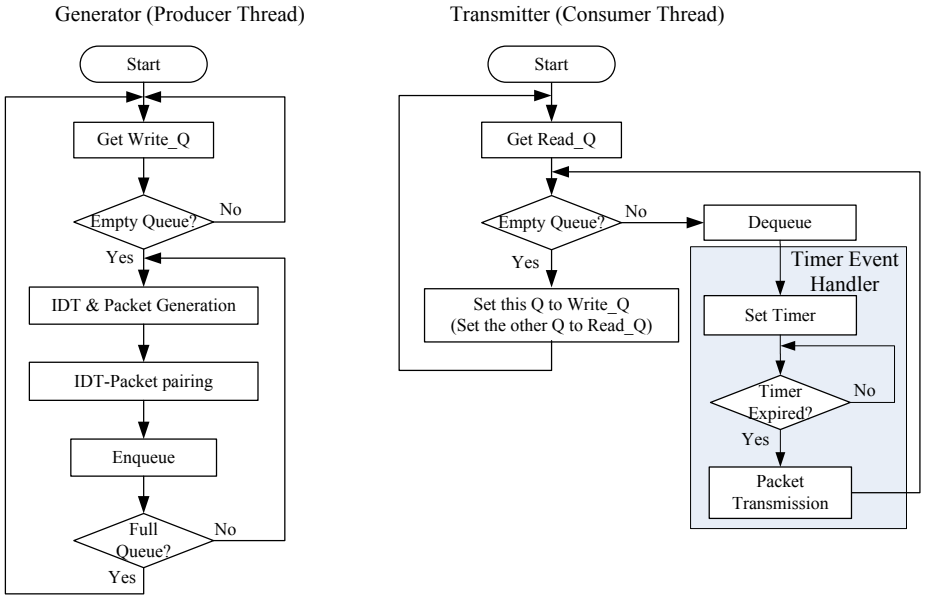

Fig. 4. Flow charts for (ⓐ left) Generator and (ⓑ right) Transmitter threads of the proposed scheme.

5 Evaluation

In this section, we compared generation results with analytical model based on previous analyzed gaming traffic. We captured produced packets of L4D and WOW at the receiver side and analyzed them into distributions of IDT and packet size as performed in the previous section. We show an intuitive comparison between the expected and generated histograms in fig. 5 and 6. In the figures, all expected

histograms were drawn by analytical models for L4D or WOW. Fig. 5 shows that distribution graphs for both IDT and packet sizes of L4D are similar to ones of analytical model.

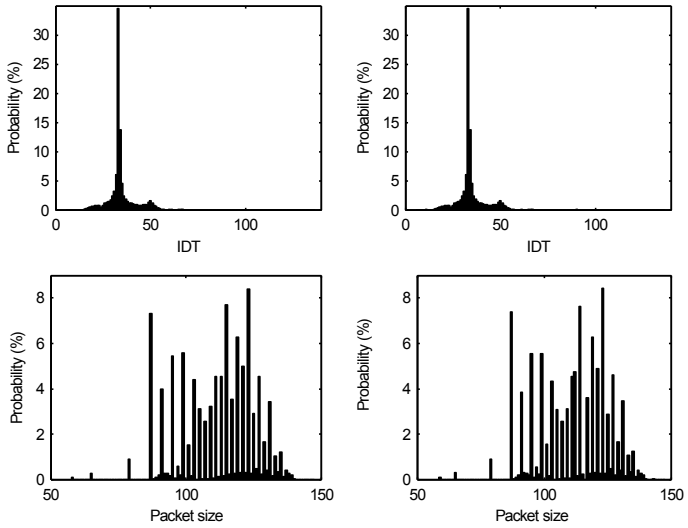

Fig. 5. Histograms for L4D; (ⓐ left-top) expected distribution for IDT, (ⓑ right-top) distribution of generated packets for IDT, (ⓒ left-bottom) expected distribution for packet size, and (ⓓ right-bottom) distribution of generated packets for packet size.

In spite of traffic generated by analytical model for IDT and packet size of WOW in fig. 6, however, there is a large discrepancy between generated packets and analytical model. It is caused by Nagle algorithm and Delayed Acknowledgments of TCP. Nagle algorithm is a means of improving the efficiency of TCP networks by reducing the number of packets that need to be sent over the network. It works by combining a number of small outgoing messages, and sending them all at once. In the other hand, Delayed Acknowledgement is used to reduce the number of packets sent on transmission media. If the socket layer immediately acked every packet, this would result in a lot of wasted acks, as usually the application sends a response shortly after receiving something, so TCP would be issuing two acknowledgments, one from the socket layer, one from the application layer. So the socket layer waits one or two hundred milliseconds, and if there is still nothing to send, sends an ack. Due to these techniques, even though traffic generator schedules packet transmission according to analytical model, TCP stack prevents some packets from leaving for a short time. The fig. 6 shows IDTs of generated packets are delayed until about 200ms and some small packets are united into one. Comparing with analytical model, ratio of small packets is reduced and some bigger sized packets are emerged. In this example, protocol header of WOW consists of TCP, IP, and Ethernet headers and its length is 54bytes.

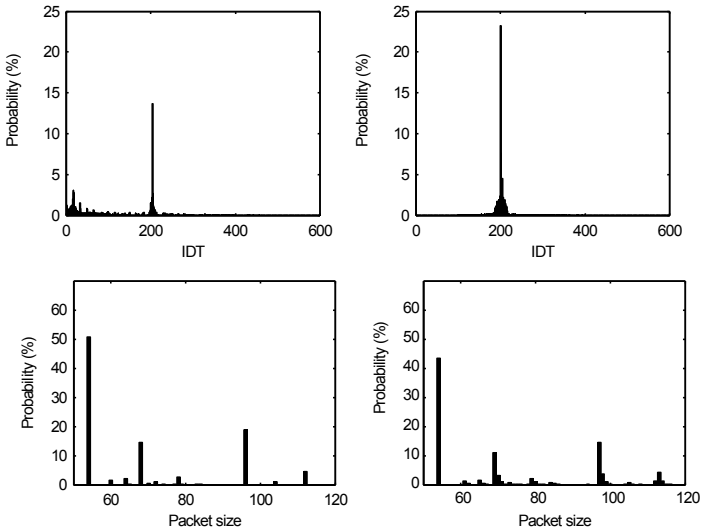

Fig. 6. Histograms for WOW; (ⓐ left-top) expected distribution for IDT, (ⓑ right-top) distribution of generated packets for IDT, (ⓒ left-bottom) expected distribution for packet size, (ⓓ right-bottom) distribution of generated packets for packet size,.

From the fig. 6, we know applications expecting real time responses can react poorly with Nagle algorithm. Thus, we disabled the Nagle algorithm by setting the `TCP_NODELAY` socket option. Fig. 7 shows histogram for distributions of packets generated without Nagle algorithm. After removal of Nagle algorithm, distributions of generated packet are close to ones of analytical model.

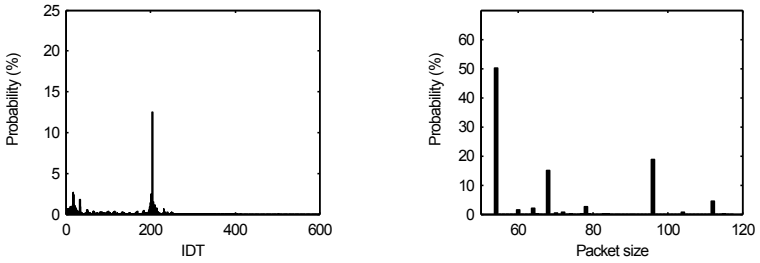

Fig. 7. Histograms for distribution of packets generated without Nagle algorithm; (ⓐ left) IDT of WOW and (ⓑ right) packet size of WOW.

Finally, we displayed Q-Q plots of each model for a more detailed comparison. It is commonly useful in order to examine whether a generated dataset fits well an expected distribution. On this plot, the corresponding quartiles of both expected and generated distributions are graphed against each other, so that the deviations may be easily identified. If the two distributions being compared are similar, the points in the

Q-Q plot will approximately lie on the line y = x. Fig. 8 shows Q-Q plots for the IDT and packet size of generated traffic.

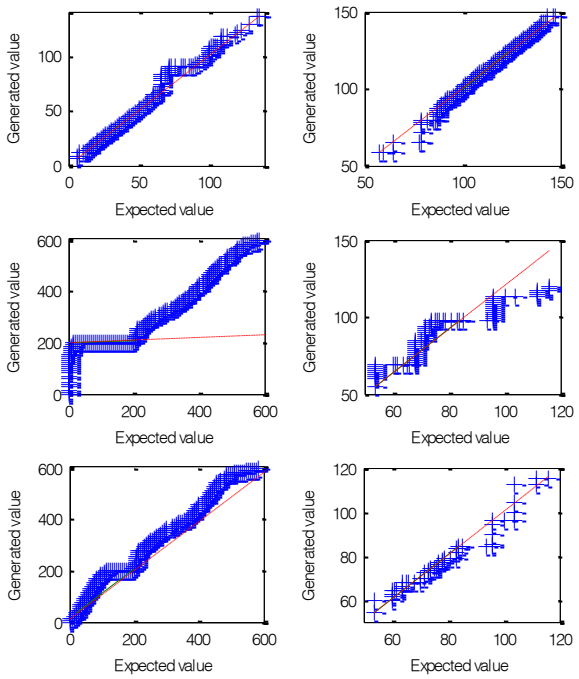

Fig. 8. Q-Q plots for distributions of generated packets; (ⓐ left - top) IDT of L4D, (ⓑ right - top) packet size of L4D, (ⓒ left – mid) IDT of WOW, (ⓓ right – mid) packet size of WOW, (ⓔ left – bottom) IDT of WOW without Nagle algorithm, and (ⓕ right – bottom) packet size of WOW without Nagle algorithm.

It shows in detail which parts fit together appropriately. The Q-Q plot shows a pronounced difference among distributions of packets generated for L4D and WOW before and after disabling Nagle algorithm. From the fig. 8 (top) and (bottom), we know that the revised proposed traffic generator guarantees goodness of fit in the measured traffic distribution. Even though we inactivated Nagle algorithm, IDT of generated WOW is relatively worse than ones of generated L4D. Since the TCP provides various traffic control mechanisms, for example, a congestion control, flow control service, Delay Acknowledgement, and etc. as well as Nagle algorithm, arranged IDT of the traffic generator is distorted by TCP.

6 Conclusion and Future Work

In this paper, we proposed an online gaming traffic generator reflecting user behavior patterns. We analyzed the packet size and IDT for various games of different genres and introduced traffic characterizations of popular network games. The

proposed traffic generator generates an IDT and packet based on analytical model of the gamer behaviors, and then transmits the packet according to the IDT.

In order to evaluate the proposed traffic generator, we compared generation results with analytical model based on previous analyzed gaming traffic. The results showed that generated packets of WOW is much different with analytical model while distribution graphs for both IDT and packet sizes of L4D are similar to ones of analytical model. It is caused by Nagle algorithm and Delayed Acknowledgments of TCP. Thus, we disabled the Nagle algorithm by setting the `TCP_NODELAY` socket option. The results show that the revised traffic generator guarantees goodness of fit in the generated traffic distribution. However, arranged IDT of the traffic generator is distorted by TCP stack, since the TCP provides various traffic control mechanisms, for example, a congestion control, flow control service, Delay Acknowledgement, and etc. as well as Nagle algorithm.

The problem of distortion by TCP stack remains an open research issue in gaming traffic generator. In order to improve accuracy for TCP based online game, we need to design delay model for various transmission control mechanisms of TCP. As a different approach, for bypassing TCP stack, we plan to implement traffic generator with a raw socket allowing direct sending and receiving of network packets by applications. In addition, we plan to measure workload of online game server according to massive gaming traffic generation.

Acknowledgement

This work was supported by the Industrial Strategic Technology Development Program (KI002095, Online Game Quality Assurance Technology Development using Scenario Control of Massive Virtual User) funded by the Ministry of Knowledge Economy (MKE, Korea)

Reference

[1] Korea Creative Contents Agency.:2009 White Paper on Korean Games. pp. 538-543, Sep. (2009)
[2] Blizzard Entertainment, http://us.blizzard.com/
[3] Arneson, D., Beth, S., Tavakley R., Ruwart, T.:A Test Bed for a High-Performance File Server. Proceedings, Twelfth IEEE Symposium on Mass Storage Systems, pp. 26–29, Apr., (1993)
[4] Elbaum, S., Karre, S., Rothermel, G.: Improving web application testing with user session data. Proceedings. 25th International Conference on Software Engineering, pp. 49–59, May, (2003)
[5] LoadRunner, http://www.hp.com
[6] QALoad, http://www.compuware.com/
[7] Empirix, OneSight, http://www.empirix.com/
[8] Jung, Y., Lim, B., Sim, K., Lee, H., Park, I., Chung, J., Lee, J.: VENUS: The Online Game Simulator Using Massively Virtual Clients. System Modeling and Simulation: Theory and Applications, LNCS 3398, pp. 589-596. Springer, Heidelberg (2005)

[9] Cho, C., Sohn, K., Park, C., Kang, J.: Online Game Testing Using Scenario-based Control of Massive Virtual Users. 12th International Conference on Advanced Communication Technology, vol. 2, pp.1676-1680, Feb. (2010)
[10] The iperf TCP/UDP Bandwidth Measurement Tool, http://dast.nlanr.net/Projects/Iperf
[11] Sommers, J., Kim, H., Barford, P.: Harpoon: a flow-level traffic generator for router and network tests. In Proceedings of the Joint international Conference on Measurement and Modeling of Computer Systems, pp. 392-392, Jun. (2004)
[12] Barford, P., Crovella, M.: Generating representative Web workloads for network and server performance evaluation. In Proceedings of the 1998 ACM SIGMETRICS Joint international Conference on Measurement and Modeling of Computer Systems, pp. 151-160, Jun. (1998)
[13] Weigle, M., Adurthi, P., Hernández-Campos, F., Jeffay, K., Smith, F.: Tmix: a tool for generating realistic TCP application workloads in ns-2. SIGCOMM Comput. Commun. Rev. 36, pp. 65-76, Jul. (20060
[14] Analysis and Experimentation of an Open Distributed Platform for Synthetic Traffic Generation. In Proceedings of the 10th IEEE international Workshop on Future Trends of Distributed Computing Systems, pp. 277-283, May (2004)
[15] Left4Dead, http://www.l4d.com
[16] Shin, K., Kim, J., Sohn, K., Park, C., Choi, S.: Transformation Approach to Model Online Gaming Traffic. unpublished.
[17] MSDN, http://msdn.microsoft.com