Network Traces of Virtual Worlds: Measurements and Applications

Yichuan Wang*
Department of Computer Science
University of California
Davis, CA
yicwang@ucdavis.edu

Cheng-Hsin Hsu
Deutsche Telekom Inc.
R&D Laboratories USA
Los Altos, CA
cheng-hsin.hsu@telekom.com

Jatinder Pal Singh
Deutsche Telekom Inc.
R&D Laboratories USA
Los Altos, CA
j.singh@telekom.com

Xin Liu
Department of Computer Science
University of California
Davis, CA
xinliu@ucdavis.edu

ABSTRACT

Although network traces of virtual worlds are valuable to ISPs (Internet service providers), virtual world software developers, and research communities, they do not exist in the public domain. In this work, we implement a complete testbed to efficiently collect and analyze network traces from a popular virtual world: Second Life. We use the testbed to gather traces from 100 regions with diverse characteristics. The network traces represent more than 60 hours of virtual world traffic and the trace files are created in a well-structured and concise format. Our preliminary analysis on the collected traces is consistent with previous work in the literature. It also reveals some new insights: for example, local avatar/object density imposes clear implications on traffic patterns. The developed testbed and released trace files can be leveraged by research communities for various studies on virtual worlds. For example, accurate traffic models can be derived from our trace files, which in turn can guide developers for better virtual world designs.

Categories and Subject Descriptors

H.5.1 [**Information Systems Applications**]: Multimedia Information Systems

General Terms

Measurement

1. INTRODUCTION

Virtual worlds, such as *Second Life* [18], *Habbo Hotel* [4], and *Playstation Home* [15], are computer-simulated environments that allow many users to interact with each other via graphical avatars. Virtual worlds enable a plethora of interesting applications spanning entertainments, 3D shopping malls, immersive distance learning, virtual workspace, and online art galleries, and are thus becoming increasingly popular. For example, Linden Research reports that there were more than a million users logged in to Second Life in July 2010 [16].

While most online games distribute game data to users using optical media such as DVDs, virtual worlds consist of user-generated and dynamic in-world objects that can only be downloaded on-demand. The bandwidth requirements of virtual worlds, therefore, are significantly higher than online games [6]. The high-volume real-time traffic generated by virtual worlds imposes great challenges to the best-effort Internet. Hence, network traces of virtual worlds are very valuable to ISPs (Internet service providers) for better network planning [1] and software developers for better virtual world design [12].

To the best of our knowledge, there exists no publicly available network traces of virtual worlds. Research groups resort to building their own utilities to capture traces, which is a tedious and time consuming task. In this work, we conduct extensive experiments in a popular virtual world to collect network traces in different environments (e.g., diverse avatar and object density) and with various avatar actions (e.g., walk and run). Most importantly, we share the resulting traces with research communities to stimulate the future research on virtual worlds [17].

Our main contributions are summarized as follows:

- A fully automated testbed is developed for efficient and large-scale virtual world traffic analysis.
- We collect Second Life traces from 100 diversified regions. The total length of our traces is over 60 hours, and the resulting trace files are over 3 GB in size.
- We validate previous work in this area, and provide new insights. For example, local avatar/object density has strong correlation with traffic patterns, which was never reported in the literature.

The rest of this paper is organized as follows. We review related work in Sec. 2. We give an overview of virtual worlds in Sec. 3. Sec. 4 describes our measurement methodology. We present the network traces in Sec. 5. Sec. 6 briefly discusses some potential applications and Sec. 7 concludes the paper.

*This work was done when Y. Wang was an intern at Deutsche Telekom R&D Labs USA.

MMSys'11, February 23–25, 2011, San Jose, California, USA.

2. RELATED WORK

Two types of traces have been collected from virtual worlds: (i) traces of the locations and movements of avatars/objects and (ii) traces of timestamped network packets. Traces of avatars/objects are usually gathered by crawlers that traverse through many regions of a virtual world. For example, Varvello et al. [20] build a crawler that collects information about avatars, objects and server status from $\sim$ 13,000 regions. La and Michiardi [7] and Liang et al. [9] also build crawlers to derive mobility models for avatars. Some avatar/object traces are made publicly available [11], which are complementary to the network traces we collected.

Network traces from virtual worlds have also been collected in various studies [1–3, 5, 8, 12]. Liang et al. [8] implement a Second Life proxy to collect network packets exchanged between clients and servers. Fernandes et al. [2] use packet sniffer to capture packets from/to an official Second Life client. Kinicki and Claypool [5] conduct a similar, but more comprehensive measurement study. Ferreira and Morla [3] develop a Second Life testbed to systematically collect network traces resulted by various avatar actions. Oliver et al. [12] collect network traces from real Second Life users: 30 users participating in a virtual conference. They also gather network traces from controlled avatar actions in two regions. To the best of our knowledge, unlike our work, none of the network traces used in [1–3, 5, 8, 12] is publicly available.

3. SECOND LIFE: AN OVERVIEW

In this section, we give a high-level overview on Second Life, which is a popular virtual world. Second Life follows the client-server network model. Companies such as Linden Research [18], and non-profit organizations such as OSgrid [14] host Second Life servers, and users (or *residents*) use Second Life clients (called *viewers*) to connect to these servers. The viewers support various in-world actions, including stand, walk, run, fly, and teleportation. *Teleportation* refers to an instant change of an avatar's location. In general, user commands are sent from viewers to servers for processing and the results are transmitted back to viewers for rendering and displaying. Second Life is built and customized by users, and users create new 3D structures using *primitives* (or prims, which are basic 3D objects) as building blocks, and upload image files as textures for these 3D structures. These 3D objects are distributed to viewers *on-demand*, i.e., the servers transmit a 3D object to a viewer once the object is visible to that viewer. Viewers usually cache recently downloaded 3D objects.

Multiple servers are used in Second Life, and they can be classified into two groups: simulation and administration servers. Simulation servers run the logics to simulate physics in the virtual world, and execute user scripts written in Linden script language (LSL). LSL allows users to programmatically control objects' behavior. As simulating physics is computationally intensive, the virtual world is divided into more than 30,000 *regions*, where a typical region has a size of 256x256 m^2 and is managed by a simulation server. Each viewer is connected to one or more simulation servers, and is handed over to other simulation servers once the avatar moves into different regions.

Administration servers include login, user, space, data, and utility servers. Before participating in the virtual world, a viewer sends the avatar's username and password to the login server. Upon authentication, the login server determines the start location of the avatar and directs the viewer toward the simulation server manging that location. During simulation, the user server routes instant messages, and the space server coordinates all the simulation servers for a seamless, unified virtual world. The data server is essentially a central database used by other servers, and the utility servers handle miscellaneous tasks.

4. METHODOLOGY

4.1 Region Classes

Earlier works on Second Life consider very few regions. For example, Kinicki and Claypool [5] and Liang et al. [8] only consider three regions. While these studies reveal the implications of avatar/object density on network traffic, the number of regions is too small to *interpolate* the traffic patterns in regions other than those chosen ones. To address this limitation, we choose many regions with diverse avatar/object density as follows. We obtained the complete region list from an online database[1] on Aug 20, 2010, which contains 31,543 regions. We implement a region crawler, which iteratively teleports an avatar to each region and gathers the number of avatars/objects in it.

We successfully collected avatar/object density from 22,717 regions. Missed regions are either too busy, private, or with adult contents. The maximum number of avatars is 93, but most regions have very few avatars, e.g., 57% of regions have only one avatar. The maximum number of objects is 15,000, and the distribution is more uniform compared to that of avatars. We divide the regions into 25 *classes*. We let $\boldsymbol{X} = \{1, 2, 4, 8, 16, \infty\}$ and $\boldsymbol{Y} = \{0, 3000, 6000, 9000, 12000, \infty\}$, and define class $\boldsymbol{C}_{i,j}$ $(1 \leq i, j \leq 5)$[2] be all regions with number of avatars $x \in [X_i, X_{i+1})$ and number of objects $y \in [Y_i, Y_{i+1})$. In our experiment, we collect network traces from four random regions in each class. Although excluding private or adult regions, the selected regions are statistically representative in terms of object and avatar numbers. The collected trace shows statistics similarity to previous work in which adult and private regions were considered [20].

4.2 Actions and Scripts

Users interact with the virtual world through various *actions*, such as stand, rotation (yaw, pitch, roll), walk, run, jump, fly, teleportation, and grab. We consider the following actions in our experiments.

1. **Stand:** Stay in the current location. This can happen, e.g., when users chat by typing or speaking.
2. **Walk:** Walk straight, which can be due to keyboard inputs or LSL scripts. The default walking speed is around 2 m/sec, which may be changed through LSL scripts or affected by other factors such as terrains.
3. **Yaw:** Change the avatar's orientation. This occurs when users check the surroundings through avatar's eyes. Yawing 360 degrees takes about 5 secs.
4. **Run:** Run straight. This is similar to walk, except a different avatar animation is rendered. The running speed is around 5 m/sec.
5. **Fly:** Take off from the ground, fly straight, and land on the ground again. The flying speed is about 15 m/sec. Therefore, avatars can fly across a 256x256 m^2 region in 17 to 25 secs.
6. **Teleportation:** Change location instantaneously.

With these actions, we define several *scripts*. Each script starts at a random location and runs for one minute. We program the viewer to deliberately prevent avatars from crossing region boundaries. We consider the following scripts.

[1] `http://www.gridsurvey.com`.

[2] In this paper, we use bold symbols to represent vectors.

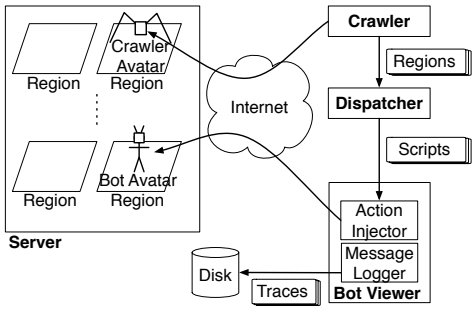

Figure 1: Second Life testbed setup.

1. **Stand:** Stand for one minute.
2. **Yaw:** Rotate avatar's orientation for 12 secs, and repeat this 5 times.
3. **Walk:** Walk in a random direction for 12 secs, and repeat this 5 times.
4. **Run:** Run in a random direction for 12 secs, and repeat this 5 times.
5. **Fly:** Fly in a random direction for 12 secs, and repeat this 5 times.
6. **Teleportation:** Teleport to a random location in the same region, wait for 12 secs, and repeat this 5 times.
7. **YWRFT:** Sequentially perform four actions: yaw, walk, run, and fly, each for 12 secs. Then teleport to a random location in the same region and wait for 12 secs.
8. **TFRWY:** Reversed YWRFT script.

For scripts other than Yaw, we turn an avatar's direction by directly setting its orientation, which takes virtually no time to complete.

4.3 Testbed Implementation

We design and build a testbed to collect Second Life network traces. We make the following decisions:

1. **Modifying a GUI viewer.** Viewers can be categorized into two classes: GUI viewers and text-based chat clients. We have augmented both a GUI viewer (Snowglobe [19]) and a chat client (TestClient of LibOpenMetaverse [10]), and used them to collect network traces. Our experimental results indicate that chat clients ignore several multimedia packet types, and may lead to unrepresentative traffic patterns.
2. **Logging packets at the viewer.** Packets can be logged: (i) at a viewer, (ii) at a proxy, and (iii) with a packet sniffer, such as Wireshark [21]. While using a proxy (or a packet sniffer) is easier than modifying a viewer to log network traces, additional networking and processing latency may result in deviated packet timestamps.
3. **Using official Linden servers.** While private servers can be set up using the open-source OpenSim [13] implementation, doing so prevents us from capturing the actual traffic patterns in live Second Life networks.

Following these design decisions, we implement our testbed as illustrated in Fig. 1. Our testbed contains two parts: Server on the left, and the measurement tools, including Crawler, Dispatcher, and Bot Viewer, on the right. We run the tools on a PC with 2.4 GHz Intel processor, which is connected to the Internet via a dedicated link with 10 Mbps bandwidth in both directions. The same link is shared with common office applications such as emails and web browsing. Next, we present our measurement tools in details.

```
1: Crawler collects avatar/object density of all regions and categorizes them into 25 classes
2: for all Class C_{i,j}, where 1 ≤ i, j ≤ 5 do
3:    Let C' be four random regions from C_{i,j}
4:    for all Region r in C' do
5:       Dispatcher compiles eight scripts S_r
6:       for all script s in S_r do
7:          Bot Viewer performs s
8:          Bot Viewer saves traces of s into files
9:       end for
10:   end for
11: end for
```

Figure 2: Pseudocode to collect network traces.

Table 1: Packet trace format

Field	Example	Description
Timestamp	12834907204	Unix Time in usec
Protocol	UDP	Transport protocol
Direction	in	Downlink or uplink
Remote IP	216.82.23.202	IP address
Remote Port	13001	TCP/UDP port
Packet Type	LayerData	Second Life packet type
Payload Size	81	Second Life packet size

Crawler. We implement Crawler based on TestClient of LibOpenMetaverse [10]. Crawler first logins to the virtual world with a starting region. It then checks its current location and aborts if it's not in the specified starting region. Last, Crawler sends a query packet to Server for the number of avatars/objects in that region. The regions are then classified into 25 classes (see Sec. 4.1 for details).

Dispatcher. Dispatcher is implemented in Python and Bourne shell. Dispatcher first generates the eight scripts mentioned above for each considered region. The avatar starts at random location in each script. All the scripts are saved for future reference. Dispatcher then calls Bot Viewer to follow each script and collect network traces.

Bot Viewer. We implement the Bot Viewer by adding two new components: Action Injector and Message Logger, to Snowglobe [19]. After logging in a region, Action Injector reads the script created by Dispatcher, and injects actions into Bot Viewer. Since we are doing this in the core engine of Bot Viewer, we can accurately control the action time and collect exact location information among other useful statistics. Bot Viewer exits after each script is finished. We perform several sanity checks in Bot Viewer, e.g., we rerun a script if a teleportation gets rejected or a login authentication fails. Message Logger logs all network packets, including their timestamp, size, type, and remote address. It also logs the avatar's location and surrounding avatar/object density.

4.4 Trace Collection

We systematically collect network traces of eight scripts from 100 regions with diverse characteristics. Fig. 2 shows the high-level steps used in trace collection. The for-loop starting from lines 2, 4, and 6, iterate through the considered region classes, random regions, and action scripts, respectively.

We collect network traces in two setups: *uncached* and *cached*. We consider both cases because viewer cache can significantly reduce network traffic amount [6]. For uncached experiments, we

Table 2: Script record format

Field	Example	Description
Type	Script	Record type
No. Actions	5	No. actions
Action 1	Walk 10	Action
Action 2	Stand 20	Action
...	...	...

Table 3: Location record format

Field	Example	Description	Field	Ex.	Description
Type	Location	Record type	Local Objects	185	Object density
Timestamp	12543524234	Unix time	Local Avatars	3	Avatar density
Region	Morris	Region name	Global Objects	949	Object density
Global Pos.	912,834,1	Pos. vector	Global Avatars	7	Avatar density
Local Pos.	22,127,2	Pos. vector			

Table 4: Action record format

Field	Example	Description
Type	Action	Record type
Timestamp	12543524234	Unix time
Action	Walk 10	Action

Table 5: Statistics trace format

Field	Example	Description
Region	Morris	Name of the region
Crawled Object Density	9484	Object density from Crawler
Crawled Avatar Density	10	Avatar density from Crawler
Global Object Density	9730.5	Mean global object density from Bot Viewer
Global Avatar Density	14.9	Mean global avatar density from Bot Viewer
Local Object Density	181.5	Mean local object density from Bot Viewer
Local Avatar Density	3.6	Mean local avatar density from Bot Viewer

instruct the viewer to clean its cache before running each script. For cached experiments, we only clean the cache before running the first script in a region, and for each region, we repeat the eight scripts twice without cleaning the cache. The first run of the eight scripts warms up the cache, and we collect network traces during the second run. We started the trace collection on Aug 25, 2010.

5. TRACES

5.1 Format

To assist readers better utilizing the traces, we processed the raw trace files into a well-structured and concise format. The trace files contain lines of *fields* separated by a pipe character (|). We define the fields in the following.

Packet trace. Packet trace files log the traffic to and from the Bot Viewer during each script execution. Each line represents a Second Life packet, and lines are sorted based on timestamp. Table 1 lists the fields of packet traces.

Location trace. Location trace file contains three types of *records*: script, position, and action. Each record is saved in a line. Script record appears at the beginning of each location trace file, and indicates the planned actions of every script. Table 2 shows its format. Location records are periodically saved with 1-sec interval. As presented in Table 3, a location record indicates region name, global/local position of our bot avatar, global avatar/object density in the region, and local avatar/object density at the current location. Action records are saved whenever the bot avatar performs a new action. The timestamps, actions, and action parameters are saved in action records, as illustrated in Table 4.

Statistics trace. Statistics trace reports avatar/object density of each region. The numbers are computed across all experiments, which provide readers the ground truth of the region characteristics. Table 5 lists its fields.

5.2 Analysis

We report the characterizing statistics of the collected traces. We consider three metrics: throughput, packet size, and interarrival time. We consider six scripts (YWRFT and TFRWY are skipped due to the space limitations). We first report the statistics of a randomly chosen region *Tokugawa*. This region has 2531 objects and 1 avatar, and our bot avatar encountered 93 local objects and 1 local avatar on average during the time of the trace collection.

Downlink traffic, uncached, Tokugawa. We show the Cumulative Distribution Function (CDF) of downlink traffic for uncached experiments in Fig. 3. Fig. 3(a) illustrates the correlation between avatar action and downlink traffic, which is also observed in several previous work, including [5]. Fig. 3(b) reveals that packets can be categorized into two groups: $\sim$200 bytes and $\sim$1000 bytes. We found that packets with size $\sim$1000 bytes are mostly multimedia packets, such as texture packets. Fig. 3(c) shows that actions impose insignificant impacts on interarrival time.

Uplink traffic, uncached, Tokugawa. We plot the CDF curves of uplink traffic for uncached experiments in Fig. 4. Fig. 4(a) reveals that the uplink throughput is lower by an order of magnitude comparing to downlink throughput (see Fig. 3(a)). Fig. 4(b) shows that about 95% of the uplink packets are smaller than 200 bytes. Fig. 4(c) shows that uplink interarrival time is higher than downlink, and Stand and Teleport have higher interarrival time.

Cached, Tokugawa. We plot sample CDF curves for cached experiments in Fig. 5. In Figs. 5(a) and 5(b), we observe a throughput reduction of about 50% in both downlink and uplink, compared to results from the uncached experiments (see Figs. 3(a) and 4(a)). Fig. 5(c) shows that the number of packets with size $\sim$1000 bytes are significantly reduced when cache is used, which indicates a decent amount of multimedia data were retrieved from the cache.

Downlink traffic, uncached, aggregated. Next, we report the aggregated statistics across 100 regions. We plot the CDF curves of downlink traffic for uncached experiments in Fig. 6. Uplink and cached results are omitted due to the page limitations; interested readers can generate them from our trace files [17]. The CDF curves in Figs. 6(b) and 6(c) are aligned with those in Figs. 3(b) and 3(c). However, comparing Fig. 6(a) against Fig. 3(a), we found that the correlation between action and throughput is *weaker* in aggregated form: the CDF curves in Fig. 6(a) are quite close to each other. This observation implies that avatar action may not be the strongest factor affecting the traffic pattern.

Correlation between avatar/object density and traffic pattern. To evaluate other factors affecting traffic patterns, we compute the mean throughput, packet size, and interarrival time across all action scripts of individual regions. We tried three types of density: crawled, global, and local, and found that local density has the strongest correlation with the traffic pattern. We then plot a few sample figures in Fig. 7 to illustrate how local avatar/object density affects downlink traffic patterns in uncached experiments.

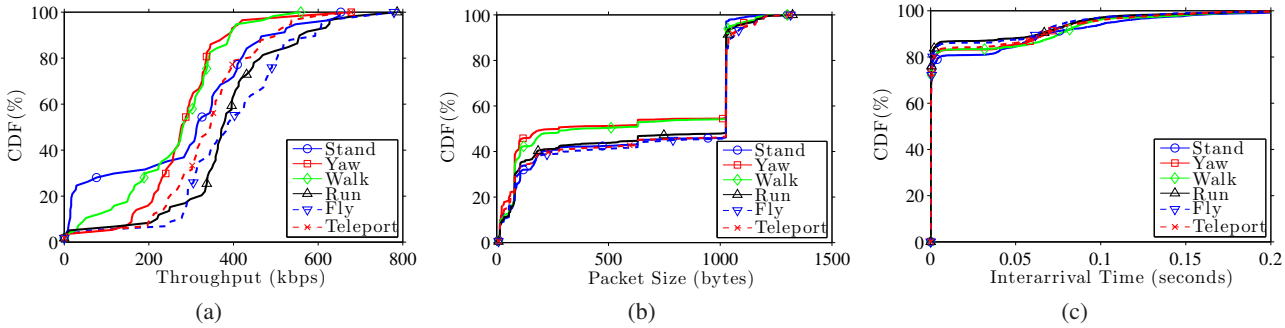

Figure 3: Downlink statistics of an uncached region: (a) throughput, (b) packet size, and (c) interarrival time.

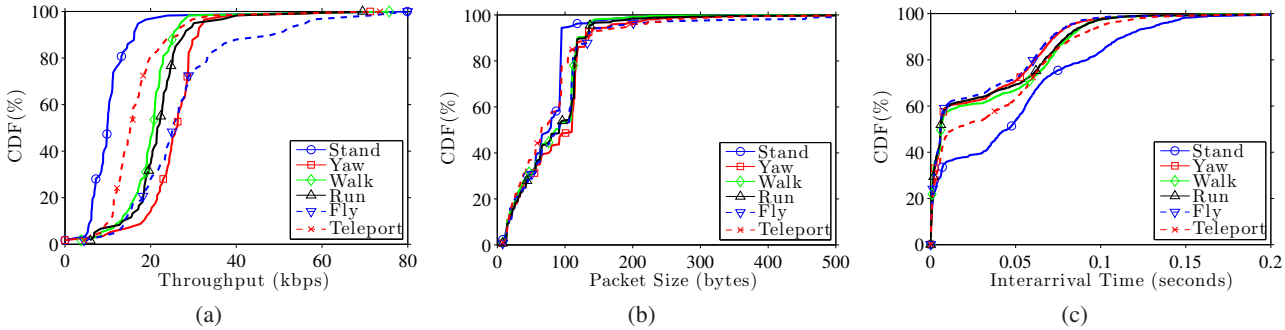

Figure 4: Uplink statistics of an uncached region: (a) throughput, (b) packet size, and (c) interarrival time.

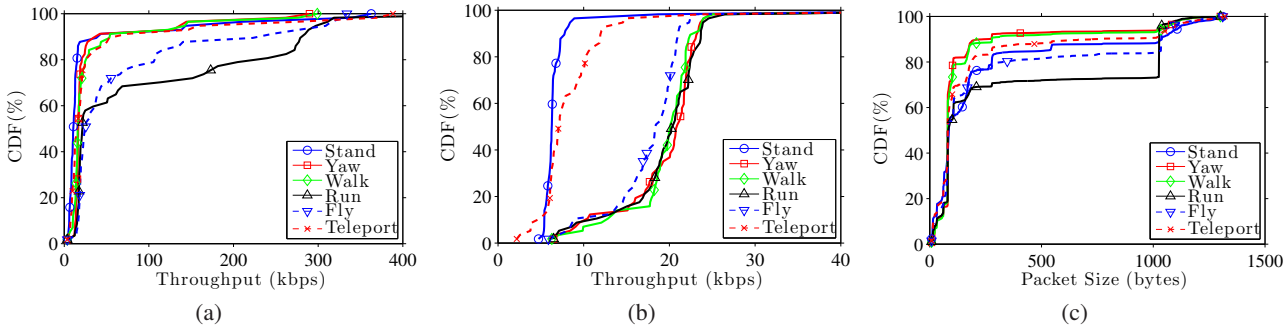

Figure 5: Statistics of a cached region: (a) downlink throughput, (b) uplink throughput, and (c) downlink packet size.

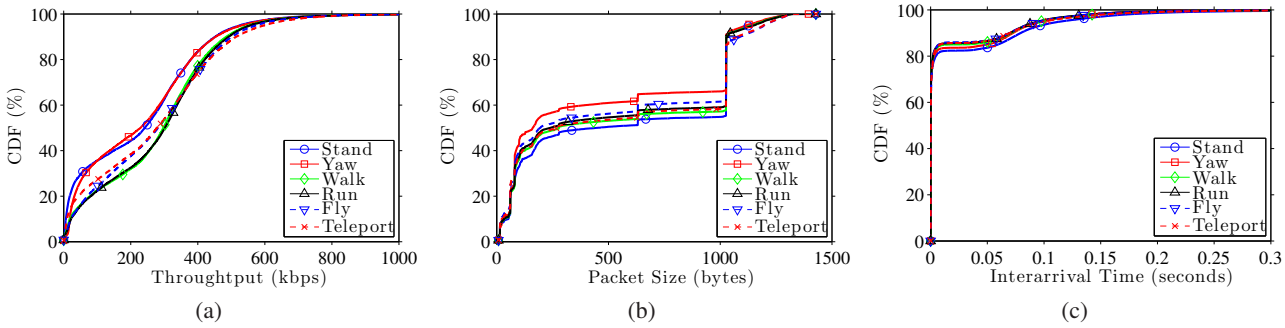

Figure 6: Aggregated uncached downlink statistics: (a) throughput, (b) packet size, and (c) interarrival time.

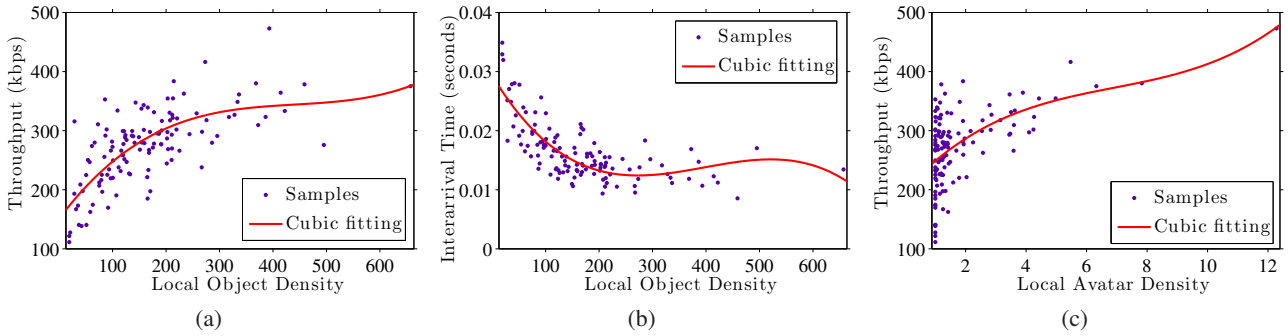

Figure 7: Correlation between avatar/object density and traffic pattern: (a) throughput vs. local object density, (b) interarrival time vs. local object density, and (c) throughput vs. local avatar density.

Fig. 7(a) shows a strong correlation between throughput and local object density, which can be modeled by a cubic function. Similarly, Fig. 7(b) reveals the correlation between interarrival time and local object density, and Fig. 7(c) illustrates the correlation between throughput and local avatar density.

6. APPLICATIONS

Traffic modeling in virtual worlds. Several research groups [1, 3] have pointed out that modeling network traffic of virtual worlds is valuable for: (i) ISPs to configure their networks, (ii) virtual world developers to achieve better Quality-of-Service (QoS), and (iii) research communities to generate synthetic traffic for simulations. Antonello et al. [1] model packet size and interarrival time using several common distributions such as beta, gamma, and log-normal. Ferreira and Morla [3] argue that there is a strong correlation between packet size and interarrival time, and propose to categorize the packets into multiple groups based on their sizes and then model packets in each group with a separate distribution.

A few approaches can be applied to refine the existing models in the literature. For example, packets of each packet type can be modeled with a different distribution. In addition, as showed in Sec. 5.2, region characteristics, such as avatar/object density, can be considered as *parameters* in traffic models. These extensions may not be possible without our extensive network traces.

Improving Quality-of-Service. Oliver et al. [12] observe that some packet types are very sensitive to packet loss and delivery delay, while others are more loss and delay tolerant. This leads to a possible QoS mechanism that prioritizes packets on their packet types for better user experience. Designing and evaluating such QoS mechanisms heavily rely on our network traces and the traffic models derived from them.

7. CONCLUSIONS

We implemented an automated testbed to systematically collect network traces from virtual worlds. We used the testbed to gather extensive traces from 100 diverse Second Life regions, which result in more than 60 hour traffic trace and more than 3 GB trace files. In comparison, most previous studies only consider a few regions, and more importantly, they did not make their traces public. We analyzed the virtual world traffic patterns using the collected traces. Our results are consistent with work in the literature, and also reveal new insights, e.g., local avatar/object density imposes clear implications on the downlink throughput. Our network traces [17] can stimulate the research on virtual worlds.

8. REFERENCES

[1] R. Antonello, S. Fernandes, J. Moreira, P. Cunha, C. Kamienski, and D. Sadok. Traffic analysis and synthetic models of Second Life. *Multimedia Tools and Applications*, 15(1):33–47, February 2009.

[2] S. Fernandes, F. Antonello, J. Moreira, D. Sadok, and C. Kamienski. Traffic analysis beyond this world: the case of Second Life. In *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'07)*, Urbana-Champaign, IL, June 2007.

[3] M. Ferreira and R. Morla. Second Life in-world action traffic modeling. In *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'10)*, pages 3–8, Amsterdam, The Netherlands, June 2010.

[4] Habbo Hotel official site, August 2010. `http://www.habbo.com`.

[5] J. Kinicki and M. Claypool. Traffic analysis of avatars in Second Life. In *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'08)*, pages 69–74, Braunschweig, Germany, May 2008.

[6] S. Kumar, J. Chhugani, K. Changkyu, D. Kim, A. Nguyen, P. Dubey, C. Bienia, and Y. Kim. Second Life and the new generation of virtual worlds. *IEEE Computer*, 41(9):46–53, September 2008.

[7] C. La and P. Michiardi. Characterizing user mobility in Second Life. In *Proc. of ACM Workshop on Online Social Networks (WOSN'08)*, pages 79–84, Seattle, WA, August 2008.

[8] H. Liang, M. Motani, and W. Ooi. Textures in Second Life: Measurement and analysis. In *Proc. of IEEE International Conference on Parallel and Distributed Systems (ICPADS'08)*, pages 823–828, Melbourne, Australia, December 2008.

[9] H. Liang, R. Silva, W. Ooi, and M. Motani. Avatar mobility in user-created networked virtual worlds: Measurements, analysis, and implications. *Multimedia Tools and Applications*, 45(1-3):163–190, October 2009.

[10] LibOpenMetaverse official site, August 2010. `http://www.libsecondlife.org`.

[11] Measurement and analysis of large distributed virtual environments, August 2010. `http://nemesys.comp.nus.edu.sg/projects/secondlife`.

[12] I. Oliver, A. Miller, and C. Allison. Virtual worlds, real traffic: Interaction and adaptation. In *Proc. of ACM conference on Multimedia systems (MMSys'10)*, pages 305–316, Scottsdale, AZ, February 2010.

[13] OpenSimulator official site, August 2010. `http://opensimulator.org`.

[14] OSgrid official site, August 2010. `http://www.osgrid.org`.

[15] Playstation Home official site, August 2010. `http://us.playstation.com/psn/playstation-home`.

[16] Second Life economic statistics, August 2010. `http://secondlife.com/statistics/economy-data.php`.

[17] Second Life network traces, September 2010. `http://12.71.54.173/sl`.

[18] Second Life official site, August 2010. `http://secondlife.com`.

[19] Snowglobe official site, August 2010. `http://snowglobeproject.org`.

[20] M. Varvello, F. Picconi, C. Diot, and E. Biersack. Is there life in Second Life? In *Proc. of ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT'08)*, Madrid, Spain, December 2008.

[21] Wireshark official site, August 2010. `http://www.wireshark.org`.