

Online Game Testing Using Scenario-based Control of Massive Virtual Users

Chang-Sik Cho*, Kang-Min Sohn*, Chang-Jun Park*, Ji-Hoon Kang**

**HD Game Research Team, ETRI, Korea*

***Dept. of Computer Science and Engineering, Chungnam National University, Korea*

cscho@etri.re.kr, sogarian@etri.re.kr, chjpark@etri.re.kr, jhkang@cnu.ac.kr

Abstract— The stability and performance of game servers are major issues in online games because the online game servers must be able to handle and verify a large number of connections. Test automation has been used in order to reduce testing time of online game servers.

In this paper, we propose the VENUS II system, which supports blackbox testing and scenario-based testing as well as simple load testing. The massive virtual clients automatically generate packet loads to test the stability of game servers like the existing approaches. The main difference, however, is that we used the game grammar and game map to describe the game logics instead of using dummy game client code. So the test client code doesn't need to be re-written when a new game is to be tested. And the complex scenarios such as attack, party play and waypoint movement can be tested and moreover effectively because the scenarios can be flexibly constructed by combining protocols and actions already defined in game grammar. We have applied our tools on the ELMA online game application to verify the effectiveness of our method.

Keywords— Online Game Testing, Massive Virtual Users, Load Test, Blackbox Testing, Game Grammar, Virtual Game Map

I. INTRODUCTION

A MMOG (Massively Multiplayer Online Game) is a multiplayer video game which can be played via a game server over the internet, with other players around the world. MMOGs can enable players to cooperate and compete with each other on a large scale, and include a variety of gameplay types, representing many video game genres. A MMORPG (Massively Multiplayer Online Role-Playing Game) is a genre of computer role-playing games in which a very large number of players interact with one another within a virtual game world. The virtual game world continues to exist and evolve while the player is away from the game. Online games like Aion[1] and World of Warcraft[2] are the representative of MMORPG games[3].

MMOGs are deployed using a client-server system architecture and might actually be run on multiple separate servers. The servers must be able to handle and verify a large number of connections. So the stability and performance of game servers become major issues in online games. To meet such requirements for game servers, large scale of game users actually participate the game to test its performance before deploying the game. In order to reduce such testing time, it is recommended using test automation. One key genre that has had reported success with test automation is that of MMOGs. They are not only easy to integrate into current game code, but also flexible enough to be reused in future games[3].

Game test automation has been used in The Sims Online (TSO) game. The TSO team used a subset of the main game client to create a test client in which the GUIs are mimicked via a script-driven control system[4]. VENUS (Virtual Environment Network User Simulator) system also has been developed to provide a beta test environment to the game developers to test the online games efficiently. With virtual client engine in the VENUS system, the user can easily simulate any kind of online games for beta test. The load limit point and bottleneck of a game server can be verified by simulating massively virtual clients in an online game environment[5,6].

The TSO automation and VENUS system have saved many person-hours, led to better game quality and been more efficient than manual testing. But they have some weak points.

First, test client code must be re-written when the target game for testing is changed. They all have used a subset of the main game client for test client code. So the test client code should be re-written as the target game for testing is changed. In our approach, the game grammar and virtual game map are used by which game logics are described. So when the target game for testing is changed, only game grammar and game map information are changed and the test client code needs not to be modified. The game packets can be re-generated according to the defined game grammar. In other word, we support blackbox testing.

Second, they don't support a scenario-based testing. In TSO, the testing has been achieved by unit of actions which are represented by simple script. In VENUS system, the actual game packets are captured and the loads are re-generated by simply modifying the captured packets. It is difficult for both to test interactive, conditional and iterative actions such as multi-user collaborative play. In VENUS II system, scenarios are created by combining game protocols and actions, which are defined in the game grammar. The conditional constructs and loop constructs are used in scenario construction for complex scenario.

Finally, they don't provide intuitive user interface. Normally, the game testers are not familiar with the programming language or script, so intuitive user interface should be supported. We support the scenario-based testing and the scenario is basically generated in the virtual map interface. The tester only selects the virtual user and inputs the actions with mouse clicks.

In this paper, we propose the VENUS II system, which supports blackbox testing and scenario-based testing as well as simple load testing. The game grammar and virtual game map are used to archive these goals. The game logics are described by defining game grammar and virtual game map, so the test client code itself needs not to be re-written when a new game is to be tested. The virtual clients automatically generate packets according to the game grammar. The complex scenarios such as attack, party play and waypoint movement can be tested by combining actions. And the tool manipulation is easy to use.

The rest of this paper is organized as follows: Section 2 explains the related works for online game testing. Section 3 defines the model of scenario-based virtual user system which supports blackbox testing. We will explain the components in the VENUS II system, the game grammar and the virtual game map. Section 4 shows implementation results. Finally, section 5 concludes our paper with future works.

II. RELATED WORKS

Developing and deploying massively multiplayer permanent state world is very difficult due to the distributed nature and large scale of the game system[3]. In general, the game developers use a set of beta tests to archive the stability of online game server. The general process of a beta test takes several steps as follows. First, the game company recruits a list of candidate testers to perform the beta test through the advertisement. Second, the selected beta testers play the game and send the feedback information to the game developers. Finally, the game developers analyze the error reports from the feedback and debug the game application. However, this ordinary testing process consumes a lot of development resources such as cost, time and etc. In many cases, the beta testers' error reports give a vague clue of an error to the developers and make it so hard for the developers to repeat the error environment in exactly the same condition

that the beta tester had. And this kind of testing strategy is hard to apply to the console based online game[3].

The server testing technology using virtual clients has been studied very actively. The industry-standard load testing solutions such as LoadRunner[7], QALoad[8] and e-Load[9], can emulate hundreds or thousands of concurrent users and prevent costly performance problems. These solutions enable to measure end-to-end performance, diagnose application and system bottlenecks and tune for better performance. They help reduce the costs and time required to test and deploy new applications and systems. But their key infrastructure components are web servers,[10,11] database servers and etc, so the game server component cannot be applied easily. An on-line game server employs complicated logic compared with a general server application. Therefore, it is difficult to test stability of an on-line game server application using a conventional packet reuse method[12]. Online game server testing introduces the new research issues into this area.

Game test automation has been used in The Sims Online (TSO) game. The TSO team used a subset of the main game client to create a test client in which the GUIs are mimicked via a script-driven control system. To the main game server, these test clients look identical to actual connected game player. To control the load testing, they used the LoadRunner[9], a commercially available load generation system. Bridge code hooks LoadRunner into the TSO test client and enables LoadRunner to control up to thousands of simulated users against a given candidate server cluster. Configuring the client to run without a GUI significantly reduces the memory footprint and allows many more test client per load generation box. Using existing game client has minimized the need to create new code, but test client code should be re-written when a new game is to be tested. The testing is archived by simple action script which supports no conditional and loop construct, not by test scenarios.

VENUS (Virtual Environment Network User Simulator) system has been developed to provide a beta test environment to the game developers to test the online games efficiently. With virtual client engine in the VENUS system, the user can easily simulate any kind of online games for beta test. The load limit point and bottle-neck of the game server can be verified by simulating massively virtual clients in an online game environment. Further, it is very helpful because it stores an internal game data from the game server to the database. The VENUS system has been applied to many different online games to test its usefulness. But, the dummy game clients should be re-written by using VENUS SDK (Software Development Kit) for a new game testing. And the testing is focused on the simple load test instead of scenario-based testing.

III. SCENARIO-BASED CONTROL OF MASSIVE VIRTUAL USERS

As previously mentioned, VENUS II system supports blackbox testing and scenario-based testing as well as simple

load testing. Also it provides intuitive user interface. These are archived by using the game grammar and virtual game map. The game logics are analyzed and described by defining game grammar and virtual game map, so the test client code needs not to be re-written when a new game is to be tested. In this section, the overall architecture, the game grammar and the game map will be explained.

A. Overall Architecture

Figure 1 illustrates an overall architecture of VENUS II system. VENUS II system consists of packet capturing tool, packet analyzing tool and virtual user control tool. The packet capturing tool collects packets transmitted between a game client and a game server and stores the packet to the packet DB in a list form. The packet analyzing tool reads and analyzes packet data from the packet DB and generates the game grammar and game map. The virtual user control tool constructs a test scenario and generates packet loads according to the game grammar.

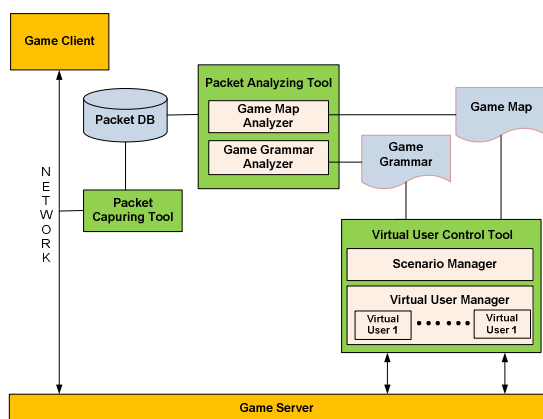


Figure 1. VENUS II Architecture

1) Packet Analyzing Tool

The packet analyzing tool analyzes packet data from the packet DB and generates the game grammar and game map. The packet analyzing tool is useful when the game code and/or game protocols are unknown to game tester. That is because the game developers and the game testers do not belong to same organization and the game developers don't provide the game source code to game testers because of security issue. So the game testers have no access to source code. In addition, the packet analyzing tool can be used to verify the test scenario of virtual users and to analyze action patterns of the real game users.

The game grammar and virtual game map are generated manually and/or partly automatically. All the packet protocols and actions need not to be analyzed and defined in

the game grammar. The subset of the protocols and actions is used for testing.

2) Virtual User Control Tool

Virtual user control tool consists of scenario manager and virtual user manager. Scenario manager constructs test scenarios by combining actions by using conditional and loop constructs. The examples of scenario are attack, party play and waypoint movement. The testers select the game objects in virtual game map and input the actions sequentially.

The virtual user manager initializes, terminates and controls the virtual users and generates an actual load based on the scenario. The virtual user is an activity entity of a single online game user. The game servers cannot distinguish a virtual user from a human game player. Generating a heavy load is needed to test a stability of the game server application. Since there can be a plurality of virtual user manager each generating a plurality of users, massive virtual users can be generated. Therefore the generated virtual users give a set of stresses to the game servers during the online game testing.

B. Game Grammar

The game grammar consists of packet structure rule, packet protocols and actions. The packet structure rule defines overall packet structure such as header rule, encryption rule, endian rule, timestamp rule and etc. The packet protocols define the transmitted individual packet messages between the game server and the game client. The actions are the meaningful user activities such as login, movement, combat and etc, and consist of a series of packet protocols.

The test scenario is constructed with the actions by using conditional and loop statements. The scenario is converted into a script internally when it is inputted by mouse click in the user interface. The scenario is conceptually described as follows.

Go to position A, B and C sequentially. In C, hunt 7 mobs, then go to D position. There, sell your items.

We will explain game grammar with example. Figure 2 show the two protocols of ELMA game. ELMA is a typical MMORPG game in its alpha test period and is used to test our system's feasibility. In ELMA, the packets always start with 0x21. Timestamp, packet length, protocol ID and parameters are followed. Its endian rule is little endian and it has no encryption rule. Header is 7 byte hexadecimal, protocol ID is 7 byte hexadecimal and the size of parameters is dependent on protocol ID.

In the protocol analyzing, the protocol ID and corresponding parameters are defined. In the example, packet 1 is move protocol and packet 2 is attack protocol. The move protocol ID is 0x02, 0x00, 0x00 and 0x00 and the attack protocol is 0x05, 0x04, 0xf2 and 0x07. The parameters of the move protocol consist of the current position, destination

position and the velocity speed. Each type is (float, float, float), (float, float, float) and float.

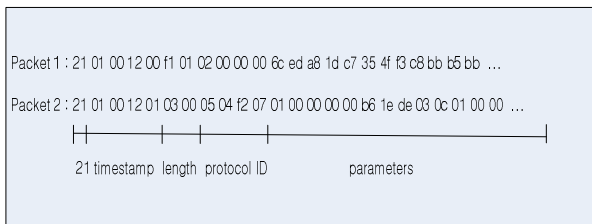


Figure 2. Packets Construction in ELMA Game

The process of creating games grammar is as follows. First, the overall structure of packets is analyzed. The packet structure consists of header, protocol ID and parameters. And packet header consists of the starting indicator, timestamp and length. Each component consists of TYPE and SIZE. The encryption rule and endian rule also are defined. The grammar of the packet structure is as follows

```
##PACKET_STRUCTURE::HEADER(TYPE,SIZE)::PROTOCOL_ID(
TYPE,SIZE)::PROTOCOL_PARAMETER(TYPE,SIZE)##
##PACKET_HEADER::0x21::TIMESTAMP(TYPE,SIZE)::LENGT
H(TYPE,SIZE)##
##ENDIAN_RULE::LITTLE##
##ENCRYPTION_RULE::NONE##
```

Then, individual protocols such as move, attack and skill are analyzed. Corresponding parameters are analyzed too. A protocol contains one or many parameters. The protocol ID and each parameter consist of TYPE and SIZE. The grammar of the packet protocols is as follows

```
##PROTOCOL_ID(TYPE,SIZE)##PARAMETER(TYPE,SIZE)##...#
# PARAMETER(TYPE,SIZE)##
```

At last, actions are defined. The action is the meaningful user activities such as login, movement and combat and has no conditional construct and loop construct. It is represented with a series of packet protocols and has the time constraints. Time constraints define the time interval between two protocols. The grammar of the packet actions is as follows.

```
##ACTION_NAME::SEND PROTOCOL1::RECEIVE
PROTOCOL2::WAIT n::SEND PROTOCOL4::RECEIVE
PROTOCOL4: WAIT n::...##
```

The process of generating games packets is as follows. First, the test scenario is converted into the action script. Second, actions are converted into the series of packets by referring the gram grammar. Third, the individual packets are created with the protocol ID and parameters. Finally, the complete packet will be generated by applying the packet

structure rule. Sending the packet can be delayed for the designated time period defined in the actions grammar.

C. Virtual Game Map

The virtual map is also generated by packet analyzing tool. At first, the virtual game map is blank. The game users or virtual game users randomly and massively move around the game world. They find where can go and where is not allowed to go. The game world is searched gradually by gathering the position information generated by game users' movement. Finally, the tester can place the landmarks such as village, store and hunting area manually.

The virtual game map displays the game object such as virtual users, real game users and NPC objects. The tester can view the position and status of game objects by selecting them in the virtual map. And the tester can command the actions for game object. The waypoint can be pointed by mouse click and group of virtual users can move according to the waypoint. In the virtual map, the virtual user can be copied and removed.

IV. IMPLEMENTATION RESULTS

We have applied VENUS II system to ELMA game testing to verify our system. The login/logout, movement (left, right, up, down) and attack actions are defined with grammar in the packet analyzing tool, and the massive packets are generated in virtual user control tool according to the game grammar. Waypoint is inputted in the virtual map and the selected virtual users moved the designated way. The stability of the ELMA game servers has been verified by our tools.

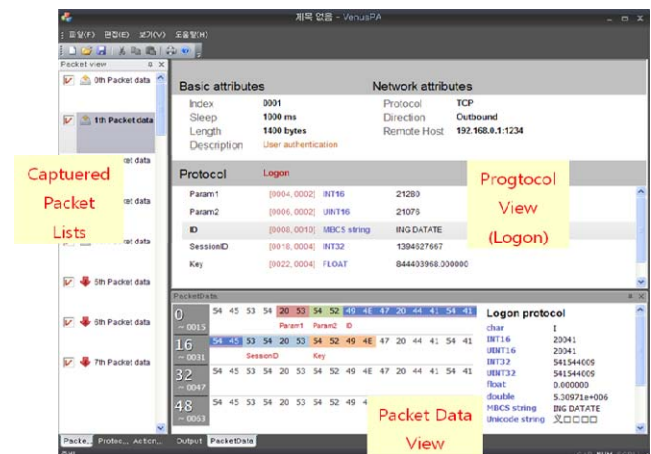


Figure 3. Screenshot of Packet Analyzer

Figure 3 is a screenshot of packet analyzing tool. At first, the network interface, IP addresses and port numbers of game servers and game clients are specified for selective capturing. The captured packet lists are displayed in the left hand side. Winpcap[13] library has been used for capturing and filtering

the game packets. In the right hand side, there are the defined protocols and action lists i.e. the game grammar. In the upper part of screenshot, the game grammar of login protocol is specified and in the lower part, captured binary data are colored according to the defined grammar.

Figure 4 is a scene of virtual user control tool. In the left hand side, there is the virtual map and in the right hand side, there is the virtual user manager. The virtual users(circles) and mobs(triangles) are displayed in the virtual map according to their position. The virtual user manager can create a single virtual user or virtual user group. The login/logout, movement and attack actions are provided by the windows button control. Figure 5 is a scene from ELMA game client when the virtual users are moving to designated position, i.e. waypoint scenario. The virtual user, mob and human real user are displayed in human real user's screen.

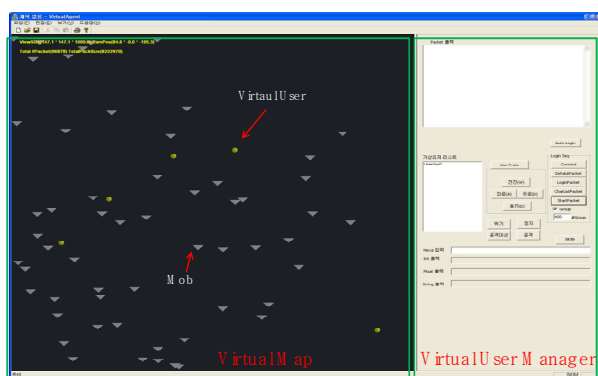


Figure 4. Screenshot of Virtual User Control Tool



Figure 5. Screenshot of ELMA Game Client

V. CONCLUSIONS

In this paper, we propose the VENUS II system, which supports blackbox testing and scenario-based testing as well as simple load testing. In the previous test automation, the

developers must provide their client code to the game testers, because test client code has based on the dummy game client code without a GUI. In our approach, the game logics are described with game grammar and virtual game map, so the test client code needs not to be re-written and re-used when a new game is to be tested. The virtual clients automatically generate packet load according to the game grammar. The complex scenarios such as attack, party play and waypoint movement can be tested by combining actions. And scenario control and virtual users' control are easy to use by providing windows controls and virtual game map.

To verify the effectiveness of our method, we have applied our tools on the ELMA online game application. The login/logout, movement (left, right, up, down) and attack actions are defined with grammar in the packet analyzing tool, and the massive packets are generated in virtual user control tool according to the game grammar. The stability of the ELMA game servers has been verified by our tools. Our future works are to improve our VENUS system, to provide more intelligent test scenarios and make it as a standard model for online game testing.

ACKNOWLEDGMENT

This work was supported by the IT R&D program of MKE/ITA. [2009-S-041-01, Online Game Quality Assurance Technology Development using Scenario Control of Massive Virtual User]

REFERENCES

- [1] AION. [Online]. Available: <http://aion.plaync.co.kr/>
- [2] World of Warcraft Community Site. [Online]. Available: <http://www.worldofwarcraft.com/>
- [3] Charles P. Schultz, Robert Bryant and Tim Langdell, *Game Testing All in one*, Thomson Course Technology PTR, 2005.
- [4] Larry Mellon, "Automated Testing of Massively Multi-Player Systems: Lessons Learned from The Sims Online," in *GDC 2003*, Spring, 2003.
- [5] Bum Hyun Lim, Jin Ryong Kim and Kwang Shim, "A Load Testing Architecture for Network Virtual Environment," in *Proceedings of 8th International Conference on Advanced Communication Technology*, 20-22 Feb, 2006, pp. 848-852
- [6] YungWoo Jung, Bum-Hyun Lim, Kwang-Hyun Sim, Hunjoo Lee, IlkyuPark, Jaeyong Chung, and Jihong Lee, "VENUS: The Online Game Simulator Using Massively Virtual Clients," in *Proceedings of AsiaSim 2004*, Oct, 2004, pp. 589-596
- [7] LoadRunner. [Online]. Available: <http://www.hp.com>
- [8] QALoad. [Online]. Available: <http://www.compuware.com/>
- [9] Empirix, OneSight. [Online]. Available: <http://www.empirix.com/>
- [10] S. Elbaum, S. Karre, G. Rothermel, "Improving web application testing with user session data," in *Proceedings of 25th International Conference on Software Engineering*, 3-10, May, 2003, pp. 49 - 59
- [11] D. A. Menasce, "Load testing of Web sites," *IEEE Internet Computing*, July-Aug, 2002, pp. 70-74
- [12] Wu-chang Feng, Francis Chang, Wu-chi Feng and Jonathan Walpole "Provisioning Online Games: A Traffic Analysis of a Busy Counter-Strike Server," in *Proceedings of the second ACM SIGCOMM Workshop on Internet measurement workshop*, Nov, 2002, pp. 151-156
- [13] WinPcap: The Windows Packet Capture Library. [Online]. Available: <http://www.winpcap.org/>