Network Element Simulation Based on Log Files

Bahadır Taşdemir, İzzet Çelik, Ferit Ünlü, Caner Reşber, and Ömer Sönmez
Alcatel-Lucent, İstanbul, Turkey
Email: {bahadir.tasdemir, izzet.celik, ferit.unlu, caner.resber, omer.sonmez}@alcatel-lucent.com

***Abstract*—There are a great variety of network elements in use at built systems. These network elements are uniquely configured and lively working under critical and non-critical jobs. In order to maintain the systems, unexpected situations must be handled as soon as possible, especially for the systems which are serving to the customers. However, it is not very easy to reproduce the same occurred incidents because of being ought to provide the same configuration or some of the access restrictions. Log based network element simulator is providing a real simulation of a network element behavior at any previous time depending on the log files. Assume that there is a product that is provided to a customer and it generated an error. When the customer reports the issue, just demanding the log file will be enough, and then the simulator will parse the log file and start to act like the NE depending on the request and response information where the situation occurred. The logs are kept when the error was occurred so the behavior will be the same with that specific time interval of the related NE.**

***Index Terms*—logging, network element, simulation, simulator, communications, network access**

I. Introduction

Network element (NE) is defined as a facility or equipment used to provide a telecommunications service. Such term also includes features, functions, and capabilities that are provided by means of such facility or equipment, including subscriber numbers, databases, signaling systems, and information sufficient for billing and collection, or used in the transmission, routing, or other provision of a telecommunications service [1]. Today's most commonly used systems are constructed of wide networks that contain a great variety of those network elements. All those network elements are installed mostly with different configurations. Those network element configurations contain wide range of settings and additional-constructions (e.g. blades [2]). Almost every system is private hence access to these systems and network elements are forbidden. If something unexpected occurs during a process and some transactions, the situation must be returned back to normal for sure by reproducing the incident. However, fixing the system does not preclude blocking it; the maintenance must not be harmful or must not break the access restriction rules to a private region or device.

Log Based Network Element Simulator is fed with the log files that are generated by the related NEs thus the behavior at the related time of the NE is identically simulated. The simulator parses the log file, generates request & response pairs and gives responses to the requests identical to the ones that are generated by the related NE. This provides; the real behavior, not breaking the connection restrictions, not accessing to the NE on site, not needing to create and configure an identical NE, not needing to keep a live device for maintenance, fast response to the system needs and device type independence on support.

II. Network Security

One of the main reasons of the network access problem is the control of the activities for the security issues.si

Access control contains what a user can do and what programs executing on behalf of the users are allowed to do. By this, access control seeks to prevent the activities that can break the security issues.

Controlling the access is not a complete solution for the security of a system; it must be enhanced with auditing. Access control is concerned with limiting the activity of legitimate users. It is enforced by a reference monitor which mediates every attempted access by a user (or program executing on behalf of that user) to objects in the system. The reference monitor consults an authorization database in order to determine if the user attempting to do an operation is actually authorized to perform that operation. Authorizations in this database are administered and maintained by a security administrator. The administrator sets these authorizations on the basis of the security policy of the organization. Users may also be able to modify some portion of the authorization database, for instance, to set permissions for their personal files. Auditing monitors and keeps a record of relevant activity in the system [3].

III. Related Work

There are different kinds of simulators for network elements that can be investigated at the following.

SONET Network Element Simulator: "A network simulator for use as a testing tool for a network management system includes an input device for inputting data containing network elements, their attributes, and information about their configuration, for storage in memory. User-Defined scenario instructions for specifying particular network behavior and user directives are input in real time, or in advance of simulation. The simulator includes an input/output processing component that receives commands from the network management system and user-scenario data or

Manuscript received January 3, 2015; revised June 30, 2015.

doi: 10.18178/ijsps.4.3.192-197

directives and forwards received commands and directives to a response generator processing component that generates appropriate responses in accordance with the network element behavior data stored in memory." [4]

SNMP Simulator: "Typical use case for this software starts with recording a snapshot of SNMP objects of donor Agents into text files using "snmprec.py" tool shipped with Simulator distribution. Or if you'd better query your donor SNMP Agent with Net-SNMP's snmpwalk tool, that information could also be used as a source of information by the Simulator. Another option is to generate snapshots directly from MIB files with "mib2dev.py" tool. The latter appears useful whenever you do not posess a physical donor device. Once you have your snapshots at hand, Simulator script "snmpsimd.py" could be run over the snapshots responding to SNMP queries in the same way as donor SNMP Agents did at the time of recording." [5].

IV. Network Logging

For security and other reasons, the network must be logged regularly and in a healthy way. But it must be applied meticulously:

Logging can cause problems at two extreme ways.

A. Too Little Logging

Too little logging is an obvious problem for the network. When a problem is identified, there may be insufficient information to reproduce what happened.

B. Too Much Logging

The opposite may also be true. Too much logging masks the situation and often leads to poor security practices. For example, security devices may generate enormous amounts of logs for later analysis, but the analysis may be performed only when some specific problem is encountered. The logs may accumulate into a larger and larger queue that may be deleted or trimmed before the data analysis is complete.

Too little logging, too much logging, and too little analysis are still important and noteworthy problems [6].

V. Log Based NE Simulator

Here, the main specialty of the Log Based Network Element Simulator is the practical production phase that takes a very short time to provide simulation (also it is a detailed process even for the network simulators [7]). Generally network elements simulators expect some configurations and preliminaries but Log Based Simulator cuts out this part off by taking the advantage of logging mechanism. Just acquiring the processed network element log file and providing into the log based simulator is sufficient for the preparation, an identical NE imitation would be ready within minutes.

A. Procedure

For the log based network element simulator, the steps are processed as (see Fig. 1):

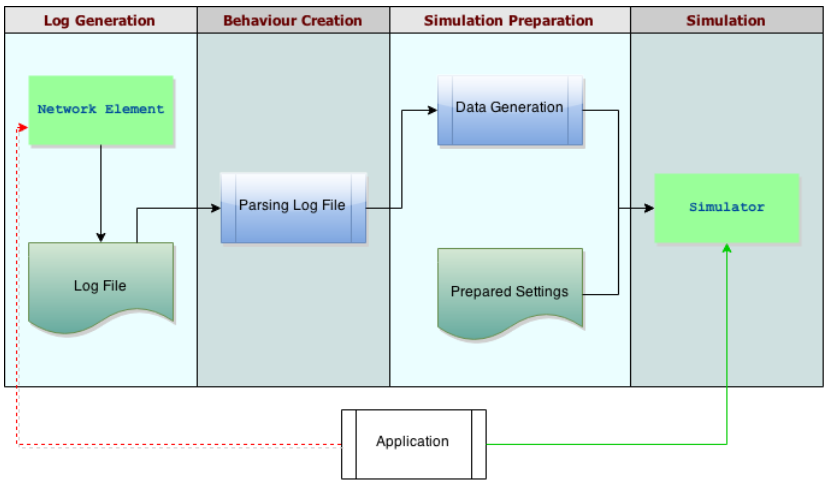

Figure 1. Log based network element simulation process

B. Network Element

Network Element is the device that is placed inside a network and processes the dedicated job.

C. Log File

Log file is created by the network element or by a process running on the network to be able to keep a track of the job.

D. Parsing Log File

Parsing log file is the process of analyzing and investigating the file to be able to detect the requests that are sent to the NE and responses that are created by it.

E. Data Generation

Data generation is the creation process of the data structure consists of requests and responses for handling the transaction traffic.

F. Prepared Settings

This file contains the needed settings for the simulator to be used on the network element simulation such as port number, user name, password etc.

G. Simulator

Simulator is the machine that is connected to the network and runs the log based NE simulator. After the

first run, this machine will behave identical to the NE which the log files are taken from.

The parsing mechanism is constructed by regex patterns that are catching permanent log blocks. Those patterns (date, time, send/receive and etc. see Fig. 2) are used at the logging phase of the project that runs on the related network element.

```
AG_REGEX = "<(\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2},\\d{3})>";
G_REGEX = "\\[(\\d{2}\\.\\d{2}\\.\\d{2} \\d{2}:\\d{2}:\\d{2}\\.\\d{3
ORMAT = "yyyy-MM-dd HH:mm:ss,SSS";
RMAT = "yyyy.MM.dd HH:mm:ss.SSS";

GEX = "<\\d{4}-\\d{2}-\\d{2} \\d{2}\\:\\d{2}\\:\\d{2},\\d{3}> " + GS
EGEX = "<\\d{4}-\\d{2}-\\d{2} \\d{2}\\:\\d{2}\\:\\d{2},\\d{3}> " + G
EX = "\\[\\d{2}\\.\\d{2}\\.\\d{2} \\d{2}\\:\\d{2}\\:\\d{2}\\.\\d{3}\
GEX = "\\[\\d{2}\\.\\d{2}\\.\\d{2} \\d{2}\\:\\d{2}\\:\\d{2}\\.\\d{3}
```

Figure 2. Log file parsing system regex patterns

When the data between those blocks are captured the needed data structure (see Fig. 3) is constructed to make the simulator ready. Here, a list with a sequential indexing mechanism is used to store request and response data and the response to the requests in an order (order in the log file) to provide data independence between same request character sequences.

Figure 3. Data structure class definition

Then the system runs on a server machine by the power of Java SSH library [8]. The running thread opens a connection per device connection request. When a new connection is created, the index of the request & response pairs' list is resets and starts from beginning. This provides requester to get blocks of request constructions. Also, the main index can be reset any time by sending a command to the simulator (see Fig. 4).

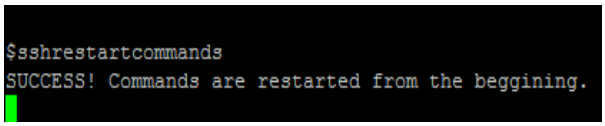

Figure 4. Sample default commands used in simulator

The network element simulator parses the log file's requests and responses with "exact characters" that the NE logged, so when a request is sent to the simulator, the request must be constructed with the identical characters which are logged to the file, otherwise the response will not be generated because the unavailability of request recognition.

VI. Usage

There are main steps must be processed to run and use the Log Based Network Element Simulator.

A. Configuration

The configuration of the simulator is created very handy to provide user friendliness. There is a file called "SSH_SIM.properties" (see Fig. 5) contains all of the configurations needed.

```
1 ssh.simulator.host=        1
2 ssh.simulator.port=23
3 ssh.simulator.logfilepath=./SSH_SIM_data/SSH_SIM_PAS_SAMPLE.log
4 ssh.simulator.is.delaying.active=false
5 ssh.simulator.is.command.echo.in.response.active=false
6 ssh.simulator.username=(     1
7 ssh.simulator.password=(       )
8 ssh.simulator.initial.prompt=r1sb01bgw:ACT-SCM:1.10(r0)>=1:gstool:vMGx>
9 ssh.simulator.command.refresh=false
```

Figure 5. NE simulator settings file (just a sample)

1) ssh.simulator.host

This setting keeps the host information of the network element simulator. It totally depends on the server machine that runs the simulator. The server's host address must be placed here.

2) ssh.simulator.port

A port number suitable for the simulator must be entered here. This port mustn't be used by any other processes on the server.

3) ssh.simulator.logfilepath

In this line, the file path of the log file that is going to be used by the simulator is given. The path must be correct to start the simulator properly.

4) ssh.simulator.is.delaying.active

While the NE's working, there some delays happen because of the network connection transactions. If the behavior is demanded to be identically the same with the delaying times than this setting must be set to "true".

5) ssh.simulator.is.command.echo.in.response.active

When a request is sent to a network element, the response comes with a command echo which contains the sent request. If this is a demanded behavior for the simulator then this setting must set to "true". Otherwise it can be set to "false".

6) ssh.simulator.username

While establishing a connection to an NE, a username & password credentials must be provided for the security. This field is for the username.

7) ssh.simulator.password

This field is for the second credential "password".

8) ssh.simulator.initial.prompt

As soon as connected to an NE, it produces an initial prompt that indicates it is ready. Here, it can be set for the simulator.

9) ssh.simulator.command.refresh

As described before, simulator executes responses to the given commands with the same order where the request & response pairs stated inside the parsed log file. If the request blocks are demanded to send in a random order than this feature must be set to "true" to let simulator search each command from the beginning in the case when the command is not found in the next place of the index.

B. Simulator Installation to a Server

To install the simulator to a server machine, the files of it must be sent to a specific folder of the target but it

mustn't be forgotten that the machine must have a JVM installed on it because the simulator is developed by using Java Programming Language and it is a Java Executable File (see Fig. 6).

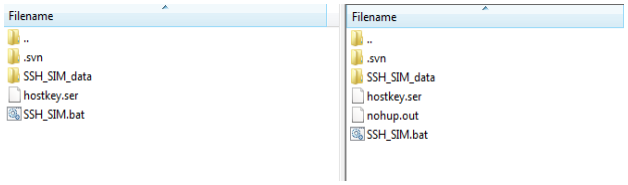

Figure 6. Simulator transmission using filezilla

C. Running the Simulator

As accomplishing all the needed settings for the simulator now it is time to run it on a demanded server machine. Here a linux server is used for the sample.

1) Connect to the server

Here the connection is made via putty (see Fig. 7).

Figure 7. Server connection via putty

2) Access to the file location

The location must be switched to the simulators path (see Fig. 8).

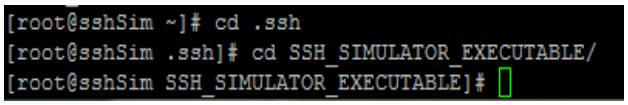

Figure 8. Simulator location access

3) Start the simulator

The simulator must be started as a process inside the server machine to let it open even being disconnected to the machine. The process is provided by using the linux process starting commands (see Fig. 9 and Fig. 10).

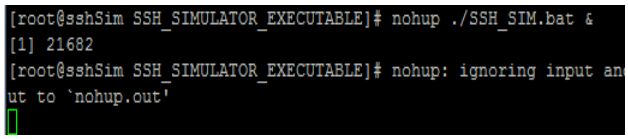

Figure 9. Starting the simulator process

Figure 10. Checking if simulator started with "ps-ef" linux command

D. Connecting to the Simulator

To connect to the simulator, the IP of the server machine and the port number of the simulator must be achieved (see Fig. 11 and Fig. 12).

In Fig. 12, the values are taken from the settings file as mentioned before.

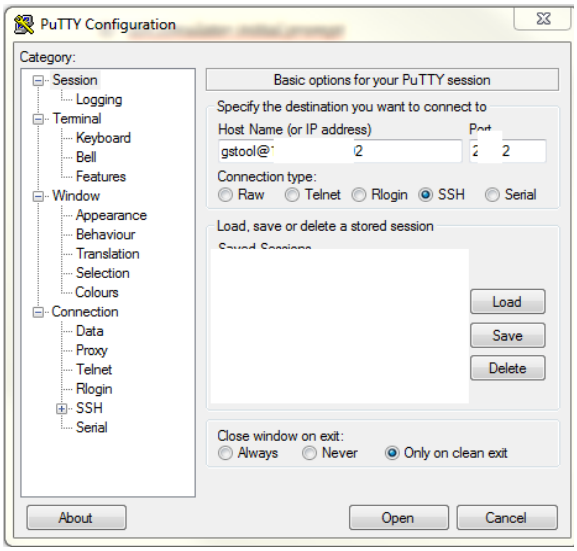

Figure 11. Simulator access

Figure 12. First connection to the simulator

E. Using the Simulator

By completing the simulator creation steps, an imitation of the related NE is provided to use at any dedicated process. Here some commands placed inside the log file will be tested to show how the simulator is running (see Fig. 13).

```
[04.02.15 12:08:17.720][SEND] view performance statistics interval
[04.02.15 12:08:18.039][RECV] view performance statistics interval
APV performance history interval = 15 minutes.
r1sb01mgw:ACT-SCM:1.10(r0)>=127:gstool:vMG1>

[04.02.15 12:08:18.039][SEND] def vmg scope none
[04.02.15 12:08:18.375][RECV] def vmg scope none
r1sb01mgw:ACT-SCM:1.10(r0)>=128:gstool:vMGx>

[04.02.15 12:08:18.375][SEND] def vmg scope 2
[04.02.15 12:08:18.735][RECV] def vmg scope 2
r1sb01mgw:ACT-SCM:1.10(r0)>=129:gstool:vMG2>

[04.02.15 12:08:18.735][SEND] view performance statistics interval
[04.02.15 12:08:19.059][RECV] view performance statistics interval
APV performance history interval = 15 minutes.
r1sb01mgw:ACT-SCM:1.10(r0)>=130:gstool:vMG2>

[04.02.15 12:08:19.059][SEND] def vmg scope none
[04.02.15 12:08:19.407][RECV] def vmg scope none
r1sb01mgw:ACT-SCM:1.10(r0)>=131:gstool:vMGx>
```

Figure 13. Sample log file

In the Fig. 11, a sample log file which contains some request & response pairs is shown. Here, the line contains

"[SEND]" patterns holds the request and the line contains "[RECV]" pattern holds the response data. When the command "view performance statistics interval" is sent to the simulator the response is taken immediately (see Fig. 14).

```
$view performance statistics interval
APV performance history interval = 15 minutes.
r1sb01mgw:ACT-SCM:1.10(r0)>=127:gstool:vMG1> def vmg scope none
r1sb01mgw:ACT-SCM:1.10(r0)>=128:gstool:vMGx> def vmg scope 2
r1sb01mgw:ACT-SCM:1.10(r0)>=129:gstool:vMG2> view performance statistics interval
APV performance history interval = 15 minutes.
r1sb01mgw:ACT-SCM:1.10(r0)>=130:gstool:vMG2> def vmg scope none
r1sb01mgw:ACT-SCM:1.10(r0)>=131:gstool:vMGx> view ethernet alarm configuration
ETH port Error-Counters Alarm state: disable
ETH port Error-Counters Alarm thresholds:
Lower threshold: 2
Upper threshold: 10
r1sb01mgw:ACT-SCM:1.10(r0)>=132:gstool:vMGx>
```

Figure 14. Requests are run on the simulator

When the requests placed inside the log file are sent to the simulator, the responses are produced as placed at the same log file. The structure of the log file is not the main concern; here the logged data usage for the simulation is important.

VII. Symptoms

When something unexpected happens with a network element the incident must be investigated. For most of the times, the NE is either cannot be reachable or restricted to be reached. Also, the identical NE is very hard to find apart from the live system with the exact configurations. If the incident causes a critical issue then the things start to become more complicated.

As hard as the identical network element creation is, also when there needed to be a network constructed with more than one NE, it is really hard to find all of them at the same time and place. By using the log based network element simulator, more than one network element simulation instance can be created and there can be a network constructed with those NEs. This will decrease the money cost of the work as well as decreasing the needed time and work effort.

In addition to the network element simulation, log based network element simulator also decreases the process time runs on the network for a dedicated job (see Fig. 15).

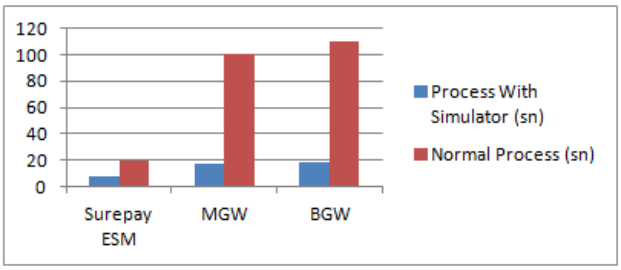

Figure 15. Performance of network application

With the log based network element simulator, just the log files generated by the related NE are needed. Within a couple of minutes, NE replication is live and ready to be processed. With this availability, the connection, testing, running commands, applications and etc. become very handy and this provides better systems, applications and maintenances.

VIII. Conclusion

Log based network element simulator provides an identical replication of the network elements currently running on the live (mostly critical) systems to test, debug, develop new product and maintain it. The little interventions into networks will be able to done within minutes and the systems won't even be affected by this.

The usage area of the log based network element simulator is not limited with just the NEs; it can simulate any device that generates logs and works connected to a network, such as a call simulator.

For the possible future works of the project, there can be a very usable interface that enables users to easily create network element simulators at any time and amount. This platform provides the serving of the multiple users at a time at any number of network elements thus the users and servers won't be blocked and the main processes would go on without being blocked.

Simulating an NE by log files that are generated under some processes provides a better workaround for testing and development phases of the network projects.

References

[1] Certiorari to the United States Court of Appeals for the Eighth Circuit, Verizon Communs. Inc. v., FCC, 535 U.S. 467, 492, 2002.
[2] Blade server definition. [Online]. Available: http://searchdatacenter.techtarget.com/definition/blade-server
[3] R. S. Sandhu and P. Samarati, "Access control: Principle and practice," *IEEE Communications Magazine*, vol. 32, pp. 40-48, Sep. 1994.
[4] SONET network element simulator. [Online]. Available: http://www.google.com/patents/US6108309
[5] SNMP simulator. [Online]. Available: http://snmpsim.sourceforge.net/
[6] P. W. Dowd and J. T. McHenry, "Network security: It's time to take it serious," *Computer*, vol. 31. pp. 24-28, Sep. 1998.
[7] N. Golmie, A. Koenig, and D. Su, "The NIST ATM network simulator: Operation and programming," NISTIR 5703, U.S. Department of Commerce, Aug. 1995.
[8] Apache MINA. [Online]. Available: http://mina.apache.org/sshd-project/

Bahadır Taşdemir was born in Batman, Turkey on 1988. He studied computer science in department of computer engineering at Dokuz Eylul University between 2009 and 2013. He worked at Turkcell after graduating in 2013 as a consultant software developer. Currently, he is working at Alcatel-Lucent as a senior software developer. His research interests are simulation & emulation tools, artificial intelligence.

Izzet Çelik was born in Tokat in 1978. He holds a bachelor's degree of computer science from Bilkent University. He worked as software developer at Telenity and Sony Eurasia ISC. Currently, he is employed as solution architect at Alcatel-Lucent. His interests are tool development and automation in IMS parameter auditing, architectural improvements in parameter auditing tools.

Ferit Ünlü was born in Izmir, Turkey on 1982. He studied engineering physics at Istanbul Technical University in 2005. He started to study computer engineering master degree at Bahcesehir University in 2009. Since 2005 he has been working as a java software developer for banking and telecom companies. Currently, he is working at Alcatel-Lucent as an expert software developer.

Caner Resber was born in Malatya, Turkey on 1990. He graduated from computer engineering at Anadolu University in 2013. Currently, he is working at Alcatel-Lucent as software developer. He is interested in simulation & emulation tools, high performance computing and cloud solutions.

Ömer Sönmez was born in Izmir, Turkey on 1988. He graduated from Anadolu University Department of Computer Engineering in 2011. Currently, he is working as a senior software developer at Alcatel-Lucent. Big Data, Java, Android development are among his interests.