

Research Article

Single-Board-Computer-Based Traffic Generator for a Heterogeneous and Hybrid Smart Grid Communication Network

¹Do Nguyet Quang, ¹Ong Hang See, ¹Lai Lee Chee, ¹Che Yung Xuen and ²Shashiteran A/L. Karuppiah

¹Department of Electronics and Communication Engineering, Universiti Tenaga Nasional (UNITEN),
Selangor 43000, Malaysia

²Tenaga Nasional Berhad Research, Selangor 43000, Malaysia

Abstract: In smart grid communication implementation, network traffic pattern is one of the main factors that affect the system's performance. Examining different traffic patterns in smart grid is therefore crucial when analyzing the network performance. Due to the heterogeneous and hybrid nature of smart grid, the type of traffic distribution in the network is still unknown. The traffic that popularly used for simulation and analysis no longer reflects the real traffic in a multi-technology and bi-directional communication system. Hence, in this study, a single-board computer is implemented as a traffic generator which can generate network traffic similar to those generated by various applications in the fully operational smart grid. By placing in a strategic and appropriate position, a collection of traffic generators allow network administrators to investigate and test the effect of heavy traffic on performance of smart grid communication system.

Keywords: Kernel space, network performance, network traffic, smart grid, stress test, traffic generator

INTRODUCTION

Power is generated at the power stations, transported over the transmission system and then delivered to the end users via the distribution network. Flows on the distribution network are generally one way. However, in smart grid, electricity and information flow in two-way direction. Smart grid is an electricity distribution system where information and communications technologies are added to the existing power network. It helps deliver electricity more efficiently and reliably, for example re-routing power, shifting loads, managing outages on the network, maximizing the electricity distributed, predicting failures and hence reducing maintenance cost (Nadim *et al.*, 2010). Previously, distributor can only view the power flows until the substations. But now with smart meters, they can see what is going on in their networks down to the end consumers. It is a powerful tool for managing the network since it can adjust loads remotely. Smart grid transforms the way that distributors have operated in the past. It helps them to carry more electricity through their existing infrastructure without digging up the street to lay new cables and respond quickly to the changes in consumer demand. In addition, it also provides customers with the real time information they need. They can save power by running their home

appliances when the power is at its cheapest or shutdown when peaks occur.

Since the smart grid communication network is made up of a variety of communication devices, the network traffic is one of the main concerns when evaluating the network performance. Using proper tools to generate representative network traffic is therefore very important. Currently, there are a large number of tools and utilities available with different functions to serve different purposes. Some tools are used for laboratory testing where they can be used to analyze and evaluate the performance of new systems with the implementation of real networking hardware (Gang *et al.*, 2010). Meanwhile, there are some other tools that can also be used for emulation and simulation environments. Without the tools to generate realistic network traffic patterns, new systems might operate with unpredictable behavior and sometimes unacceptable performance when they are actually implemented in real environments. Especially in smart grid, the traffic pattern is complicated since it is generated from a large number of applications in the network. Diverse communication technologies in smart grid also create major challenges for conventional methods, as the legacy techniques are designated only for a single communication technology or standard in the network. Due to the heterogeneous and hybrid communication

Corresponding Author: Do Nguyet Quang, Department of Electronics and Communication Engineering, UNITEN, Selangor 43000, Malaysia

This work is licensed under a Creative Commons Attribution 4.0 International License (URL: <http://creativecommons.org/licenses/by/4.0/>).

paradigm of smart grid, the characteristics of traffic carried in the network are not clearly known (Zhong *et al.*, 2010). Therefore, a traffic generator is implemented in this study with a purpose of allowing administrators to analyze and evaluate the network performance of a heterogeneous and hybrid smart grid communication system.

MATERIALS AND METHODS

Smart grid test-bed:

Network infrastructure: TNBR-UNITEN smart grid test bed is used to serve as a control environment for carrying out network tests to address various issues on utility communication network. The main focus is on the utility communication for the distribution network, from 33 or 11 kV to the consumers. In this smart grid testbed, there are totally 15 substations as shown in Fig. 1. A server room acts as a control center which monitors all the power substations remotely. A display system is located at the control center for monitoring and controlling purposes. At the remote side, a number of devices are installed at each substation. Initial testing will be carried out in the lab under simulated and control environment. Field tests will be conducted with single board computing systems developed in the testbed.

Communication architecture: There are three main components in the hierarchical architecture of smart grid: Home Area Network (HAN), Neighborhood Area Network (NAN) and Wide Area Network (WAN). HAN is an information and communication network that allows access to in-home appliances to support different distributed applications. NAN provides connection between smart meters and local access point by collecting data from multiple HANs and delivers the data to a data concentrator. WAN is the data transport network that allows the grid and core utility systems communicate with each other by carrying metering data to the control centers. It forms the communication backbone and conveys communication between electrical devices and the control centers. Electrical devices are installed at different locations in the network, along transmission line and in the substations. These devices capture data and transport information to control centers through wide area networks. At the same time, the control centers also monitor and manage the electrical devices on the network through WAN (Wenye *et al.*, 2011).

Communication is a crucial component of smart grid as the next generation intelligent and reliable power delivery system. In smart grid communication network, no single technology can cater for the entire power delivery grid system. Therefore, both communication mediums, wired and wireless are proposed in this smart

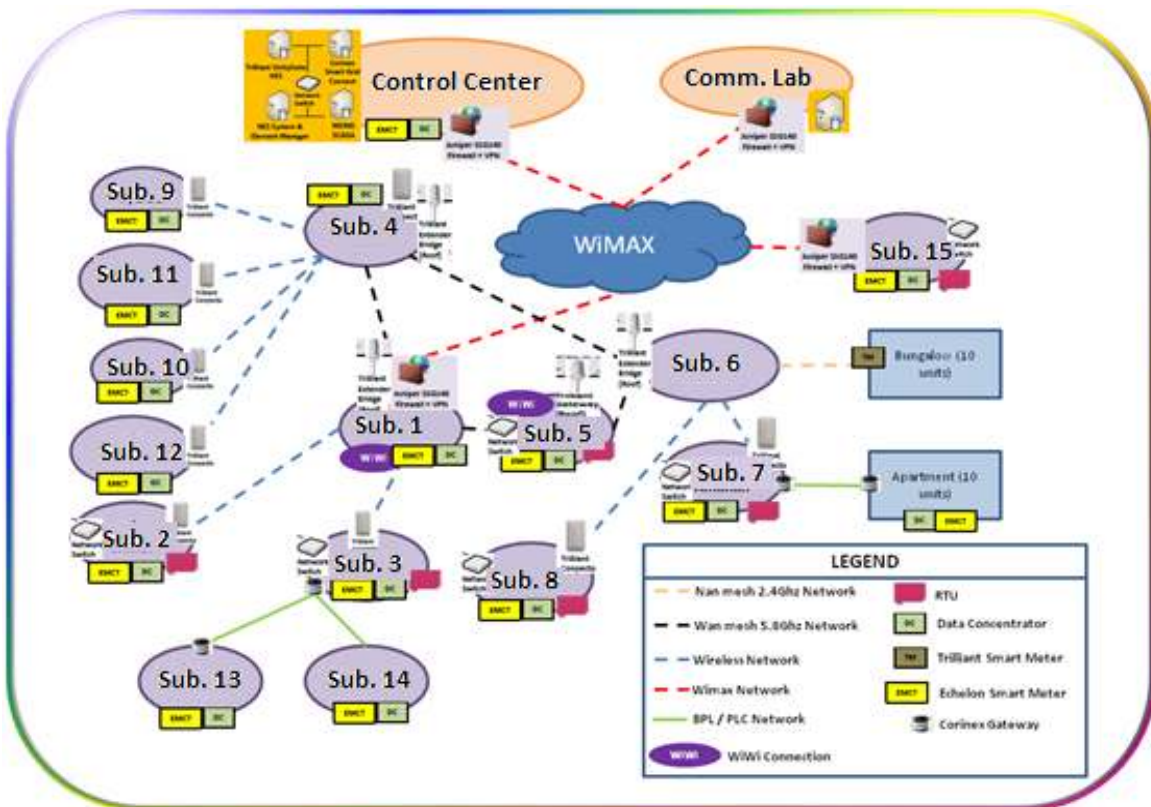


Fig. 1: Smart grid network diagram

grid testbed. The communication technologies that support the network include Power Line Communication (PLC), unlicensed Radio Frequency (RF) and WiMax/4G.

LITERATURE REVIEW

Traffic generator: Due to the large scale nature of smart grid, traffic will be generated by diverse applications and passed through various types of networks. A mixture of both real-time and non-real-time traffic will be generated and distributed across different parts of smart grid (Zhong *et al.*, 2010). As a result, there is a high possibility that bottleneck might occur. A bottleneck is a point on a network where the flow of data is limited by network components or resources. The flow of data depends on the network bandwidth. If the volume of the generated traffic is larger than what is supported by the network capacity, the network bottleneck will happen. A bottleneck slows down data flows, which in turn reduces the efficiency and productivity of data transmission across a network. A common factor that causes bottleneck is network interruption. For example, the presence of high-resolution graphics on a TCP/IP network will create a bottleneck in the data flow, causing the data to move more slowly in the transmission process. Network bottleneck also happens because too many devices are connected to the same switch. If all of these devices operate simultaneously, the switch might not be able to handle and hence causing problems which affect the network performance. Moreover, bottleneck conditions can also develop due to hardware failures. For instance, data flows across the network following multiple paths or links. If several links are down, all the traffic that is supposed to be carried by these links now fall into only a few links that are left. This will result in a bottleneck if these links are unable to manage the overload traffic volume.

Muller *et al.* (2012) investigated the traffic volume created by smart grid services and specifies the types of traffic in a distributed simulation environment under various scenarios. Different devices were included in these scenarios such as PLC data concentrator, RF base stations and wired concentrators. The study compared different technologies together with various traffic engineering approaches in order to validate the scalability of the proposed Information and Communication Technology (ICT) system architecture. These technologies include wired (DSL and cable), wireless (WiMax, GPRS, etc.) and Power Line Communication (Broadband and Narrowband). The impact of Machine-to-Machine (M2M) traffic on the QoS was also investigated to examine the effect of additional load on the system. A traffic generator was used to create smart metering data based on the Smart Message Language. It was used to transmit smart metering values on TCP layer in small packages. The

results obtained from the investigation were used to analyze the network performance depending on available ICT, as the ICT components represent the higher layer traffic generators of the smart grid infrastructure.

Similarly, a test bed was developed by Julien *et al.* (2010) to study the Precision Time Protocol (PTP) and determine the impact of new features and requirements for time synchronization on the performance of smart grid. A network traffic generator was built as a part of the testbed to examine network load scenarios. The network traffic in the testbed was divided into two paths, one for configuration of the nodes and switches, while the other is dedicated to PTP and substation traffic. The first path was established using the Ethernet port of the host computer and the second path was set up using the Ethernet port on the PTP devices. Network traffic scenarios were used to determine the influence of the network loads on the synchronization performance. The network traffic models were utilized as network profiles that provide a framework for traffic load patterns within the substation and smart grid.

Network traffic deals with a massive amount of data flowing to or from various devices in smart grid and different types of traffic are generated by these devices. As described by Paricio *et al.* (2010), traffic can appear in different forms, including both uni and bi-directional. These traffic types come from the nature of numerous applications in smart grid and each communication protocol implements them in different ways. The study proposed a network management system designed to control diverse applications in smart grid network. This system has an ability to monitor the network availability, extract data and configure various services for smart grid. The management architecture consists of two basic layers, namely network mediation layer and management applications layer. In the network mediation layer, the Mediation Device (MD) controls a large number of other smart grid elements, including a substation, groups of sensors, or AMI concentrators, which in turn connected to tens to hundreds of meters. The MD controls different types of traffic coming from these devices and performs traffic profiling for network applications using SNMP protocol. The traffic profiling at MD is in the form of uniform traffic distribution to ensure the low bandwidth requirements of the network.

According to Victor *et al.* (2010), a large volume of data will be generated by smart grid applications, not only from the power generators, but also from the customers and the distribution system. Thus, traffic congestion might occur in the network. In addition, it happens not only because of the heterogeneous and hybrid nature of smart grid, but also due to network attacks. A typical attack happens when a large volume of traffic is sent simultaneously and continuously to the network nodes or even the control centre. As a result, normal data cannot be handled properly.

Table 1: Different types of traffic generators

Tool	GUI	User space	Kernel space	Server-client configuration
PackETH	√	√	×	×
Traffic	√	√	×	×
MGEN	×	√	×	×
Packgen	×	√	×	×
PacGEN	×	√	×	×
Mtools	×	√	×	×
Rude and crude	√	×	√	√
Kute	×	×	√	√

The effects of traffic attacks on delay in smart grid communication network are also discussed by Zhuo *et al.* (2010). Due to consistent monitoring of electricity devices, a large amount of traffic flows are periodic. As network traffic in power networks is time-critical, delay is more concerned. A number of experiments were carried out in order to evaluate several types of attacks against the power network. Attack model was selected to determine its impact on the network availability. Traffic flood was chosen to be an attacker and Iperf was used to generate traffic over the testing network. Iperf is one of the most common network traffic generators. In this experiment setup, it was used to generate legitimate but useless UDP traffic to occupy the communication channel.

It is also mentioned by Tarek *et al.* (2010) that large traffic is generated from a large number of applications, each transmitting a small amount of data periodically. Applications such as smart meters send data at a low rate. However, they are implemented in a huge number of nodes. Those nodes collectively create large amounts of traffic, causing congestion on the network. As the traffic gets nearer to the control centre, its amounts as well as the degree of congestion increase. As a result, a significant amount of packets will be lost and thus increasing the rate of packet retransmission, leading to a significant amount of energy loss and delivery delay.

In recent years, more and more different types of traffic generators were created depending on different network protocols and according to different layers in the OSI model. The characteristics of several traffic generating tools are described in Table 1.

PackETH (Retrieved from: <http://packeth.sourceforge.net/sourceforge/Home.html>) is a packet generating tool for Ethernet. Running on Linux and some ports for Windows and MAC, PackETH allows user to create and send any possible packet or sequence of packets on the Ethernet link. It supports several protocols, including Ethernet II, UDP, TCP, ICMP, IPv4, IPv6 and so on. PackETH provides a number of features to support sending sequence of packets such as number of packets, delay between packets, UDP payload, etc.

Traffic (Retrieved from: <http://robert.rsa3.com/traffic.html>) is a network traffic generator used to test the effects of massive amount of traffic on a network. This tool does not measure throughput or response time. It supports several operating systems, including Linux, Free BS and Microsoft Windows. It provides a

Graphical User Interface (GUI) to generate TCP/UDP traffic from client to server to test firewalls/routers.

MGEN (Retrieved from: <http://cs.itd.nrl.navy.mil/work/mgen/index.php>) is a tool to generate real-time traffic patterns by providing emulation of unicast and/or multicast UDP and TCP/IP traffic. The traffic generated by MGEN can be used for network analysis and evaluation. It provides administrators the ability to carry out IP network performance tests and measurements using TCP and UDP/IP traffic. The data obtained from MGEN log can also be used to calculate performance statistics on throughput, packet loss rates and communication delay. It runs on Unix-based and Win32 platforms.

Packgen (Retrieved from: <http://packgen.rubyforge.org/files/README.html>) is a network packet generator with given bandwidth that supports both UDP and TCP flows. This utility sends packets at different time interval based on the bandwidth and the size of packets to generate. Packgen can generate various flows of data described by their names, destination, bandwidth, packet size and time range.

PacGen (Retrieved from: <http://pacgen.sourceforge.net/>) is an Ethernet IP TCP/UDP packet generating tool for Linux operating system. This tool provides custom packets with different configurable layers (Ethernet, IP, TCP and UDP) together with custom payloads.

Mtools (Retrieved from: <http://www.grid.unina.it/grid/mtools/>) contains a collection of tools used to generate UDP traffic in Linux and FreeBSD operating systems. It uses One-way-delay Meter to send UDP packets to a host in the network and measure their transmission time and use Round-trip-time Meter at the receiver side to send these packets back to the sender.

RUDE and CRUDE (Retrieved from: <http://rude.sourceforge.net/>) is a traffic generator running on Linux, Solaris and FreeBSD. RUDE (Real-time UDP Data Emitter) is a small program that generates traffic to the network. CRUDE (Collection for RUDE) is another program located on the other side of the network. CRUDE is used to collect the traffic generated from RUDE. These programs can generate and measure only UDP traffic. Both RUDE and CRUDE utilize the kernel to achieve the maximum performance.

KUTE (Retrieved from: <http://caia.swin.edu.au/genius/tools/kute/>) is another traffic generator developed in the kernel that can generate UDP packets. It consists of two Linux kernel modules: a sender and a receiver. A sender module sends packets directly to the hardware driver and can be used to test network devices such as routers or switches. A receiver module receives the packets from the driver, count the number of packets and measure the packet inter-arrival times. It is used together with the sender module for various performance testing.

All of the aforementioned traffic generators were designed for Linux operating system. However, these

traffic generating tools still have certain limitations. Therefore, a traffic generator is implemented in this project in order to overcome these problems. The fundamental idea of implementing a new traffic generator arises from the lack of the available ones. For example, some traffic generators like PackETH, Traffic or Rude and Crude have a GUI which occupies a lot of memory storage. Meanwhile, the traffic generator running on Soekris does not need a GUI, which can reserve more space in the memory and save execution time. In order to be installed in the Soekris board, the program to generate traffic must be light enough, yet at the same time has the basic function of a traffic generating tool.

In addition, most of the existing traffic generators are user-based programs. By running as user-space applications, overheads are created in the process of generating traffic which causes problems in network performance. User space traffic generation uses socket Application Programming Interface (API) available for a given system. It is more popular, simpler and easier to debug. However, in terms of speed and timing, kernel space traffic generators appear to be more desirable since the overall performance can be improved in terms of sending rates. Therefore, the traffic generator implemented in this project is developed from the Linux kernel in order to bypass the overhead of a user-application, allowing it to achieve the best performance and utilize maximum network bandwidth.

Programming in the kernel is more complicated than in the user space. As a result, only a few traffic generators are built for the kernel such as Rude and Crude and KUTE. Nevertheless, Rude and Crude and KUTE require server-client configuration, i.e., they must be installed at both ends of a tested link. It is redundant in this particular project because there is no need for capturing the generated data. Besides, KUTE is not well maintained and incompatible with later Linux kernel. It works only with Linux 2.6 and was tested up to kernel 2.6.16 (Victor, 2012).

In short, while there are many different types of traffic generators available, only a few of them can provide the necessary features for testing. To serve the testing purposes of this project, a traffic generator must not have a GUI, run on kernel space and does not require sever-client configuration. So far none of the existing tools can provide the exact combination of these desired features. Consequently, a new traffic generator is implemented in this study in order to meet the specific objectives and requirements. The purpose of putting the traffic generator at these locations is to generate one-way packet traffic from source to destination node in a network and carry out stress tests to analyze the network behavior under heavy load scenarios. The results obtain from these tests help network administrators to evaluate the performance of a communication network, identify the locations where possible problems might occur and implement a plan to secure these problems if they happen.

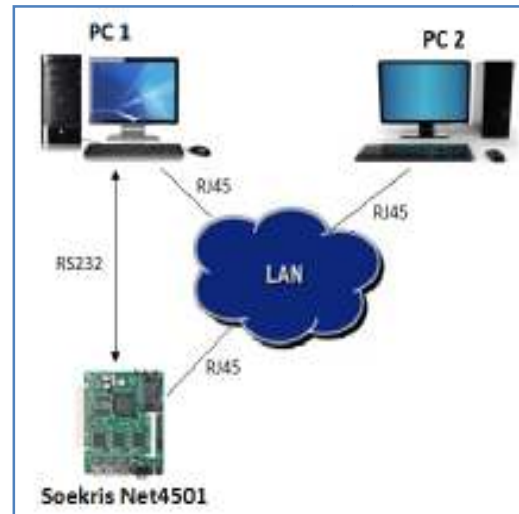


Fig. 2: System architecture

Development environment: The development environment was set up using two computers and a Soekris board (Net4501) as described in Fig. 2, where PC1 is connected to Soekris 4501 using serial cable (RS232). All devices are connected to LAN using crossover cables (RJ45) and TCP/IP protocol (IPv4). PC2 comes with Intel (R) Core (TM) i5-2400 with 3.10 GHz CPU and 8 GB of RAM. The operating system is Ubuntu 11.04 and kernel 2.6.38-8. The Soekris board belongs to Net4501 series which has Single Chip AMD ElanSC520 with an integrated core chipset. The main memory is 64 Mbytes PC 100 SDRAM soldered on board. The board also has a 1 GB CF card for program and data storage. Soekris Net4501 runs on Debian 4.0 (Etch) as an operating system with kernel 2.6.18.

Ubuntu 9.04 and Ubuntu 11.04 are chosen as operating systems for PC1 and PC2, respectively. Minicom (Fig. 3), a text-based serial port communication program, is installed in PC1 as a way to communicate with Soekris board through RS232 cable. Wireshark, a well-known network protocol analyzer, is installed in PC2 to capture the data packets generated from Soekris Net4501 (Fig. 4). This single board computer is running on Debian Etch (4.0) with the operating system stored in a 1GB compact flash card. The Linux operating system running on Soekris board does not require a GUI. All commands are executed from the terminal in a text form. Therefore, Minicom is a software tool used to get access to the single board computer. After the board is powered up, it will take a few minutes to boot up. Users can view the booting process from the Minicom terminal.

Traffic generator implementation: Each traffic generator is a single board computer which can be accessed remotely from the control center Using Secure Shell (SSH). After logging into the single board

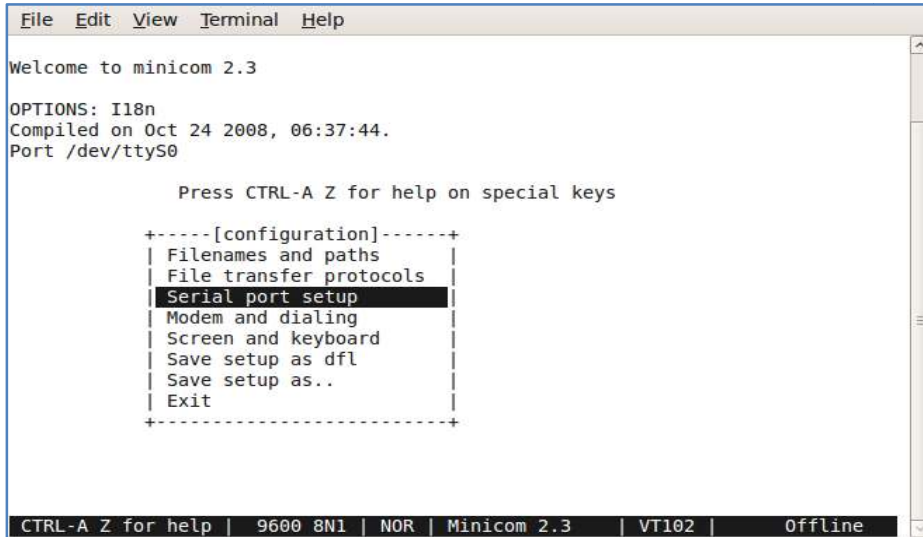


Fig. 3: Minicom

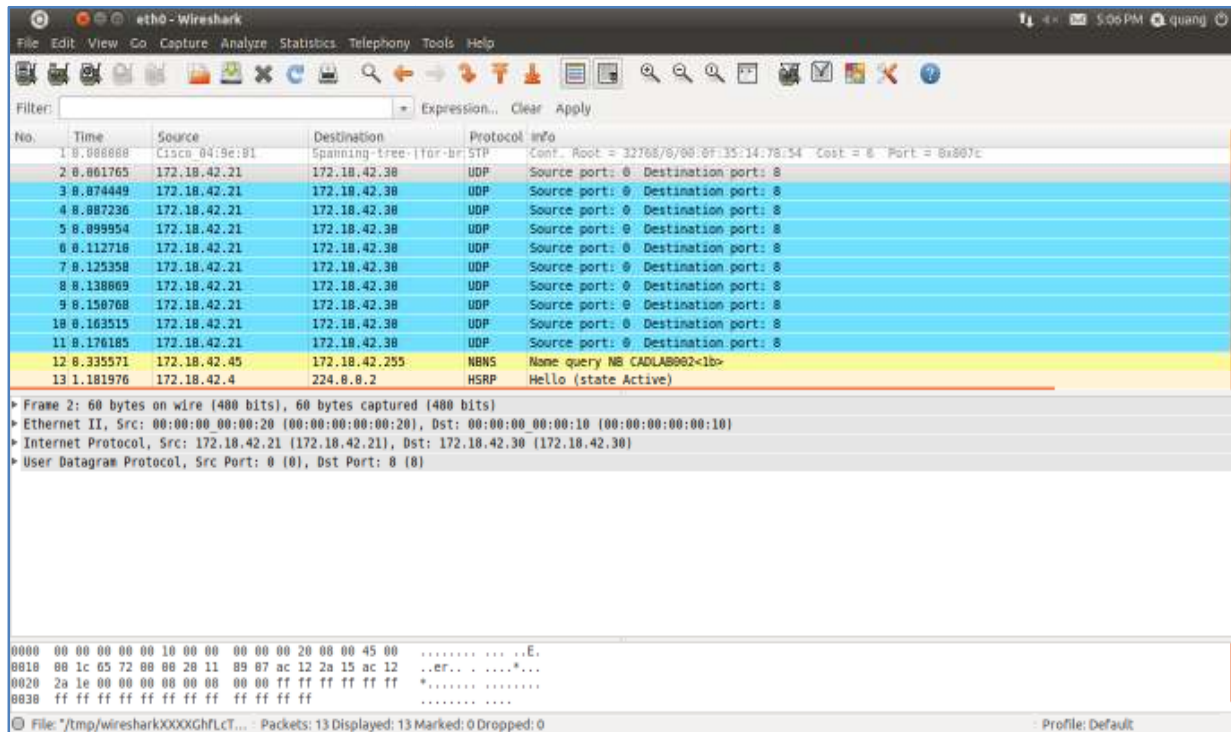


Fig. 4: Wire shark

computer using a valid username and password, network administrators can generate traffic by executing a set of commands from the terminal or console. A countdown script is written to start the traffic generator at a specific time as shown in Fig. 5.

Figure 6 shows the process of generating packets using a loadable kernel module. First of all, a module is built, compiled and loaded in the kernel space, creating an empty file system. Then, another file is created in the user space to generate a constant number of packets.

This number is then outputted to a file. After it is written to the empty file created in the kernel, the module reads the value and generates data packets based on this number. Finally, the file system is removed and the module is unloaded by the kernel.

The traffic generator is used to generate UDP traffic. UDP is a network protocol used to transfer data across an IP-based network. Unlike TCP, UDP does not perform handshaking or check errors. This protocol allows individual packets to drop without resending,

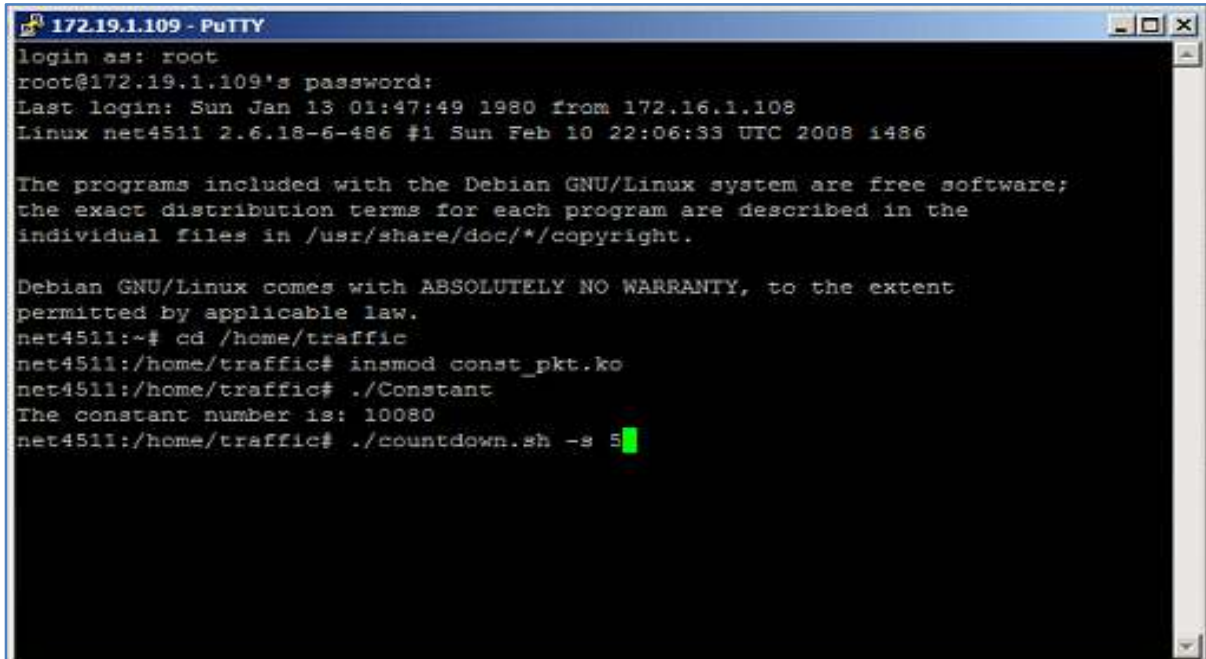


Fig. 5: Generate traffic from the command line

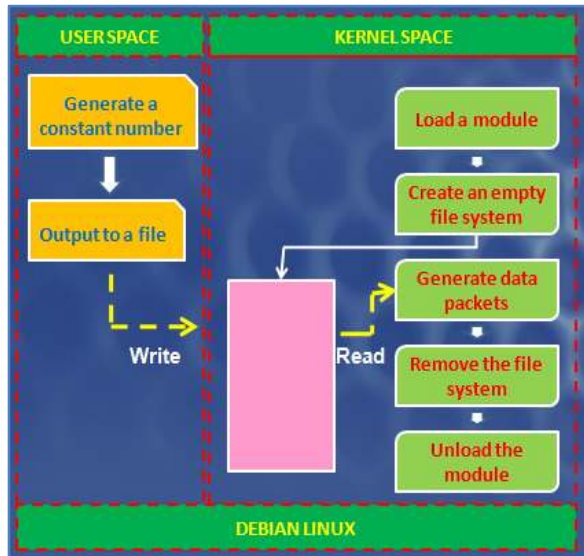


Fig. 6: Generate packets using loadable kernel module

does not check if the transmitted packets were received and even does not guarantee the right order of the received packets. Hence, UDP is referred to as unreliable and connectionless protocol. Nevertheless, since UDP skips several procedures and only focuses on transmission, it has lower overhead and thus faster than TCP or other network protocols. Due to this fast mechanism, UDP is chosen as a network protocol for traffic generator in smart grid communication network. In the context of smart grid, the function of a traffic generator is to carry out stress tests. In order to achieve better throughput and maximum traffic possible, UDP is

suitable for performing stress tests in smart grid, where error checking is not necessary in the application. The main purpose of the stress tests is to generate traffic at the potential bottlenecks in the network and test outside the boundary of what the network can handle, observe the way the network behaves and record the observation for future analysis.

Stress test: At several substations in the smart grid testbed, a large number of devices such as PLC modem, RTU, extender, connector, data concentrator and smart meters are connected to the same network switch. As a result, Soekris traffic generators are placed at these locations in order to investigate the network performance under bottleneck conditions. A series of stress tests are performed by putting several traffic generators at different substations within smart grid testbed. In particular, six identical traffic generators are implemented and then inserted into the network as shown in Fig. 7. These single-board computers are placed at six different substations where network bottlenecks might occur.

Figure 8 displays the flow chart for conducting stress test in smart grid testbed. First of all, a traffic generator is implemented in order to inject traffic into the communication network. To generate traffic, users need to log into the traffic generator remotely using Telnet. Network administrators can choose any number of data packets to be generated and this value is created in the user space. Network traffic is generated from the Linux kernel space to achieve the maximum performance. Packet loss, latency and throughput on various links of different parts on the communication

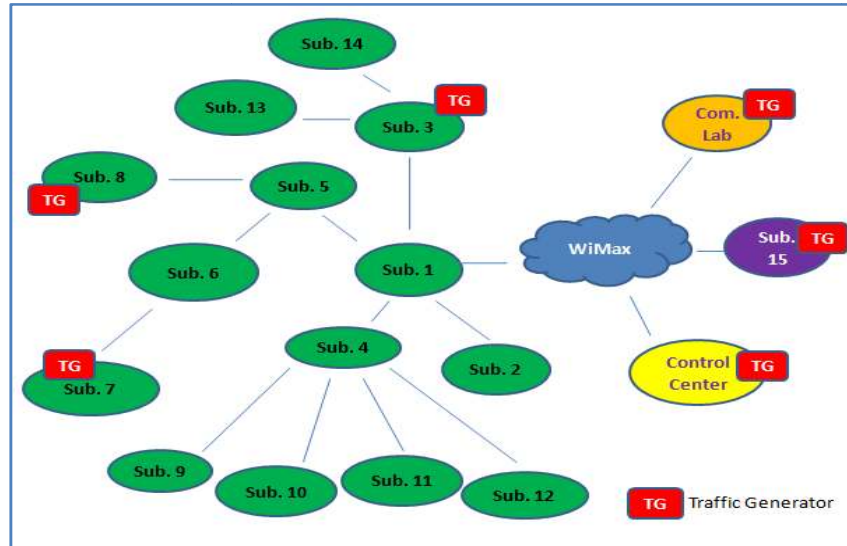


Fig. 7: Stress test for smart grid testbed

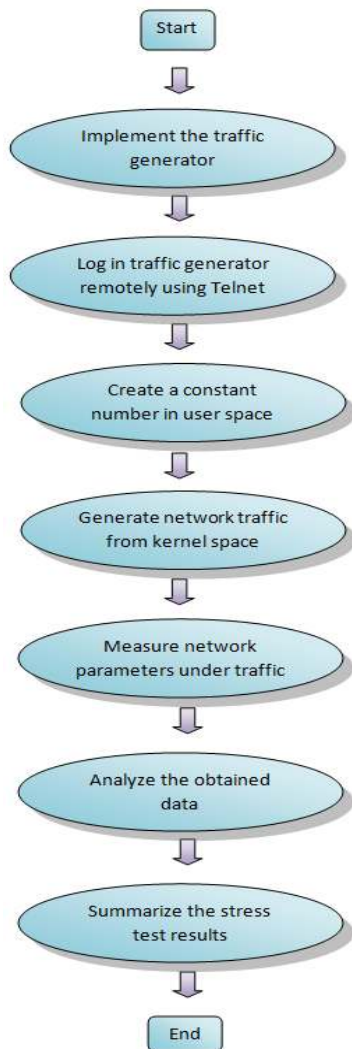


Fig. 8: Flow chart for carrying out stress test

network are measured under traffic condition. The obtained data is analyzed and the stress test results are summarized determine the capacity and limitation of smart grid communication network.

RESULTS AND DISCUSSION

Once the loadable kernel modules have been stored in the compact flash card, a fixed IP address is assigned to the Soekris board and the traffic generator has been implemented successfully. These single board computers are then placed at several substations in the smart grid testbed. After powering up the Soekris board and connecting the board's Ethernet port to the substation's switch using RJ45 cable, the board will boot up from the compact flash card by itself. The board can be accessed remotely from a computer at the base station of smart grid testbed by using Secure Shell (SSH). After logging into the Soekris board using SSH, network administrators can execute commands at the terminal to generate traffic. Firstly, a successfully-built module is loaded into the kernel is shown in Fig. 9.

The amount of traffic depends on the number of data packets generated and the time interval between packets. When the time interval is constant, the number of packets can be changed by modifying a program from the user space. The functions of this program are to avoid altering the kernel module code and to save compilation time. Every time the program is called, a constant number is generated and written into the /proc file system in the kernel space as displayed in Fig. 10.

Besides a program to generate a constant number of data packets, another script is also written in Linux user space to synchronize the time for traffic generation (Fig. 11). This script countdowns to a specific time to


```
File Edit View Terminal Help
-
net4511:/home/traffic# make
make -C /lib/modules/2.6.18-6-486/build SUBDIRS=/home/traffic
make[1]: Entering directory `/usr/src/linux-headers-2.6.18-6-486'
make[1]: Warning: File `/usr/src/linux-headers-2.6.18-6-486/arch/i386/Makefile.e
make[2]: Warning: File `scripts/Makefile.lib' has modification time 8.4e+08 s ie
LD /home/traffic/built-in.o
CC [M] /home/traffic/const_pkt.o
/home/traffic/const_pkt.c: In function 'procfile read':
/home/traffic/const_pkt.c:113: warning: ISO C90 forbids mixed declarations and e
/home/traffic/const_pkt.c:126: warning: ISO C90 forbids mixed declarations and e
make[2]: warning: Clock skew detected. Your build may be incomplete.
Building modules, stage 2.
make[2]: Warning: File `scripts/Makefile.lib' has modification time 8.4e+08 s ie
MODPOST
CC /home/traffic/const_pkt.mod.o
LD [M] /home/traffic/const_pkt.ko
make[2]: warning: Clock skew detected. Your build may be incomplete.
make[1]: warning: Clock skew detected. Your build may be incomplete.
make[1]: Leaving directory `/usr/src/linux-headers-2.6.18-6-486'
net4511:/home/traffic# insmod const_pkt.ko
/proc/bufferlk created
net4511:/home/traffic#
```

Fig. 9: Load a kernel module

```
File Edit View Terminal Help
net4511:/home/traffic# make
make -C /lib/modules/2.6.18-6-486/build SUBDIRS=/home/traffic
make[1]: Entering directory `/usr/src/linux-headers-2.6.18-6-486'
make[1]: Warning: File `/usr/src/linux-headers-2.6.18-6-486/arch/i386/Makefile.e
make[2]: Warning: File `scripts/Makefile.lib' has modification time 8.4e+08 s ie
LD /home/traffic/built-in.o
CC [M] /home/traffic/const_pkt.o
/home/traffic/const_pkt.c: In function 'procfile read':
/home/traffic/const_pkt.c:113: warning: ISO C90 forbids mixed declarations and e
/home/traffic/const_pkt.c:126: warning: ISO C90 forbids mixed declarations and e
make[2]: warning: Clock skew detected. Your build may be incomplete.
Building modules, stage 2.
make[2]: Warning: File `scripts/Makefile.lib' has modification time 8.4e+08 s ie
MODPOST
CC /home/traffic/const_pkt.mod.o
LD [M] /home/traffic/const_pkt.ko
make[2]: warning: Clock skew detected. Your build may be incomplete.
make[1]: warning: Clock skew detected. Your build may be incomplete.
make[1]: Leaving directory `/usr/src/linux-headers-2.6.18-6-486'
net4511:/home/traffic# insmod const_pkt.ko
/proc/bufferlk created
net4511:/home/traffic# ./Constant
The constant number is: 10000
net4511:/home/traffic#
```

Fig. 10: Generate a constant number

```
File Edit View Terminal Help
net4511:/home/traffic# make
make -C /lib/modules/2.6.18-6-486/build SUBDIRS=/home/traffic
make[1]: Entering directory `/usr/src/linux-headers-2.6.18-6-486'
make[1]: Warning: File `/usr/src/linux-headers-2.6.18-6-486/arch/i386/Makefile.e
make[2]: Warning: File `scripts/Makefile.lib' has modification time 8.4e+08 s ie
LD /home/traffic/built-in.o
CC [M] /home/traffic/const_pkt.o
/home/traffic/const_pkt.c: In function 'procfile read':
/home/traffic/const_pkt.c:113: warning: ISO C90 forbids mixed declarations and e
/home/traffic/const_pkt.c:126: warning: ISO C90 forbids mixed declarations and e
make[2]: warning: Clock skew detected. Your build may be incomplete.
Building modules, stage 2.
make[2]: Warning: File `scripts/Makefile.lib' has modification time 8.4e+08 s ie
MODPOST
CC /home/traffic/const_pkt.mod.o
LD [M] /home/traffic/const_pkt.ko
make[2]: warning: Clock skew detected. Your build may be incomplete.
make[1]: warning: Clock skew detected. Your build may be incomplete.
make[1]: Leaving directory `/usr/src/linux-headers-2.6.18-6-486'
net4511:/home/traffic# insmod const_pkt.ko
/proc/bufferlk created
net4511:/home/traffic# ./Constant
The constant number is: 10000
net4511:/home/traffic# ./countdown.sh -s 5
```

Fig. 11: Run a countdown script

```

File Edit View Terminal Help
wed Jan 23 22:07:17 MALT 1980
-----
Seconds: 0
Minutes: 0
Hours: 0
Days: 0
Weeks: 0
procfile_read (/proc/buffer1k) called
eth0: increased tx threshold, txcfg 0xd0f01004.
eth0: increased tx threshold, txcfg 0xd0f01000.
eth0: increased tx threshold, txcfg 0xd0f01000.
10000 packets
Goodbye
procfile_read (/proc/buffer1k) called
10000procfile_read (/proc/buffer1k) called

net4511:/home/traffic#
    
```

Fig. 12: Generate traffic after 5 sec

```

File Edit View Terminal Help
wed Jan 23 22:07:17 MALT 1980
-----
Seconds: 0
Minutes: 0
Hours: 0
Days: 0
Weeks: 0
procfile_read (/proc/buffer1k) called
eth0: increased tx threshold, txcfg 0xd0f01004.
eth0: increased tx threshold, txcfg 0xd0f01000.
eth0: increased tx threshold, txcfg 0xd0f01000.
10000 packets
Goodbye
procfile_read (/proc/buffer1k) called
10000procfile_read (/proc/buffer1k) called

net4511:/home/traffic# rmmod const_pkt.ko
/proc/buffer1k removed
net4511:/home/traffic#
    
```

Fig. 13: Unload a kernel module

read the/proc file system and hence generates traffic after a predefined period of time. The function of this script is to wait for 5 sec before generating the data (Fig. 12).

After the traffic generation process is complete, a loadable kernel module needs to be removed from the kernel as shown in Fig. 13.

The traffic generators are used to test different networks, including WiMax, Mesh RF and Power Line Communication. The procedure to carry out stress test for different technologies involves two steps: generating traffic and performing ping test. Since the kernel module used to generate traffic is configurable, users can change the number of data packets, packet size and delay between packets in order to generate desirable network traffic.

When being placed at various substations within the mesh RF network, each single board computer is able to generate Ethernet II packets with the maximum traffic of 28 Mbps. The delay between each packet is set to zero and the packet size is 1518 bytes (maximum Ethernet frame size). In WiMax and PLC networks, the delay between packets is increased and hence, the traffic duration also increases. Once the traffic is generated, packet loss and latency are measured to analyze and evaluate the network performance under stressed conditions.

The obtained results of stress tests are summarized in Table 2. At 28 Mbps, traffic does not have significant impact on the network if the duration of traffic flow is less than 450 ms. A significant increase is shown in communication latency when the traffic duration is

Table 2: Stress test

Technology	Traffic	Duration	Delay (ms)	Packet loss (%)
WiMax	370 Kbps	30 s	1142.40	5.5
Mesh RF	28 Mbps	<450 ms	0.70-23.50	0
		450 ms-1s	22.56-42.07	0
		>1 s	-	100
PLC	12.6 Mbps	6 s	83.78	7.0

within the range from 450 ms to 1s. With the traffic burst lasting more than 1s, the network experiences 100% of packet loss. At 370 Kbps and 12.6 Mbps, the generated traffic causes significant increase of delay in WiMax and PLC networks. The packet loss is also outside of the acceptable range (more than 5%).

The results obtained from the stress test show that the maximum capacity of traffic in the communication network occupies 50% of the rated bandwidth. The maximum traffic burst allowable is 27 Mbps for 400 ms. This ensures critical traffic such as SCADA control messages can be issued without interruption. The results also indicate that the current network capacity is able to support around 25,000 m and 200 RTUs.

CONCLUSION

This study introduces a single-board-computer-based Soekris Net4501 as a traffic generator to investigate the traffic arising in smart grid communication network. The traffic generator uses loadable kernel modules to create UDP packets according to the network traffic that might occur in smart grid communication system. The basic idea of implementing a new traffic generating tool is to represent the real traffic in multi-technology and bi-directional communication network in smart grid. This traffic generator can also be used to carry out stress tests in order to evaluate the performance of the existing communication system. As for future study, the function of the traffic generator can be enhanced by using other series of Soekris board, such as Net 6501 which has higher specifications, or updating Linux kernel to a later version. The 1 GB CF card can also be replaced by other cards that have larger storage capability to generate higher traffic volume. In addition, the traffic generator can be improved by generating packets that support not only UDP protocol but also Ethernet, TCP, ICMP and so on. The purpose of this traffic generation is to help administrator learn about the nature of network and study its behavior, which can be used to analyze and evaluate the performance of heterogeneous and hybrid smart grid communication network.

ACKNOWLEDGMENT

I would like to thank all the lecturers and students of Universiti Tenaga Nasional who have been involve directly and indirectly in this research. I would also like to extend gratitude to TNBR engineers and technicians

for their cooperation in all owing this research to take place.

REFERENCES

- Gang, L., D. Debraj and S. Wen-Zhan, 2010. Smart grid lab: A laboratory-based smart grid test bed. Proceedings of the 1st IEEE International Conference on Smart Grid Communication, Gaithersburg, pp: 143-148.
- Julien, A., J. Fletcher, D. Anand and C. Vasseur, 2010. An IEEE 1588 time synchronization testbed for assessing power distribution requirements. Proceedings of the International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS), Portsmouth, pp: 13-18.
- Nadim, A.H., 2010. Smart Grids for Dummies. In: Logica, L. (Ed.), West Sussex. John Wiley and Sons Ltd.
- Paricio, G.A., J. Oliver and D. Gosch, 2010. An intelligent agent-based distributed architecture for smart-grid integrated network management. Proceedings of the 1st IEEE Workshop on Smart Grid Networking Infrastructure, Denver, pp: 1013-1018.
- Tarek, K., K. Naik, M. Alsabaan and A. Nayak, 2010. Transport protocol for smart grid infrastructure. Proceedings of the 2nd International Conference on Ubiquitous and Future Networks (ICUFN), Jeju Island, pp: 320-325.
- Victor, C.V., 2012. Dynamic content generation for evaluation of network applications. Ph.D. Thesis, School of Electrical Engineering and Computer Science, Washington State University, New York.
- Victor, O.K.L., F. W. Felix and Z. Jin, 2010. Communication requirements for risk-limiting dispatch in smart grid. Proceedings of IEEE International Conference on Communication Workshops (ICC) Capetown, pp: 1-5.
- Wenye, W., X. Yi and K. Mohit, 2011. Computer Networks: A Survey on the Communication Architectures in Smart Grid. Raleigh, Elsevier.
- Zhong, F., K. Georgios, E. Costas, S. Mahesh and S. Mutsumu, 2010. The new frontier of communications research: Smart grid and smart metering. Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking, Passau, pp: 115-118.
- Zhuo, L., L. Xiang, W. Wenye and W. Cliff, 2010. Review and evaluation of security threats on the communication networks in the smart grid. Proceedings of Military Communications Conference-Cyber Security and Network Management, San Jose, pp: 1830-1835.