International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:2, No:3, 2008

The Implementation of Remote Automation Execution Agent over ACL on QOS POLICY Based System

Hazly Amir, and Roime Puniran

***Abstract*—**This paper will present the implementation of QoS policy based system by utilizing rules on Access Control List (ACL) over Layer 3 (L3) switch. Also presented is the architecture on that implementation; the tools being used and the result were gathered. The system architecture has an ability to control ACL rules which are installed inside an external L3 switch. ACL rules used to instruct the way of access control being executed, in order to entertain all traffics through that particular switch. The main advantage of using this approach is that the single point of failure could be prevented when there are any changes on ACL rules inside L3 switches. Another advantage is that the agent could instruct ACL rules automatically straight away based on the changes occur on policy database without configuring them one by one. Other than that, when QoS policy based system was implemented in distributed environment, the monitoring process can be synchronized easily due to the automate process running by agent over external policy devices.

***Keywords*—**QOS, ACL, L3 Switch.

I. Introduction

IN order to provide reliable and sustained QOS in TCP/IP network, well defined network architecture and application installed inside the network should be accommodated correctly. Conventional implementation approaches on QoS policy system will alter the users by manipulating the operation using centralized policy management application. An effective approach to force network control is that QoS-based policy management must have knowledge of the capabilities of each of users attached into the network [1].

The centralized QoS-based policy system shows the lack of good management, control mechanism, coordinate and configuration parameters. It's refused to give us a consistent fashion result when these approaches are going to be deployed in wide-range network or distributed environment [3]. Hardware requirement as well as system maintenance and software used must be able to reach zero percent of down time. It's required for an organization to invest more resources to ensure the QoS policy based system being installed ad suits their needs includes the backup system employed in order to avoid any bad circumstance happen.

As shown before, the deployment of QoS system and how QoS is being defined by the network policies is left to the network administrator. Thus, the automation processes include the cited task between agent and L3 switch are derived from the network administrator point of view [2].

Other proposed solution was to define the policies and let an intelligent network devices implement them. The traffics will go through a respective routers and it will be filtered. The routers will decides which QoS policy for a particular traffic flow based on their priority, service type, resource usage etc [4]. In this case, we had proposed the implementation of QoS-based policy system over L3 switch. We implement the system over L3 switches rather than router regarding a couple of things. First, we need a device that reliable enough to serve large amount of traffic without forwarding that traffic into multiple route.

The reason is that the L3 switch could pass the traffic as fast as Layer-2 (L2) switch and yet the decision on how was the traffic being transmitted would be made just like a router [5]. Our network testbed employ the QoS policy based system behind the core router, which is the traffic are being prioritized before it going to be route to the respective subnet. Thus, it's more appropriate if the policy could be defined through ACL in L3 switch before it going to be routed externally.

The objective of this research is, to eliminate single point of failure in a centralize policy server. Besides that, this architecture could prevent or at least minimized the network burden when single centralized QoS policy server had to be a red dot bridging the internal network and core network. L3 switch installed inside the network will serve the real traffic and it can be considered that a switch was a most critical element in the network. To avoid a critical exception on that, we minimize the human–manual interfere during the policy configuration especially on ACL. Thus, agent was developed to execute remote automation process. This agent will emulate all human-input command over the L3 switch through the command instruction scripting. The overall script was constructed using Ckermit [12] programming language.

II. Related Literature

Several agent-based QoS policy systems were proposed and are widely implemented. S.S Manvi et al. [7] describe the mobile agent based QoS simulation model on multimedia communication environment. This work states that mobile agent paradigm significantly reduce bandwidth consumption and network traffic in contrast to client/server paradigm. Every node in the networked multimedia system must have an agent execution environment. Two type of agent are going to

be installed; static agent (recognized as a server agent) and mobile agent. Mobile agents are traveled on each of nodes where the server agents are installed inside and also negotiation were made in between with the service provider in order to prioritize the policy regarding bandwidth usage within the network. This implementation are work well in multimedia centralized communication network which is all nodes are sharing same mobile environment and multimedia content. Outside that, this proposal not addresses the most issues arise on 'best effort' QoS policy based network environment where, all users deserve to get certain level of bandwidth guarantees whatever services their used even in different platform or different application.

In Weiyi Li et al. [8] extended abstract, despite to archive differentiate QoS among network users, and there will be a requirement for the active configuration of QoS. These network services and resource configuration requires an enhanced control infrastructure as well as associated pricing mechanism as part of session establishment. The issues arise regarding the latency due to the message propagation from one host to another. Especially today's broadband networks when much more time required in transmitting a large packet. Therefore, this works introduce signaling mechanism with mobile agent. Agent-based signaling is used in negotiations of cost and resources. A mobile agent can be constructing to perform multiple tasks. The mobile agent may be traveled widely to collect information and the same agent can issue the appropriate management control on it. The problem is that this mobile environment and all the element are installed in centralized platform which is the traffic are also goes through the same machine. These not prevent the network burden and bottle neck on physical specification issues.

III. Our Implementation Architecture

Our proposed system was derived from several implementation issues in the field. The packets are being probed through Packet Capture Terminal (PACAT) script in IPCON Box. All probing packet basically formatted as a raw data and being polled and processed into a readable data before they were dumped into a system database.

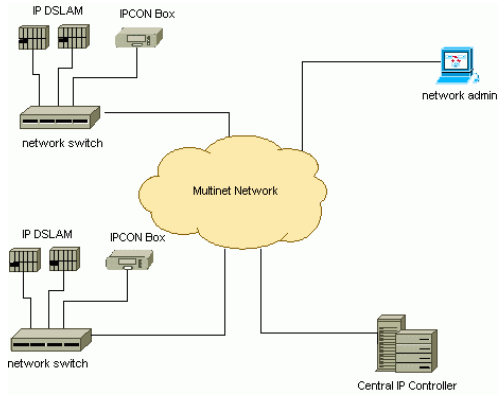

Fig. 1 IPCON QoS policy system architecture

PACAT Box is a device which is installed after the ISP core network. It can be considered as a passive box which is, no critical execution implement inside. We choose to use L3 switch instead of single centralized server to avoid different subnet being created on each of an interface and to prevent packet flow through multiple hop.

We used Multinet Network, a private ISP network environment in our system implementation. Initially, our mediation system will collect basic data from IP DSLAM devices located at scattered exchanges using IPCON Box. These data will be translated into valuable information. Regarding on Fig. 1, IPCON Controller, a program being installed inside Central IP Controller, will analyzed that data and from there, they need to make a decision whether to execute policy rules based on data, send a notification to the network administrator or waiting for any further instruction from an administrator.

IPCON Agent, a policy agent, are programmed and installed in each of every IPCON Box. They share the same platform with PACAT script. L3 switch were installed as a 'bridge' in between the core network and a billing system. L3 switches used to be a main traffic highway of internal packet flow. Thus the configurations for sure are regularly made.

IPCON Agent was created in order to emulate human-manual command line input. The simple rationale behind that is that if there were several switches are installed distributed, the automation process (e.g ACL instruction) can be synchronized and the risk of human-error could be decreased.

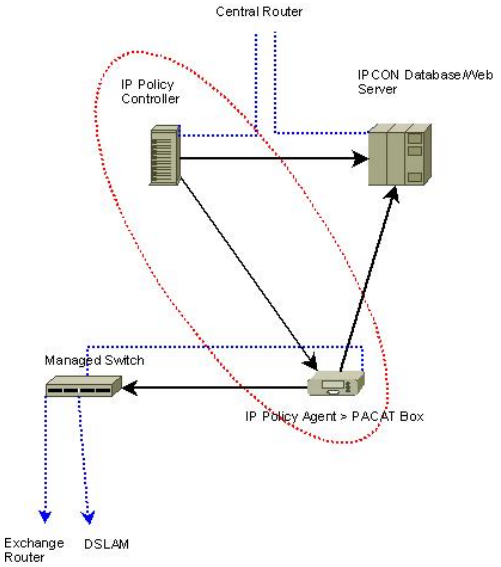

Fig. 2 IPCON QoS policy system development area

Regarding Fig. 2, our QoS policy based system involves 2 main devices; 1 – IP Policy Agent (IPCON Agent) in PACAT Box 2 – IP Policy Controller (IPCON Controller). Agent that is installed inside IP Policy Agent box is responsible to interact directly to L3 switches regarding instruction came form IPCON Controller.

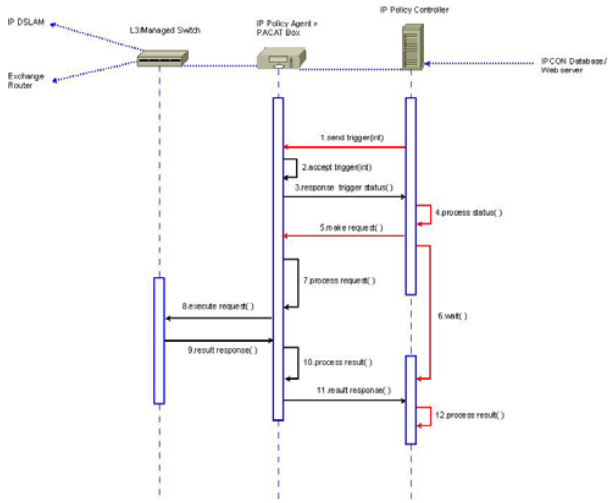

Fig. 3 Sequence Diagrams of Policy agent and Policy Controller

Fig. 3 illustrate the flow process between IPCON Agent and IPCON Controller. The system starts when IPCON Controller triggered the process. Triggering the process basically initiated when there are any changes made within the databases. It could be happen if there is some users changing their packages or admin need to do something over some user's services (block, warning or changing the packages). IPCON agent starts their works when socket on RMI on their services accepted the incoming request from the IPCON Controller (step 2). Triggering status are responded whether the IPCON Agent ready to run the process or append it. IPCON Controller start initiated the process if IPCON Agent are ready (step 4) and then wait until IPCON Agent processes the request (step 7). The result (session and transaction log) of the process will be gathered at the end of the processes (step 11 and 12).

There is only 2 type of policy rules used in this work; block and unblock. The rules was set up on ACL inside the L3 switch and each rules indicate 12 ACL sequence rules which is, all of these must be emulate by IPCON Agent. Each users initially are be given the amount of bandwidth volume regarding the packages their purchase for a particular time. If the usage of volume exceeded the initial amount, the administrator has a capacity to block or pending these users based on their IP address until those particular users clean their status (by paying a bill or purchase the other package). Our concerned is, how was this will be going to be happen if there is a thousand of users in different area attaching different QoS policy server? Thus, the implementations of automation process to execute all of rules are being our main priority. These rules are installed in every L3 switch which is connected to DSLAM directly to CPE of the users. ACL rules inside the L3 switch will be used to implement our policy. Basically, an administrator will defined the rules need to be executed. This rule will be sent into IPCON Agent, and those agents are responsible to read that rules instruction and executed it automatically.

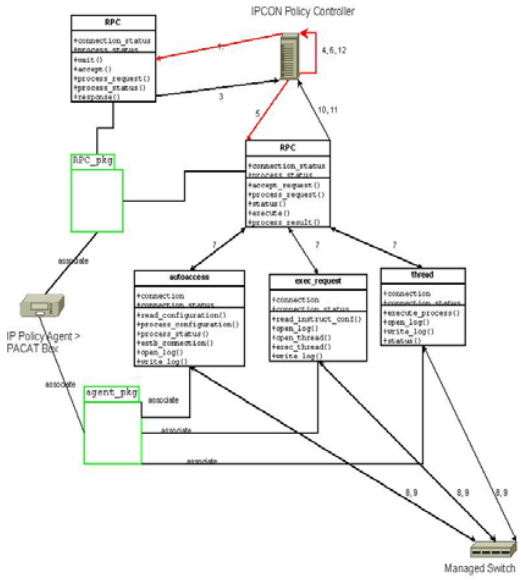

Fig. 4 Simple Class Diagrams of Policy agent and Policy controller

IPCON Agent that installed inside PACAT Box is generally stand-alone-script. From IPCON Agent point of view, its merely impossible if we let them alone since we want to trigger them when there are any changes occur on the database. Since IPCON Agent only works if there is an attention invoked from IPCON Controller, at least there is a function / mechanism used to accept the request. Thus, as mention through Fig. 4, it's possible if there is some RMI function implemented together in PACAT Box besides the IPCON Agent. This function will accept any incoming request from IPCON Controller. Besides accepting incoming request, RMI also responsible to execute automate script (CKermit). See step 7 on Fig. 4. After script finishing their process, these RMI will determine the status and respond the answer back to the IPCON Controller.

We manage not to use RMI to record the entire log (session and transaction log). IPCON Agent script can manage them after all.

IV. Experiment and Result

IPCON Agent was formed entirely in C-Kermit script. The architecture of C-Kermit IPCON Agent script was showed in Fig. 5.

There were several approaches we have been used to enable the connection between IPCON Agent and IPCON Controller. We had developed remote invocation application using Java RMI in order to apply remote execution between controller and agent.

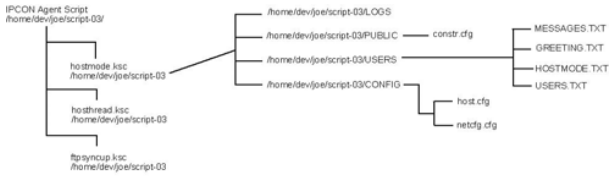

Fig. 5 Policy Agent Scripts Architecture

Physically, we used 3 types of machines which are can be categorized into 2 segments. Regarding an architecture illustrated on Fig. 3, we use NetGear-GSM-7312 series as our main L3 switch connected to the outside network and NexGate-NSA-1041 series as an agent platform running DSL 3.0 UNIX OS, CKermit library and Java RMI, as shown in Table I.

TABLE I
IMPLEMENTATION SETUP TOOLS

IP Policy Agent / PACAT Box	• NexGate-NSA-1041 • DSL 3.0 OS
L3 Switch	• NetGear-GSM-7312 • Standard IOS

Based on the research, CKermit (technically inherited from Kermit Protocol) is a computer file transfer/management protocol and a set of communications software tools [9] [10]. It provides a consistent approach to the file transfer, terminal emulation, script programming, and character set conversion across many different computer hardware and OS platforms.

The IPCON Agent main script (hostmode.ksc) was re-invented from original Kermit 95 hostmode.ksc [11]. Other than that, several threads were used such as hosthread.ksc and ftpsyncup.ksc to implements multiple inter-networking processes such as FTP, read-write-execute process, LOG and error handling.

Regarding Fig. 5, top level script consists of hostmode.ksc, hosthread.ksc and ftpsyncup.ksc. The initiator of the processes besides providing users main GUI was the hostmode.ksc. hosthread.ksc was constructed to implement the thread process between IPCON Agent and L3 switch, meanwhile ftpsyncup.ksc will execute FTP process in order to transfer LOG files from agent to controller. ftpsyncup.ksc is an optional. It's not required as a basic iteration process for the whole system.

Actually, users can choose to enable or disable the FTP process regarding the situation and needs.

In order to block or unblock users based on their IP address, there is sequences of command input have to be issued over L3 switch. Manual human-input scheme provide 12 rules in order the administrator complete the task.

```
joe=enable
joe=
joe=?
joe=
joe={show access-list interface 0/2 in }
joe=
joe=exit
joe=
joe=logout
joe=n
joe=
```

Fig. 6 Instruction script for show access list interface

To execute all 12 rules as an automation process, CKermit must issue the instruction file where all respective command lines are exposed to instruct the rules. Fig. 6 shows the example of the instruction file where it will indicate the command that is an administrator should be issued manually in a regular basis configuration. The example shows the access-list type on interface 0/2. It will indicate the current status on that interface.

All 12 rules will instruct the different command line interface. Regarding test case result, in order to block/unblock users based on their IP address, the process starts by enabling configuration mode on L3 switch. The most important rules need to be issued by an administrator was a permit/deny instruction which is being presents in test case number 5. Test case number 12 was the final stage used to show all access-list being configured in the system. Table II shows the sample of test case.

As an example, Fig. 7 will show the list of access-lists in the interface 0/2 when administrator manually telnet the L3 switch and issue a command line.

```
[root@127 omnetpp-3.3]# telnet 172.16.10.157
Trying 172.16.10.157...
Connected to 172.16.10.157.
Escape character is '^]'.

(GSM7312)
User:admin
Password:*******
(GSM7312) >
(GSM7312) >
(GSM7312) >
(GSM7312) >en
Password:

(GSM7312) #show access-lists interface 0/2 in

ACL Type             ACL ID               Sequence Number
-------- ------------------------------ ---------------
IP       102                            2
IP       100                            3
IP       110                            4

(GSM7312) #
```

Fig. 7 Human input command lines to show access-list interface

IPCON Agent emulates this same process when *hostmode.ksc* calling *hosthread.ksc* threads and read the execution instruction defined on automation script (as shown in Fig. 6) with the key attribute was *joe*. *For-loop* statement was used in order to enable loop process in every *joe* attribute. Instruction sequence will be register in memory buffer before *hostmode.ksc* issues CKermit *connect* method to send all of these instructions into L3 switch. L3 switch that accept these packet, starts to process them in sequence as a normal human manual input command. Fig. 8 shows the example of emulation process, issuing same input commands line issued in Fig. 7.

TABLE II
TEST CASE FOR AUTOMATE EXECUTION

Acronym:
S = Success
F = Fail

Test Case
TEST CASE 1: Accessing ACL (Access Control List)

Command	Purpose	Status(S/F)	Remarks	Rate (%)
enable config	Access ACL	S	NIL	100

..
..
..

TEST CASE 5: Permit/deny Rules

Command	Purpose	Status	Remarks	Rate (%)
Enable Config { mac access-list extended <name> permit ***or*** deny <srcmac> <dstmac> <ethertypekey> vlan <0-4095> cos <0-7> assign-queue <queue-id> redirect <interface> }	To permit / deny ACL	S	NIL	100

..
..
..

TEST CASE 12: Show Access Lists

Command	Purpose	Status	Remarks	Rate (%)
enable { show access-lists interface <slot/port> in }	To show Access List	S	NIL	100

Test case is created to define the result yields when the same command line was executed by using 2 different approaches; human-manual input and automation execution. System test conducted in order to achieves a number of objectives.

a. To satisfy the output between human-based input emulation and automate-based CLI instruction.
b. To satisfy the both result are comparable.

In a system test, we use command input as a main metrics to indicate the successful result. Comparable test between manual command-input and automate execution will be ran and the result will be rated. Besides that, LOG file will record the status of execution.

Fig. 8 Automation execution process to emulate show access-list interface

Last but not least, the LOG file will be created at the end of every session and transaction. Basically, there would be two type of LOG in our system; logs that indicate the transaction status and log that gathered the result of every session.

When the system starts running, all the process includes the execution of configuration file and access status will be recorded as a transaction log. The timestamp will be used to indicate the time the process was running. It used in order for administrator to debug any error which is occur. This transaction log only records the processes that are running and interacts internal. Fig. 9 shows the example of transaction log.

```
Transaction Log: C-Kermit 8.0.211, 10 Apr 2004
 Linux
Mon Dec  3 11:32:53 2007

11:32:53 - IPCON Agent started...
11:32:56 - STARTING...
11:32:56 - Read configuration file: /usr/local/IPCON/CONFIG/host.cfg
11:32:56 - OK. Config file found...
11:32:57 - Configuration file=/usr/local/IPCON/CONFIG/host.cfg
11:32:59 - Loading configuration ...
11:33:01 - READ NETWORK CONFIG FILE...
11:33:01 - OK. Network config file found...
11:33:03 - Configuration file = /usr/local/IPCON/CONFIG/netcfg.cfg
11:33:05 - Loading configuration ...
11:33:07 -  3 Dec 2007 - Start host script /usr/local/IPCON/hostmode
11:33:07 - Current directory: /root/
11:33:07 - Host:172.16.10.157
11:33:07 - login:admin
11:33:07 - Connecting to 172.16.10.157 as user admin
11:33:09 - OK...
11:33:11 - Password echo off....
11:33:11 - Take /usr/local/IPCON/hosthread.ksc
11:33:11 - READ THREAD.....
11:33:11 - OK. Config file found...
11:33:12 - Read=enable
11:33:13 - Read=
11:33:14 - Read=?
11:33:15 - Read=
11:33:16 - Read=show access-list interface 0/2 in
11:33:17 - Read=
11:33:18 - Read=exit
11:33:19 - Read=
11:33:20 - Read=logout
11:33:21 - Read=n
11:33:22 - Read=
11:33:23 - Command transfered ....
```

Fig. 9 Transaction log

When the IPCON Agent starts to issues an automation instruction over L3 switch, the system will use session log to indicate the status of that particular session. Session log only record the session made by an agent regarding an automation scripts.

The response command line from L3 switch and any changes being made following the instruction on automation script then will be recorded in that session log. The status of the session (success or fail) would be indicated at the top of the log file. Meanwhile, the remaining details will indicate every single response during the automation execution process made from agent over the L3 switch.

Fig. 10 show the session log details produced during process.

```
11:33:23 - Process Success !

(GSM7312)
User:admin
Password:*******
(GSM7312) >enable
Password:

(GSM7312) #?

arp                     Purge a dynamic or gateway ARP entry.
clear                   Reset configuration to factory defaults.
configure               Enter into global config mode.
copy                    Uploads or Downloads file.
disconnect              Close active remote session(s).
dot1x                   Configure dot1x privileged exec parameters.
exit                    To exit from the mode.
Ezconfig                Run the easy configuration.
help                    Display help for various special keys.
ip                      Configure IP parameters.
logout                  Exit this session. Any unsaved changes are lost.
network                 Configuration for inband connectivity.
ping                    Send ICMP echo packets to a specified IP address.
reload                  Reset the switch.
script                  Apply/Delete/List/Show/Validate Configuration Scripts.
set                     Set Router Parameters.
show                    Show configured data.
sshcon                  Configure SSH connection parameters.
telnet                  Telnet to a remote host.
telnetcon               Configure telnet connection parameters.
terminal                Set terminal line parameters.
traceroute              Trace route to destination.
--More-- or (q)uit
vlan                    Type 'vlan database' to enter into VLAN mode.

(GSM7312) #
(GSM7312) #show access-lists interface 0/2 in

ACL Type             ACL ID              Sequence Number
-------- ------------------------------ ---------------
IP       102                            2
IP       100                            3
IP       110                            4

(GSM7312) #
(GSM7312) #exit

(GSM7312) >
(GSM7312) >logout

The system has unsaved changes.
Would you like to save them now? (y/n) n
```

Fig. 10 Session log

V. Conclusion

The paper presents the implementation of remote automation execution agent, its comparable issues between the human manual inputs, also the system design and tools being used. Initially, the QoS policy based systems are sets for all users that used 'best-effort' network where each users interfaces are defined on each of different port. The users which are unable to fulfill the requirement of usage will be block. And currently, the system only promotes the block/unblock function as an indication of QoS policy implementation in this research. The result shows that an agent could execute almost all command lines on L3 switch and produce the result which is 100 percent accurate as human-manual input command line. In our implementation, there is only 12 compulsory L3 switch command line interface which is it will instruct a rules in a way to block/unblock user. By the way, the scalability of the system also depends on how was the capability and specification of the devices.

References

[1] M. Stevens, W. Weiss, H. Mahon, B. Moore, J. Stassnerr, G. Waters, A. Westerinen, Policy Framework, Internet Draft IETF (1999)
[2] Marcelo Borges Ribeiro, Lisandro Zambenedetti Granville, Maria Janilce Bosquiroli Almeida, Liane Margarida, An Architecture to Monitor QoS in a Policy-Based Network, XXI Simpósio Brasileiro de Redes de Computadores (2003)
[3] Silvano Gai, John Strassner, David Durham, Shai Herzog, Hugh Mahon, Francis Reichmeyer, QoS Policy Framework Architecture, Network Working Group IETF(1999)
[4] Haiming Huang, Andreas Mainssner, Wolfgang Schoenfeld, Ralf Steinmetz, QoS Policy Framework and Its Implementation, Communication Technology Proceedings, WCC - ICCT 2000. International Conference on (2000)
[5] http://computer.howstuffworks.com/lanswitch15.htm
[6] http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-2/switch_evolution.html
[7] http://computer.howstuffworks.com/lanswitch15.htm
[8] http://computer.howstuffworks.com/lanswitch16.htm
[9] http://www.spacedaily.com/news/iss-03zq.html
[10] http://www.columbia.edu/kermit/faq.html#licence
[11] http://ftp.nluug.nl/networking/kermit/k95/
[12] http://www.columbia.edu/kermit/

Hazly Amir is a BEE (Hons) graduate from UTM Malaysia majoring in Telecommunication (1999) and MEE (2003). His interest involved QoS, P2P control, MPLS, diffserv, traffic generation, digital modulation and Software Radio based baseband processing.

Roime Puniran graduated from FSCIT UNIMAS Malaysia, received his Degree in Computer Science in 2004. Currently he is attached to Network Services & Management Cluster as a researcher which is given the responsibility in the development of multi platform networking & programming.