BACK-END APPLICATION FOR MONITORING MESH NETWORK

By

Ajayi Olabode O.

Project Supervisor:
Prof Bill Tucker

Project Co-Supervisor:
Mr. MJ. Norman

Project Advisor:
Mr. Carlos Rey-Moreno
Mr A. Kruger

INTRODUCTION

Project Goals:

- To reduce high rate of data log on Mesh Potatoes per day, weekly, or even monthly and keep the CPU utilization consumed to a minimum
- To provide monitoring and information that can be used to either predict future failures or to diagnose network problems (e.g. Mankosi Community in the Eastern Cape for rural mesh networks)

REQUIREMENTS

What Is Expected From The Software Solution?

- To manage and monitor mesh network activity in an effective way
- To help the network manager to save time expended on data collection

What Is Not Expected From This Software?

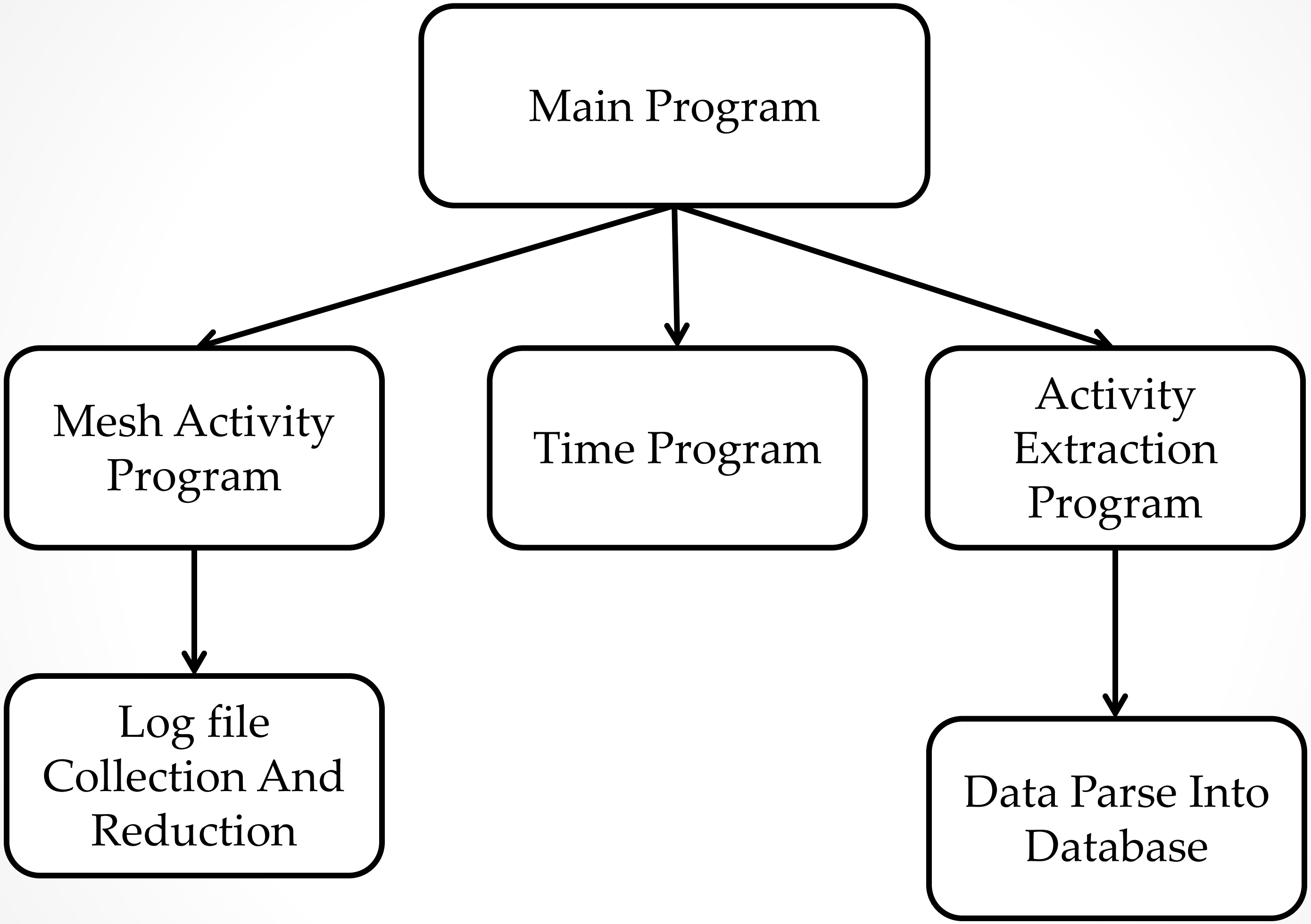
- Not to override existing system functionality
- Should not disrupt the network activity

PROPOSED SOLUTIONS

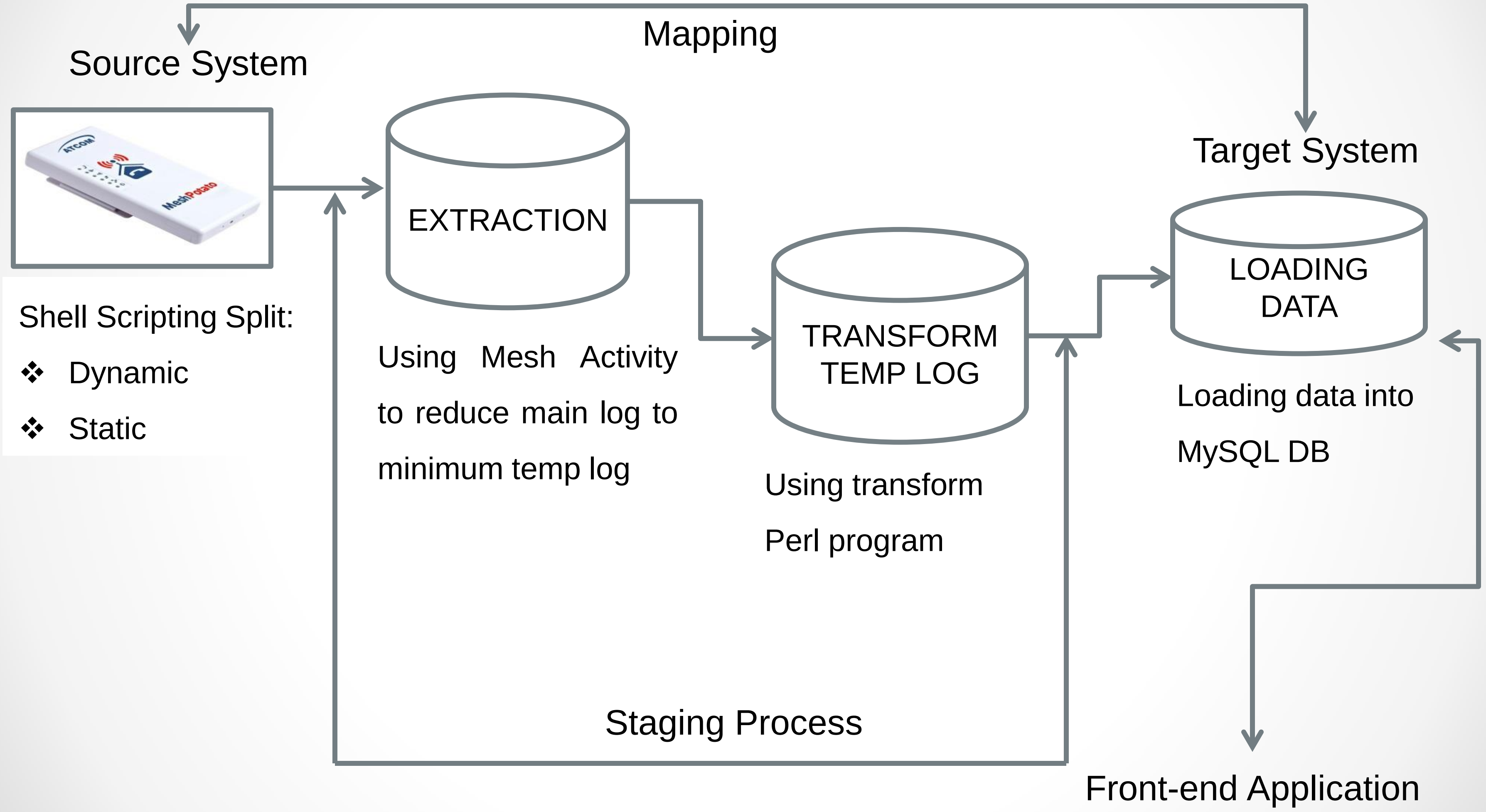
The requirements employed the

- ❖ Concept of data mining using Extracting, Transforming, and Loading (ETL) for this project database management system.
- ❖ Use of Small Campus Enterprise Network firmware
- ❖ Easy data communication and better quality of service (wireless links).

HIGH LEVEL VIEW OF THE SYSTEM



DESIGN & IMPLEMENTATION



Mapping

Source System

Shell Scripting Split:

- ❖ Dynamic
- ❖ Static

EXTRACTION

Using Mesh Activity to reduce main log to minimum temp log

TRANSFORM TEMP LOG

Using transform Perl program

Target System

LOADING DATA

Loading data into MySQL DB

Staging Process

Front-end Application

TESTING STRATEGIES

- Performance testing (of script)
- Unit testing
- Security testing (on the database)
- Recovery testing

PERFORMANCE TESTING

We conducted performance testing to track resources over time and know exactly what we are getting at. We further:

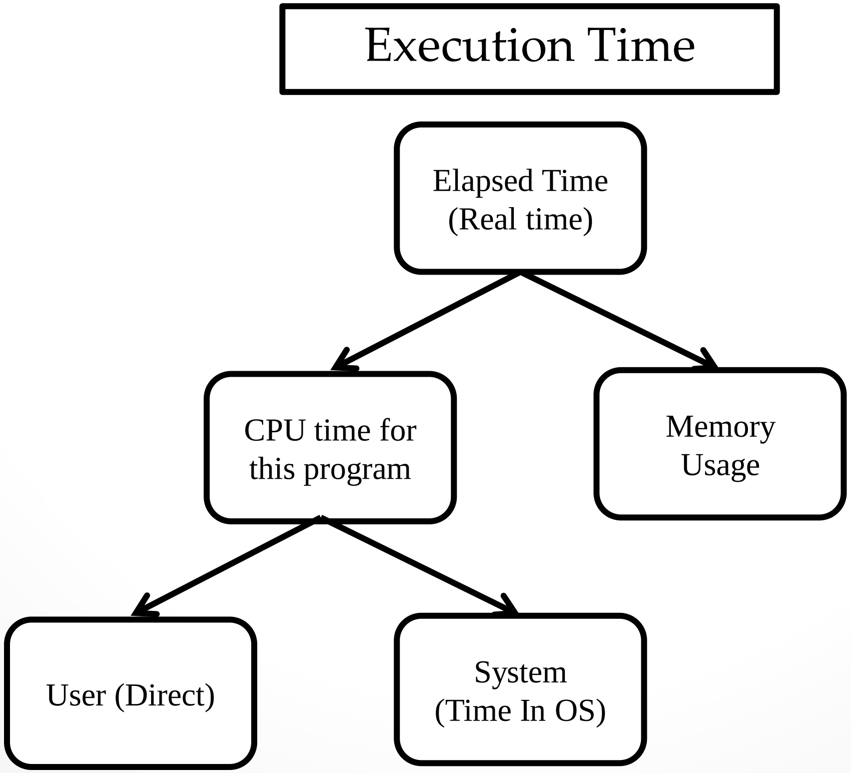
- ❖ Compare the application timing to the OS timing.
- ❖ Test the performance and speed by finding the performance for our application on a particular machine.

A SNAPSHOT OF PERFORMANCE TESTING

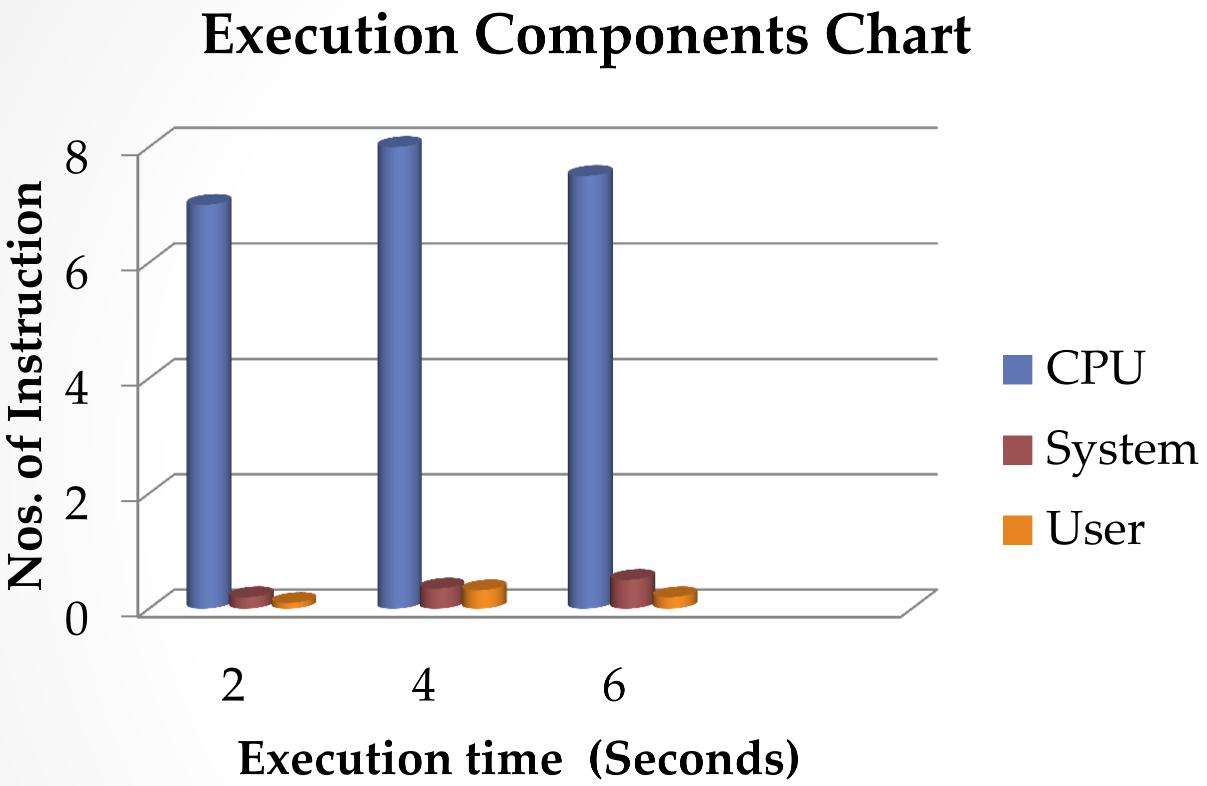
Execution time can be divided into two parts.

❖ User time is spent running the application program itself.

❖ System time is when the application calls operating system



PERFORMANCE TESTING (COMPARISON)

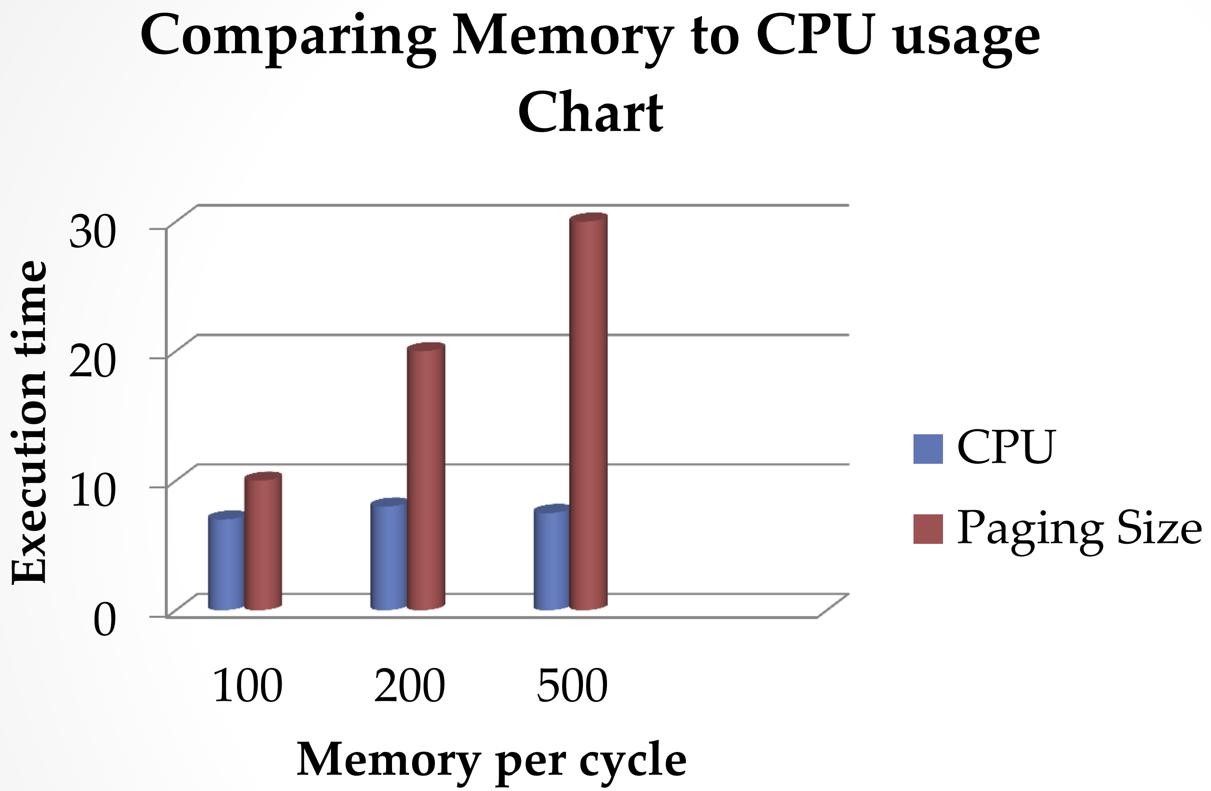


Notes:
Cumulative Sum (System time) = 0.51 sec
Average per Instructions = 0.0085 sec

Cumulative Sum (User time) = 4.5 sec
Average per Instructions = 0.75 sec

Execution time = CPI * cycle time * Nos of Instruction
Note:
CPI = CPU per Instructions

PERFORMANCE TESTING (COMPARISON)



Total Maximum of CPU time = 5570 clock cycles
CPU Average per Instructions = 557 sec
Average Execution Time = 92.83 sec

Maximum Paging Size Usage = 332772 (bytes)
Average per size = 5546.2 bytes

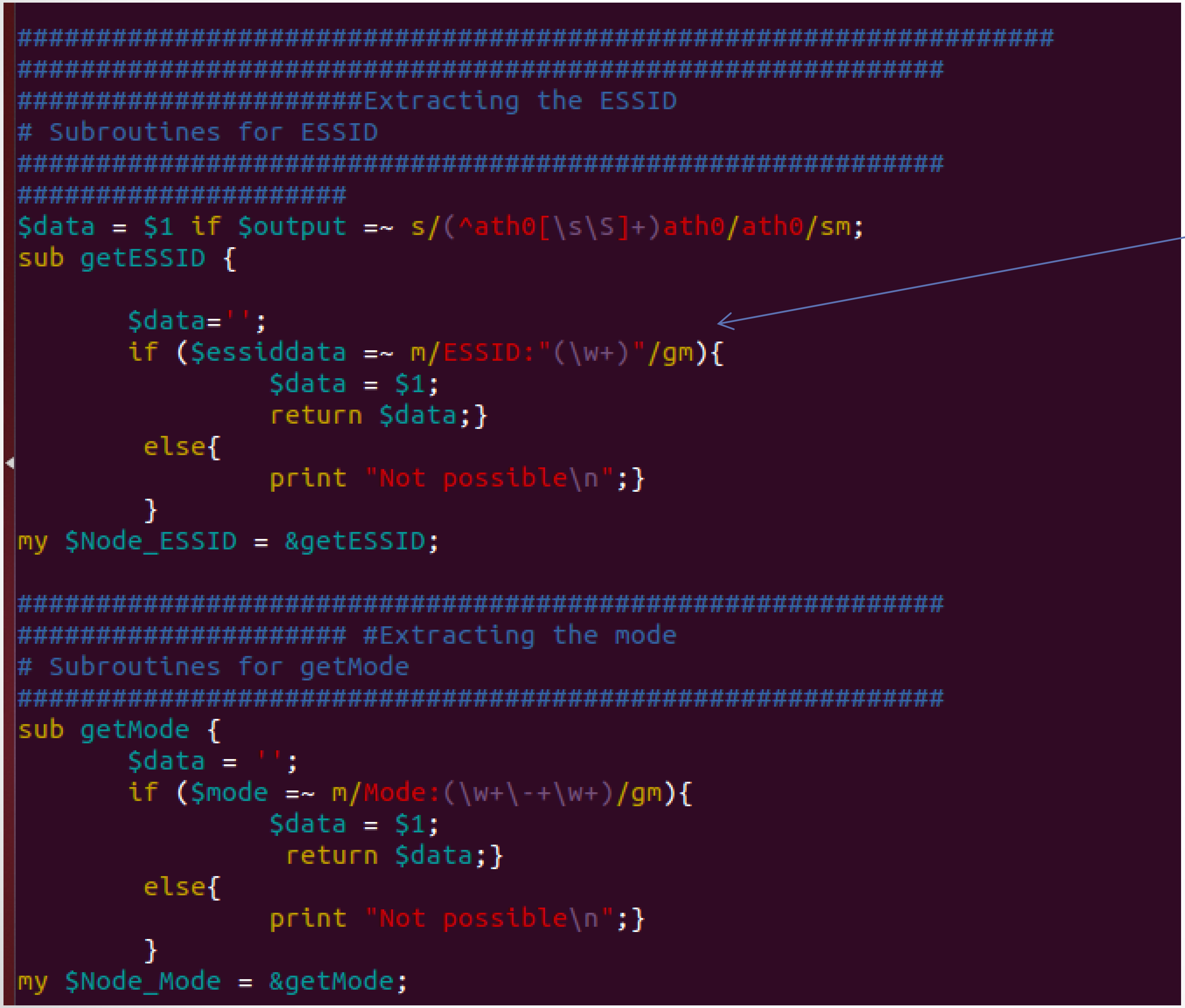
CPU Usage = (User + System)/ Total Nos. of Instr.

UNIT TESTING

We conducted unit testing on the internal processing logic of the program. The following are highlighted:

- ❖ Functions : We ensure information flow properly into and out of the program unit under test.
- ❖ All program including errors handling paths are checked and tested.

Unit Testing View



```
##################################################################
###########################################################
######################Extracting the ESSID
# Subroutines for ESSID
###########################################################
#####################
$data = $1 if $output =~ s/(^ath0[\s\S]+)ath0/ath0/sm;
sub getESSID {

       $data='';
       if ($essiddata =~ m/ESSID:"(\w+)"/gm){
                $data = $1;
                return $data;}
        else{
                print "Not possible\n";}
        }
my $Node_ESSID = &getESSID;

###########################################################
##################### #Extracting the mode
# Subroutines for getMode
###########################################################
sub getMode {
       $data = '';
       if ($mode =~ m/Mode:(\w+\-+\w+)/gm){
                $data = $1;
                 return $data;}
        else{
                print "Not possible\n";}
        }
my $Node_Mode = &getMode;
```

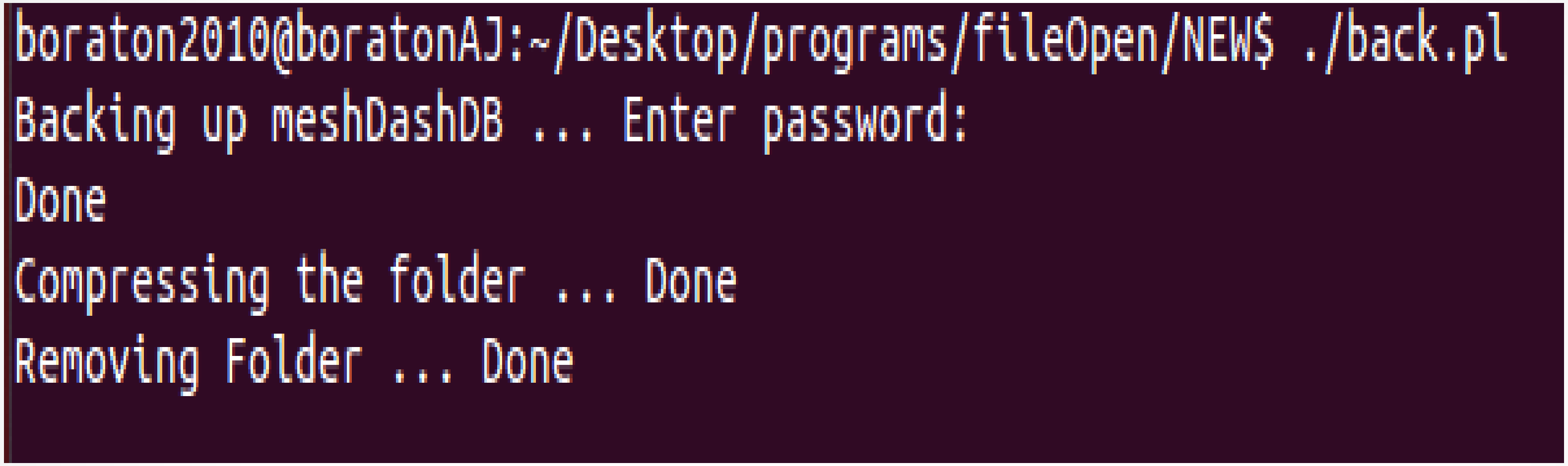
Function checked

SECURITY TESTING

We conducted security testing to protect against vulnerabilities. We implemented two out security elements. The two included are:

- ❖ Authentication : Credential verification between clients and servers
- ❖ Authorization : By supplying user ID and password

Security Testing View

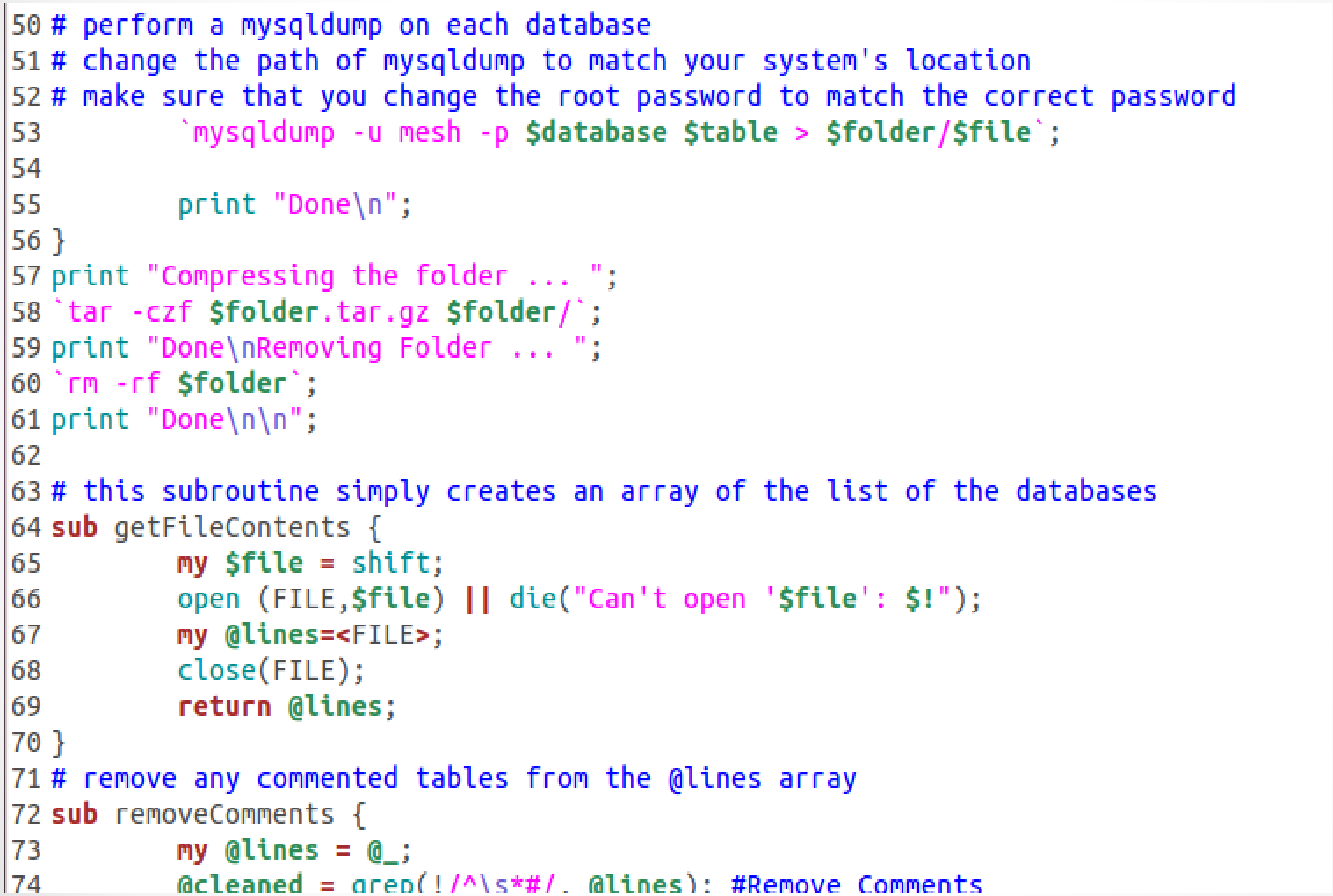


```
boraton2010@boratonAJ:~/Desktop/programs/fileOpen/NEW$ ./back.pl
Backing up meshDashDB ... Enter password:
Done
Compressing the folder ... Done
Removing Folder ... Done
```

RECOVERY TESTING

We conducted recovery testing by creating a backup of the entire database. This is done automatic using a cron job for effective correctness and better system optimization.

Backup Code View



```
# perform a mysqldump on each database
# change the path of mysqldump to match your system's location
# make sure that you change the root password to match the correct password
        `mysqldump -u mesh -p $database $table > $folder/$file`;

        print "Done\n";
}
print "Compressing the folder ... ";
`tar -czf $folder.tar.gz $folder/`;
print "Done\nRemoving Folder ... ";
`rm -rf $folder`;
print "Done\n\n";

# this subroutine simply creates an array of the list of the databases
sub getFileContents {
        my $file = shift;
        open (FILE,$file) || die("Can't open '$file': $!");
        my @lines=<FILE>;
        close(FILE);
        return @lines;
}
# remove any commented tables from the @lines array
sub removeComments {
        my @lines = @_;
        @cleaned = grep(!/^\s*#/, @lines); #Remove Comments
```

TESTING ISSUES AND RESULTS

Our testing strategy reviewed some issues which is most applicable to any network scripting application. These include:

- ❖ Mesh Potatoes software versions
- ❖ Precision inaccuracy
- ❖ Mixed mode operation
- ❖ Incorrect arithmetic precedence and initialization

REFERENCES

[1.] S. Surana, R. Patra, and E. Brewer, *Simplifying Faulty Diagnosis in locally Managed Rural WiFi Networks, University of California,* Beckery, USA.

[2.] L. Subramanian, S. Surana, R. Patra, M. Ho, A. Sheth, and E. Brewer. Rethinking Wirelesss for the Developing World. *Hotnets-V*, 2006.

[3.] K. Chebrolu, B. Raman, and S. Sen. Long-Distance 802.11b Links: *Performance Measurements and Experience*. In ACM MOBICOM, 2006.

DEMO

The following outlines I will be demonstrating:

- Live Crontab job demonstration on Mesh Potatoes: Collection of log file on Mesh Potatoes
- SCP to perform Extracting, Transforming and loading into database (An Integrated Program and testing the performance with timing)
- Backing up the system for data recovery
- Aging the log file from the Mesh Potatoes

QUESTIONS AND ANSWERS

Thank you