

Alert Monitoring Application

CSC 665 Computer Networks

Fall – 2018

12-06-2018

Abstract:

In the present world, the need for e-commerce has become a key in our everyday life. As we can see, the number of websites each day keeps increasing, out of which many websites are time-sensitive. The need for connectivity is crucial in e-commerce sites especially websites that deal with actions like constant currency transactions and sensex information. For example, if amazon drops its connectivity chain for 30 seconds, think about number of users that would be impacted. Hence the need for monitoring has become an important necessity. While a host or server is being monitored when there is an issue in connectivity a user action is required for which alerting is needed. The idea of this project is to create a web based alert monitoring application interface which constantly checks the connectivity of a website host or IP. When there is a drop in connection, the application will trigger an email to alert the user. In this project, open source operating systems/functions/applications/programs, LINUX, shell scripting, SMTP, ping, telnet and traceroute were selected due to being fast and compatible for monitoring and alerting purposes.

Report by:

Vignesh Sivanandha Rao

Instructor:

Dr. Ajay Katangur

Missouri State University

Introduction:

Present-day technology has seen more advancements in fast few decades than couple of centuries earlier. The networks and connectivity used in current technology is complex that troubleshooting them requires real-time monitoring and thorough understanding about data interpretation. Monitoring is the term whose meaning lies within the name, it refers to knowing the state of a system. Monitoring helps organizations to nail down complications even before they could give rise to incidents, breakage or down-times. There are 2 types of monitoring, *Proactive Monitoring* and *Reactive Monitoring*. Proactive Monitoring is where the systems are monitored for their storage space (and increase the swap necessarily) restart applications for performance etc. On the other hand, reactive monitoring refers to taking actions on incidents and highlight the areas of concern. This project involves both reactive and proactive monitoring where the application is monitored and the action can be taken only when a connection drop-out is noticed however on doing this continuously, the reaction time can be reduced and with few drop-outs actions can be performed even before the link/host is completely down. The main task of monitoring is to detect faults and help the users in eliminating them. The other major advantage of monitoring is *Alerting*. The notification about the monitored failure to the user is known as alerting. Alerting can be done through sending an email, flash message etc. In organizations, these alerts are then logged in the form of tickets and actions are taken appropriately. This report will briefly explain the operating system, applications, functionality and programs used in this project and their working in the contents below.

Contents:

Operating System – Linux (Ubuntu)

Programming Language – Shell Scripting

Protocol – Simple Mail Transfer Protocol (SMTP)

Operations performed – Ping, Telnet and Traceroute

Functionalities:

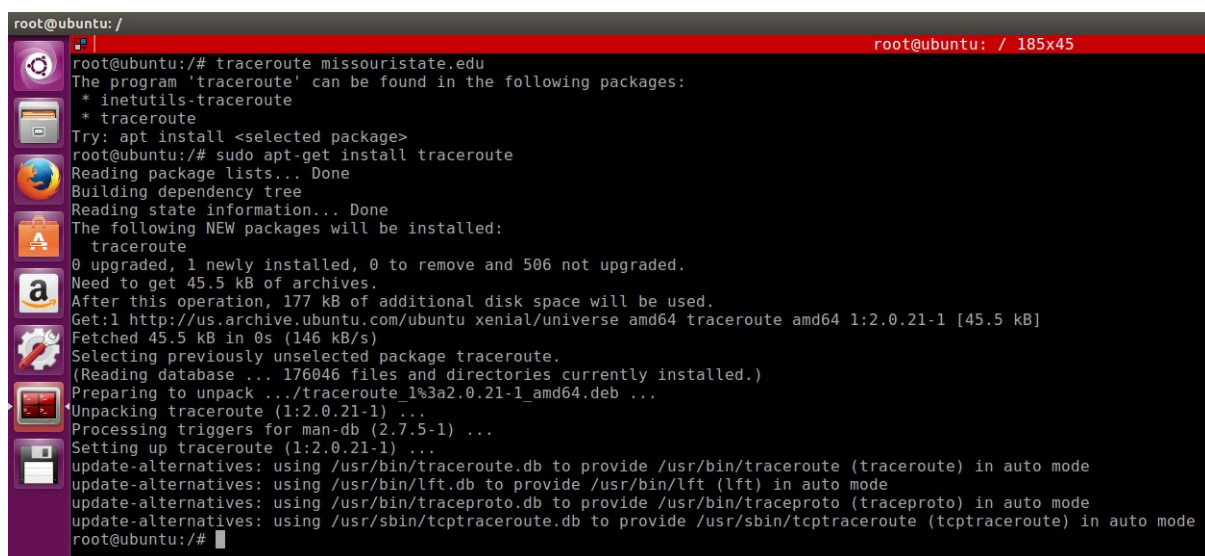
The ultimate motive of this project is to check the connectivity of a website and I've done three major operations to check the connectivity. Ping, Telnet and Traceroute.

Internet ping works similar to echo-location. It sends a small packet of information containing ICMP ECHO_REQUEST to the specified host which then sends an ECHO_REPLY packet in return. The two major information we can draw from a ping is *Access* and *Time*. We can use ping to see if we are able to reach another host or server. With ping we can come to know if we have access to the particular host or server. We can also see the time taken for the packet of information to be returned to our machine using ping. From the ping information we can get to know which sites have the faster network access and would be more efficient. Ping has various options to specify the count of ping requests to be made by using “ping -c”, to specify the size of the packet to be sent as request by using “ping -s”. Ping generally resolves the IP of a specified hostname, however we can use “ping -a” to resolve the hostname by pinging an IP.

Telnet is the other function I have used in my project to check the open ports. Ports are like doorways for the internet traffic to pass through. With the required port being closed, the network traffic will not be able to find its way through, just like how you get stuck in a room if the door is locked. While websites use http and https for internet surfing it is mandatory for both the ports to be open while a request is sent or received. To check this we use the command “telnet <hostname> <portno>”.

Traceroute is a command which can display the path taken by the packet information sent from our computer. It will list down all the routers it passes through until it reaches the destination. It can also tell you how long each hop from one router to the other has taken. Traceroutes are extremely useful when we are trying to figure out issues like “*Destination*

Net Unreachable” as it prints the result of where the connection has failed. The traceroute I used in this project sends 60 byte packets and the traceroute will attempt to reach the destination within 30 hops, each hop tested 3 times. Whenever there is no response from a particular router, it will try another time. To setup a working traceroute I required package called internet utilities which I setup using (inetutils-traceroute) function. To download and install this package I used the command “*sudo apt-get install traceroute*”. Once one I verified the working by using the command “*traceroute <hostname>*”.



```

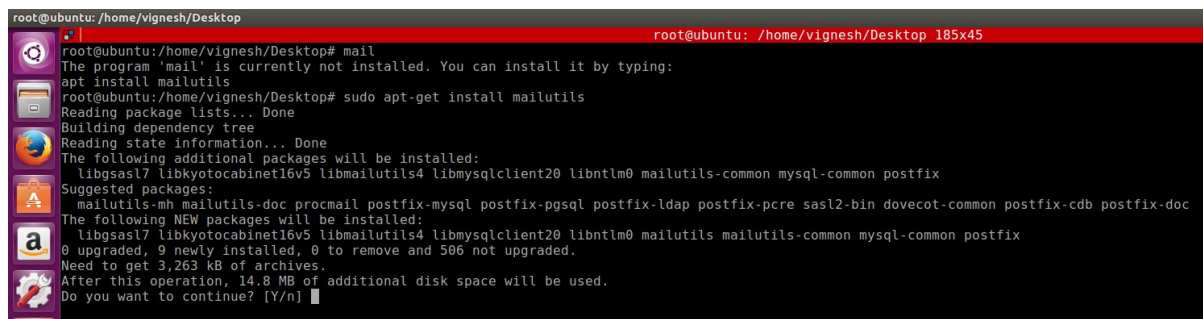
root@ubuntu: /
root@ubuntu: / 185x45
root@ubuntu: /# traceroute missouristate.edu
The program 'traceroute' can be found in the following packages:
* inetutils-traceroute
* traceroute
Try: apt install <selected package>
root@ubuntu: /# sudo apt-get install traceroute
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  traceroute
0 upgraded, 1 newly installed, 0 to remove and 506 not upgraded.
Need to get 45.5 kB of archives.
After this operation, 177 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/universe amd64 traceroute amd64 1:2.0.21-1 [45.5 kB]
Fetched 45.5 kB in 0s (146 kB/s)
Selecting previously unselected package traceroute.
(Reading database ... 176046 files and directories currently installed.)
Preparing to unpack .../traceroute_1%3a2.0.21-1_amd64.deb ...
Unpacking traceroute (1:2.0.21-1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up traceroute (1:2.0.21-1) ...
update-alternatives: using /usr/bin/traceroute.db to provide /usr/bin/traceroute (traceroute) in auto mode
update-alternatives: using /usr/bin/lft.db to provide /usr/bin/lft (lft) in auto mode
update-alternatives: using /usr/bin/traceproto.db to provide /usr/bin/traceproto (traceproto) in auto mode
update-alternatives: using /usr/sbin/tcptraceroute.db to provide /usr/sbin/tcptraceroute (tcptraceroute) in auto mode
root@ubuntu: /#

```

Fig [1] Installation of Traceroute function in LINUX

SMTP (Simple Mail Transfer Protocol) is an email server that is responsible for sending emails and receiving emails. SMTP is part of the application layer of the TCP/IP protocol. SMTP uses “store and forward” process where it moves emails on and across networks. SMTP provides a set of codes that simplify the communication of email messages between servers. In SMTP, when a message is sent out, the server breaks it up into chunks of information which the receiving server can understand. Sometimes these chunks of information is passed through number of computers and they also maintain their MTAs (Mail Transfer Agent) and Remote senders. Note: SMTP can only handle txt and cannot transfer images, attachments or other graphical contents. To set up an SMTP mail client I’ve used

mail utilities function, which can be set up using the command “*sudo apt-get install mailutils*”. Once this is run the mail utilities agent will prompt to select the mail server configuration type in the Postfix Configuration. According to my needs I selected “*Internet Site*” in which the mails will be sent and received directly using SMTP. Once the mail server configuration type is selected, the postfix configuration will prompt to choose a system mail name. After the mailbox configuration is done I used the command “*mail <recipient>*” to check if the mail server is working fine. Below are the screenshots explaining the installation and configuration of mail utilities aka SMTP configuration.



```

root@ubuntu: /home/vignesh/Desktop
root@ubuntu: /home/vignesh/Desktop# mail
The program 'mail' is currently not installed. You can install it by typing:
apt install mailutils
root@ubuntu: /home/vignesh/Desktop# sudo apt-get install mailutils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libgsasl7 libkyotocabinet16v5 libmailutils4 libmysqlclient20 libntlm0 mailutils-common mysql-common postfix
Suggested packages:
  mailutils-mh mailutils-doc procmail postfix-mysql postfix-pgsql postfix-ldap postfix-pcre sasl2-bin dovecot-common postfix-cdb postfix-doc
The following NEW packages will be installed:
  libgsasl7 libkyotocabinet16v5 libmailutils4 libmysqlclient20 libntlm0 mailutils mailutils-common mysql-common postfix
0 upgraded, 9 newly installed, 0 to remove and 506 not upgraded.
Need to get 3,263 kB of archives.
After this operation, 14.8 MB of additional disk space will be used.
Do you want to continue? [Y/n]

```

Fig [2] Installation of mail utilities function in LINUX

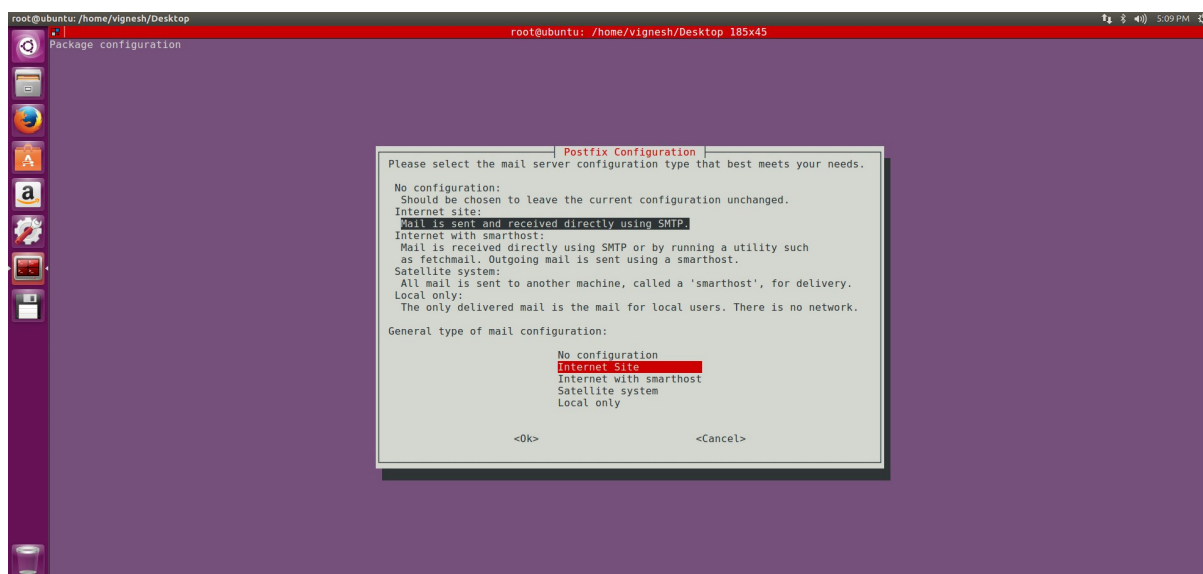


Fig [3] Choosing mail server configuration type SMTP

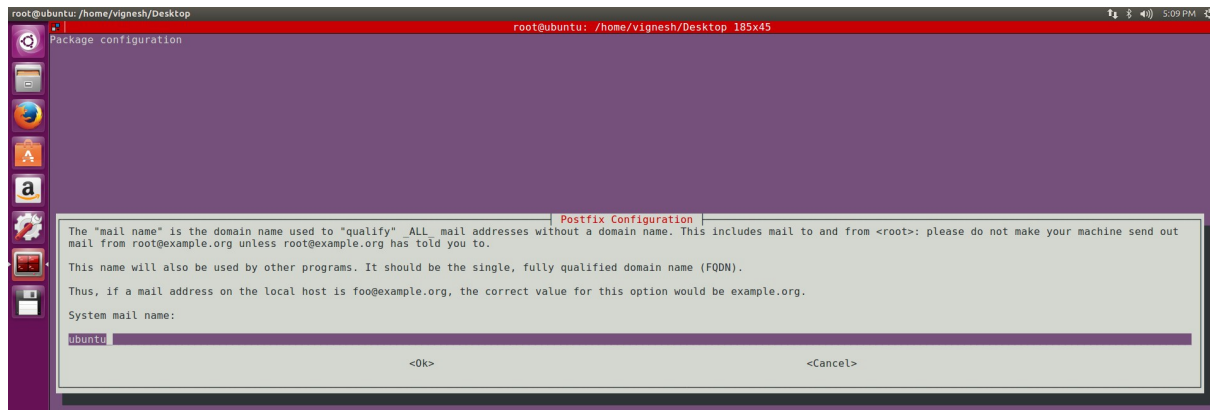


Fig [4] Choosing the mail name

Using the above components, I created a bash script which first starts with ping and once ping has been successful it moves on telnet through ports 80 and 443. Once the telnet has completed successfully the script then traceroutes the website. These information is written to a log file and is stored in a specified location with the timestamp in the file name. Finally, a loop checks for any drops in the log file, if found it sends out an email to the user alerting for an action required from the user end and also logs them in the destined location. The above steps can be performed in a script file which can run only once if executed once however to check a website connection 24x7, I required a job which can run the script in until the process is killed. To perform this, I used crontab.

Cron is a linux function which is like Task Scheduler in windows. In cron we can schedule a command or a script on our server to run automatically at the specified time and date. Cron job doesn't need a kick off. It can automatically start when the time is read from the server. Generally cron jobs can have three components in them. The *date and time info*, *command/script* and the *output*. I used five fields for specifying day, date and time followed by the command/script to be executed at the interval specified. As I wish to run my monitoring script every minute and 24x7, I have specified “* * * * *” in front of the script which has been explained in detail in the Fig [5] below:

*	*	*	*	*	command to be executed
-	-	-	-	-	
				+-----	day of week (0 - 6) (Sunday=0)
			+-----		month (1 - 12)
		+-----			day of month (1 - 31)
	+-----				hour (0 - 23)
+-----					min (0 - 59)

Fig [5]: Structure of a Cron entry

```

root@ubuntu: /home/vignesh/Desktop/project 92x45
root@ubuntu:/home/vignesh/Desktop/project# crontab -l
##Cron job to make the below script run every minute 24x7
## * * * * * /home/vignesh/Desktop/project/monitor.sh

##Cron job to purge old logs
## * * * * * /home/vignesh/Desktop/project/purge.sh

* * * * * /home/vignesh/Desktop/project/hostmonitor.sh
root@ubuntu:/home/vignesh/Desktop/project#

```

Fig [6]: Representation of my crontab entry using command crontab -l (1)

With all the setup I did earlier, memory management might become an issue once the job keeps running and starts acquiring more space. To eradicate this, I have created another script to purge old log files. Since immediate actions would have been performed on connection drop-outs, we may not need log files older than a certain period. To show immediate results I have written a script to purge files older than 30 minutes. With the cycle of both the jobs running, we can always maintain our log folder clean and make it have the log files of the last 30 minutes only. Fig [7] below shows the script file for purging old logs.

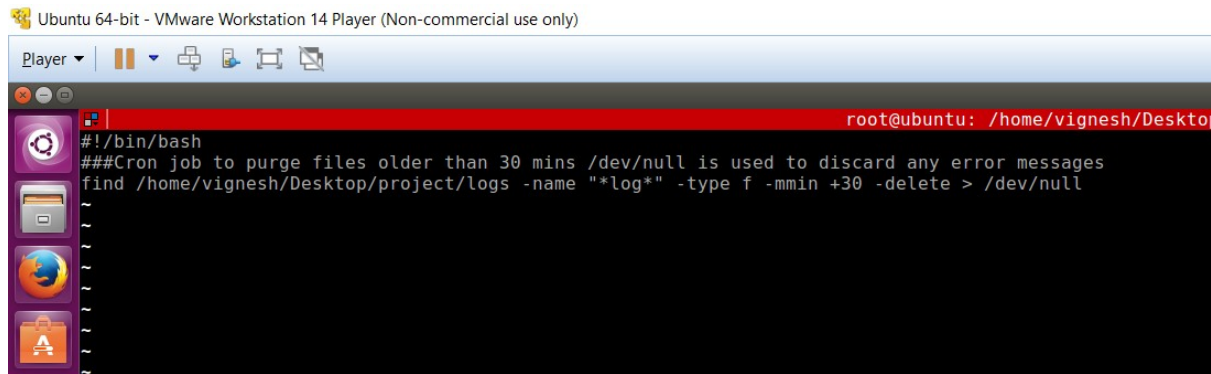


Fig [7]: Purge script which is executed every minute to purge files older than 30 minutes

For testing purpose, I have also created a host monitoring script which can ping the specified host and based on the results it can trigger an email to the user. As I cannot test it out on any major website, I created this script to show the alerting mechanism.

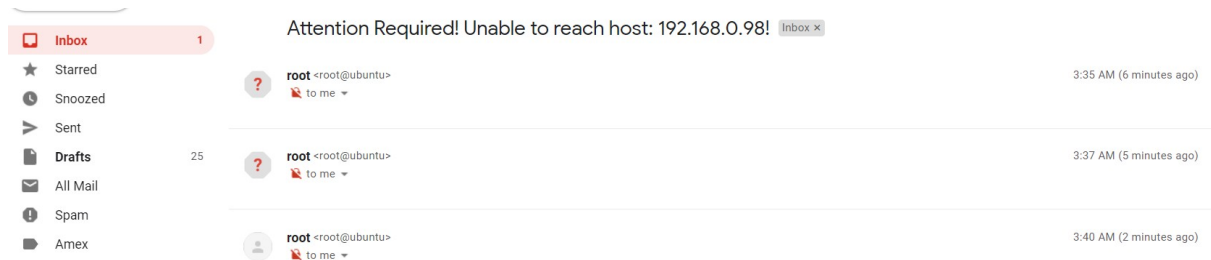


Fig [8]: Displays the email triggered as the host went down

I have created three script files, intranetmonitor.sh, internetmonitor.sh and hostmonitor.sh. In intranetmonitor.sh, the script file looks for “*” sign in the final log file to which means there is a connection drop-out. This script will be adequate to monitor intranet urls where connection dropouts aren’t common. In internetmonitor.sh, the script file looks for the packets sent, and received and triggers and email if the number of packets received is less than 5 (default number of packets set). During testing I got a good example for this, where I received only 4 packets from aa.com despite sending 5. The example is captured in the log file 201812060359_internet_monitoring.log which happened at 03:59. Figures [9] and [10] show the screenshots of the error captured and the email alert triggered at the same time.

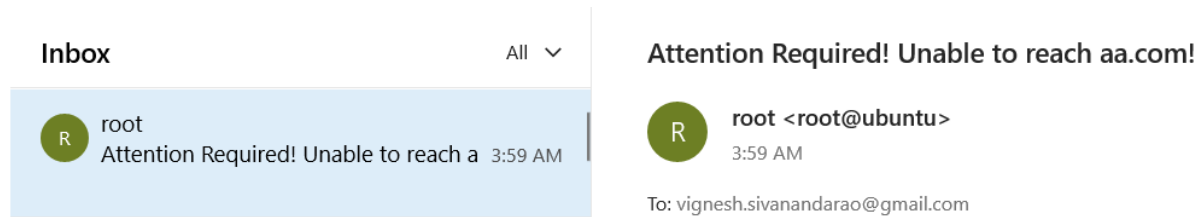


Figure [9]: The server has triggered an email to the mail client as there was 20% packet loss

```

Ubuntu 64-bit - VMware Workstation 14 Player (Non-commercial use only)
Player
root@ubuntu: /home/vignesh/Desktop/project
root@ubuntu: /home/vignesh/Desktop/project 185

PING aa.com (23.67.240.159) 56(84) bytes of data.
64 bytes from a23-67-240-159.deploy.static.akamaitechnologies.com (23.67.240.159): icmp_seq=1 ttl=128 time=56.5 ms
64 bytes from a23-67-240-159.deploy.static.akamaitechnologies.com (23.67.240.159): icmp_seq=2 ttl=128 time=71.0 ms
64 bytes from a23-67-240-159.deploy.static.akamaitechnologies.com (23.67.240.159): icmp_seq=4 ttl=128 time=57.2 ms
64 bytes from a23-67-240-159.deploy.static.akamaitechnologies.com (23.67.240.159): icmp_seq=5 ttl=128 time=60.9 ms

--- aa.com ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4021ms
rtt min/avg/max/mdev = 56.520/61.431/71.012/5.787 ms

telnet through port 443
Trying 104.80.208.143...
Connected to aa.com.
Escape character is '^]'.

telnet through port 80
Trying 23.67.240.159...
Connected to aa.com.
Escape character is '^]'.

traceroute to aa.com (23.67.240.159), 30 hops max, 60 byte packets
 1 192.168.78.2 (192.168.78.2) 0.183 ms 0.115 ms 0.082 ms
 2 192.168.0.1 (192.168.0.1) 4.783 ms 4.750 ms 4.698 ms
 3 173-19-62-1.client.mchsi.com (173.19.62.1) 46.664 ms * *
 4 * * 172.30.3.129 (172.30.3.129) 20.974 ms
 5 68-66-73-18.client.mchsi.com (68.66.73.18) 29.234 ms 43.085 ms 43.057 ms
 6 po10.stlmo001er1.mchsi.com (68.66.73.105) 43.016 ms 29.847 ms 29.721 ms
 7 stlo-b1-link.telvia.net (213.248.85.84) 28.365 ms 41.307 ms 43.107 ms
 8 chi-b21-link.telvia.net (62.115.113.173) 43.095 ms 43.024 ms 42.949 ms
 9 centurylink-ic-334981-chi-b21.c.telvia.net (62.115.162.21) 42.885 ms 42.813 ms 42.761 ms
10 dax-edge-06.inet.qwest.net (67.14.134.170) 52.534 ms 52.433 ms 52.278 ms
11 67.148.171.106 (67.148.171.106) 71.225 ms 69.537 ms 67.762 ms
12 a23-67-240-159.deploy.static.akamaitechnologies.com (23.67.240.159) 61.363 ms 61.418 ms 67.449 ms

Thu Dec 6 03:59:36 CST 2018: Connection failure identified!
201812060359 internet_monitoring.log (END)

```

Figure [10]: The log file has logged the packet loss as well as connection failure

Conclusion:

With all the above setup, this project can not only act as an application but can also be a life saver for production support organizations where the site should be up and running 24x7 and even a single second of downtime can be crucial. Since this project is platform dependent, I have included a video to show the basic working of it. The monitoring scripts can be modified to test any url on the web. The purge script can also be modified to purge files older than any specific time.

Monitoring is a must needed implementation in today's world where everyone and everything is dependent on internet connectivity. No matter how strong the connectivity is, there are factors which keeps affecting our internet connectivity every now and then. Though there would be impact because of connectivity loss, it can be minimized drastically if it is noticed quickly. That is where this project would serve its purpose.

References:

Fig [5]: <http://www.adminschoice.com/crontab-quick-reference>

<https://whatismyipaddress.com/smtp>

https://www.livinginternet.com/i/ia_tools_ping.htm#works

https://support.code42.com/CrashPlan/4/Troubleshooting/Test_your_network_connection

<https://www.mediacollege.com/internet/troubleshooter/traceroute.html>