

Transparent Proxy Cache Server using Raspberry Pi

Anu Rose Jolly^{1*}, M. Kalyan Chakravarthi¹, Naveen Kumar Jindal² and Dinesh Birlasekaran²

¹School of Electronics Engineering, VIT University, Chennai - 600127, Tamil Nadu, India; jlaavr4@gmail.com, maddikerakalyan@vit.ac.in

²VIDEO BU, ION, Nokia India Pvt. Lmted, Kandanchavadi, Chennai – 600096, Tamil Nadu, India; naveen_kumar.jindal@nokia.com, dinesh.birlasekaran@nokia.com

Abstract

Objectives: The online traffic surge rates have rapid growth in the recent years. The rate has gone high, making the service providers unable to provide the quality of service they assure. The surging cannot be controlled and the only solution lays on the other side, so the providers are forced to choose another methodology of delivering the service in quality. **Methods/Statistical Analysis:** In this paper the process of implementing an alternative is proposed. The work is concentrated on creating a Raspberry Pi to work as a transparent cache server with the smart phones enacting as the client. **Findings:** The issues of streaming videos are ending up being more critical to customers as they pay for over-the-top substance yet, still experience exceptionally less than impressive streams. To get a quick response and decrease access latency, utilizing a cache memory is prudent. A transparent proxy cache server wipes out numerous disadvantage of ordinary methodology. Transparent proxy cache server can be conveyed in two levels, one at switch level and another at router level. The attention is primarily on http result reserve which stores URLs of Previously accessed query results. These caches might be deployed in various machines, acting about as a proxy cache cluster, or exists together in the same machine. In this proposed framework we introduce a straight forward web store middle server, to upgrade the execution. The only work to be done regarding the promise of content availability is continuous monitoring of the cache. **Application/Improvement:** The proposed methodology can serve the right speed of content delivery by facilitating access of data from the cache while giving no hint that the data has been retrieved through IP-spoofing of the server which here is a small sized computer the Raspberry Pi.

Keywords: Traffic, Quality of Service, Surge, Raspberry Pi, Latency, Proxy, Cache, IP-Spoofing

1. Introduction

Data centre platforms and cloud¹ endure disappointments and performance degradation from extensive movement surges brought about by both external (e.g., DDoS assaults) or internal (e.g operator errors, workload changes, routing misconfigurations)² factors. Traffic overload could have significant financial and bandwidth availability implications for data providers. The multidimensional surge in content delivery for end-users³ has lead to blast of new content designs and an exponential increment in the size and complexity⁴ on the advanced content delivery network.

Content delivery is a fundamental idea to meet the heterogeneous prerequisites of web clients⁵ utilizing different web access innovations⁶. Be that as it may, content adaptation interferes along with the effectiveness of web caching⁷.

Transparent Caching is a one of a kind innovation that at the same time benefits a substance proprietor, network operator, and in particular a broadband or remote subscriber. Online video activity keeps on spiralling upward every second. It works over a much more extensive arrangement of over the top content and traffic⁸ (as much as 75% of an provider subscriber broadband traffic is video streams⁹ and document downloads¹⁰), it is

*Author for correspondence

implanted inside the carrier network and gives the provider control over what to store, when to store, and how fast to quicken the delivery¹¹. Transparent cache needs to address the high demand of Internet content thirst as could reasonably be expected¹². It naturally ingests and serves content as it gets to be well known and typically does not require operator assistance to constantly alter the system or the reserving answer for backing another mainstream service¹³.

Transparent Caching does not require alteration of any framework or program settings for the end user. The execution advantages ought to be programmed as the main proof of storing ought to be better end-to-end execution to the client¹⁴. Deployed throughout a carrier network, it will enhance endusers' experience and lessen transporters' peering costs¹⁵, yet just in the event that it conveys the components and insight required to adjust to continually changing client conduct and substance designs, and above all, scales economically to tens or several gigabits every second¹⁶.

2. Experimental Setup

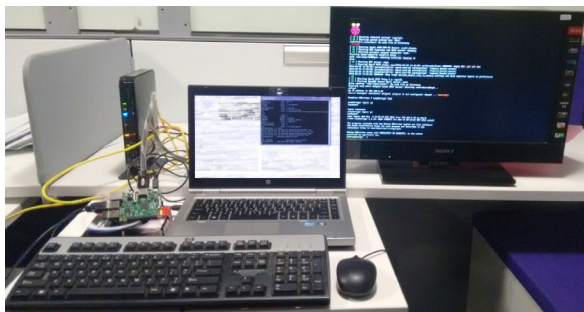


Figure 1. Real-time hardware setup.

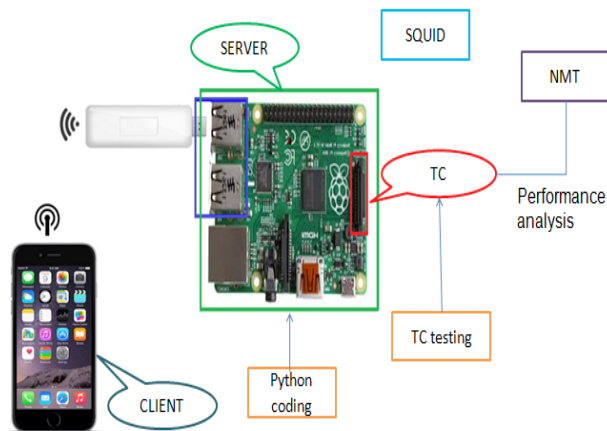


Figure 2. Block diagram of the real time set up.

The work is based on the process of converting the Raspberry Pi as the Server while the Micro SD card on the RPi acts as the cache and the smart phones acts as the client. The monitoring of the cache is done as well as the system testing is done by generating the traffic and checking the sync of cache with the server and the client. The whole implementation is shown in Figure 1 and Figure 2.

2.1 Server

The Raspberry Pi acts as the server for the system. Raspberry Pi is a low-cost credit card sized computer having chips and I/O connectors. The basic board with its features is shown in Figure 3.

The particularities which make Raspberry Pi different from other embedded boards are:

- Broadcom BCM2836 Arm7 Quad Core Processor augmented Single Board Computer at 900 MHz
- 1 GB RAM with 40 pin extended GPIO and 4 x USB 2 ports
- 4 pole Stereo output and Composite video port output at 1080 P with Full size HDMI and CSI camera port
- Micro SD port for loading your operating system and storing data
- Micro USB power source

10/100 Ethernet Port to rapidly associate the Raspberry Pi to Internet

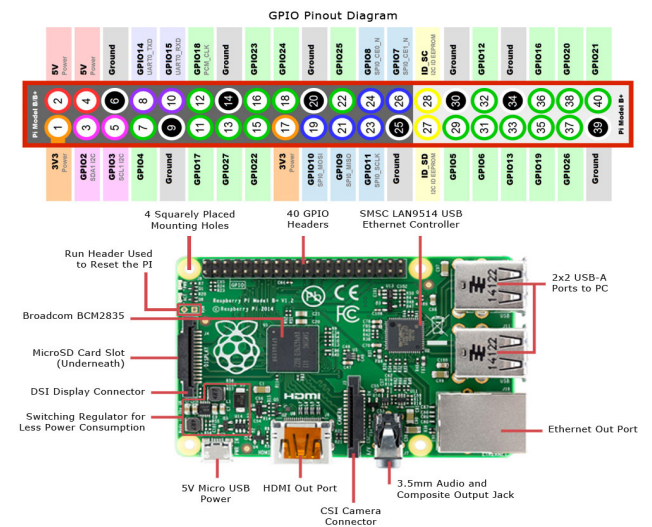


Figure 3. Raspberry Pi.

The step to trigger the OS into raspberry Pi follows the under-given sequence

- Step 1. Booting the OS
- The OS booting follows a few steps:
- Step 2. Selecting the images
- The latest image file for the OS is downloaded from raspberrypi.org/downloads.
- Step 3. Unzipping the image file
- 7-zip supports the unzipping of the downloaded image file.
- Step 4. Writing to Micro SD card
- SD card is mounted with the image file using the Win32 Disk Imager software. The image and the device are selected for writing the image. SD card class determines the speed of the mounting process.
- Step 5. Inserting SD card into RPi
- The SD card is inserted into RPi. On power on the Rpi has the OS loaded.
- Step 6. Accessing RPi

There are 2 methodologies available to access the Raspberry pi. Firstly the GUI can be launched from the boot screen by using the command `statrx` and secondly by using the putty terminal which is an open source emulator having a serial console with network file transfer application used to see the results.

2.2 SQUID and NMT

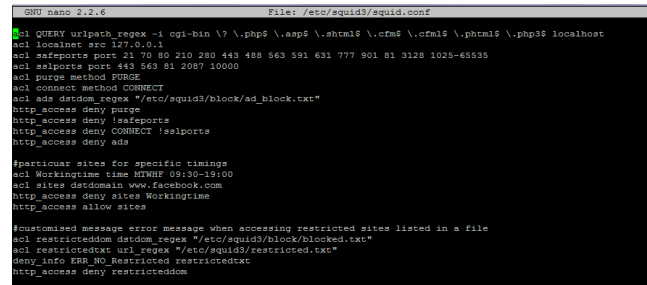
Squid is a Unix-based proxy server that reserves Internet content more like a requestor than its origin point. Squid underpins storing of a wide range of sorts of Web articles, including those got to through HTTP and FTP. It caches the often accessed Web pages, media documents and other contents and quickens the response time as well as reduces bandwidth congestion.

The squid analyser gives the statistics of data that has been accessed through the cache. The squid user access report gives the information based on the majorly accessed sites, the sites and users, the elapsed time for a request and produces the report for each user with the period of access.

The squid config file is changed in order to act as a web filtering. This increases the security defence, by enabling the setup to successfully handle the virus's threats and attacks. The changes made in the config file are shown in Figure 4.

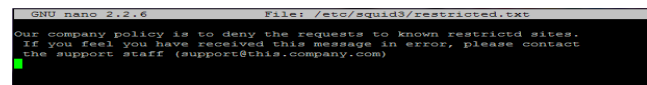
The blocked URLs and the text to be displayed when accessing restricted URLs are entered in .txt files to

respective folders to work as expected. The files are shown as in Figure 5 and Figure 6.



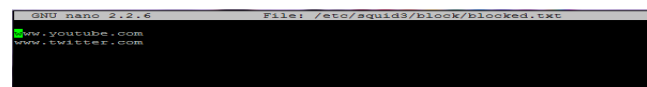
```
GNU nano 2.2.6 File: /etc/squid3/squid.conf
acl QUERY urlpath_regex ^1 cgi-bin \? \.php$ \.asp$ \.html$ \.css$ \.xml$ \.png$ localhost
acl localnet src 127.0.0.1
acl safeports port 21 70 80 210 280 443 488 563 591 631 777 901 81 3128 1025-65535
acl sslports port 443 563 81 2087 10000
acl purge method PURGE
acl connect method CONNECT
acl ads domain_regex ^/etc/squid3/block/ads_block.txt*
http_access deny purge
http_access deny !safeports
http_access deny CONNECT !sslports
http_access deny ads
#Restrictor sites for specific timings
acl Workingtime time MTHW 09:30-19:00
acl sites datdomain www.facebook.com
http_access deny sites Workingtime
http_access allow sites
#Customized message error message when accessing restricted sites listed in a file
acl restrictiondomain url_regex ^/etc/squid3/block/restricted.txt*
acl restrictedtxt url_regex ^/etc/squid3/restricted.txt*
deny_info ERR_NO_Restricted restrictedtxt
http_access deny restrictedtxt
```

Figure 4. The SQUID config file.



```
GNU nano 2.2.6 File: /etc/squid3/restricted.txt
Our company policy is to deny the requests to known restricted sites.
If you feel you have received this message in error, please contact
the support staff (support@this.company.com)
```

Figure 5. Text to be displayed.

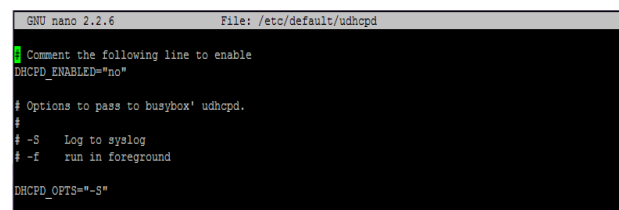


```
GNU nano 2.2.6 File: /etc/squid3/block/ads_block.txt
www.youtube.com
www.twitter.com
```

Figure 6. Blocked URLs.

2.3 Client

The smart phone acts as the client which accesses the server through the transparent cache. The smart phone communicates through the Wi-Fi adapter connected to the RPi. The configuration of the client occurs when the Rpi is configured for the adapter to act as an Access Point. This feature is enabled by using `hostapd`; configuration shown in Figure 7, which is used for access points and authentication servers.



```
GNU nano 2.2.6 File: /etc/default/udhcpd
# Comment the following line to enable
DHCPD_ENABLED="no"
#
# Options to pass to busybox' udhcpd.
#
# -S Log to syslog
# -f run in foreground
DHCPD_OPTS="-S"
```

Figure 7. Udhcpd config file.

The hosted cannot enable the IP to the adapter and for the same we use `udhcpd` and `dnsmasq` which acts as a small DHCP server by enabling the range of accessible IPs and enabling tethering on smart phones respectively. The respective change in configuration and installations is shown in Figure 8 and Figure 9.

```

GNU nano 2.2.6 File: /etc/hostapd/hostapd.conf
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ctrl_interface_group=0
ssid=SQUIDRPI
channel=1

hwaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0

wpa=2
wpa_passphrase=raspberry
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP

ieee80211n=1
hw_mode=g
    
```

Figure 8. Hosted config file.

```

pi@raspberrypi ~$ sudo apt-get install -y hostapd dnsmasq
Reading package lists... Done
Building dependency tree
Reading state information... Done
hostapd is already the newest version.
The following extra packages will be installed:
  dnsmasq-base libnetfilter-conntrack3
The following NEW packages will be installed:
  dnsmasq dnsmasq-base libnetfilter-conntrack3
0 upgraded, 3 newly installed, 0 to remove and 12 not upgraded.
Need to get 404 kB of archives.
After this operation, 825 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libnetfilter-conntrack3 armhf
 1 [32.2 kB]
Get:2 http://mirrordirector.raspbian.org/raspbian/ wheezy/main dnsmasq-base armhf 2.62-3+deb7u3
 [16
 4 kB]
Get:3 http://mirrordirector.raspbian.org/raspbian/ wheezy/main dnsmasq all 2.62-3+deb7u3 [16
 4 kB]
Selected previously unselected package libnetfilter-conntrack3:armhf.
(Reading database ... 50960 files and directories currently installed.)
Unpacking libnetfilter-conntrack3:armhf (from .../libnetfilter-conntrack3_1.0.1-1_armhf.deb) ...
Selecting previously unselected package dnsmasq-base.
Unpacking dnsmasq-base (from .../dnsmasq-base_2.62-3+deb7u3_armhf.deb) ...
Selecting previously unselected package dnsmasq.
Unpacking dnsmasq (from .../dnsmasq_2.62-3+deb7u3_all.deb) ...
Processing triggers for man-db ...
Setting up libnetfilter-conntrack3:armhf (1.0.1-1) ...
Setting up dnsmasq-base (2.62-3+deb7u3) ...
Setting up dnsmasq (2.62-3+deb7u3) ...
[ OK ] Starting DNS forwarder and DHCP server: dnsmasq.
pi@raspberrypi ~$ sudo nano /etc/hostapd/hostapd.conf
pi@raspberrypi ~$ sudo nano /etc/network/interfaces
pi@raspberrypi ~$ sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.bak
pi@raspberrypi ~$ sudo nano /etc/dnsmasq.conf
    
```

Figure 9. Dnsmasq installation.

The pi has to find the location of the config file from the /etc/default/hostapd as shown in Figure 10.

The wpa_supplicant file is edited as shown in Figure 11 for the required configuration of the Wi-Fi hotspot which links the client to the TC. This file is used to execute the key arrangement with a WPA Authenticator and it controls the IEEE 802.11 authentication of the wlan driver.

```

GNU nano 2.2.6 File: /etc/default/hostapd Modified
# Defaults for hostapd initscript
#
# See /usr/share/doc/hostapd/README.Debian for information about alternative
# methods of managing hostapd.
#
# Uncomment and set DAEMON_CONF to the absolute path of a hostapd configuration
# file and hostapd will be started during system boot. An example configuration
# file can be found at /usr/share/doc/hostapd/examples/hostapd.conf.gz
DAEMON_CONF="/etc/hostapd/hostapd.conf"
#
# Additional daemon options to be appended to hostapd command:-
#
# -d show more debug messages (-dd for even more)
# -K include key data in debug messages
# -t include timestamps in some debug messages
#
# Note that -B (daemon mode) and -P (pidfile) options are automatically
# configured by the init.d script and must not be added to DAEMON_OPTS.
#
#DAEMON_OPTS=""
    
```

Figure 10. Default hostapd file.

The setting up of communication between the clients and the Wi-Fi requires NAT. For the same the following Figure 12 has the steps leading to Figure 13.

```

GNU nano 2.2.6 File: /etc/wpa_supplicant/wpa_supplicant.conf
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="SQUIDRPI"
    key_mgmt=WPA-PSK
    psk="raspberrypi"
}
    
```

Figure 11. Enabling authentication for the hotspot generated.

```

pi@raspberrypi ~$ sudo nano /etc/sysctl.conf
pi@raspberrypi ~$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
pi@raspberrypi ~$ sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
pi@raspberrypi ~$ sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
pi@raspberrypi ~$ sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
pi@raspberrypi ~$ sudo update-rc.d hostapd enable
update-rc.d: using dependency based boot sequencing
pi@raspberrypi ~$ hostapd -dd /etc/hostapd/hostapd.conf
    
```

Figure 12. Starting IP forwarding on boot up.

```

GNU nano 2.2.6 File: /etc/sysctl.conf Modified
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables
# See sysctl.conf(5) for information.
#
#kernel.domainname = example.com
#
# Uncomment the following to stop low-level messages on console
kernel.printk = 3 4 1 3
#####
# Functions previously found in netbase
#
# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1
net.ipv4.ip_forward=1
    
```

Figure 13. Changes in sysctl config.

After all these changes the system is rebooted along with service network being restarted. The present ifconfig will be as in Figure 14.

```

pi@raspberrypi ~$ ifconfig
eth0    Link encap:Ethernet HWaddr b8:27:eb:ad:c8:83
        inet addr:192.168.5.44 Bcast:192.168.5.255 Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:111 errors:0 dropped:0 overruns:0 frame:0
        TX packets:115 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:10547 (10.2 KiB) TX bytes:16812 (16.4 KiB)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:65536 Metric:1
        RX packets:10 errors:0 dropped:0 overruns:0 frame:0
        TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:934 (934.0 B) TX bytes:934 (934.0 B)

wlan0   Link encap:Ethernet HWaddr e8:4e:06:2d:c7:11
        inet addr:192.168.42.21 Bcast:192.168.42.255 Mask:255.255.255.0
        UP BROADCAST MULTICAST MTU:1500 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
    
```

Figure 14. If config file.

The after effect of the above configuration changes and installations the Wi-Fi hotspot is enabled and so the client can access it. The Figure 15 shows the Wi-Fi made visible in the client.

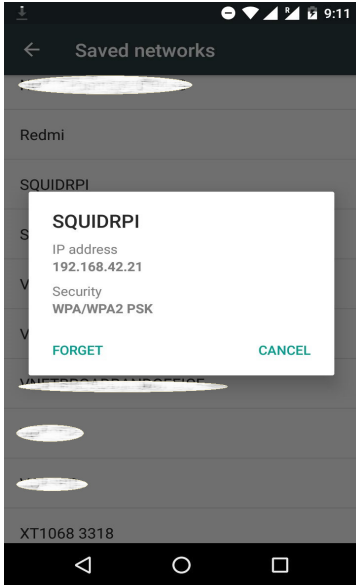


Figure 15. Wi-Fi created viewed in the client.

2.4 TC (Transparent Cache)

The SD card in the Raspberry Pi acts as the Cache in the implemented system. The transparent caching enables quickening of origin caching. This permits mezzanine substance to be offloaded to a committed media server for ensuing change to versatile bitrates for adaptive delivery.

A transparent cache has the features of multiservice caching; automatically selection to popular content, its transparent to both subscriber or the server and content origin or the client, based on the operator settings and can be controlled by the administrator.

2.5 TC Testing

To verify that each time the client tries to access the server the request is sent to the cache and if it is a cache miss then only forwarded to the server. No direct pull from the server is made by the client. The log file is configured to list the client trying to access the data and the mode in which the data is pushed through the cache by using the variables like TCP_MISS, TCP_HIT, TCP_REFRESH_MODIFIED, TCP_REFRESH_UNMODIFIED. The log files are shown as below in Figure 16.

2.6 Traffic Generation

Python, the high level programming language is used for programming. The several libraries of python like requests and urllib are used for the accessing of http requests and Urls. The Urls can be called in loops for simple program-

ming while to create traffic they are to be accessed parallel and for that to be implemented we use multi threading. The Figure 17 shows the flowchart for the implementation.

```

1447733331.544 97 192.168.5.14 TCP_MISS/200 1274 GET http://translate.google.com/translate_a/element.js? - DIRECT/212.159.143.170
1447733331.940 97 192.168.5.14 TCP_MISS/200 553 GET http://www.google-analytics.com/collect? - DIRECT/212.159.143.170
1447733332.129 98 192.168.5.14 TCP_MISS/200 358 GET http://www.google-analytics.com/collect? - DIRECT/212.159.143.170
1447733332.900 245 192.168.5.14 TCP_MISS/200 334 GET http://bam.nr-data.net/1/14P0c0048? - DIRECT/50.31.144.174
1447733333.129 797 192.168.5.14 TCP_REFRESH_MODIFIED/200 23979 GET http://www.squid-cache.org/Doc/config/logformat? - DIRECT/212.159.143.170
1447733333.109 491 192.168.5.14 TCP_REFRESH_MODIFIED/200 3973 GET http://www.squid-cache.org/default.css - DIRECT/212.159.143.170
1447733333.174 547 192.168.5.14 TCP_REFRESH_MODIFIED/200 1102 GET http://www.squid-cache.org/cfman.css - DIRECT/212.159.143.170
1447733333.485 739 192.168.5.14 TCP_REFRESH_MODIFIED/200 29194 GET http://www.squid-cache.org/Images/img4.jpg - DIRECT/212.159.143.170
1447733333.600 442 192.168.5.14 TCP_REFRESH_MODIFIED/200 821 GET http://www.squid-cache.org/Images/img5.gif - DIRECT/212.159.143.170
1447733333.601 460 192.168.5.14 TCP_REFRESH_MODIFIED/200 835 GET http://www.squid-cache.org/Images/img3.gif - DIRECT/212.159.143.170
1447733333.602 493 192.168.5.14 TCP_REFRESH_MODIFIED/200 849 GET http://www.squid-cache.org/Images/img1.gif - DIRECT/212.159.143.170
1447733333.603 457 192.168.5.14 TCP_REFRESH_MODIFIED/200 488 GET http://www.squid-cache.org/Images/img5.gif - DIRECT/212.159.143.170
1447733333.675 504 192.168.5.14 TCP_REFRESH_MODIFIED/200 505 GET http://www.squid-cache.org/Images/img2.gif - DIRECT/212.159.143.170
1447733333.686 564 192.168.5.14 TCP_REFRESH_MODIFIED/200 486 GET http://www.squid-cache.org/Images/img7.gif - DIRECT/212.159.143.170
1447733333.784 429 192.168.5.14 TCP_REFRESH_UNMODIFIED/304 291 GET http://www.squid-cache.org/Doc/config/logformat? - DIRECT/212.159.143.170
1447733333.129 494 192.168.5.14 TCP_REFRESH_UNMODIFIED/304 282 GET http://www.squid-cache.org/default.css - DIRECT/212.159.143.170
1447733333.613 449 192.168.5.14 TCP_REFRESH_UNMODIFIED/304 282 GET http://www.squid-cache.org/cfman.css - DIRECT/212.159.143.170
1447733333.630 449 192.168.5.14 TCP_REFRESH_UNMODIFIED/304 283 GET http://www.squid-cache.org/Images/img1.jpg - DIRECT/212.159.143.170
    
```

Figure 16. Log file format.

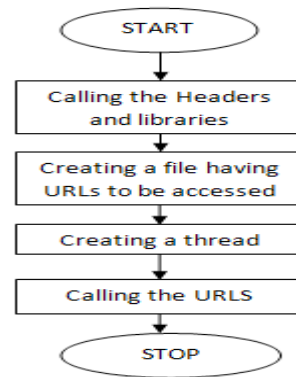


Figure 17. Program Flow chart for traffic generation.

3. Results

By deploying the methodology proposed it can be made sure that the data is obtained from the cache and the content delivery of the provider can be evaluated as well as the betterment of the service can be done by looking at the data hit rate. Figure 18 and 19 shows the report generated by the Squid analyzer. In the report the time taken for data access can be viewed and it leads to a better note that the cache hit rate can increase the data availability and thus the providers can satisfy the upcoming online traffic surge.

4. Future work

The work can be extended to have a continuous check of the cache in every defined time instance and the probability of cache hit be increased so that very little server access is being done. The traffic generation can be extended by

increasing the incoming content traffic and the resultant outcomes in order leading to know the efficiency of the system.

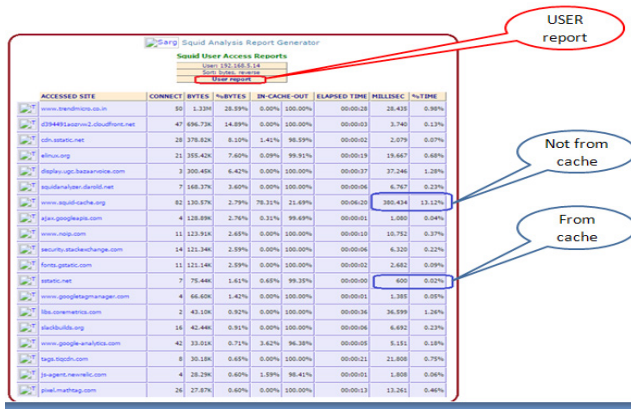


Figure 18. The User specific report with the elapsed time for each site.



Figure 19. Report of the sites and users accesses it along with the top no of sites being accessed.

5. Conclusion

The methodology can be a boon to the providers to handle the increasing online traffic and so that they can handle the online traffic surge. This work effectively gives a solution to control the hit rate to the server caused by traffic surge. This can be implemented to upgrade the service quality the providers give and to the customers the satisfaction of reception of the content delivered.

6. Acknowledgement

We would like to thank the VIT University and Nokia for giving us such an opportunity to carry out this research work and also for providing us the requisite resources and infrastructure for carrying out the research. The results of this work required a considerable measure of direction and would thank my guides for the timely guidance and suggestions which helped in the fulfilment of the work.

7. References

- Wang Y, Zhang Y, Singh Y. Net fuse: Short-circuiting traffic surges in the cloud. IEEE ICC - Next-Generation Networking Symposium; 2013. p. 1-5.
- Xu W, Wang F, Bhattacharyya S, Zhang Z. A real-time network traffic profiling system. 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. USA. 2007. p. 595-605.
- A Study of the Impact of Network Traffic Pacing from Network and End-User Perspectives. Available from: <http://www.nal.ics.es.osaka-u.ac.jp/personal/s-hanay/iccn11.pdf>. 2011
- Jonckheere MTS, Nez-Queija RN, Prabhu B. Performance analysis of traffic surges in multi-class communication networks. IEEE 22nd International Tele-traffic Congress; 2012.
- Fudan CW, Fudan JW, Zeng J. Network traffic awareness architecture for universal redundancy elimination. International Conference on Electronic & Mechanical Engineering and Information Technology; Harbin, China. 2011.
- Bouchenak S, Cox A, Dropsho S, Mittal S, Waenepoel W. Caching dynamic web content: Designing and analyzing an aspect-oriented solution. Lecture Notes in Computer Science. 2006; 4290:1-21.
- Buchholz S, Schill A. Adaptation aware web caching: Caching in the future pervasive web. Kommunikation in Verteilten Systemen (KiVS) Part of the Series Informatik Aktuell. 2003; 55-66.
- Jia X, Li D, Du H. On optimal replication of data object at hierarchical and transparent web proxies. IEEE Transactions on Parallel and Distributed Systems. 2005; 16(8):673-85.
- Nishimura S, Shimamura M, Koga H, Ikenaga T. Transparent caching scheme on advanced relay nodes for streaming services. International Conference on Information Networking; Japan. 2012. p. 404-19.
- Rohini G, Srinivasan A. Multi server based cloud-assisted real-time translating for Http live streaming. Indian Journal of Science and Technology. 2016 Jan; 9(3):1-5.

11. Gao Y, Zhang Y, Zhou Y. A cache management strategy for transparent computing storage system. *Communications in Computer and Information Science*. 2012; 320:651-8.
12. Dhomeja LS, Malkani YA, Shaikh AA, Keerio A. Transparent caching of virtual stubs for Improved performance in ubiquitous environments. *International Journal of UbiComp*. 2011; 2(4):1-14.
13. Lenon J, Gardenghi C, Augusto M, Bardi GA. An authentication middleware for squid proxy-cache: A single sign-on approach. *12th International Conference on Computational Science and its Applications*; 2012. p. 138-41.
14. Transparent Distributed Web Caching with Minimum Expected Response Time. 2003. Available from: <http://www.nlc-bnc.ca/obj/s4/f2/dsk3/ftp04/MQ65661.pdf>
15. Bouras C, Konidaris A, Kostoulas D. Predictive prefetching on the web and its potential impact in the wide area. *World Wide Web*. 2004 Jun; 7(2):143-79.
16. Kalarani S, Uma GV. Improving the efficiency of retrieved result through transparent proxy cache Server. *4th International Conference on Computing, Communications and Networking Technologies*; India. 2013. p. 1-8.