DLSU RESEARCH CONGRESS 2020
"Building Resilient, Innovative, and Sustainable Societies"
June 17-19, 2020

Network Link Redirector Based on Internet Application Usage

Juan Alfonso Felipe[1], Ramon Miguel Perdices[1], Titus Suarez[1] and *Gregory Cu[1]
[1] *De La Salle University*
*juan_felipe@dlsu.edu.ph, ramon_perdices@dlsu.edu.ph, titus_suarez@dlsu.edu.ph, * gregory.cu@dlsu.edu.ph*

Abstract: Organizations with connections for two or more Internet service providers (ISP) have become more common place in recent years. Among the current technologies to enhance the use of the multiple connections are load balancers. These solutions usually consider bandwidth usage as the metric for balancing load. Some organizations require traffic segregation based on network application since the multiple ISP connection may not have the same bandwidth or same data cap.

The pfSense router/firewall is one such example of a load balancer but is not able to segregate network traffic based on network application. This study aims to provide a software-based solution using pfSense with a "package plugin" that allows it to identify the application being used and route/redirect network traffic to a specific Internet connection. This allows prioritization of applications with higher needs and enables one to customize their network based on application and data consumption. It also takes into consideration ISP connection with a data cap which restricts the use of the connection after the limit is reached.

Key Words: network traffic redirection, network data consumption monitoring, pfSense package

1 INTRODUCTION

In recent years it has become more common for organizations to have two or more Internet connection. In certain set-ups, the ISP (Internet Service Provider) connection can have a bandwidth or data cap when reached, restricts the usage of their services. There exist numerous methods and techniques to route and segregate the data in network traffic in order to optimize and customize the network to the end-users needs. One common technology capable of routing data traffic in a network is firewall/router with pfSense software.

PfSense is an open-source FreeBSD-based routing software that can act as a firewall as well as a load balancer. pfSense is a popular project with more than a million downloads (Spice Works, 2018) as well as having thousands of enterprises using their software as a trusted open source network security solution (pfSense, 2020). Funtionality of pfSense can be extended using packages thru its Package Manager. (Netgate, 2020).

However, pfSense currently does not have application layer support (Grace, 2020) as it has been removed a few years ago, due to having the drawbacks of creating heavy CPU load. This means that pfSense

has no feature for redirecting or segregating network traffic based on the network application being used. This is a problem for pfSense users that require certain application to be prioritized or those that require certain applications to be dedicated to a particular ISP. For example, a network user, who has two ISPs, uses a computer with pfSense software for his router may find a need to have messaging applications pass through an ISP connection (ISP1) which has a data limit in order to free the other ISP connection (ISP2) for watching YouTube videos or other video streaming applications.

The study aims to create a system that redirect traffic based on the user's application preferences. The main objective is to develop the network application redirector / segregator as a package for pfSense.

Specifically, this study aims to:

- To develop an application package that allows the user to specify the ISP for internet applications to be redirected.
- To create a traffic redirection algorithm based on the application being used.
- To implement a bandwidth management scheme that considers the state of the data cap and redirects data once the data cap is reached or reaches a certain point.

For this study, several existing technologies were used in order to help create the Application redirection system. All the technologies utilized are either packages or plug-ins applied to pfSense.

2 SYSTEM IMPLEMENTAION

2.1 System Setup

The network setup of the system uses a pfSense firewall that has three network connections: two connections to the Internet and one to the internal / private network. Connection to the Internet has one connection to the LTE router while the other is a direct connection to the network laboratory of the college, as shown in Fig. 1. The LTE router and network laboratory connection simulate the "ISP 1" and "ISP 2" connection mentioned earlier. The system is a developed package running inside the pfSense firewall. The package specifically uses snort (Snort, 2020), vnStat (Toivola, 2020), and openAppID (Netgate, 2020) to segregate traffic.

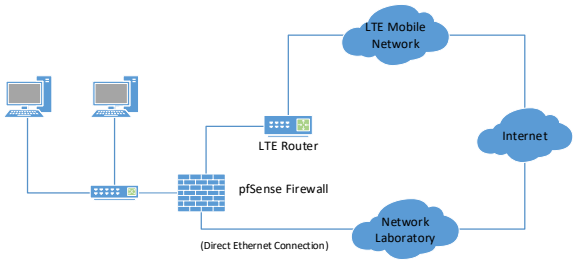

Fig. 1. System Diagram Overview

2.2 Implementation

2.2.1 Application Identification and Rule Generation

In order to segregate network traffic to a specific Internet connection, pfSense was installed with Snort and OpenAppID. This enabled the pfSense firewall to identify the network applications (e.g. Netflix, Youtube, Twitter, Google, Netflix) in the network traffic. All this information is logged inside pfSsense. Users can later categorize certain applications like video streaming, social network and so on later. Also, the user can set priority number to an application to allow it to pass through a specific link.

A daemon was developed to collect data and sorts the logs. The sorted logs are then passed to an extraction process that checks each log for the IP addresses matching it to a corresponding application. This information is saved in a list (as a CSV file) of IP addresses for an application.

The list of IP addresses corresponding to a network application is used as parameters for the firewall rules to be created. Rule creation is based on the IP address from the Application Database using the "easyrule" function from pfSense where it goes into an identification process and the rule gets tagged based on its application name. The created rule is an

egress rule in which Internet interface it is to use. Rules were created sequentially based on the logs.

As an example, collected network traffic from Snort can identify that the Twitter website has an IP address of 104.244.42.194, this is logged in the list (an example list is shown in Fig. 2). This information is then used to create a rule using the IP address of Twitter as destination address and configured to egress to a certain link. An example rule created is shown in Fig. 3. Rules are created sequentially from the list and does not merge all the IP address of an application in one rule. Also, rules are initially blank upon startup so all traffic goes through the default interface.

```
imgur,104.16.0.35
netflix,54.187.51.79
alibaba,198.11.189.30
imgur,151.101.24.193
imgur,151.101.196.193
reddit,151.101.193.140
reddit,151.101.9.140
reddit,3.225.77.184
bluekai,104.105.19.224
netflix,35.164.154.29
twitchtv,52.37.202.185
twitchtv,52.27.66.73
twitter,104.244.42.2
dropbox,104.16.99.29
twitter,104.244.43.131
twitter,104.244.42.129
bing,204.79.197.200
twitter,104.244.42.194
```

Fig. 2. Example Log of IP Address & Application

	Source : Port	Destination : Port	Gateway	Description
IPv4	any : any	104.244.42.194 : any	FAST_LINK	Application / Website Name

Fig. 3. Created Rule for Twitter Application

2.2.2 Data Cap Tracker, Priority Transfer and Redirection

Data monitoring was achieved by running vnstat commands and collecting the output. This output was parsed to get the data usage and checks if the total amount of data usage has exceeded the data cap. A built-in function in vnStat is used to reset data usage information every month.

The system gradually transfer links based on priority level of the applications and the priority level allowed to pass through the faster connection. The system then switches the rules that fall below the allowed priority level to the slower link. Table 1 further explain what happens when a certain percentage of data usage is reached.

Table 1 Priority Transfer Percentages

PERCENTAGES	< 10%	< 20%	< 30%	< 40%	< 50%	< 60%
ALLOWED PRIORITY LEVELS	>= 1	>= 2	>= 3	>= 4	>= 5	>= 6
PERCENTAGES	< 70%	< 80%	< 90%	< 100%		
ALLOWED PRIORITY LEVELS	>= 7	>= 8	>= 9	10		

When the data usage reaches a certain percentage, there are possible scenarios that can occur. Scenario A was addressed by redistributing the rules between the two connections; Scenario B was addressed by redirecting all the rules to the lower priority link if the data cap is reached; Scenario C was addressed by prompting the user if they are willing to continue using main ISP, otherwise the rules are made to redirect everything to the slower link. Table 2 shows the different scenarios the submodule can use.

Table 2. Scenario Types

Type	Scenario
A	Lower Bandwidth
B	Cut connection
C	Extra Payment

3 Test and results

Network setup for testing is seen in Fig. 1Fig. 1. System Diagram Overview using 2 WAN connections, one is connected to a prepaid LTE Router and the other is connected to the Network Laboratory of the college. Both are connected to a pfSense firewall. The pfSense firewall was running on computer with an Intel Core i3 370M 2.4GHz processor and 4GB memory.

3.1 Application Identification Test

The Application Identification test aims to determine whether the application extraction from the logs and subsequent updating of the csv is successful. This was done by visiting numerous sites that are part of the OpenAppID and verified by checking the extracted logs file as well checking the web interfaces of the applications uploaded to the csv.

As can be seen in Table 3. Application Identification Test Table The extraction of logs and

uploaded of information to the CSV showed a 63.63% success rate conducted with 22 test cases. 8 cases failed it either displayed a wrong app or Snort failed to detect it; if Snort and OpenAppID fail to identify then the system cannot add it to its own database. This occurrence could be seen from sites that have changed names, the site is outdated, or simply the information in OpenAppID has not been updated to be able to detect or identify them.

Table 3. Application Identification Test Table

Application	Successful Extraction from Logs		Successful upload to CSV	
	Expected	Actual	Expected	Actual
Bing	Yes	Yes	Bing	Yes
Bluestacks	Yes	Yes	Bluestacks	Yes
Dailymotion	Yes	Yes	Dailymotion	Yes
Daum	Yes	No	Daum	Yes
Facebook	Yes	Undetected	Facebook	Yes
FileDropper	Yes	No	FileDropper	Yes
Gamespy	Yes	Yes	Gamespy	Yes
Github	Yes	Yes	Github	Yes
Google	Yes	Yes	Google	Yes
IGN	Yes	Yes	IGN	Yes
Mafiawars	Yes	Undetected	Mafiawars	Yes
Myspace	Yes	Yes	Myspace	Yes
Netflix	Yes	Yes	Netflix	Yes
Reddit	Yes	Yes	Reddit	Yes
ShowClix	Yes	Undetected	ShowClix	Yes
Spotify	Yes	Yes	Spotify	Yes
StayFriends	Yes	Undetected	StayFriends	Yes
Target	Yes	Yes	Target	Yes
Twitchtv	Yes	Yes	Twitchtv	Yes
Twitter	Yes	Yes	Twitter	Yes
Vimeo	Yes	No	Vimeo	Yes
Weibo	Yes	No	Weibo	Yes

3.2 Dynamic Identification and Redirection Test

The Dynamic Identification test aims to check if the system can successfully redirect traffic based on data cap consumed and the three data type scenarios. During this test the data cap limit was lowered to 100MB in order to see immediate results. The Dynamic Redirection test was conducted by observing the Snort Alerts and by running the traceroute command for each Application for every 10% data used.

As seen in the Dynamic Redirection Table (see Table 4), the redirection works properly and redirects the necessary applications between the two ISP connections with 100% success rate. However, one flaw noticed in the redirection is that the more firewall rules that need to be redirected, the slower the redirection process. When tested with the complete list of 2779 applications and each with numerous IP addresses, the redirection process slows significantly due to the incredible number of firewall rules that need to be changed. As such, for testing purposes the number of websites was lessened to 21 to immediately see the results. Due to the system design, the speed of the redirection also affects other processes such as the data cap tracker module as it cannot update the csv file while it is in use by the redirection module.

Table 4. Dynamic Redirection Table

App	Priority Lvl	Result	0-9	10-19	20-29	30-39	40-49
Twitter	1	Expected	High	Low	Low	Low	Low
		Actual	High	Low	Low	Low	Low
Github	4	Expected	High	High	High	High	Low
		Actual	High	High	High	High	Low
Netflix	8	Expected	High	High	High	High	High
		Actual	High	High	High	High	High
App	Priority Lvl	Result	50-59	60-69	70-79	80-89	90-99
Twitter	1	Expected	Low	Low	Low	Low	Low
		Actual	Low	Low	Low	Low	Low
Github	4	Expected	Low	Low	Low	Low	Low
		Actual	Low	Low	Low	Low	Low
Netflix	8	Expected	High	High	High	Low	Low
		Actual	High	High	High	Low	Low

As can be seen in Table 5, Table 6, and Table 7 the redirection module successfully redirected the application to the slower link once the priority of the application no longer met the conditions of the consumed data.

Table 5. Twitter Snort/Traceroute Results Before & After 10%

Snort Results			
Before		After	
Source	Destination	Source	Destination
192.168.1.5	104.244.42.67	172.16.4.62	104.244.42.67
Traceroute Results			
Before		After	
First Hop		First Hop	
192.168.1.1		172.16.4.1	

Table 6. Github Snort/Traceroute Results Before & After 10%

Snort Results			
Before		After	
Source	Destination	Source	Destination
192.168.1.2	34.196.247.240	172.16.4.62	34.196.247.240
Traceroute Results			
Before		After	
First Hop		First Hop	
192.168.1.1		172.16.4.1	

Table 7. Netflix Snort/Traceroute Results Before & After 90%.

Snort Results			
Before		After	
Source	Destination	Source	Destination
192.168.1.5	54.187.176.196	172.16.4.62	34.213.69.2
Traceroute Results			
Before		After	
First Hop		First Hop	
192.168.1.1		172.16.4.1	

3.3 Redirection Latency Test

The Redirection Latency Test aims to check how long does it take for the system to redirect all the app's firewall rules. During this test the number of apps is increased gradually to check on how many apps does it take for till the system slows down, a software was built to timestamp the start and end of the Redirection.

As can be seen in Table 8 the time duration of the redirection slows down due to the increase number of apps in the system that has multiple firewall rules. It can be observed in that there is a more significant jump in redirection time such between 121 and 182 as well as 261 and 293 this is due to some applications having numerous IP addresses such as Netflix and twitter. With Netflix and twitter, each IP addresses is turned into a pfSense rule. Nonetheless the more applications the longer redirection will take. However, some application may have more weight on the system than others.

Table 8. Redirection Latency Test

Number of Rules	Average Redirection Time Duration (seconds)
101	5
121	6
182	13
210	15
235	16
250	16
261	17
293	20

3.4 System Latency Test

The system latency test aims to check the effect of the system on the speed of the internet connection. This was tested using the site https://www.speedtest.net (Ookla, 2020). The internet speed was checked with direct connection from the NETLAB to host PC and then once again with the system between the host and the NETLAB connection.

As can be seen in Table 9 there was a significant drop in download speed while upload speed is almost the same. The download speed dropped significantly with 6 to 7 Mbps lost while upload speed showed an insignificant drop with less than 1 Mbps. The implementation of the application identification, rule generation and redirection has impacted the system in download speed.

Table 9. Internet Speed with and Without the System Average Results (10 Trials)

Without System			With System		
Ping	Download Speed	Upload Speed	Ping	Download Speed	Upload Speed
1	91.68 Mbps	94.25 Mbps	1.2	84.68 Mbps	94.24 Mbps

4 CONCLUSION

4.1 Conclusion

The created application package was able to segregate or redirect network traffic based on the network application. The system is able to successfully extract application from the snort logs and subsequently generate firewall rules for the applications that dictate which ISP connection they pass through. Application detection accuracy was only 63.63% as some of the rules in OpenAppID are outdated.

The constant monitoring of data usage and the connection status of the ISP link allows redirection of the applications based on the priority level assigned to them by the user, using the priority algorithm as shown in Table 5, Table 6, and Table 7. It is also successful in handling the three scenarios that can take place when the data cap is reached as well as when the connection is cut.

The system however slows down when there are many applications and rules to configure and reload. At 293 rule entries, it takes about 20 seconds to redirect traffic.

4.2 Recommendation

It is recommended that the system should run in a hardware that has better specifications because when running the system, the CPU usage spikes to an average of 60% and memory usage at around 25%. The current system runs with an Intel Core i3-370M CPU with 4GB of RAM and uses two USB to LAN adapters. The system also slows down the more rules are added to the system, so the researchers recommend that future studies mediate this flaw by deleting rules that have not been used recently or find a way to merge numerous rules of one application into a single rule to reduce pfSense workload. It is also recommended to create a more efficient algorithm for redirection process, as well as update the code of the system to Python 3.2, as this version of Python is more updated.

5 REFERENCES

Grace, M. (2020). *pfSense Removes L7 Support - Recommends Snort*. Retrieved from Sinefa: https://blog.sinefa.com/blog/2016/8/24/pfsense-removes-layer-7-support-and-recommends-snort

Netgate. (2020, March). *Application Detection on pfSense Software*. Retrieved from Netgate Blog: https://www.netgate.com/blog/application-detection-on-pfsense-software.html

Netgate. (2020, March). *Using the Package Manager*. Retrieved from Netgate pfSense Document: https://www.netgate.com/docs/pfsense/packages/package-manager.html

Ookla. (2020, April). *Speed Test*. Retrieved from Speed Test: https://www.speedtest.net

pfSense. (2020, March). *Take a Tour of pfSense*. Retrieved from pfSense: https://www.pfsense.org/about-pfsense/

Snort. (2020). *Snort - Network Intrusion Detection and Prevention System*. Retrieved from Snort: https://www.snort.org/

Spice Works. (2018, March). *Overview of pfSense*. Retrieved from Spice Works: https://community.spiceworks.com/products/29317-pfsense

Toivola, T. (2020, April). *vnStat*. Retrieved from vnStat: https://humdi.net/vnstat/