SRT411 Take Home Assignment

Michael Nguyen

March 19, 2016

Defining the Problem

Introduction

On March 18th, I let Wireshark run starting from 9:00am until 3:00pm, while instructing my younger brother that he may do whatever he liked on my computer. Some possible scenarios that could come out of this are captures of packets going to various different servers around the world, and a plentitude of possible protocols used. By the time I arrived home, the total packet capture was up to roughly 6.6 million packets and when exported added up to a whopping 1GB file. Knowing the limitation of knitting PDF, if i were to display all lines, by either viewing or printing this in one go would be very resource extensive on individuals trying to open this PDF. My goal is to try to figure out what happened during the time I left Wireshark on, until the time I arrived home and present this information in a way that shows that I uncovered some of this information.

Constructing Questions

Firstly after setting my working directory and importing the *.csv* file, i took a summary of the data in order to see what i was working with. Below is the code that was used: >data=read.csv("mar18")
>info<-summary(data)
>write.csv(info,"info.csv")

```
info=read.csv("info.csv")
print(info)
```

```
##    X         X.....No.      X.....Time       X..........Source
## 1 NA Min.   :      1   Min.   :    0   192.168.1.2   :2647794
## 2 NA 1st Qu.:1663104   1st Qu.: 1336   13.107.4.50   :2084780
## 3 NA Median :3326208   Median : 2126   69.28.210.33  : 292683
## 4 NA Mean   :3326208   Mean   : 5733   149.56.0.38   : 224188
## 5 NA 3rd Qu.:4989312   3rd Qu.: 8131   198.71.80.215 : 199342
## 6 NA Max.   :6652415   Max.   :20854   162.254.199.80: 180900
## 7 NA              <NA>            <NA> (Other)       :1022728
##       X........Destination      X........Protocol       X....Length
## 1 192.168.1.2   :3918079   TCP         :3484267   Min.   :   42.0
## 2 13.107.4.50   :1058182   GVSP        :2555242   1st Qu.:   78.0
## 3 149.56.0.38   : 224566   UDP         : 424644   Median :  194.0
## 4 198.71.80.215 : 199629   ARP         :  76420   Mean   :  615.5
## 5 162.254.199.80: 179217   HTTP        :  48429   3rd Qu.: 1506.0
## 6 162.254.199.90: 146645   CLASSIC-STUN:  23381   Max.   :11594.0
## 7 (Other)       : 926097   (Other)     :  40032              <NA>
##          X...............................Info
## 1 61233  >  28015  Len=11           : 105648
## 2 57138  >  27101  Len=11           :  83732
## 3 28015  >  61233  Len=52           :  72915
## 4 28015  >  61233  Len=11           :  22813
## 5 [TCP segment of a reassembled PDU]:  15804
## 6 27101  >  57138  Len=11           :  14215
## 7 (Other)                           :6337288
```

```
#Since it would be too resource extensive
#to import the dataset into this PDF, workarounds had to be made.
```

In order for us to even look at the summary in this PDF, I had to write the output from looking at summary function to a *csv* or another readable format by *R* and read that instead. This process would be very important in order to construct this document with easy acessibility.

Now, that we have a glance at the information at hand we can begin to develop questions. Some simple ones that could be examined are:

1. How many unique Destination IPs are there?
2. What Protocols were used?
3. Can we look at the frequency of packets throughout the day?
4. Is there any applications or conversations we can look at or identify?

And furthermore we question which way we can plot or visualize this data:

5. Can this data be represented on a map?
6. What constraints do we need to take into account for graphing?

I will attempt to answer these questions if-possible in visual representations, and explore other options where available.

Assessing Available Data

Dataset Information

Looking back at the summary of the dataset in the first part, we can see taht there are **7 different variables** These are as listed below: * No. (The packet number) * Time (ms) * Source IP * Destination IP * Protocol * Packet Length * Information Field of packet

Apart from that we can use our other functions such as summary and mutate to see counts and frequency or create new columns using the current existing ones. It is most likely

Other Sources

Geo-location

In order to solve one of the questions posted in the first section, such as determining geolocation, I will be using this function I found here , and the employment of the function below with a simple example:

```
library(rjson)
freegeoip <- function(ip, format = ifelse(length(ip)==1,'list','dataframe'))
{
    if (1 == length(ip))
    {
        # a single IP address
        require(rjson)
        url <- paste(c("http://freegeoip.net/json/", ip), collapse='')
        ret <- fromJSON(readLines(url, warn=FALSE))
        if (format == 'dataframe')
            ret <- data.frame(t(unlist(ret)))
        return(ret)
    } else {
        ret <- data.frame()
        for (i in 1:length(ip))
        {
            r <- freegeoip(ip[i], format="dataframe")
            ret <- rbind(ret, r)
        }
        return(ret)
    }
}

#Using the first non-private IP found in my dataset

freegeoip("13.107.4.50")
```

```
## $ip
## [1] "13.107.4.50"
##
## $country_code
## [1] "US"
##
## $country_name
## [1] "United States"
##
## $region_code
## [1] ""
##
## $region_name
## [1] ""
##
## $city
## [1] ""
##
## $zip_code
## [1] ""
##
## $time_zone
## [1] ""
##
## $latitude
## [1] 38
##
## $longitude
## [1] -97
##
## $metro_code
## [1] 0
```

With this we can figure out Longitude and Latitude of an *IP*, but we will need to play with our data a bit so we can feed this function more than one value at a time.

Processing Information

Now we proceed with information screening to develop results. There will several parts within this section documenting different methods or techniques used to filter the information from the dataset.

Destinations

First I want to look at the amount of unique destination addresses within the dataset. Using the code below:

> udest <- unique(data$Destination)

```
length(udest$x)
```

```
## [1] 6286
```

```
#We find that there are 6286 unique reults for the Destination Address
#Will eventually need to see which ones are relevant
```

Again since I there is no dataset present in this PDF, normally by just taking the length(udest) from result of using the unique function on the **data$Destination** .

Protocols Usage

To see a general list of which protocols were used we would also want to do a unique or summary of the protocols used.

> proto<-summary(data$Protocol)

```
#Display of Protocols and Count and Frequency
#We add the frequency and round it for easier workability.
proto <-mutate(proto,freq = round(Count/66524.15,digits=3))
proto
```

```
##        Protocol  Count  freq
## 1           ARP  76420 1.149
## 2       BROWSER     74 0.001
## 3        CAT-TP      1 0.000
## 4  CLASSIC-STUN  23381 0.351
## 5        DCERPC      1 0.000
## 6       DCP-PFT      9 0.000
## 7          DHCP     13 0.000
## 8        DHCPv6     12 0.000
## 9           DIS      2 0.000
## 10           DNS    2588  0.039
## 11           FTP      56  0.001
## 12      FTP-DATA       4  0.000
## 13          GVSP 2555242 38.411
## 14          HTTP   48429  0.728
## 15      HTTP/XML     588  0.009
## 16          ICMP      55  0.001
## 17        ICMPv6    4097  0.062
## 18        IGMPv3     334  0.005
## 19          IPv6      90  0.001
## 20          KNET       2  0.000
## 21        LANMAN      36  0.001
## 22           LLC       2  0.000
## 23          LLDP      22  0.000
## 24         LLMNR     920  0.014
## 25          MDNS      86  0.001
## 26           MP4       1  0.000
## 27          NBNS     253  0.004
## 28          NBSS      36  0.001
## 29          OCSP       7  0.000
## 30      Pathport       5  0.000
## 31          PNRP    1026  0.015
## 32         QUAKE       2  0.000
## 33        QUAKE2       1  0.000
## 34          QUIC    5274  0.079
## 35          RTCP    2002  0.030
## 36       SIGCOMP       2  0.000
## 37           SMB     117  0.002
## 38          SMB2      27  0.000
## 39          SNMP     323  0.005
## 40          SSDP    5056  0.076
## 41           SSL       3  0.000
## 42           TCP 3484267 52.376
## 43        TLSv1     451  0.007
## 44       TLSv1.2   16444  0.247
## 45         UAUDP       2  0.000
## 46           UDP  424644  6.383
## 47           XID       8  0.000
```

```
#Note: it is possible to achieve similar data using verisr's getenum
#although format will be off.
```

There are 47 different Protocols found within the capture. Altough the information regarding frequency was rounded to 3 digits pass the decimal, we still get a good picture of the overall results. The Protocol **GVSP** which is neither one of the two standard transport protocols **TCP** and **UDP** had a surprising amount of packets. Upon looking this up, it would be appropriate because **GVSP** is used for video streaming or VOIP using **UDP** packets.

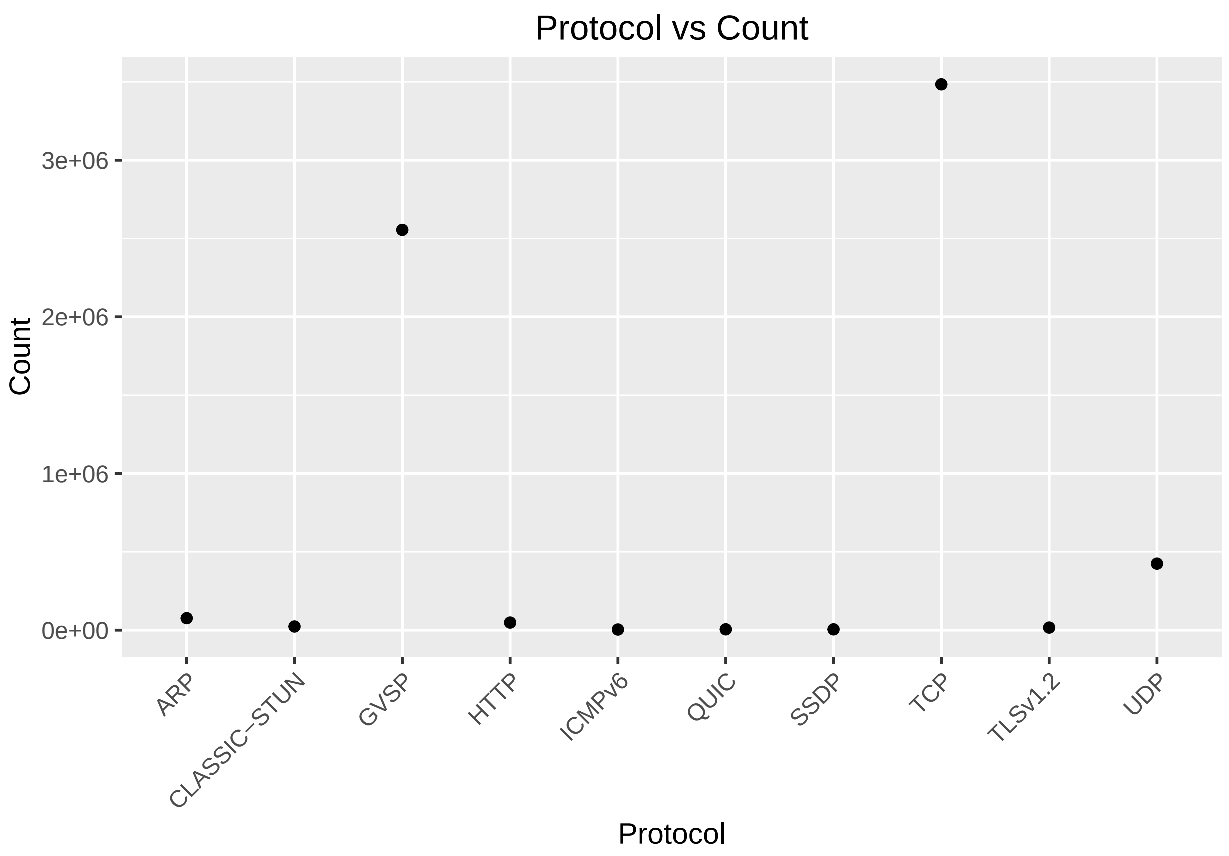
Visualizing & Viewing Transformation

Graphs

In this section we develop our intial graphics for the data that has been processed attempting to seperate fields by use of graphics for easier readbility and also ensure that these graphs do not overutilize resources wtihin the system. We will also try to answer some of the questions posed in the first section.

A

```r
#Graph using freq
#Seperate top 10 Protocols
proto1<-top_n(proto,10,Count)
qplot(Protocol,Count,data=proto1)+theme(axis.text.x = element_text(angle = 45, hjust = 1))+ggtitle("Pro
```

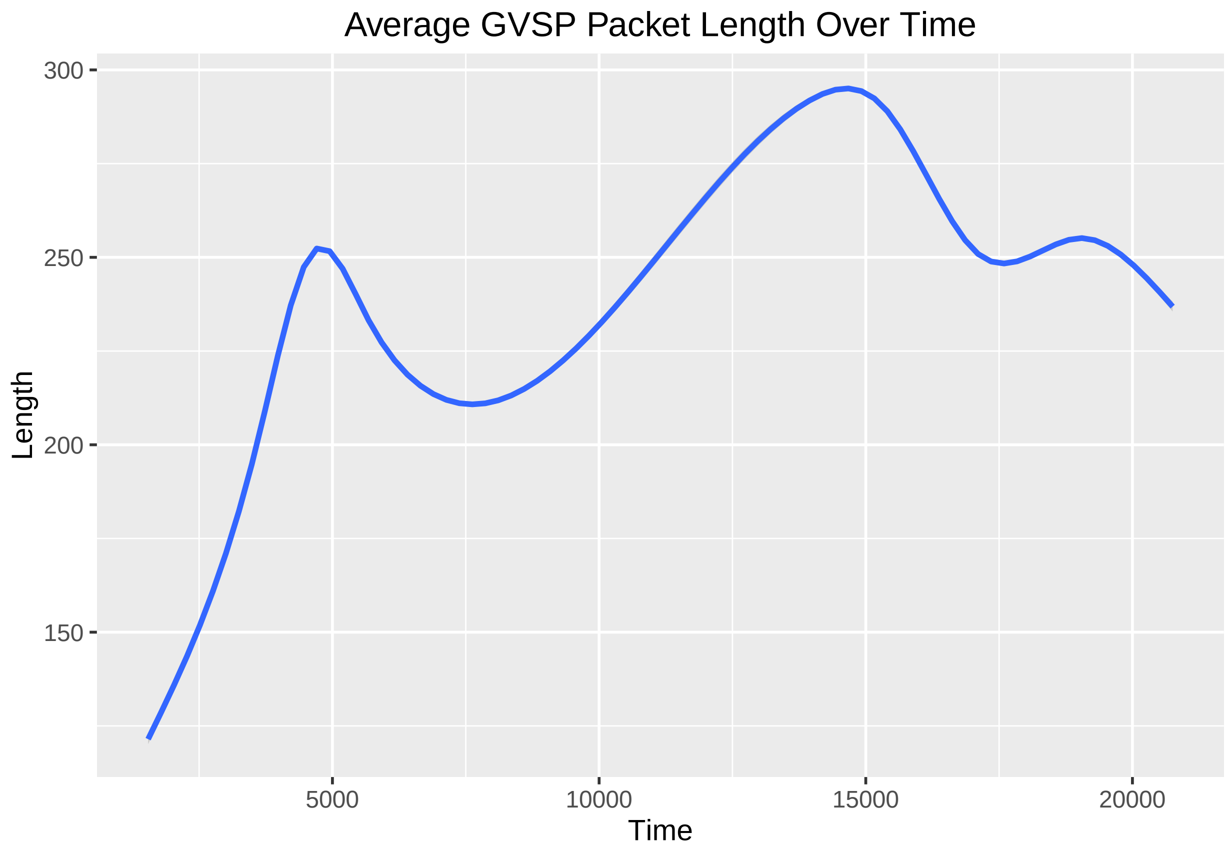


That was just an overview of which protocols were most often used during the session, now we begin to look at the stand out protocol for me which was **GVSP**

gd <-select(data,Time,Protocol)
gd <-filter(gd,Protocol==“GVSP”)

B

```
g1<- ggplot(gd,aes(Time,Length))
g1<- g1+geom_smooth()
g1 <-g1+labs(title="Average GVSP Packet Length Over Time",xlabs="Time (ms)")
print(g1)
```



With approximately 38% or 2.3million packets, it would very intensive on any system to plot this information and convey it in a PDF. This graph plots the average length over time of GSVP packets, and is a rough approximation of how the application was sending and receiving packets. If this were plotted in either **geom_dotplot** or **geom_jitter** it would nearly impossible to load.

C

by_time <- group_by(data,Protocol,Destination)
sumtime<- summarise (by_time)

Now we will comb through the specific IPs used and the associated protocol that was the destination of the communication. By performing a group_by function we can seperate the data by the fields we want. In this case I want to find out all the **HTTP** Destination IPs.

```
onlyhttp<-filter(sumtime,Protocol=="HTTP")
head(onlyhttp)
```

```
##   Protocol    Destination
## 1     HTTP  104.16.26.216
## 2     HTTP  104.193.82.53
## 3     HTTP   104.20.45.97
## 4     HTTP   104.20.46.97
## 5     HTTP  104.23.129.82
## 6     HTTP 104.238.216.36
```

Now we can use our **freegeoip** function to tack on a Longitude and Latitude to each IP, after making some modifications to the data frame.

> dest1 <- select(data,Protocol,Destination)
> dest1 <-filter(dest1,Protocol==“HTTP”)
> dest2 <-unique(dest1$Destination)
> final<-freegeoip(dest3)
> world<-filter(final,latitude!=0 & longitude!=0)

Getting a sample of the cleaned IPs, that now have geo-location

```
head(world)
```

```
##                ip country_code  country_name region_code   region_name
## 1 206.248.168.136           CA        Canada                          
## 2      72.21.91.8           US United States          CA    California
## 3  198.41.214.187           US United States          CA    California
## 4   23.76.117.127           US United States          MA Massachusetts
## 5 206.248.168.177           CA        Canada                          
## 6   184.86.40.154           US United States          MA Massachusetts
##            city zip_code           time_zone latitude longitude metro_code
## 1                                            43.6425  -79.3873          0
## 2  Santa Monica    90405 America/Los_Angeles  34.0119 -118.4682        803
## 3 San Francisco    94107 America/Los_Angeles  37.7697 -122.3933        807
## 4     Cambridge    02142    America/New_York  42.3626  -71.0843        506
## 5                                            43.6425  -79.3873          0
## 6     Cambridge    02142    America/New_York  42.3626  -71.0843        506
```

```
library(rworldmap)
```

```
## Loading required package: sp
```

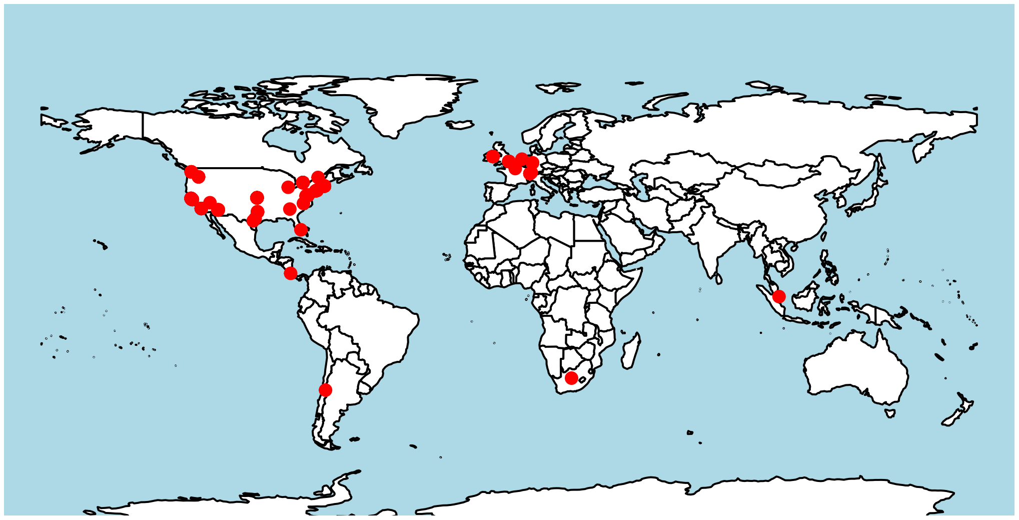
```
## ### Welcome to rworldmap ###
```

```
## For a short introduction type :   vignette('rworldmap')
```

```
map <- getMap(resolution = "HIGH")
```

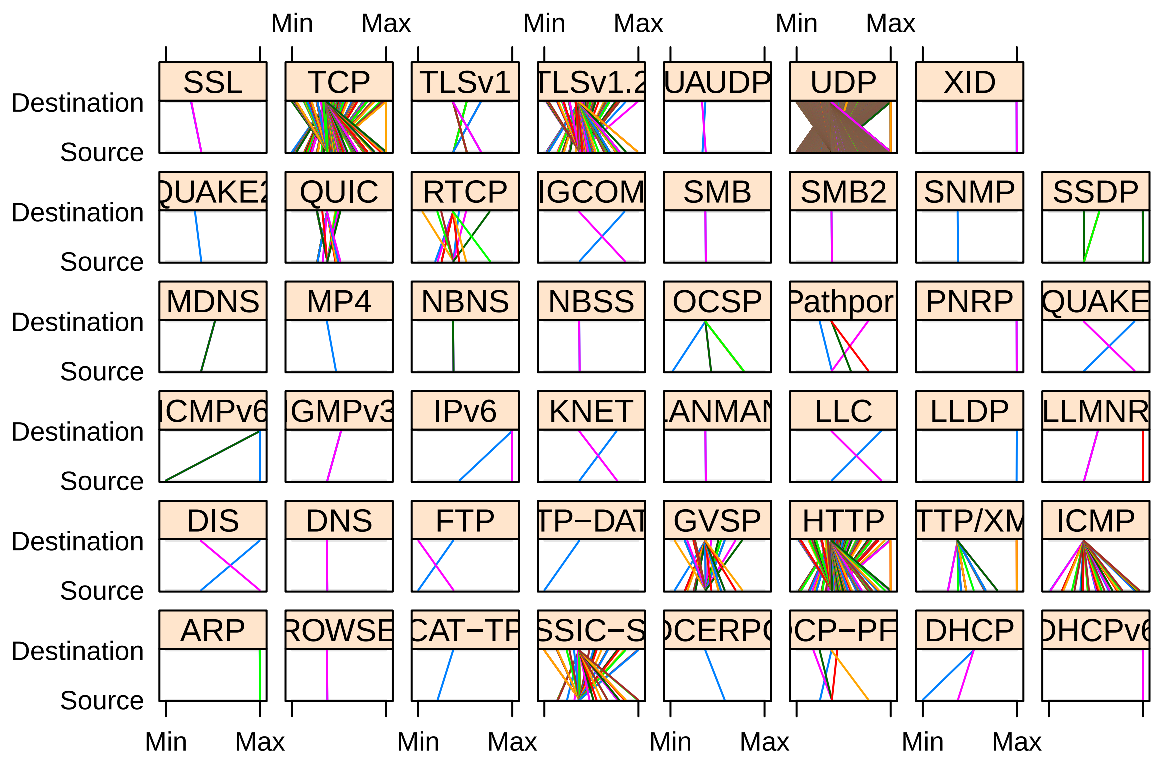
```
## Warning in getMap(resolution = "HIGH"): resolution should be set to one of :coarse low less islands
## setting to coarse as default
```

```r
plot(map, bg="lightblue", col="white", ylim=c(-60, 90), mar=c(0,0,0,0))
points(world$longitude, world$latitude, col="red", cex=0.8,pch=19)
```



This plot of the Destination IPs and their relative location to the world shows where the **HTTP** packets travelled to. The majority resign in either the east or west coast of North America, however there were connections made to 5 continents in total (minus Antartica, and Australia). We get a good idea of where web servers are located. ###D Attempting to show the stream of data depicting the relationship of Source and Destination IP grouped by Protocol

```r
parallelplot(~extratime[1:2] | extratime$Protocol)
```



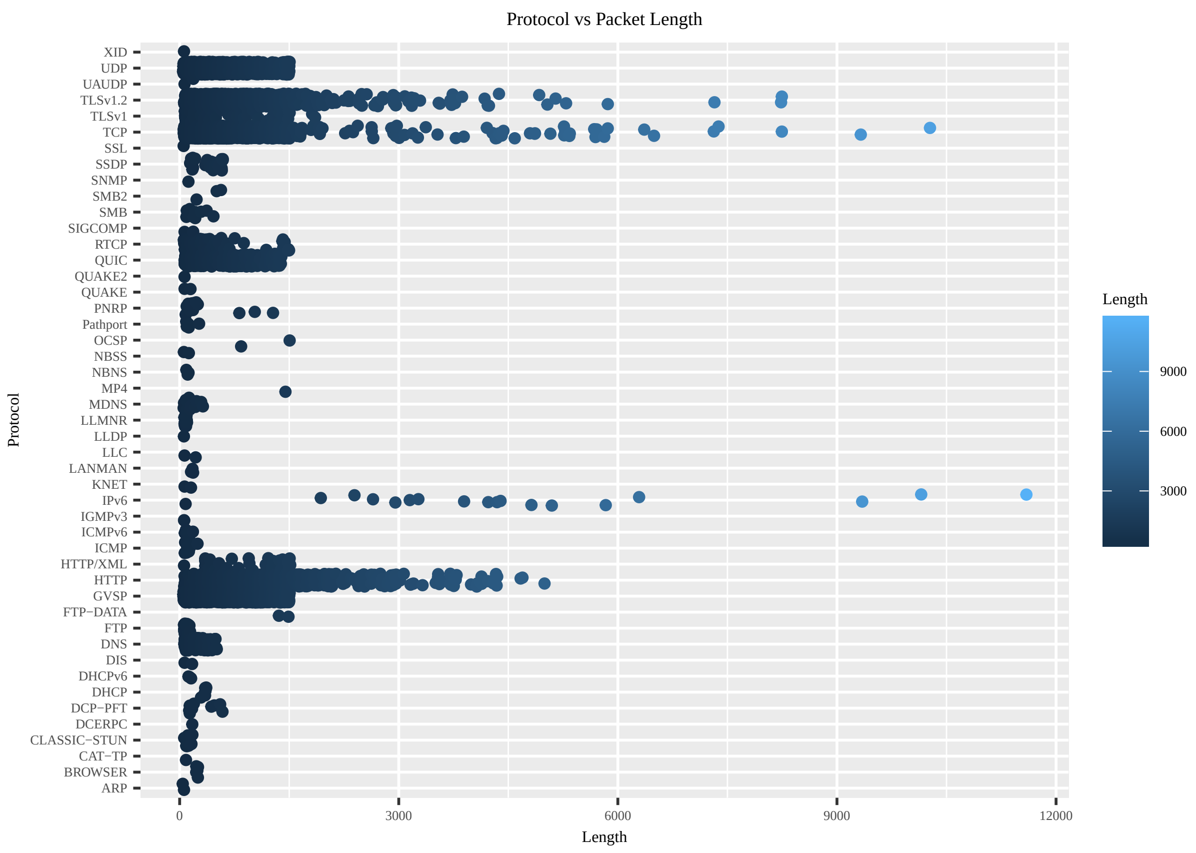
The column after the pipe adds the *facet* feature to the graph and allows us to graph each relationship in regards to its protocol. It is very clear and vivid in the graphs of **DHCP** and **OCSP**, where you can see that all the various source IPs are connected to the same destination IP.

E

Finally, in the last demonstration, we take a look at the packet length in comparison to the protocol used.

```r
ss<-ggplot(vers,aes(Length,Protocol))
ss<- ss+geom_jitter(aes(color = Length))
ss<-ss+geom_smooth()
ss<- ss+theme(text=element_text(family="Times", size=6))
ss <-ss+labs(title="Protocol vs Packet Length")
print(ss)
```

```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```



Although, there are flags for error when using **geom_smooth**, with the addition of it, the graph dots from the different rows are not as tighly knit together, making it easier to distinguish the lengths of each packet within a protocol.

Conclusion and Final Thoughts

Apart from graphing the information, I tried very hard to keep this PDF as minimal resource-intensive as possible. I knew that using certain plots would drastically slow down the rendering of this PDF when opened. Although my dataset was quite large, I believe this method of reading smaller tables and files (usually hidden within R markdown) made it so the file was much quicker to access. Compiling with knitr itself was quite quick with times of under 30seconds. I will try to upload this, and view it on a different computer to check my hypothesis.

I believe i was able to answer the questions that were made in the first section; However I found it quite difficult to map out frequency for all protocols throughout the day. With a bit more work, I can improve on visual portions of the graphs but the constraint that is the dataset itself was ever-present within the work. Normally loading the dataset using **read.csv()** would take roughly 5~10mins and graphing anything related to dots would usually throw out an error. Although this is a first look for myself and big data, I think the data represented covered some of the topics and also things I found interesting.

The lack of a duration column within this dataset has got me thinking. With it and with such a large dataset we can use the duration to figure out on average how fast it takes for a protocol to function.

Notes

This **.rmd** file will most likely be up on my GitHub It can found at the following link