Decoding Anonymized Packet Data

Finding useful patterns and trends in anonymized packet data.

Shawn McCaffrey
Shrikant Pandhare
Paul Pasquale
David Rennicker

11/19/2008

Contents

Introduction 3

Part I- Population Profiling Tools and Techniques 4

  Summary 6

Part II- Media Profiles 7

  Video on Demand Services 8

    YouTube and YouTube HD 8

    Hulu Standard and High Definition 10

    Discovery Channel 13

  Live Video Services 16

    CNN Live Video 16

  Audio on Demand Services 18

    Pandora 18

  Live Audio Services 22

    Radio Paradise 22

    CNN Live Radio 23

Part III- Applying Profiling Techniques Learned to Flow Data 26

  Summary Data 26

  Application to Real Flow Data 29

  Conclusion and Enhancements to SiLK Tools 33

Introduction

Researchers studying Internet traffic face two significant problems when attempting to learn more about their populations. First, IP addresses are frequently anonymized so that a researcher does not know the IP address of the user or the server. This is a problem because the researcher cannot visit the IP, or research the IP addresses being mentioned to determine what services are being accessed, or what the profile of the users are like accessing the services. The second problem is that companies many times to save space, and reduce overhead, capture packets in a reduced form. One such form is flow data, which is essentially a summarization and aggregation of packet traffic. A second reduced form is choosing not to capture the payload section of the packet. This method preserves the individual packets, but it just drops the data section or replaces it with random data to either save space, or increase privacy. This once again creates the problem that it is harder for the researchers to know anything about the population being studied, or the services being accessed.

Through this report we intend to develop techniques for demystifying anonymous packet capture data. The first section of the report will talk about ways that a researcher can determine information about a population even with limited knowledge from packet capture data. The second section of the report will then talk about ways that a researcher can try and determine what services a user may be accessing based solely on the shape of the traffic flows. This will allow a researcher, if using data with anonymized IP addresses, to still be able to have an educated guess on what service is being accessed by a user. For the purposes of this report we will be focusing on media services, specifically play on demand audio and video, as well as live audio and video. The third section of the report will then show our attempts to use characteristics discovered in part two of the report, to try and find media services in flow data, and the challenges that this presents.

Part I- Population Profiling Tools and Techniques

A useful piece of data that a researcher would be interested in obtaining would be the operating system (OS) of systems involved in a data flow. Since the data the researcher is using is either anonymized, or is flow data it makes it hard for the researcher to actively determine the OS, therefore passive OS fingerprinting techniques must be used. Passive OS fingerprinting can be achieved based on various factors and with different levels of accuracy depending on the data available. Some OS fingerprinting can be achieved by flow data, such as observing the ephemeral ports used when a host connects to network services. More precise OS fingerprinting is done by examining packet header data such as ephemeral ports, window size, max segment size, TTL, packet size, and so on – this information is usually included in full packet captures. Applications such as p0f (a passive OS fingerprinting tool) attempt to examine this information in order to fingerprint a device's operating system. It is important to note that nmap performs active OS fingerprinting and therefore is not a valid tool to use in our implementation. Our passive techniques involve examining data from past or current connections by doing nothing other than observing traffic. On the other hand, active fingerprinting involves sending packets to a destination node and observing the response.

When fingerprinting with flow data, the only real indicator is ephemeral source port. Although an operating system identity can't be established with 100% confidence from this information alone, it can be established at a high level of confidence. Different operating systems use different ephemeral port ranges. Ephemeral ports are the temporary client-side ports used when opening up a TCP connection and the range of ephemeral ports are usually a default setting in the operating system's TCP stack. The client will start assigning ephemeral ports for connections at a certain number, and increase the port number by 1 for each new connection. When the port is incremented past the highest number in the range, it will reset to the lowest number once again and the next port assigned will be at that number. Therefore, by observing the traffic from a certain host for long enough, the ephemeral port range of that host can be established, and the operating system could theoretically be determined.

Ephemeral port ranges for the most popular operating systems are as follows[1]:

Windows XP and Previous: 1024 – 4999

Windows Vista and Server 2008: 49152- 65535[2]

[1] http://www.ncftp.com/ncftpd/doc/misc/ephmeral_ports.html

Apple Mac OS: 49152 - 65535

Linux (2.4 kernel): 32768 – 61000

Free BSD: 1024 – 65535 (assigned randomly, does not follow the linear pattern)

Solaris: 32768 – 65535

Although there is some overlapping of ranges, we can look at their boundary limits in order to establish the OS fingerprint. For example, a host that starts its range at 1024 can be Windows or Free BSD, but a host that doesn't exhibit a port higher than 4999 is guaranteed to be a Windows machine.

This method is not foolproof. Many operating systems provide mechanisms for manually changing the ephemeral port range, and smart administrators that don't want their machines to be profiled in this way may change the setting, therefore invalidating this mechanism. On the other hand, the majority of machines on the network probably haven't had this "tweaking" performed, so results on a large scale should be fairly accurate.

When fingerprinting packet data, there are some fields within the IP and TCP packet can be examined when determining the client system's OS. These fields include window size (WSS), initial TTL, maximum segment size (MSS), and NOPs. Of course, we can also examine ephemeral port data here as well. Data from these different fields, when combined, can help determine the operating system. In fact, an application entitled p0f (available at http://lcamtuf.coredump.cx/p0f.shtml) keeps a database of fingerprints for many operating systems, and assigns probabilities by hosts. The input to this application is a packet capture (pcap) file.

The most important element of fingerprinting by packet is examining the first few packets of any flow. This is where the client and server are "handshaking" via the TCP protocol, and are exchanging their TCP capabilities. While the server might be able to sustain a packet size of 64, the client may prefer a packet size of 60, and so all future packets from the server may only be of size 60. This event is very significant as this packet size could be linked to a specific operating system, but the important issue is which machine *initially* sets an option in this manner, not which machines use this option later on; the server may choose to drop down to size 60 to better communicate with the client).

[2] http://blogs.msdn.com/drnick/archive/2008/09/19/ephemeral-port-limits.aspx

Summary

The following techniques are available as methods of determining the operating system from a restricted data set:

When the data set consists of packet flow captures using the “traditional” settings with anonymized IP addresses that remain consistent in the data set

- Monitoring ephemeral port connects from an IP addresses, ideally determining the minimum and maximum value used.

When the data set consists of individual packet captures, with the data field of the packet dropped or filled with random data, anonymized IP addresses that remain consistent in the data set

- Monitoring ephemeral port connects from an IP addresses, ideally determining the minimum and maximum value used.
- Possible use of tools such as p0f which looks at a variety of characteristics of the packet and compares them to known signatures.

Part II- Media Profiles

Using the techniques described in Part I of the paper can give researchers information on the population of the network, specifically the OS. However none of these techniques give any information on what is being services are being accessed by the users of the network. In this section, we investigate a variety of media services, and analyze the traffic generated by visiting the services to determine if there are any distinctive patterns which give away the presence that a specified media service is being used, that a researcher could then compare traffic flows to try and determine what a specific flow represents. These patterns are most useful when looking at anonymized IP packet capture data, since their generation is based off of the size of individual packets, so looking at flow data would not have as much utility, since flow data is the summation and aggregation of these individual packets.

The following represent the result of the research into a selection of media services. The tests were conducted over high bandwidth connections (6Mbps cable, or campus internet connections), and the home IP address, and the IP address of the service that is being connected to is clearly marked. The tests were repeated to ensure results were not because of latency or network congestion errors. The graphs, when appropriate are expressed on a 60 second scale, except when the connection was less than 60 seconds, or the graph needed to show a longer scale to illustrate a point.

Video on Demand Services

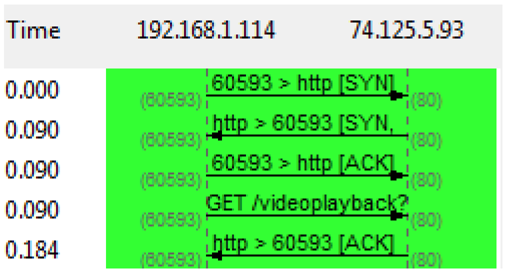
Video on demand services are services, which provide videos to the user by them specifically picking a video and having the video play.

YouTube and YouTube HD

YouTube and YouTube HD are encoding in the H.264 codec, with standard quality having a resolution of 320x240 and 22 KHz audio and high quality having a resolution of 448x336 and 44 KHz audio quality[3]. To analyze the YouTube traffic, we used a 0:24 clip that was available in both YouTube regular quality as well as YouTube HD[4].

Traffic Analysis

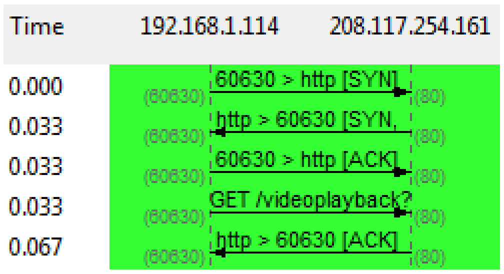
Home Address **Server Address**



```
Time    192.168.1.114          74.125.5.93
0.000   (60593)  60593 > http [SYN]     (80)
0.090   (60593)  http > 60593 [SYN,     (80)
0.090   (60593)  60593 > http [ACK]     (80)
0.090   (60593)  GET /videoplayback?    (80)
0.184   (60593)  http > 60593 [ACK]     (80)
```

YouTube Partial TCP Stream

Home Address **Server Address**



```
Time    192.168.1.114          208.117.254.161
0.000   (60630)  60630 > http [SYN]     (80)
0.033   (60630)  http > 60630 [SYN,     (80)
0.033   (60630)  60630 > http [ACK]     (80)
0.033   (60630)  GET /videoplayback?    (80)
0.067   (60630)  http > 60630 [ACK]     (80)
```

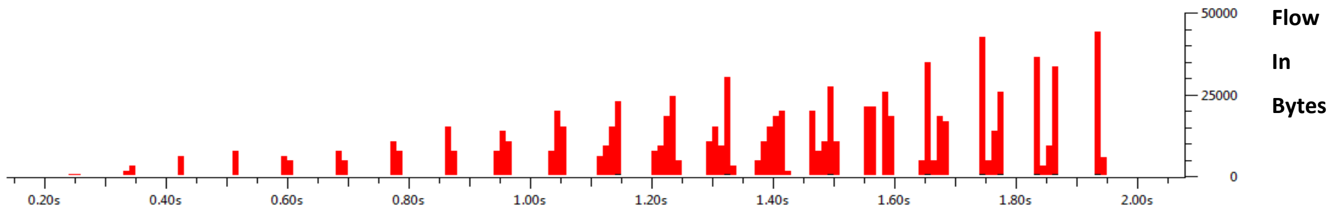
YouTube HD Partial TCP Stream

The first interesting thing to note is that the two services use completely different IP ranges, which were consistent throughout testing. Querying the ARIN[5] database, the results are that YouTube uses a 74.125.0.0/16 network block, where YouTube HD uses 208.117.224.0/19, which was registered three months after the YouTube block. This would suggest that currently HD media is not nearly as prevalent on YouTube, since with the smaller network block it restricts the number of possible servers that could be used to stream the media. This is something to note, as since HD content becomes more prevalent, YouTube will need to add additional capacity, which may use a different IP range, so if a network administrator were to ban based on IP it would be logical to assume that the IP range for YouTube's HD service will be expanding. However in anonymized data, this difference would not be as obvious.

[3] http://googlesystem.blogspot.com/2008/03/youtube-tests-higher-resolution-videos.html
[4] http://www.youtube.com/watch?v=3uftByymXNg
[5] http://ws.arin.net

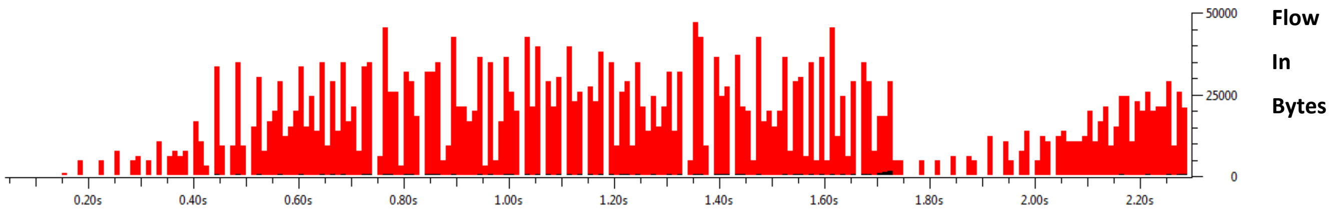
The YouTube service itself uses HTTP over port 80 to send the Flash movie. Unfortunately, for profiling purposes there is nothing interesting in the TCP stream going on, that would differentiate the stream from any other HTTP traffic.

The next part of the analysis is to analyze what do the traffic profiles look like for these two services, and how do they compare to each other.



Red- Incoming to Client Black-Outbound from Client

YouTube



Red- Incoming to Client Black-Outbound from Client

YouTube HD

The first thing illustrated by these graphs is the lack of data sent by the client. In both cases, the client bars are negligible. That is because the client is sending almost no data in this conversation. The three main things that are causing the client to send data are:

- TCP incorrect checksum errors, commonly caused when the network adapter is responsible for calculating the checksum value instead of the CPU and since the capture is done before the packet reaches the network adapter, the checksum will be incorrect[6].
- TCP duplicate ACKs received, which can be caused by TCP segments going missing, or being delayed due to congestion[7].

[6] http://wiki.wireshark.org/TCP_checksum_offload
[7] http://condor.depaul.edu/~jkristof/technotes/tcp.html

- The GET request from the browser, asking to load the webpage for the video.

The second thing illustrated by the graphs, and that which would be useful for profiling, is the very distinctive shapes both graphs have, and the transmission patterns. The YouTube stream sent 884 kilobytes of data to the user, where as the YouTube HD stream sent 3.3 megabytes. However, despite the almost four times size difference, the difference in time it took for the conversation to complete between the two services was minimal, 2.02 seconds for the YouTube transmission, compared to 2.3 seconds for the YouTube HD transmission, a 14% difference to send 3.73 times as much data. This anomaly can be explained by looking at the network flow graphs. YouTube seems to send chunks of data waiting almost as much as .1seconds between transmissions and the transmissions can easily be broken down into discrete bursts. However, the YouTube HD transmission is an almost continuous transfer, with few areas of no transmission. While there is less activity between 1.7 seconds and 2.0 seconds, this corresponds with an increase in the client sending data, the majority of which were TCP duplicate ACK packets, which could indicate network congestion. These two very distinctive shapes provide a useful visual identification pattern for the YouTube streams. Since the file is being sent over regular port 80, as file delivery, the difference in transmission speed must be being regulated by the server, and not the protocol.

Summary and tools for identification

- Both YouTube and YouTube HD are served over TCP port 80.
- YouTube flows are characterized by the video being completely buffered well before the end of the clip (assuming available bandwidth); the movie is transferred as opposed to streamed.
- YouTube and YouTube HD flows are differentiated in that YouTube HD has a much higher rate of transmission meaning that it takes almost the same amount of time to fully load either a YouTube or YouTube HD clip.

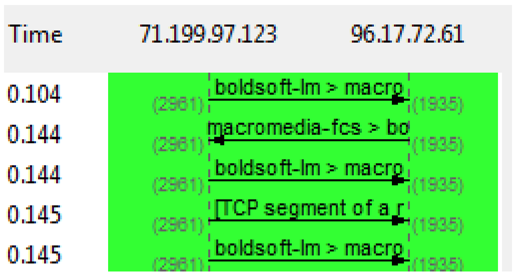
Hulu Standard and High Definition

Hulu standard definition video is encoded using the VP6 codec, at a resolution of 640x360 while high definition video is provided at 720x1280 resolution using the H.264 codec[8]. To analyze Hulu traffic we used a 1:00 clip from *Quantum of Solace* on Hulu, and Hulu HD.

[8] http://www.streamingmedia.com/article.asp?id=10114&page=2&c=31

Traffic Analysis

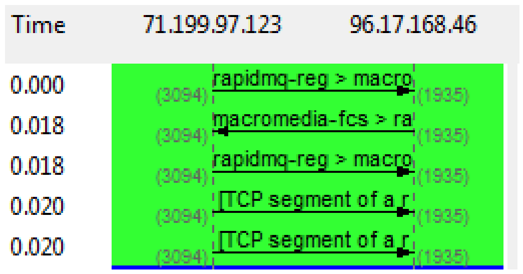
Home Address **Server Address**



```
Time    71.199.97.123          96.17.72.61
0.104   (2961) boldsoft-lm > macro (1935)
0.144   (2961) macromedia-fcs > bo (1935)
0.144   (2961) boldsoft-lm > macro (1935)
0.145   (2961) [TCP segment of a r (1935)
0.145   (2961) boldsoft-lm > macro (1935)
```

Hulu Partial TCP Stream

Home Address **Server Address**

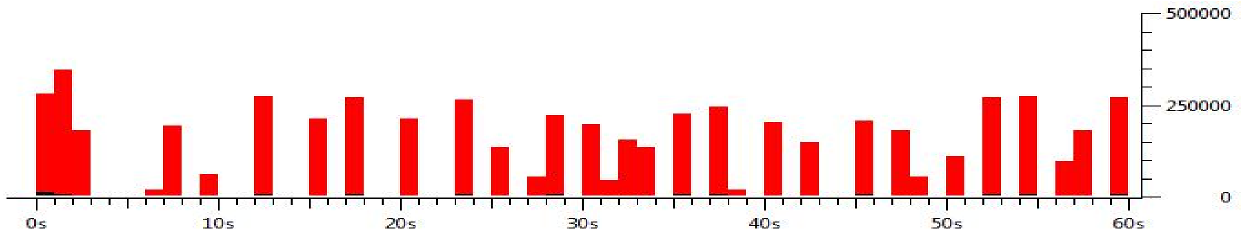


```
Time    71.199.97.123          96.17.168.46
0.000   (3094) rapidmq-reg > macro (1935)
0.018   (3094) macromedia-fcs > ra (1935)
0.018   (3094) rapidmq-reg > macro (1935)
0.020   (3094) [TCP segment of a r (1935)
0.020   (3094) [TCP segment of a r (1935)
```

Hulu HD Partial TCP Stream

Looking at the stream, there are two significant differences with Hulu, compared to YouTube. First, the server is sending data to the client using port 1935, as opposed to port 80, which was used by YouTube. The significance of this is that the Hulu player is Macromedia Shockwave based, so as a result the server uses a dedicated port (1935) to send the data to the clients.

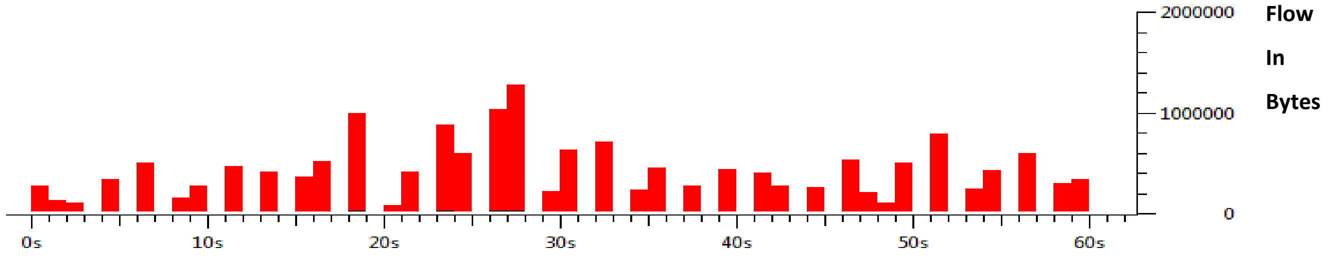
The second difference is that Hulu appears to house both the standard and HD content on the same 96.17.x.x subnet. However, cross-referencing the information with ARIN, reports the 96.16.0.0/15 block is owned by Akamai, a company that provides value by allowing content hosts to locate themselves physically closer to the end user through. This is an important distinction for both network administrators and researchers; whereas 208.117.224.0/19 is definitively owned by YouTube, so blocking access to that range of IPs, or seeing traffic from that range of IPs would be a positive indicator of YouTube traffic, seeing traffic from the 96.16.0.0/15 block is not always a positive indicator of Hulu traffic as Akamai could be hosting other content on IP addresses within this block. Once again, when using anonymized data these nuances in hosting and providing will not be evident.



Flow In Bytes

Red- Incoming to Client **Black-Outbound from Client**

Hulu



Red- Incoming to Client Black-Outbound from Client

Hulu HD

The data flow graphs are mostly unremarkable. The Hulu and Hulu HD streams appear to be transmitting at fairly the same pattern of a burst, a rest, and then a burst again. The main difference between the two is that Hulu HD has a much greater magnitude of transmission (2.2 Mbps), compared to the Hulu standard (.75 Mbps).

Comparing this graph to the YouTube graph, there is one marked difference. First, with the YouTube stream, YouTube managed to finish downloading the entire 23 second clip in approx 2.2 seconds for both the standard and HD version. With Hulu, the 60 second clip continued to be downloaded for the duration of watching. This is because Shockwave is a streaming protocol, over port 1935, where as YouTube uses file delivery, HTTP over port 80. So, with HTTP it delivers a file of fixed length, trying to serve it as quick as possible, unless the server is using an artificial (outside the protocol) method for constricting bandwidth. However, the Flash protocol provides for the server setting the outgoing transmission rate, so delivery of content will continue at this rate until one side stops the connection.

Summary and tools for identification

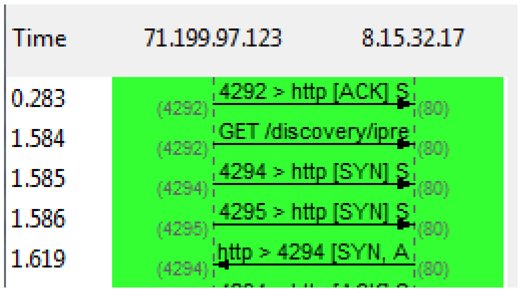
- Hulu and Hulu HD travel are served on port 1935, the port for Shockwave Flash, and the client uses a random ephemeral port which changes with every new connection to the website.
- Hulu and Hulu HD streams have an almost identical shape, differing in only the magnitude of the flows, and not the density as in YouTube vs. YouTube HD.
- Hulu prefers not to send the entire file at one time, and instead sends it incrementally as it plays.

Discovery Channel

Discovery Channel uses a player powered by Move Networks, using a proprietary streaming methodology, as well as a proprietary encoding mechanism[9]. Discovery only offers one type of stream, so no comparative analysis between streams is possible. The analysis is based off a study off the first 60 seconds of a 45 minute long television program.

Traffic Analysis

Home Address **Server Address**



Time	71.199.97.123		8.15.32.17
0.283	(4292)	4292 > http [ACK] S	(80)
1.584	(4292)	GET /discovery/ipre	(80)
1.585	(4294)	4294 > http [SYN] S	(80)
1.586	(4295)	4295 > http [SYN] S	(80)
1.619	(4294)	http > 4294 [SYN, A	(80)

Discovery Channel Partial Stream

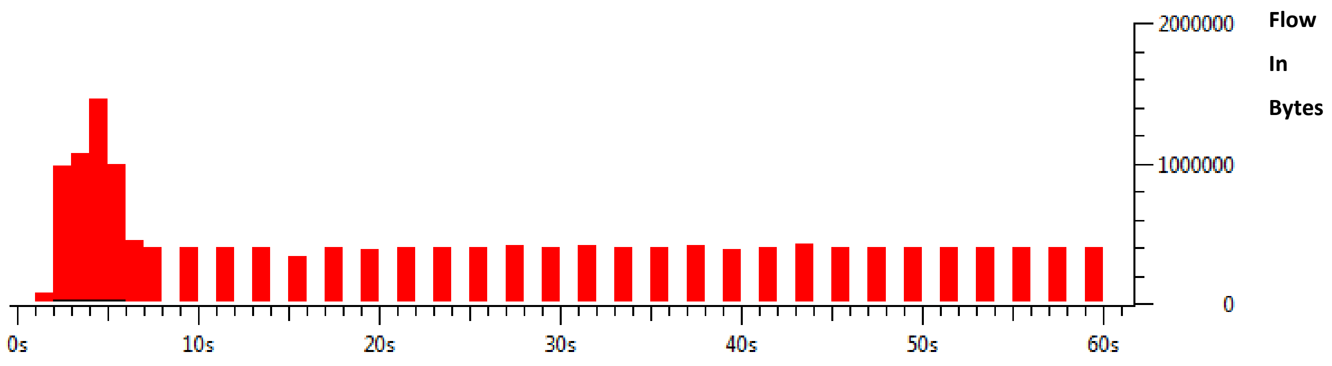
Looking at the partial stream graph, immediately something interesting is evident. The client appears to be talking to the server across three different ports (4292, 4294, 4295) which is unusual; however the server is serving the data over port 80, using well formed TCP. Looking at the TCP conversations, the server appears to be balancing the load across all three ports almost perfectly, but each of the three conversations are all use the same pair of IPs:

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A->B	Packets A<-B	Rel Start	Duration
71.199.97.123	4295	8.15.32.17	http	5912	6127776	1777	4135	1.585732	76.0053
71.199.97.123	4292	8.15.32.17	http	6300	6352283	1836	4464	0	77.435
71.199.97.123	4294	8.15.32.17	http	6747	6961128	2008	4739	1.585114	75.9248

The reason for this split might be due to the software Discovery uses to stream the videos. The player uses a mash up of Flash and Microsoft Silverlight and proprietary technology to stream video[10], all still

[9] http://www.movenetworks.com/move-media-services/move-simulcode
[10] http://www.richardleggett.co.uk/blog/index.php/2008/03/28/movemedia_silverlight_hd_video_player

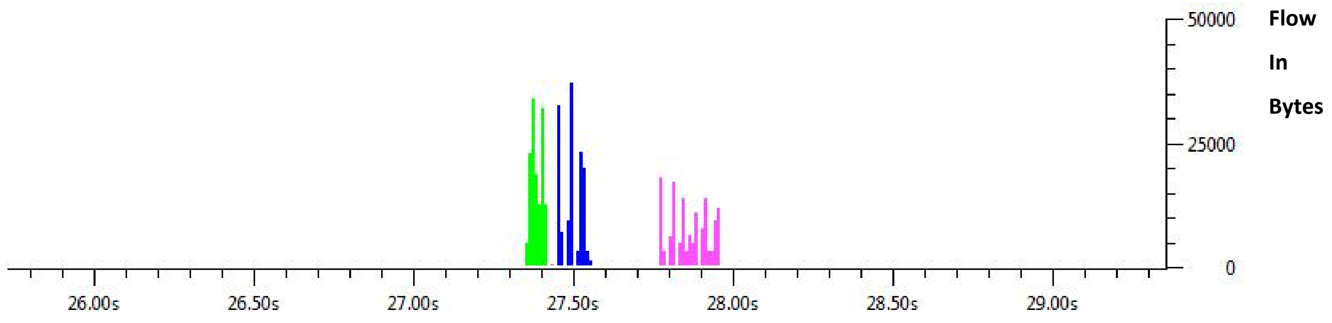
using port 80, and boasts of features called adaptive stream, which allows bandwidth to be instantly throttled depending on the connection.



Red- Total Incoming to Client Black-Outbound from Client

Discovery

The bandwidth pattern Discovery follows is front loaded, which is different compared to the other two services discussed thus far. Presumably, this is because the Discovery stream is buffering a few seconds of media before playing, so that a temporary network interruption won't cause the video to skip or pause. The media is then streamed at a very consistent and very predictable pattern. This pattern presumably would continue until the end of the media. The three ports that the media uses are all used during each transmission burst, and it appears that data is requested from them based on numerical port order as shown:



Green-Port 4292 Blue-Port 4294 Magenta-Port 4295

Close-up of a data transmission from the server to the client

Summary and tools for identification

- When using flow data, Discovery channel media can be found by looking for three flows, almost all simultaneously started, all going to the same IP address coming from the same IP address, with the server streaming the file on port 80 and the client using 3 close to consecutive ports, all of which have similar amounts of bytes transmitted.
- Discovery buffers a portion of the media for approximately five seconds before playing, which is represented by a very dense and bandwidth intensive initial traffic flow followed by groups of packets coming in at a very predictable and consistent pattern as the media plays.

Live Video Services

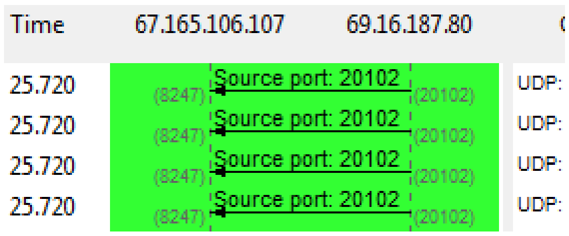
Live video services are those services which provide one live video stream that many users receive and watch, so that all users are watching the same stream.

CNN Live Video

CNN Live Video uses Flash to provide the video, as well as the Octoshape Grid Delivery Enhancement plug-in, which allows media to be streamed in a fashion similar "peer to peer" to provide availability and higher bandwidth[11]. This traffic is a 60 second sample from CNN Live, an Internet broadcast of live programming airing on CNN.

Traffic Analysis

Home Address **Server Address**



```
Time      67.165.106.107      69.16.187.80
25.720   (8247) Source port: 20102 (20102)   UDP:
25.720   (8247) Source port: 20102 (20102)   UDP:
25.720   (8247) Source port: 20102 (20102)   UDP:
25.720   (8247) Source port: 20102 (20102)   UDP:
```

CNN Video Live Partial Stream

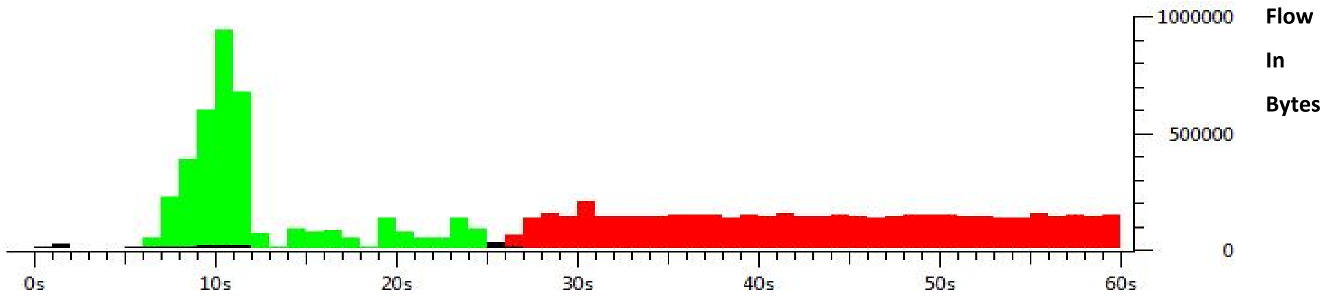
The traffic that is shown on CNN's website is much different than any traffic seen previously. CNN Video is the only service to use UDP for transmission, which does not guarantee reliability. This makes sense, since the video is live and time-efficiency is of maximum importance, so the overhead of TCP would be a hindrance.

A second area of interest is that there are separate streams being used by CNN (not shown in partial stream). Both streams come from a source identified as Highwinds CDN Group, which provides similar services to Akamai. The first stream has duration of almost 25 seconds, and is present while there is a loading screen in the video player. The second stream begins after the loading screen ends, and is the live video feed itself:

[11] http://www.reuters.com/article/pressRelease/idUS125043+23-Sep-2008+PRN20080923

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A->B	Packets A<-B	Rel Start	Duration
69.16.187.53	554	67.165.106.107	8247	3797	3860936	2834	963	6.619096	19.748
69.16.187.80	20102	67.165.106.107	8247	12747	13087334	12321	426	25.666195	93.0088

Port 554 is the real-time streaming protocol (RTSP), but that doesn't seem to be extremely useful in diagnosis. However, two guesses can be made based on the available information. First, the video could be being buffered on the .53 address, and then switched over to the .80 address when the feed is activated as a form of load balancing. Alternatively, the "Loading" graphic and animation could be considered a movie and that is what is being streamed on .53, and then the actual show is streamed on .80.



Red-.53 Traffic Green-.80 Traffic Black-Outbound from client

CNN Live

This very distinctive and unique pattern of the spike in traffic, followed by the constant traffic being sent, into perpetuity. Would seem to be a very useful tool for researchers trying to profile this service in data, but there is an important caveat. Since the red and green traffic both have different source IPs, they would be different flow records in flow data; as a result this distinctive graph will not appear as a flow record, instead there will be two separate graphs.

Summary and tools for identification

- CNN Live uses UDP to transmit packets.
- An initial short(less than 6 seconds) and high byte volume occurs from one IP address which is then followed by relatively low traffic for 15 seconds. After this, the IP address switches and there is a constant stream of traffic at about 100,000bytes per second.

Audio on Demand Services

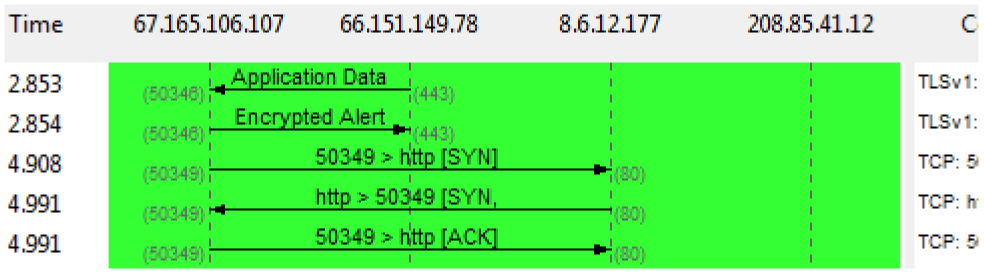
Audio on demand services are services which provide streaming audio to users based on pre-recorded content they pick, so many users may all be requesting different media from one server.

Pandora

Pandora is an Internet radio on demand service, which picks a song based on the preferences of the user, and then the user can choose to listen to the song, or veto the song, which then causes the service to pick another song based on the updated preferences of the user. The testing for this protocol was to listen to 30 seconds of a song, force the system to change songs by vetoing the current song, listen to a 2nd song, and then stop the capture 15 seconds into the 3rd song. Pandora streams at 128 kbps through a Flash application[12].

Traffic Analysis

Home Address	SSL Server	Music Server	Music Server 2

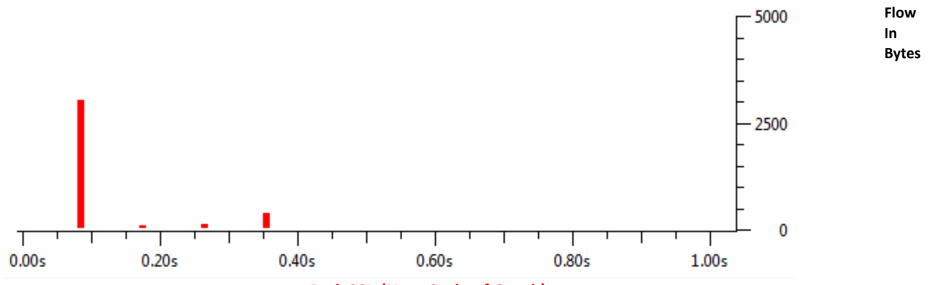

Time
67.165.106.107
66.151.149.78
8.6.12.177
208.85.41.12
C
2.853
(50346) Application Data (443)
TLSv1:
2.854
(50346) Encrypted Alert (443)
TLSv1:
4.908
(50349) 50349 > http [SYN] (80)
TCP: 5
4.991
(50349) http > 50349 [SYN, (80)
TCP: h
4.991
(50349) 50349 > http [ACK] (80)
TCP: 5

Pandora Partial Stream

The Pandora stream looks significantly different than any of the other streams encountered thus far. There are three unique IP addresses, not including the IP address of the client. The explanation for this can be illustrated by looking at the partial stream flow, as well as the data flow graph for Pandora.

The partial stream flow shows traffic on port 443, so it is immediately identifiable as SSL. Pandora requires credentials to log in, so this traffic is simply just the credential being automatically provided by the browser. Here, the SSL connection is being used as the control channel, for the actual audio stream which is then transmitted over HTTP on port 80.

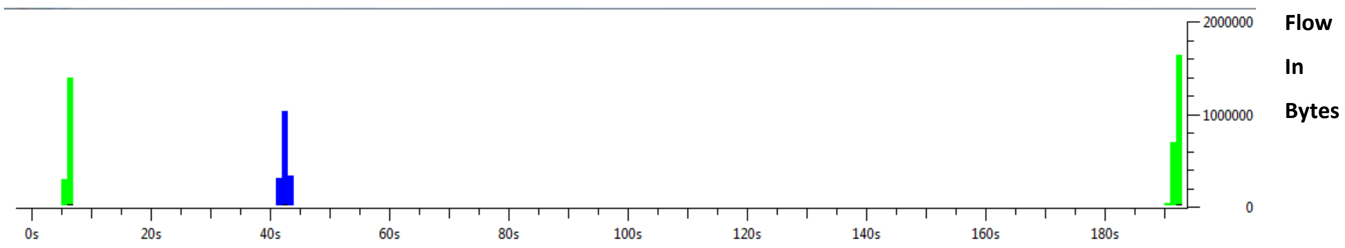
[12] http://blog.pandora.com/faq/



Red -SSL (Note Scale of Graph)

Pandora Zoom In

The next two IP addresses then are then able to be identified by looking at the full data flow graph for the Pandora service:

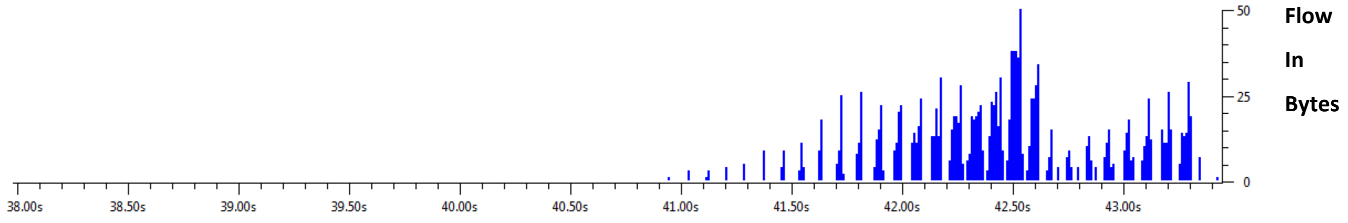


Red-SSL Green-Music Server 1 Blue-Music Server 2 Black-Outbound from client

Pandora

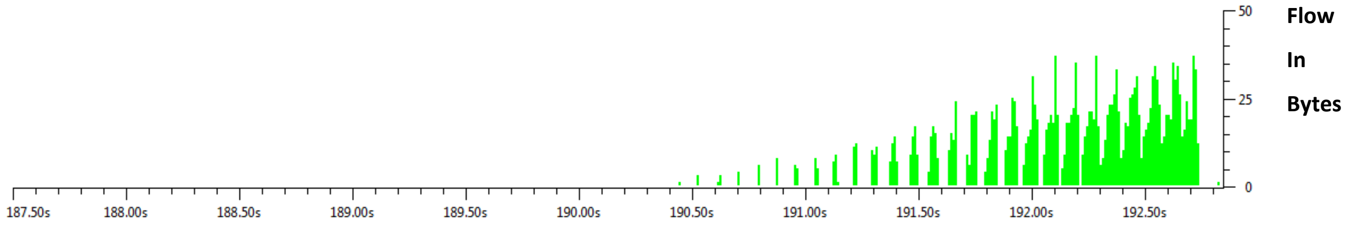
Since as part of the testing we know three songs were played, and we know at what times they were played, both the green and the blue IP addresses can be identified as music servers, where the green IP served the 1st and 3rd song, and the blue IP served the 2nd. This could be a method of load balancing, or some songs may be stored only on certain servers. Since Pandora does not let you pick songs specifically, and the software auto picks for you, it is impossible to duplicate this test with the same songs to see if the pattern holds. Important to note from a flow perspective, when the 1st song is buffered, it a SYN$\rightarrow$SYN/ACK sequence is established, and when the 1st song is finished buffering a FYN$\rightarrow$FYN/ACK sequence is sent. Therefore, even though songs 1 and 3 use the same ports, and the same IP addresses, they will be two separate flows since the connection is closed after each song is buffered fully.

Pandora appears to buffer the song fully at the start, and then not buffer again until the next song is selected. An interesting question is if there is a difference in packet flow between rejecting a current song and making the server pick a new one based on new criteria, or letting a song finish and then listening to the next song. The second case might have a slightly different flow, since the server might start to pre-buffer the next song as you approach the end of the currently playing song, on the assumption that you will not reject the song currently playing since it is almost over. Since the second song was the product of a rejection, and the third song just played after that song with no user intervention packet data can be examined:



Blue-Music Server 2

Pandora- Buffering of Song 2, Caused by Rejection of Previous Song



Green-Music Server 1

Pandora- Buffering of Song 3, Caused by Ending of Previous Song

Based on these two graphs, there appears to be little to no evidence of pre-buffering. Both songs started buffering when either the previous song ended by rejection or the song was over, and there appears to be no evidence of packets being sent early as a song became close to finishing.

Summary and tools for identification

- Pandora uses SSL to authenticate a user, and then transmits the audio over standard TCP port 80 to a random port on the client.
- The entire song is buffered before playing, usually in less than five seconds, on a sufficiently fast connection. The next song may or may not then originate from the same IP as the current song.

- When looking in packet flow records, a strategy could be to look for short(3 to 8 seconds) high volume flows 3 to 5 minutes apart going to the same IP address.

Live Audio Services

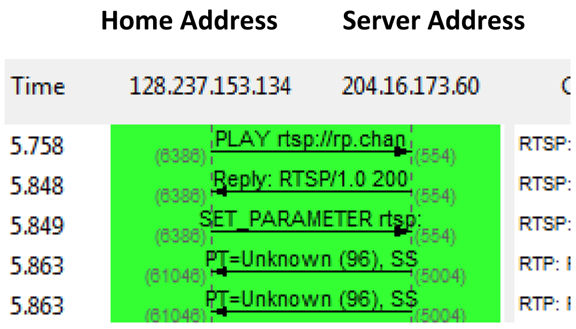
Live audio services are services which stream one feed to many users, so that all users are listening to the same live stream.

Radio Paradise

Background

Radio Paradise is a popular streaming radio station, where listeners tune into a stream and listen to a live radio feed, which all other users are listening to at the same time. For testing purposes, a 90 second feed of the stream was sampled. Radio Paradise plays using a 64Kbps audio stream, in a Windows Media applet.

Traffic Analysis

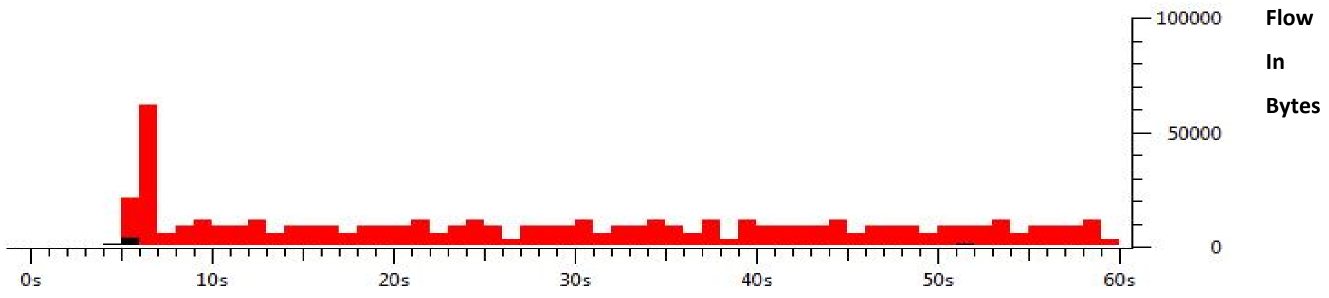


```
Time    128.237.153.134          204.16.173.60       C
5.758   (6386) PLAY rtsp://rp.chan          (554)   RTSP:
5.848   (6386) Reply: RTSP/1.0 200          (554)   RTSP:
5.849   (6386) SET_PARAMETER rtsp:          (554)   RTSP:
5.863   (61046) PT=Unknown (96), SS         (5004)  RTP: I
5.863   (61046) PT=Unknown (96), SS         (5004)  RTP: I
```

Radio Paradise Partial Stream

Radio Paradise is differentiated from other protocols seen thus far, as it uses RTSP and RTP as the transport protocols. RTSP is the protocol, which negotiates the connection and sets up preliminary parts of the connection, where as the RTP protocol is used for one-way connections to send live or stored streams[13].

[13] http://www.apple.com/quicktime/technologies/streaming/



Red- Incoming to Client Black-Outbound from Client

Radio Paradise

Radio Paradise's traffic flow exhibits a fairly standard pattern for streaming media, similar to CNN Live Video, with a large spike in the beginning, possibly where the protocol is being set up, and there is buffering of media, and then a fairly consistent low volume traffic flow for the duration.

Summary and tools for identification

- Radio Paradise initializes the connection using RTSP, and then transmits using RTP to a random port on the client.
- There is a quick spike at the beginning of the connection, followed by continuous transmission as the media plays.

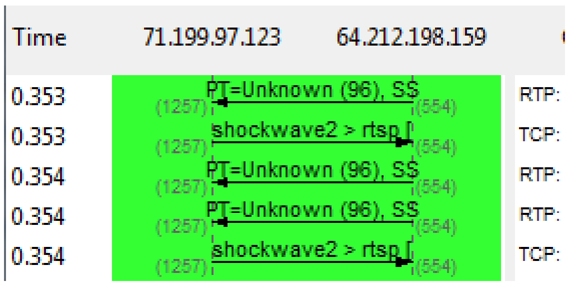
CNN Live Radio

Background Information

CNN Live Radio provides a live stream of radio provided by CNN. The radio is a 32Kbps stream, played through a Windows Media applet. The analysis was done based on a 90 second clip of the live radio stream.

Traffic Analysis

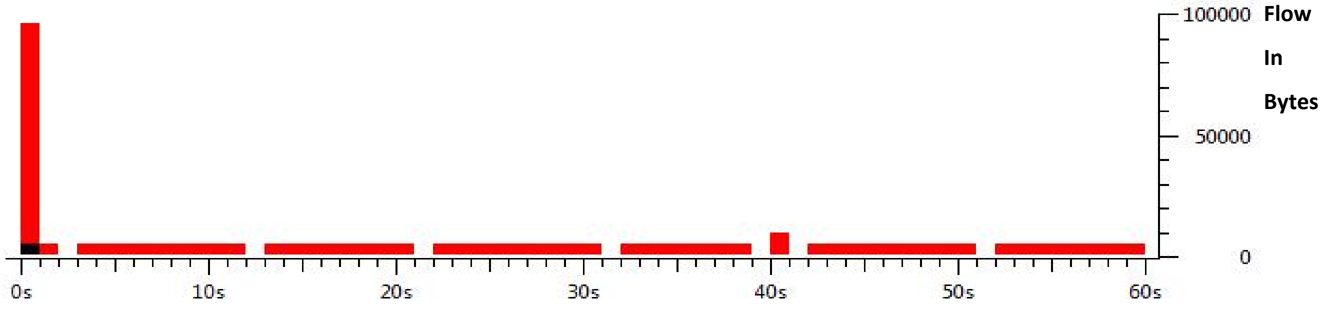
Home Address **Server Address**



Time	71.199.97.123	64.212.198.159	
0.353	(1257) PT=Unknown (96), SS (554)		RTP:
0.353	(1257) shockwave2 > rtsp [(554)		TCP:
0.354	(1257) PT=Unknown (96), SS (554)		RTP:
0.354	(1257) PT=Unknown (96), SS (554)		RTP:
0.354	(1257) shockwave2 > rtsp [(554)		TCP:

CNN Radio Live Partial Stream

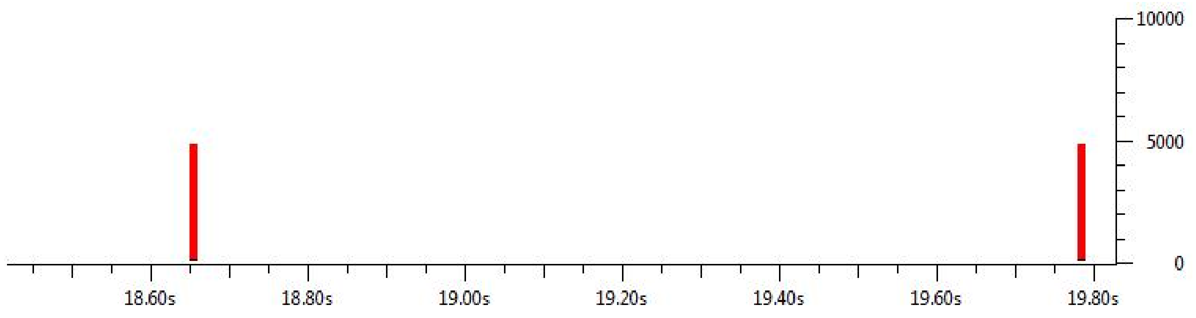
CNN Live is for the most part exactly the same as Radio Paradise, using RTP. The only interesting part is that while the server is using the standard RTP port (554), it is being sent a very specific port on the client (1257). This port is used for Macromedia Shockwave, so it makes it clear to an observer or researcher that the stream is being played through a Shockwave Flash interface, or that the player is being instantiated through Shockwave Flash interface calls.



Red- Incoming to Client Black-Outbound from Client

CNN Radio Live

There is nothing remarkable about this stream compared to the other live media streams. The primary difference is that there seems to be very predictable breaks in the streaming audio, as opposed to Radio Paradise where the client continually received data from the server. However, under closer examination, it turns out that the breaks in the graph are just an artifact of the graphing tool.



Flow

In

Bytes

Red- Incoming to Client Black-Outbound from Client

CNN Radio Live Zoom In

This does not mean there is not a pattern however, with how the packets are sent, when zooming in to use a much smaller scale it is clear the packets are sent about one second apart. When zooming in to an even smaller scale (intervals of .001, not pictured due to length of graph), it is confirmed that packets are only sent every 1.10 to 1.20 seconds. This is still in contrast to Radio Paradise, which sends on almost continuous basis.

Summary and tools for identification

- CNN Radio uses the RTP protocol (port 554) server side, and sends the packets to the Shockwave port (port 1257) on the client.
- There is a quick spike at the beginning of the connection, significantly larger than the buffering seen in Radio Paradise, followed by continuous transmission with very small breaks(1.10 to 1.20 seconds), as the media plays.

Part III- Applying Profiling Techniques Learned to Flow Data

The third part of our project is trying to apply what we learned from the second part, to flow data. We first detail a list of characteristics that were observed in watching packet flow data, and divide them based on how useful they are when looking at flow data. We then provide a description of what we expect the flow data records would look like, based on these characteristics. Finally, we apply the characteristics gathered to a real anonymized flow data, data set, looking for media services and detailing problems that arise when looking for specific media services within flow data.

Summary Data

Based on the profiling techniques in part two, we can create a summary table for each of our protocols, to help try and determine which would be easiest to look for in the flow data.

Legend: (+) Useful for flow analysis profiling	(-) Not useful , or of limited utility, for flow analysis profiling
YouTube (+) Data is sent at approximately .45 Mbps, but data transfer rate is regulated by software on the server and is not part of the protocol, so this might not be consistent. (-)Uses port 80 to transmit data, so blends in with HTTP traffic. (+)Same TCP connection stays open for length of transmission.	**YouTubeHD** (+) Data is sent at approximately 1.4 Mbps, but data transfer rate is regulated by software on the server and is not part of the protocol, so this might not be consistent. (-)Uses port 80 to transmit data, so blends in with HTTP traffic. (+)Same TCP connection stays open for length of transmission.
Hulu (+)Data transfer rate appears average a constant .75 Mbps, as the media is streamed even if the client has more bandwidth available. (+)Uses port 1935, Flash using RTMP, but there is a significant volume of Flash traffic making utility of this questionable. (+)Same TCP connection stays open for length of	**Hulu HD** (+) Data transfer rate appears to average a constant 2.2 Mbps, as the media is streamed even if the client has more bandwidth available. (+)Uses port 1935, Flash using RTMP, but there is a significant volume of Flash traffic making utility of this questionable. (+)Same TCP connection stays open for length of

transmission.	transmission.
Discovery Channel (-)Uses port 80 to transmit data, so blends in with HTTP traffic. (-) Very busy first 5 seconds of traffic. (+) Streams data from port 80 to 3 ephemeral ports on the client, creating 3 flows with similar size and duration.	**CNN Live Video** (+) Uses two streams to send data, the first using port 554 using RTSP. The second stream then sends the data over UDP. (-)/(+)Very busy first ten seconds on RTSP transmission. Remaining transmission over UDP is a consistent 100KB/sec.
Pandora (+) SSL login before song plays. (+) Creates a separate flow for every song sent. (+)Server uses port 554, to serve data to a client. (+) New song delivered every 3 to 7 minutes, depending on length of song. (+) Server serves song as quick as possible, usually in less than 5 seconds.	**Radio Paradise** (+)Server uses port 554, to serve data to a client. (-)Very busy for first second. (-)Continuous transmission with no pauses as in CNN radio.
CNN Live Radio (-)Very busy for first second. (-)/(+)After busy first second, data is sent at a constant rate of 5 packets totaling 5KB, with a one second pause between each group of 5 packets. (+)Server uses port 554, to serve data to a client. (+)The client uses port 1257, Shockwave specific, to receive data from a server. (+)Same TCP connection stays open for length of transmission.	

Based on the above summary and considering only the traits that would be relevant to flow data, and SiLK analysis, we would expect that when looking for these media services in flow data, the characteristics would be as followed:

YouTube/YouTube HD- The server uses port 80 to deliver the content, so it blends in with other port 80 traffic. Additionally, the data transfer rate from the server is not built into the protocol, since it simply uses well-formed TCP, as a result this rate may not stay consistent. Each YouTube video watched will have a different TCP session, and each flow from YouTube should have a relatively short length compared to the length of the video, since the video is sent using file delivery not streaming.

Hulu/Hulu HD- The server uses port 1935 to deliver the content, which is rarer than port 80 traffic, so it is helpful for profiling. The data transfer rate for the service has a better chance of remaining consistent with what was observed, because the protocol used has support for the server setting the maximum data transfer rate. The video is streamed, so the length of the flow should be close to the length of the video being watched.

Discovery Channel- The server uses port 80 to deliver the content over HTTP. However, the Discovery Channel requires a proprietary player so there is evidence that video is streamed over time, as opposed to file delivery. The length of the flow will be proportional to the length of the video(a 10 minute video, should have close to a 10 minute. flow). The server serves the data to the client from port 80, but to three ephemeral ports on the client. This means that for every one video served by Discovery, there will be three streams sent to the client, all from the same port on the server to three different ports on the client. These three streams will all be approximately the same length, and close (within 15%) of the same size. This will create three separate flows in the SiLK toolset. Currently SiLK does not support any type of clustering algorithm which would make it easier to look for these three flows.

CNN Live Video- The server uses TCP port 554 to initially stream data, and then uses UDP to continue streaming the data. This would create two separate flow records for each video watched. The UDP flow is being streamed in "real-time", at a rate of approximately 100 KB/sec.

Pandora- The server first established a control channel using SSL. Then, for each song Pandora plays, a separate TCP connection is created, with the server sending the data from port 554 to an ephemeral port on the client. The song is sent as a file delivery, so the flow length will be relatively short compared to the length of the song (for example a 4 minute song might be delivered in 8 seconds.)

Radio Paradise- The server serves data using port 554 to an ephemeral port on the client. This service uses streaming, so the length of the flow should be equal to the length the user listens to the radio station.

CNN Live Radio- The server serves data using port 554 to an ephemeral port on the client. This is streaming audio, so the length of the flow should be equal to the length of time the user listens to the stream. The average flow rate is 5KB/sec.

Application to Real Flow Data

Due to the fact that flow data concentrates more on the overlying TCP or UDP connection rather than packet capture, it seemed at first that flow data would not play a large role in identifying the behavior patterns previously mentioned in this paper. After examining flow data, this hypothesis proved true, but some interesting information was obtained anyway – although it would be difficult to predict what specific media service was being used based on flow data, it was possible to determine with relatively high accuracy that a client machine was accessing media services in general. Let’s take a closer look at some examples, which were taken from anonymized European ISP data on 9 September 2008. An important caveat of this data is that it was sampled at 1:100, so when determining size or bit rate, we had to multiply the flow record by one hundred to get a projected value. While this value should be fairly accurate for longer flows, for shorter flows it has more of a margin of error, since there is so much extrapolation involved.

Pandora

As stated in the previous sections, Pandora’s connection behavior shows that it simply downloads each song an a compressed format, one song at a time. It downloads a song, plays it, and towards the end of that song downloads another so that playback can continue without interruption. Pandora also gives the user the option to “veto” a song that isn’t appealing, therefore stopping playback immediately and sending a new song down (which would mean another download). In examining flow data, a pattern emerged:

Source IP	Destination IP	Source Port	Destination Port	Start	End	Seconds	Size in MB
107.133.229.221	58.190.251.249	554	2185	5:10:06 AM	5:10:19 AM	00:13.0	4.43

107.133.229.221	58.190.251.249	554	2305	5:15:38 AM	5:16:12 AM	00:34.0	3.15
107.133.229.221	58.190.251.249	554	2333	5:16:29 AM	5:16:37 AM	00:08.0	2.03

The table above shows three connections between a client and server on port 554, which is the known port that Pandora communicates on. The first flow starts at 5:10:06 AM and lasts for about 13 seconds, where 4.43 MB are downloaded. This is the typical download of an MP3 file, which is about 1MB per minute of compressed music. So, the user just downloaded a song that is about 4.5 minutes long. The flow ends, and there is no connection for about 4.5 minutes (when the song is about to end). The next connection starts to download the next song. Another connection begins only a few seconds after that, probably due to the user either choosing to skip the current song or switching to another Pandora radio station. Of course, this behavior won't confirm a Pandora connection without verifying the IP address, but the type of connection can be isolated (we know that the user is listening to some type of streaming media service).

Radio Paradise

Radio Paradise's behavior looks completely different than Pandora. Unlike Pandora, which is an on-demand service, Radio Paradise (and other similar Internet Radio services) stream music in a broadcast fashion on a bit-rate.

Source IP	**Destination IP**	**Source Port**	**Destination Port**	**Start**	**End**	**Seconds**	**KBits/s**
112.7.36.105	**119.219.7.30**	**554**	**1378**	**9/9/08 3:32**	**9/9/08 5:51**	**8372**	**80**
112.7.36.105	**118.136.22.19**	**554**	**4231**	**9/9/08 4:54**	**9/9/08 5:53**	**3547**	**79**
112.7.36.105	**118.136.33.37**	**554**	**1269**	**9/9/08 5:29**	**9/9/08 5:53**	**1446**	**71**
112.7.36.105	**58.190.66.237**	**554**	**1045**	**9/9/08 2:16**	**9/9/08 4:35**	**8328**	**93**
112.7.36.105	**58.190.66.237**	**554**	**1052**	**9/9/08 4:36**	**9/9/08 5:53**	**4645**	**79**

112.7.36.105	**97.224.22.148**	**554**	**50331**	**9/9/08 1:57**	**9/9/08 5:53**	**14190**	**87**
112.7.36.105	**97.224.55.150**	**554**	**1312**	**9/9/08 3:04**	**9/9/08 5:53**	**10142**	**82**
112.7.36.105	**118.136.32.217**	**554**	**4048**	**9/9/08 2:22**	**9/9/08 5:53**	**12676**	**85**
112.7.36.105	**118.137.227.20**	**554**	**1507**	**9/9/08 2:29**	**9/9/08 5:53**	**12202**	**89**
112.7.36.105	**118.138.62.206**	**554**	**4611**	**9/9/08 5:06**	**9/9/08 5:53**	**2820**	**71**
112.7.36.105	**119.219.160.39**	**554**	**2258**	**9/9/08 4:53**	**9/9/08 5:53**	**3640**	**75**
112.7.36.105	**58.190.231.127**	**554**	**49190**	**9/9/08 1:42**	**9/9/08 2:45**	**3757**	**106**
112.7.36.105	**118.136.190.161**	**554**	**1203**	**9/9/08 2:37**	**9/9/08 5:53**	**11794**	**86**
112.7.36.105	**118.138.207.155**	**554**	**2548**	**9/9/08 3:51**	**9/9/08 5:53**	**7348**	**86**
112.7.36.105	**119.219.124.133**	**554**	**1651**	**9/9/08 0:43**	**9/9/08 1:18**	**2111**	**98**

The table above shows many outgoing connections from a server on a known streaming media port, 554. These connections last from a few minutes to a few hours, which is typical for Internet radio usage. Additionally, the bitrates are all about 75-100 Kbps, which reflects online audio streaming.

Another Video Protocol?

As more flow data was analyzed, more patterns started to emerge. As previously hypothesized, the exact source or media protocol was difficult to pinpoint, but it was clear that there was some kind of media delivery taking place. This is shown in yet another media pattern that was discovered, this time showing video.

Source IP	Destination IP	Source Port	Destination Port	Start	End	Seconds	KBits/s
57.49.241.85	97.224.79.23	554	57200	9/9/08 4:29	9/9/08 5:00	1836	488

57.49.241.85	118.136.163.193	554	1220	9/9/08 4:34	9/9/08 5:02	1688	482
57.49.241.86	119.219.244.47	554	2566	9/9/08 4:58	9/9/08 5:02	261	470
57.49.241.86	119.219.182.229	554	43715	9/9/08 4:34	9/9/08 5:01	1641	488
57.49.241.87	118.136.36.212	554	1329	9/9/08 4:35	9/9/08 5:02	1626	477
57.49.241.88	118.201.77.155	554	53656	9/9/08 4:51	9/9/08 4:57	363	482
57.49.241.88	97.224.215.251	554	2257	9/9/08 4:39	9/9/08 4:51	722	439
57.49.241.88	118.136.165.221	554	50750	9/9/08 5:08	9/9/08 5:17	537	415
57.49.241.88	119.219.246.224	554	53857	9/9/08 4:51	9/9/08 5:21	1791	448
57.49.241.89	119.219.77.51	554	1431	9/9/08 4:46	9/9/08 4:46	17	298
57.49.241.89	97.225.234.153	554	2106	9/9/08 4:35	9/9/08 4:36	84	443

The connections above show a subnet of servers delivering media to many different clients over port 554, which is a known streaming media port. These connections range from a few seconds to a few minutes, and are at a bit-rate of about 500Kbps, which is the typical transmission rate for streaming video. Services such as CNN and others use this port to deliver streaming video, and it is possible that these connections are either live video or on-demand video that is delivered to clients.

With more flow data to analyze, it is extremely likely that even more interesting media patterns would emerge. Despite this, the patterns cannot be attributed to exact behavior, as mentioned previously and demonstrated when using packet capture data. Instead, flow data can just provide overarching insight into the behavior of connections on the network. Since packet capture data is a bit more expensive in terms of resources, perhaps a future scheme would include examining flow data and launching packet capture data later if particular trends of interest are being discovered.

Conclusion and Enhancements to SiLK Tools

When using limited packet data, which is individual packet capture data but of only selected packet fields, it is much easier to decode anonymized data. Decoding of information about the client can be done by monitoring ephemeral port usage pattern, or using a passive tool to analyze the packet data. Either of these methods can give insight into what operating system is being used by the client. Decoding of information about the server can by looking at graphical representation of the conversations between client and server. Many of the media services we looked at had unique graphical representations, so it makes it easier to differentiate the media services. For decoding using graphical representation, all the packet information needed is the source and destination IPs, which can be anonymized as long as they are kept consistent, source and destination port, and the size of the packet.

When using flow data, it is much harder to decode anonymized data. Since flow data rolls-up all the packets into one flow record, graphical analysis is not possible. As a result, it is harder to be positive, which services are being accessed, and other characteristics have to be looked at to try and make a best guess, based on specific characteristics.

When doing our research, we found two areas which could be incorporated into the SiLK toolset which would make research easier, as well as give more certainty into the services used. First, there should be an easier way to determine average bit rate of a flow. Right now, it is somewhat difficult and all of the streaming services have a different average bit rate, ergo it would be useful to be able to see average bit rate of a flow and use that as a differentiator for finding a media service. A second useful addition to the SiLK tools, would be if the rw-tools supported some sort of clustering. This is evident in the Discovery Channel streaming, where we know that the Discovery Channel serves data on port 80, to 3 ephemeral ports on the client. Since these three streams are all established very close in time to each other, the three ephemeral ports used on the client should all be close together, assuming is in an OS assigns ephemeral ports linearly. However, right now there is no easy way using the rw-tools to search for 3 streams that have destination ports within +/- 5 ports of each other, and have a size within +/- 15% of each other. There is currently no easy way to do this with the rw-tools.