

Eetu Immonen

Assignment 1: IPv4 Packets

This assignment is all my own work. This work has not been submitted for assessment in any other context. I have not knowingly allowed others to copy my work.



Contents

1	Introduction	3
2	Platform and Experimental Parameters	3
2.1	Hardware and Network Details	3
2.2	Software Details	4
2.3	Experimental Parameters	4
3	Main Analysis of IPv4 Packets	6
3.1	IPv4 Packet Size Distribution	6
3.2	IPv4 Protocol Breakdown	7
3.3	IP Flags	7
3.4	IP Type of Service	8
4	Secondary Analysis of BitTorrent IPv4 Packets	8
5	Conclusion	10

1 Introduction

This coursework for Networks and Communications (NET0183) course analyses IPv4 traffic and tries to reproduce Wolfgang John and Sven Tafvelin's experiment on "*Analysis of Internet Backbone Traffic and Header Anomalies Observed*" conducted in 2006 in Göteborg.

First, in chapter 2 platform and experimental parameters are specified and differences between this coursework and the original John and Tafvelin's experiment are explained. Chapter 3 is the main analysis that tries to reproduce the findings of the original experiment, most notably the bimodal nature of the cumulative IPv4 packet size distribution.

Chapter 4 contains some bonus work on tracing BitTorrent traffic, similar cumulative IPv4 packet size distribution chart is showed and briefly analyzed like in chapter 3. The last chapter wraps this coursework up with a short conclusion.

2 Platform and Experimental Parameters

2.1 Hardware and Network Details

The experiments were run on Lenovo ThinkPad R61 (89185QG) laptop. WLAN connection was used to connect the laptop to the Internet. The speed of the WLAN link to the router during the experiments was 54.0 Mbps and the signal strength was very good, so the link did not become a bottleneck for the experiments. Hardware details are below:

Motherboard:	Northbridge: Intel PM965 Revision C0. South Bridge: Intel 82801HBM (ICH8-ME) Revision C0
Processor:	Intel(R) Core(TM)2 Duo CPU T7100 @ 1.80GHz. 791.1 MHz frontside bus, L1 data cache 2 x 32 kB, L1 instructions cache 2 x 32 kB, L2 cache 2048 kB
Memory:	1x 1 GB DDR2 @ 667 MHz
Harddrive:	Seagate Serial-ATA 120 GB
WLAN Adapter:	Intel(R) Wireless WiFi Link 4965AGN. Supports data rates up to 300 Mbps with Intel(R) Next-Gen Wireless-N access point

2.2 Software Details

IPv4 packets were captured with Wireshark network protocol analyzer and WinPcap packet capture and network monitoring library. Firefox web browser was used to generate the network traffic for the main analysis of IPv4 packets. The BitTorrent experiment was conducted with μ Torrent. The operating system is Windows XP. Details of the used software versions and the operating system are below:

Operating System: Windows XP Professional Service Pack 3. Kernel version 5.1.2600.5913. 2600.xpsp_sp3_gdr.091208-2036

Wireshark: Version 1.2.6 (SVN Rev 31702)

WinPcap: Version 4.1.1 (packet.dll version 4.1.0.1753)

Firefox: Version 3.5.8 (Mozilla/5.0)

μ Torrent: Version 1.8.5 (build 17414)

2.3 Experimental Parameters

The main capture of IPv4 packets was done with Wireshark network protocol analyzer software to capture the first 54 bytes of each frame whereas the original experiment “Analysis of Internet Backbone Traffic and Header Anomalies Observed” conducted by Wolfgang John and Sven Tafvelin in 2006 used hardware to capture the packets. They used optical splitters attached to Endace DAG6.2SE cards that captured the first 120 bytes of each frame.

The other differences between the original and my experiment was that I captured the WLAN traffic between my laptop and the wireless router and limited the traffic to only one application, browsing Wikipedia with Firefox. The original study captured the packets on a high-speed optical fibre and it did not limit the applications generating the IPv4 traffic. Further, I did only single trace of 50,000 frames compared to 160 traces spread over 20 consecutive days Wolfgang John and Sven Tafvelin conducted. All in all, a much broader cross-section of IPv4 traffic was captured by the original experiment than in my reproduction.

My main trace of the IPv4 traffic was done on Saturday, 13th March 2010, between 7:43 and 7:53 PM. Each packet was limited to the first 54 bytes and capture was stopped once 50,000

truncated packets were collected. Capture filter ip and host 10.1.1.117 was used to limit the capture to only IPv4 packets to and from my own computer. During the trace, I used only Firefox to browse various Wikipedia pages. Figure 1 illustrates the Wireshark capture options.

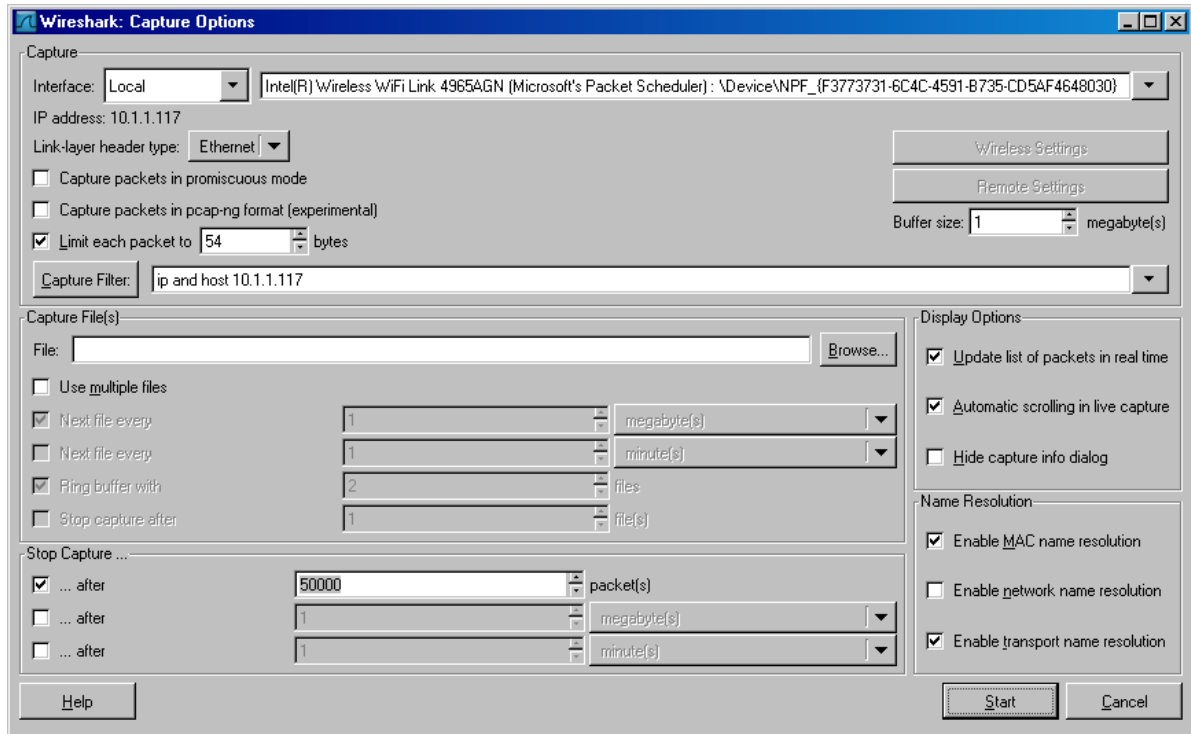


Figure 1: Wireshark capture options for the main trace

The BitTorrent trace, discussed in chapter 4 was done on Monday, 15th March 2010, between 2:53 and 2:55 AM. The Internet connection and link to the router was different from the main trace, but it should not have an effect to the trace since the speed to the WLAN router was the same, 54.0 Mbps. Because of the different connection, my IP address was different, but other experimental parameters were the same.

The downloaded torrent file was `ubuntu-9.10-desktop-i386.iso`, which contained the latest version of Ubuntu operating system in iso image format. The size of the file was 689 MB, although the file was not fully downloaded. The piece size of the torrent file was 512 kB, totalling 1380 pieces.

3.2 IPv4 Protocol Breakdown

The IPv4 protocol breakdown is presented in table 1. DNS and NBNS rows are denoted with a star (*) because they are application layer protocols and their percentage breakdowns are calculated from total UDP packets which is their payload carrier.

Name	Packets	Data (bytes)	% Packets	% Data
TCP	46,851	44,416,466	93.70 %	99.16 %
UDP	2,910	334,657	5.82 %	0.75 %
DNS*	2,907	334,381	99.90 %	99.92 %
NBNS*	3	276	0.10 %	0.08 %
ICMP	239	43,324	0.48 %	0.10 %

Table 1: IPv4 Protocol Breakdown

My trace consisted of only two transport layer protocols, TCP and UDP. The Internet Control Message Protocol, ICMP, is technically an Internet Layer protocol because it is not usually used directly by network applications. Nevertheless, ICMP is an integral part of IP because it relies on IP to perform its tasks.

My results are very similar to the results of John and Tafvelin's experiment. The only notable difference is in UDP traffic, their portion of UDP was around 7.80 % of total captured IPv4 packets. That can mostly be explained by the fact that their experiment captured some UDP Denial of Service (DoS) traffic.

TCP was the major transport layer protocol because loading Internet pages utilize TCP's error correction facilities. Almost all UDP datagrams were Domain Name Server, DNS, communications used to resolve domain names when I was opening new Wikipedia pages.

3.3 IP Flags

I further analyzed my Wikipedia main trace with Wireshark. First I made an analysis of the IP flags (fragment bits). With a display filter `ip.flags == 0x02` I could select all the packets which had the *Don't Fragment* option selected. It turned out that 93.70 % of all traced IPv4 packets have the don't fragment bit (DF) set. Filter `ip.flags == 0x00` revealed that 6.30 % of all captured packets did not have any IP flags set. There were no packets with *More Fragments* (MF) bit set and neither any packet with invalid IP flag values.

Don't fragment option was used a bit more in my trace compared to 91.30 % in John and Tafvelin's experiment. My 6.30 % of none IP flags set was lower for their experiment where 8.65 % of all observed packets did not use either DF or MF bit.

Since I limited my trace to only one online application it is not a surprise that there were no invalid IP flag values. In their experiment Wolfgang John and Sven Tafvelin found some invalid fragment bits which were most probably caused by misconfigured DNS server.

3.4 IP Type of Service

In my trace the IP Type of Service (TOS) field did not contain any Explicit Congestion Notifications (ECN) or Differentiated Services. It means that differentiated services field of all packets was set to zero (0x00), which is the default value. John and Tafvelin's experiment showed that 83.1 % of the observed IPv4 packets stored the value of zero in the TOS field. Since my trace was so specific and limited compared to their experiment it does not come as a big surprise that my trace did not contain packets that utilized the TOS field.

4 Secondary Analysis of BitTorrent IPv4 Packets

In addition to my main trace with Firefox, I captured BitTorrent data while downloading a file with μ Torrent. The IPv4 packet size distribution for that trace is presented below in figure 3.

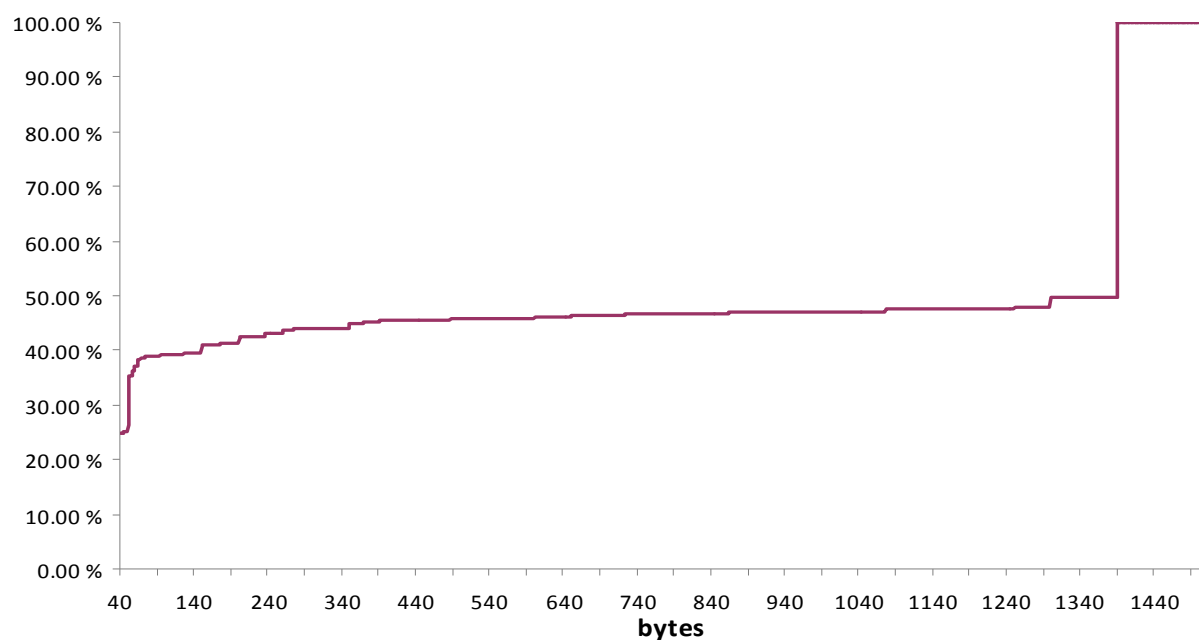


Figure 3: Cumulative IPv4 Packet Size Distribution for BitTorrent Trace

The biggest difference between IPv4 packet size distribution between the main trace with Firefox and the BitTorrent trace is that BitTorrent seems to prefer packet sizes that are smaller than the maximum size of 1500 bytes, despite that the chart is clearly binomial. 39.15 % of total packets fall between 40 and 100 byte size. There were no packets bigger than 1390 bytes. 50.27 % of all packets were exactly 1390 bytes.

Between 100 and 1390 there were more notable modes than in the main trace. The biggest mode was at 1300 bytes and the second biggest was at 150 bytes, they contributed 1.83 % and 1.23 % of all packets. Other smaller modes were at 201 bytes, 351 bytes and 260 bytes (1.03 %, 0.87 % and 0.64 % respectively). On the other side of the scale, there was also a small mode at 1076 bytes with 0.57 % of all packets.

Further analysis of the biggest modes between 100 and 1390 bytes revealed that all 1300 bytes long packets contained class selector 1 (0x08) in differentiated services codepoint field. Class selector 1 was found in 41.10 % of all observed BitTorrent packets. All 150 bytes long packets had both PUSH and ACK flags set in TCP flags (0x18). All except one packet sized 201 bytes was using UDP.

Protocol	Packets	Data (bytes)	% Packets	% Data
TCP	47,663	38,887,971	95.33 %	98.52 %
UDP	2,337	585,394	4.67 %	1.48 %

Table 2: IPv4 Protocol Breakdown of the BitTorrent Trace

TCP was used in 95.33 % of packets and UDP in 4.67 % as shown in table 2. The data is still carried mostly via TCP (98.52 %), but UDP is used a bit more than in regular browser traffic (1.48 %). What is the biggest difference compared to the protocol breakdown of the main trace is that UDP is used to actually carry data and DNS constitutes only 0.01 % of all UDP traffic. No ICMP was used during BitTorrent trace.

5 Conclusion

I was quite surprised how similar my cumulative IPv4 packet size distribution graph was compared to Wolfgang John and Sven Tafvelin experiment, keeping in mind that they captured and analyzed over 10 billion packets from high-speed optical internet backbone whereas my trace captured and analyzed only 50,000 packets and limited and filtered the traffic to one online application and personal WLAN.

The size distribution remains binominal as Wolfgang John and Sven Tafvelin's experiment showed, with further emphasis on 1500 bytes long packets. Although, as the brief BitTorrent trace shows, the graph could be a bit different if traces from many online applications would be combined.

The major difference between the frequencies of protocols carried as payload was the smaller quantity of UDP packets in my trace. This can mostly be explained by that there were some UDP DoS attacks collected in John and Tafvelin's experiment. Further, I think that including BitTorrent and online games to my results, the amount of UDP traffic would increase because in some cases BitTorrent and some online games does not need the same kind of error detection used in displaying web pages with TCP.

I think that the cumulative IP packet distribution might not remain bimodal in the future because BitTorrent- and other P2P-traffic contribute a big portion of the web traffic around the world. Also the emerging IPv6 protocol can mix the packet size distribution even more.