Distributed System and Networks
Networks Coursework

Elliot Purvis
Student ID: 28561716

3rd Nov 2017

Part I
IPv6, Ping & Traceroute

1 Ping testing

In this section we investigate the use of IPv6 on the internet using ping and traceroute utilities. Below are the listed hosts, including resolved IPv4 and IPv6 addresses, as well as average, minimum and maximum round trip times.

2 What conclusions can be drawn about the adoption of IPv6 on the internet?

Clearly IPv6 adoption is increasing, with almost all mainstream sites adopting the standard. That said, from the selection below there are some notable exceptions, including major search provider Yahoo, and more surprisingly notable sites relating directly to Computer Science, Programming and technology, with Github and Stackoverflow still offering no IPv6 support. Most surprisingly is Reddit, with 'the frontpage of the internet' still having no IPv6 support. So while IPv6 support is certainly on it's way, the lack of IPv6 support from several mainstream sites may prove to be a stumbling block, especially when trying to persuade more ISP's to support IPv6.

3 What conclusions can you draw from the data you have gathered about the relative performance of IPv4 and IPv6?

The relative performance of IPv4 and IPv6 seems almost identical, with IPv6 certainly faring no worse in terms of round trip times. In almost all attempted sites, IPv6 is either identical or faster than it's IPv4 alternative, with one or two exceptions where IPv6 is slower , albeit within a margin of error of two to three ms. In some cases, IPv6 is notable faster, for example when pinging Yahoo, almost 20 ms is saved. However, with the high rrt times of the IPv4 pings, this may also be within a margin of error.

4 How confident are you in these conclusions? What factors does your data gathering not take into account and how could the testing be improved?

I am confident in the conclusions drawn from this data, largely concluding that IPv6 is practically equivalent in performance to IPv4. Our testing methodology however does not take into account larger packet sizes, with the windows and $ping$ $[host]$ and $ping - 6$ $[host]$ commands used in the testing only sending 32 bytes of data, far from the maximum payload size of IPv6 (64 kilobytes). Further to this, no experimentation is made with varying IPv4 header sizes when compared to the fixed 40 byte header of IPv6, implemented in an attempt to facilitate efficient parsing of the packet header. This could potentially affect routing speeds, decreasing packet round trip times. We also make no acknowledgement of fragmented and extension packets, with extension packets replacing the options section of the traditional IPv4 header, and larger fragmented packets being essential in many of the common tasks we perform daily online.

5 What do you notice about the RTT of Google-controlled services? Why do you think this is and can you confirm your theory with traceroute and traceroute6?

Google's services all return the same round trip time, in this case 4ms. This is clearly impossible for a server hosted in Australia (in the case of google.com.au). It is likely that google is using some form of global Anycast network, routing client traffic to a node on the basis of geographical distance, congestion and other measures. In this instance, traffic targeted at any of Google's services, whether Google.com.au, or Youtube.com, is being directed at a single block of IP addresses, likely a datacentre in Western Europe, if not England. This is shown in the traceroute results below, showing the final hop for several google services landing within a small IP range. In the case of Google.com.au, a connection to an Australian Server is made (demonstrated by the far higher RRT, at almost 150ms), before Anycast routes the traffic back towards servers hosted in Western Europe; just as if we had connected to Google.co.uk. In fact, the final directed hop on both Google.co.uk and Google.com.au are handled by the same node (216.58.212.99).

5.0.1 Google.co.uk

```
  1    13 ms    16 ms     2 ms  192.168.0.1
  2     *        *        *     Request timed out.
  3    16 ms    12 ms    18 ms  sotn-core-2b-xe-813-0.network.virginmedia.net [...]
  4     *        *        *     Request timed out.
  5    24 ms    64 ms    19 ms  tele-ic-7-ae2-0.network.virginmedia.net [...]
  6    25 ms    21 ms    21 ms  74.125.52.226
  7    22 ms    29 ms    41 ms  108.170.246.193
  8    23 ms    23 ms    23 ms  108.170.238.151
  9    22 ms    28 ms    32 ms  lhr35s06-in-f3.1e100.net [216.58.212.99]
```

5.0.2 Google.com.au

```
  1     2 ms     2 ms     2 ms  192.168.0.1
  2     *        *        *     Request timed out.
  3    13 ms    11 ms    19 ms  sotn-core-2b-xe-813-0.network.virginmedia.net
       [62.255.45.93]
  4     *        *        *     Request timed out.
  5    15 ms    19 ms    19 ms  tele-ic-7-ae2-0.network.virginmedia.net
       [62.253.175.34]
  6    19 ms    21 ms    21 ms  74.125.52.226
  7    23 ms    24 ms    21 ms  108.170.246.193
  8    22 ms    26 ms   131 ms  108.170.238.151
  9    23 ms    29 ms    21 ms  lhr35s06-in-f99.1e100.net [216.58.212.99]
```

5.0.3 Youtube.com

```
  1     6 ms     3 ms     1 ms  192.168.0.1
  2     *        *        *     Request timed out.
  3    37 ms    13 ms    17 ms  sotn-core-2b-xe-813-0.network.virginmedia.net
       [62.255.45.93]
  4     *        *        *     Request timed out.
  5    40 ms    19 ms    19 ms  tele-ic-7-ae2-0.network.virginmedia.net
       [62.253.175.34]
  6    38 ms    38 ms    45 ms  74.125.48.190
  7    21 ms    24 ms    22 ms  108.170.246.225
  8    24 ms    22 ms    18 ms  108.170.238.149
  9    18 ms    17 ms    23 ms  lhr35s06-in-f110.1e100.net [216.58.212.110]
```

As shown above, the final hop for each of these services lands in the same IP range, albeit with some detours along the way. This is most easily demonstrated with IPv4 (due to the more page-friendly address range), but functions just the same on IPv6; directing users' traffic to an optimal server based on their distance, both from a geographical and topological perspective, but also considering congestion, and likely a wide range of other factors. This type of distributed system allows google to operate a large CDN, serving content from datacentres as close to the end user as possible, keeping latency and transfer times down.

6 What do you notice about IPv4 results for Netflix and ebay? Why do you think this happens?

When setting up DNS records for a website, it is common for companies to setup multiple records directing to the same servers, however this causes problems in itself, such as being penalised by search engines for duplicate content. Hence, when setting up A-records (and AAAA records for IPv6), it is common for companies to chose between either yoursite.com or www.yoursite.com. In the case of Ebay, the later is chosen. As shown below, when pinging www.ebay.co.uk with both IPv4 and IPv6, the site responds as normal.

6.0.1 ebay.co.uk

```
>ping -4 ebay.co.uk

Pinging ebay.co.uk [66.135.201.153] with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 66.135.201.153:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

6.0.2 www.ebay.co.uk

```
>ping -4 www.ebay.co.uk

Pinging e11847.g.akamaiedge.net [23.44.102.198] with 32 bytes of data:
Reply from 23.44.102.198: bytes=32 time=10ms TTL=55
Reply from 23.44.102.198: bytes=32 time=5ms TTL=55
Reply from 23.44.102.198: bytes=32 time=5ms TTL=55
Reply from 23.44.102.198: bytes=32 time=5ms TTL=55

Ping statistics for 23.44.102.198:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 5ms, Maximum = 10ms, Average = 6ms
```

As is clear, www.ebay.co.uk resolves to the consumer facing website. ebay.co.uk however, while resolving to an IP address does not return pings, either refusing connections on port 80/443 or because the server is not reachable.

7 Why is there such a large variance in RTT between google.co.uk, github.com and iinet.net.au?

The variance in the Round trip times for each of these three sites is caused by their massive variance in geolocation. We can show this by running a traceroute, then running an online Geolocation tool on the IP of the nal hop. In the case of Github (192.30.253.113), the nal hop is located in San Francisco. The same stands true for iinet.net.au, which is clearly located in Australia. When pinging google services, were redirected by the aforementioned Anycast network, so while the pingable IP of google.co.uk resolves to California, were more likely being served by a server in western Europe, if not the UK itself. This massive geographical dierence between the three servers, with one in Australia (or south east Asia), one in Europe and one on the East coast of the United States, is likely the cause of the dierence in round trip times.

Part II
Sockets and Wireshark

8 Write a client program / script to send your username to each of tese servers and print the response received

For each of the servers, a seperate script was used, albeit largely the same process. These scripts are referred to as TCP.py and UDP.py

TCP.py

```
import sys
import socket


server_address = ('wsn.ecs.soton.ac.uk', 5002)
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "Starting up socket on port 5002. "
sock.connect(server_address)

response = ""
try:
    message="ep1e16"
    #print("Sending %s to %s on port %s." % message, server_address)
    sock.sendall(message)

    next_response = sock.recv(16)
    while next_response != '':
        response += next_response
        next_response = sock.recv(16)


finally:
    print "Response : %s" % response
```

UDP.py

```
import sys
import socket


server_address = ('wsn.ecs.soton.ac.uk', 5005)
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
print "Starting up socket on port 5005. "
sock.connect(server_address)

response = ""
try:
    message="ep1e16"
    #print("Sending %s to %s on port %s." % message, server_address)
    sock.sendall(message)

    while 1:
        d = sock.recvfrom(1024)
        reply = d[0]
        addr = d[1]

        print 'Server reply : ' + reply

except socket.error, msg:
    print 'Error code : ' + str(msg[0]) + ' Message ' + msg[1]
    sys.exit()
```

The text returned by each server is included below.

TCP

```
RESTART: C\Users:Elliot\Google Drive\Uni\Coursework\Distributed Systems
     and Networks\Python\TCP.py
Starting up socket on port 5002.
Response: Hello ep1e16, the date is 01/11/17 your code: qnpn
```

UDP

```
RESTART: C\Users:Elliot\Google Drive\Uni\Coursework\Distributed Systems
     and Networks\Python\UDP.py
Starting up socket on port 5005.
Response : Hello ep1e16, the date is 01/11/17
```

Figure 1: Appendix A: Ping Test Results (1.1)

Host	IPv4	AvgRTT	MinRTT	MaxRTT	IPv6	AvgRTT	MinRTT	MaxRTT
google.com	216.58.206.46	6	5	7	2a00:1450:4009:801::200e:	4	4	4
google.co.uk	216.58.206.35	4	4	4	2a00:1450:4009:812::2003:	4	4	4
google.com.au	172.217.23.3	4	4	4	2a00:1450:4009:812::2003:	4	4	4
youtube.com	216.58.206.46	4	4	4	2a00:1450:4009:801::200e:	4	4	4
instagram.com	52.87.65.142	79	79	81	2406:da00:ff00:36a4:eb60:	81	81	81
yandex.ru	77.88.55.60	38	38	39	2a02:6b8:a::a:	37	37	37
wikipedia.org	91.198.174.192	9	9	9	2620:0:862:ed1a::1:	12	12	12
stackoverflow.com	151.101.129.69	3	3	3	N/A	N/A	N/A	N/A
linkedin.com	108.174.10.10	80	80	80	2620:109:c002::6cae:a0a:	78	78	78
yahoo.com	98.138.253.109	103	103	103	2001:4998:58:c02::a9:	81	81	82
yahoo.co.uk	72.30.203.4	81	81	81	N/A	N/A	N/A	N/A
yahoo.com.au	98.137.236.150	141	141	142	N/A	N/A	N/A	N/A
facebook.com	157.240.1.35	4	4	4	2a03:2880:f129:83:face:b00c:0:25de	4	4	4
Netflix.com	54.154.117.233	TO	TO	TO	2a01:578:3::3430:2cc4	16	15	16
ebay.co.uk	66.211.181.20	TO	TO	TO	N/A	N/A	N/A	N/A
bbc.co.uk	212.58.244.22	3	3	4	2001:41c1:4008::bbc:1	3	3	4
www.bbc.co.uk	212.58.246.93	8	8	8	N/A	N/A	N/A	N/A
reddit.com	151.101.65.140	3	3	3	N/A	N/A	N/A	N/A
github.com	192.30.253.113	78	76	78	N/A	N/A	N/A	N/A
taobao.com	140.205.94.189	TO	TO	TO	N/A	N/A	N/A	N/A
mail.ru	94.100.180.200	52	52	52	2a00:1148:db00:0:b0b0::1:	54	54	54
wikia.com	151.101.64.194	3	3	4	2a04:4e42::194:	3	3	3
loopsofzen.uk	N/A	N/A	N/A	N/A	2001:8b0:0:30::666:102	5	5	6
iinet.net.au	203.173.50.151	367	367	368	2001:4478:1310:1fff:203:173:50:151	358	358	359