Traffic on TCP port 53

and *DNS Response Sizes*

at U Auckland

CAIDA/WIDE Workshop, March 2006

Nevil Brownlee

Why look at TCP port 53?

- Recent reports [Randy, Daniel] suggest that
  - Successive DNS requests can go to different anycast root servers
  - Routes can be unstable – they can switch quickly
- UDP can cope with such switches, TCP would not cope well
- So .. how much *DNS over TCP* traffic is there?
  - Expect to see zone transfers,
  - and a few name lookups
- Used *NeTraMet* to collect data at Auckland

NeTraMet meter setup at Auckland

- Meter observes all Internet traffic in/out
- Meter can run SRL rulesets run to produce
  - DNS root/gTLD RTTs
  - Other rulesets, as needed from time to time
- Reworked NeTraMet's *TurnaroundTime* code to handle DNS over TCP
  - NeTraMet uses the timestamps for the packets that carry the first n bytes of the DNS request and response
- Ran ruleset to observe flows on TCP port 53
  - Ruleset tries to use first packet as *source* of flow
- Also ran tcpdump to gather headers of TCP 53 packets

Flow Data File, Dec 05 – Jan 06

```
##NeTraMet v5.1:  -c900 -r dns-tcp-wire.rules  localhost eth3 \
   10000 flows  starting at 17:32:23 Thu 22 Dec 2005
#Format: flowruleset flowindex firsttime sourcepeertype sourcetranstype \
   sourcepeeraddress destpeeraddress  d_tooctets d_fromoctets \
   d_topdus d_frompdus  d_tolostpdus d_fromlostpdus  (d_toturnaroundtime)

#Time: 21:15:00 Thu 22 Dec 2005 localhost Flows from 1246168 to 1336150
20 11   94590 1 6  130.216.1.2 67.15.35.19  176 74  2 1  0 0  (0)
20 30  234963 1 6  130.216.1.1 216.26.160.5  74 0  1 0  0 0  (0)
20 77 1247060 1 6  130.216.1.1 204.152.184.64  37034 1659476  561 1098 \
                   0 0  (0)
20 78 1276973 1 6  218.25.41.136 130.216.112.11  156 0  2 0  0 0  (0)
20 79 1292934 1 6  130.216.1.1 205.171.14.195  386 4620  5 4 \
                   0 0  (5 2 10 7000  1 14 0 0  1982)
20 80 1292989 1 6  130.216.1.2 205.171.9.242  452 4752  6 6 \
                   0 0  (5 2 10 7000  1 14 0 0  2996)
20 81 1299794 1 6  130.216.1.1 194.30.63.66  0 60  0 1  0 0  (0)
20 82 1330338 1 6  130.216.1.1 62.45.94.130  176 74  2 1  0 0  (0)
20 83 1331230 1 6  202.108.12.66 130.216.35.35  318 300  5 5  0 0  (0)
#EndData: localhost
```

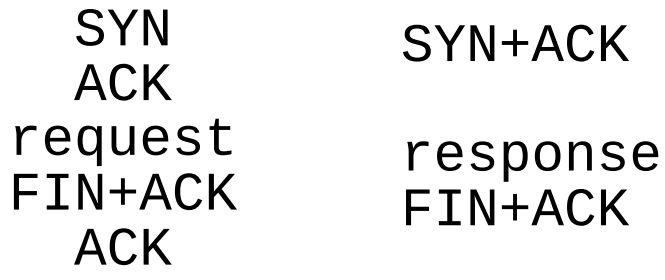
Name lookups

Zone transfer

- However ..
  - What are all the other flows?
  - What can we infer from the *To/From PDU* counts?

Successful DNS transactions: To/From pdu counts

- Example TCP connection:
  - 5 packets *To,* 3 packets *From*



- Actual connections depend on behaviour of nameserver and TCP stack. We often see ..
  - Length (2 bytes) sent as separate packet (before request or response)
  - Some TCP stacks ACK responses quickly, i.e. not piggybacked with FINs
  - (We never see the request piggybacked with the handshake ACK)

- Summary: successful transactions are just normal TCP connections

Unusual successful flows (1)

```
#Time: 00:45:00 Fri 30 Dec 2005 localhost Flows from 62986155 to 63076135
20 2024 62986181 1 6  132.205.96.87 130.216.1.2  27323 20129  374 265 \
   0 0  (5 2 10 7000  53 14 0 0  6 7 6 6 5 13 6 9 5 6 4 6 6 9 7 6 6 8 6 \
     7 6 5 7 6 7 6 5 7 6 6 6 8 7 7 6 6 8 7 6 6 11 7 7 8 8 6 6 9 7 6 6 7 8)
20 2025 62986248 1 6  132.205.96.87 130.216.1.1  27257 20195  373 266 \
   0 0  (5 2 10 7000  53 14 0 0  5 8 5 7 5 7 7 8 7 7 7 7 6 7 6 7 4 5 8 6 \
     7 7 6 8 6 4 8 7 6 8 7 7 6 6 9 6 7 7 8 8 7 6 9 8 7 7 9 6 6 7 7 4 8)
```

- These flows looked odd because their times are so small, only around $0.8ms$
  - They are a sequence of requests from 132.205.96.87 – a nameserver outside U Auckland,
  - to our nameservers, 130.216.1.2 and 130.216.1.1
  - For each of the two flows' 53 transactions, 6 packets were sent and 5 received
  - What we're observing here is the time for our servers to respond to incoming requests

Unusual successful flows (2)

```
#Time: 09:45:00 Sun 25 Dec 2005 localhost Flows from 23026114 to 23116188
20 460 15215803 1 6  130.216.165.190 192.175.48.1  14586 8943  165 132 \
   0 0  (5 2 10 7000  33 14 0 0  1604 1615 1604 1607 1601 1605 1611 1610 \
     1601 1611 1604 1616 1609 1616 1610 1611 1615 1605 1617 1610 1599 \
          1605 1604 1612 1606 1600 1606 1612 1620 1607 1612 1612 1605)
```

- This flow had 33 transactions, each taking about $161ms$
- 130.216.165.190 is *not* one of our local cacheing nameservers
  - It appears to be a (misconfigured?) user machine
- 192.175.48.1 is *prisoner.iana.org.*
  - *prisoner* is one of IANA's 'blackhole' servers
  - Those servers respond to inverse lookups of RFC 1918 *addresses*

And now, *unsuccessful* flows

```
#Time: 08:30:00 Thu 29 Dec 2005 localhost Flows from 57136202 to 57226182
20 1516 41774122 1 6  130.216.30.45 192.175.48.1  2664 1626  30 24  0 0 \
   (5 2 10 7000  6 14 0 0  1607 1641 1641 1626 1642 1630)
20 1541 42476533 1 6  210.21.230.2 130.216.1.2  480 360  8 6  0 0  (0)
20 1735 51723486 1 6  202.108.12.66 130.216.50.1  234 0 3 0  0 0  (0)
20 1750 52000507 1 6  130.216.1.1 216.26.160.6  513 1950  7 5  0 0  (0)
20 1752 52031729 1 6  130.216.1.1 216.26.160.5  450 1042  6 4  0 0 \
   (5 2 10 7000  1 14 0 0  1890)
20 1754 52086344 1 6  61.135.158.30 130.216.50.1  78 0  1 0  0 0  (0)
20 1764 52208013 1 6  202.108.12.67 130.216.50.1  234 0  3 0  0 0  (0)
20 1857 57158452 1 6  194.206.43.189 130.216.1.1  480 360  8 6  0 0  (0)
```

- Lots of requests are simply ignored
  - They're the ones that get 0 packets *From* their destination hosts
- Others exchange packets, but don't get matching requests/responses
  - They have to/from counts like 8 6 and 7 5
  - Needed to look their packet headers with tcpdump ..

Address Scans

```
#Time: 05:30:00 Sat 31 Dec 2005 localhost Flows from 73336183 to 73426163
20 2509 73408748 1 6  130.216.253.15 217.172.172.67  0 60  0 1  0 0  (0)
20 2510 73410133 1 6  130.216.25.8 217.172.172.67  0 60  0 1  0 0  (0)
20 2511 73410153 1 6  130.216.226.23 217.172.172.67  0 60  0 1  0 0  (0)
20 2512 73411127 1 6  130.216.86.100 217.172.172.67  0 60  0 1  0 0  (0)
20 2513 73411318 1 6  130.216.70.127 217.172.172.67  0 60  0 1  0 0  (0)
20 2514 73411505 1 6  130.216.17.49 217.172.172.67  0 60  0 1  0 0  (0)
20 2515 73412596 1 6  130.216.87.92 217.172.172.67  0 60  0 1  0 0  (0)
20 2516 73412614 1 6  130.216.178.24 217.172.172.67  0 60  0 1  0 0  (0)
20 2517 73412800 1 6  130.216.17.1 217.172.172.67  0 60  0 1  0 0  (0)
```

- This example shows a host scan through our network, 130.216/16
- SRL ruleset incorrectly gives 217.172.172.67 as the flow's destination
  - That's because the ruleset looks for *destination* port 53,
  - but these packets use port 53 as their *source*
- We see address scans like this every few days

DDoS attack (1)

```
#Time: 18:45:00 Sat 31 Dec 2005 localhost Flows from 78106188 to 78196168
20 2615 76520807 1 6  61.155.6.99 130.216.35.35  1134 900  18 15  0 0  (0)

61.135.158.29.2347 > 130.216.35.35.53: S 4954:4978(24) win 2048
130.216.35.35.53 > 61.135.158.29.2347: S 2514:2514(0) ack 4955 win 5840
61.135.158.29.2347 > 130.216.35.35.53: R 4955:4955(0) win 0

130.216.35.35.53 > 61.135.158.29.2347: S 2514:2514(0) ack 4955 win 5840
61.135.158.29.2347 > 130.216.35.35.53: R 4955:4955(0) win 0
  ...
```

- Remote host is trying to open TCP connections
  - Three connection attempts, using ports 2347, 2372 and 2394
  - External host sends SYN, we respond with SYN+ACK
  - External host terminates connection with RST
  - Meter is *outside* fi rewall – we retry the SYN+ACK four times
  - 6 packets sent, 5 received, $\times$ 3 ports $\Rightarrow$ 18 15
- Looks like a DDoS attack (source address spoofed)
  - We reply to the spoofed address, it responds with RST

DDoS attack (2)

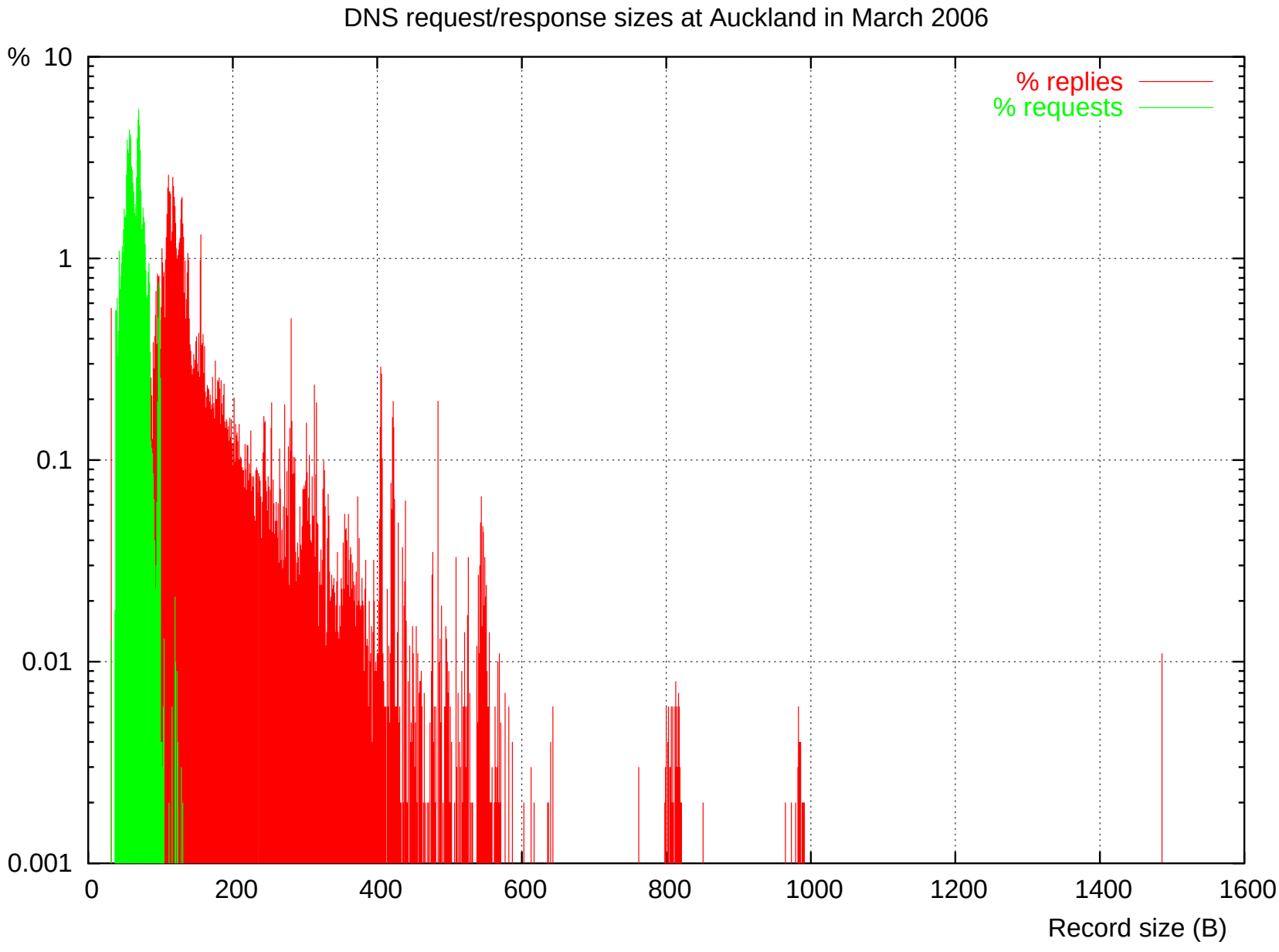
```
#Time: 10:30:00 Mon  2 Jan 2006 localhost Flows from 92416202 to 92506182
20 2630 76823167 1 6  61.135.158.29 130.216.35.35  2268 1800  36 30  0 0  (0
#Time: 11:15:00 Mon  2 Jan 2006 localhost Flows from 92686139 to 92776119
20 2630 76823167 1 6  61.135.158.29 130.216.35.35  1134 900  18 15  0 0  (0
#Time: 11:30:00 Mon  2 Jan 2006 localhost Flows from 92776118 to 92866199
20 2630 76823167 1 6  61.135.158.29 130.216.35.35  1134 900  18 15  0 0  (0
#Time: 12:45:00 Mon  2 Jan 2006 localhost Flows from 93226116 to 93316197
20 2630 76823167 1 6  61.135.158.29 130.216.35.35  1134 900  18 15  0 0  (0
#Time: 15:00:00 Mon  2 Jan 2006 localhost Flows from 94036130 to 94126110
20 2630 76823167 1 6  61.135.158.29 130.216.35.35  1134 900  18 15  0 0  (0
#Time: 15:45:00 Mon  2 Jan 2006 localhost Flows from 94306168 to 94396148
20 2630 76823167 1 6  61.135.158.29 130.216.35.35  1134 900  18 15  0 0  (0
```

- These attacks keep happening, every 1/4 to 3 hours
- Most – if not all – of them come from addresses within Chinese ISP address ranges
- They're part of the Internet 'background noise'

Traffic on TCP port 53: Conclusion

- At Auckland we see:
  - a steady trickle of DNS requests over TCP
  - a few zone transfers at scheduled intervals
  - a few common attack patterns
- Now we need to categorise the patterns so as to recognise and count them over a long period
- *We want to track TCP port 53 usage so as to discover whether DNS over TCP is increasing over time*
- Comment: RFC 2671 (EDNSO) allows DNS record sizes up to 65535, with or without fragmentation
  - We see lots of responses with > 512 bytes
  - Maybe a nameserver could send back a large response as a set of IPv4 fragments?

DNS record size distributions



DNS sizes, 2005 and 2006