# BU-2 How Protocols Work

16 June 2009


**Ray Tompkins**

Founder & CEO | gearbit


**SHARK**FEST **'09**

Stanford University

June 15-18, 2009

# How Protocols Work

Presentation Overview

- The Challenge
- Understanding How Protocols Work
- Understanding How Applications Fail

# The challenge:

- Companies today rely on computer based applications for every part of their business
- When these applications are slow or fail, the company is not able to perform in an efficient manner
- These application problems take time to resolve
- By understanding the underlying protocols, we can shorten the time it takes to resolve problems

# How Protocols Work

## Understanding How Protocols Work

# Importance of Understanding Protocol Operation

- If you don't understand how the primary protocols operate, you will not be successful in resolving network problems
- Most applications use the same key protocols
- Observing the operation of these protocols will help you to determine if the protocol is working correctly or operating improperly

# Application Flow

- DNS Lookup
- ARP for Address
- Establish TCP Connection
- Send Request
- Receive Response
- Close Connection

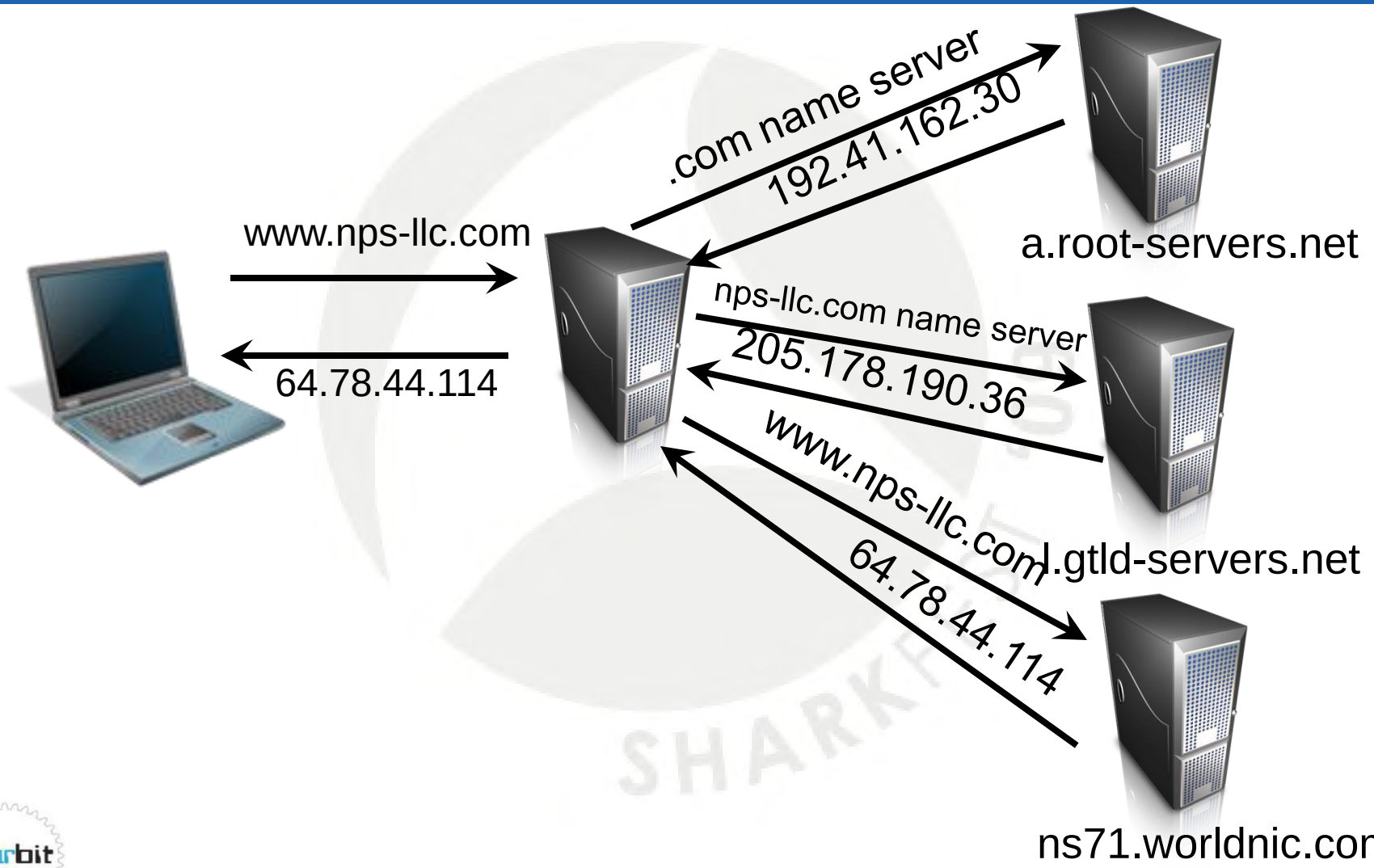
# Which Applications use this Flow?

- Web
- SQL
- Transaction processing
- Imaging
- Data Warehousing
- CRM
- ERP

# DNS Lookup

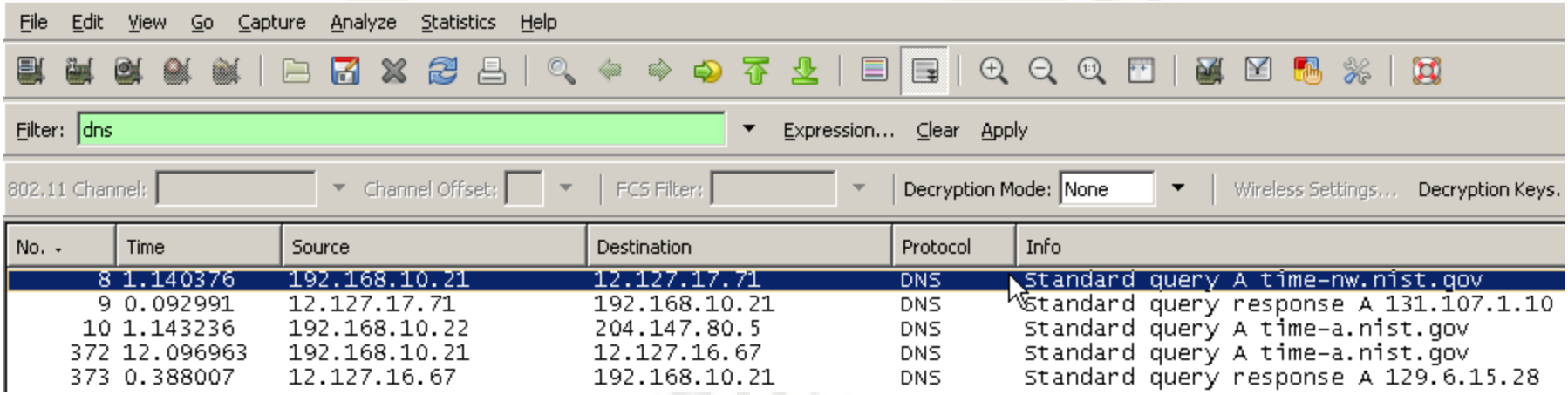
- Required to resolve DNS Name to IP Address
- Can not proceed with application until this is complete
- Slow DNS servers can impact all applications for a company
- Found many instances where a client computer is using the wrong DNS server

# DNS Lookup



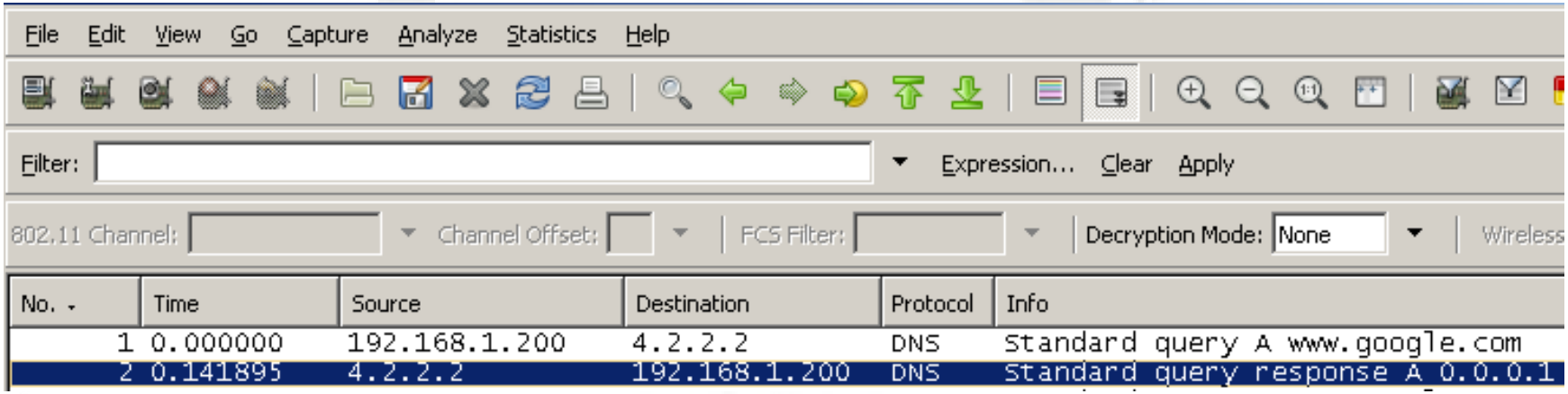
# DNS Lookup - Good

- Frame 8 – DNS Query For time-nw.nist.gov
- Frame 9 – DNS Response 131.107.1.10
- Response time 92.991 milliseconds



# DNS Lookup - Bad

- Frame 1 – DNS Query For www.google.com
- Frame 9 – DNS Response 0.0.0.1
- Will this get us to www.google.com?



# ARP for MAC Address

- Before we can send a frame, we MUST have the MAC address for the destination device
- The IP address is resolved to the MAC address using the Address Resolution Protocol (ARP)
- If we cannot resolve the IP address, or we get the wrong MAC address, we cannot get the frame to its destination

# ARP Request

```
⊟ Address Resolution Protocol (request)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    Sender MAC address: Netgear_01:05:51 (00:09:5b:01:05:51)
    Sender IP address: 192.168.10.21 (192.168.10.21)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.10.1 (192.168.10.1)
```

# ARP Response

```
⊟ Address Resolution Protocol (reply)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (0x0002)
    Sender MAC address: ZyxelCom_e5:c1:32 (00:a0:c5:e5:c1:32)
    Sender IP address: 192.168.10.1 (192.168.10.1)
    Target MAC address: Netgear_01:05:51 (00:09:5b:01:05:51)
    Target IP address: 192.168.10.21 (192.168.10.21)
```

# ARP Cache

```
C:\Documents and Settings\mpennac>arp -a

Interface: 10.0.0.114 --- 0x10003
  Internet Address      Physical Address      Type
  10.0.0.1              00-16-b6-85-8b-20     dynamic
  10.0.0.50             00-03-6d-1b-9d-a5     dynamic
  10.0.0.51             00-13-d4-b2-85-37     dynamic
  10.0.0.120            00-c0-17-a3-02-a1     dynamic
  10.0.0.121            00-c0-17-a1-00-6e     dynamic
```

# Route to Server

- Once we know the MAC address of the server or the default router the packets must be able to get from the client to the server
- The packets may always follow the same route, or take a different route each time
- If the packets are lost along this route the application will be slow
- If the packets are delayed along this route the application will be slow

# Route to Server

```
C:\Documents and Settings\mpennac>tracert new.networkprotocolspecialists.com

Tracing route to new.networkprotocolspecialists.com [69.89.31.170]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  rtr-rv082.nps-llc.com.local [10.0.0.1]
  2    <1 ms    <1 ms    <1 ms  h-66-134-176-241.sttnwaho.covad.net [66.134.176.
241]
  3    12 ms    10 ms     9 ms  172.31.255.253
  4    13 ms     9 ms    10 ms  192.168.23.65
  5    20 ms     8 ms    10 ms  66.236.9.169.ptr.us.xo.net [66.236.9.169]
  6    49 ms     9 ms    10 ms  p4-3-0.mar2.seattle-wa.us.xo.net [207.88.83.141]

  7     8 ms    10 ms    10 ms  p5-1-0-0.rar2.seattle-wa.us.xo.net [65.106.0.137
]
  8    88 ms    38 ms    41 ms  p5-0-0-0.rar1.denver-co.us.xo.net [65.106.0.54]

  9    52 ms    52 ms    52 ms  p0-0-0-0.mar1.saltlake-ut.us.xo.net [65.106.6.82
]
 10    54 ms    51 ms    53 ms  p1-0.chr1.saltlake-ut.us.xo.net [207.88.83.102]

 11    51 ms    51 ms    53 ms  ip65-46-48-66.z48-46-65.customer.algx.net [65.46
.48.66]
 12    54 ms    52 ms    52 ms  box370.bluehost.com [69.89.31.170]

Trace complete.
```

# Establish the Connection

- For this discussion, we will focus on TCP based applications
- Before data can be sent over a TCP connection, the connection must first be established
- This is done with the Three-way Handshake
  - Client sends a TCP SYN packet
  - Server responds with at TCP SYN/ACK
  - Client responds with a TCP ACK
- The delta time between the TCP SYN and the TCP SYN/ACK represents the roundtrip delay of the circuit

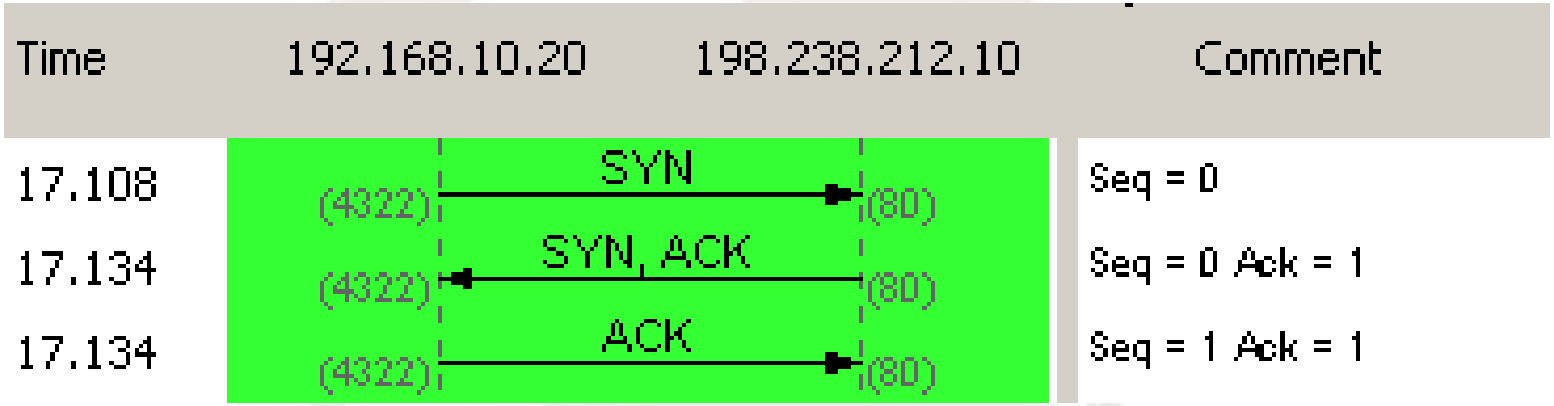
# Establish the Connection

- The device establishing the connection will send a beginning TCP sequence number
- It is important to note that Wireshark converts this to a relative sequence number
- For example
  - The initiating device may use a sequence number of 253875
  - Wireshark will display it as sequence number 0

# Establish the Connection

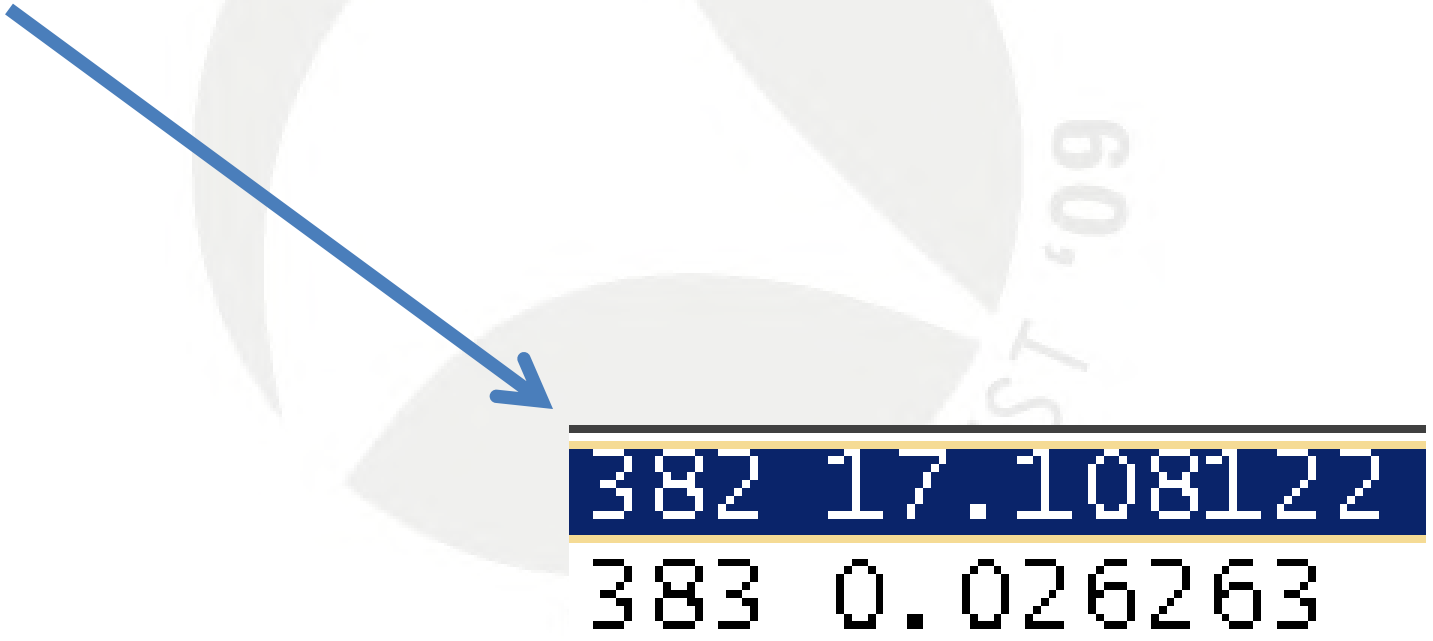
- The server will respond with its own starting TCP sequence number and an Acknowledgement Number equal to the initiator's Sequence Number plus 1
- The initiator will respond with an Acknowledgement Packet with an Acknowledgement Number equal to the server's Sequence Number plus 1
- Once this occurs, the connection is established.

# Establish Connection



# Establish Connection

| No. | Time | Source | Destination | Protocol | Info |
|---|---|---|---|---|---|
| 382 | 17.108122 | 192.168.10.20 | 198.238.212.10 | TCP | trim-event > http [SYN] |
| 383 | 0.026263 | 198.238.212.10 | 192.168.10.20 | TCP | http > trim-event [SYN, |



# Sending the Request

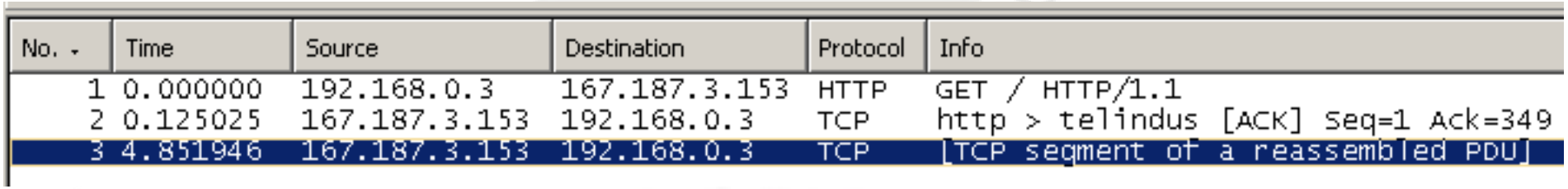
- Now that the connection is established, we send our request.
- This request may be for:
  - An object on a webpage
  - A field in a database
  - A server side transaction
  - A segment of a file from a file server
- In most cases, we will halt further processing until we receive a response to this request

# TCP ACK

- If it takes a long time to get a response to the request, TCP may acknowledge the TCP data segment, even though there is no data to return
- This indicates that the packet was received by the server, but it is taking longer that 200ms to return the requested data
- This tells us that the network is working fine, but the server may be slow

# Send Request – Get Response

- The example below shows a HTTP Get and HTTP Get Response
- Frame 1 – HTTP GET
- Frame 2 – TCP Ack from the server
- Frame 3 – Data Frame from the server
- The Ack indicates that the TCP frame reached the server within 125 milliseconds, but it took 4.8 seconds to get the data



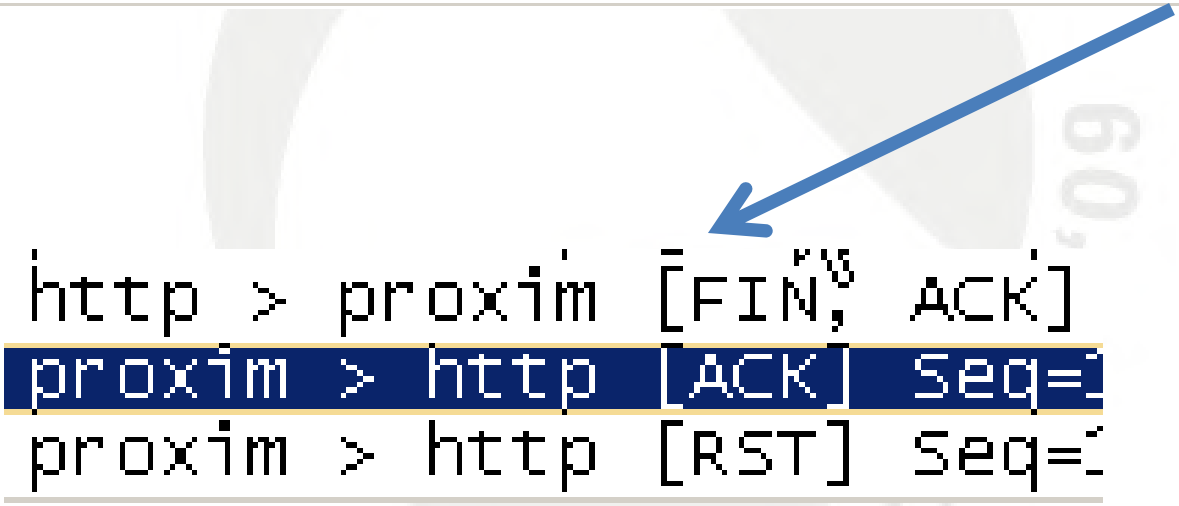
| No. | Time | Source | Destination | Protocol | Info |
|---|---|---|---|---|---|
| 1 | 0.000000 | 192.168.0.3 | 167.187.3.153 | HTTP | GET / HTTP/1.1 |
| 2 | 0.125025 | 167.187.3.153 | 192.168.0.3 | TCP | http > telindus [ACK] Seq=1 Ack=349 |
| 3 | 4.851946 | 167.187.3.153 | 192.168.0.3 | TCP | [TCP segment of a reassembled PDU] |

# Closing the Connection

- After the requests and responses are complete the connection is closed
- There may be multiple request/response pairs for a single connection
- In most cases the application does not pause for the connection close process

# Close Connection

| | | | | | |
|---|---|---|---|---|---|
| 208 | 0.144945 | 192.168.0.3 | 128.11.10.249 | TCP | proxim > http [ACK] Seq=1843 |
| 272 | 1.060079 | 128.11.10.249 | 192.168.0.3 | TCP | http > proxim [FIN, ACK] Seq= |
| 273 | 0.000150 | 192.168.0.3 | 128.11.10.249 | TCP | proxim > http [ACK] Seq=1843 |
| 283 | 4.854511 | 192.168.0.3 | 128.11.10.249 | TCP | proxim > http [RST] Seq=1843 |



# Understanding How Applications Fail

# DNS

- Slow DNS Server
- Wrong DNS Server
- No A record for name being queried
- Bad response by DNS server
- Slow PTR record lookup
- Packet loss between client and DNS server

# ARP

- Duplicate IP Addresses
- Proxy ARP
- No response from server
- Wrong MAC address returned

# Routing

- Bad routes
- Packet loss along route
- High delay along route

# Connection Setup

- Port not open on server
- Server slow to respond to TCP connection request
- Load balancer problems
- Packet loss during setup. TCP retransmission time for connection setup is 3 seconds!

# Request/Response

- Server slow to respond to request
- Request packet lost
- Packet loss
- Server can’t find requested data

# Closing the Connection

- Client uses Reset to close connection, this works but not the right way to close a connection
- Client leaves connection open for a long time, using up valuable resources on the server
- FIN packets are lost

# How to contact us at gearbit

Ray Tompkins

info09@gearbit.com
www.gearbit.com