# CS615 - System Administration

# Networking II

Department of Computer Science
Stevens Institute of Technology
Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/615/

# Get your instruments and play along!

Start a FreeBSD instance:
`ami-0de268ac2498ba33d`

# HW1 Comments

- provide more details in your README
- learn to format your README in a readable manner
- carefully review the problem statement
- if anything is unclear, don't guess; ask on the mailing list
- pay attention to the discussions on the mailing list

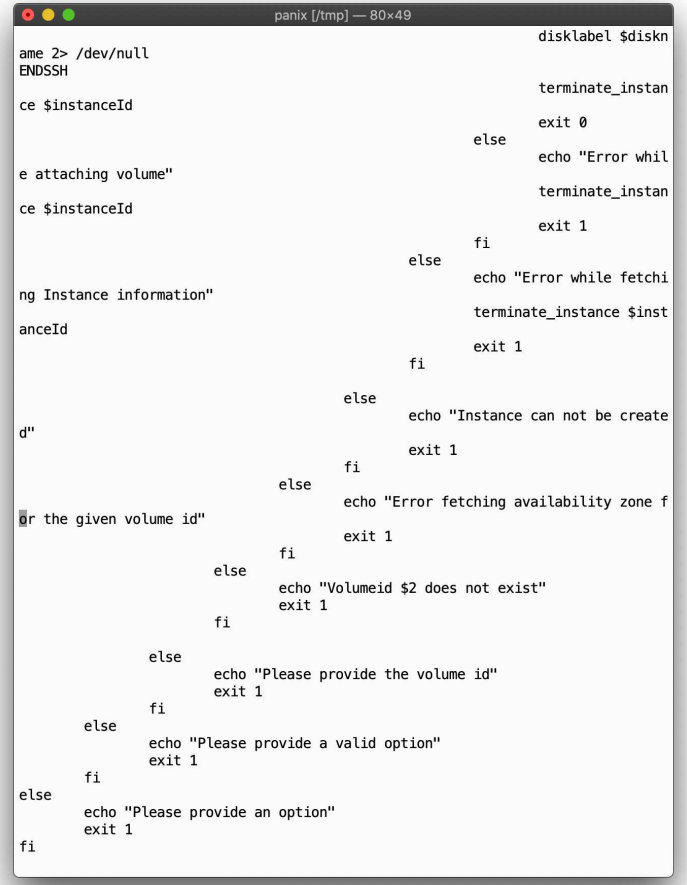
# HW1 Comments

- incremental improvements (MVP); make it work, then iterate (but remain able and willing to overhaul if needed)
- make no assumptions about the directory from which your program will be invoked
- avoid temporary files wherever possible
- consider efficiency (e.g., validate first, then create an instance)
- (understand how to) verify that your program works
- write your tool to look and feel like any other system tool

# HW1 Comments

- do not hardcode information that is specific to your own account
- do not hardcode a single AMI; you need to be able to support multiple availability zones
- use an exit handler to terminate the instance after your program completes (on success *or* error)
- do not specify `-i` to `ssh(1)`

# HW1 Comments: avoid waterfall code



# HW1 Comments

Writing shell scripts is not very different from writing programs in other languages.

- use `getopt(1)` to parse command-line arguments
- use functions
- check commands and functions for return values

Some useful links:

- `https://www.netmeister.org/blog/writing-shell-scripts.html`
- `https://www.netmeister.org/blog/mktemp.html`
- `https://github.com/koalaman/shellcheck`
- `https://google.github.io/styleguide/shell.xml`

# Layer 1: Repeater Hub


Half-duplex, cheap, obsolete.

# Layer 1:

Black Team:

Error Detection and Correction on the Link Layer
`https://is.gd/6xYRGH`

# Layer 2: Network Switch


MAC bridge, full-duplex, segmentation, CIDR, STP

`https://is.gd/HPhuRG`

# Layer 3: Router


connect networks, forward packets, routing tables, BGP

# A simple example

```
$ telnet www.yahoo.com 80
```

# A simple example

```
$ telnet www.yahoo.com 80
Trying 98.138.219.232...
Connected to atsv2-fp-shed.wg1.b.yahoo.com.
Escape character is '^]'.
HEAD / HTTP/1.0
```

# A simple example

```
$ telnet www.yahoo.com 80
Trying 98.138.219.232...
Connected to atsv2-fp-shed.wg1.b.yahoo.com.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.0 200 OK
Date: Mon, 04 Mar 2019 17:41:59 GMT
Via: http/1.1 media-router-fp1010.prod.media.ne1.yahoo.com
Server: ATS
[...]
```

# A simple example

What exactly happens?

# Let's collect some data...

```
laptop$ ssh ec2-user@<instance-name>
$ su
script commands.out
ifconfig -a; route -n get default
cat /etc/resolv.conf
tcpdump -w tcpdump.out port not 22 >&/dev/null &
arp -d -a
ping -n -c 3 8.8.8.8
ktrace -i telnet www.yahoo.com 80
HEAD / HTTP/1.0

kill %1
kdump > kdump.out
chmod a+r kdump.out
exit
laptop$ scp ec2-user@<instance-name>:*out /tmp/
```

# A simple example

What exactly happens?

- local host connects to remote host
- sends command
- receives data

# A simple example

How exactly do we connect to the remote host?

- look up hostname
- open connection to IP address

# A simple example

How exactly do we look up a hostname?

# A simple example

```
$ ktrace -i telnet www.yahoo.com 80
Trying 72.30.35.9...
Connected to atsv2-fp-shed.wg1.b.yahoo.com.
Escape character is '^]'.
HEAD / HTTP/1.0

[...]
$ kdump >trace
```

# ...open a few files...

```
[...]
   735 ktrace   RET   execve -1 errno 2 No such file or directory
   735 ktrace   CALL  execve(0xbfbfe7e0,0xbfbfed00,0xbfbfed10)
   735 ktrace   NAMI  "/usr/bin/telnet"
   735 ktrace   NAMI  "/libexec/ld-elf.so.1"
   735 telnet   RET   execve JUSTRETURN
[...]
   735 telnet   CALL  open(0x80066edc5,0x100000<O_RDONLY|O_CLOEXEC>)
   735 telnet   NAMI  "/etc/nsswitch.conf"
   735 telnet   RET   open 3
[...]
   735 telnet   CALL  open(0x800671afd,0x100000<O_RDONLY|O_CLOEXEC>)
   735 telnet   NAMI  "/etc/hosts"
   735 telnet   RET   open 3
[...]
   735 telnet   CALL  open(0x80066e6b5,0x100000<O_RDONLY|O_CLOEXEC>)
   735 telnet   NAMI  "/etc/resolv.conf"
   735 telnet   RET   open 3
[...]
   735 telnet   CALL  read(0x3,0x800c3be40,0x8000)
   735 telnet   GIO   fd 3 read 70 bytes
       "# Generated by resolvconf
        search ec2.internal
        nameserver 172.16.0.23
```

# ... query a DNS server ...

```
[...]
   735 telnet   CALL  socket(PF_INET,0x10000002<SOCK_DGRAM|SOCK_CLOEXEC>,IPPROTO_IP)
   735 telnet   RET   socket 3
   735 telnet   CALL  connect(0x3,0x800a43914,0x10)
   735 telnet   STRU  struct sockaddr { AF_INET, 172.16.0.23:53 }
   735 telnet   RET   connect 0
   735 telnet   CALL  sendto(0x3,0x800c96400,0x1f,0,0,0)
   735 telnet   GIO   fd 3 wrote 31 bytes
       0x0000 e614 0100 0001 0000 0000 0000 0377 e777       |.............www|
       0x0010 0579 6168 6f6f 0363 6f6d 0000 0100 01         |.yahoo.com.....|
[...]
   735 telnet   CALL  recvfrom(0x3,0x800c71e00,0x10000,0,0x7fffffffd640,0x7fffffffd22
   735 telnet   GIO   fd 3 read 97 bytes
       0x0000 e614 8180 0001 0003 0000 0000 0377 7777       |.............www|
       0x0010 0579 6168 6f6f 0363 6f6d 0000 0100 01c0       |.yahoo.com......|
       0x0020 0c00 0500 0100 0000 3c00 160d 6174 7376       |........<...atsv|
       0x0030 322d 6670 2d73 6865 6403 7767 3101 62c0       |2-fp-shed.wg1.b.|
       0x0040 10c0 2b00 0100 0100 0000 3c00 0448 1e23       |..+.......<..H.#|
       0x0050 09c0 2b00 0100 0100 0000 3c00 0448 1e23       |..+.......<..H.#|
       0x0060 0a                                            |.|
[...]
```

# A simple example

How exactly do we look up a hostname?

- look up various local files
- open a connection to a DNS server's IP
- ask DNS server to resolve hostname
- get back IP

And then?

# ...communicate with the remote host...

```
   735 telnet   GIO   fd 1 wrote 21 bytes
       "Trying 72.30.35.9...
       "
   735 telnet   RET   write 21/0x15
   735 telnet   CALL  socket(PF_INET,0x1<SOCK_STREAM>,IPPROTO_TCP)
   735 telnet   CALL  connect(0x3,0x8002650f0,0x10)
   735 telnet   STRU  struct sockaddr { AF_INET, 72.30.35.9:80 }
[...]
   918 telnet   GIO   fd 0 read 16 bytes
       "HEAD / HTTP/1.0
       "
   918 telnet   RET   read 16/0x10
   918 telnet   CALL  select(0x4,0x80025e1d8,0x80025e1c8,0x80025e1d0,0x229058)
   918 telnet   RET   select 1
   918 telnet   CALL  sendto(0x3,0x226490,0x11,0,0,0))
   918 telnet   GIO   fd 3 wrote 17 bytes
       "HEAD / HTTP/1.0\r
       "
[...]
   918 telnet   RET   select 1
   918 telnet   CALL  recvfrom(0x3,0x226040,0x400,0,0,0)
   918 telnet   GIO   fd 3 read 324 bytes
       "HTTP/1.0 200 OK\r
        Date: Mon, 04 Mar 2019 17:44:09 GMT\r
```

# Ok, so how does this work?

- determine which nameserver to query
- ask who has a route to the nameserver
- open socket to well defined port on remote IP
- send queries
- open socket to requested port on remote IP

# A simple example

Finding the next hop:

```
$ tcpdump -t -n -r /tmp/tcpdump.out arp
reading from file /tmp/tcpdump.out, link-type EN10MB (Ethernet)
ARP, Request who-has 10.183.114.1 tell 10.183.114.37, length 28
ARP, Reply 10.183.114.1 is-at fe:ff:ff:ff:ff:ff, length 28
ARP, Request who-has 10.183.114.37 tell 10.183.114.1, length 28
ARP, Reply 10.183.114.37 is-at 22:00:0a:b7:72:25, length 28
```

# A simple example

Performing the DNS query:

```
$ tcpdump -t -n -r tcpdump.out udp port 53
reading from file tcpdump.out, link-type EN10MB (Ethernet)
IP 10.183.114.37.53383 > 172.16.0.23.53: 20378+ A? www.yahoo.com. (31)
IP 172.16.0.23.53 > 10.183.114.37.53383: 20378 5/0/0
CNAME atsv2-fp-shed.wg1.b.yahoo.com., A 72.30.35.9, A 72.30.35.10,
A 98.138.219.231, A 98.138.219.232 (129)
IP 10.183.114.37.10551 > 172.16.0.23.53: 60792+ AAAA? www.yahoo.com. (31)
IP 172.16.0.23.53 > 10.183.114.37.10551: 60792 5/0/0 CNAME atsv2-fp-shed.wg1.b.yahoo.
AAAA 2001:4998:58:1836::11, AAAA 2001:4998:44:41d::3,
AAAA 2001:4998:44:41d::4, AAAA 2001:4998:58:1836::10 (177)
```

# A simple example

Establishing the connection to the server:

```
$ tcpdump -t -n -r tcpdump.out tcp port 80
IP 10.183.114.37.45403 > 72.30.35.9.80: Flags [S], seq 5028151, win 65535,
options [mss 1460,nop,wscale 6,sackOK,TS val 598758437 ecr 0], length 0
IP 72.30.35.9.80 > 10.183.114.37.45403: Flags [S.], seq 1983855029, ack 5028152, win
options [mss 1460,sackOK,TS val 1888595968 ecr 598758437,nop,wscale 8], length 0
IP 10.183.114.37.45403 > 72.30.35.9.80: Flags [.], ack 1, win 1026,
options [nop,nop,TS val 598758465 ecr 1888595968], length 0
```

# A simple example

Sending the HTTP request:

```
IP 10.183.114.37.45403 > 72.30.35.9.80: Flags [P.], seq 1:18, ack 1, win 1026,
options [nop,nop,TS val 598762755 ecr 1888595968], length 17: HTTP: HEAD / HTTP/1.0
IP 72.30.35.9.80 > 10.183.114.37.45403: Flags [.], ack 18, win 57,
options [nop,nop,TS val 1888600285 ecr 598762755], length 0
IP 10.183.114.37.45403 > 72.30.35.9.80: Flags [P.], seq 18:20, ack 1, win 1026,
options [nop,nop,TS val 598764209 ecr 1888600285], length 2: HTTP
IP 72.30.35.9.80 > 10.183.114.37.45403: Flags [.], ack 20, win 57,
options [nop,nop,TS val 1888601739 ecr 598764209], length 0
```

# A simple example

Receiving the HTTP response:

```
IP 72.30.35.9.80 > 10.183.114.37.45403: Flags [P.], seq 1:325, ack 20, win 57,
options [nop,nop,TS val 1888601741 ecr 598764209], length 324: HTTP: HTTP/1.0 200 OK
```

# A simple example

Terminating the connection:

```
IP 72.30.35.9.80 > 10.183.114.37.45403: Flags [F.], seq 325, ack 20, win 57,
options [nop,nop,TS val 1888601741 ecr 598764209], length 0
IP 10.183.114.37.45403 > 72.30.35.9.80: Flags [.], ack 326, win 1021,
options [nop,nop,TS val 598764236 ecr 1888601741], length 0
IP 10.183.114.37.45403 > 72.30.35.9.80: Flags [F.], seq 20, ack 326, win 1026,
options [nop,nop,TS val 598764236 ecr 1888601741], length 0
IP 72.30.35.9.80 > 10.183.114.37.45403: Flags [.], ack 21, win 57,
options [nop,nop,TS val 1888601768 ecr 598764236], length 0
```

# Notables from this simple example

“Simple” is, as usual, relative.

# Notables from this simple example

"Simple" is, as usual, relative.

- host configuration assumed
- network architecture (internal or across the internet) not relevant (here)
- even simple examples cross multiple layers and protocols (HTTP, DNS; TCP, UDP, ARP)
- we haven't even scratched the surface

# TCP/IP Basics: Protocol Layers

| | Layer | Function |
|---|---|---|
| 4. | Application Layer | End-User application programs |
| 3. | Transport Layer | Delivery of data to applications |
| 2. | Network Layer | Basic communication, addressing, and routing |
| 1. | Link Layer | Network Hardware and device drivers |
| | Physical Layer | Cable or physical medium |

Examples of protocols for each layer:

- Simple Mail Transfer Protocol (RFC 821)
  Hypertext Transfer Protocol (RFC 2616)
- Transmission Control Protocol (RFC 793, tcp(4))
  User Datagram Protocol (RFC 768; udp(4))
- Internet Protocol (RFC 791; ip(4))
  Internet Control Message Protocol (RFC 792; icmp(4))
- Address Resolution Protocol (RFC 826; arp(4))

# TCP/IP Basics: Protocol Layers (OSI Model)

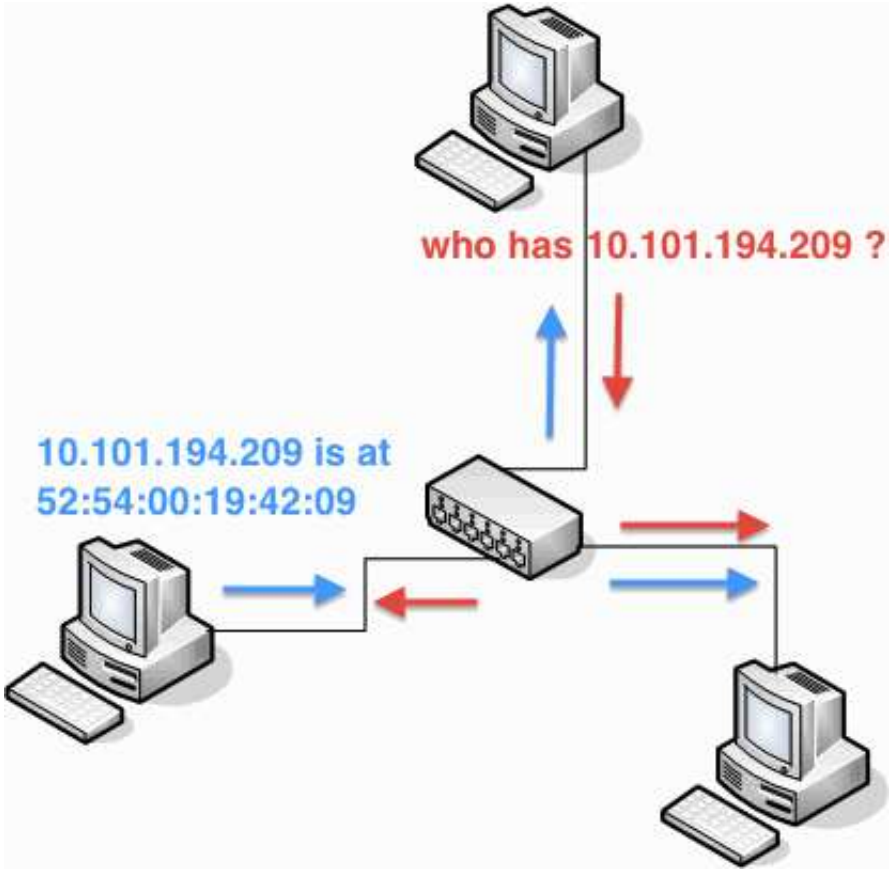
| OSI Model | | | |
|---|---|---|---|
| | **Data unit** | **Layer** | **Function** |
| **Host layers** | Data | 7. Application | Network process to application |
| | | 6. Presentation | Data representation, encryption and decryption, convert machine dependent data to machine independent data |
| | | 5. Session | Interhost communication, managing sessions between applications |
| | Segments | 4. Transport | End-to-end connections, reliability and flow control |
| **Media layers** | Packet/Datagram | 3. Network | Path determination and logical addressing |
| | Frame | 2. Data link | Physical addressing |
| | Bit | 1. Physical | Media, signal and binary transmission |

# TCP/IP Basics: ARP

Ethernet Address Resolution Protocol
– or –
Converting Network Protocol Addresses to 48-bit Ethernet Address for Transmission on Ethernet Hardware

```
$ arp -a
falcon.srcit.stevens-tech.edu (155.246.89.89) at 00:07:e9:09:ca:10 [ether] on eth0
grohl.srcit.stevens-tech.edu (155.246.89.9) at 00:16:3e:cf:6b:5b [ether] on eth0
hoth.srcit.stevens-tech.edu (155.246.89.10) at 00:1e:68:8e:79:d8 [ether] on eth0
cinema.srcit.stevens-tech.edu (155.246.89.67) at 00:25:90:1e:05:51 [ether] on eth0
vlan16.cc.stevens-tech.edu (155.246.89.1) at 00:00:5e:00:01:02 [ether] on eth0
vader.srcit.stevens-tech.edu (155.246.89.5) at 00:23:8b:a9:dd:60 [ether] on eth0
nirvana.phy.stevens-tech.edu (155.246.89.33) at 00:1e:68:0f:99:a2 [ether] on eth0
```

# TCP/IP Basics: ARP



# TCP/IP Basics: ARP

Ethernet Address Resolution Protocol
– or –
Converting Network Protocol Addresses to 48-bit Ethernet Address for Transmission on Ethernet Hardware

```
ARP, Request who-has 10.114.62.1 tell 10.114.63.209, length 28
ARP, Reply 10.114.62.1 is-at fe:ff:ff:ff:ff:ff, length 28
ARP, Request who-has 10.114.63.209 (ff:ff:ff:ff:ff:ff) tell 0.0.0.0, length 28
ARP, Reply 10.114.63.209 is-at 12:31:3d:04:30:23, length 28
ARP, Request who-has 10.114.63.209 (ff:ff:ff:ff:ff:ff) tell 0.0.0.0, length 28
ARP, Reply 10.114.63.209 is-at 12:31:3d:04:30:23, length 28
ARP, Request who-has 10.114.63.209 (ff:ff:ff:ff:ff:ff) tell 0.0.0.0, length 28
ARP, Reply 10.114.63.209 is-at 12:31:3d:04:30:23, length 28
```

# TCP/IP Basics: ND

## Neighbor Discovery Protocol

```
$ ndp -n -a
Neighbor                                Linklayer Address  Netif Expire      S Flags
2001:470:30:84:e276:63ff:fe72:3900  e0:76:63:72:39:00  xennet0 permanent R
fe80::21b:21ff:fe45:bf54%xennet0    00:1b:21:45:bf:54  xennet0 21m52s    S R
fe80::21b:21ff:fe7a:7269%xennet0    00:1b:21:7a:72:69  xennet0 23h59m59s S R
fe80::e276:63ff:fe72:3900%xennet0   e0:76:63:72:39:00  xennet0 permanent R
fe80::1%lo0                          (incomplete)      lo0     permanent R
$
```

# TCP/IP Basics: ND

## Neighbor Discovery Protocol

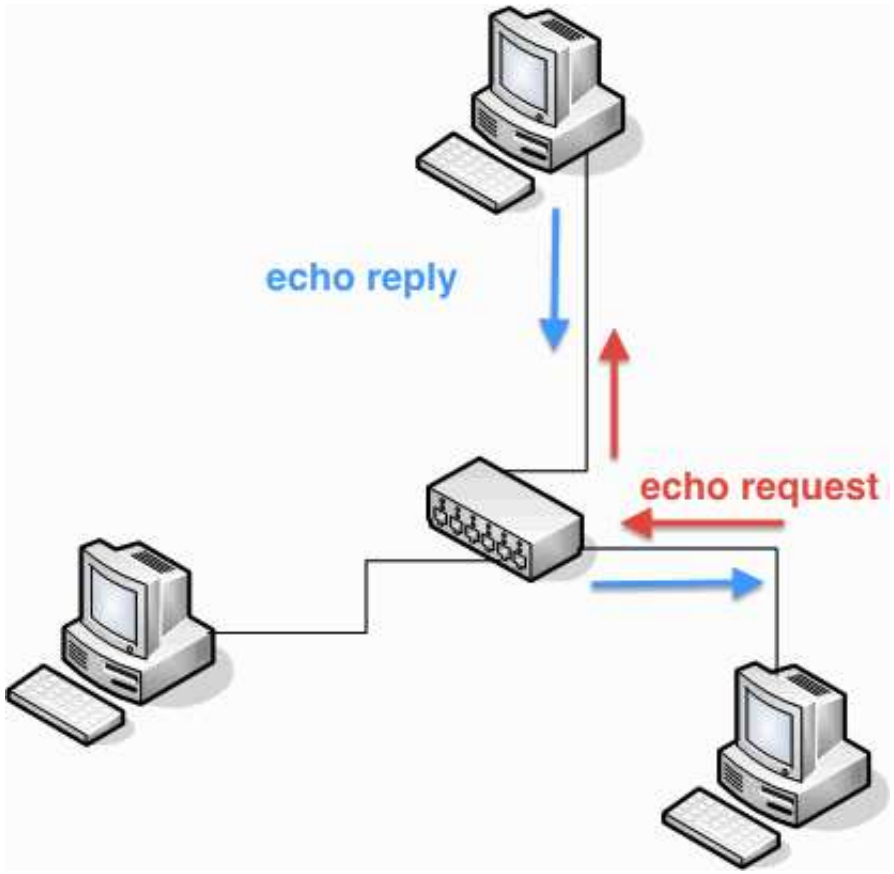
```
IP6 fe80::21b:21ff:fe7a:7269 > ff02::1:ff62:3400: ICMP6,
        neighbor solicitation, who has 2001:470:30:84:e276:63ff:fe62:3400, length 32
IP6 2001:470:30:84:e276:63ff:fe72:3900 > ff02::1:ff7a:7269: ICMP6,
        neighbor solicitation, who has fe80::21b:21ff:fe7a:7269, length 32
IP6 fe80::21b:21ff:fe7a:7269 > 2001:470:30:84:e276:63ff:fe72:3900:
        ICMP6, neighbor advertisement, tgt is fe80::21b:21ff:fe7a:7269, length 32
```

# TCP/IP Basics: ICMP

## Internet Control Message Protocol

```
$ ping -c 3 www.yahoo.com
PING any-fp.wa1.b.yahoo.com (67.195.160.76): 56 data bytes
64 bytes from 67.195.160.76: icmp_seq=0 ttl=53 time=30.888 ms
64 bytes from 67.195.160.76: icmp_seq=1 ttl=53 time=23.193 ms
64 bytes from 67.195.160.76: icmp_seq=2 ttl=53 time=25.433 ms

----any-fp.wa1.b.yahoo.com PING Statistics----
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 23.193/26.505/30.888/3.958 ms
$
```

# TCP/IP Basics: ICMP: Ping



# TCP/IP Basics: ICMP

## Internet Control Message Protocol

```
$ tcpdump -r tcpdump.out -n icmp
IP 10.234.84.220 > 207.237.69.79: ICMP echo request
IP 207.237.69.79 > 10.234.84.220: ICMP echo reply
IP 10.234.84.220 > 207.237.69.79: ICMP echo request
IP 207.237.69.79 > 10.234.84.220: ICMP echo reply
IP 10.234.84.220 > 207.237.69.79: ICMP echo request
IP 207.237.69.79 > 10.234.84.220: ICMP echo reply
```

# TCP/IP Basics: ICMP6

Internet Control Message Protocol for IPv6

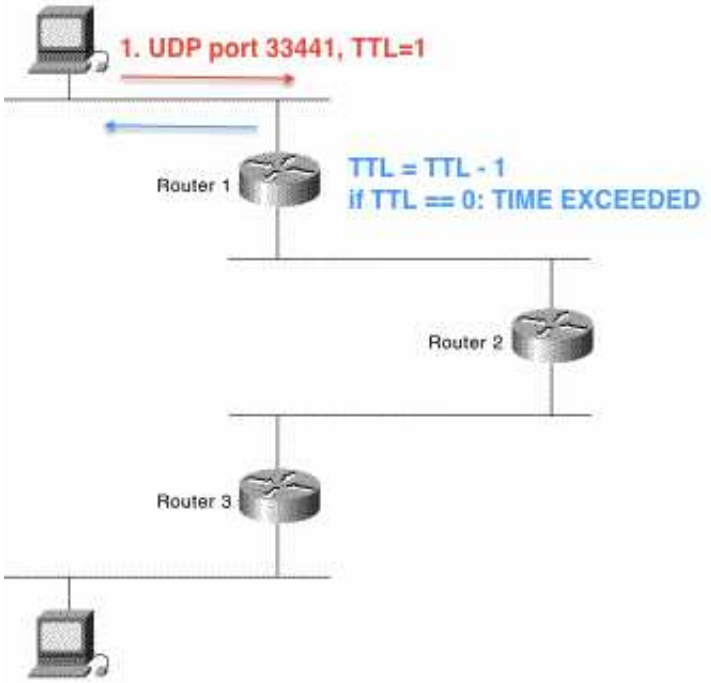
```
$ ping6 -c 3 www.netbsd.org
PING6(56=40+8+8 bytes) 2001:470:30:84:204:d7b0:0:1 -->
                       2001:4f8:3:7:2e0:81ff:fe52:9a6b
16 bytes from 2001:4f8:3:7:2e0:81ff:fe52:9a6b, icmp_seq=0 hlim=57 time=74.316 ms
16 bytes from 2001:4f8:3:7:2e0:81ff:fe52:9a6b, icmp_seq=1 hlim=57 time=71.260 ms
16 bytes from 2001:4f8:3:7:2e0:81ff:fe52:9a6b, icmp_seq=2 hlim=57 time=71.321 ms

--- www.netbsd.org ping6 statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 71.260/72.299/74.316/1.747 ms
```

# TCP/IP Basics: ICMP6

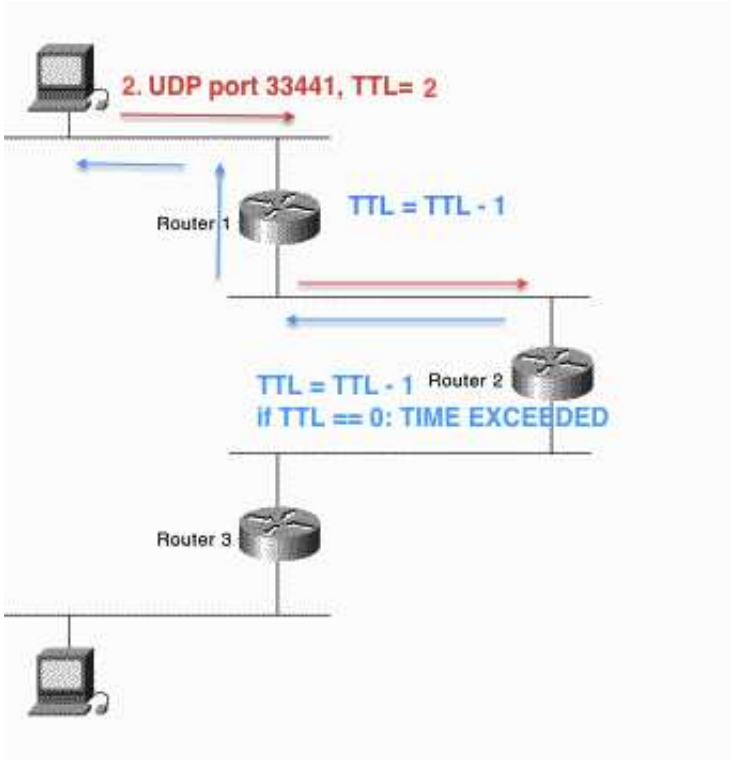
## Internet Control Message Protocol for IPv6

```
IP6 2001:470:30:84:204:d7b0:0:1 >
   2001:4f8:3:7:2e0:81ff:fe52:9a6b: ICMP6, echo reque st, seq 0, length 16
IP6 2001:4f8:3:7:2e0:81ff:fe52:9a6b >
   2001:470:30:84:204:d7b0:0:1: ICMP6, echo reply , seq 0, length 16
IP6 2001:470:30:84:204:d7b0:0:1 >
   2001:4f8:3:7:2e0:81ff:fe52:9a6b: ICMP6, echo request, seq 1, length 16
IP6 2001:4f8:3:7:2e0:81ff:fe52:9a6b >
   2001:470:30:84:204:d7b0:0:1: ICMP6, echo reply , seq 1, length 16
IP6 2001:470:30:84:204:d7b0:0:1 >
   2001:4f8:3:7:2e0:81ff:fe52:9a6b: ICMP6, echo reque st, seq 2, length 16
IP6 2001:4f8:3:7:2e0:81ff:fe52:9a6b >
   2001:470:30:84:204:d7b0:0:1: ICMP6, echo reply , seq 2, length 16
```

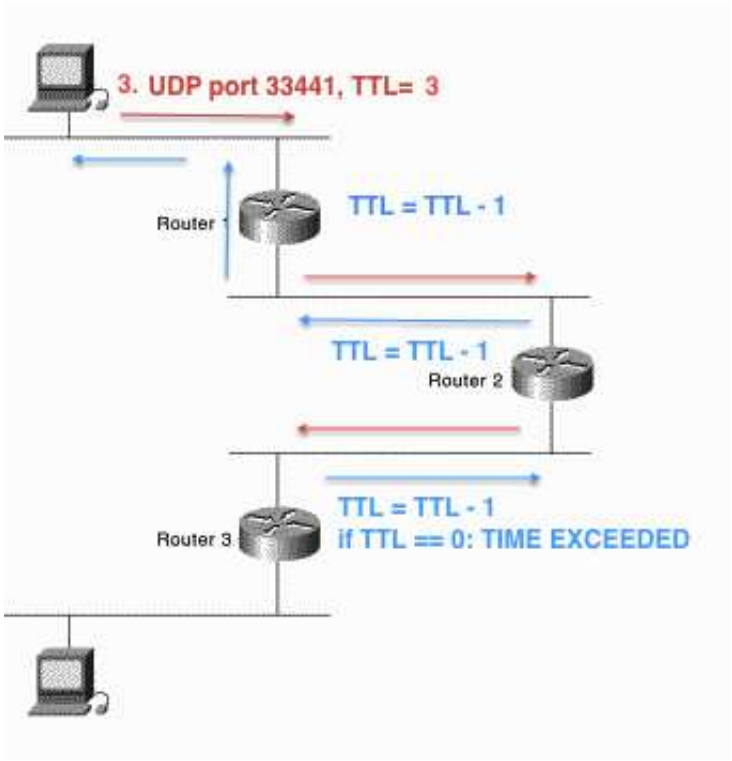
# TCP/IP Basics: ICMP: Traceroute



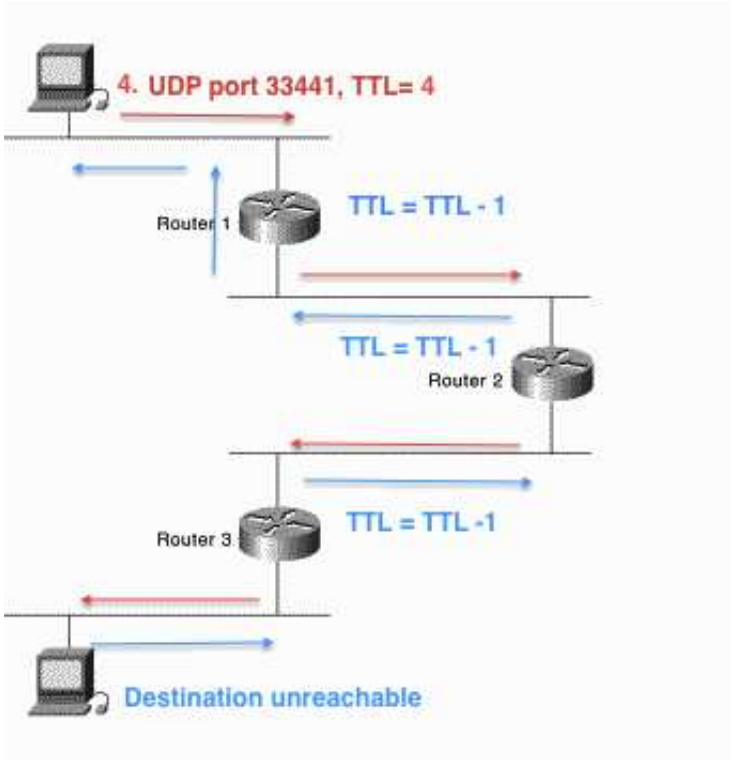
# TCP/IP Basics: ICMP: Traceroute



# TCP/IP Basics: ICMP: Traceroute



# TCP/IP Basics: ICMP: Traceroute



# TCP/IP Basics: ICMP

## Internet Control Message Protocol

```
$ traceroute www.netbsd.org
traceroute to www.netbsd.org (204.152.190.12), 64 hops max, 40 byte packets
 1  eth2-3a.core1.nav.nyc.access.net (166.84.0.1)  0.256 ms  0.165 ms 0.181 ms
 2  l3v1.nyc.access.net (166.84.66.14)  1.570 ms  1.556 ms  1.437 ms
 3  gige-g3-3.core1.nyc4.he.net (209.51.171.25)  4.963 ms  2.422 ms  1.457 ms
 4  10gigabitethernet2-3.core1.ash1.he.net (72.52.92.86)  8.423 ms  8.769 ms  7.683 m
 5  10gigabitethernet1-2.core1.atl1.he.net (184.105.213.110)  21.898 ms 19.647 ms  19
 6  isc.gige-g2-1.core1.atl1.he.net (216.66.0.50)  77.465 ms  77.921 ms 80.519 ms
 7  iana.r1.atl1.isc.org (199.6.12.1)  77.302 ms  78.230 ms  81.782 ms
 8  int-0-5-0-1.r1.pao1.isc.org (149.20.65.37)  81.860 ms  83.780 ms 84.160 ms
 9  int-0-0-1-0.r1.sql1.isc.org (149.20.65.10)  81.543 ms  80.193 ms 84.434 ms
10  www.netbsd.org (204.152.190.12)  81.986 ms  81.008 ms  82.604 ms
$
```

# TCP/IP Basics: ICMP

## Internet Control Message Protocol

```
IP (tos 0x0, ttl 1, id 44866, offset 0, flags [none], proto UDP (17), length 40)
    166.84.7.99.44865 > 149.20.53.86.33435: [udp sum ok] UDP, length 12
IP (tos 0xc0, ttl 64, id 48796, offset 0, flags [none], proto ICMP (1), length 68)
    166.84.0.1 > 166.84.7.99: ICMP time exceeded in-transit, length 48
IP (tos 0x0, ttl 2, id 44869, offset 0, flags [none], proto UDP (17), length 40)
    166.84.7.99.44865 > 149.20.53.86.33438: [udp sum ok] UDP, length 12
IP (tos 0x0, ttl 3, id 44872, offset 0, flags [none], proto UDP (17), length 40)
    166.84.7.99.44865 > 149.20.53.86.33441: [udp sum ok] UDP, length 12
IP (tos 0x0, ttl 4, id 44875, offset 0, flags [none], proto UDP (17), length 40)
    166.84.7.99.44865 > 149.20.53.86.33444: [udp sum ok] UDP, length 12
IP (tos 0x0, ttl 252, id 6760, offset 0, flags [none], proto ICMP (1), length 56)
    154.24.25.109 > 166.84.7.99: ICMP time exceeded in-transit, length 36
...
IP (tos 0x0, ttl 248, id 0, offset 0, flags [none], proto ICMP (1), length 56)
    149.20.53.86 > 166.84.7.99: ICMP 149.20.53.86 udp port 33482 unreachable, length
```

# TCP/IP Basics: ICMP

Green Team:

Path MTU Discovery in Practice

`https://blog.cloudflare.com/path-mtu-discovery-in-practice/`

# TCP/IP Basics: ICMP6

## Internet Control Message Protocol for IPv6

```
$ traceroute6 www.netbsd.org
traceroute6 to www.netbsd.org (2001:4f8:3:7:2e0:81ff:fe52:9a6b) from
    2001:470:30:84:204:d7b0:0:1, 64 hops max, 12 byte packets
 1  router.vc.panix.com  0.271 ms  0.282 ms  0.155 ms
 2  2001:470:30::a654:420e  5.459 ms  1.251 ms  1.073 ms
 3  gige-g3-3.core1.nyc4.he.net  1.288 ms  2.001 ms  10.176 ms
 4  10gigabitethernet8-3.core1.chi1.he.net  26.603 ms  20.532 ms  25.029 ms
 5  2001:470:1:34::2  72.033 ms  72.377 ms  72.686 ms
 6  iana.r1.ord1.isc.org  76.288 ms  72.773 ms  71.481 ms
 7  int-0-0-1-8.r1.pao1.isc.org  73.027 ms  76.489 ms  77.507 ms
 8  int-0-0-1-0.r2.sql1.isc.org  73.555 ms  75.367 ms  74.769 ms
 9  www.NetBSD.org  72.036 ms  72.522 ms  71.39 ms
$
```

# TCP/IP Basics: ICMP6

## Internet Control Message Protocol for IPv6

```
IP6 2001:470:30:84:204:d7b0:0:1.51749 >
        2001:4f8:3:7:2e0:81ff:fe52:9a6b.33435: UDP, length 12
IP6 2001:470:30:84::3 > 2001:470:30:84:204:d7b0:0:1:
        ICMP6, time exceeded in-transit [|icmp6]
IP6 2001:470:30:84:204:d7b0:0:1.51749 >
        2001:4f8:3:7:2e0:81ff:fe52:9a6b.33436: UDP, length 12
[...]
IP6 2001:470:30:84:204:d7b0:0:1.51749 >
        2001:4f8:3:7:2e0:81ff:fe52:9a6b.33461: UDP, length 12
IP6 2001:4f8:3:7:2e0:81ff:fe52:9a6b >
        2001:470:30:84:204:d7b0:0:1: ICMP6, destination unreachable[|icmp6]
```

# TCP/IP Basics: TCP

## Transmission Control Protocol

```
$ telnet www.yahoo.com 80
Trying 98.138.219.232...
Connected to atsv2-fp-shed.wg1.b.yahoo.com.
Escape character is '^]'.
HEAD / HTTP/1.0
```

# TCP/IP Basics: TCP

```
IP 10.183.114.37.45403 > 72.30.35.9.80: Flags [S], seq 5028151, win 65535,
options [mss 1460,nop,wscale 6,sackOK,TS val 598758437 ecr 0], length 0
IP 72.30.35.9.80 > 10.183.114.37.45403: Flags [S.], seq 1983855029, ack 5028152, win
options [mss 1460,sackOK,TS val 1888595968 ecr 598758437,nop,wscale 8], length 0
IP 10.183.114.37.45403 > 72.30.35.9.80: Flags [.], ack 1, win 1026,
options [nop,nop,TS val 598758465 ecr 1888595968], length 0
IP 10.183.114.37.45403 > 72.30.35.9.80: Flags [P.], seq 1:18, ack 1, win 1026,
options [nop,nop,TS val 598762755 ecr 1888595968], length 17: HTTP: HEAD / HTTP/1.0
[...]
IP 72.30.35.9.80 > 10.183.114.37.45403: Flags [F.], seq 325, ack 20, win 57,
options [nop,nop,TS val 1888601741 ecr 598764209], length 0
IP 10.183.114.37.45403 > 72.30.35.9.80: Flags [F.], seq 20, ack 326, win 1026,
options [nop,nop,TS val 598764236 ecr 1888601741], length 0
IP 72.30.35.9.80 > 10.183.114.37.45403: Flags [.], ack 21, win 57,
options [nop,nop,TS val 1888601768 ecr 598764236], length 0
```

# TCP/IP Basics: TCP

Blue Team:

Low Rate TCP Denial of Service Attack Detection at Edge Routers
See also:

`http://oriolrius.cat/article_fitxers/326/pdf/p75-kuzmanovic.pdf`

# TCP/IP Basics: TCP

Transmission Control Protocol over IPv6

```
$ telnet www.netbsd.org 80
Trying 2001:4f8:3:7:2e0:81ff:fe52:9a6b...
Connected to www.netbsd.org.
Escape character is '^]'.
GET / HTTP/1.0
```

# TCP/IP Basics: TCP

## Transmission Control Protocol IPv6

```
IP6 2001:470:30:84:204:d7b0:0:1.58334 >
      2001:4f8:3:7:2e0:81ff:fe52:9a6b.80: S 3232473102:3232473102(0)
      win 32768 <mss 1440,nop,wscale3,sackOK,nop,nop,nop,nop,timestamp 1[|tcp]>
IP6 2001:4f8:3:7:2e0:81ff:fe52:9a6b.80 >
      2001:470:30:84:204:d7b0:0:1.58334: S 4139493123:4139493123(0)
      ack 3232473103 win 32768
IP6 2001:470:30:84:204:d7b0:0:1.58334 >
      2001:4f8:3:7:2e0:81ff:fe52:9a6b.80: . ack 1 win 4140
IP6 2001:470:30:84:204:d7b0:0:1.58334 >
      2001:4f8:3:7:2e0:81ff:fe52:9a6b.80: P 1:17(16) ack 1 win 4140
IP6 2001:4f8:3:7:2e0:81ff:fe52:9a6b.80 >
      2001:470:30:84:204:d7b0:0:1.58334: . ack 17 win 33120
```

# TCP/IP Basics: TCP

Red Team:

[CVE-2019-14899] Inferring and hijacking VPN-tunneled TCP connections.
`https://seclists.org/oss-sec/2019/q4/122`

# TCP/IP Basics: UDP

## User Datagram Protocol

```
$ nslookup www.yahoo.com
Server: 155.246.1.20
Address: 155.246.1.20#53

Non-authoritative answer:
www.yahoo.com        canonical name = fp3.wg1.b.yahoo.com.
fp3.wg1.b.yahoo.com       canonical name = any-fp3-lfb.wa1.b.yahoo.com.
any-fp3-lfb.wa1.b.yahoo.com        canonical name = any-fp3-real.wa1.b.yahoo.com.
Name: any-fp3-real.wa1.b.yahoo.com
Address: 98.139.183.24

$
```

# TCP/IP Basics: UDP

## User Datagram Protocol

```
IP (tos 0x0, ttl 64, id 0, offset 0, flags [none],
    proto UDP (17), length 59) 166.84.7.99.61507 > 166.84.67.2.53:
        [udp sum ok] 25447+ A? www.yahoo.com. (31)

IP (tos 0x0, ttl 63, id 58853, offset 0, flags [none],
    proto UDP (17), length 275) 166.84.67.2.53 > 166.84.7.99.61507:
        [udp sum ok] 25447 q: A? www.yahoo.com. 3/4/4 www.yahoo.com.
            CNAME atsv2-fp-shed.wg1.b.yahoo.com., [...]
```

# TCP/IP Basics: UDP

## User Datagram Protocol over IPv6

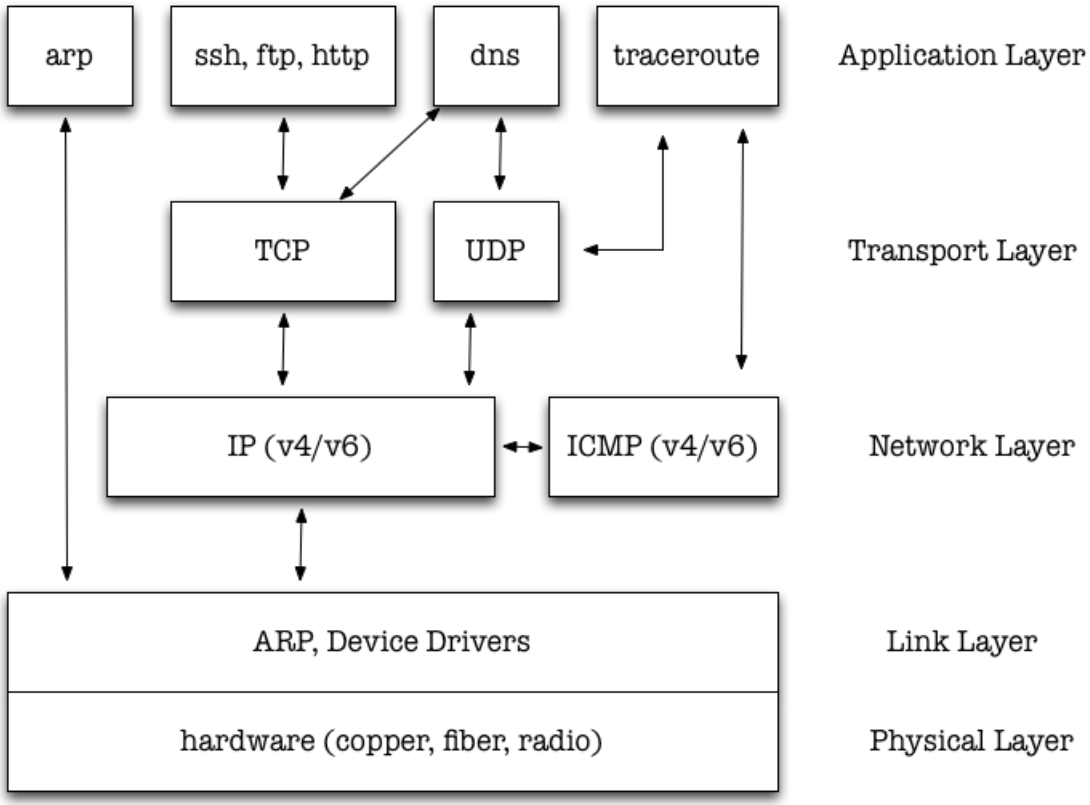
```
$ dig -6 @2001:470:20::2 www.yahoo.com

;; ANSWER SECTION:
www.yahoo.com.          300     IN      CNAME   fp3.wg1.b.yahoo.com.
fp3.wg1.b.yahoo.com.    60      IN      CNAME   any-fp3-lfb.wa1.b.yahoo.com.
any-fp3-lfb.wa1.b.yahoo.com. 300 IN     CNAME   any-fp3-real.wa1.b.yahoo.com.
any-fp3-real.wa1.b.yahoo.com. 60 IN     A       98.139.183.24

;; Query time: 51 msec
;; SERVER: 2001:470:20::2#53(2001:470:20::2)
;; WHEN: Sat Mar  3 22:49:44 2012
;; MSG SIZE  rcvd: 128
```

# TCP/IP Basics: UDP

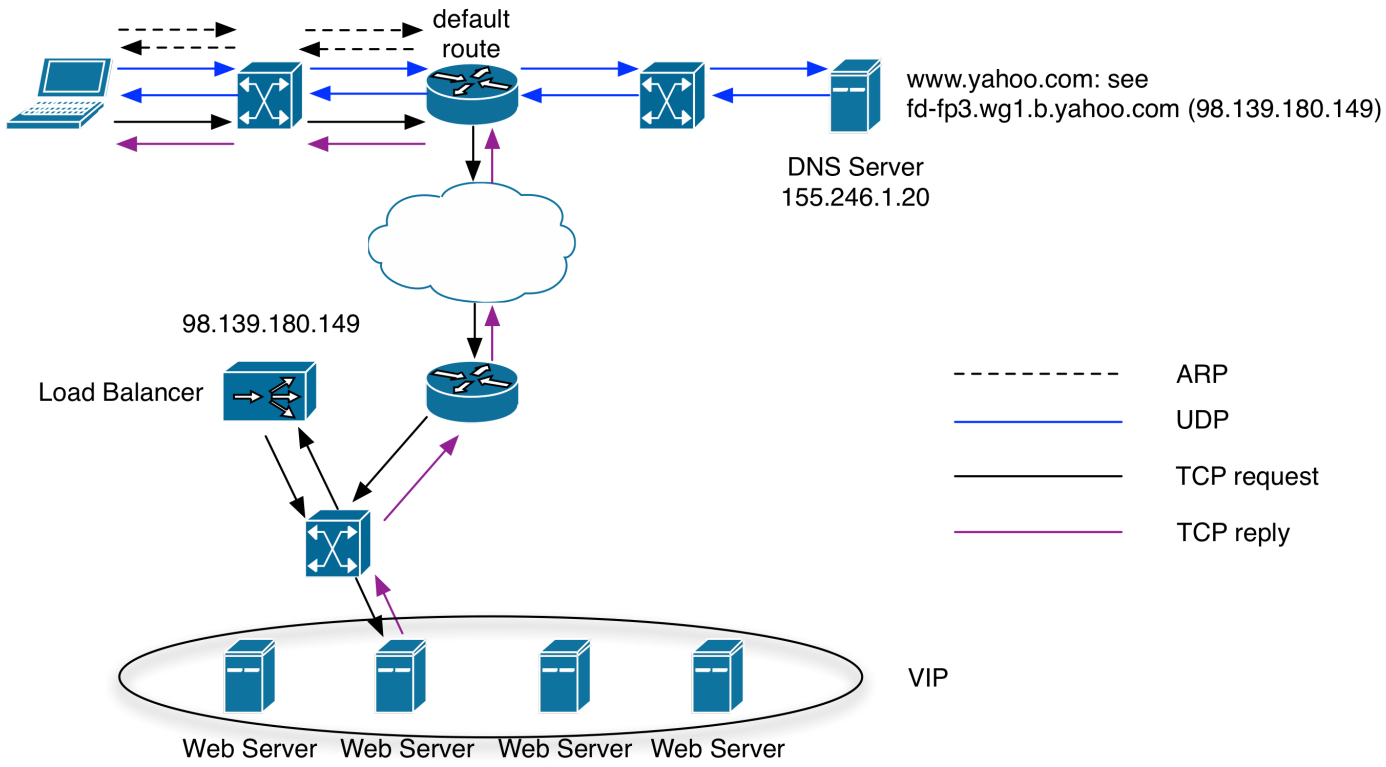
## User Datagram Protocol over IPv6

```
IP6 (hlim 64, next-header: UDP (17), length: 39)
       2001:470:30:84:204:d7b0:0:1.65037 > 2001:470:20::2.53:
       [udp sum ok] 18545+ A? www.yahoo.com. (31)

IP6 (hlim 61, next-header: UDP (17), length: 119)
       2001:470:20::2.53 > 2001:470:30:84:204:d7b0:0:1.65037:
       18545 4/0/0 www.yahoo.com.[|domain]
```

# TCP/IP Basics: Putting it all together



# Networking


default
route
www.yahoo.com: see
fd-fp3.wg1.b.yahoo.com (98.139.180.149)
DNS Server
155.246.1.20
98.139.180.149
Load Balancer
ARP
UDP
TCP request
TCP reply
VIP
Web Server
Web Server
Web Server
Web Server

# Homework

`https://www.cs.stevens.edu/~jschauma/615/s20-hw3.html`

# Reading

- `tcpdump(8)`
- `ktrace(1)` / `strace(1)`
- `tcp(4)`/`ip(4)`
- `netstat(1)`
- `nslookup(1)`

- `https://is.gd/mrrdYc`