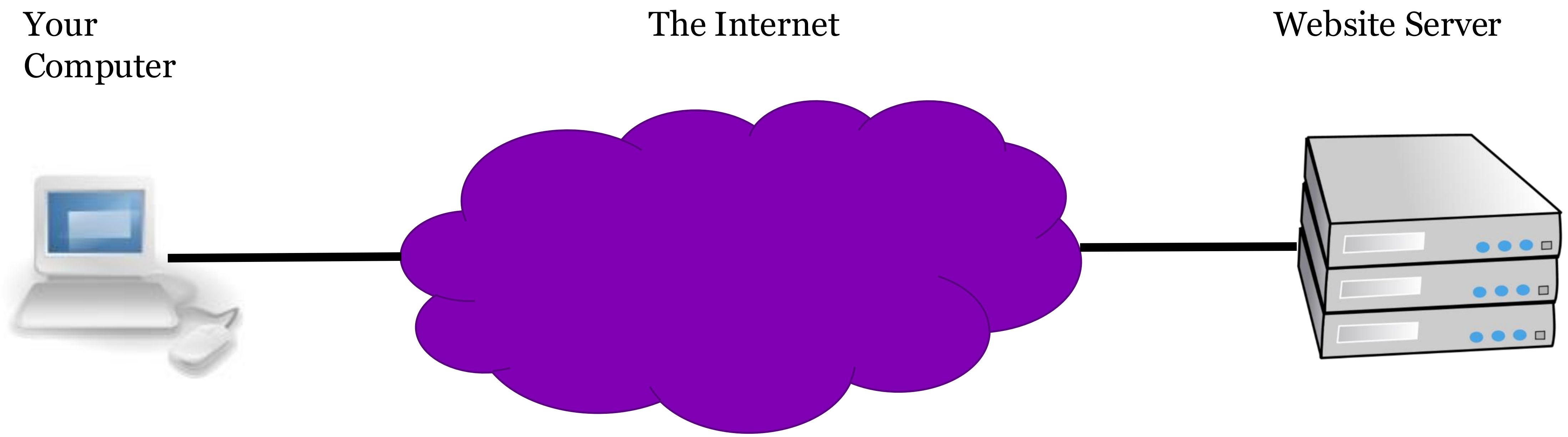
ECE458/ECE750T27: Computer Security
Networking

Dr. Kami Vaniea
Electrical and Computer Engineering
kami.vaniea@uwaterloo.ca

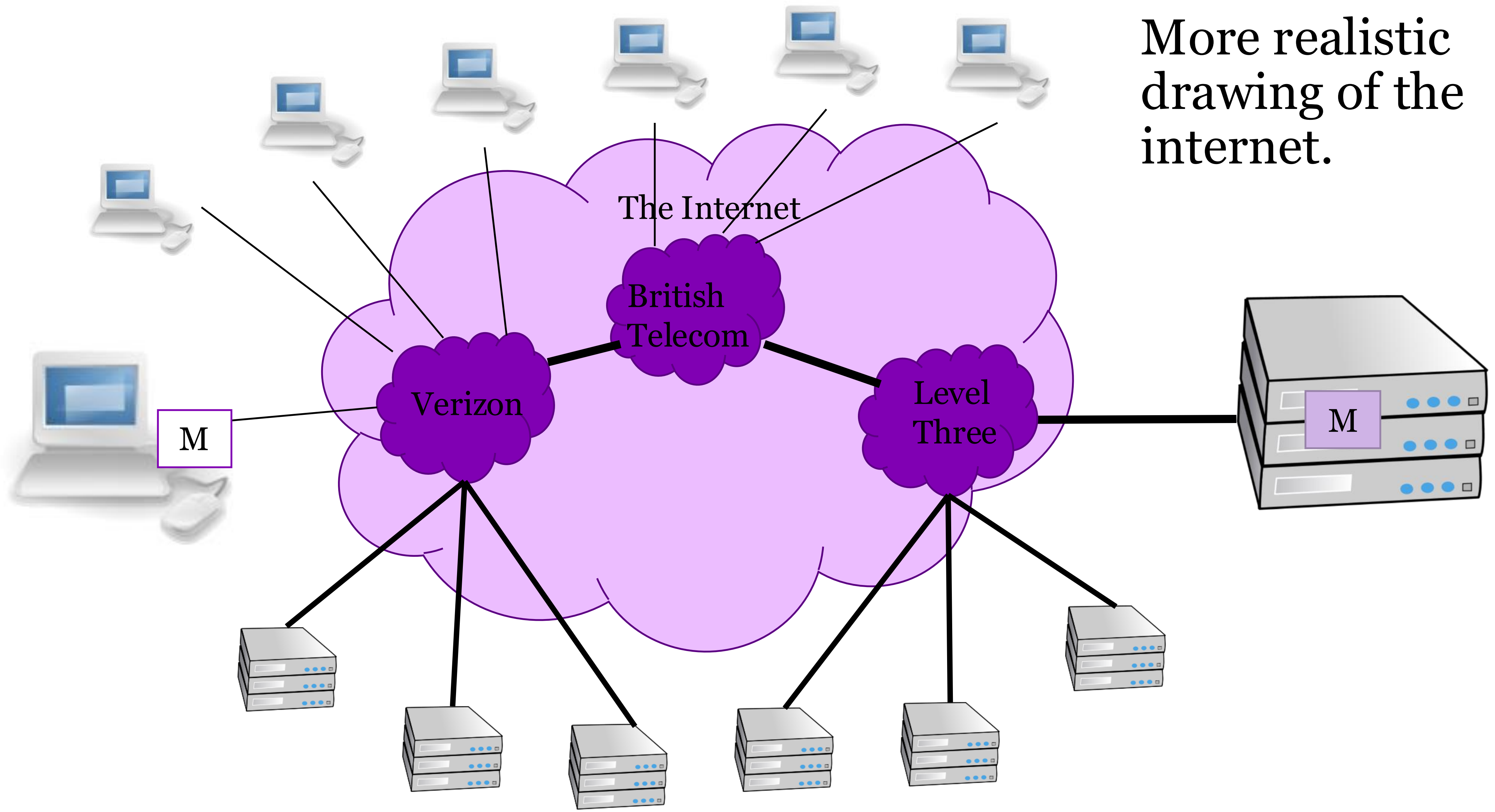
UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

TULiPS
Technology Usability Lab in Privacy and Security

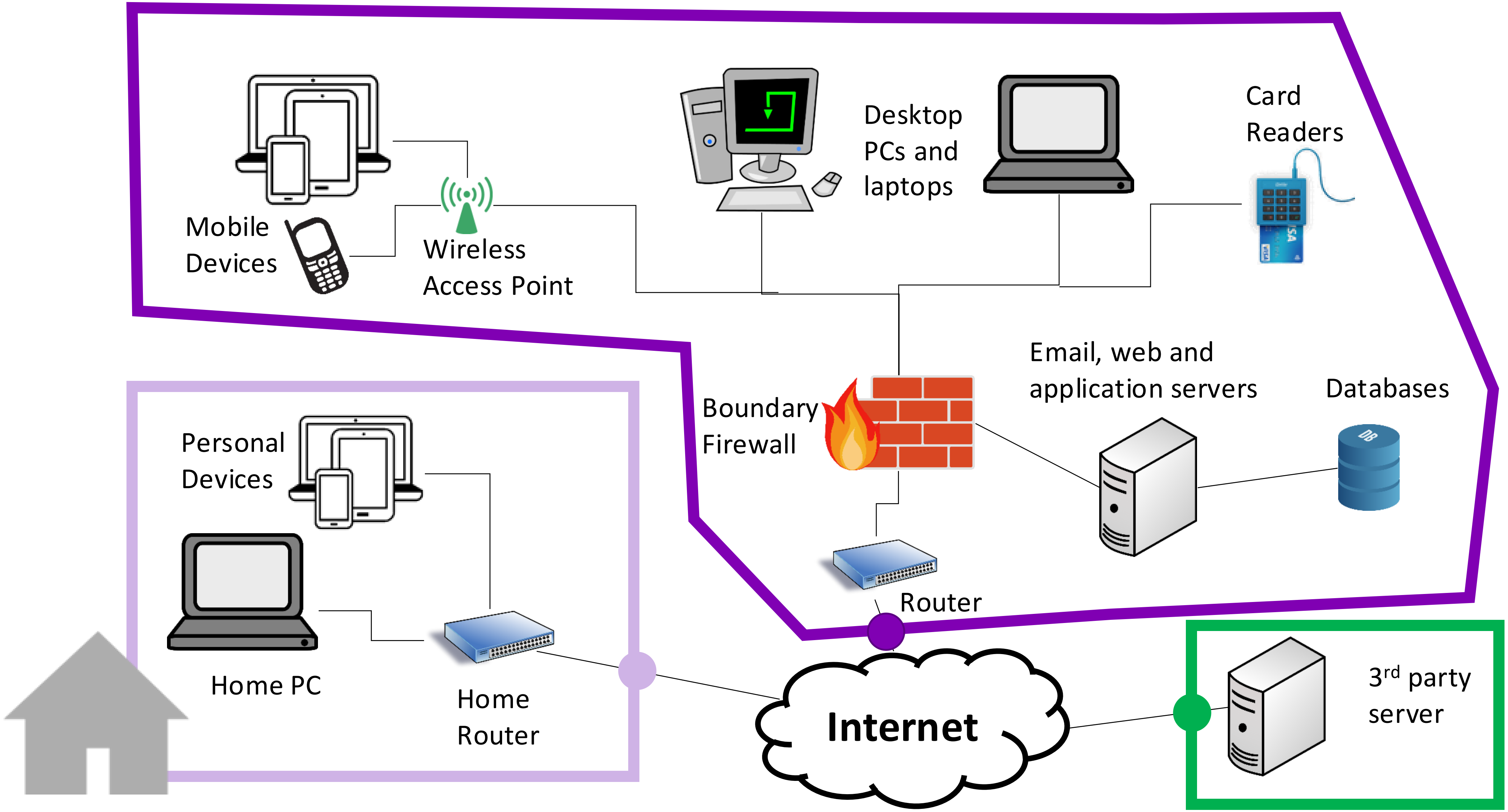


Basic standard drawing of the Internet.

Your computer (left) connects to “the cloud” (middle) which connects you to the webserver you want to talk with (right).



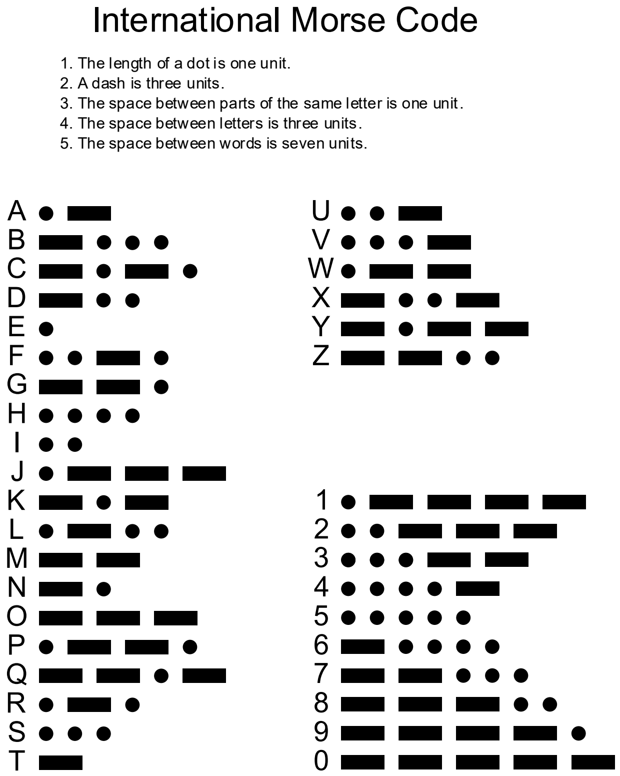
More realistic drawing of the internet.


Mobile
Devices
Wireless
Access Point
Desktop
PCs and
laptops
Card
Readers
Email, web and
application servers
Databases
Boundary
Firewall
Router
Personal
Devices
Home PC
Home
Router
Internet
3rd party
server

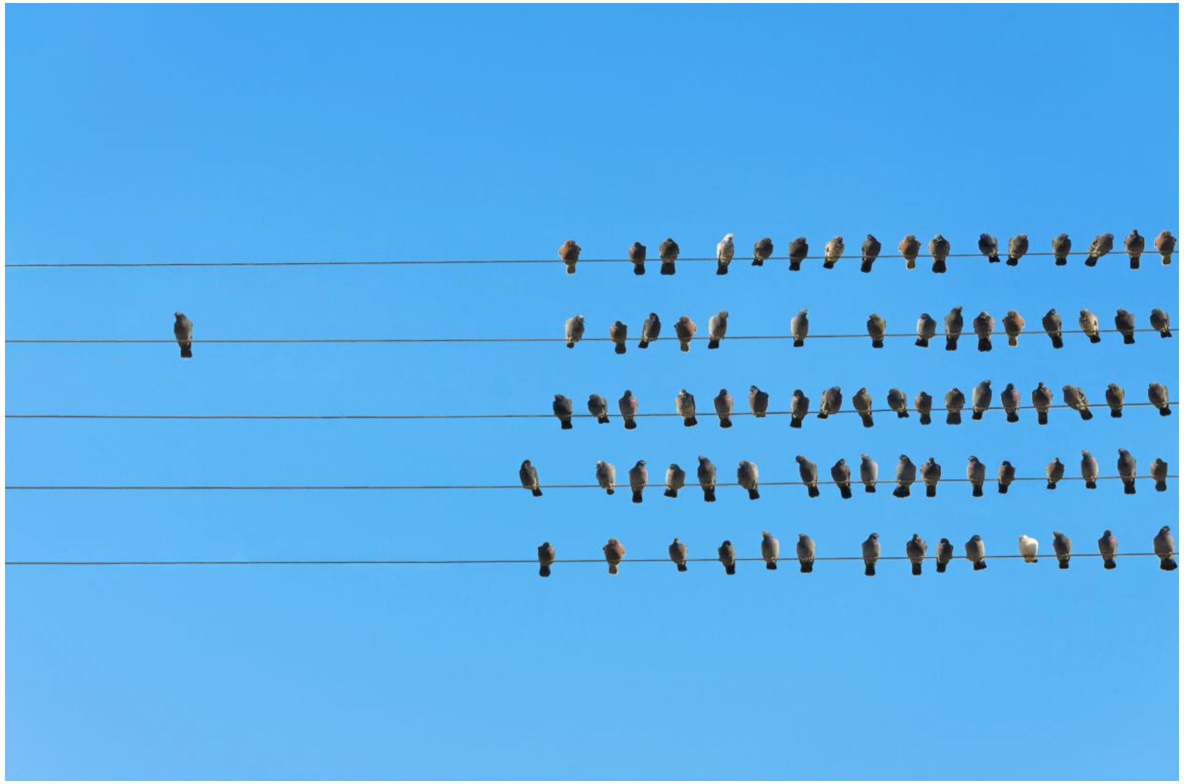
NETWORKING – A BIT OF HISTORY

Telegraphs

- Messages used to be sent via telegraph
  - 1844: First telegraph message sent
  - 1866: Telegraph wires between US and Europe
- The operator pushes down on the button creating a "beep" as long as they press



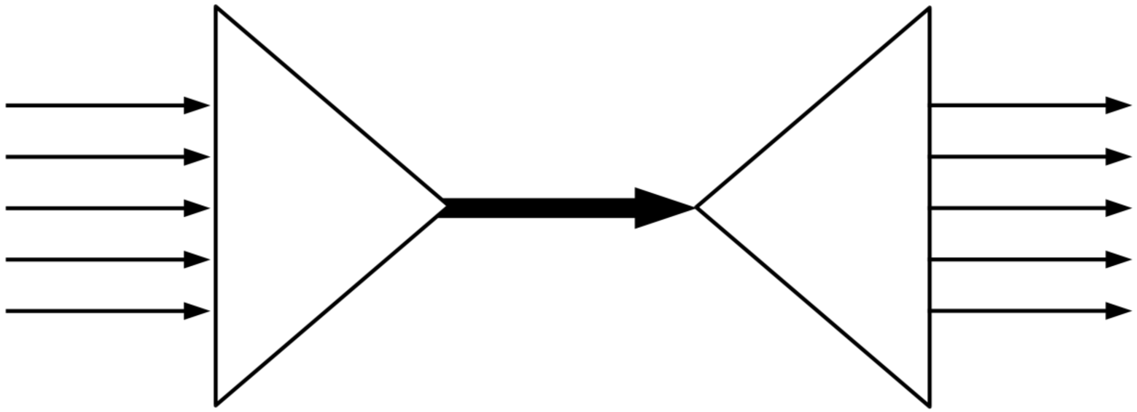
Problem: How do N things all use one wire without problems?

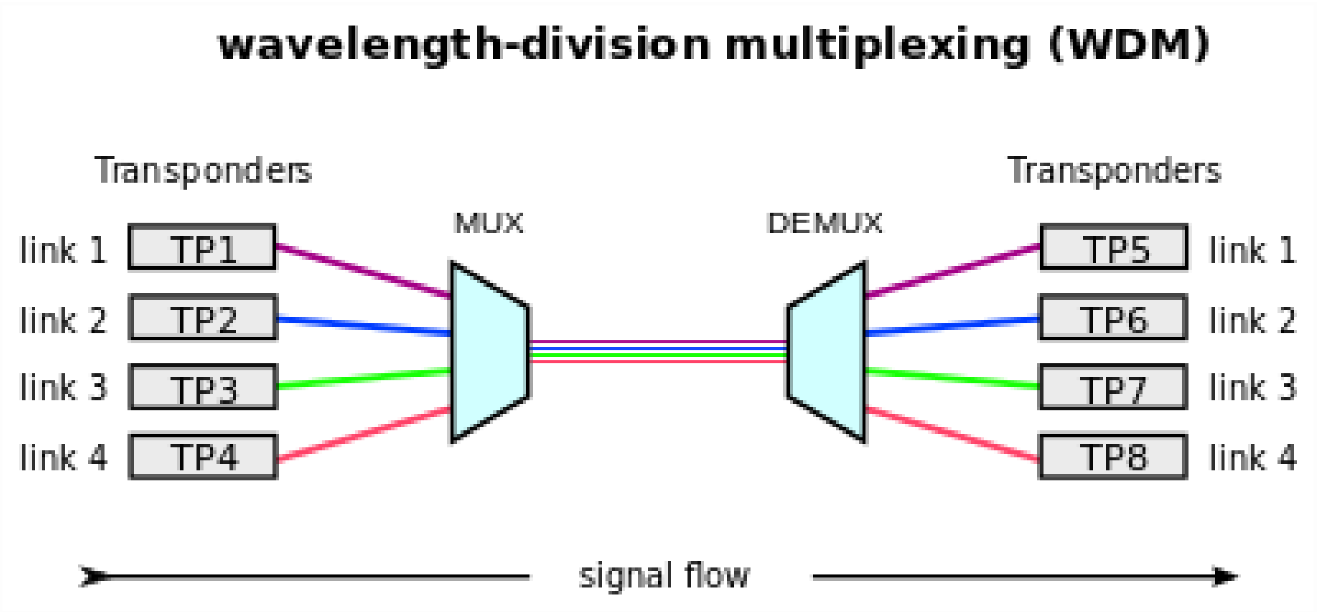


Science Photo Library

Partial Solution: Multiplexing

- Send multiple signals at once.
- Signals are just wave forms, so we could send at different frequencies.
- Good: we can now send more than one signal, but still quite expensive to direct connect two computers this way.



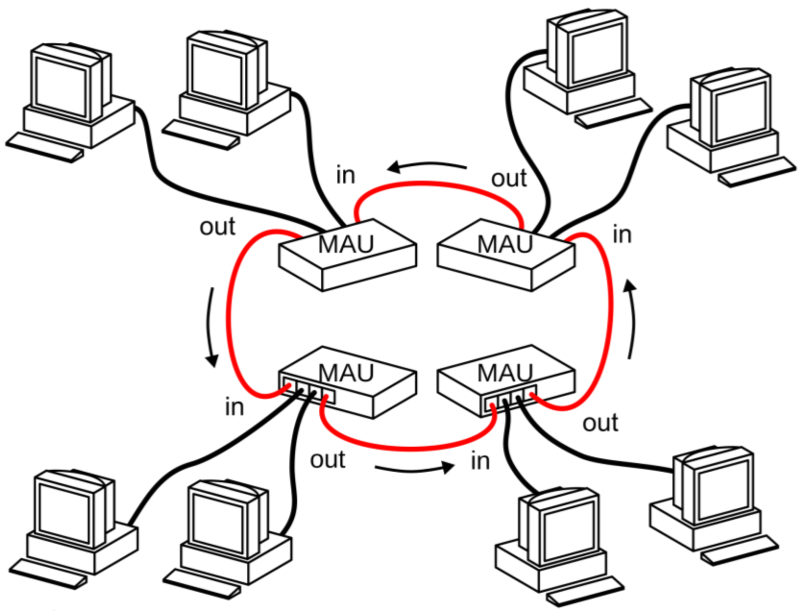


Token Ring (1984)

- Ring of computers
- A token passes around the ring
- A computer who wants to send, takes the token off the ring, transmits, then puts it back on when done

Starting Delimitator, Destination Address, Source Address, Ending Delimitator, Frame Status

SD	AC	FC	DA	SA	PDU from LLC (IEEE 802.2)	CRC	ED	FS
8 bits	8 bits	8 bits	48 bits	48 bits	Up to 4500 × 8 bits	32 bits	8 bits	8 bits



By Andrew28913 on Wikipedia

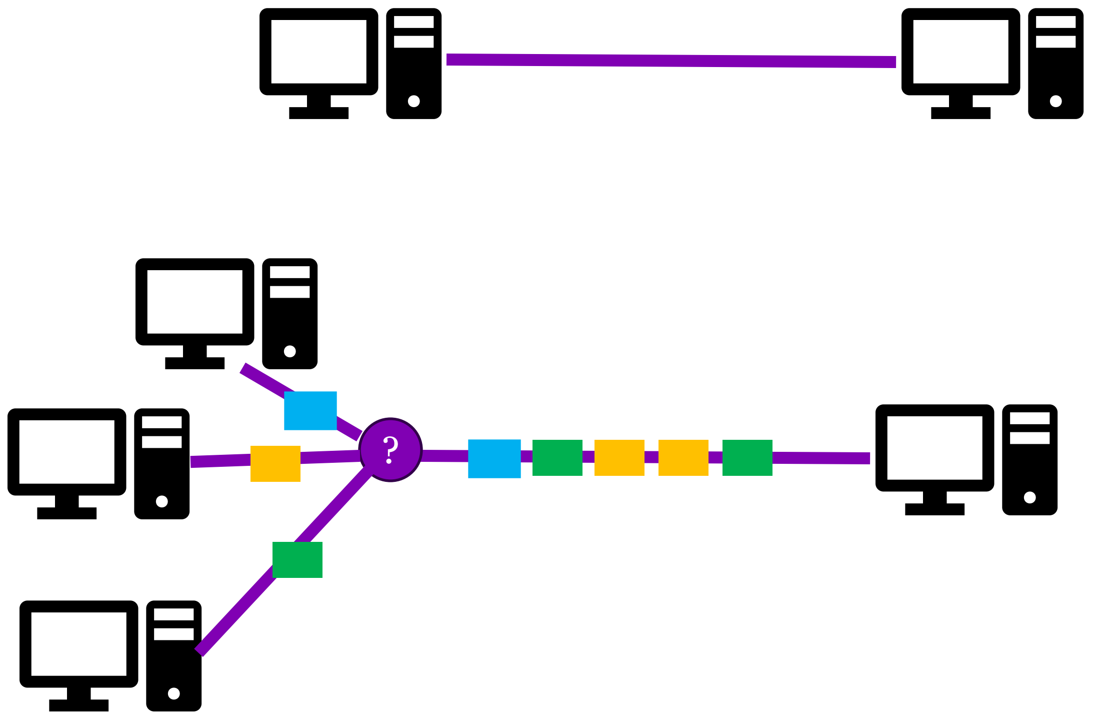
Datagram:

"A self-contained, independent entity of data carrying sufficient information to be routed from the source to the destination computer without reliance on earlier exchanges between this source and destination computer and the transporting network."

—RFC 1594

Partial solution 2: share the wire using datagrams

- Each computer breaks their messages into datagrams of a max size
- Then those datagrams are put across the wire.
- Now we can share the transmission line
- How to decide who sends when
  - Collisions
  - Congestion



Protocol

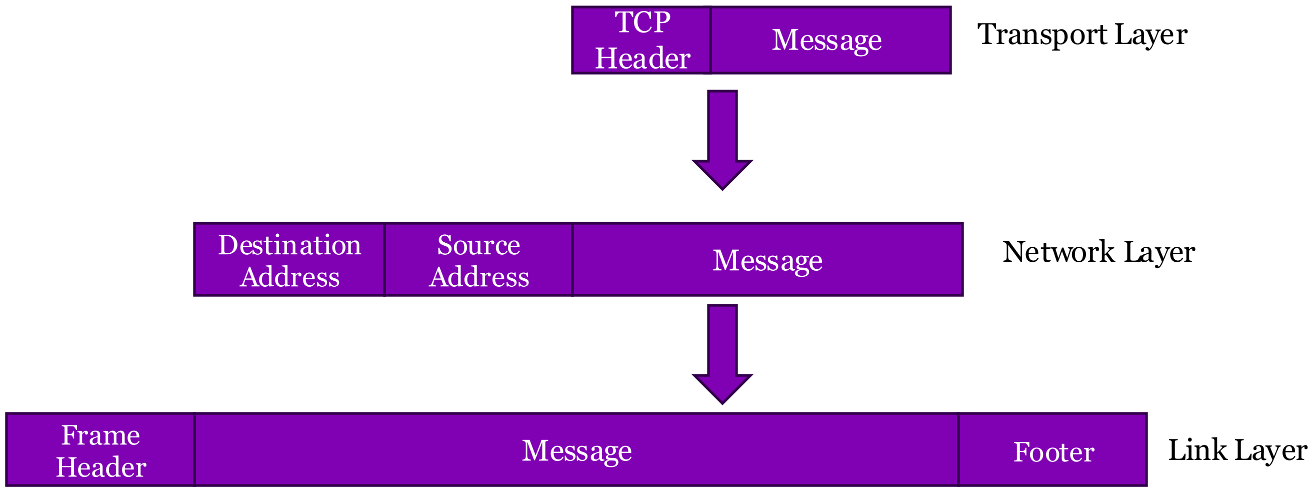
- Language or set of conventions for how two computers will interact.
- Issues like:
  - When to send
  - What the end of a transmission looks like
  - Structure of the packet sent

Starting Delimitator | Destination Address | Source Address | Ending Delimitator | Frame Status

SD	AC	FC	DA	SA	PDU from LLC (IEEE 802.2)	CRC	ED	FS
8 bits	8 bits	8 bits	48 bits	48 bits	Up to 4500 × 8 bits	32 bits	8 bits	8 bits

Packet

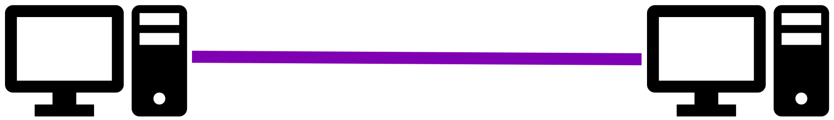
- Smallest individually addressable data unit transmitted.
- A packet is a simple concept: it has a destination address, source address and message.
- Usually created in layers by different parts of the software stack.

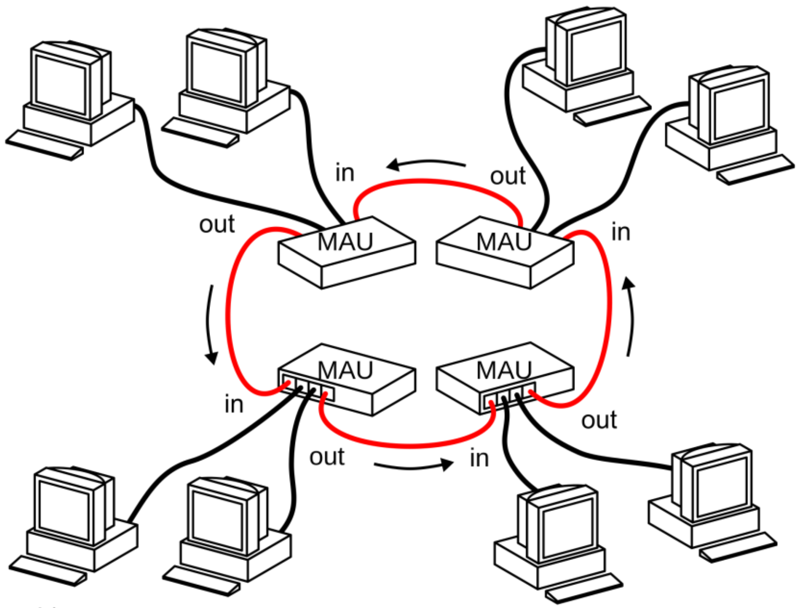


Packet addressing

Frame Header	Destination Address	Source Address	TCP Header	Message	Frame Footer

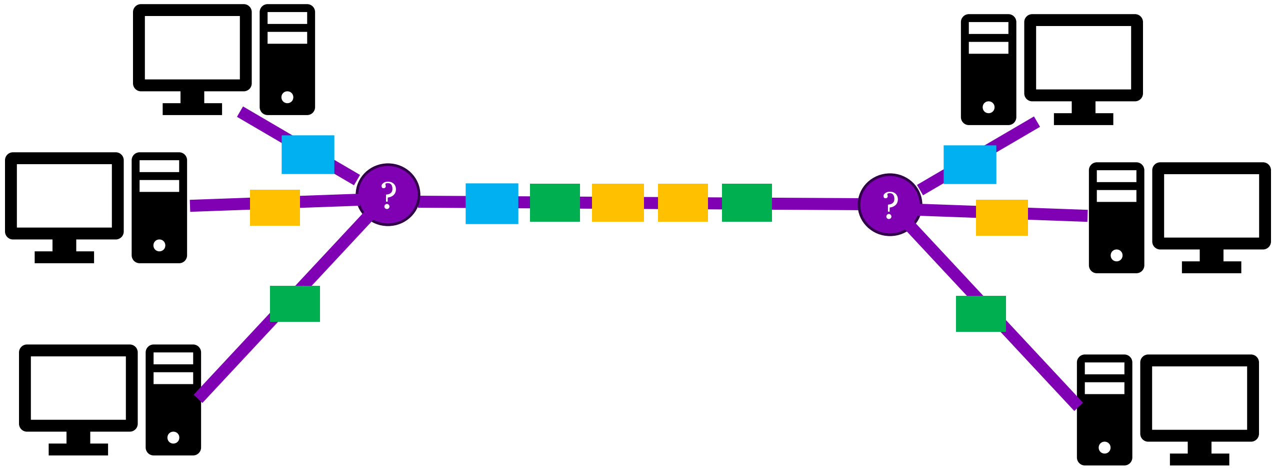
- How do computers/routers know where to send datagrams?
- How does a computer know when to pickup a packet/datagram from the network?





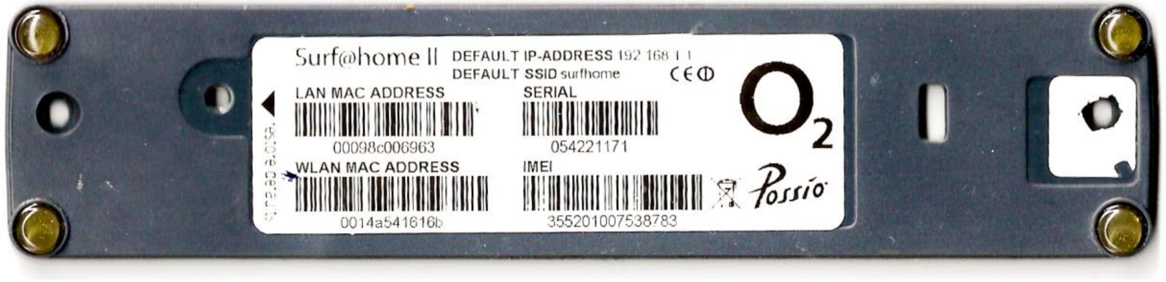
b)

By Andrew28913 on Wikipedia

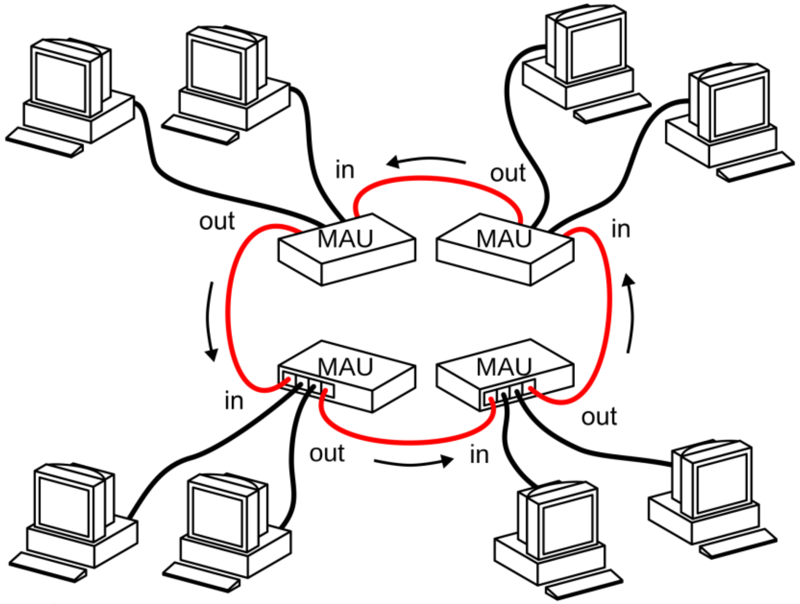


MAC address

- Medium Access Control (MAC) address
- Assigned at time of manufacture (mostly)
- Six groups of 2 hexadecimal digits
- Used to identify unique devices on a local network

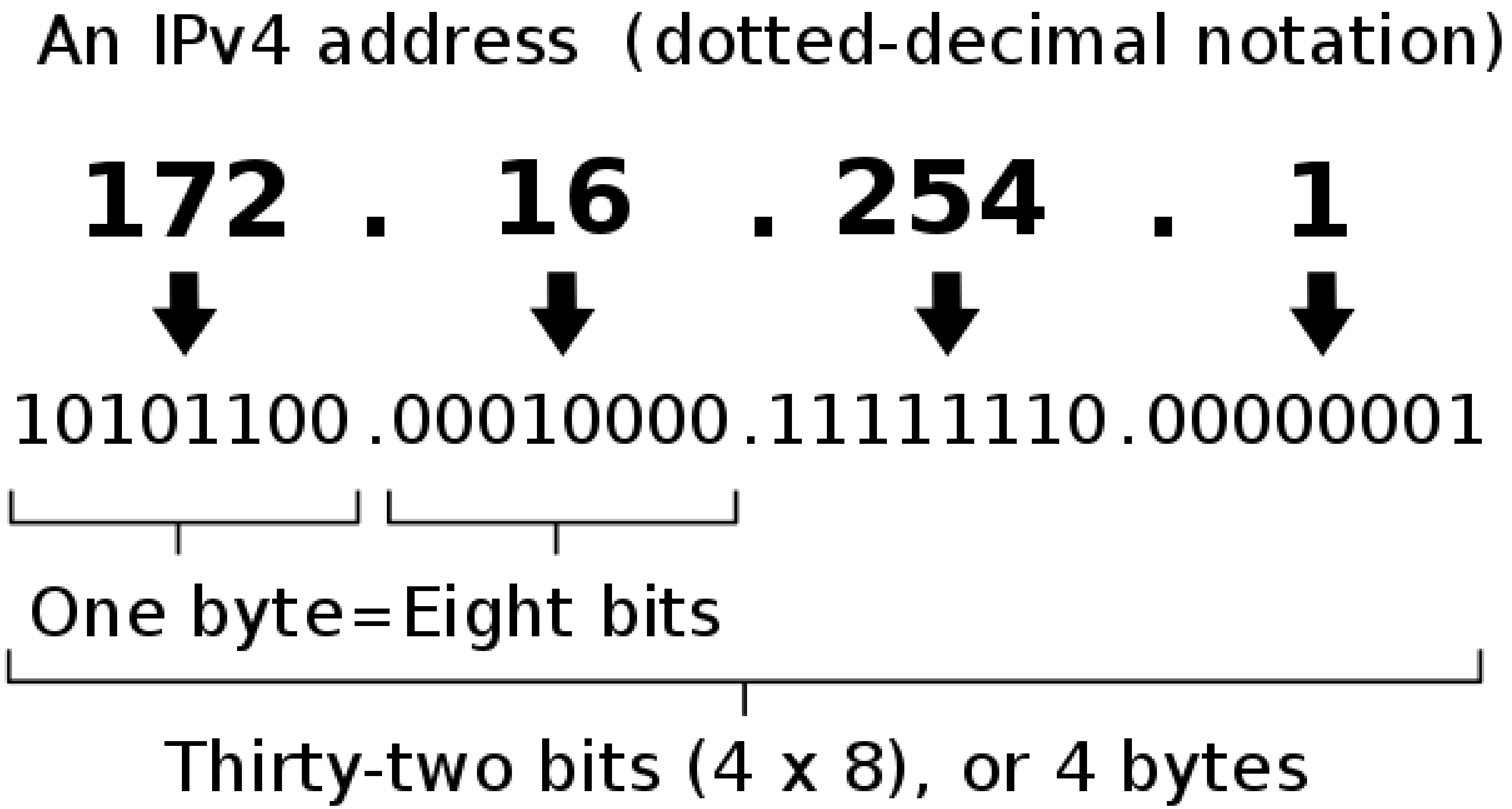


By © Raimond Spekking / CC BY-SA 4.0 (via Wikimedia Commons), CC BY-SA 4.0



By Andrew28913 on Wikipedia

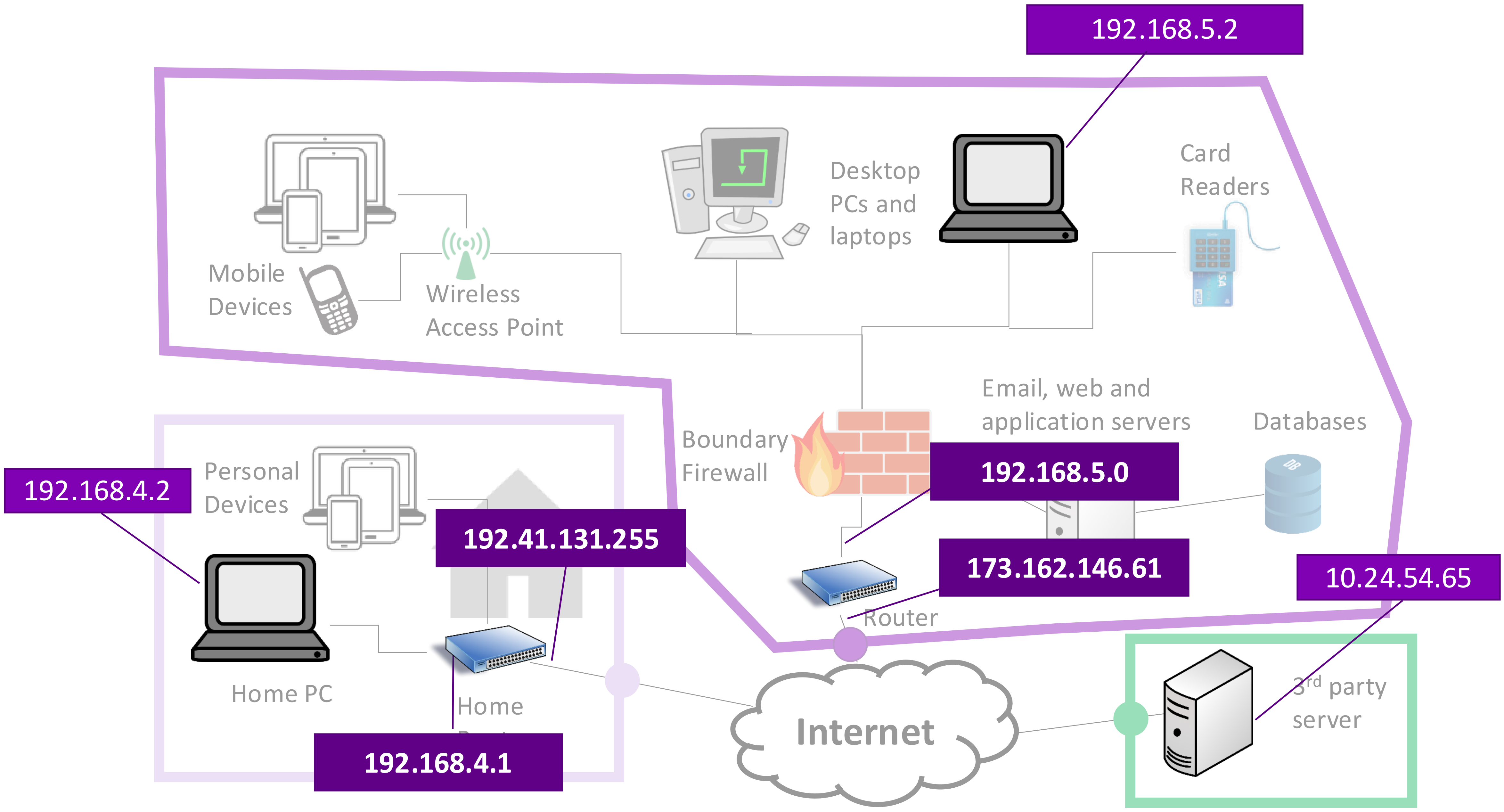
INTERNET PROTOCOL (IP) ADDRESSES


An IPv4 address (dotted-decimal notation)
172 . 16 . 254 . 1
10101100.00010000.11111110.00000001
One byte=Eight bits
Thirty-two bits (4 x 8), or 4 bytes

Every computer on a network has an IP address which is unique from the other computers.

Each interface on a computer gets one IP address.

So your WIFI would get one IP address and your wired network connection would get a different IP address if both were connected.


192.168.5.2
Mobile Devices
Wireless Access Point
Desktop PCs and laptops
Card Readers
Email, web and application servers
Databases
Boundary Firewall
192.168.5.0
173.162.146.61
Router
10.24.54.65
192.168.4.2
Personal Devices
192.41.131.255
Home PC
Home
192.168.4.1
Internet
3rd party server

```
kvaniea@brendel:~$
1240 > ifconfig
bond0: flags=5187<UP,BROADCAST,RUNNING,MASTER,MULTICAST>  mtu 1500
        inet 129.215.33.112  netmask 255.255.255.0  broadcast 129.215.33.255
        inet6 2001:630:3c1:33:222:19ff:fed5:cb52  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::222:19ff:fed5:cb52  prefixlen 64  scopeid 0x20<link>
        ether 00:22:19:d5:cb:52  txqueuelen 1000  (Ethernet)
        RX packets 367631439  bytes 246252199152 (229.3 GiB)
        RX errors 0  dropped 14546  overruns 0  frame 0
        TX packets 317902874  bytes 189161541398 (176.1 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ifconfig on Linux

ipconfig on Windows

```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : lan
   IPv6 Address. . . . . . . . . . . : fd02:9d33:f1fa::446
   IPv6 Address. . . . . . . . . . . : fd02:9d33:f1fa:0:483:b9e3:91bd:d0d1
   Temporary IPv6 Address. . . . . . : fd02:9d33:f1fa:0:80d:954d:fb96:88c5
   Link-local IPv6 Address . . . . . : fe80::483:b9e3:91bd:d0d1%3
   IPv4 Address. . . . . . . . . . . : 192.168.2.103
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.2.1
```

If you check your own IP address regularly, you will notice that it changes every time your computer changes networks.

Inside the University only the last few bits will normally change, but if you go home the whole address will most likely change.

Terminal — bash

```
guest@dnstools.ch:~> traceroute facebook.com
1 static.1.241.243.136.clients.your-server.de (136.243.241.1) 0.228 ms
2 core24.fsn1.hetzner.com (213.239.229.53) 0.230 ms
3 core1.fra.hetzner.com (213.239.229.77) 4.921 ms
4 core2.ams.hetzner.com (213.239.203.158) 10.602 ms
5 br02.ams1.tfbnw.net (80.249.209.164) 11.665 ms
6 po131.asw02.ams2.tfbnw.net (204.15.21.94) 11.682 ms
7 po231.psw01.ams2.tfbnw.net (157.240.35.163) 12.001 ms
8 173.252.67.187 (173.252.67.187) 11.678 ms
9 edge-star-mini-shv-01-amt2.facebook.com (31.13.64.35) 11.870 ms
```

http://en.dnstools.ch/visual-traceroute.html

IP addresses are organized into ranges

129.215.31.248 – 129.215.31.255	EUCS	SERV	Network Infrastructure
129.215.32.0 – 129.215.32.255	ECSC	ECSC	Informatics wire C, Kings Buildings
129.215.33.0 – 129.215.33.255	ECSC	ECSC	Informatics Forum, Potterrow
129.[illegible]34.0 – [illegible].34.255	ELEE	ELEE	EE-net5 (STRG -> new bldg)
129.[illegible]35.0 – 12[illegible]5.35.255	EXEB	EXEB	IS SG Chancellor's Building NRIE

```
kvaniea@brendel:~$
1240 > ifconfig
bond0: flags=5187<UP,BROADCAST,RUNNING,MASTER,MULTICAST>  mtu 1500
        inet 129.215.33.112  netmask 255.255.255.0  broadcast 129.215.33.255
        inet6 2001:630:3c1:33:222:19ff:fed5:cb52  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::222:19ff:fed5:cb52  prefixlen 64  scopeid 0x20<link>
        ether 00:22:19:d5:cb:52  txqueuelen 1000  (Ethernet)
        RX packets 367631439  bytes 246252199152 (229.3 GiB)
        RX errors 0  dropped 14546  overruns 0  frame 0
        TX packets 317902874  bytes 189161541398 (176.1 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

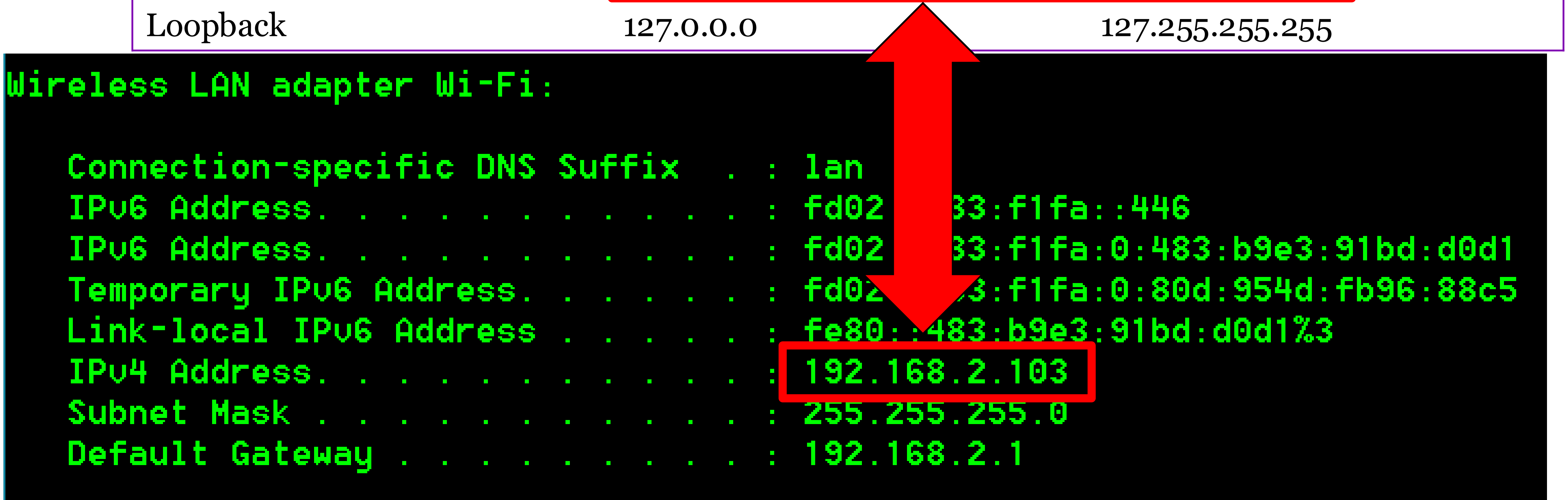
[illegible] – 129.215.36.159	EXMA	EXMA	MALTS/AVTS Appleton Tower
129.215.36.140 – 129.215.36.143	EUCS	EUCS	EUCS
129.215.36.168			

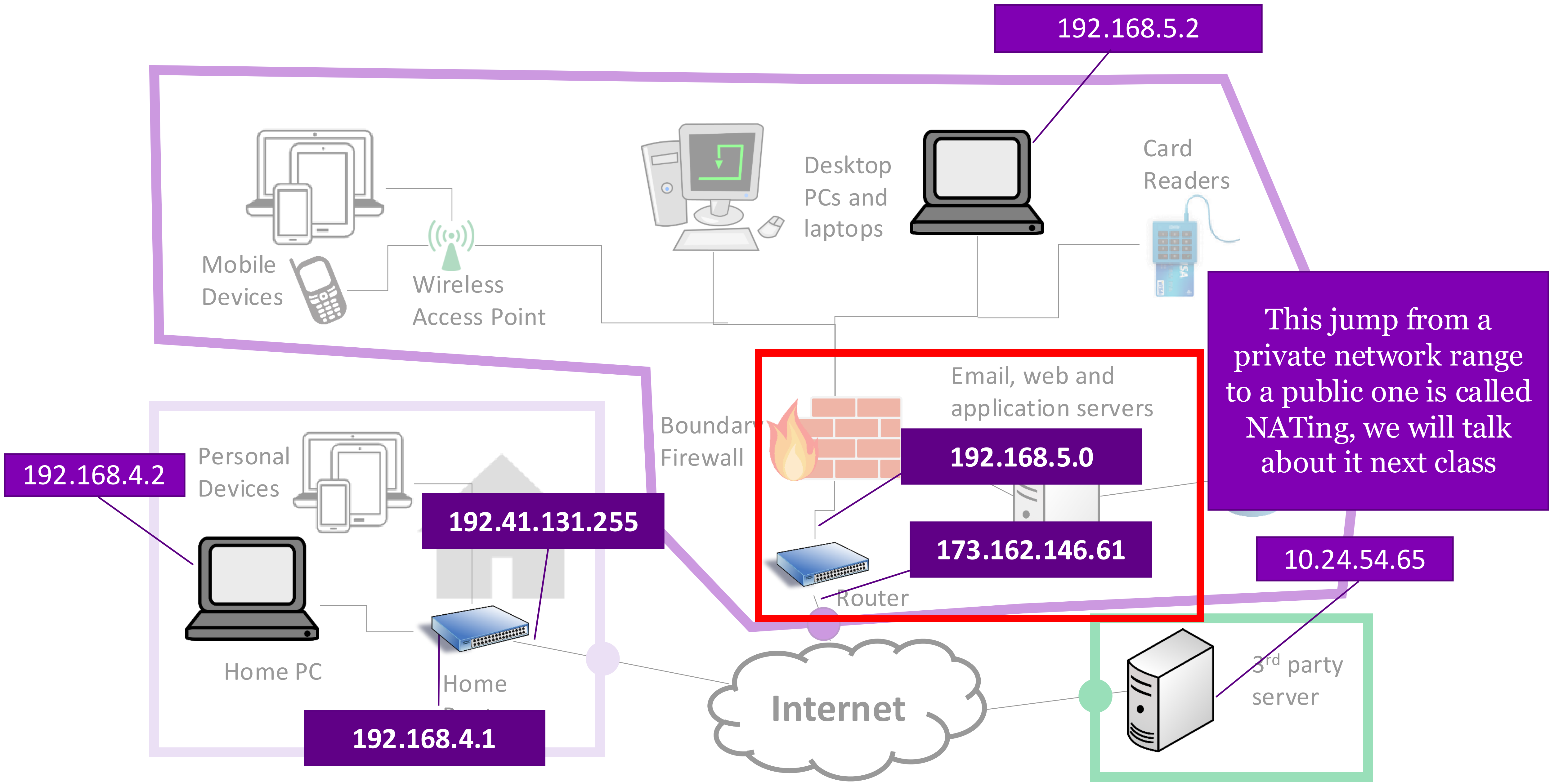
IP Reserved Ranges

Class	Start Address	End Address
A – Private	10.0.0.0	10.255.255.255
B – Private	172.16.0.0	172.31.255.255
C – Private	192.168.0.0	192.168.255.255
Loopback	127.0.0.0	127.255.255.255

IP Reserved Ranges

Class	Start Address	End Address
A – Private	10.0.0.0	10.255.255.255
B – Private	172.16.0.0	172.31.255.255
C – Private	192.168.0.0	192.168.255.255
Loopback	127.0.0.0	127.255.255.255

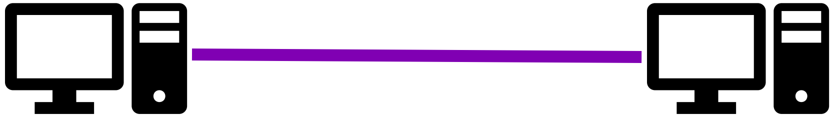


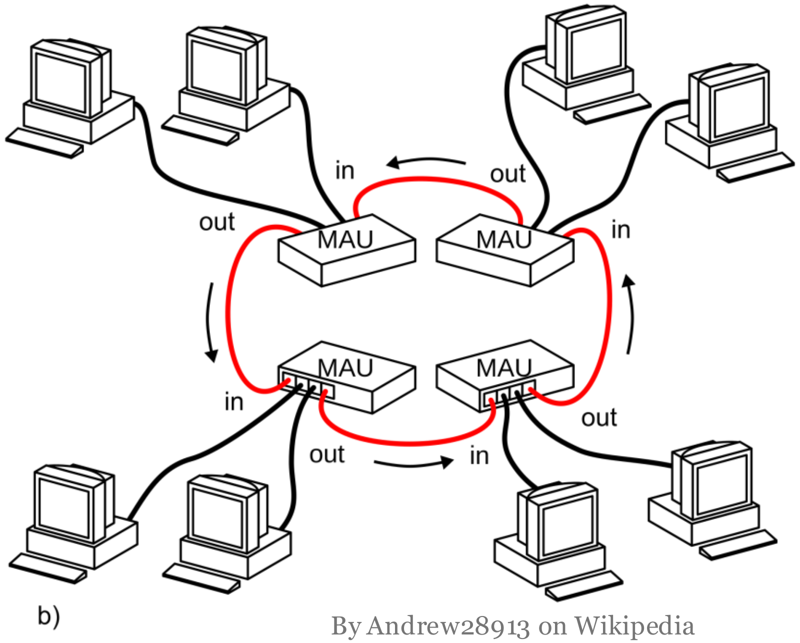


192.168.5.2
Card Readers
Desktop PCs and laptops
Mobile Devices
Wireless Access Point
Email, web and application servers
Boundary Firewall
192.168.5.0
173.162.146.61
Router
This jump from a private network range to a public one is called NATing, we will talk about it next class
10.24.54.65
192.168.4.2
Personal Devices
192.41.131.255
Home PC
Home
192.168.4.1
Internet
3rd party server

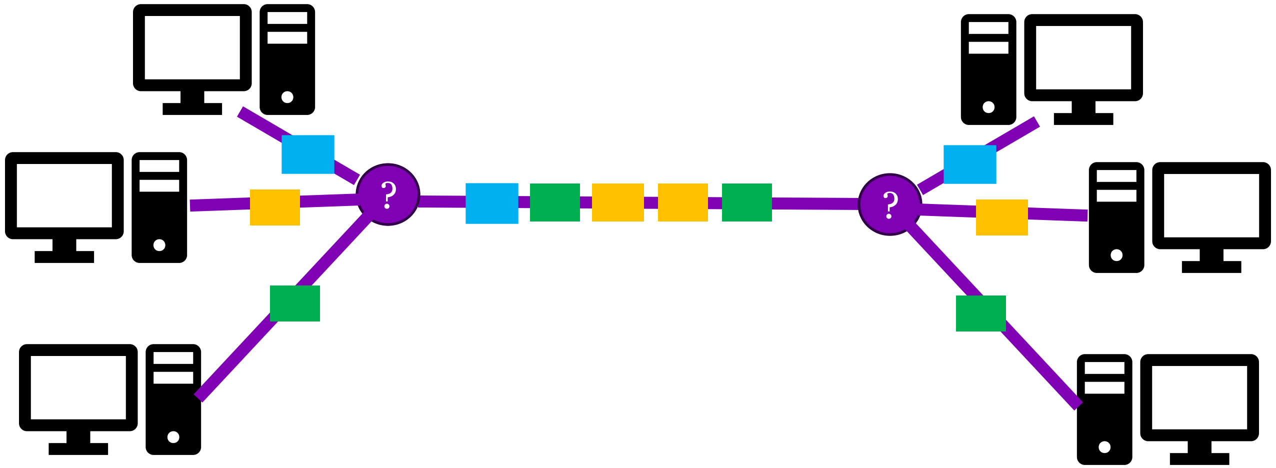
Think-pair-share

- Where is the reference monitor?
- Each packet is addressed, what ensures that only the intended recipient gets the message?





By Andrew28913 on Wikipedia



OSI NETWORK MODEL

Open Systems Interconnect model

- A good way to think about networking steps logically
- Not how software is actually built

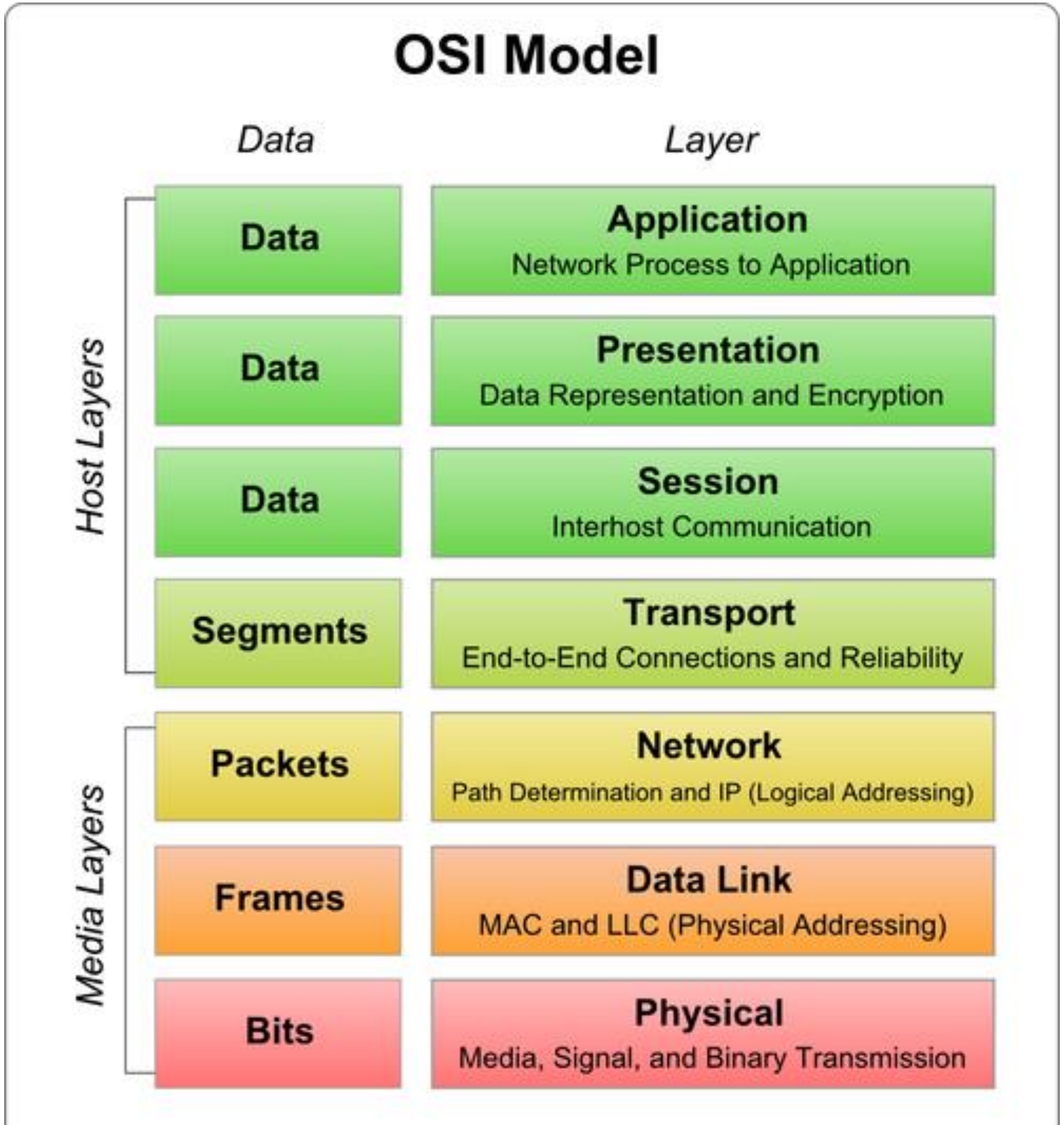
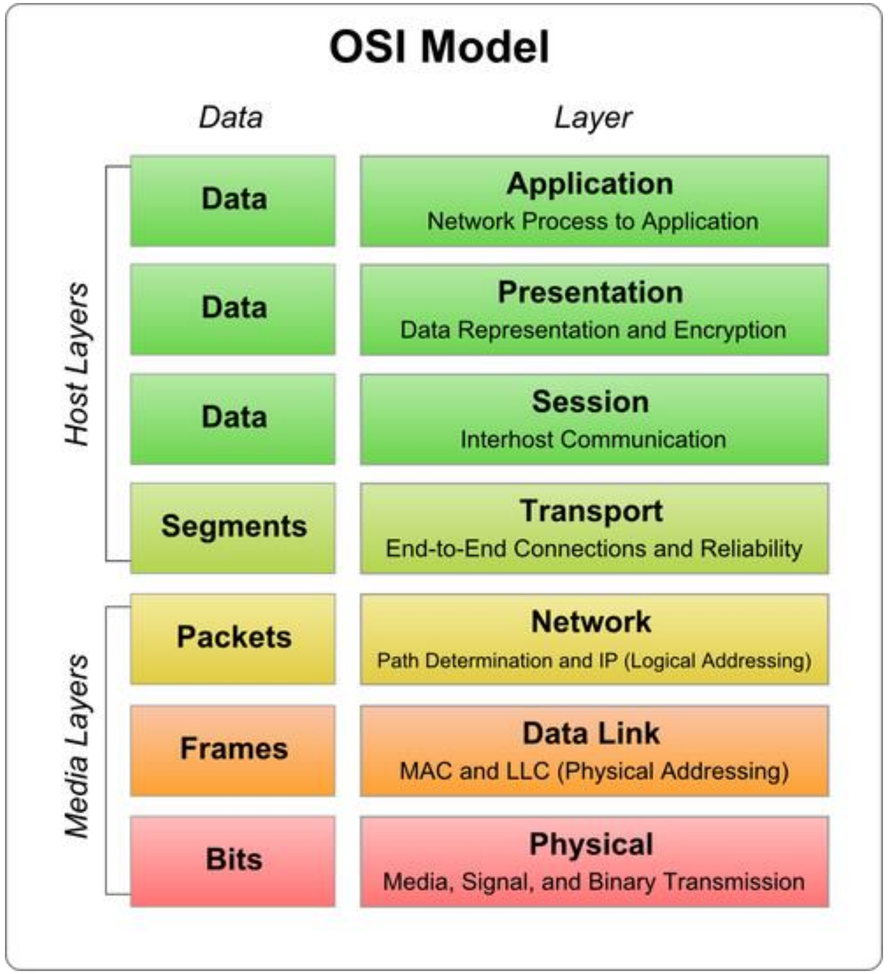


Image from: http://www.tech-faq.com/osi-model.html

OSI in terms of debugging errors



Can your browser open another website?

Do you have a viewer that supports jpg (image format)?

Can you ping the webserver you are trying to reach?

Can you ping the gateway or DNS server?

Do you have an IP address?

Is the light on the modem on?

Is the network cable plugged in?

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
          "http://www.w3.org/TR/html4/loose.dtd">
<!--DOCTYPE Needs to be the very first thing on the page, or IE 6 goes
    into quirks mode, rather than standards mode -->
<!--DOCUMENT STARTS-->
<!--START:/ssi/doctype.inc-->
<html>
<head>
<!--END:/ssi/doctype.inc-->
<!--TITLE HERE-->

<TITLE>Computer Security Course - University of Edinburgh School of Inform
<!--START:/cgi-bin/metabase-->
<!-- Metadata information automatically generated -->
<META NAME="DC.Title" CONTENT="Computer Security Course - University of Ed
<META NAME="DC.Creator" CONTENT="Neil Brown">
<META NAME="DC.Creator.Address" CONTENT="neilb@inf.ed.ac.uk">
```

Sender: Apache

7	**Application** Network process to application
6	**Presentation** Data representation and encryption
5	**Session** Interhost communication
4	**Transport** End-to-end connection and reliability
3	**Network** Path determination and IP (Logical Addressing)
2	**Data Link** MAC and LLC (Physical Addressing)
1	**Physical** Media, signal, and binary transmission

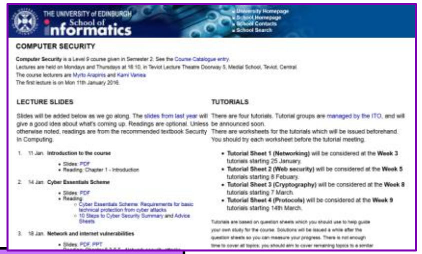
Data starts at the top of the OSI stack at level 7.

It progresses down the stack with each successive level adding or changing information.

At level 1 it travels across the physical layer to the recipient computer.

The recipient then processes the data up the stack. At level 7 an application processes the data.

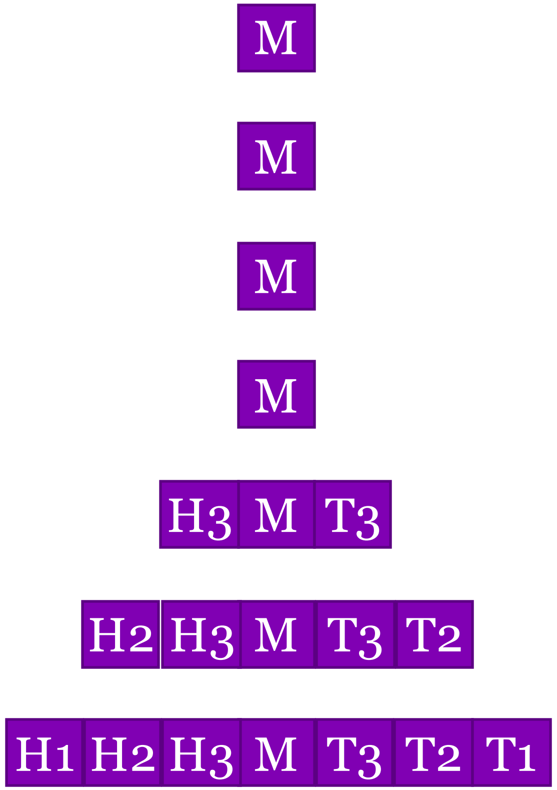
Recipient: Firefox user



7	**Application** Network process to application
6	**Presentation** Data representation and encryption
5	**Session** Interhost communication
4	**Transport** End-to-end connection and reliability
3	**Network** Path determination and IP (Logical Addressing)
2	**Data Link** MAC and LLC (Physical Addressing)
1	**Physical** Media, signal, and binary transmission

- Levels 7 and 6 involve the internal representation of the message
- Levels 5 and 4 involve setting up the connection
- Levels 3, 2, and 1 add header (H) and tail (T) information to each packet

Information is added to the message as it travels down the OSI levels

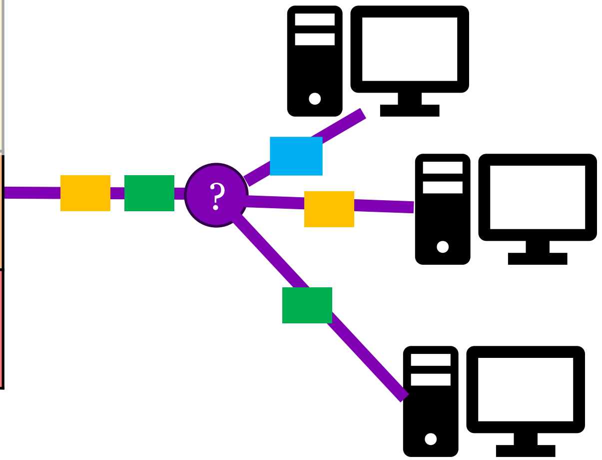


7	**Application** Network process to application
6	**Presentation** Data representation and encryption
5	**Session** Interhost communication
4	**Transport** End-to-end connection and reliability
3	**Network** Path determination and IP (Logical Addressing)
2	**Data Link** MAC and LLC (Physical Addressing)
1	**Physical** Media, signal, and binary transmission

Routers should be able to route traffic without understanding message content

7	**Application** Network process to application
6	**Presentation** Data representation and encryption
5	**Session** Interhost communication
4	**Transport** End-to-end connection and reliability
3	**Network** Path determination and IP (Logical Addressing)
2	**Data Link** MAC and LLC (Physical Addressing)
1	**Physical** Media, signal, and binary transmission

7	**Application** Network process to application
6	**Presentation** Data representation and encryption
5	**Session** Interhost communication
4	**Transport** End-to-end connection and reliability
3	**Network** Path determination and IP (Logical Addressing)
2	**Data Link** MAC and LLC (Physical Addressing)
1	**Physical** Media, signal, and binary transmission

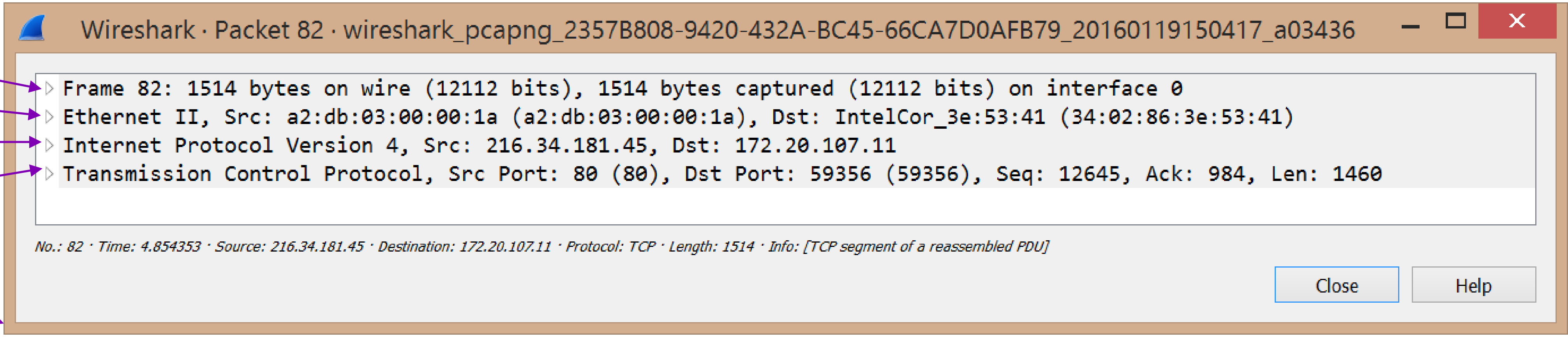


Header data on a packet

1. Physical
2. Data link
3. Network
4. Transport

…

7. Application

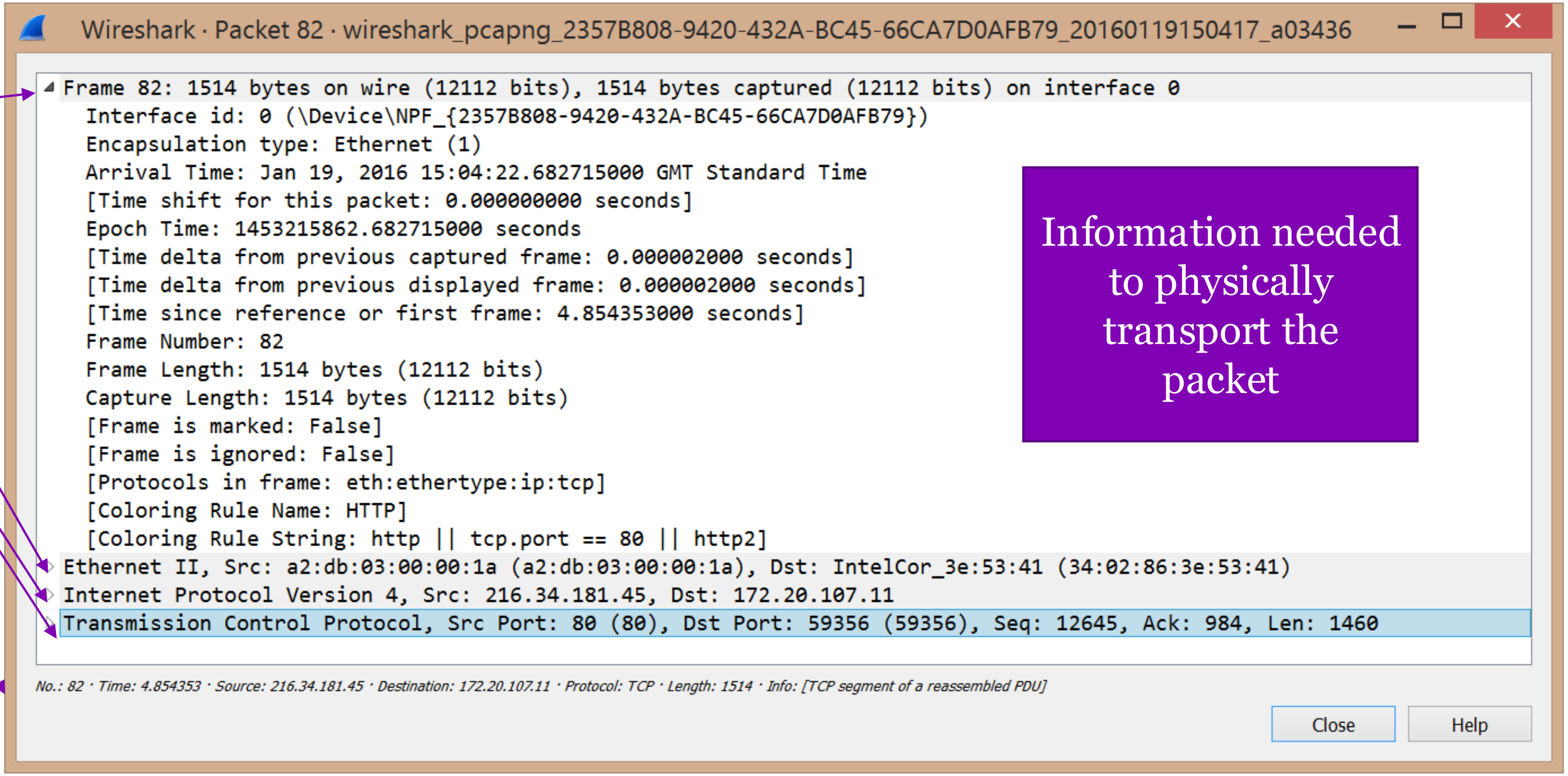


Frame header data on a packet

1. Physical
2. Data link
3. Network
4. Transport

…

7. Application

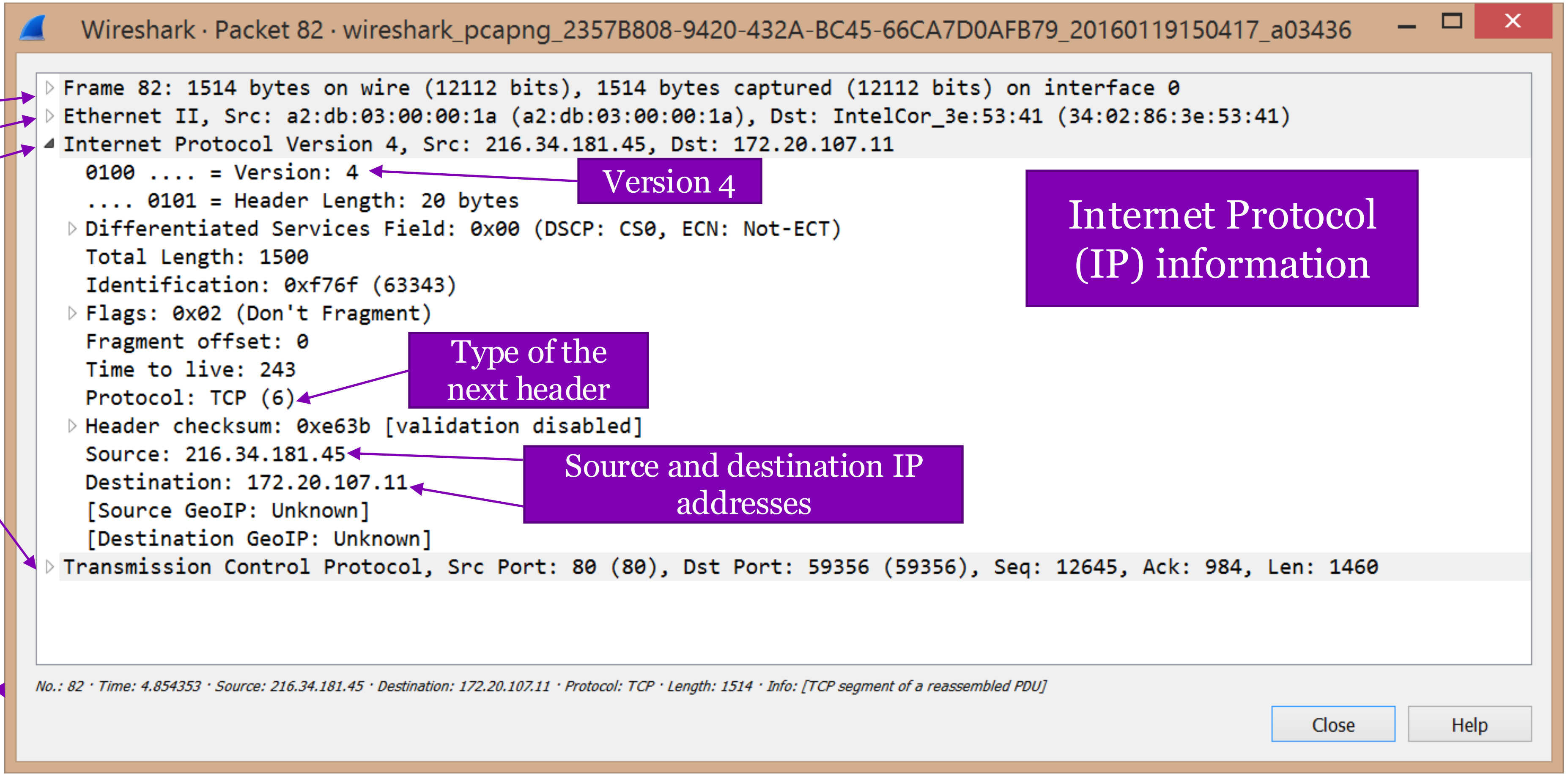


IP header data on a packet

1. Physical
2. Data link
3. Network
4. Transport

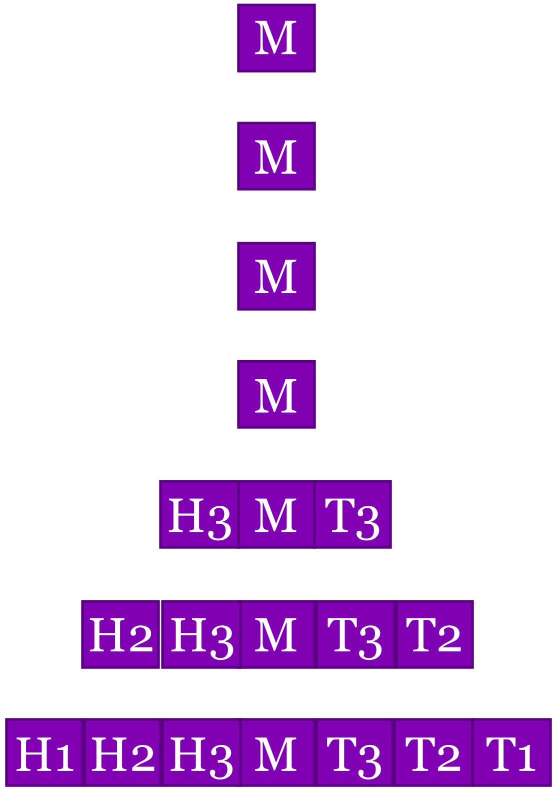
...

7. Application



Information is added to the message as it travels down the OSI levels

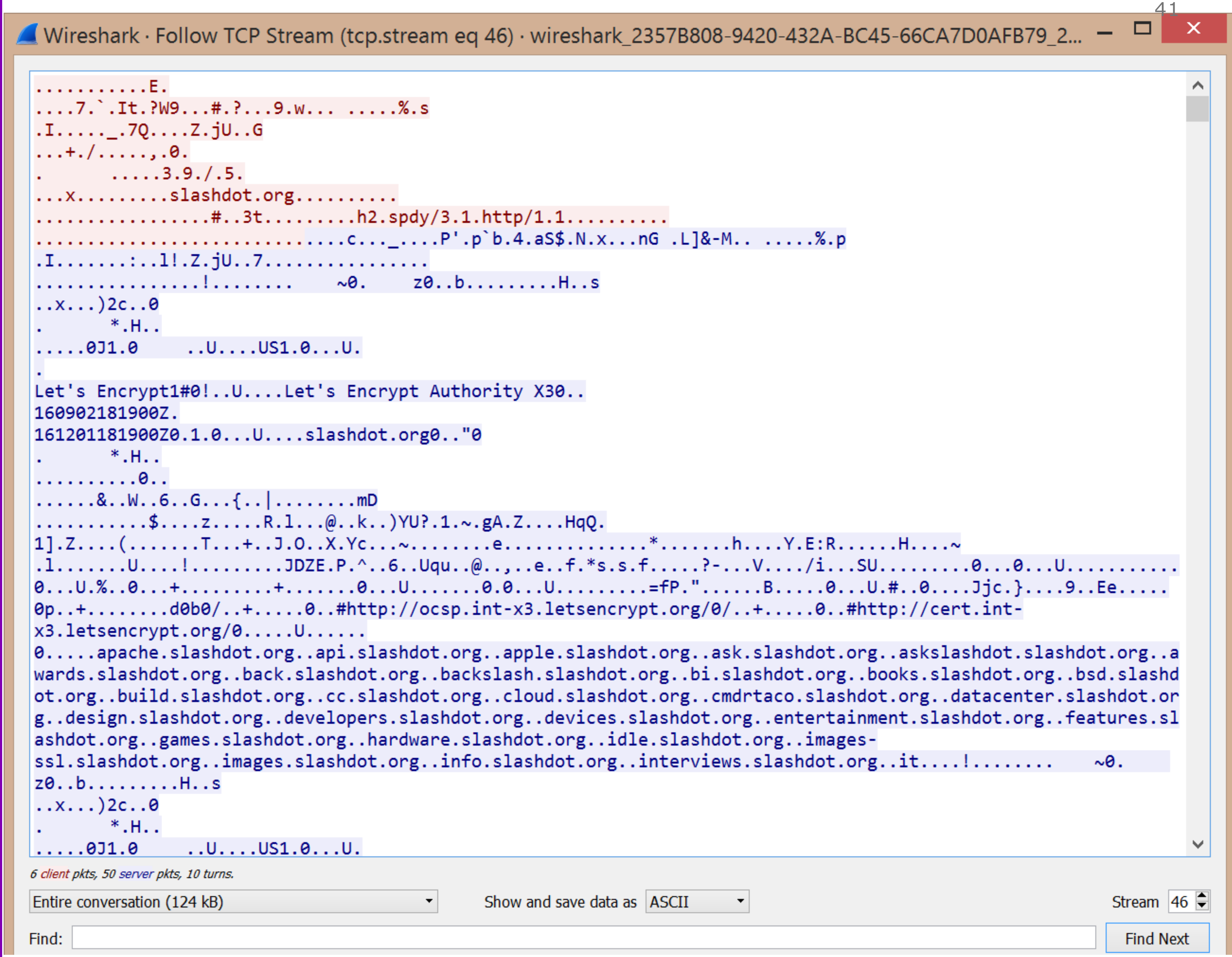
- Levels 7 and 6 involve the internal representation of the message
- Levels 5 and 4 involve setting up the connection
- Levels 3, 2, and 1 add header (H) and tail (T) information to each packet



7	**Application** Network process to application
6	**Presentation** Data representation and encryption
5	**Session** Interhost communication
4	**Transport** End-to-end connection and reliability
3	**Network** Path determination and IP (Logical Addressing)
2	**Data Link** MAC and LLC (Physical Addressing)
1	**Physical** Media, signal, and binary transmission

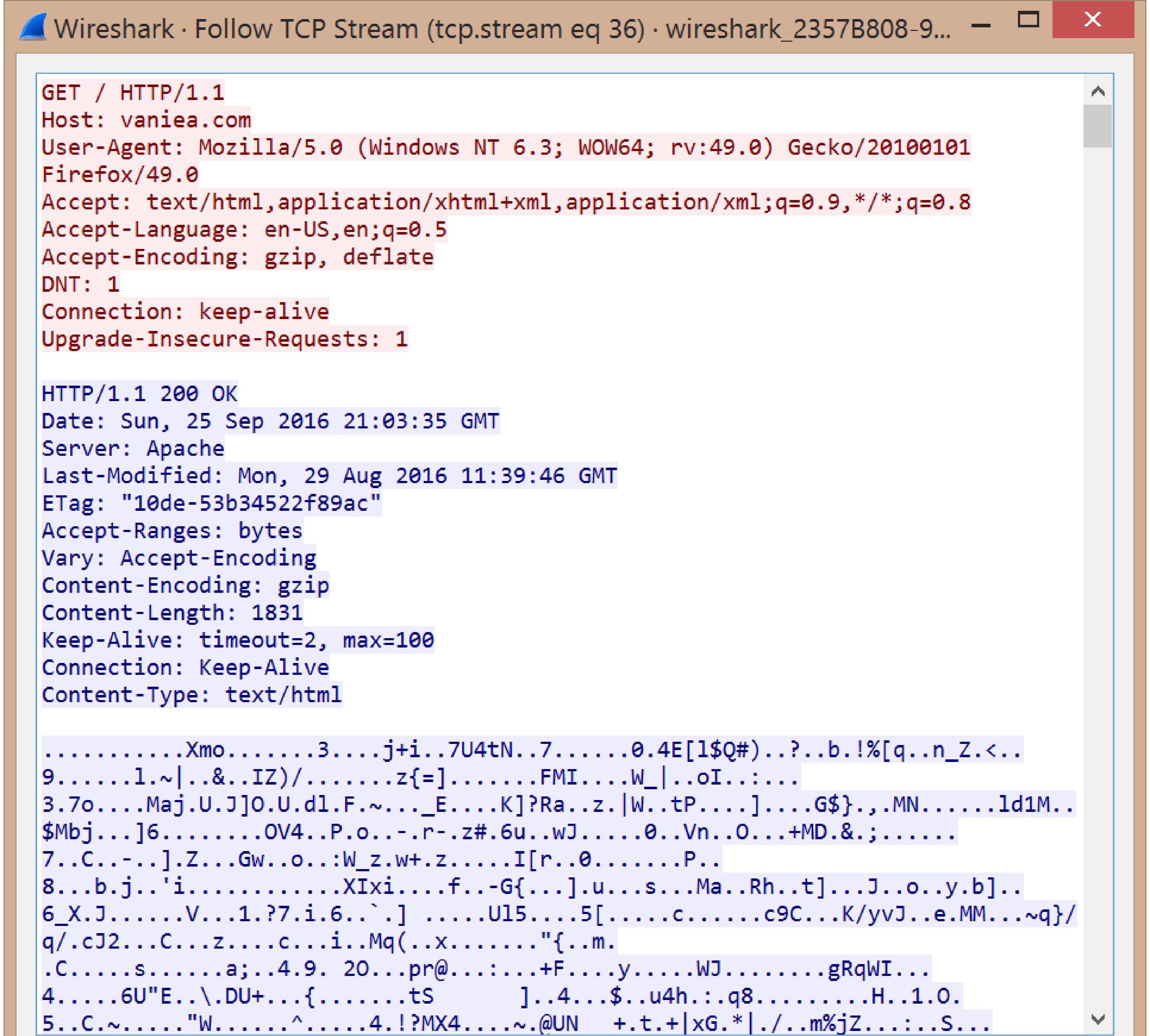
This is me visiting https://slashdot.org

- 6 packets were sent from my computer to the server
- 50 packets were sent from the server to my computer



This is me visiting http://vaniea.com

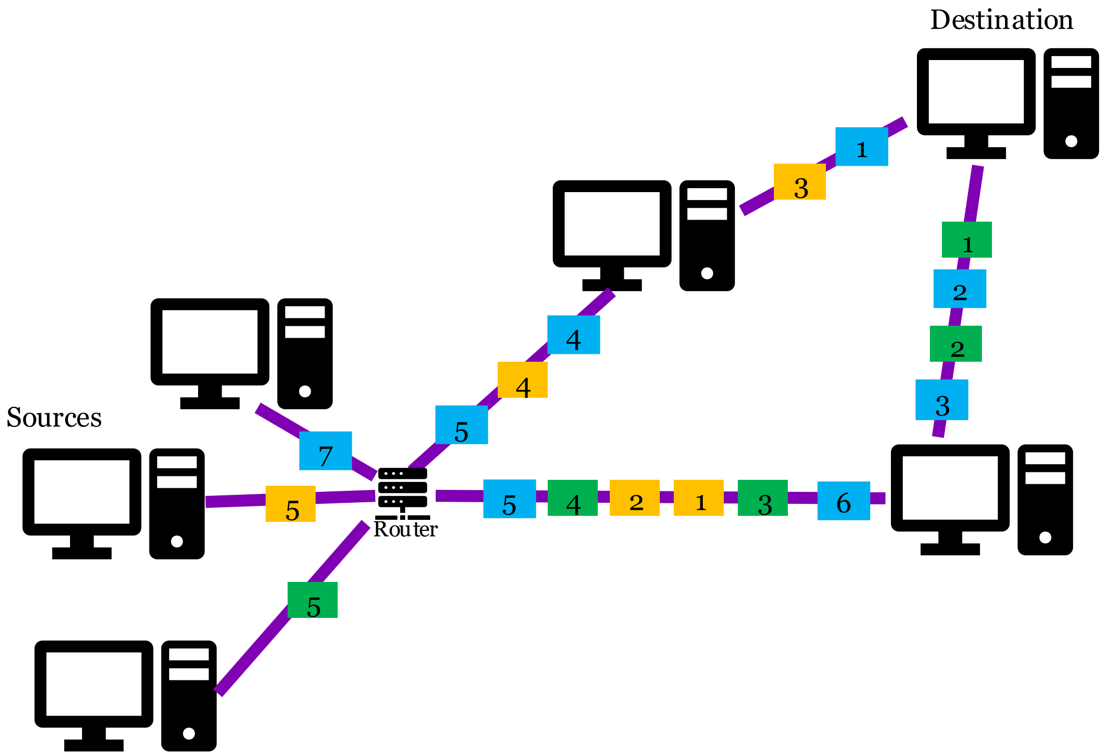
- Note the lack of https
- Why does the text look garbled anyway?



TRANSMISSION CONTROL PROTOCOL (TCP)

Transmission Control Protocol (TCP)

- Problem:
  - The network isn't very reliable, packets get lost
  - Packets arrive out-of-order
  - Congestion forces network slow-down
- Applications need the packets to all get there and be in order
- Connection needs to look invisible to upper OSI network model levels



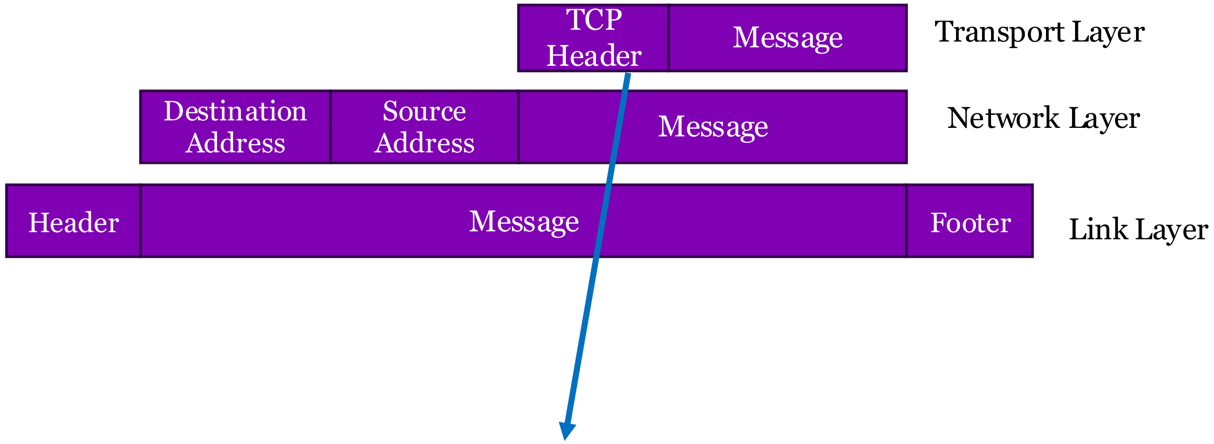
Applications should be able to send across the network without understanding network structures or issues

7	**Application** Network process to application
6	**Presentation** Data representation and encryption
5	**Session** Interhost communication
4	**Transport** End-to-end connection and reliability
3	**Network** Path determination and IP (Logical Addressing)
2	**Data Link** MAC and LLC (Physical Addressing)
1	**Physical** Media, signal, and binary transmission

7	**Application** Network process to application
6	**Presentation** Data representation and encryption
5	**Session** Interhost communication
4	**Transport** End-to-end connection and reliability
3	**Network** Path determination and IP (Logical Addressing)
2	**Data Link** MAC and LLC (Physical Addressing)
1	**Physical** Media, signal, and binary transmission

TCP

- TCP creates a "pipe" where it makes sure data is transferred with high reliability and not lost
- TCP is good for reliability and less good for speed
- TCP breaks message into smaller messages that fit well in packets
- Each packet is assigned tracking information (sequence numbers)



TCP segment header

<table>
<tr><th colspan="2">Offsets</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th></tr>
<tr><td>0</td><td>0</td><td colspan="16">Source port</td><td colspan="16">Destination port</td></tr>
<tr><td>4</td><td>32</td><td colspan="32">Sequence number</td></tr>
<tr><td>8</td><td>64</td><td colspan="32">Acknowledgment number (if ACK set)</td></tr>
<tr><td>12</td><td>96</td><td colspan="4">Data offset</td><td colspan="4">Reserved 0 0 0 0</td><td>CWR</td><td>ECE</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="16">Window Size</td></tr>
<tr><td>16</td><td>128</td><td colspan="16">Checksum</td><td colspan="16">Urgent pointer (if URG set)</td></tr>
<tr><td>20</td><td>160</td><td colspan="32" rowspan="3">Options (if data offset > 5. Padded at the end with "0" bits if necessary.)</td></tr>
<tr><td>⋮</td><td>⋮</td></tr>
<tr><td>56</td><td>448</td></tr>
</table>

https://en.wikipedia.org/wiki/Transmission_Control_Protocol

TCP: Three-part handshake

- TCP uses sequence numbers to ensure that all packets are arriving. The three-part-handshake sets up the connection and the randomly chosen sequence number start points
- Basic three-part handshake used by Alice to initiate a TCP connection with Bob.

$$A \rightarrow B: \quad \text{SYN}, X$$
$$B \rightarrow A: \quad \text{ACK}, X+1; \text{SYN}, Y$$
$$A \rightarrow B: \quad \text{ACK}, Y+1$$

- Following packets include sequence numbers (Alice sequence starting with X+2)
- Server Bob responds with Ack packets (with sequence starting Y+2) signaling how many of the packets have come through (X+n)

SYN

```
10.32.113.90     34.120.208.123   TCP   74 37010 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2121452297 TSecr=0 WS=128
34.120.208.123   10.32.113.90     TCP   74 443 → 37010 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1250 SACK_PERM=1 TSval=2389897682 TSecr=2121452297 WS=256
10.32.113.90     34.120.208.123   TCP   66 37010 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2121452305 TSecr=2389897682
```

Wireshark · Packet 203 · wlp0s20f3

```
▸ Frame 203: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp0s20f3, id 0
▸ Ethernet II, Src: IntelCor_ef:1b:77 (74:04:f1:ef:1b:77), Dst: Cisco_f3:e1:54 (d4:2c:44:f3:e1:54)
▸ Internet Protocol Version 4, Src: 10.32.113.90, Dst: 34.120.208.123
▾ Transmission Control Protocol, Src Port: 37010, Dst Port: 443, Seq: 0, Len: 0
    Source Port: 37010
    Destination Port: 443
    [Stream index: 4]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1049082332
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1010 .... = Header Length: 40 bytes (10)
  ▸ Flags: 0x002 (SYN)
    Window: 64240
    [Calculated window size: 64240]
    Checksum: 0x6e9c [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
  ▸ [Timestamps]

0000  d4 2c 44 f3 e1 54 74 04  f1 ef 1b 77 08 00 45 00   ·,D··Tt· ··w··E·
0010  00 3c 41 ed 40 00 40 06  8a 61 0a 20 71 5a 22 78   ·<A·@·@· ·a· qZ"x
0020  d0 7b 90 92 01 bb 3e 87  b9 dc 00 00 00 00 a0 02   ·{····>· ········
0030  fa f0 6e 9c 00 00 02 04  05 b4 04 02 08 0a 7e 72   ··n····· ······~r
```

Help   Close

SYN, ACK

```
10.32.113.90     34.120.208.123   TCP   74 37010 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2121452297 TSecr=0 WS=128
34.120.208.123   10.32.113.90     TCP   74 443 → 37010 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1250 SACK_PERM=1 TSval=2389897682 TSecr=2121452297 WS=256
10.32.113.90     34.120.208.123   TCP   66 37010 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2121452305 TSecr=2389897682
```

Wireshark · Packet 204 · wlp0s20f3

```
Frame 204: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp0s20f3, id 0
Ethernet II, Src: Cisco_f3:e1:54 (d4:2c:44:f3:e1:54), Dst: IntelCor_ef:1b:77 (74:04:f1:ef:1b:77)
Internet Protocol Version 4, Src: 34.120.208.123, Dst: 10.32.113.90
Transmission Control Protocol, Src Port: 443, Dst Port: 37010, Seq: 0, Ack: 1, Len: 0
    Source Port: 443
    Destination Port: 37010
    [Stream index: 4]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1537189781
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1049082333
    1010 .... = Header Length: 40 bytes (10)
    Flags: 0x012 (SYN, ACK)
    Window: 65535
    [Calculated window size: 65535]
    Checksum: 0x82aa [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
    Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
    [Timestamps]
    [SEQ/ACK analysis]
```

```
0000  74 04 f1 ef 1b 77 d4 2c  44 f3 e1 54 08 00 45 00   t···w·,D··T··E·
0010  00 3c 00 00 40 00 73 06  99 4e 22 78 d0 7b 0a 20   ·<··@·s··N"x·{·
0020  71 5a 01 bb 90 92 5b 9f  a7 95 3e 87 b9 dd a0 12   qZ····[···>·····
```

Help   Close

ACK

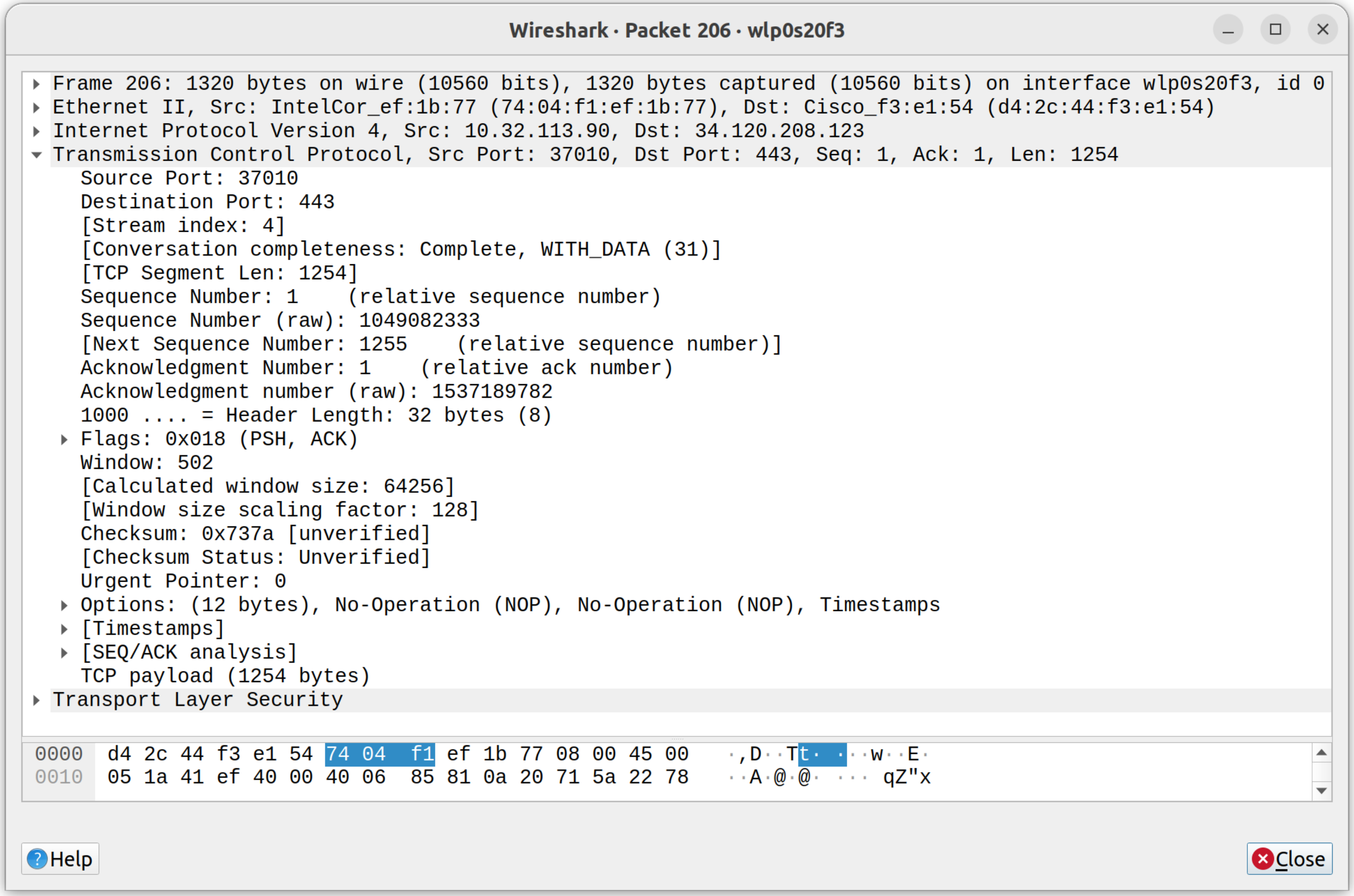
```
10.32.113.90     34.120.208.123   TCP   74 37010 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2121452297 TSecr=0 WS=128
34.120.208.123   10.32.113.90     TCP   74 443 → 37010 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1250 SACK_PERM=1 TSval=2389897682 TSecr=2121452297 WS=256
10.32.113.90     34.120.208.123   TCP   66 37010 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2121452305 TSecr=2389897682
```

Wireshark · Packet 205 · wlp0s20f3

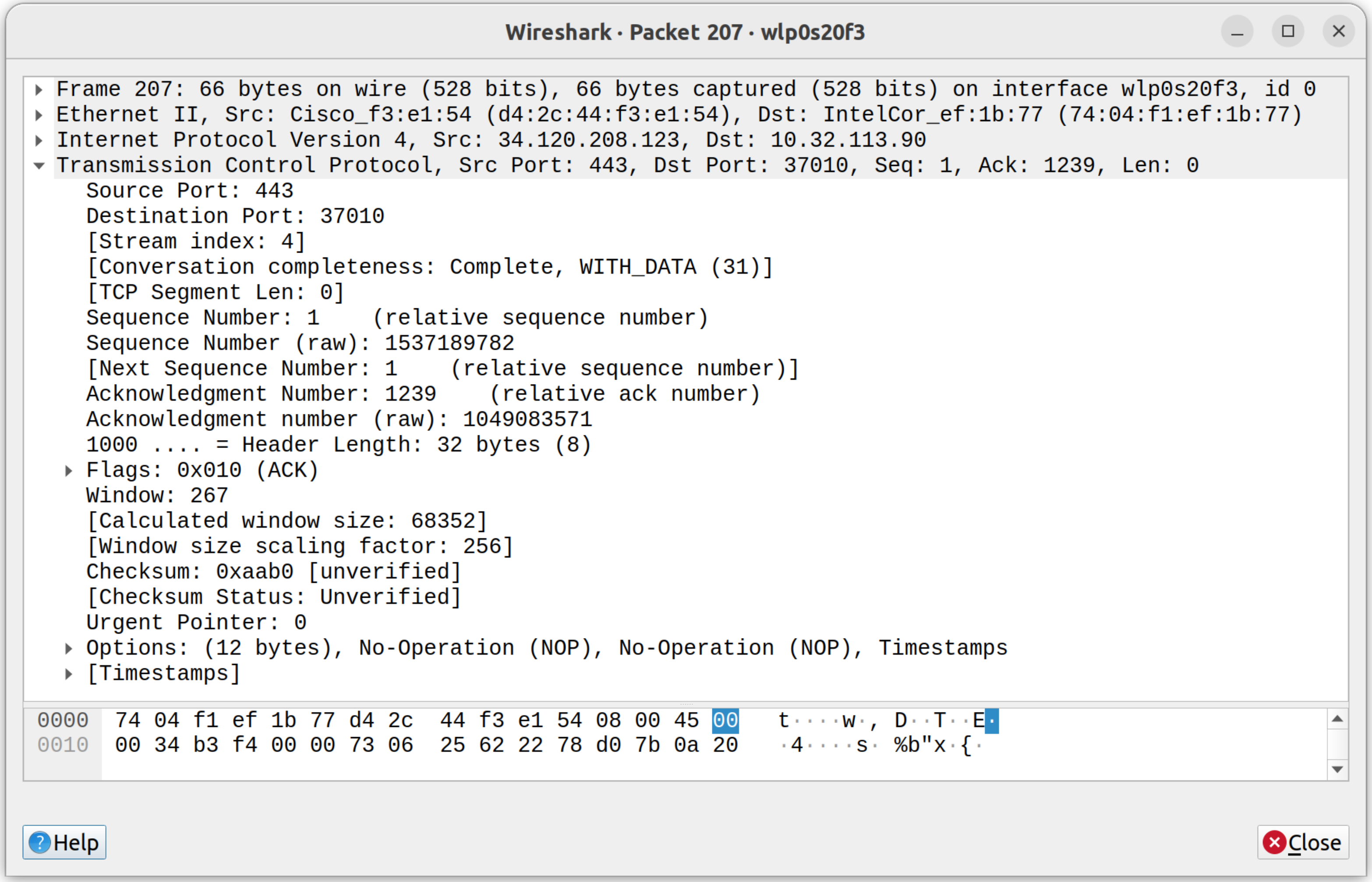
```
▸ Frame 205: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface wlp0s20f3, id 0
▸ Ethernet II, Src: IntelCor_ef:1b:77 (74:04:f1:ef:1b:77), Dst: Cisco_f3:e1:54 (d4:2c:44:f3:e1:54)
▸ Internet Protocol Version 4, Src: 10.32.113.90, Dst: 34.120.208.123
▾ Transmission Control Protocol, Src Port: 37010, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
    Source Port: 37010
    Destination Port: 443
    [Stream index: 4]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 1049082333
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1537189782
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window: 502
    [Calculated window size: 64256]
    [Window size scaling factor: 128]
    Checksum: 0x6e94 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [Timestamps]
  ▸ [SEQ/ACK analysis]

0000  d4 2c 44 f3 e1 54 74 04  f1 ef 1b 77 08 00 45 00   ·,D··Tt· ···w··E·
0010  00 34 41 ee 40 00 40 06  8a 68 0a 20 71 5a 22 78   ·4A·@·@· ·h· qZ"x
0020  d0 7b 90 92 01 bb 3e 87  b9 dd 5b 9f a7 96 80 10   ·{····>· ··[·····
```

Help   Close

TCP: First message of conversation (TLS setup)



First ACK post handshake

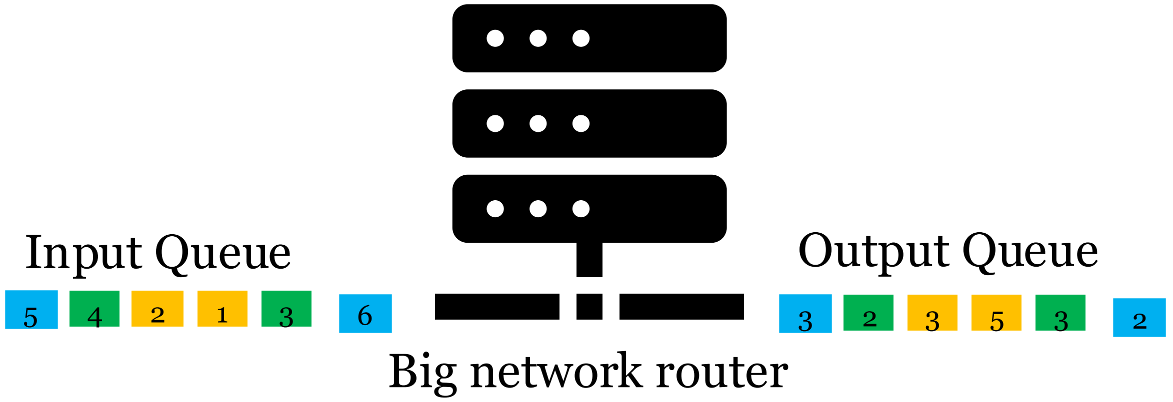


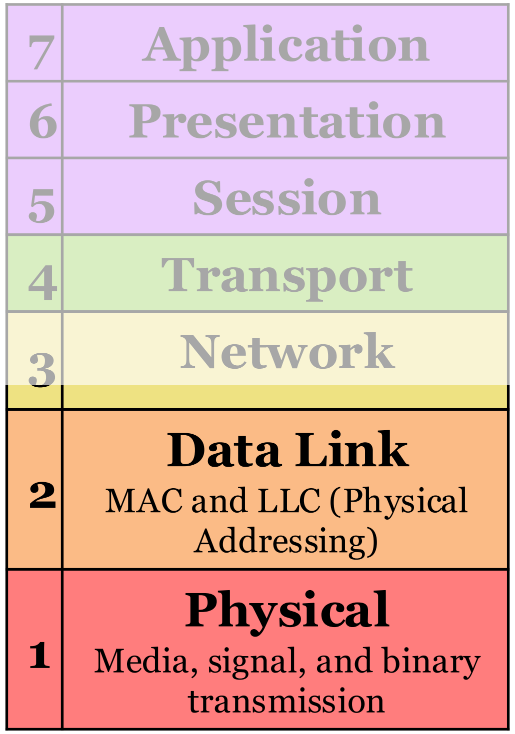
TCP connection works by coordinating the sequence numbers

Time	Source	Destination	Protocol	Length	Info
5.096284559	10.32.113.90	34.120.208.123	TCP	74	37010 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2121452297 TSecr=0 W
5.104798465	34.120.208.123	10.32.113.90	TCP	74	443 → 37010 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1250 SACK_PERM=1 TSval=238989768
5.104842223	10.32.113.90	34.120.208.123	TCP	66	37010 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2121452305 TSecr=2389897682
5.106365475	10.32.113.90	34.120.208.123	TLSv1.3	1320	Client Hello
5.114125383	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=1 Ack=1239 Win=68352 Len=0 TSval=2389897692 TSecr=2121452307
5.116321749	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=1 Ack=1255 Win=68352 Len=0 TSval=2389897694 TSecr=2121452307
5.138999171	34.120.208.123	10.32.113.90	TLSv1.3	3578	Server Hello, Change Cipher Spec, Application Data
5.139033771	10.32.113.90	34.120.208.123	TCP	66	37010 → 443 [ACK] Seq=1255 Ack=3513 Win=60800 Len=0 TSval=2121452340 TSecr=2389897716
5.142016312	10.32.113.90	34.120.208.123	TLSv1.3	130	Change Cipher Spec, Application Data
5.142391566	10.32.113.90	34.120.208.123	TLSv1.3	236	Application Data
5.142423997	10.32.113.90	34.120.208.123	TLSv1.3	1693	Application Data, Application Data
5.148618301	34.120.208.123	10.32.113.90	TLSv1.3	97	[TCP Previous segment not captured] , Application Data
5.148651920	10.32.113.90	34.120.208.123	TCP	78	[TCP Dup ACK 211#1] 37010 → 443 [ACK] Seq=3116 Ack=3513 Win=64128 Len=0 TSval=212145234
5.148618763	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=3513 Ack=2727 Win=73728 Len=0 TSval=2389897727 TSecr=2121452343
5.148618820	34.120.208.123	10.32.113.90	TCP	652	[TCP Out-Of-Order] 443 → 37010 [PSH, ACK] Seq=3513 Ack=2727 Win=73728 Len=586 TSval=238
5.148681362	10.32.113.90	34.120.208.123	TCP	66	37010 → 443 [ACK] Seq=3116 Ack=4130 Win=63616 Len=0 TSval=2121452349 TSecr=2389897727
5.149103475	10.32.113.90	34.120.208.123	TLSv1.3	97	Application Data
5.154335724	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=4130 Ack=3116 Win=76288 Len=0 TSval=2389897733 TSecr=2121452343
5.154335959	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=4130 Ack=3147 Win=76288 Len=0 TSval=2389897733 TSecr=2121452350
5.230705410	34.120.208.123	10.32.113.90	TLSv1.3	515	Application Data
5.230705747	34.120.208.123	10.32.113.90	TLSv1.3	353	Application Data
5.230951852	10.32.113.90	34.120.208.123	TCP	66	37010 → 443 [ACK] Seq=3147 Ack=4866 Win=64128 Len=0 TSval=2121452432 TSecr=2389897810
5.231846280	34.120.208.123	10.32.113.90	TLSv1.3	338	Application Data
5.232185942	34.120.208.123	10.32.113.90	TLSv1.3	105	Application Data
5.232524812	10.32.113.90	34.120.208.123	TCP	66	37010 → 443 [ACK] Seq=3147 Ack=5177 Win=64128 Len=0 TSval=2121452433 TSecr=2389897810
5.232539206	10.32.113.90	34.120.208.123	TLSv1.3	105	Application Data
5.239771080	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=5177 Ack=3186 Win=76288 Len=0 TSval=2389897818 TSecr=2121452433
32.253414829	10.32.113.90	34.120.208.123	TLSv1.3	214	Application Data
32.253452478	10.32.113.90	34.120.208.123	TLSv1.3	1258	Application Data
32.260490237	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=5177 Ack=3334 Win=78592 Len=0 TSval=2389924839 TSecr=2121479454
32.260490569	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=5177 Ack=4526 Win=81408 Len=0 TSval=2389924839 TSecr=2121479454
32.362344786	34.120.208.123	10.32.113.90	TLSv1.3	134	Application Data
32.362345123	34.120.208.123	10.32.113.90	TLSv1.3	211	Application Data
32.362345159	34.120.208.123	10.32.113.90	TLSv1.3	252	Application Data
32.362345194	34.120.208.123	10.32.113.90	TLSv1.3	105	Application Data
32.362775818	10.32.113.90	34.120.208.123	TCP	66	37010 → 443 [ACK] Seq=4526 Ack=5615 Win=64128 Len=0 TSval=2121479563 TSecr=2389924941
32.362815884	10.32.113.90	34.120.208.123	TLSv1.3	105	Application Data
32.369943709	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=5615 Ack=4565 Win=81408 Len=0 TSval=2389924948 TSecr=2121479563

Data loss happens in the network because routers can only handle so much traffic

- Routers must process every packet that comes across
- They have a fixed-size queue of packets waiting to be processed
- If there are too many packets to fit in the queue, the router “drops” the packets without telling the source

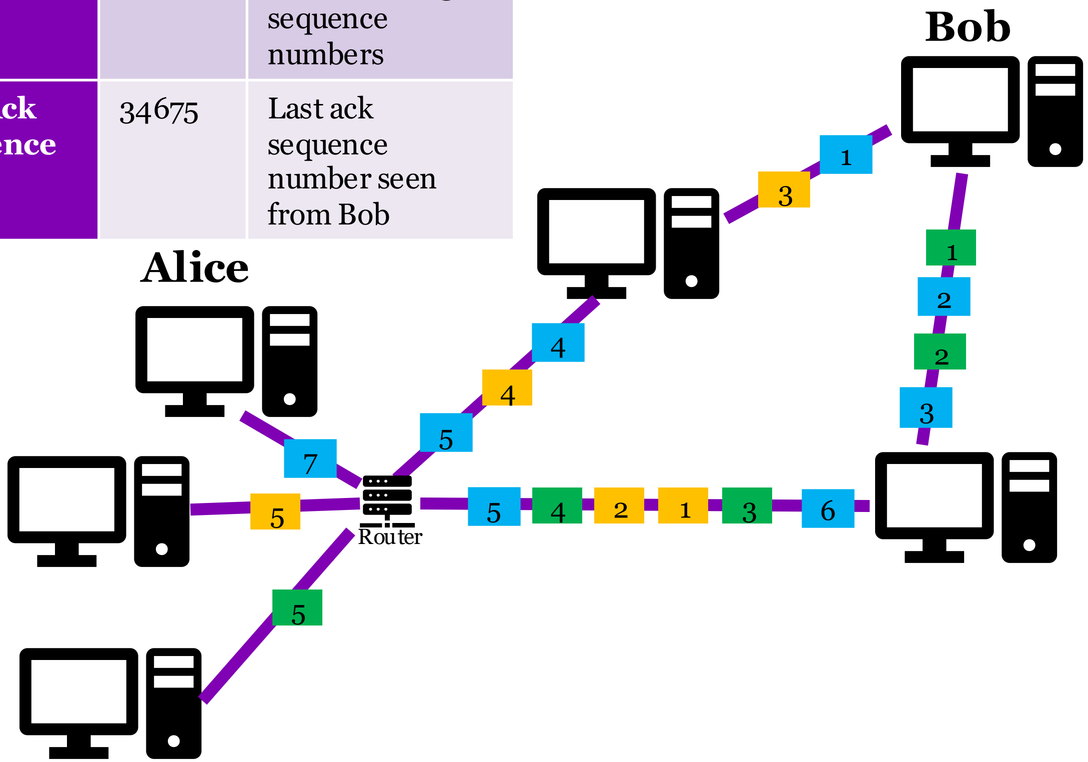




TCP detects and handles dropped packets

- Problem:
  - The network isn't very reliable, packets get lost
  - Packets arrive out-of-order
  - Congestion forces network slow-down
- Applications need the packets to all get there and be in order
- Connection needs to look invisible to upper OSI network model levels

Alice Sequence	23574 - 23654	Range of unacknowledged sequence numbers
Bob Ack Sequence	34675	Last ack sequence number seen from Bob



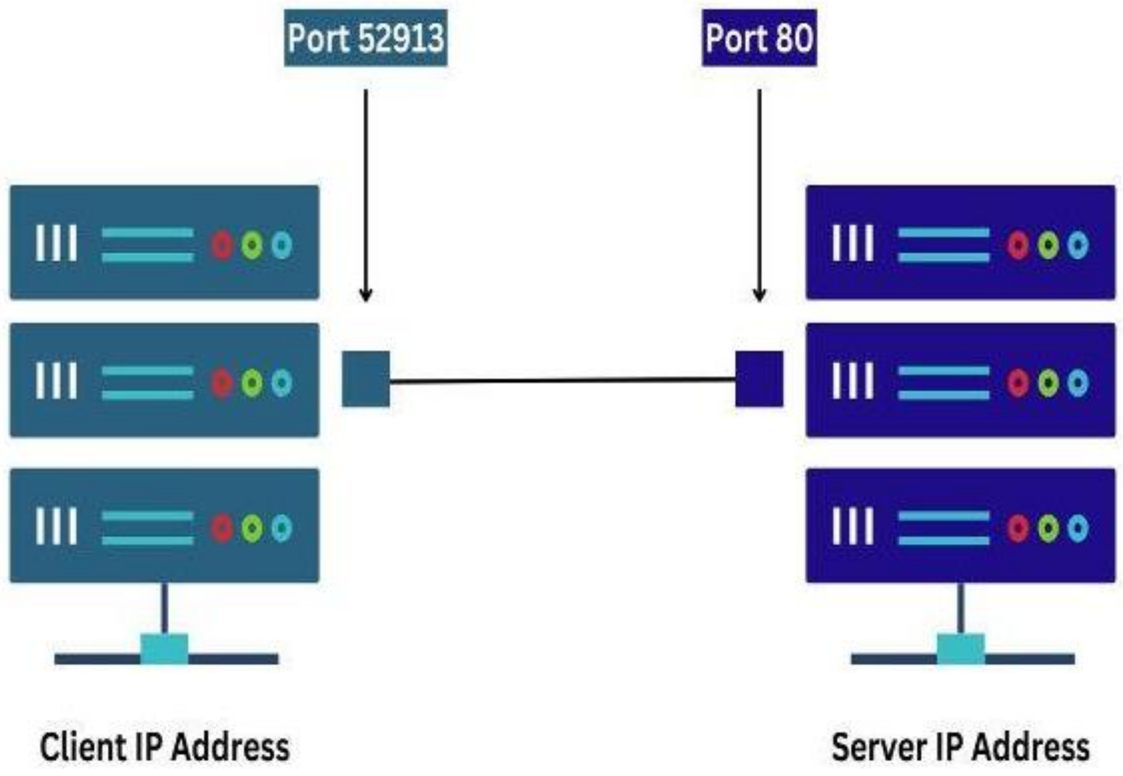
TCP connection works by coordinating the sequence numbers

Time	Source	Destination	Protocol	Length	Info
5.096284559	10.32.113.90	34.120.208.123	TCP	74	37010 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2121452297 TSecr=0 W
5.104798465	34.120.208.123	10.32.113.90	TCP	74	443 → 37010 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1250 SACK_PERM=1 TSval=238989768
5.104842223	10.32.113.90	34.120.208.123	TCP	66	37010 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2121452305 TSecr=2389897682
5.106365475	10.32.113.90	34.120.208.123	TLSv1.3	1320	Client Hello
5.114125383	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=1 Ack=1239 Win=68352 Len=0 TSval=2389897692 TSecr=2121452307
5.116321749	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=1 Ack=1255 Win=68352 Len=0 TSval=2389897694 TSecr=2121452307
5.138999171	34.120.208.123	10.32.113.90	TLSv1.3	3578	Server Hello, Change Cipher Spec, Application Data
5.139033771	10.32.113.90	34.120.208.123	TCP	66	37010 → 443 [ACK] Seq=1255 Ack=3513 Win=60800 Len=0 TSval=2121452340 TSecr=2389897716
5.142016312	10.32.113.90	34.120.208.123	TLSv1.3	130	Change Cipher Spec, Application Data
5.142391566	10.32.113.90	34.120.208.123	TLSv1.3	236	Application Data
5.142423997	10.32.113.90	34.120.208.123	TLSv1.3	1693	Application Data, Application Data
5.148618301	34.120.208.123	10.32.113.90	TLSv1.3	97	[TCP Previous segment not captured] , Application Data
5.148651920	10.32.113.90	34.120.208.123	TCP	78	[TCP Dup ACK 211#1] 37010 → 443 [ACK] Seq=3116 Ack=3513 Win=64128 Len=0 TSval=212145234
5.148618763	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=3513 Ack=2727 Win=73728 Len=0 TSval=2389897727 TSecr=2121452343
5.148618820	34.120.208.123	10.32.113.90	TCP	652	[TCP Out-Of-Order] 443 → 37010 [PSH, ACK] Seq=3513 Ack=2727 Win=73728 Len=586 TSval=238
5.148681362	10.32.113.90	34.120.208.123	TCP	66	37010 → 443 [ACK] Seq=3116 Ack=4130 Win=63616 Len=0 TSval=2121452349 TSecr=2389897727
5.149103475	10.32.113.90	34.120.208.123	TLSv1.3	97	Application Data
5.154335724	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=4130 Ack=3116 Win=76288 Len=0 TSval=2389897733 TSecr=2121452343
5.154335959	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=4130 Ack=3147 Win=76288 Len=0 TSval=2389897733 TSecr=2121452350
5.230705410	34.120.208.123	10.32.113.90	TLSv1.3	515	Application Data
5.230705747	34.120.208.123	10.32.113.90	TLSv1.3	353	Application Data
5.230951852	10.32.113.90	34.120.208.123	TCP	66	37010 → 443 [ACK] Seq=3147 Ack=4866 Win=64128 Len=0 TSval=2121452432 TSecr=2389897810
5.231846280	34.120.208.123	10.32.113.90	TLSv1.3	338	Application Data
5.232185942	34.120.208.123	10.32.113.90	TLSv1.3	105	Application Data
5.232524812	10.32.113.90	34.120.208.123	TCP	66	37010 → 443 [ACK] Seq=3147 Ack=5177 Win=64128 Len=0 TSval=2121452433 TSecr=2389897810
5.232539206	10.32.113.90	34.120.208.123	TLSv1.3	105	Application Data
5.239771080	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=5177 Ack=3186 Win=76288 Len=0 TSval=2389897818 TSecr=2121452433
32.253414829	10.32.113.90	34.120.208.123	TLSv1.3	214	Application Data
32.253452478	10.32.113.90	34.120.208.123	TLSv1.3	1258	Application Data
32.260490237	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=5177 Ack=3334 Win=78592 Len=0 TSval=2389924839 TSecr=2121479454
32.260490569	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=5177 Ack=4526 Win=81408 Len=0 TSval=2389924839 TSecr=2121479454
32.362344786	34.120.208.123	10.32.113.90	TLSv1.3	134	Application Data
32.362345123	34.120.208.123	10.32.113.90	TLSv1.3	211	Application Data
32.362345159	34.120.208.123	10.32.113.90	TLSv1.3	252	Application Data
32.362345194	34.120.208.123	10.32.113.90	TLSv1.3	105	Application Data
32.362775818	10.32.113.90	34.120.208.123	TCP	66	37010 → 443 [ACK] Seq=4526 Ack=5615 Win=64128 Len=0 TSval=2121479563 TSecr=2389924941
32.362815884	10.32.113.90	34.120.208.123	TLSv1.3	105	Application Data
32.369943709	34.120.208.123	10.32.113.90	TCP	66	443 → 37010 [ACK] Seq=5615 Ack=4565 Win=81408 Len=0 TSval=2389924948 TSecr=2121479563

PORT

Ports

- MAC and IP addresses state which device to route to.
- But what software should handle each packet?
- We don't want to bother with a global list of software.
- Each software on the computer registers with a "port", which is just a number.



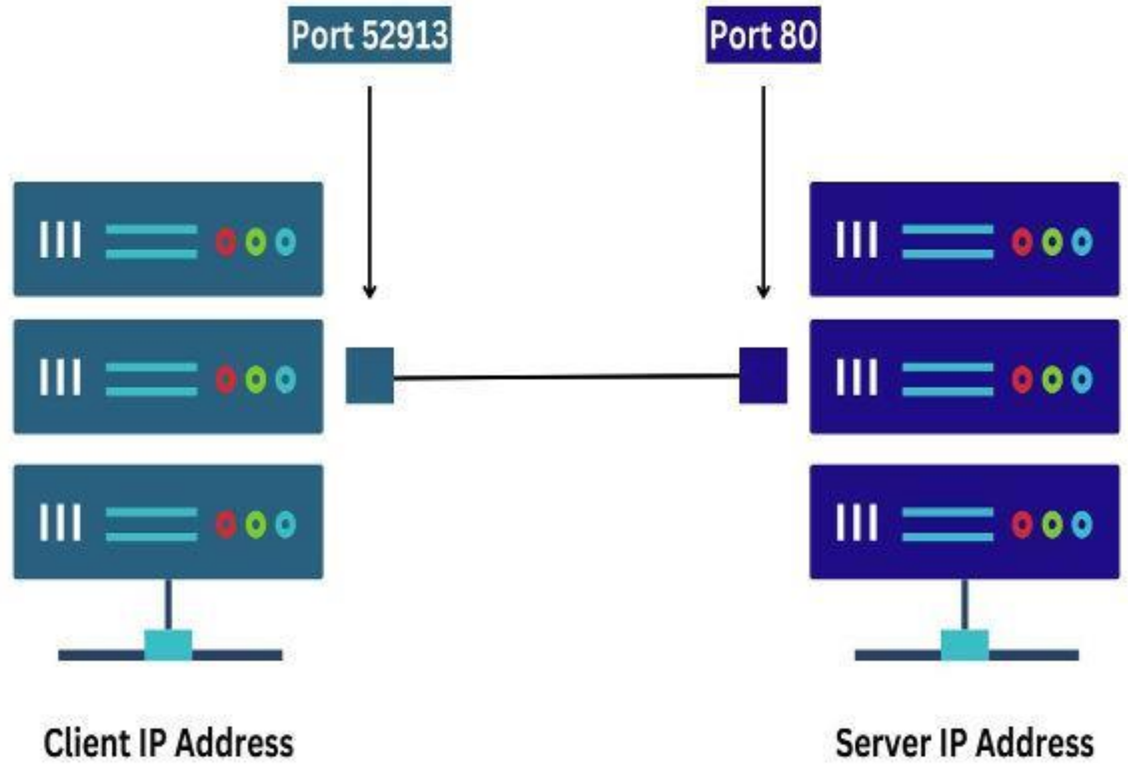
Ports

- The ports to the right are some of the most common externally visible ones.
- These ports are a *convention* not a required standard.
- If I want to run an SSH server on port 80, I can do so, though it will confuse allot of computers trying to talk to me.

Port Number	Usage
20	File Transfer Protocol (FTP) Data Transfer
21	File Transfer Protocol (FTP) Command Control
22	Secure Shell (SSH)
23	Telnet - Remote login service, unencrypted text messages
25	Simple Mail Transfer Protocol (SMTP) E-mail Routing
53	Domain Name System (DNS) service
80	Hypertext Transfer Protocol (HTTP) used in World Wide Web
110	Post Office Protocol (POP3) used by e-mail clients to retrieve e-mail from a server
119	Network News Transfer Protocol (NNTP)
123	Network Time Protocol (NTP)
143	Internet Message Access Protocol (IMAP) Management of Digital Mail
161	Simple Network Management Protocol (SNMP)
194	Internet Relay Chat (IRC)
443	HTTP Secure (HTTPS) HTTP over TLS/SSL

Ports

- All software talking over the network needs a port, but not all software is always listening.
- When internal software wants to connect externally (i.e. Firefox wants to visit a website) the client computer assigns a blank port to that software.

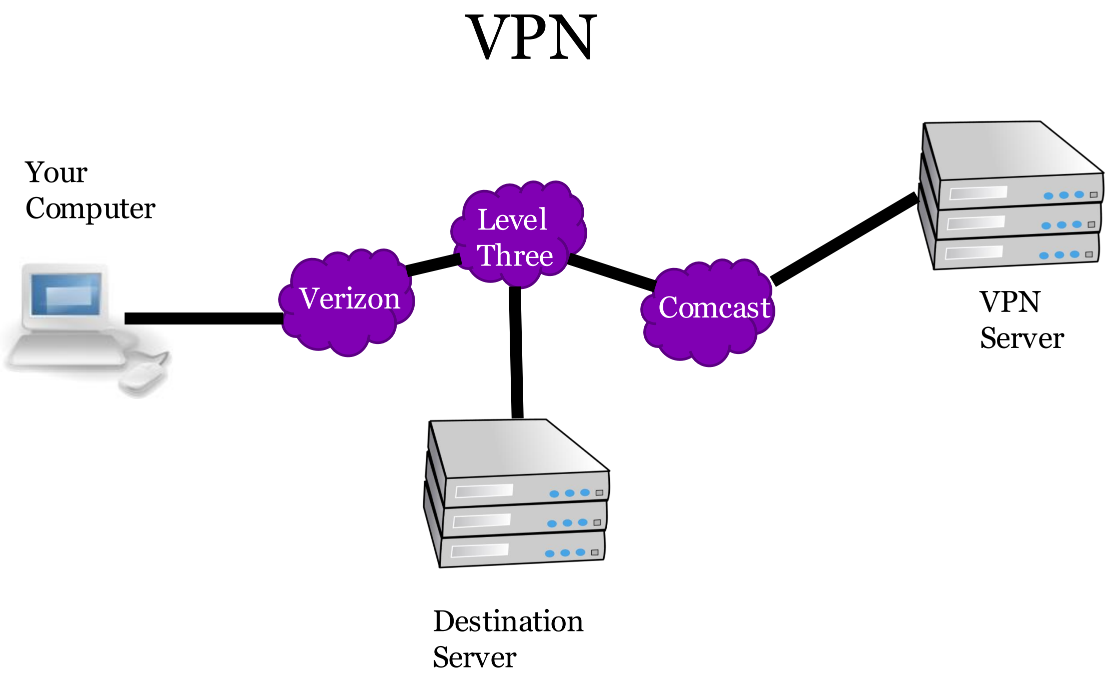


ONION ROUTING

aka Tor

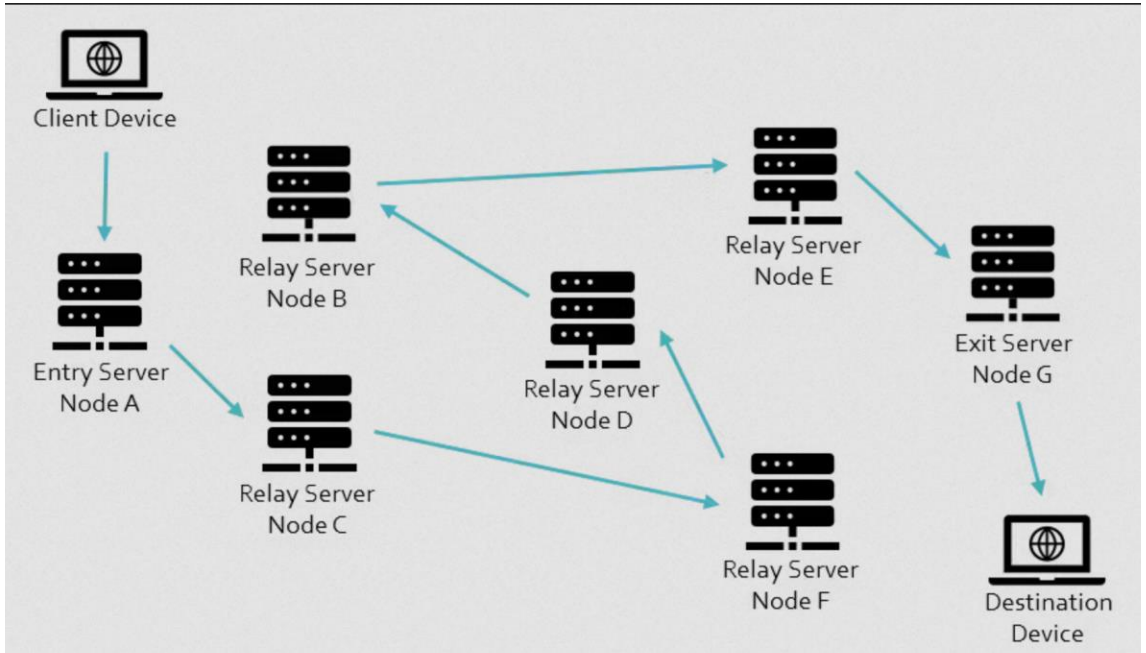
Problem: Anonymity

- VPN's were built to connect networks and ensure confidentiality of data
- Modern people also use them for anonymity, to hide what they are doing from a local threat
- But the VPN server has full knowledge of who it is connected to and all the traffic coming across
- Security weak point for hackers, governments, and law enforcement



Onion Routing

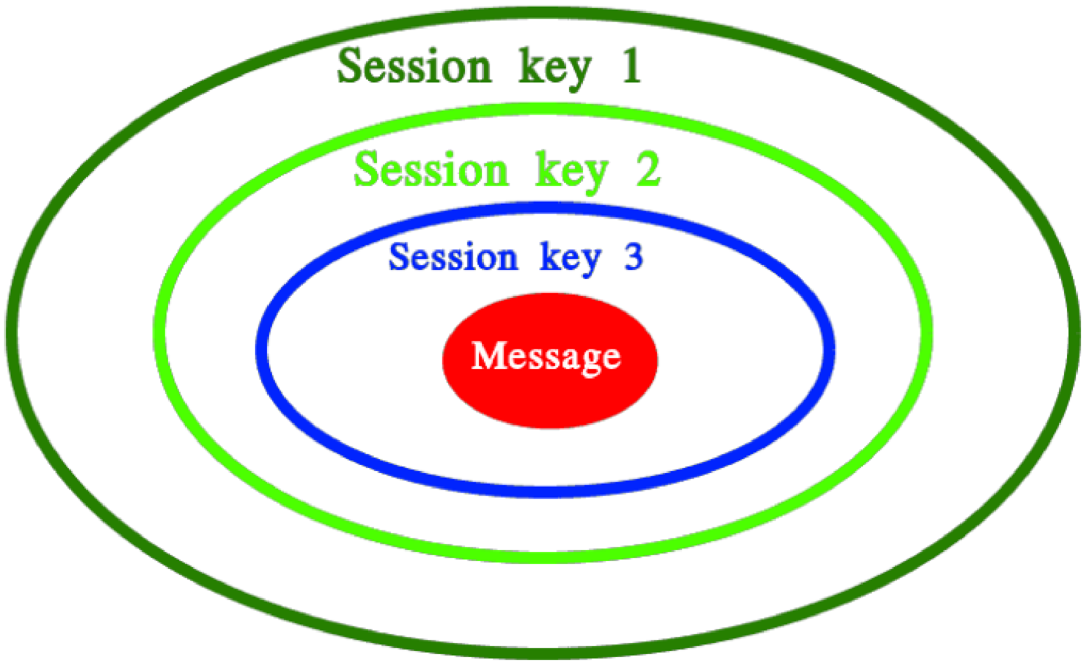
- Instead of one VPN server, how about we use several VPN servers
- Each server would know where it got traffic and where it sent traffic, but it won't know the whole path, just its neighbors
- This approach protects the client from connecting their real IP address with their traffic
  - First node (Node A) has the real IP address
  - Last node (Node G) has the real traffic
  - Nodes A and G do not know they are carrying the same person's traffic

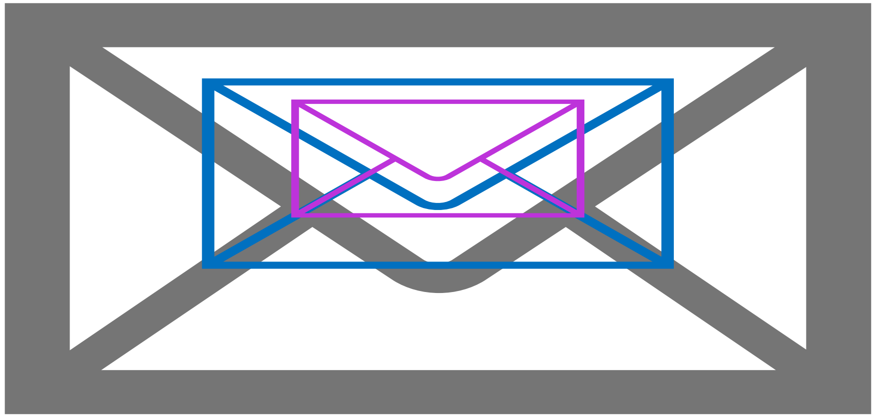


https://privacyhq.com/documentation/onion-routing-explained/

Onion Routing

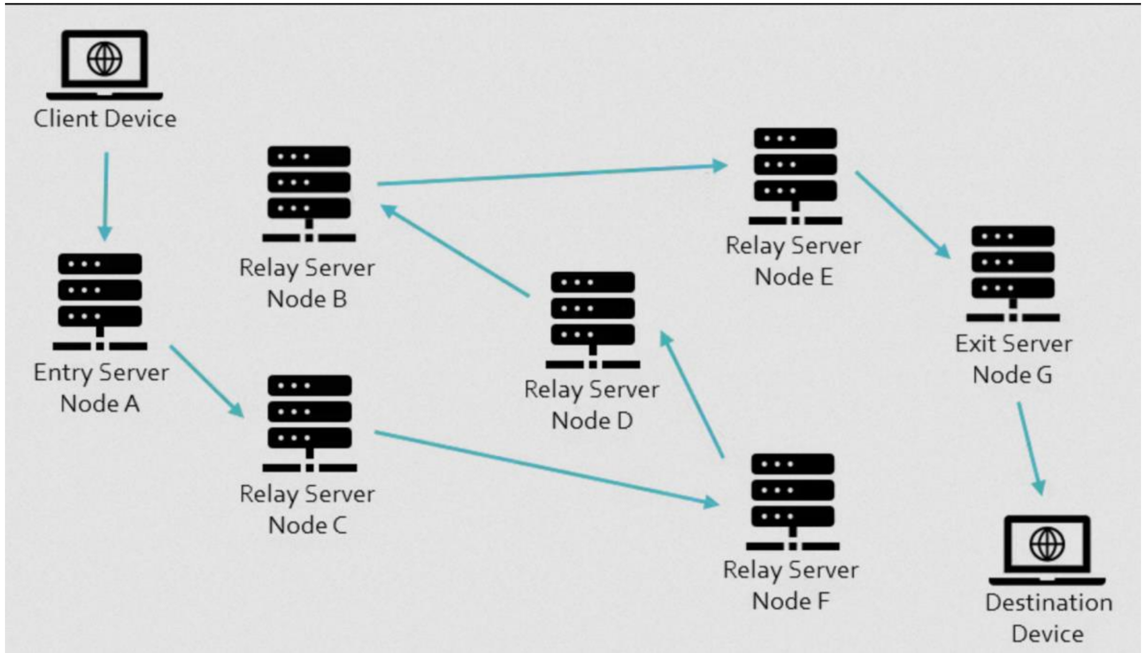
- Encryption is also important to make all this work
- The client has a list of all onion routers in network, they select a set and encrypt the message in concentric layers
- Each layer:
  - Encrypted with current node's public key
  - Address of next destination





Onion Routing

- Each Node only knows the address of where it got the packet and the address of where the packet is going
- They can only decrypt one layer of the packet
- Called onion routing because it is done in layers, with layers constructed by the client and then stripped off by the nodes



https://privacyhq.com/documentation/onion-routing-explained/

Tor

- Tor is popular software that uses onion routing
- There are many ways a user can still show who they are even if using Tor
  - Example: log into Facebook
- Routing everything across Tor could be bad because all exiting traffic could be connected together, so if one bit of traffic leaks your identity, they identity known for all traffic
- Tor is often bundled with a carefully setup browser

Example of research attempt to prevent DoS

- Idea: use capabilities (see access control slides)
- As part of a connection setup, the sender must get a capability from the destination
- Routers can verify the capabilities on packets and then discard attack packets

Xiaowei Yang, David Wetherall, and Thomas Anderson. 2005. A DoS-limiting network architecture. SIGCOMM Comput. Commun. Rev. 35, 4 (October 2005

QUESTIONS