1.2.3.4

The IP building blocks

Understanding the IP Protocol

IP Address

- Layer 3 property
- Can be set automatically or statically
- Network and Host portion
- 4 bytes in IPv4 - 32 bits

Network vs Host

- a.b.c.d/x (a.b.c.d are integers) x is the network bits and remains are host
- Example 192.168.254.0/24
- The first 24 bits (3 bytes) are network the rest 8 are for host
- This means we can have 2^24 (16777216) networks and each network has 2^8 (255) hosts
- Also called a subnet

Subnet Mask

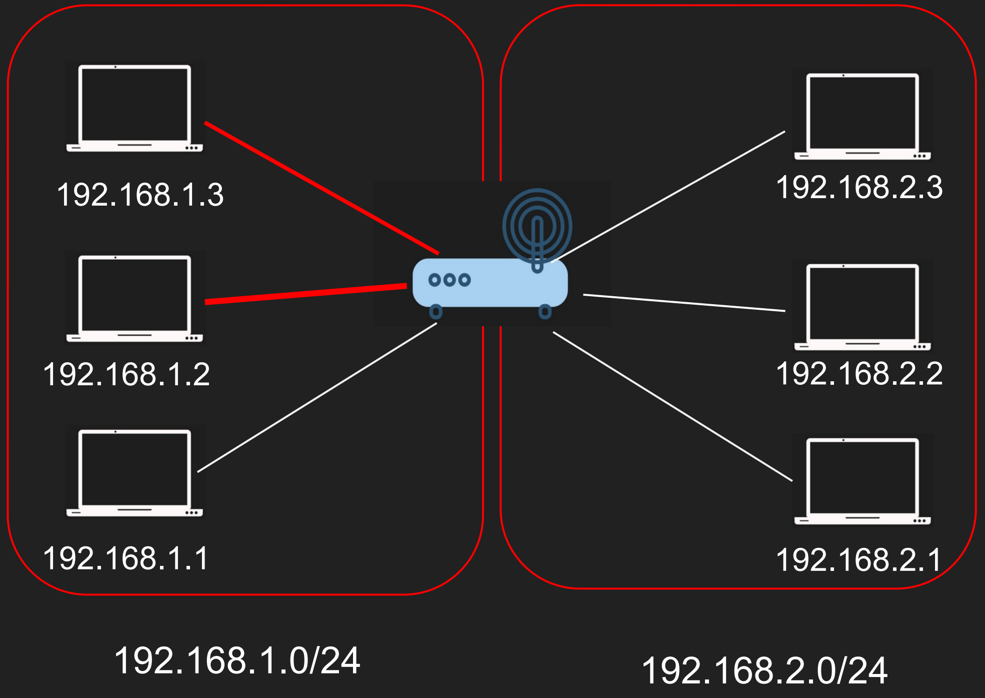
- 192.168.254.0/24 is also called a subnet
- The subnet has a mask 255.255.255.0
- Subnet mask is used to determine whether an IP is in the same subnet

Default Gateway

- Most networks consists of hosts and a Default Gateway
- Host A can talk to B directly if both are in the same subnet
- Otherwise A sends it to someone who might know, the gateway
- The Gateway has an IP Address and each host should know its gateway

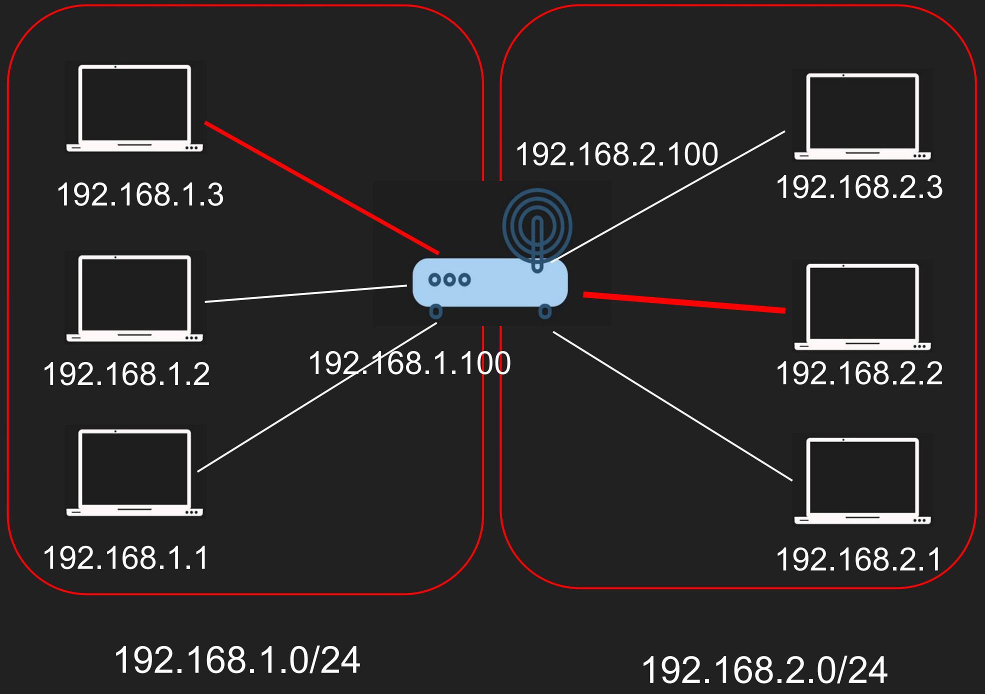
E.g. Host 192.168.1.3 wants to talk to 192.168.1.2

- 192.168.1.3 applies subnet mask to itself and the destination IP 192.168.1.2
- 255.255.255.0 & 192.168.1.3 = 192.168.1.0
- 255.255.255.0 & 192.168.1.2 = 192.168.1.0
- Same subnet ! no need to route



E.g. Host 192.168.1.3 wants to talk to 192.168.2.2

- 192.168.1.3 applies subnet mask to itself and the destination IP 192.168.2.2
- 255.255.255.0 & 192.168.1.3 = 192.168.1.0
- 255.255.255.0 & 192.168.2.2 = 192.168.2.0
- Not the subnet ! The packet is sent to the Default Gateway 192.168.1.100



Summary

- IP Address
- Network vs Host
- Subnet and subnet mask
- Default Gateway

The IP Packet

Anatomy of the IP Packet

IP Packet

- The IP Packet has headers and data sections
- IP Packet header is 20 bytes (can go up to 60 bytes if options are enabled)
- Data section can go up to 65536

IP Packet to the Backend Engineer

Actual IP Packet

<table>
<tr><th>Offsets</th><th></th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><td></td><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>0</td><td>0</td><td colspan="4"></td><td colspan="4"></td><td colspan="6"></td><td colspan="2"></td><td colspan="16"></td></tr>
<tr><td>4</td><td>32</td><td colspan="16"></td><td colspan="3"></td><td colspan="13"></td></tr>
<tr><td>8</td><td>64</td><td colspan="8"></td><td colspan="8"></td><td colspan="16"></td></tr>
<tr><td>12</td><td>96</td><td colspan="32"></td></tr>
<tr><td>16</td><td>128</td><td colspan="32"></td></tr>
<tr><td>20</td><td>160</td><td colspan="32" rowspan="3"><u>Options</u> (if IHL > 5)</td></tr>
<tr><td>⋮</td><td>⋮</td></tr>
<tr><td>56</td><td>448</td></tr>
</table>

Data

https://datatracker.ietf.org/doc/html/rfc791
https://en.wikipedia.org/wiki/IPv4

Version - The Protocol version

Offsets		0								1								2								3							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0																																
4	32																																
8	64																																
12	96																																
16	128																																
20	160	Options (if IHL > 5)																															
⋮	⋮																																
56	448																																

Data

Internet Header Length - Defines the Options length

Offsets		0								1								2								3							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0																																
4	32																																
8	64																																
12	96																																
16	128																																
20	160	Options (if IHL > 5)																															
⋮	⋮																																
56	448																																

Data

Total Length - 16 bit Data + header

Offsets		0								1								2								3							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0																																
4	32																																
8	64																																
12	96																																
16	128																																
20	160	Options (if IHL > 5)																															
⋮	⋮																																
56	448																																

Data

Fragmentation - Jumbo packets

Offsets		0								1								2								3							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0																																
4	32																																
8	64																																
12	96																																
16	128																																
20	160	Options (if IHL > 5)																															
⋮	⋮																																
56	448																																

Data

Time To Live - How many hops can this packet survive?

Offsets		0								1								2								3							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0																																
4	32																																
8	64																																
12	96																																
16	128																																
20	160	Options (if IHL > 5)																															
⋮	⋮																																
56	448																																

Data

Protocol - What protocol is inside the data section?

Offsets		0								1								2								3							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0																																
4	32																																
8	64																																
12	96																																
16	128																																
20	160	Options (if IHL > 5)																															
⋮	⋮																																
56	448																																

Data

husseinnasser

Source and Destination IP

Offsets		0								1								2								3							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0																																
4	32																																
8	64																																
12	96																																
16	128																																
20	160	Options (if IHL > 5)																															
⋮	⋮																																
56	448																																

Data

Explicit Congestion Notification

Offsets		0								1								2								3							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0																																
4	32																																
8	64																																
12	96																																
16	128																																
20	160	Options (if IHL > 5)																															
⋮	⋮																																
56	448																																

Data

Summary

- The IP Packet has headers and data sections
- IP Packet header is 20 bytes (can go up to 60 bytes if options are enabled)
- Data section can go up to 65536
- Packets need to get fragmented if it doesn’t fit in a frame

ICMP

Internet Control Message Protocol

ICMP

- Stands for Internet Control Message Protocol
- Designed for informational messages
    - Host unreachable, port unreachable, fragmentation needed
    - Packet expired (infinite loop in routers)
- Uses IP directly
- PING and traceroute use it
- Doesn’t require listeners or ports to be opened

ICMP header

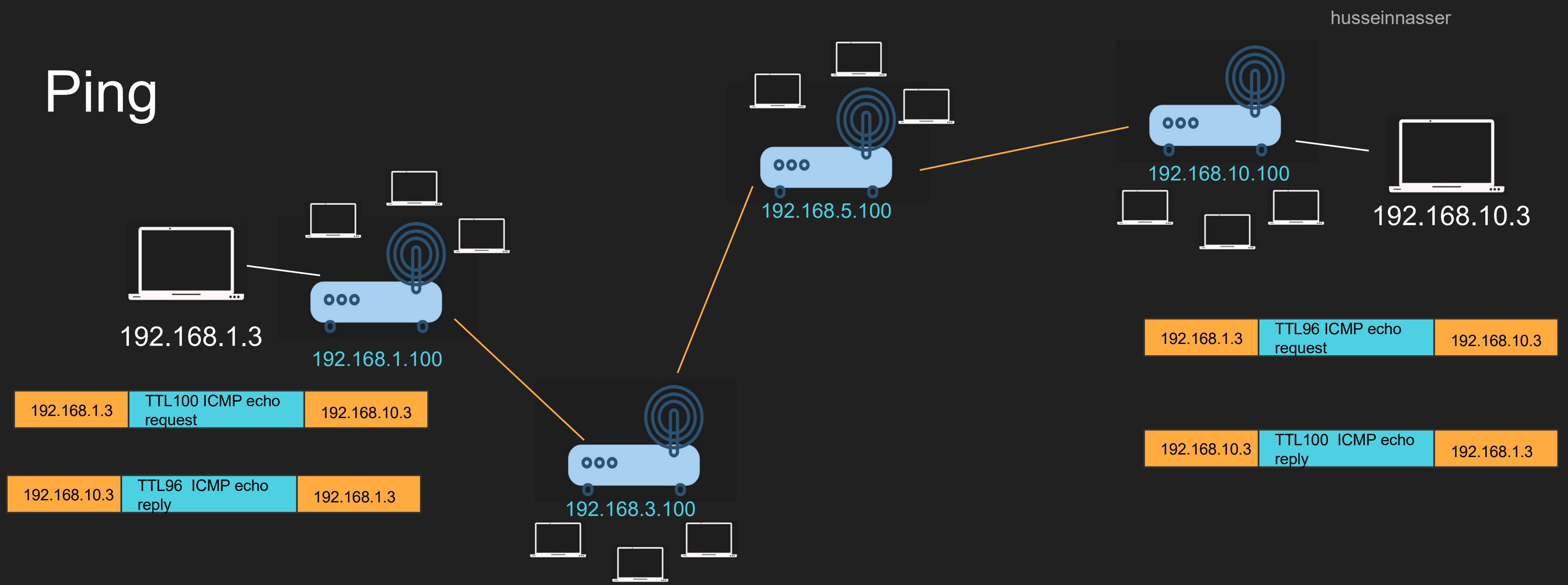
Offsets		0								1								2								3							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0																																
4	32																																

https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol
https://datatracker.ietf.org/doc/html/rfc792

ICMP

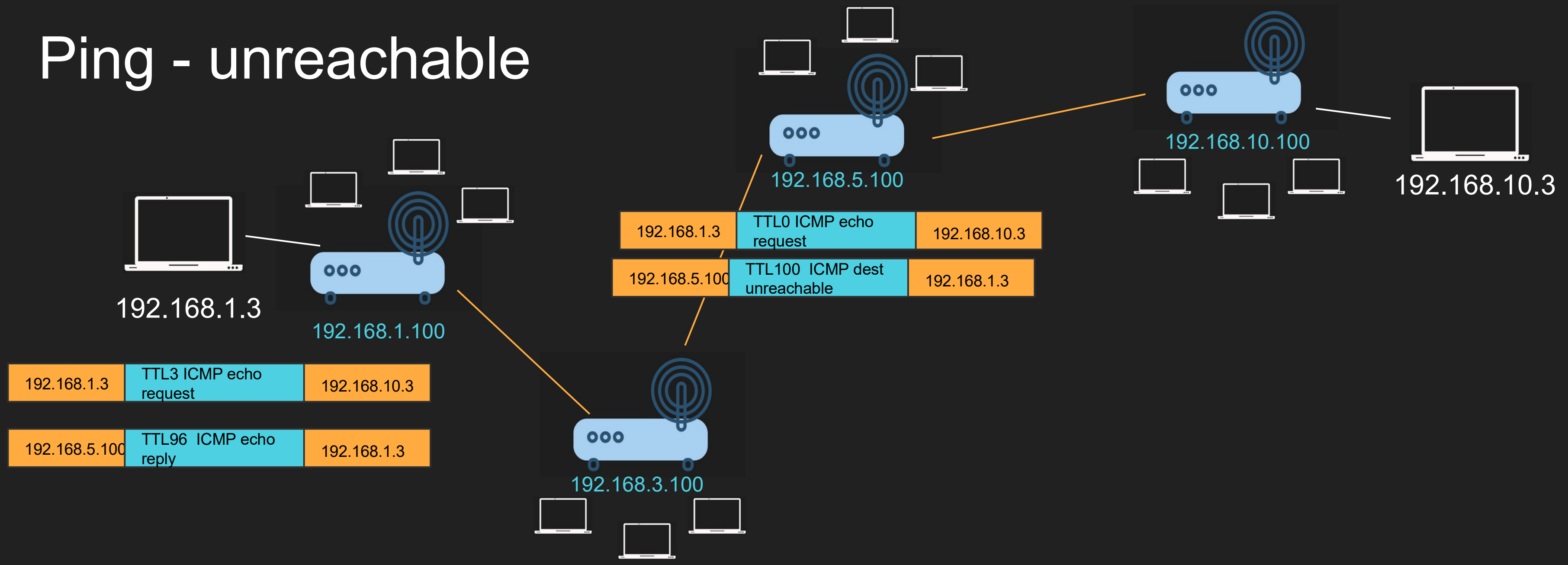
- Some firewalls block ICMP for security reasons
- That is why PING might not work in those cases
- Disabling ICMP also can cause real damage with connection establishment
  - Fragmentation needed
- PING demo

Ping


husseinnasser
192.168.1.3
192.168.1.100
192.168.3.100
192.168.5.100
192.168.10.100
192.168.10.3
192.168.1.3
TTL100 ICMP echo request
192.168.10.3
192.168.10.3
TTL96 ICMP echo reply
192.168.1.3
192.168.1.3
TTL96 ICMP echo request
192.168.10.3
192.168.10.3
TTL100 ICMP echo reply
192.168.1.3

husseinnasser

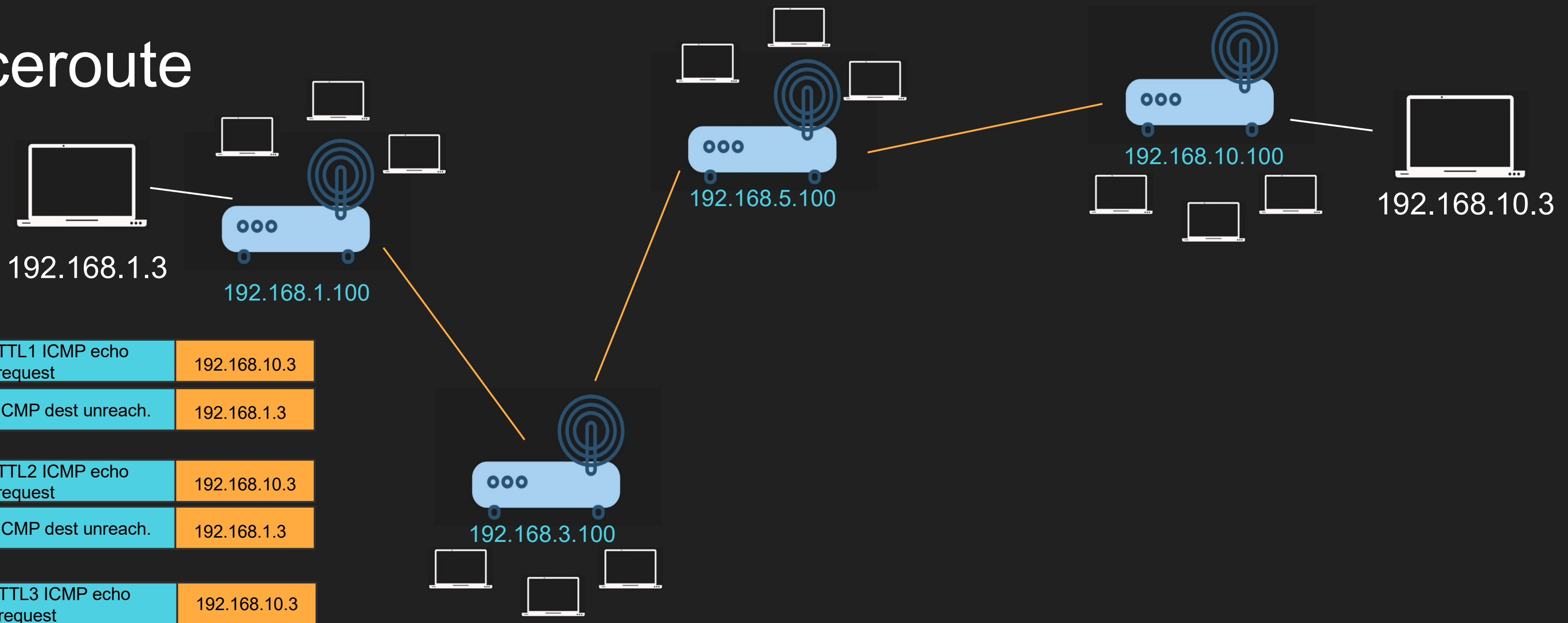
Ping - unreachable



TraceRoute

- Can you identify the entire path your IP Packet takes?
- Clever use of TTL
- Increment TTL slowly and you will get the router IP address for each hop
- Doesn’t always work as path changes and ICMP might be blocked

Traceroute



192.168.1.3	TTL1 ICMP echo request	192.168.10.3
192.168.1.100	ICMP dest unreach.	192.168.1.3
192.168.1.3	TTL2 ICMP echo request	192.168.10.3
192.168.3.100	ICMP dest unreach.	192.168.1.3
192.168.1.3	TTL3 ICMP echo request	192.168.10.3
192.168.5.100	ICMP dest unreach.	192.168.1.3
192.168.1.3	TTL4 ICMP echo request	192.168.10.3
192.168.10.100	ICMP dest unreach	192.168.1.3
192.168.1.3	TTL5 ICMP echo request	192.168.10.3
192.168.10.3	ICMP Echo reply	192.168.1.3

Summary

- ICMP is an IP level protocol used for information messages
- Critical to know if the host is available or port is opened
- Used for PING and TraceRoute
- Can be blocked which can cause problems

ARP

Address Resolution Protocol

Why ARP?

- We need the MAC address to send frames (layer 2)
- Most of the time we know the IP address but not the MAC
- ARP Table is cached IP->Mac mapping

Network Frame

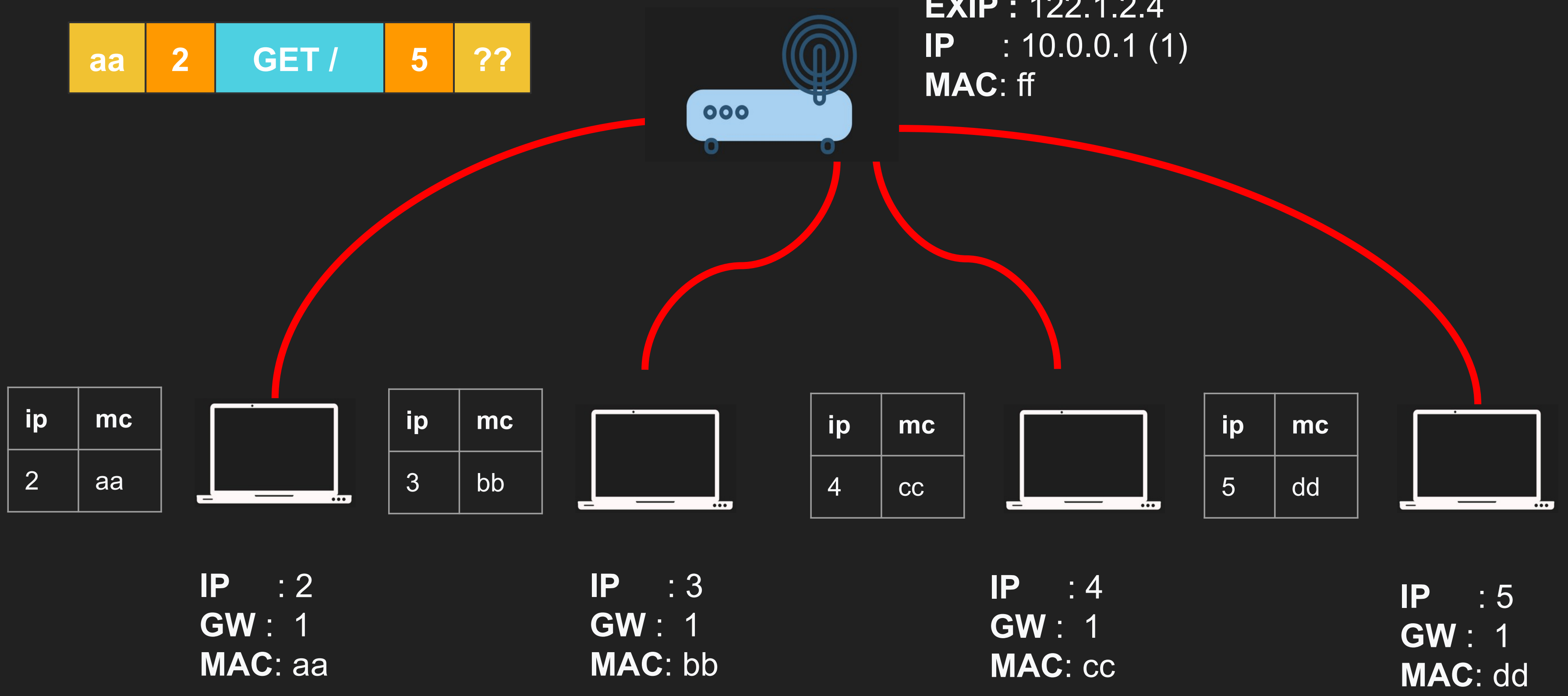
aa:bc:32:7f:c0:07	10.0.0.2	2312	GET /	8080	10.0.0.3	bb:ab:dd:11:22:33

aa	2	GET /	3	bb

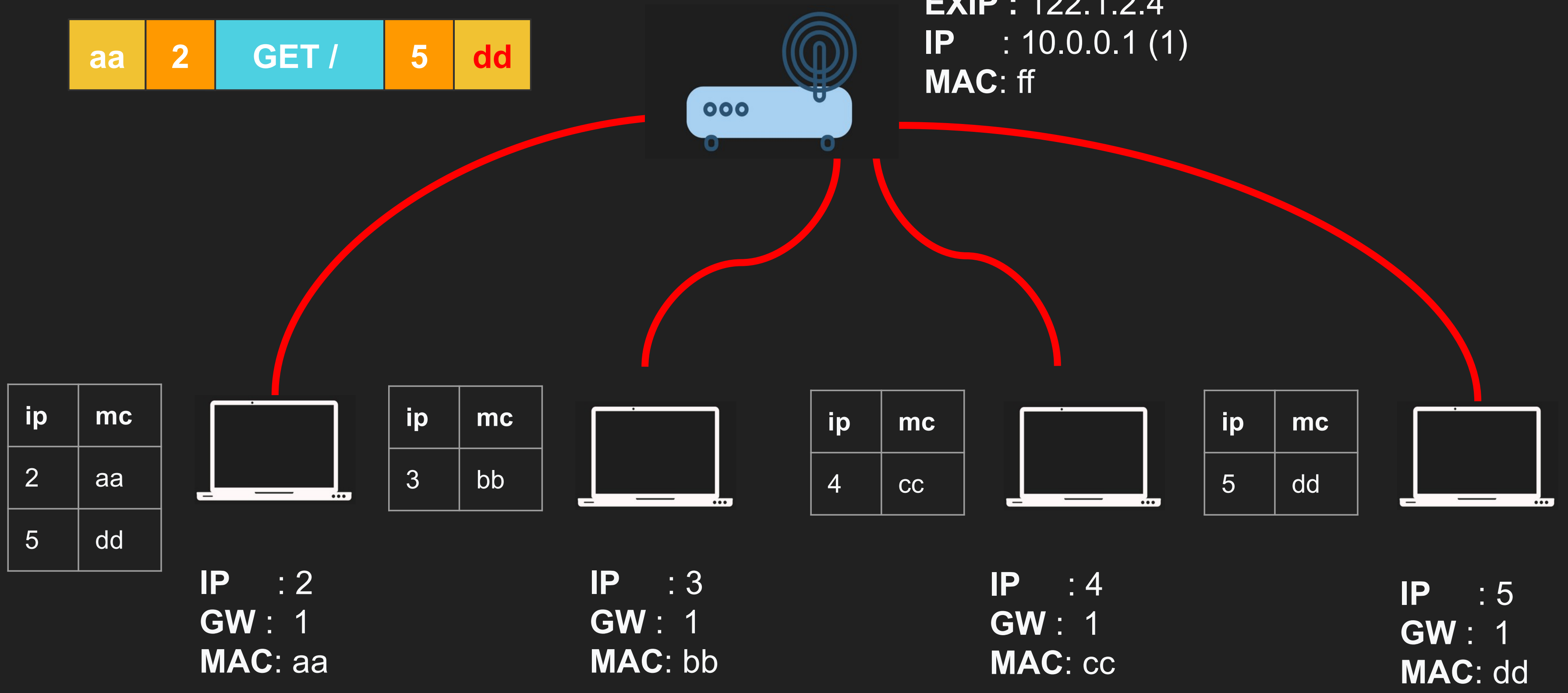
IP : 10.0.0.2
MAC: aa:bc:32:7f:c0:07

IP : 10.0.0.3
MAC: bb:ab:dd:11:22:33
Port: 8080

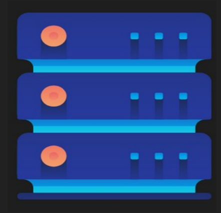
- IP 10.0.0.2 (**2**) wants to connect to IP 10.0.0.5 (**5**)
- Host 2 checks if host 5 is within its subnet, it is.
- Host **2** needs the MAC address of host **5**
- Host 2 checks its ARP tables and its not there

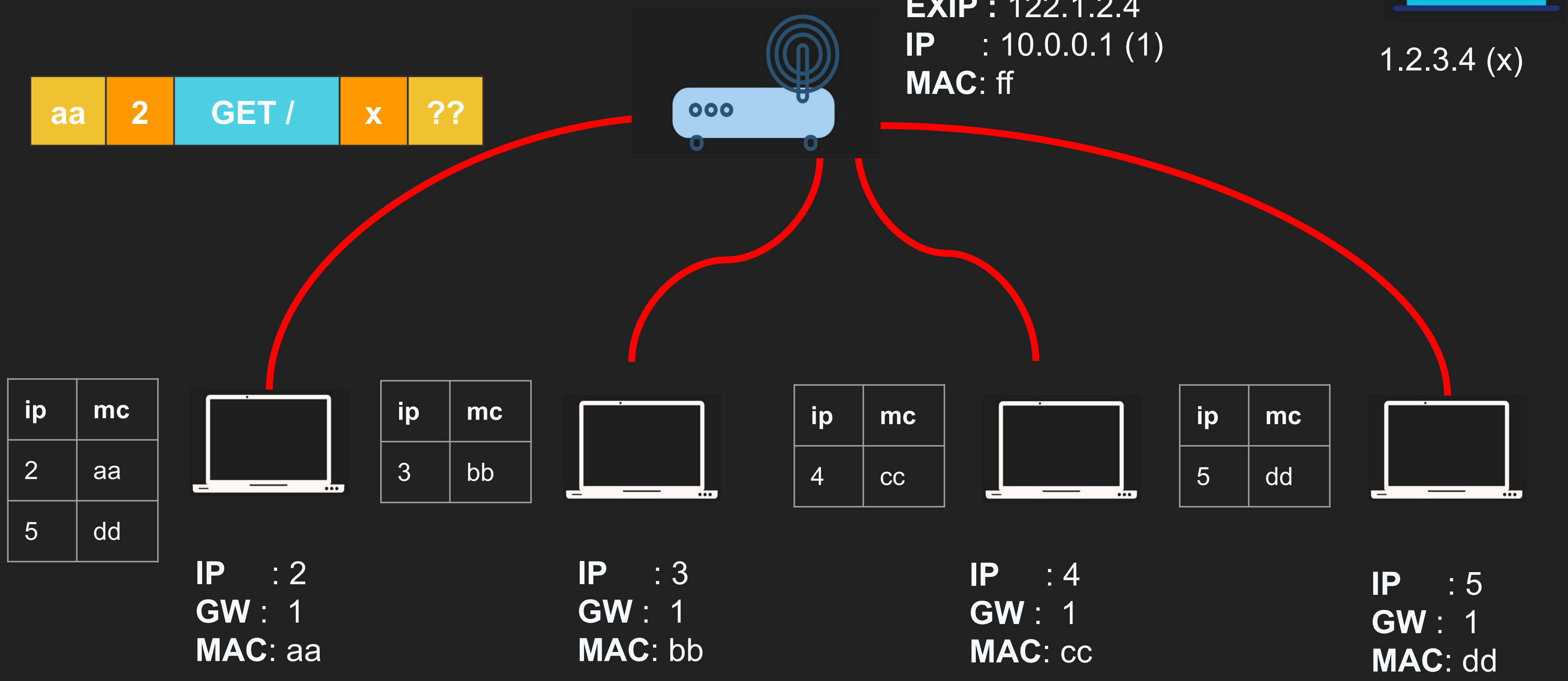


- Host 2 sends an ARP request broadcast to all machines in its network
- Who has IP address 10.0.0.5?
- Host 5 replies with dd
- Host 2 updates its ARP Table

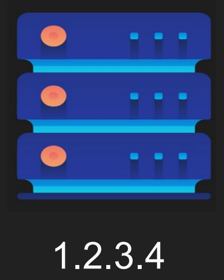


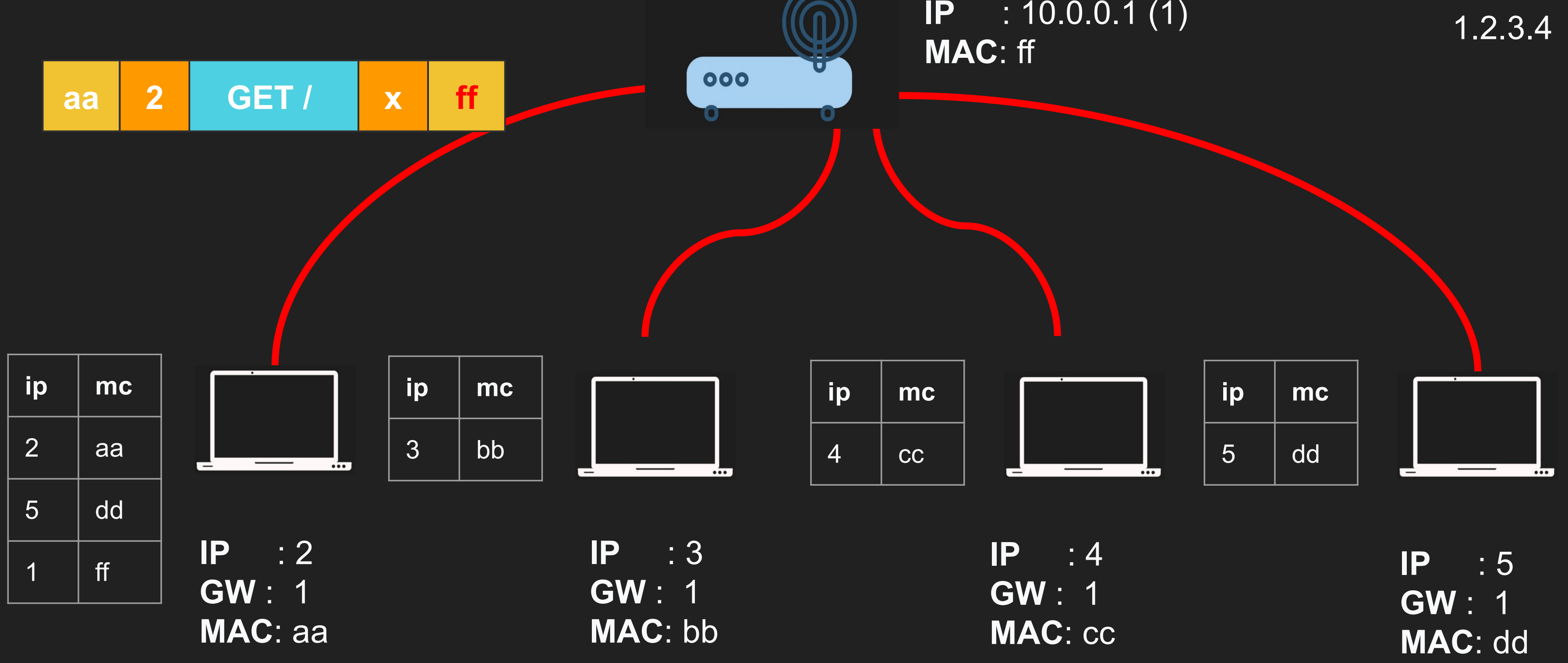
- IP 10.0.0.2 (**2**) wants to connect to IP 1.2.3.4 (x)
- Host 2 checks if 1.2.3.4 is within its subnet, it is NOT!
- Host 2 needs to talk to its gatway
- Host **2** needs the MAC address of the gateway





- Host 2 checks its local ARP table, 10.0.0.1 is not it in
- Host 2 sends an ARP request to everybody in the network
- Who has 10.0.0.1? (A DANGEROUS QUESTION)
- Gateway reply with ff
- NAT than kicks in.



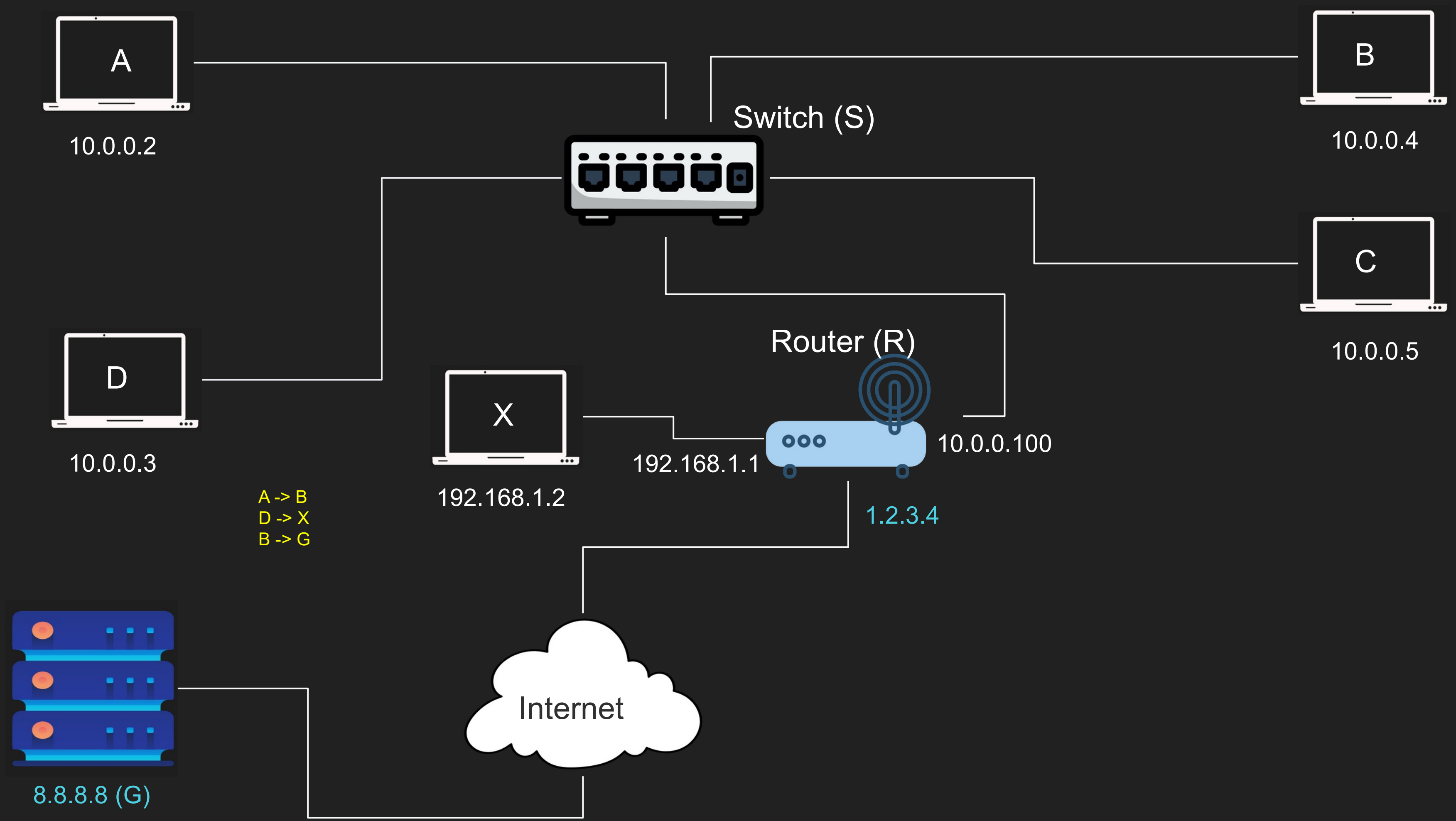


Summary

- ARP stands for Address resolution protocol
- We need MAC address to send frames between machines
- Almost always we have the IP address but not the MAC
- Need a lookup protocol that give us the MAC from IP address
- Attacks can be performed on ARP (ARP poisoning)

Routing Example

How IP Packets are routed in Switches and Routers


A
10.0.0.2
B
10.0.0.4
Switch (S)
C
10.0.0.5
D
10.0.0.3
X
192.168.1.2
Router (R)
192.168.1.1
10.0.0.100
1.2.3.4
A -> B
D -> X
B -> G
Internet
8.8.8.8 (G)

UDP

User Datagram Protocol

UDP

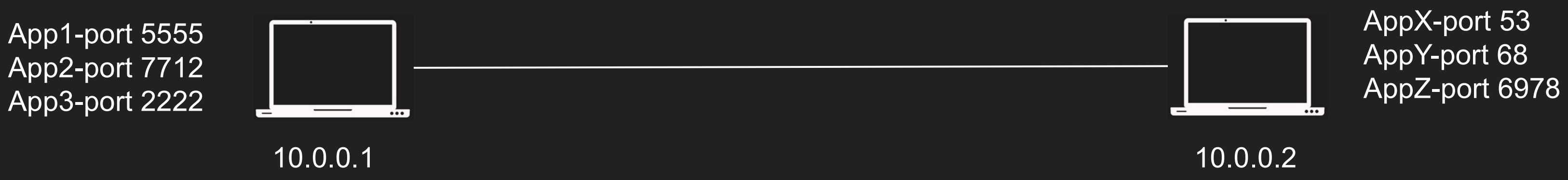
- Stands for User Datagram Protocol
- Layer 4 protocol
- Ability to address processes in a host using ports
- Simple protocol to send and receive data
- Prior communication not required (double edge sword)
- Stateless no knowledge is stored on the host
- 8 byte header Datagram

UDP Use cases

- Video streaming
- VPN
- DNS
- WebRTC

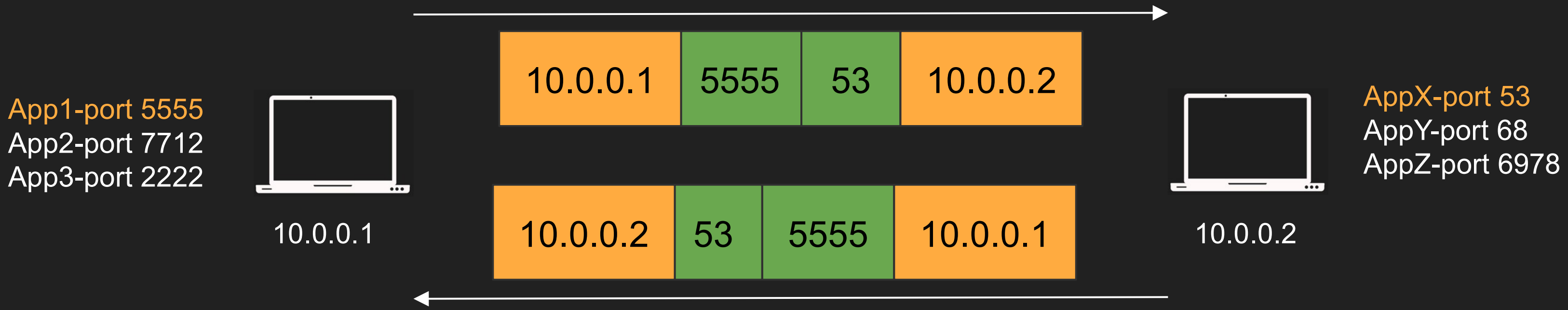
Multiplexing and demultiplexing

- IP target hosts only
- Hosts run many apps each with different requirements
- Ports now identify the “app” or “process”
- Sender multiplexes all its apps into UDP
- Receiver demultiplex UDP datagrams to each app



Source and Destination Port

- App1 on 10.0.0.1 sends data to AppX on 10.0.0.2
- Destination Port = 53
- AppX responds back to App1
- We need Source Port so we know how to send back data
- Source Port = 5555



Summary

- UDP is a simple layer 4 protocol
- Uses ports to address processes
- Stateless

UDP Datagram

The anatomy of the UDP datagram

UDP Datagram

- UDP Header is 8 bytes only (IPv4)
- Datagram slides into an IP packet as “data”
- Port are 16 bit (0 to 65535)

husseinnasser

UDP Datagram header

Offsets		**0**								**1**								**2**								**3**							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															
		Data																															

https://www.ietf.org/rfc/rfc768.txt
https://en.wikipedia.org/wiki/User_Datagram_Protocol

Source Port and Destination Port

<table>
<tr><th>Offsets</th><th></th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><td></td><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></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="16">Length</td><td colspan="16">Checksum</td></tr>
</table>

Data

Length & Checksum

<table>
<tr><th>Offsets</th><th></th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><td></td><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></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="16">Length</td><td colspan="16">Checksum</td></tr>
</table>

Data

UDP Pros and Cons

The power and drawbacks of UDP

UDP Pros

- Simple protocol
- Header size is small so datagrams are small
- Uses less bandwidth
- Stateless
- Consumes less memory (no state stored in the server/client)
- Low latency - no handshake , order, retransmission or guaranteed delivery

UDP Cons

- No acknowledgement
- No guarantee delivery
- Connection-less - anyone can send data without prior knowledge
- No flow control
- No congestion control
- No ordered packets
- Security - can be easily spoofed

TCP

Transmission Control Protocol

TCP

- Stands for Transmission Control Protocol
- Layer 4 protocol
- Ability to address processes in a host using ports
- “Controls” the transmission unlike UDP which is a firehose
- Connection
- Requires handshake
- 20 bytes headers Segment (can go to 60)
- Stateful

TCP Use cases

- Reliable communication
- Remote shell
- Database connections
- Web communications
- Any bidirectional communication

TCP Connection

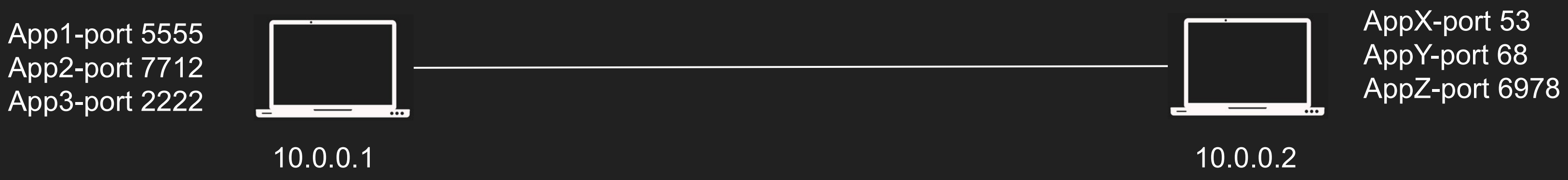
- Connection is a Layer 5 (session)
- Connection is an agreement between client and server
- Must create a connection to send data
- Connection is identified by 4 properties
    - SourceIP-SourcePort
    - DestinationIP-DestinationPort

TCP Connection

- Can’t send data outside of a connection
- Sometimes called socket or file descriptor
- Requires a 3-way TCP handshake
- Segments are sequenced and ordered
- Segments are acknowledged
- Lost segments are retransmitted

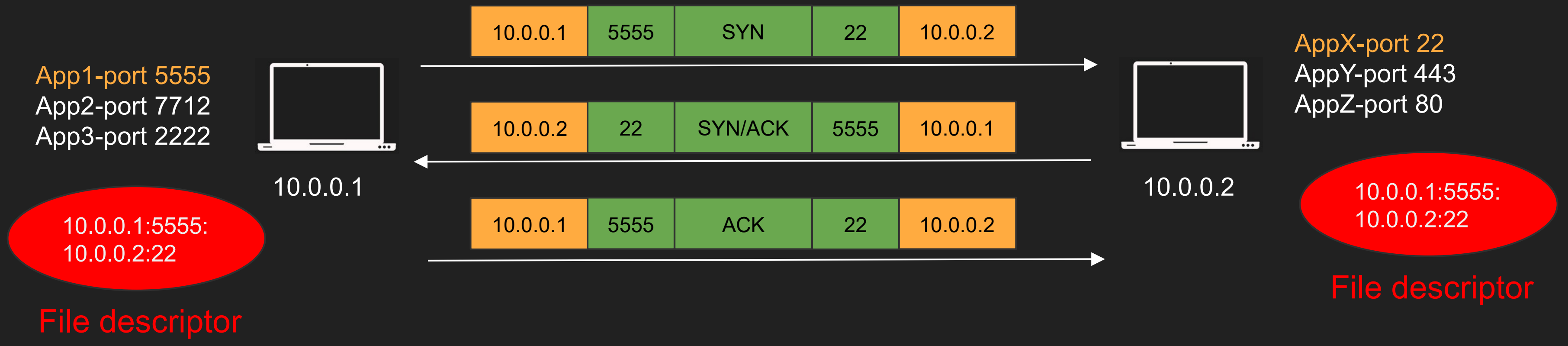
Multiplexing and demultiplexing

- IP target hosts only
- Hosts run many apps each with different requirements
- Ports now identify the "app" or "process"
- Sender multiplexes all its apps into TCP connections
- Receiver demultiplex TCP segments to each app based on connection pairs



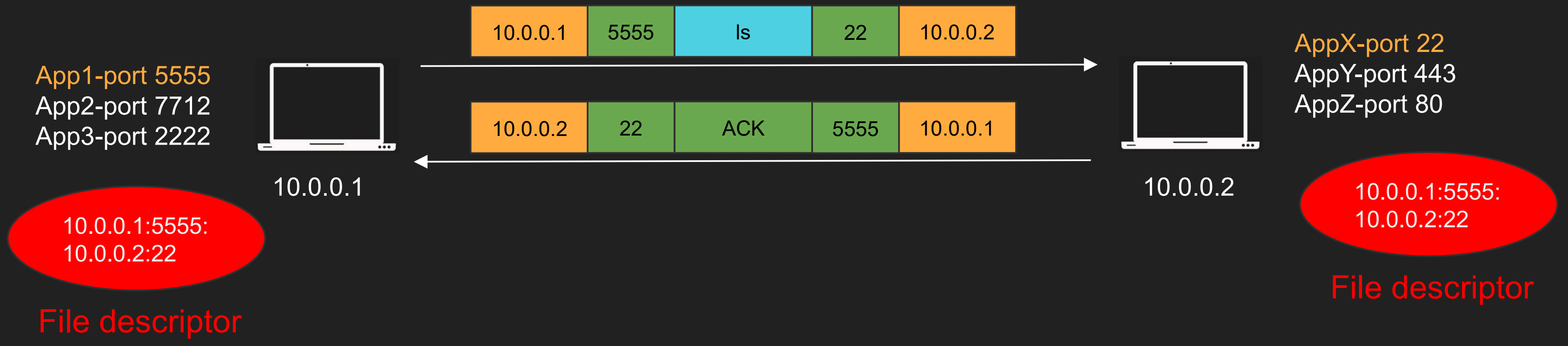
Connection Establishment

- App1 on 10.0.0.1 want to send data to AppX on 10.0.0.2
- App1 sends SYN to AppX to synchronous sequence numbers
- AppX sends SYN/ACK to synchronous its sequence number
- App1 ACKs AppX SYN.
- Three way handshake



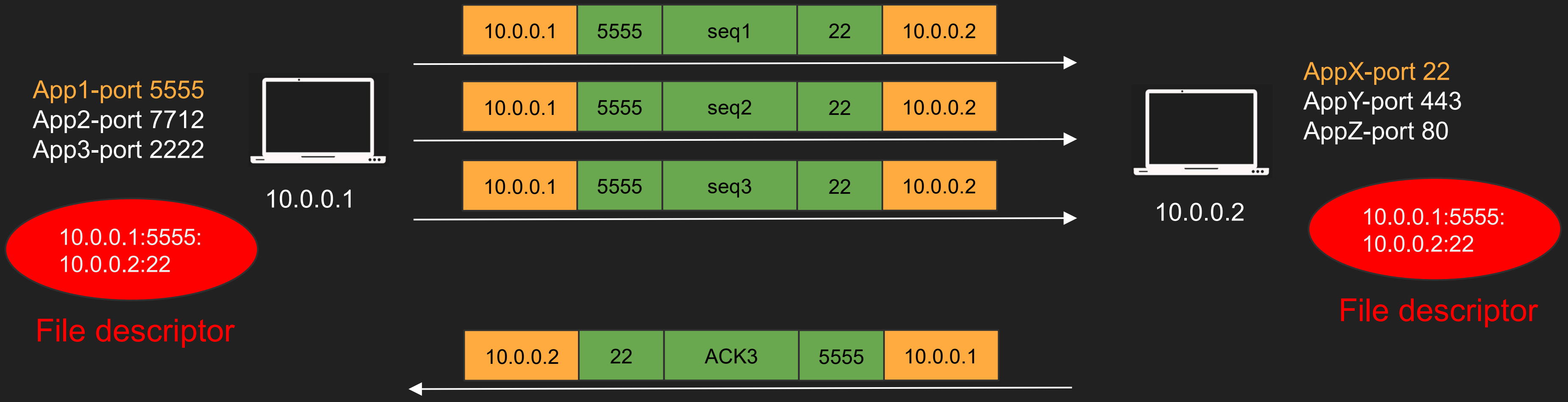
Sending data

- App1 sends data to AppX
- App1 encapsulate the data in a segment and send it
- AppX acknowledges the segment
- Hint: Can App1 send new segment before ack of old segment arrives?



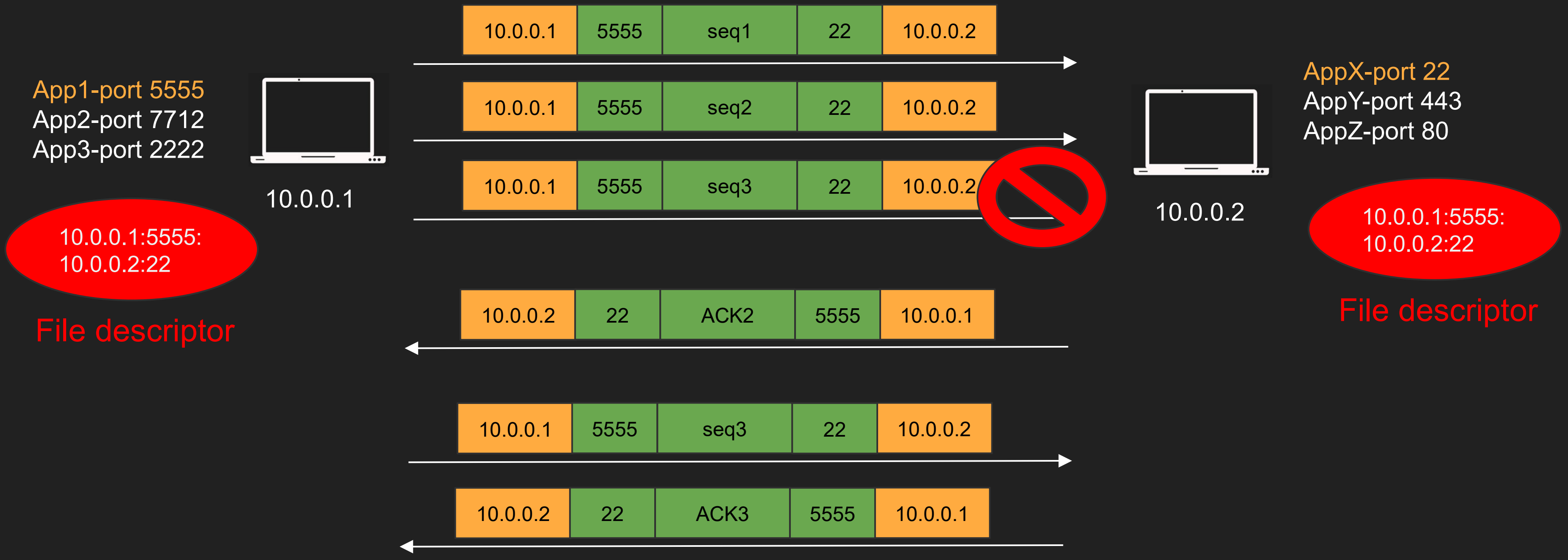
Acknowledgment

- App1 sends segment 1,2 and 3 to AppX
- AppX acknowledge all of them with a single ACK 3



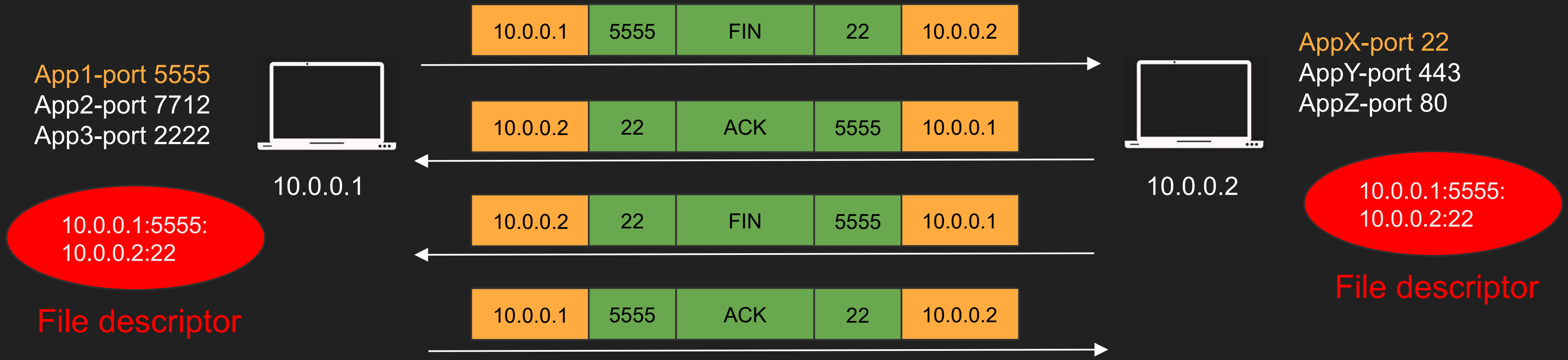
Lost data

- App1 sends segment 1,2 and 3 to AppX
- Seg 3 is lost, AppX acknowledge 3
- App1 resend Seq 3



Closing Connection

- App1 wants to close the connection
- App1 sends FIN, AppX ACK
- AppX sends FIN, App1 ACK
- Four way handshake



Summary

- Stands for Transmission Control Protocol
- Layer 4 protocol
- “Controls” the transmission unlike UDP which is a firehose
- Introduces Connection concept
- Retransmission, acknowledgement, guaranteed delivery
- Stateful, connection has a state

TCP Segment

The anatomy of the TCP Segment

TCP Segment

- TCP segment Header is 20 bytes and can go up to 60 bytes
- TCP segments slides into an IP packet as “data”
- Port are 16 bit (0 to 65535)
- Sequences, Acknowledgment, flow control and more

TCP Segment

<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></th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</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="3">Reserved
0 0 0</td><td>NS</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>60</td><td>480</td></tr>
</table>

https://en.wikipedia.org/wiki/Transmission_Control_Protocol
https://datatracker.ietf.org/doc/html/rfc793

Ports

<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></th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</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="3">Reserved
0 0 0</td><td>NS</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>60</td><td>480</td></tr>
</table>

Sequences and ACKs

<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></th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</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="3">Reserved
0 0 0</td><td>NS</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>60</td><td>480</td></tr>
</table>

Flow Control Window Size

<table>
<tr><td colspan="2">Offsets</td><td colspan="8">0</td><td colspan="8">1</td><td colspan="8">2</td><td colspan="8">3</td></tr>
<tr><td>Octet</td><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></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="3">Reserved
0 0 0</td><td>N
S</td><td>C
W
R</td><td>E
C
E</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>FI
N</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>60</td><td>480</td></tr>
</table>

9 bit flags

Offsets		0								1								2								3							
Octet		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																Urgent pointer (if URG set)															
20	160	Options (if *data offset* > 5. Padded at the end with "0" bits if necessary.)																															
⋮	⋮																																
60	480																																

Maximum Segment Size

- Segment Size depends the MTU of the network
- Usually 512 bytes can go up to 1460
- Default MTU in the Internet is 1500 (results in MSS 1460)
- Jumbo frames MTU goes to 9000 or more
- MSS can be larger in jumbo frames cases

TCP Pros and Cons

The power and drawbacks of TCP

TCP Pros

- Guarantee delivery
- No one can send data without prior knowledge
- Flow Control and Congestion Control
- Ordered Packets no corruption or app level work
- Secure and can’t be easily spoofed

TCP Cons

- Large header overhead compared to UDP
- More bandwidth
- Stateful - consumes memory on server and client
- Considered high latency for certain workloads (Slow start/ congestion/ acks)
- Does too much at a low level (hence QUIC)
    - Single connection to send multiple streams of data (HTTP requests)
    - Stream 1 has nothing to do with Stream 2
    - Both Stream 1 and Stream 2 packets must arrive
- TCP Meltdown
    - Not a good candidate for VPN

Overview of Popular Networking Protocols

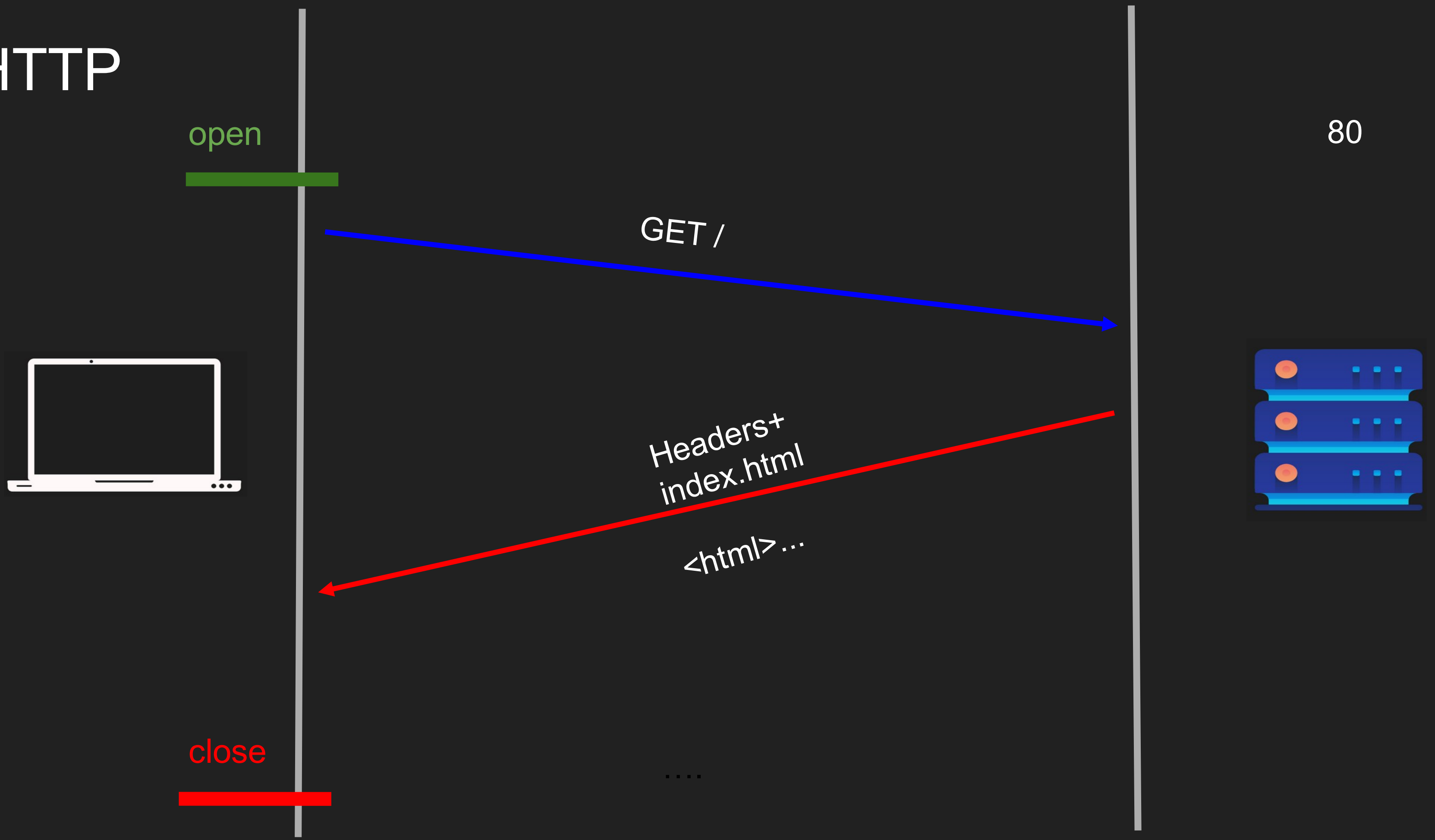
TLS

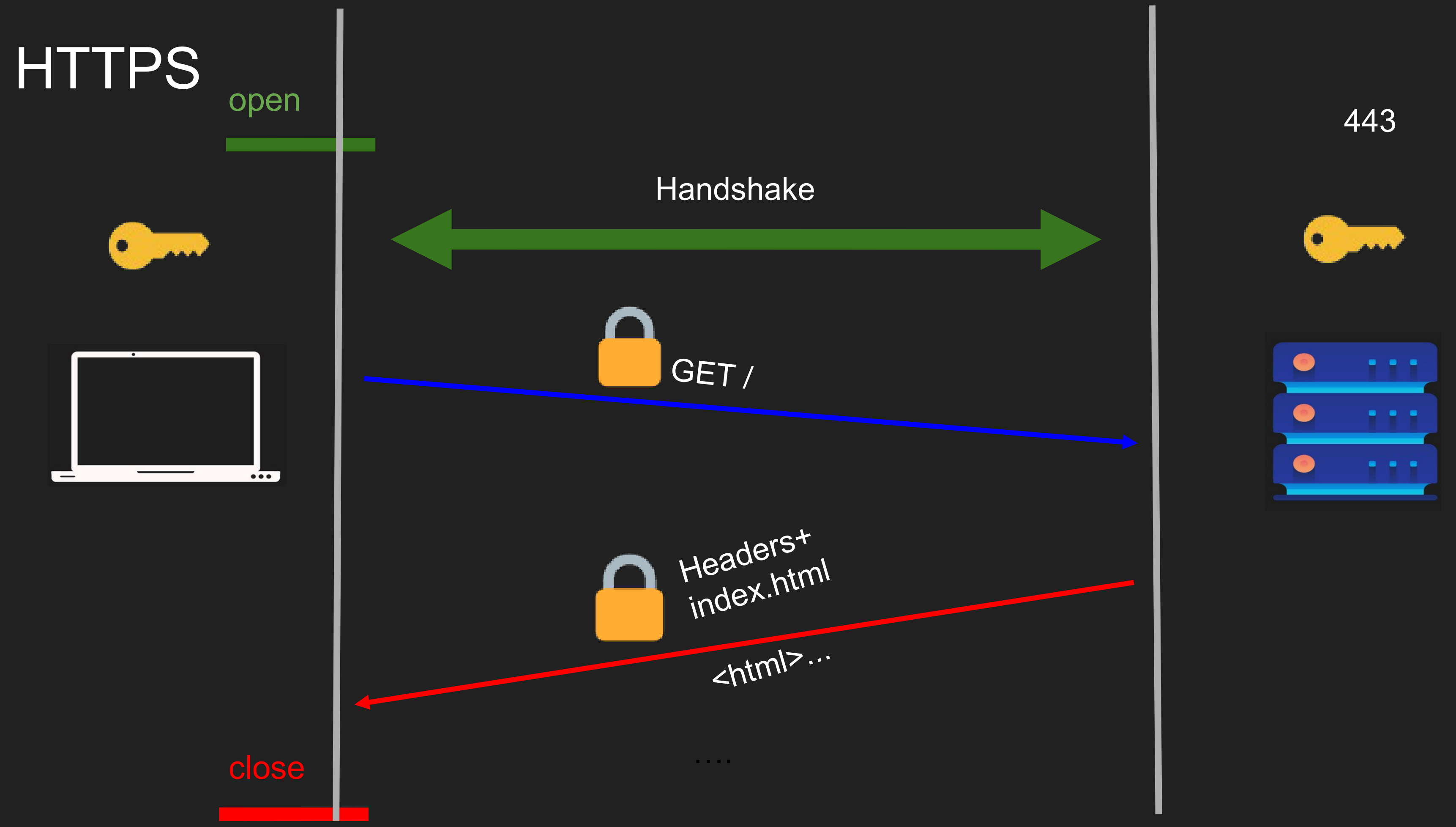
Transport Layer Security

TLS

- Vanilla HTTP
- HTTPS
- TLS 1.2 Handshake
- Diffie Hellman
- TLS 1.3 Improvements

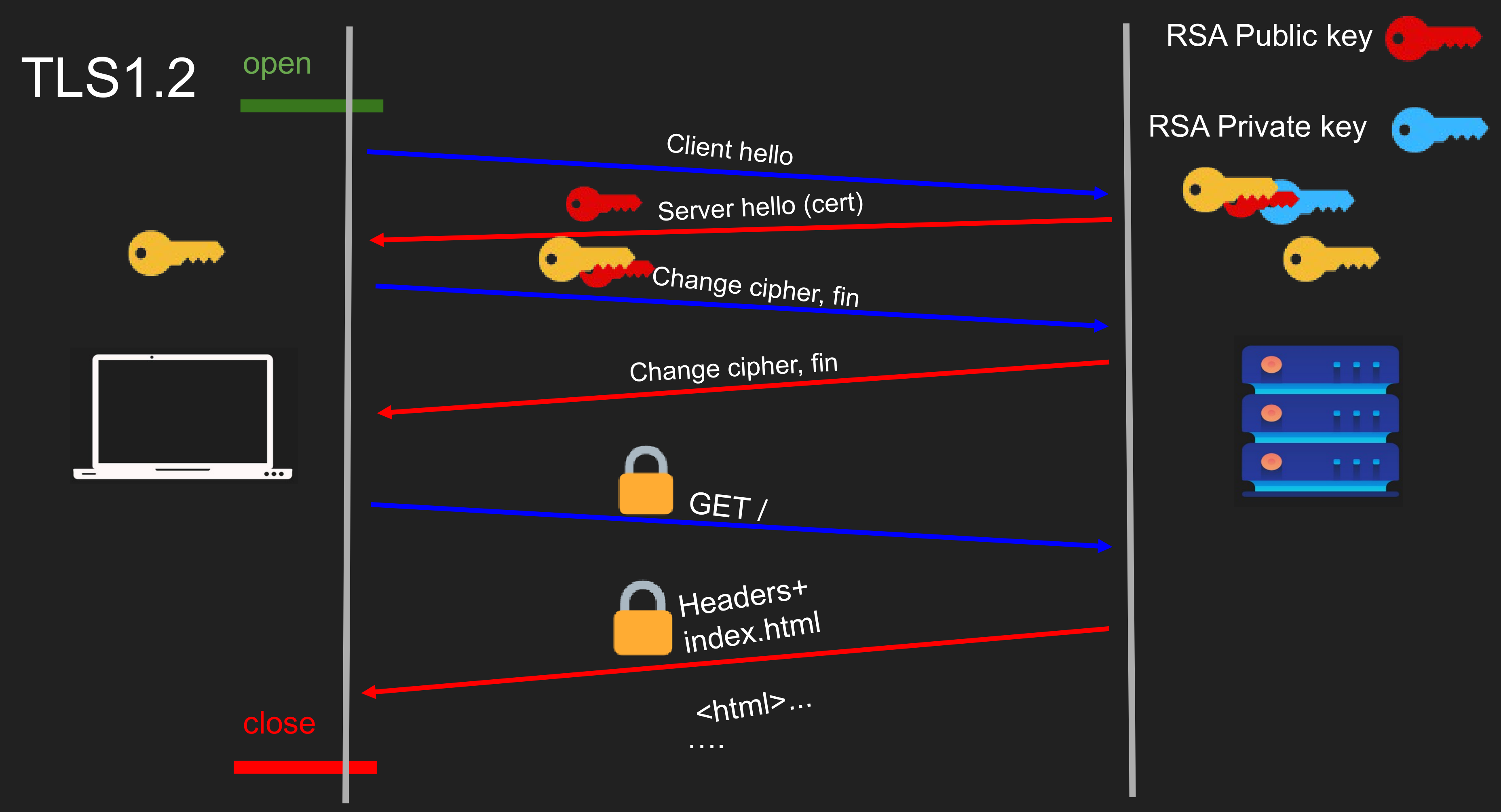
HTTP




HTTPS
open
443
Handshake
GET /
Headers+
index.html
<html>...
....
close

Why TLS

- We encrypt with symmetric key algorithms
- We need to exchange the symmetric key
- Key exchange uses asymmetric key (PKI)
- Authenticate the server
- Extensions (SNI, preshared, 0RTT)



TLS1.2
open
RSA Public key
RSA Private key
Client hello
Server hello (cert)
Change cipher, fin
Change cipher, fin
GET /
Headers+
index.html
<html>...
....
close

Diffie Hellman

Private x

+

Public g,n

+

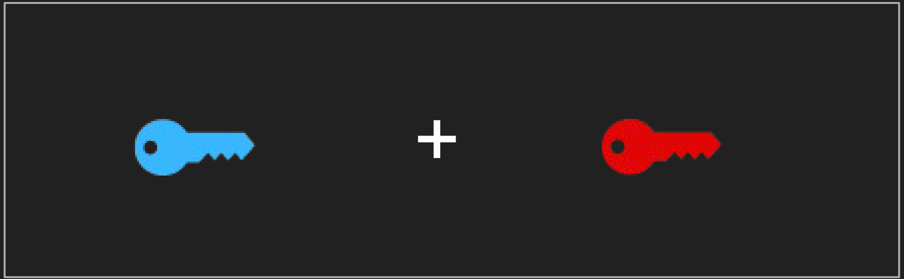
Private y

=

Symmetric key

Diffie Hellman

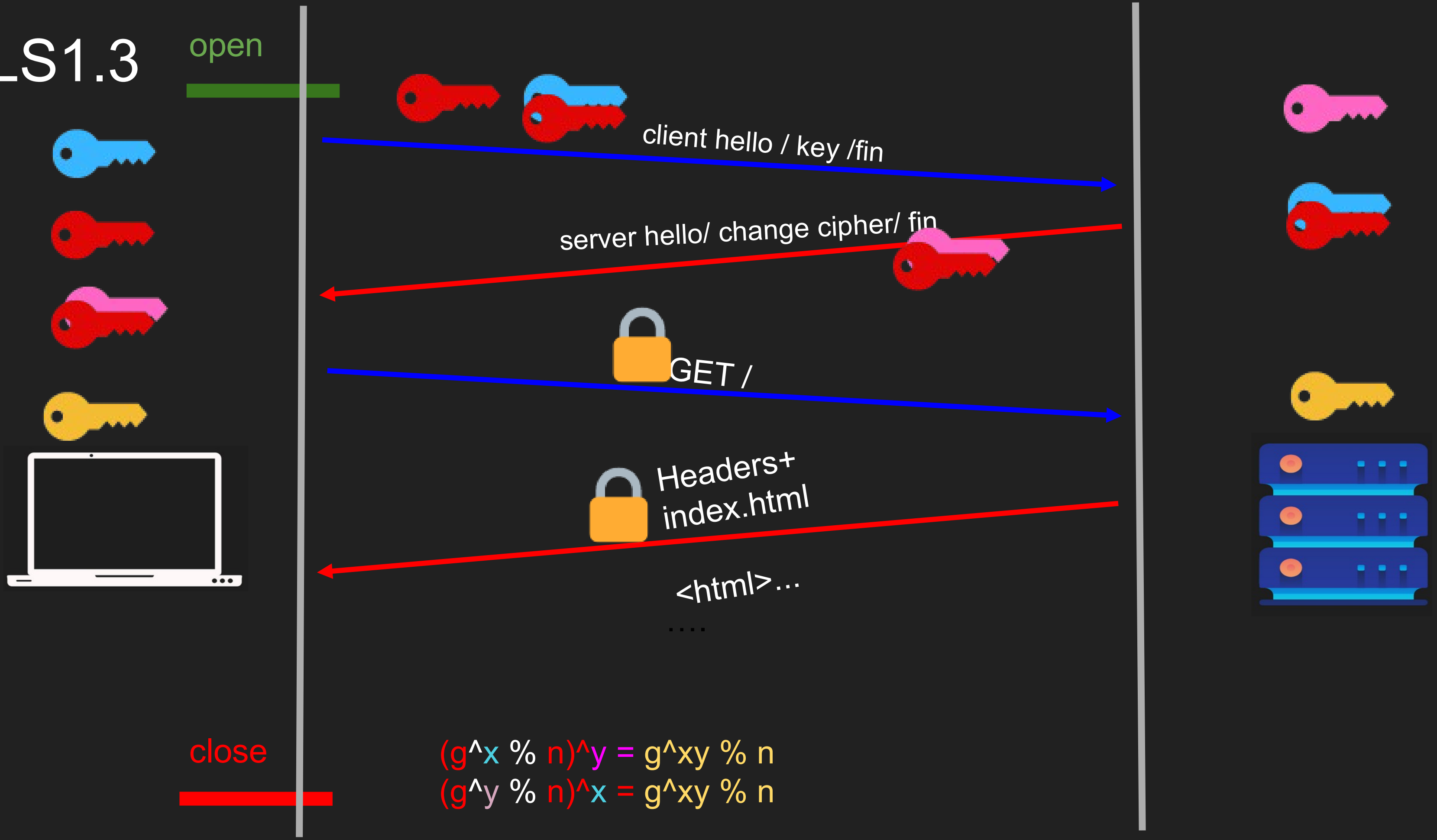
Public/
Unbreakable
/can be
shared
$g^x \% n$



Public/
Unbreakable
/can be shared
$g^y \% n$

$(g^x \% n)^y = g^{xy} \% n$
$(g^y \% n)^x = g^{xy} \% n$

TLS1.3



$$(g^x \% n)^y = g^{xy} \% n$$
$$(g^y \% n)^x = g^{xy} \% n$$

TLS Summary

- Vanilla HTTP
- HTTPS
- TLS 1.2 Handshake (two round trips)
- Diffie Hellman
- TLS 1.3 Improvements (one round trip can be zero)