Lecture 18:
Network Security

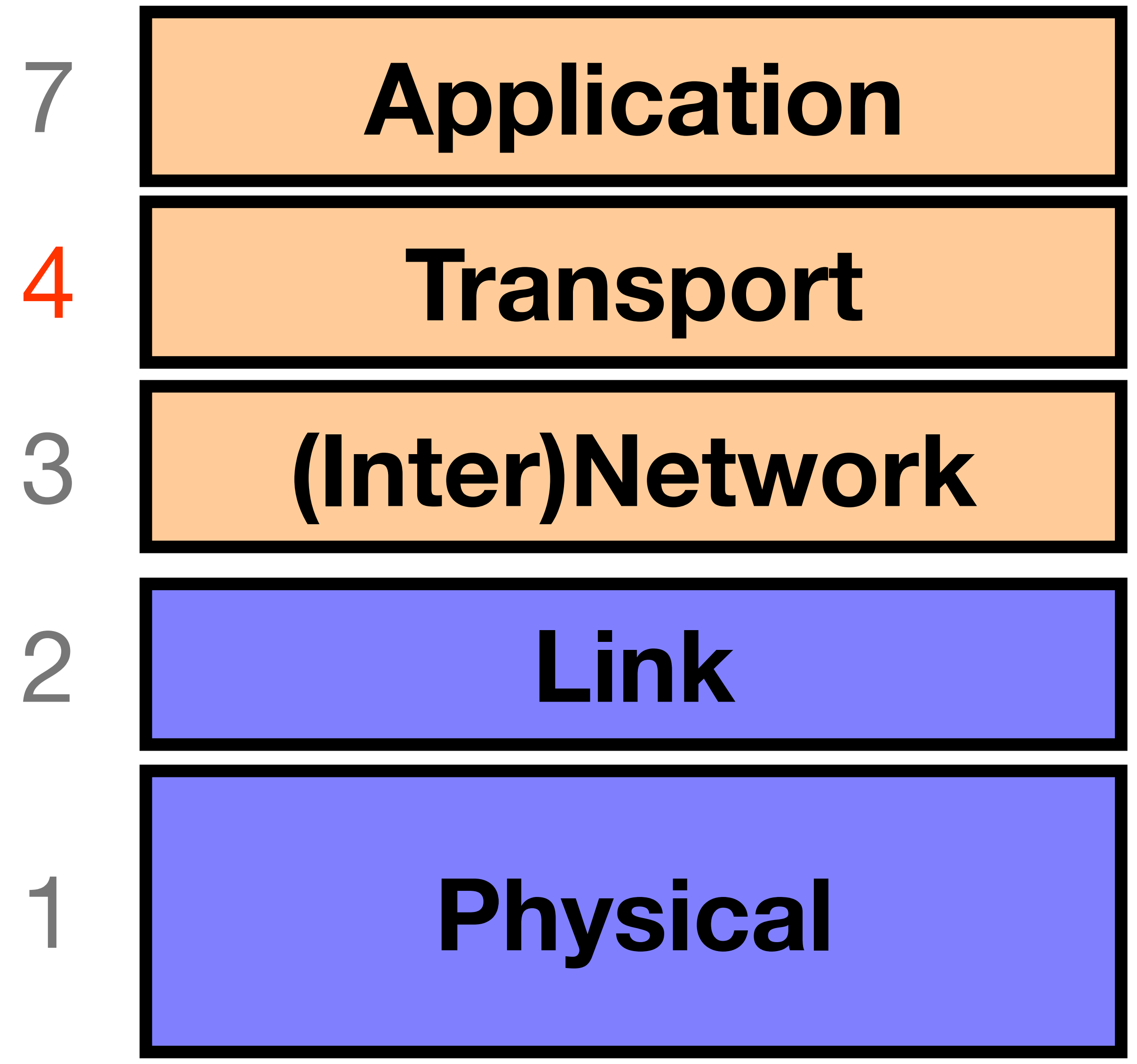
https://cs161.org

Announcements

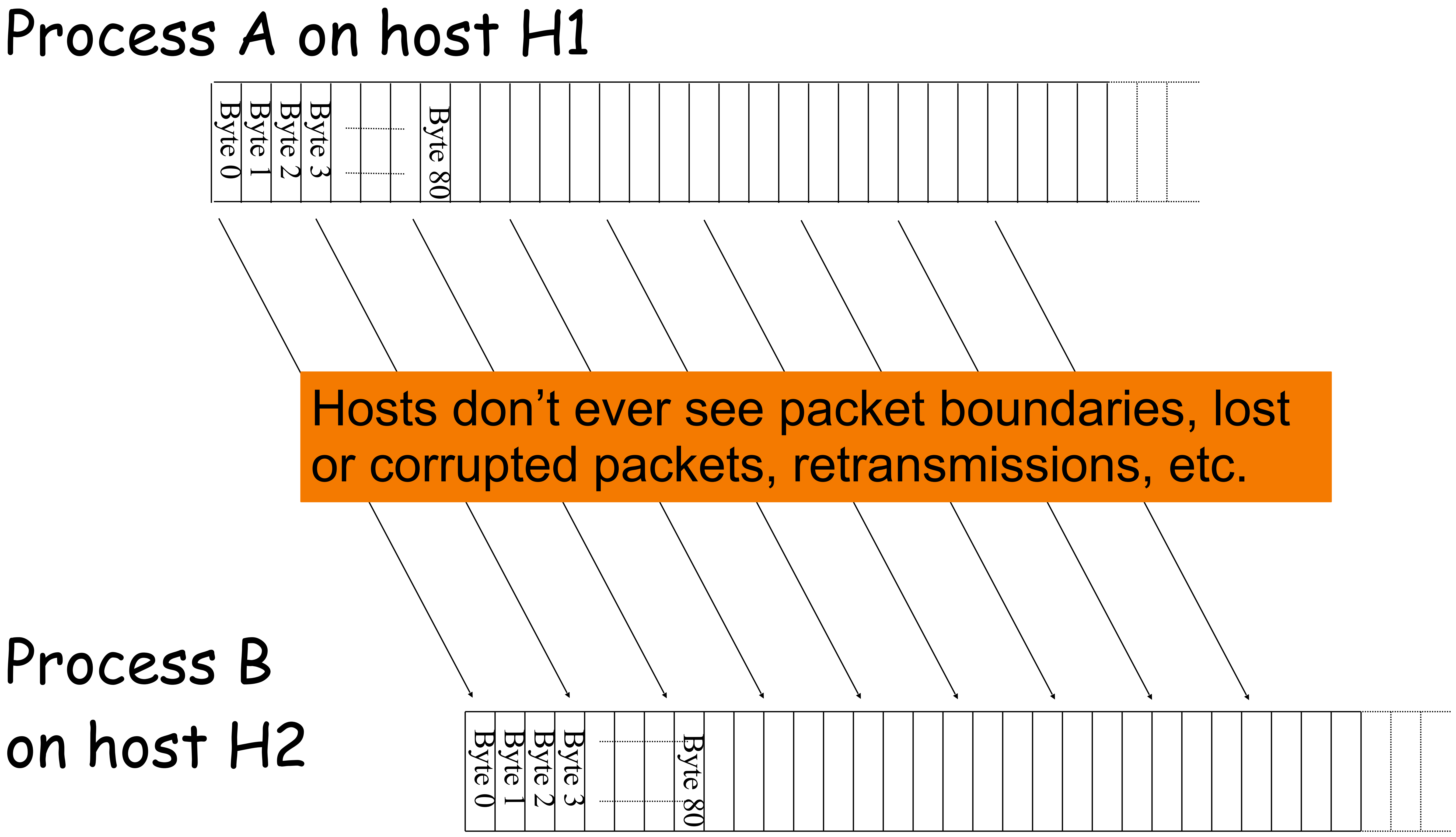
- Project 2 design doc due Friday
- Networking tutorial, Saturday 3/7, 5-7pm, in HP Auditorium (306 Soda)

"Best Effort" is Lame! What to do?

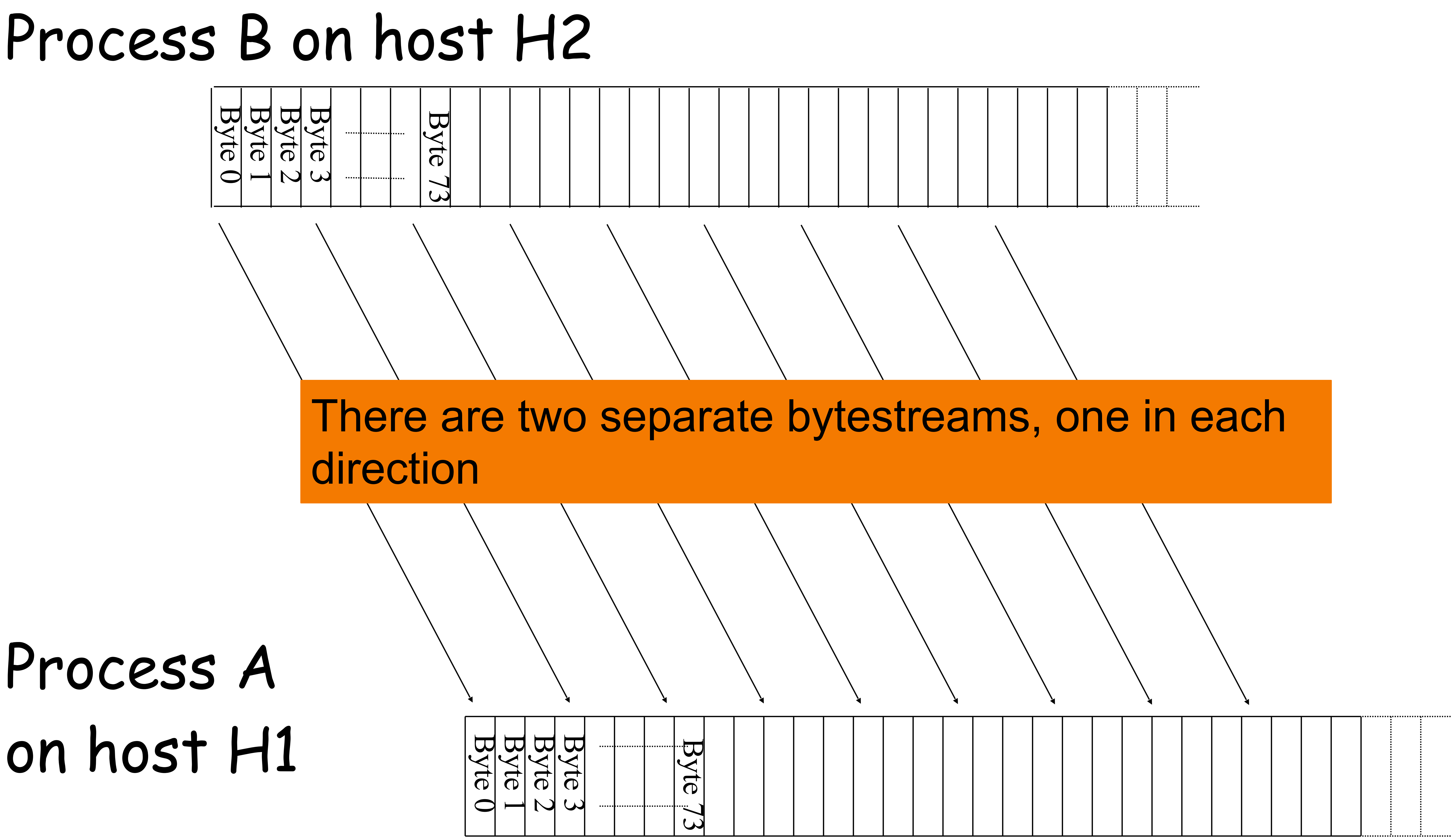
- #1 workhorse: TCP (Transmission Control Protocol)
- Service provided by TCP:
  - Connection oriented (explicit set-up / tear-down)
    - End hosts (processes) can have multiple concurrent long-lived communication
  - **Reliable**, in-order, *byte-stream* delivery
    - Robust detection & retransmission of lost data



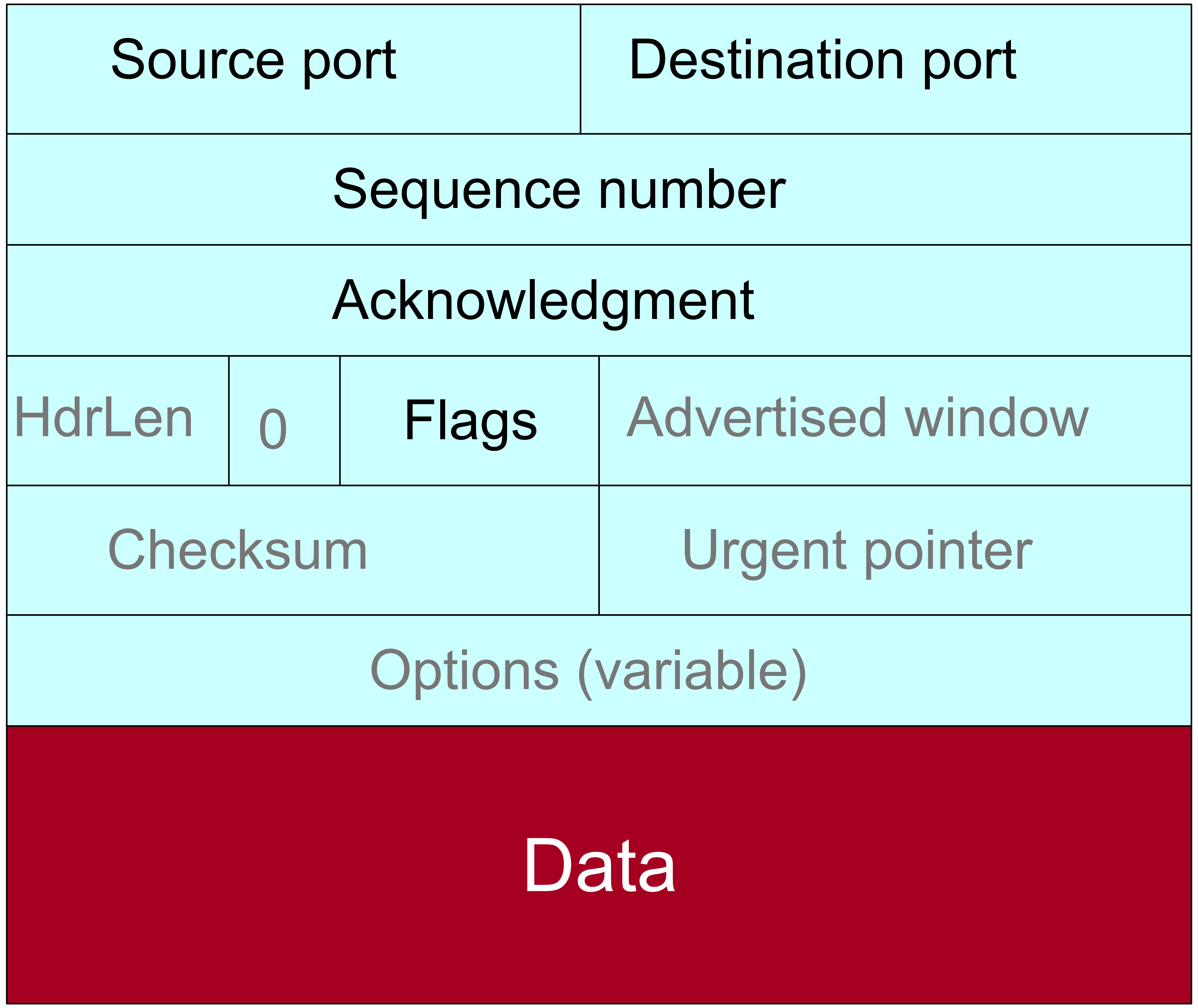
TCP "Bytestream" Service



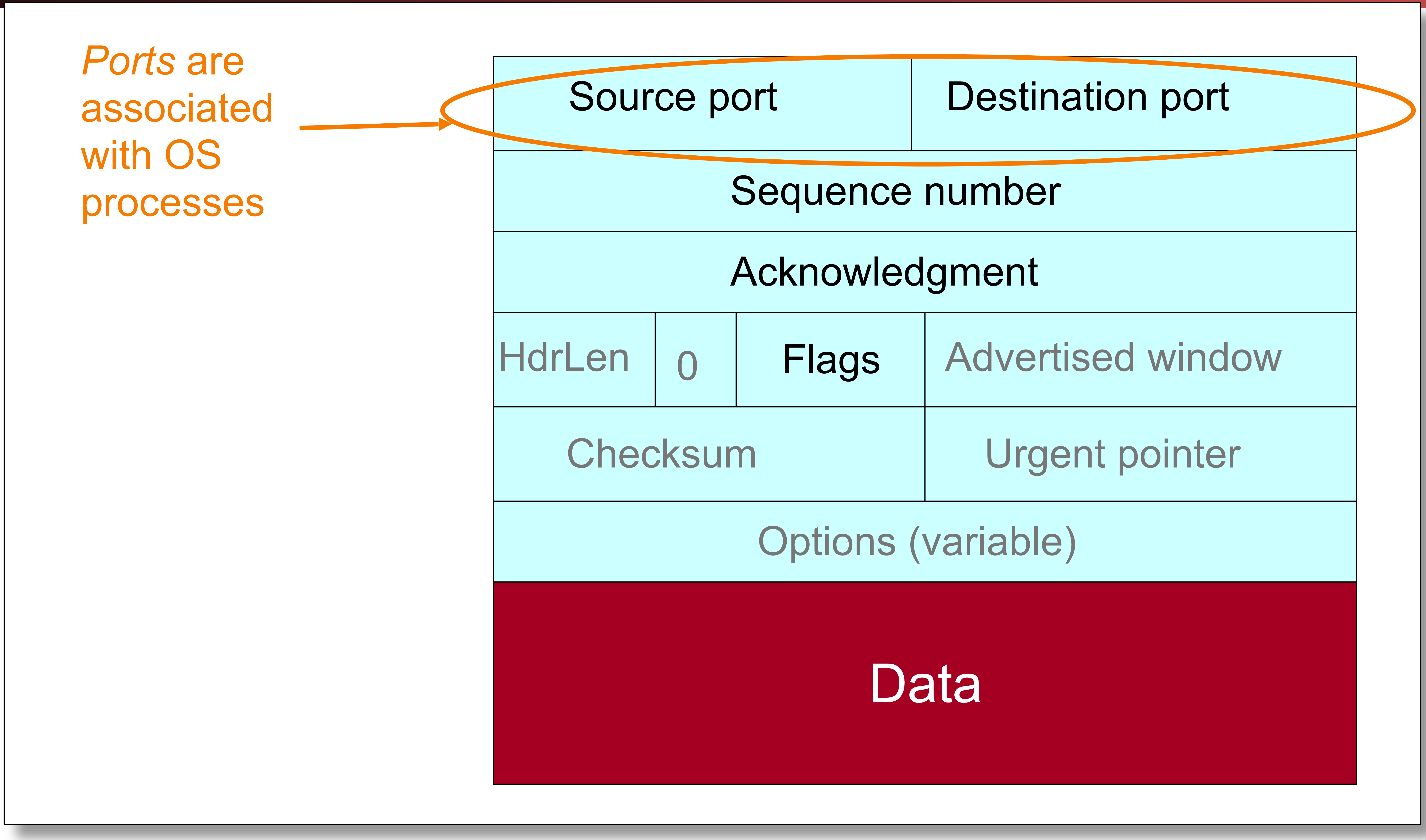
Bidirectional communication:



TCP Header

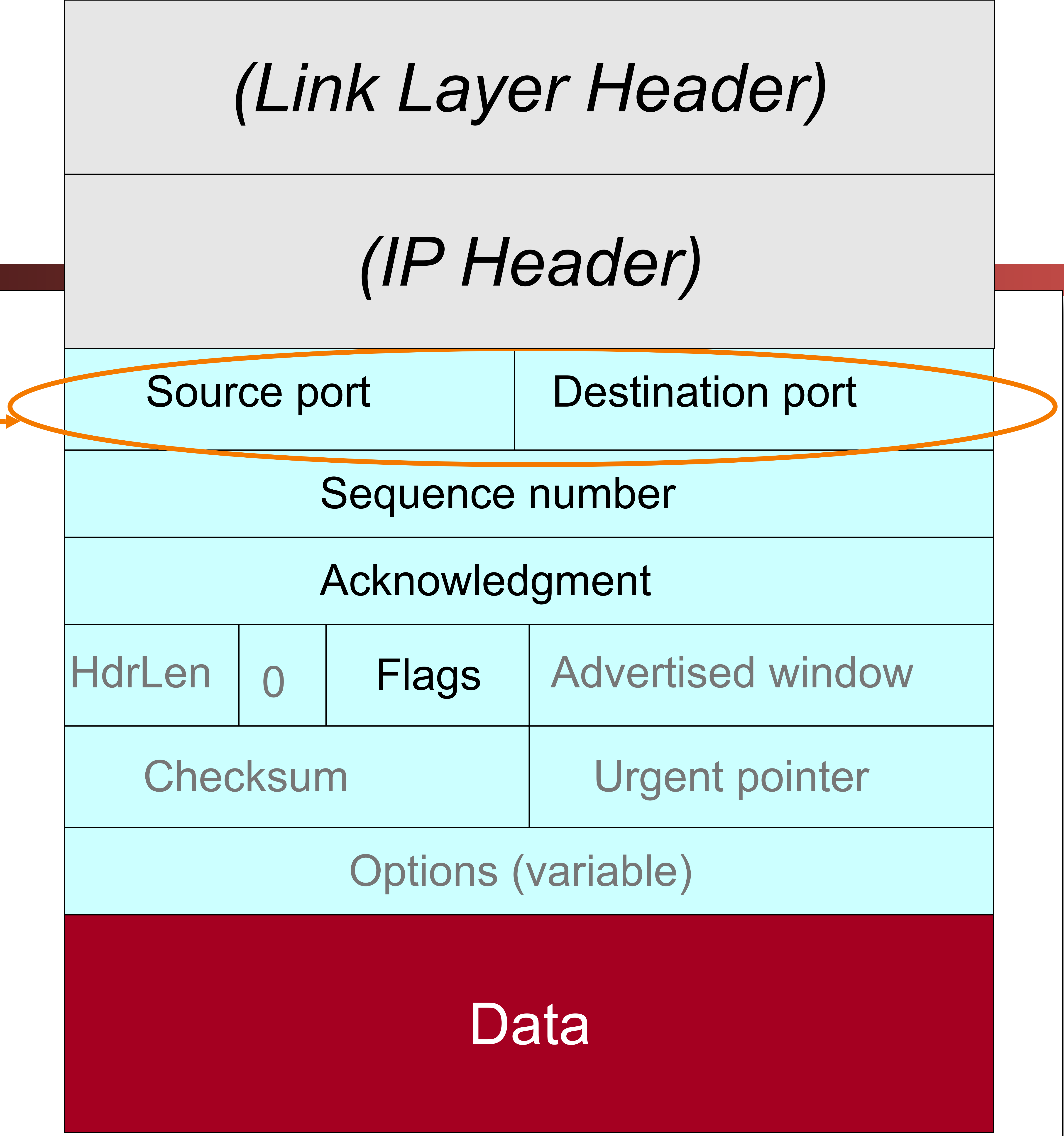

Source port
Destination port
Sequence number
Acknowledgment
HdrLen
0
Flags
Advertised window
Checksum
Urgent pointer
Options (variable)
Data

TCP Header


Ports are associated with OS processes
Source port
Destination port
Sequence number
Acknowledgment
HdrLen
0
Flags
Advertised window
Checksum
Urgent pointer
Options (variable)
Data

TCP Header

Ports are associated with OS processes



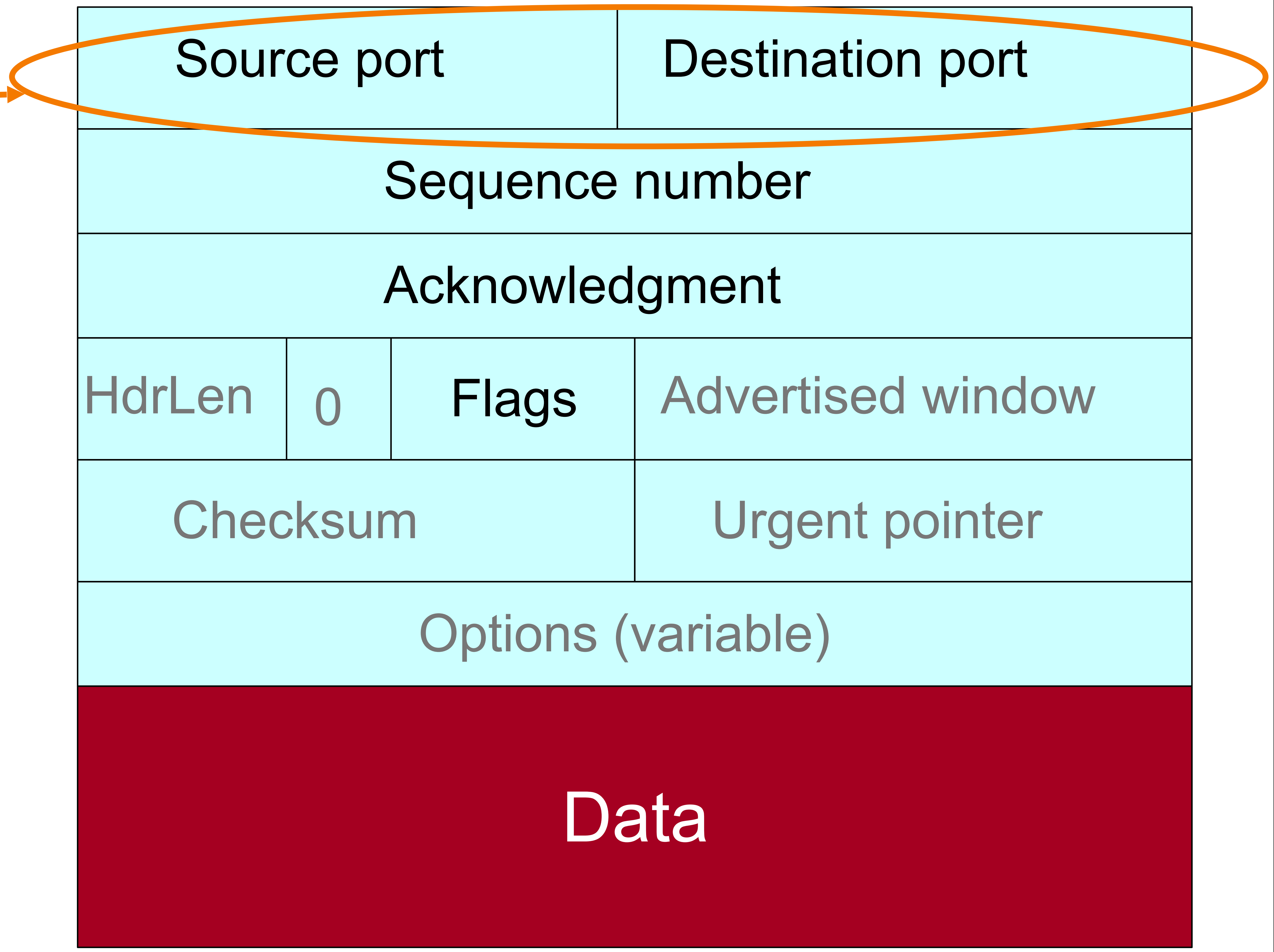
IP source & destination addresses plus TCP source and destination ports uniquely identifies a TCP connection

TCP Header

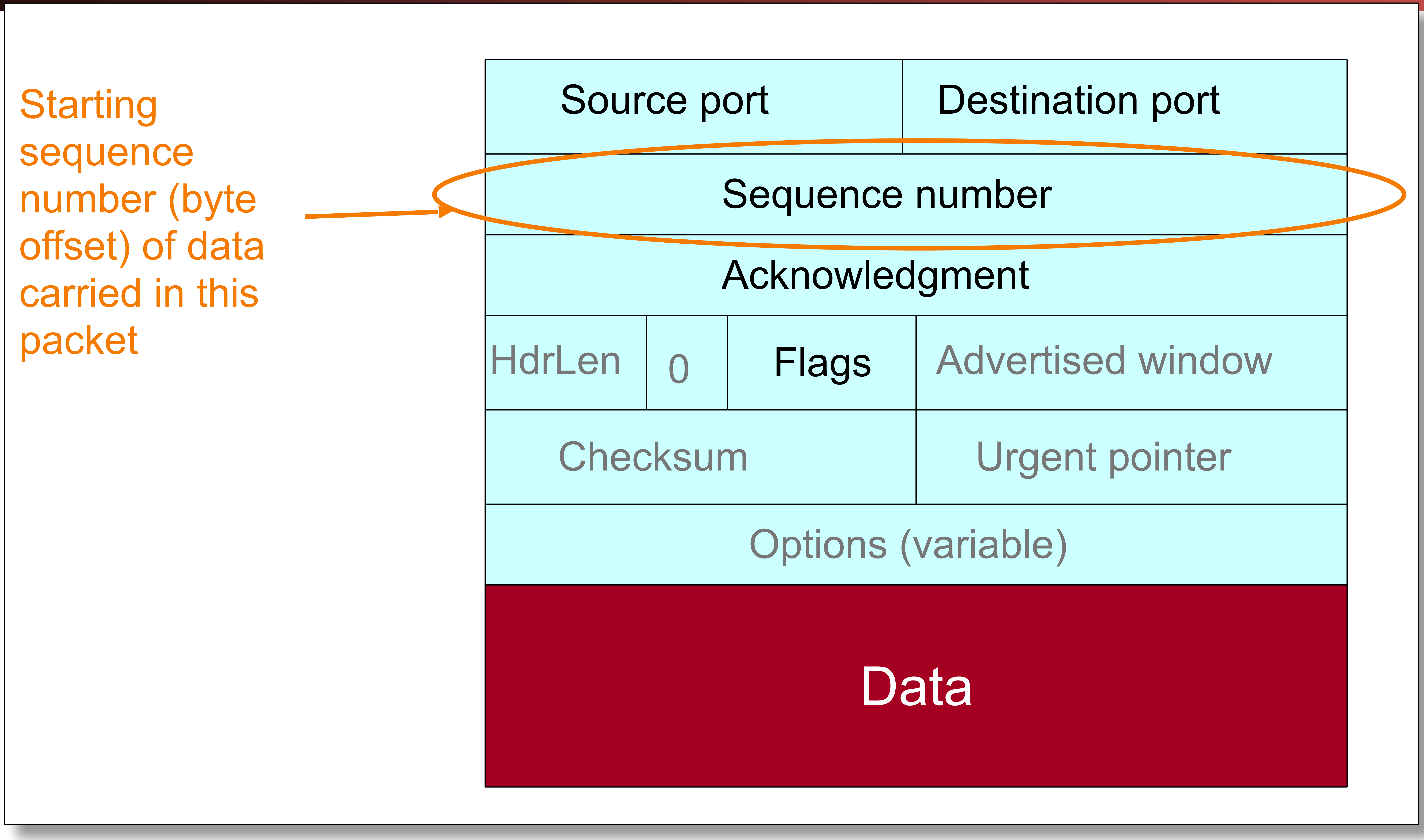
Ports are associated with OS processes

IP source & destination addresses plus TCP source and destination ports uniquely identifies a TCP connection

Some port numbers are "well known" / reserved e.g. port 80 = HTTP

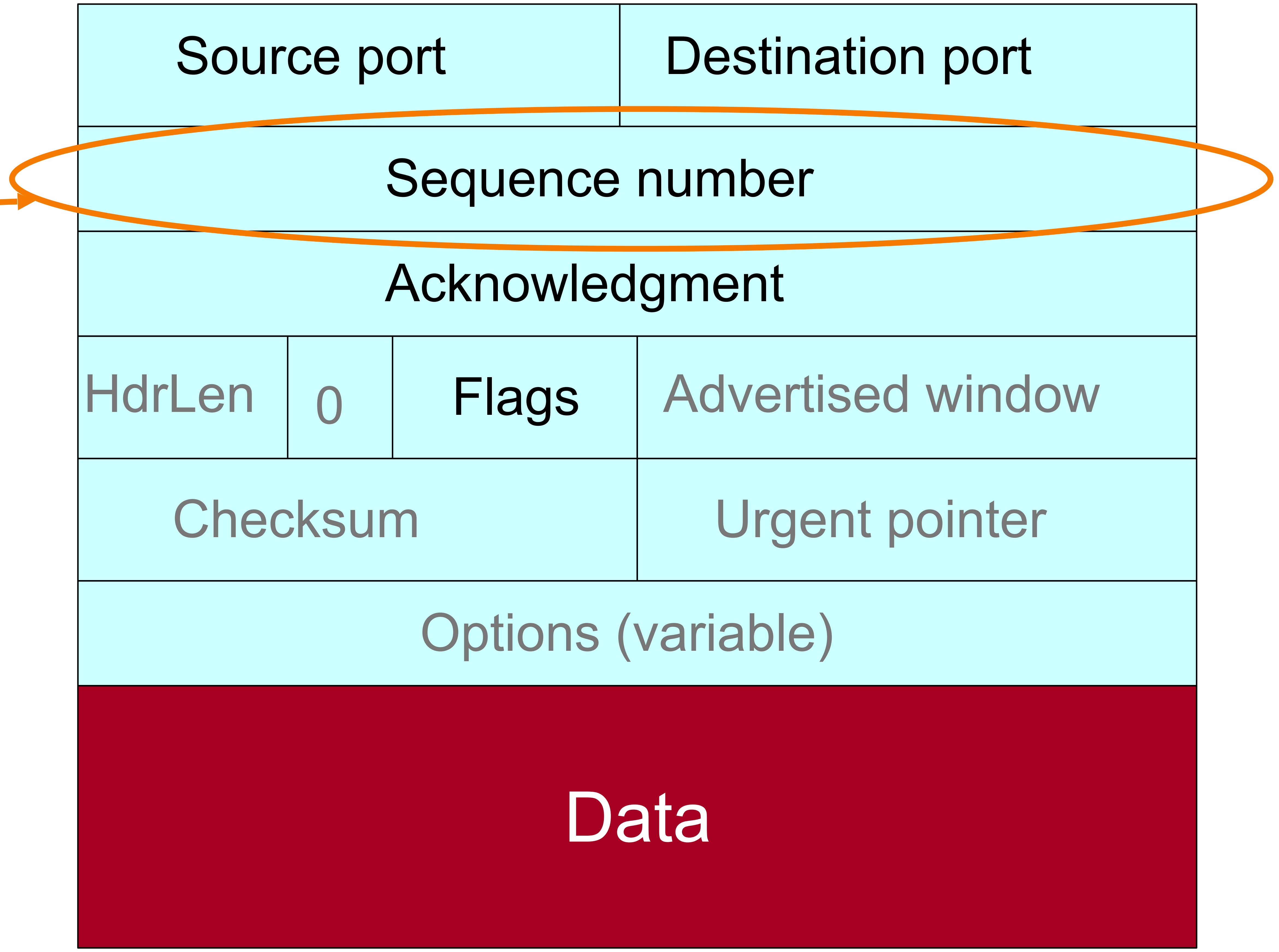


TCP Header


Starting sequence number (byte offset) of data carried in this packet
Source port
Destination port
Sequence number
Acknowledgment
HdrLen
0
Flags
Advertised window
Checksum
Urgent pointer
Options (variable)
Data

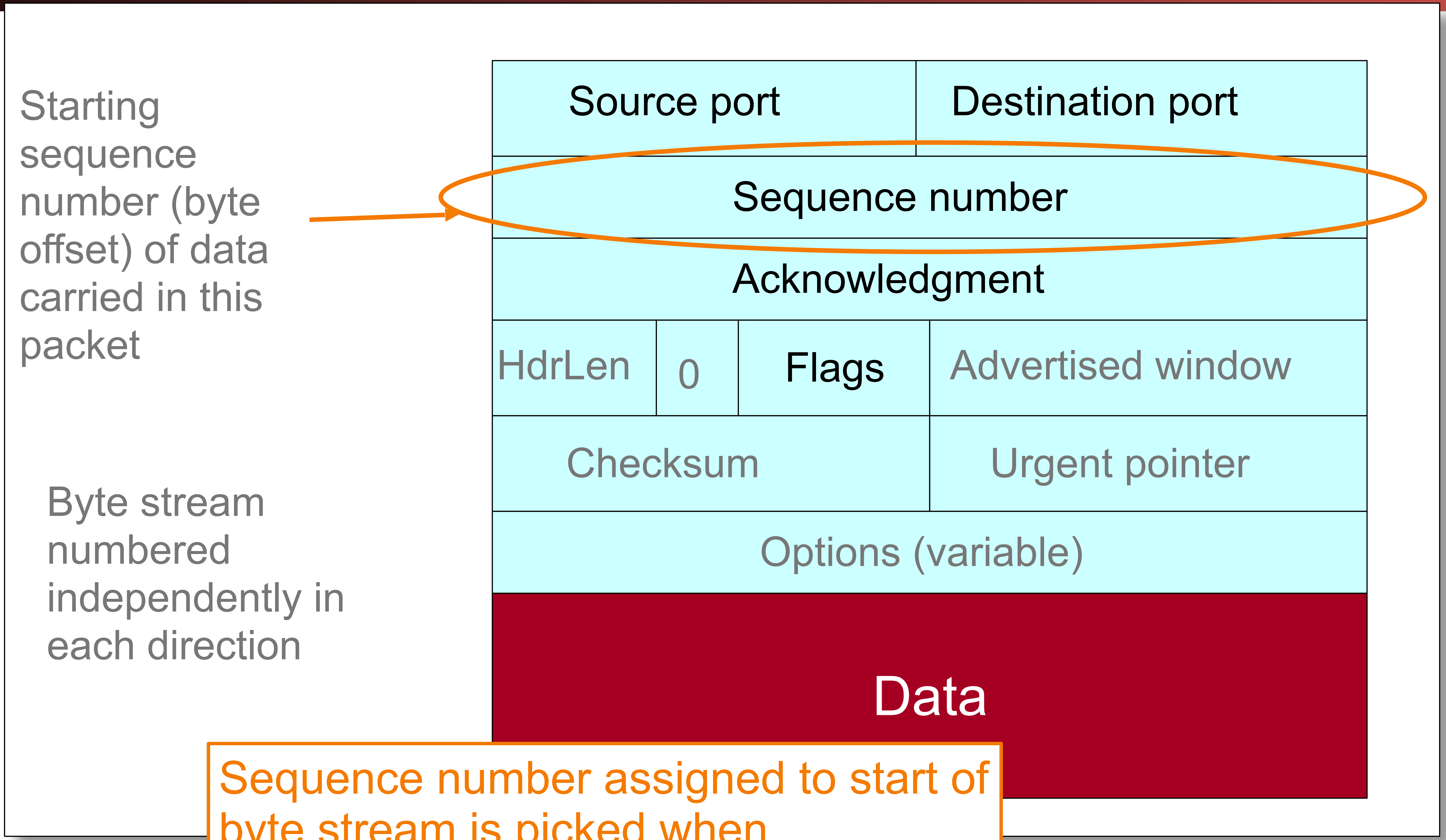
TCP Header

Starting sequence number (byte offset) of data carried in this packet



Byte streams numbered independently in each direction

TCP Header

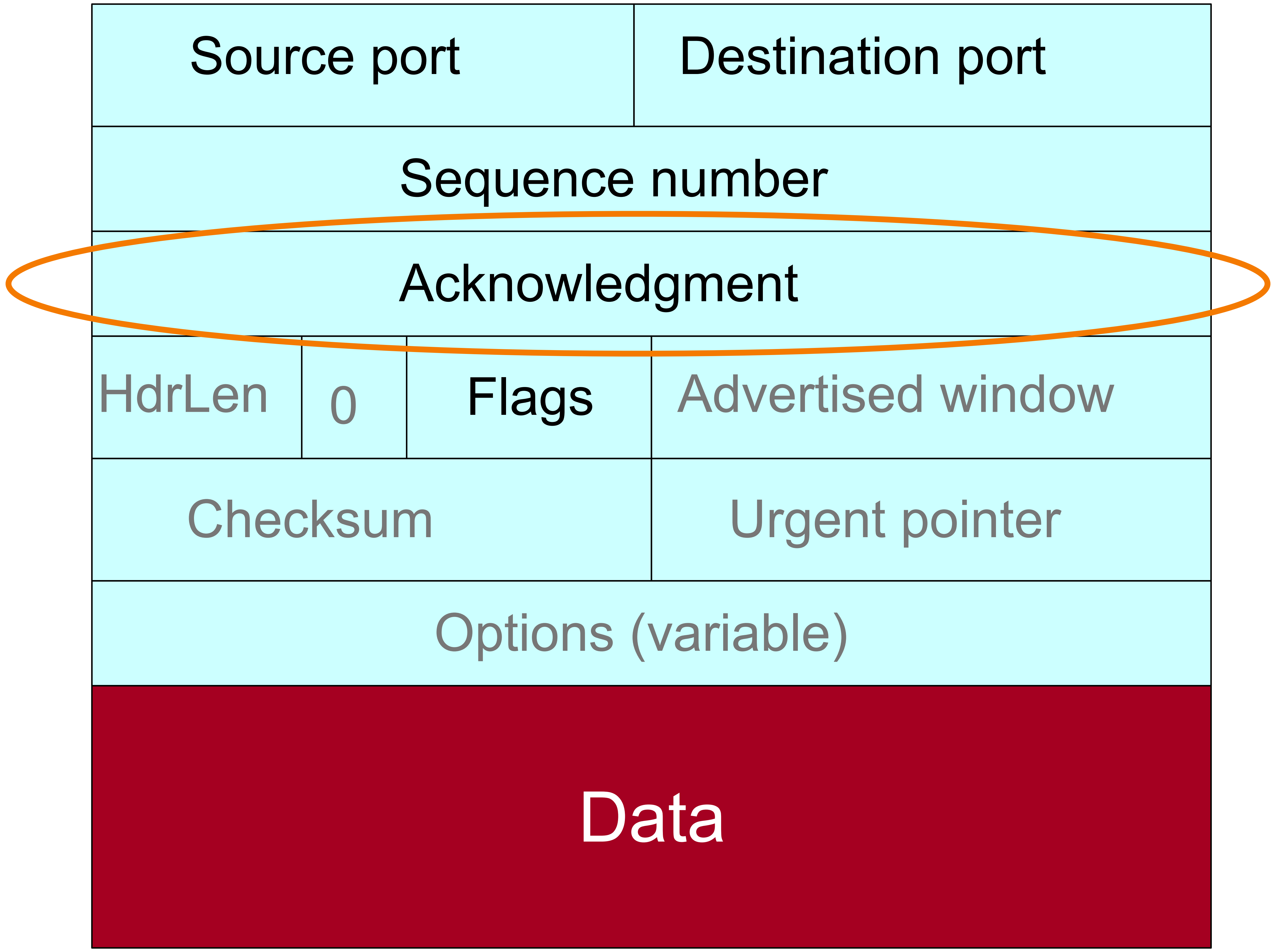


Sequence number assigned to start of byte stream is picked when connection begins; **doesn't** start at 0

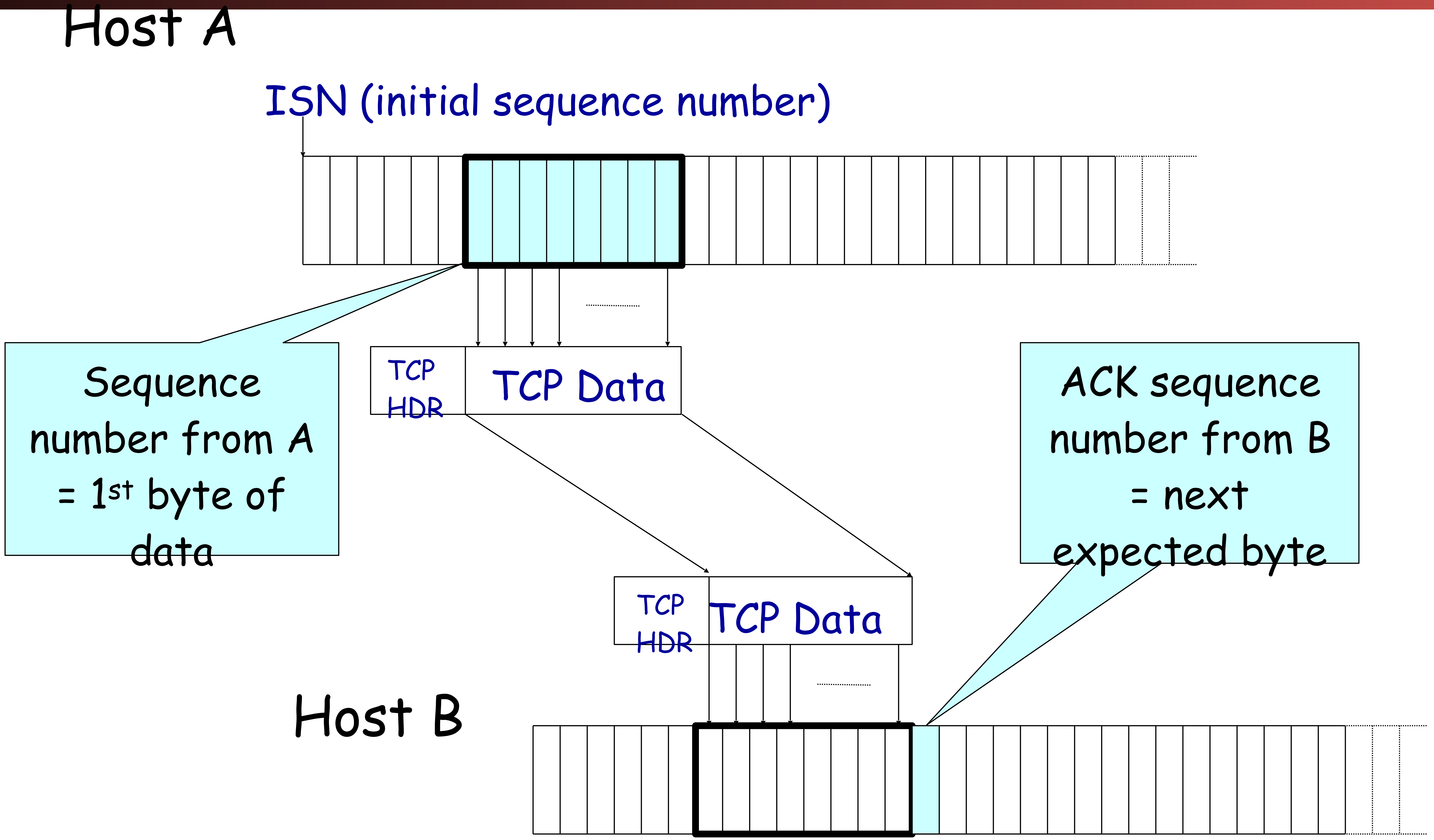
TCP Header

Acknowledgment gives seq # **just beyond** highest seq. received **in order**.

If sender sends **N** bytestream bytes starting at seq **S** then "ack" for it will be **S+N**.



Sequence Numbers

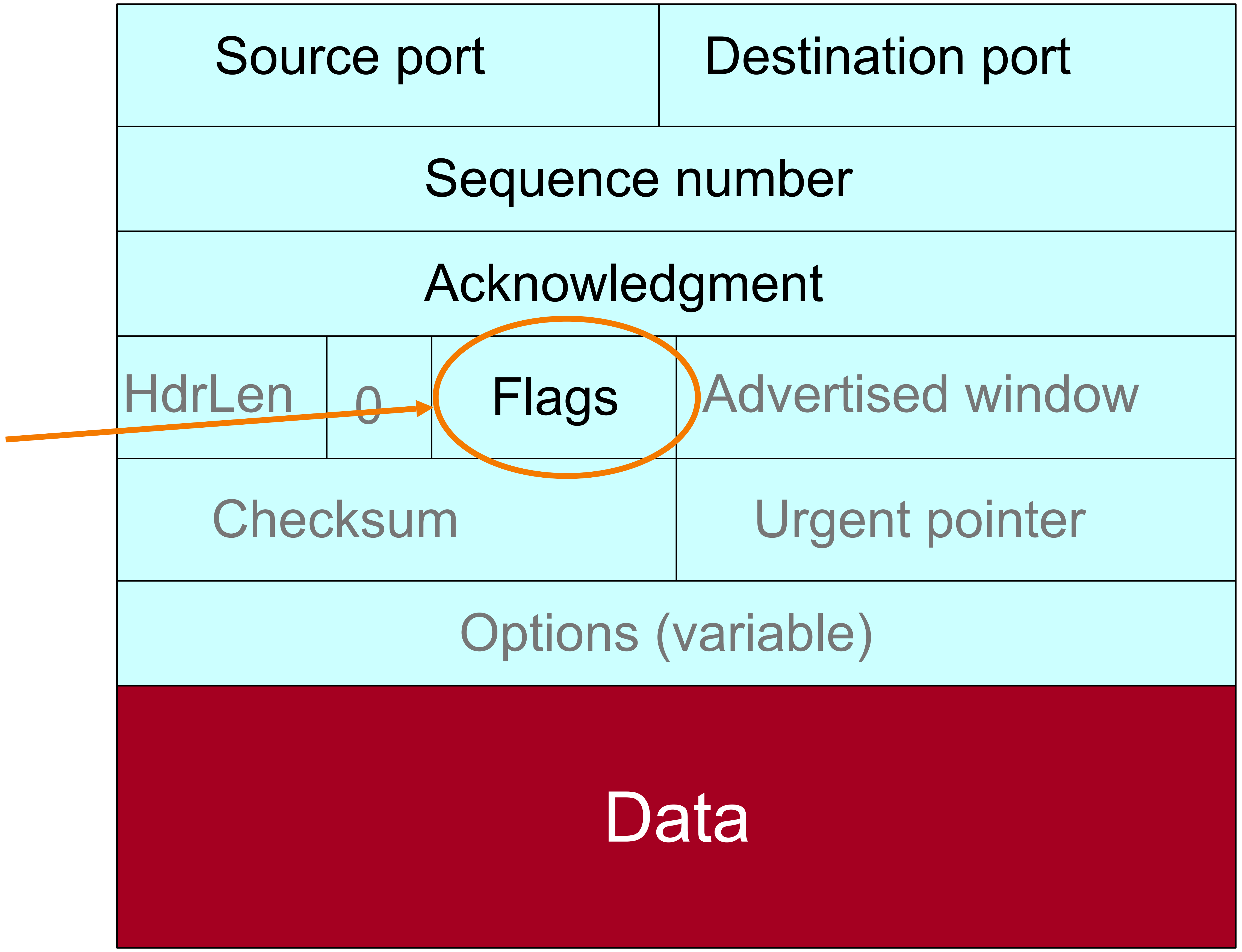


TCP Header

Uses include:

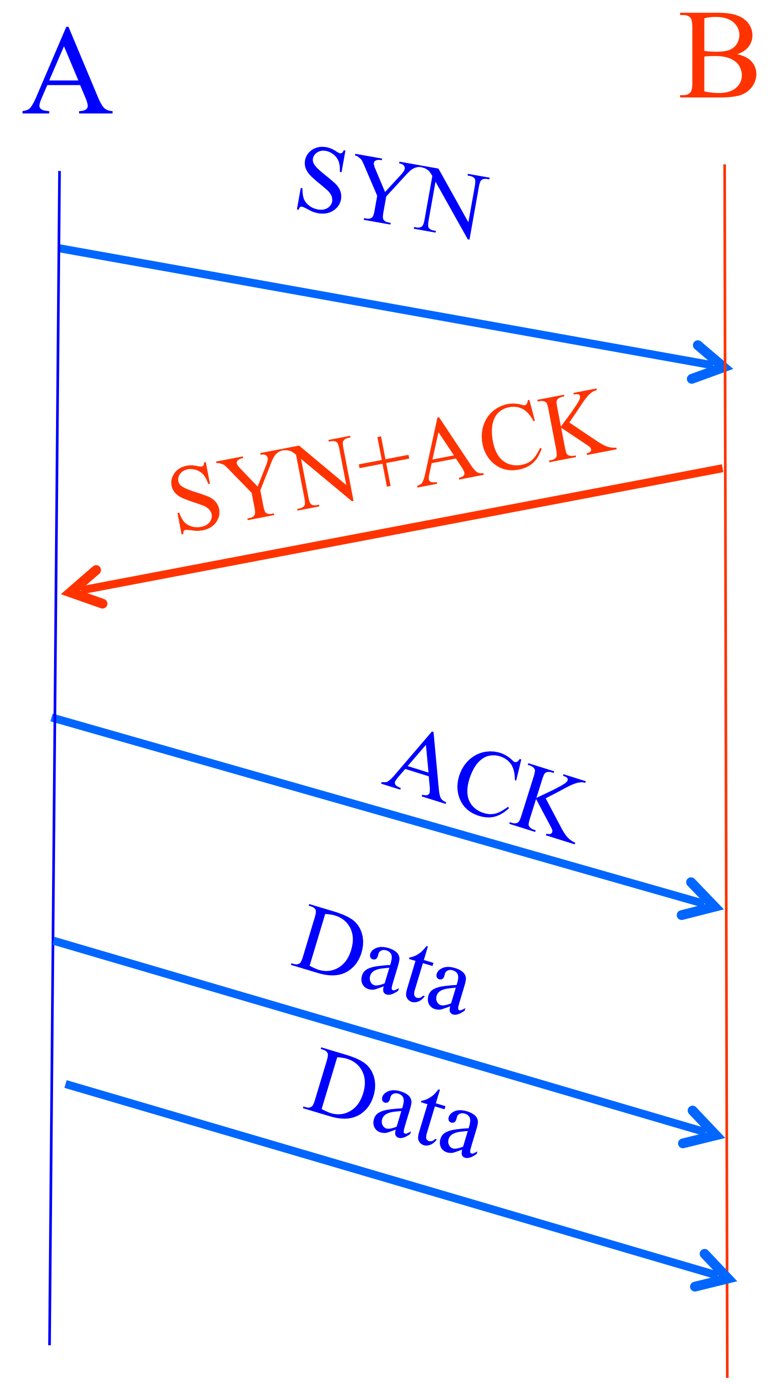
acknowledging data ("**ACK**")

setting up ("**SYN**") and closing connections ("**FIN**" and "**RST**")



Establishing a TCP Connection

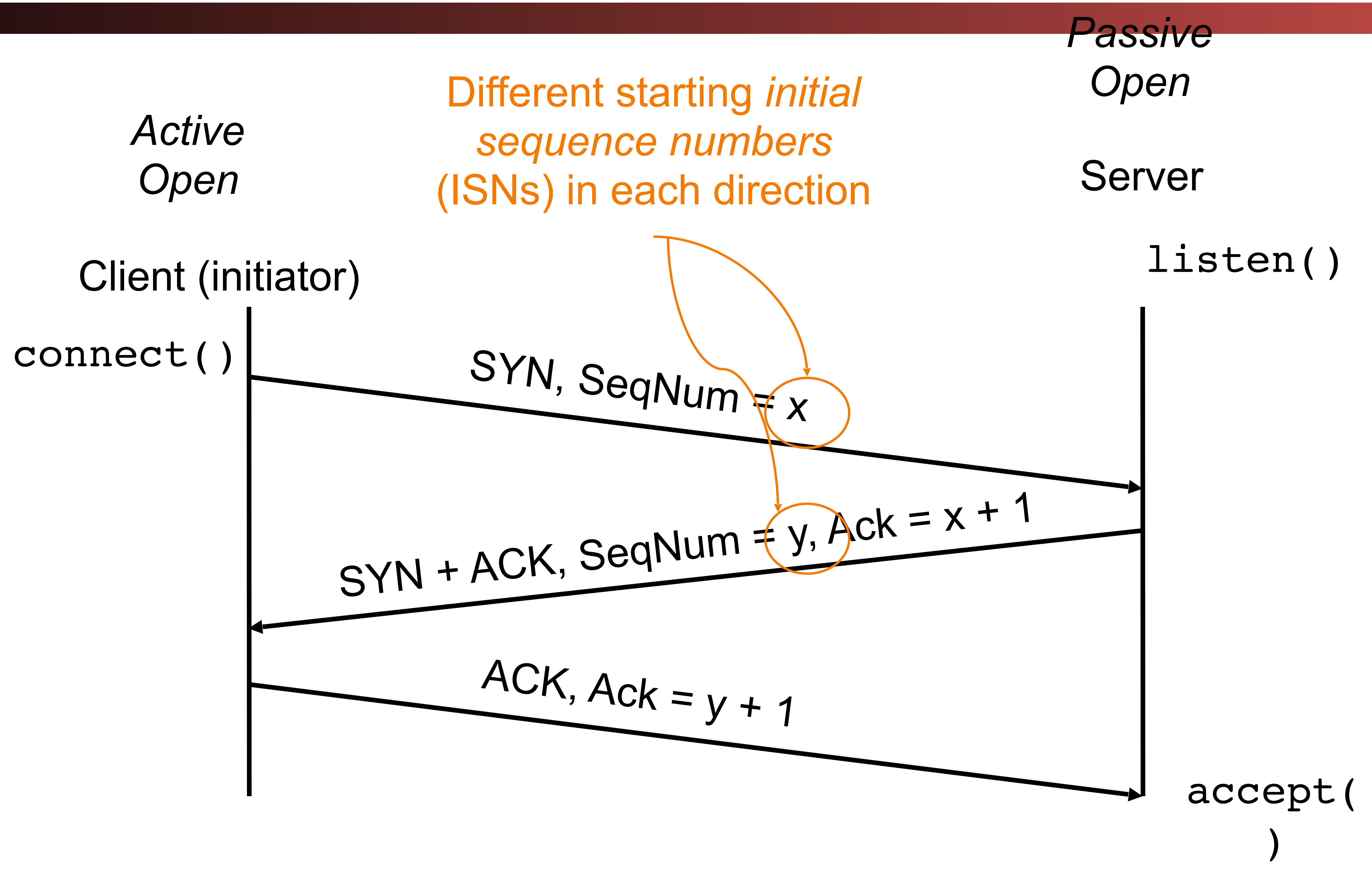
- Three-way handshake to establish connection
  - Host A sends a **`SYN`** (open; "synchronize sequence numbers") to host B
  - Host B returns a SYN acknowledgment (**`SYN+ACK`**)
  - Host A sends an **`ACK`** to acknowledge the `SYN+ACK`



Each host tells its *Initial Sequence Number* (ISN) to the other host.

(Spec says to pick based on local clock)

Timing Diagram: 3-Way Handshaking



Host Names vs. IP addresses

- Host names
  - Examples: `www.cnn.com` and `bbc.co.uk`
  - Mnemonic name appreciated by humans
  - Variable length, full alphabet of characters
  - Provide little (if any) information about location
- IP addresses
  - Examples: `64.236.16.20` and `212.58.224.131`
  - Numerical address appreciated by routers
  - Fixed length, binary number
  - Hierarchical, related to host location

So Let's Do A Google Search...

- Walk into a coffee shop
- Open a laptop
- Search google...

Coffee Shop

Coffee Shop

1. Join the wireless network

Your laptop shouts:
HEY, DOES WIRELESS NETWORK X EXIST?

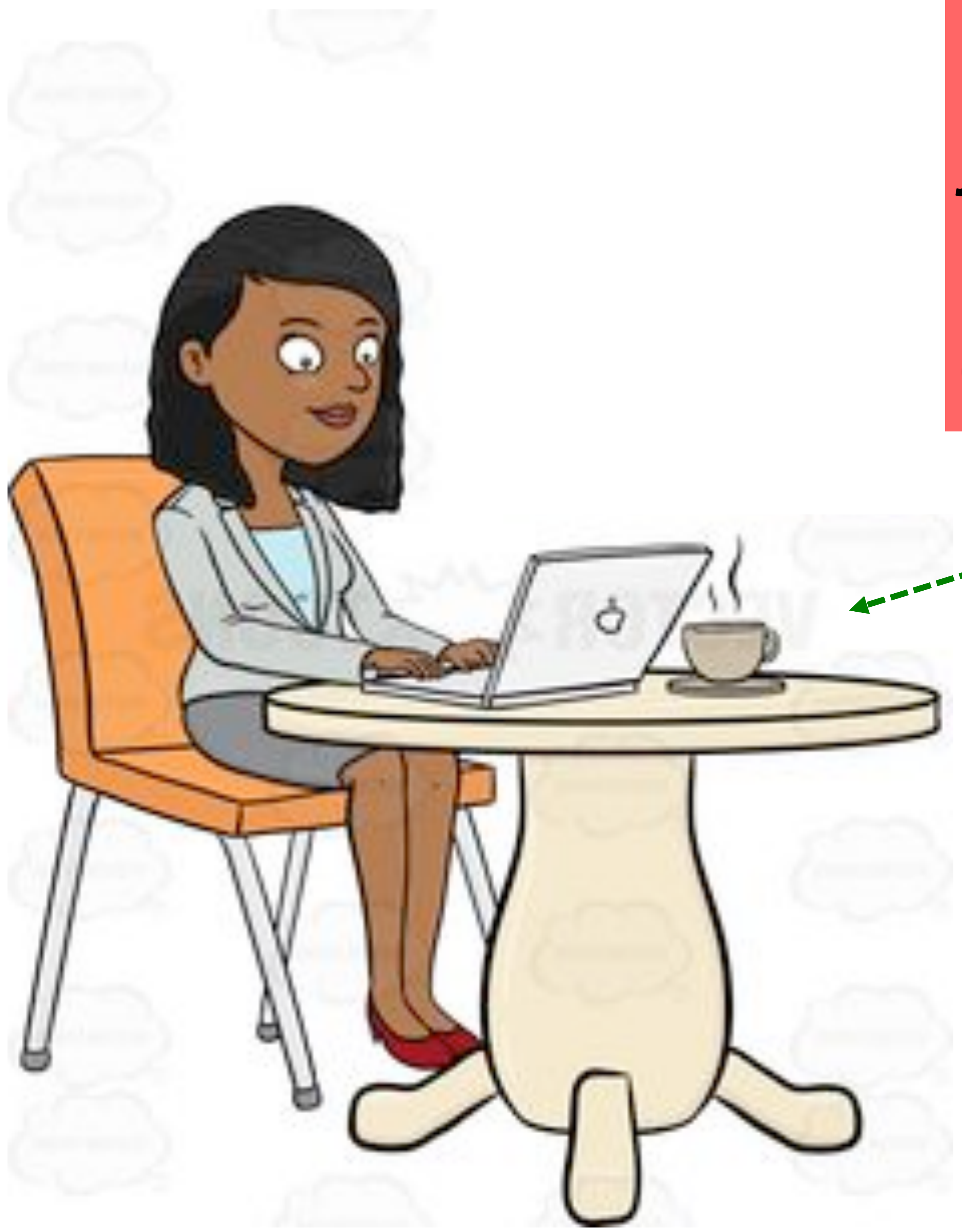
Coffee Shop

1. Join the wireless network

Wireless access point(s) continually shout:
HEY, I'M WIRELESS NETWORK Y, JOIN ME!

Coffee Shop

1. Join the wireless network



If either match up, your laptop joins the network. Optionally performs a cryptographic exchange.

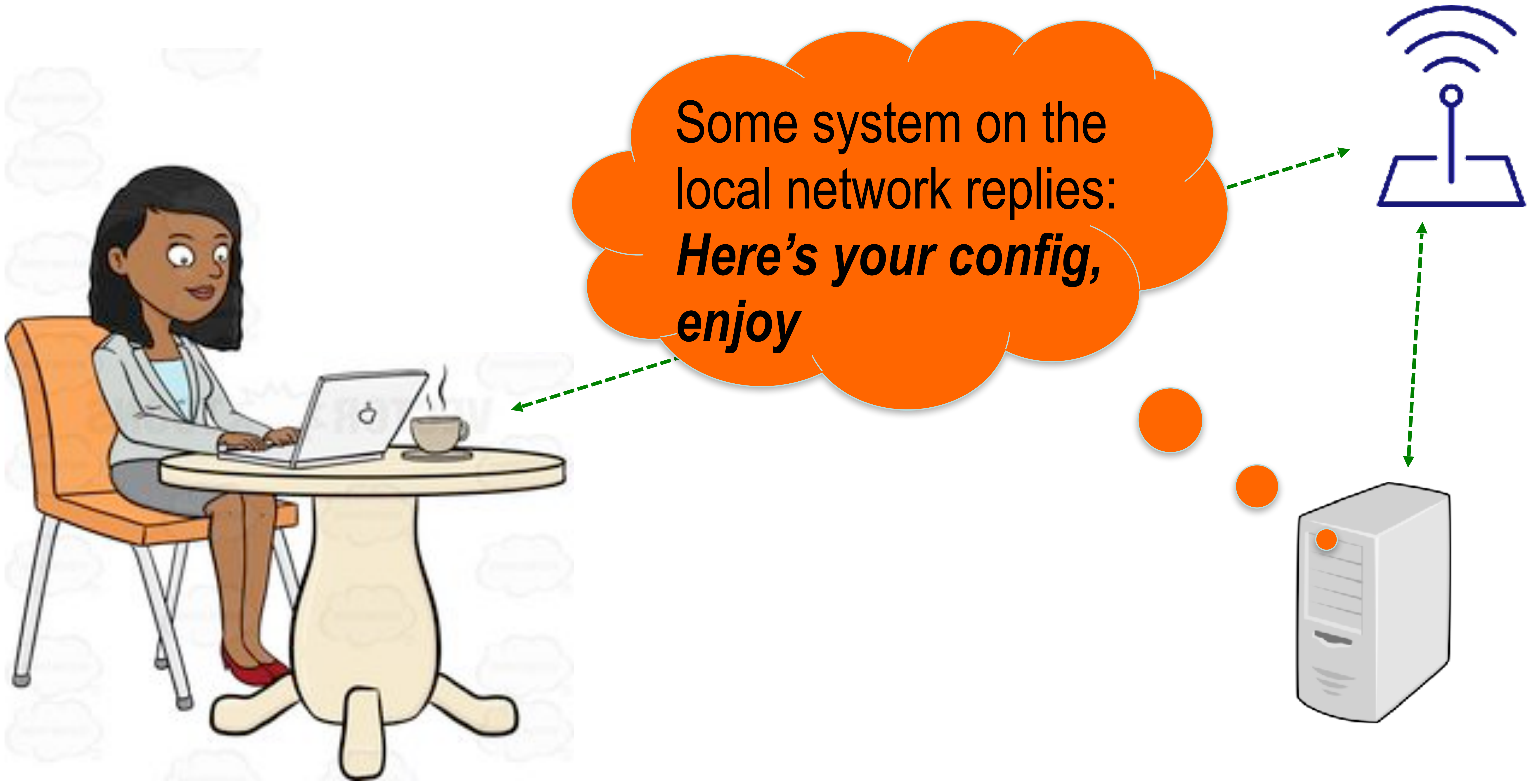
Coffee Shop

2. Configure your connection

Your laptop shouts:
HEY, ANYBODY, WHAT BASIC CONFIG DO I NEED TO USE?

Coffee Shop

2. Configure your connection



Coffee Shop

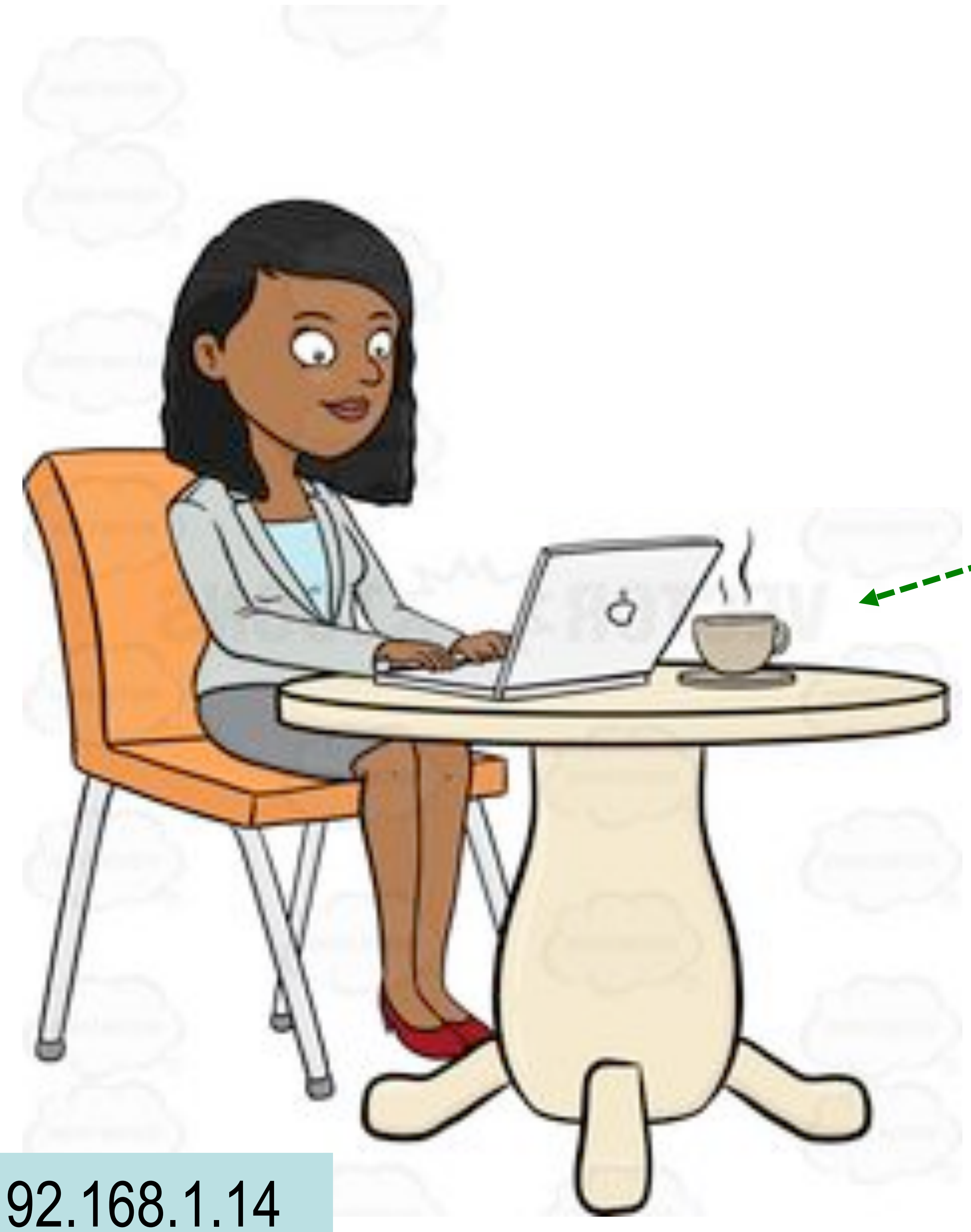
2. Configure your connection

The configuration includes:

(1) An Internet address (**IP address**) your laptop should use; typically 32 bits (IPv4).

(2) The address of a "**gateway**" system to use to access ***hosts*** beyond the local network

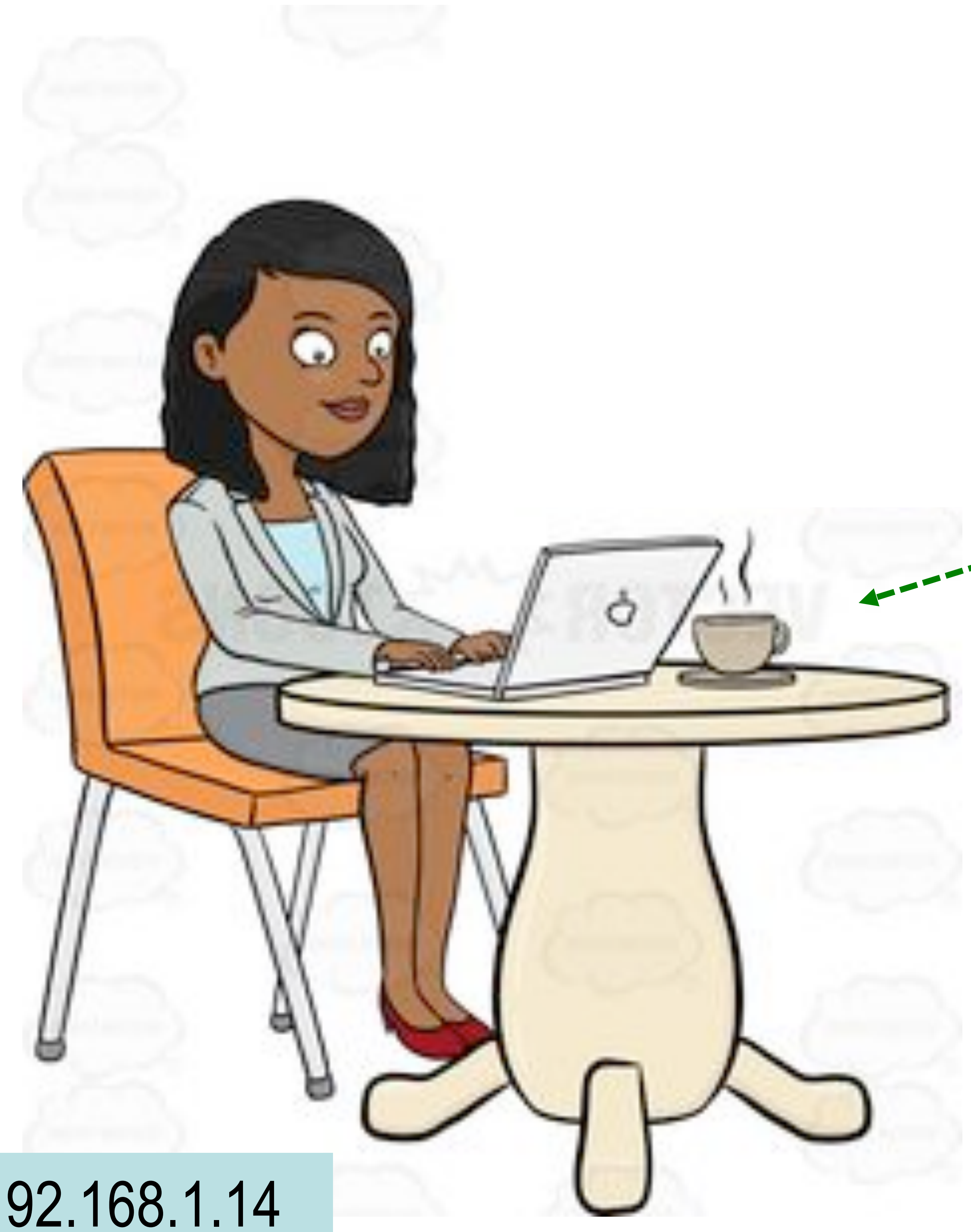
(3) The address of a **DNS server** ("***resolver***") to map names like `google.com` to IP addresses



192.168.1.14

Coffee Shop

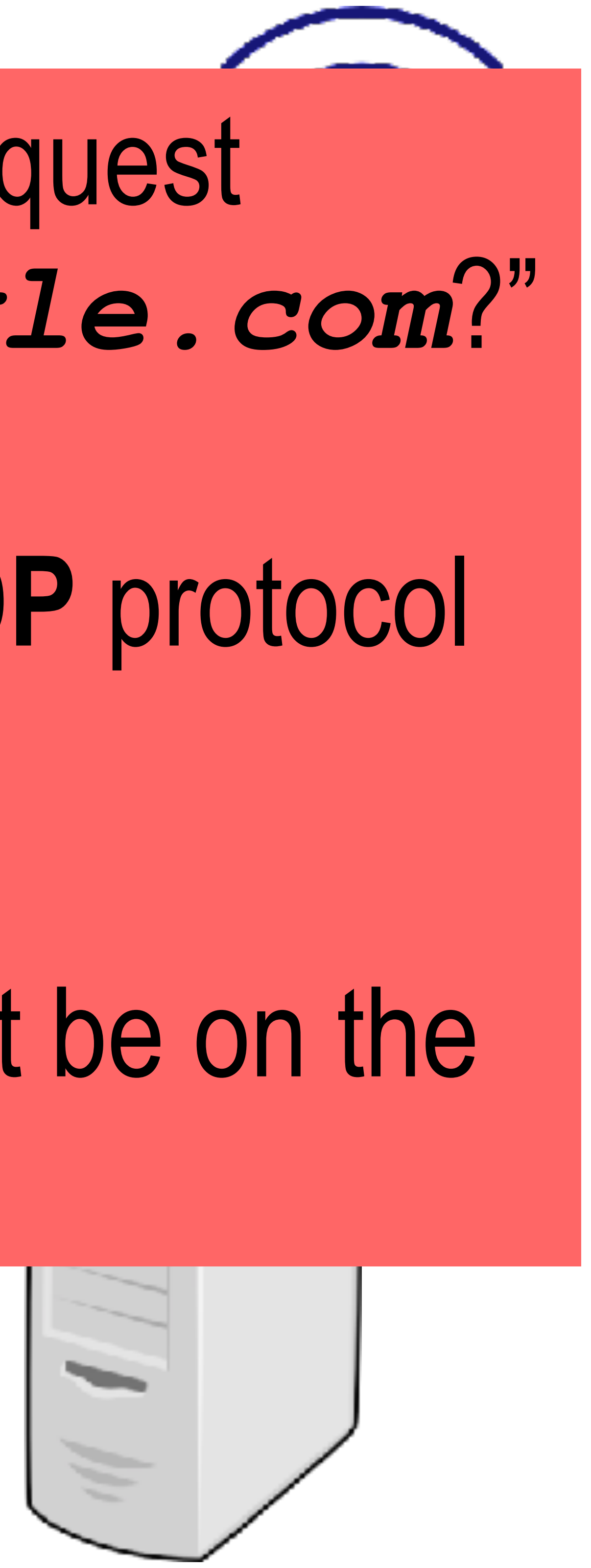
3. Find the address of **`google.com`**



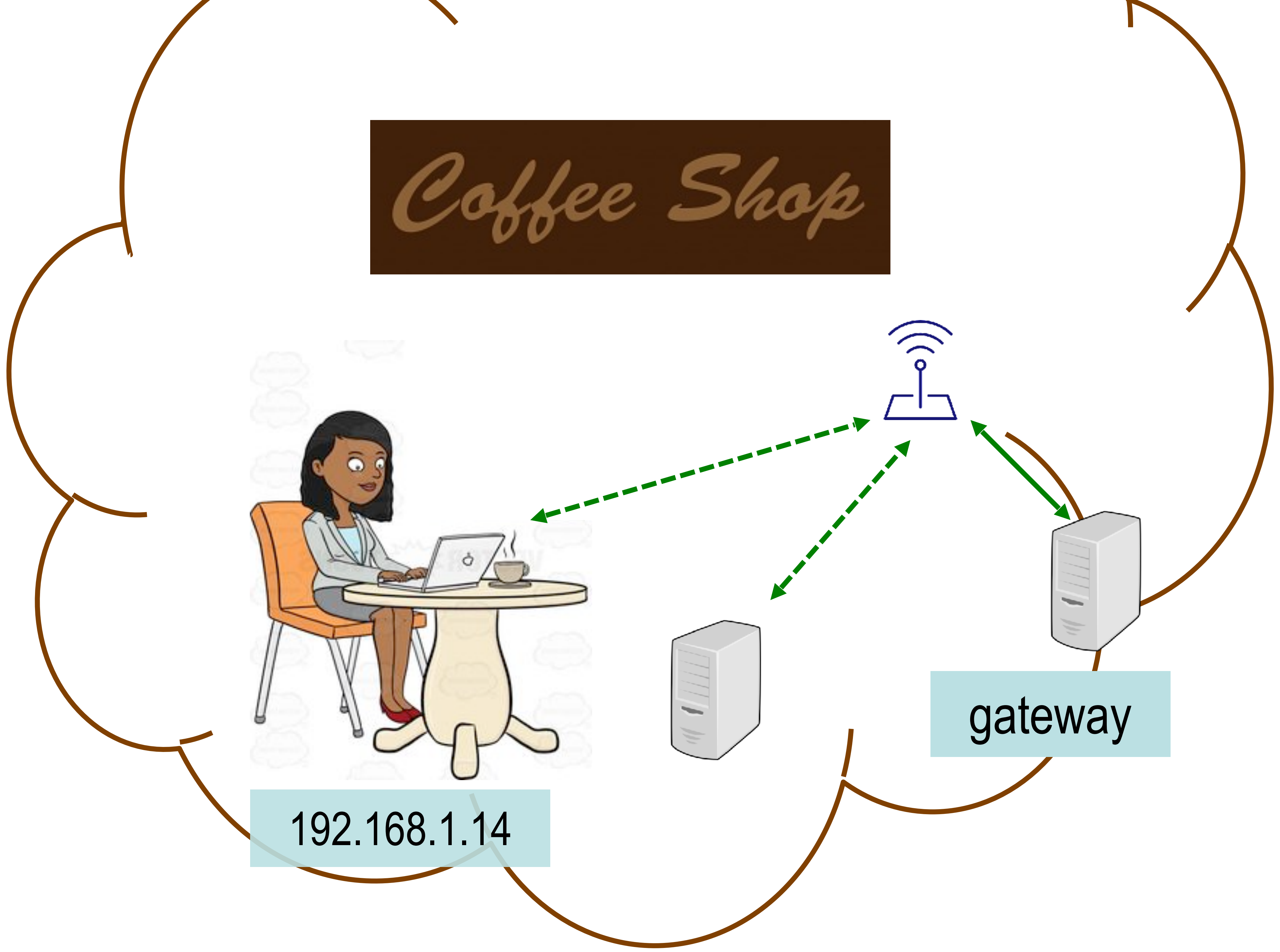
Your laptop sends a **DNS** request asking: "***address for* `google.com`**?"

It's transmitted using the **UDP** protocol (lightweight, unreliable).

The DNS **resolver** might not be on the local network.

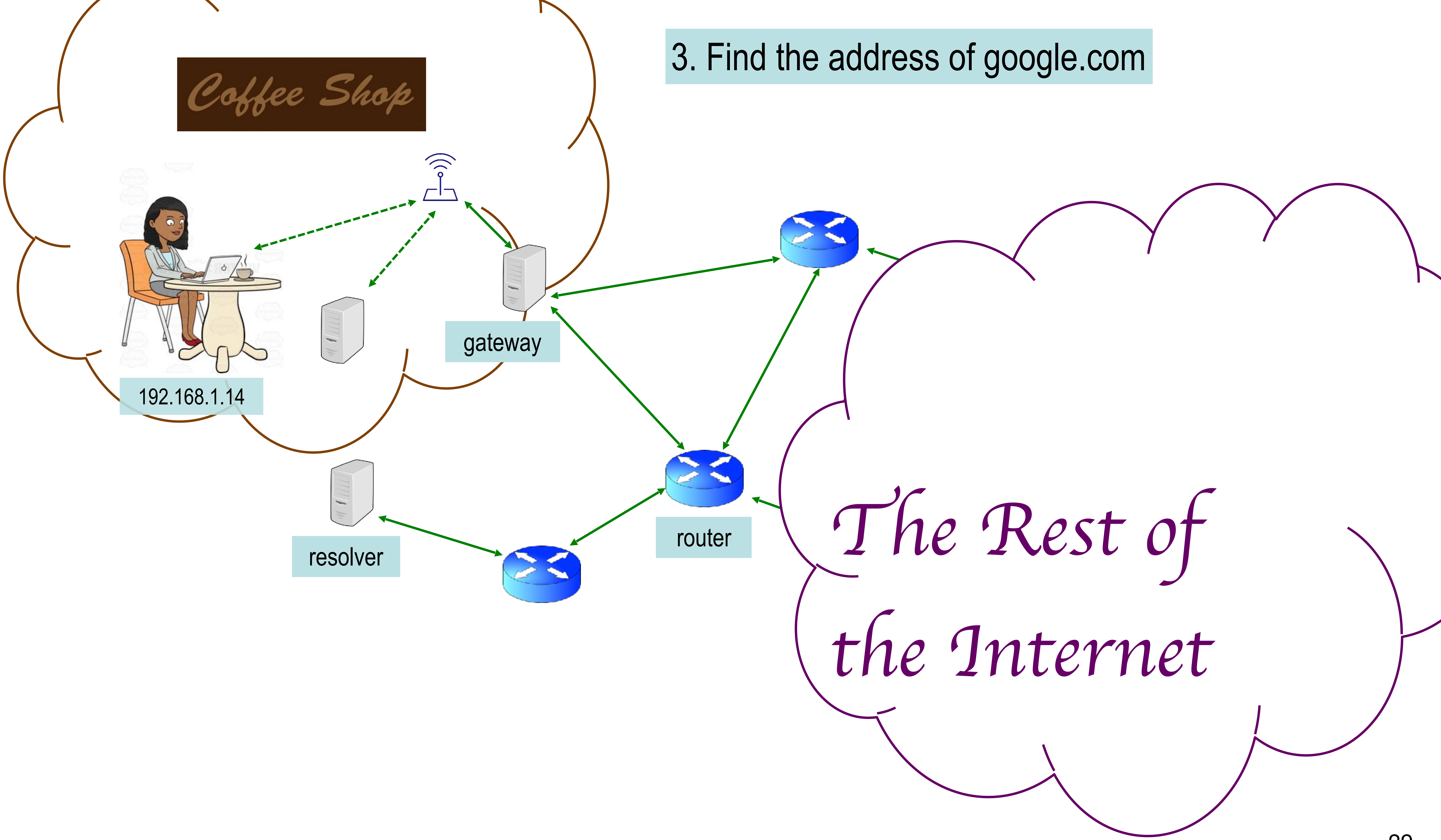


192.168.1.14

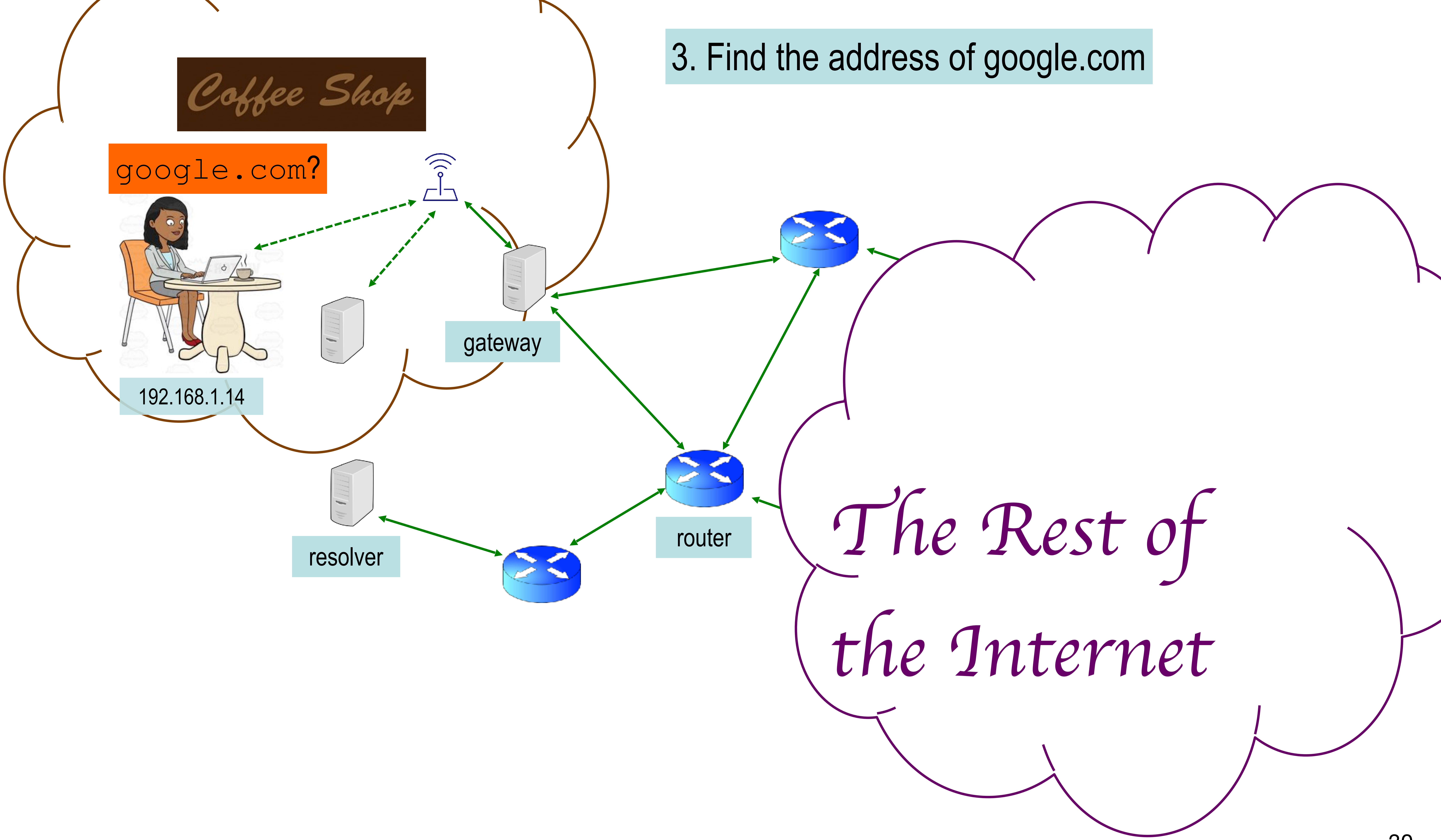


3. Find the address of google.com

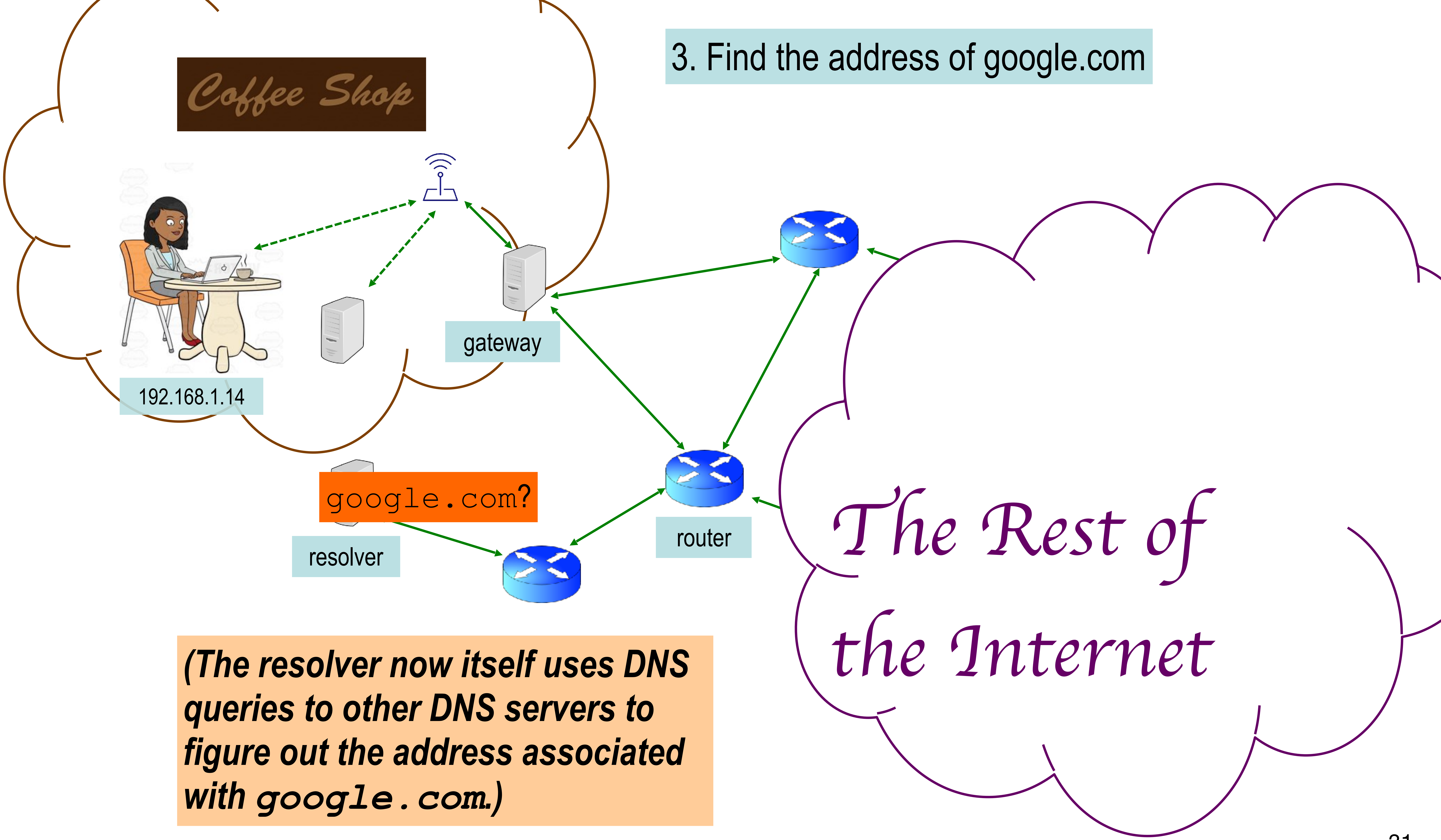
3. Find the address of google.com



3. Find the address of google.com

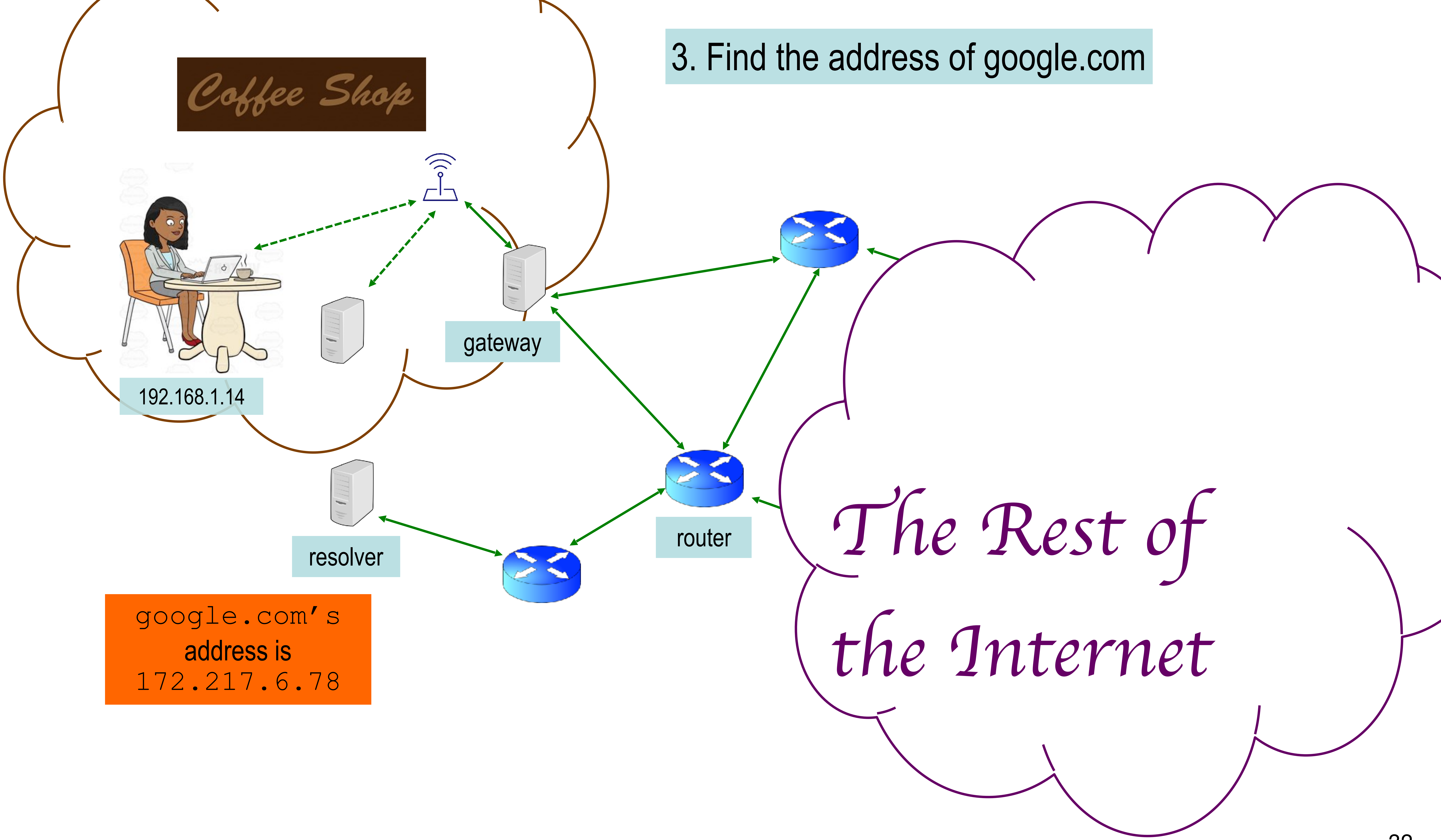


3. Find the address of google.com



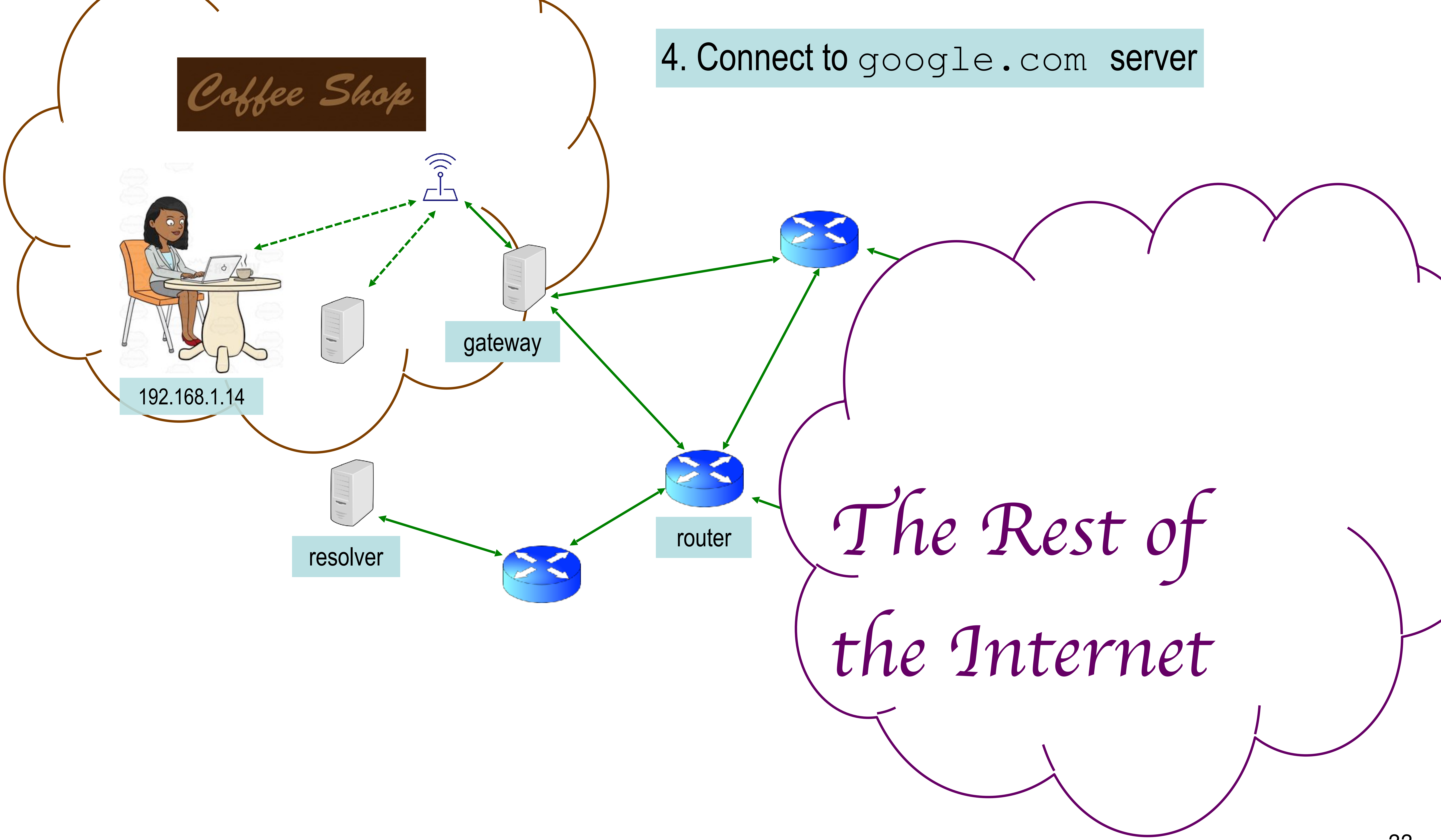
(The resolver now itself uses DNS queries to other DNS servers to figure out the address associated with ***`google.com`****.)*

3. Find the address of google.com

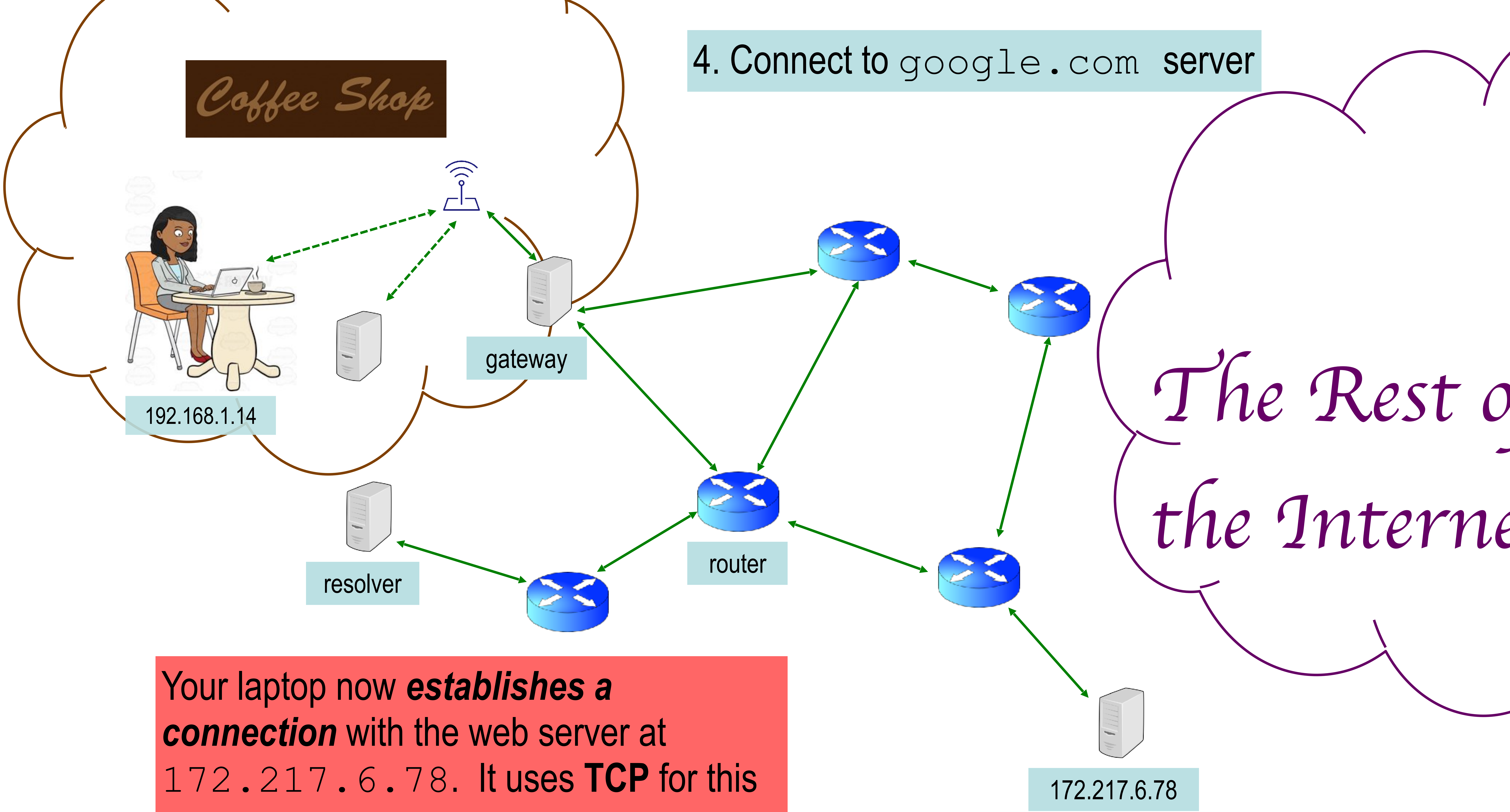


`google.com’s` address is `172.217.6.78`

4. Connect to `google.com` server

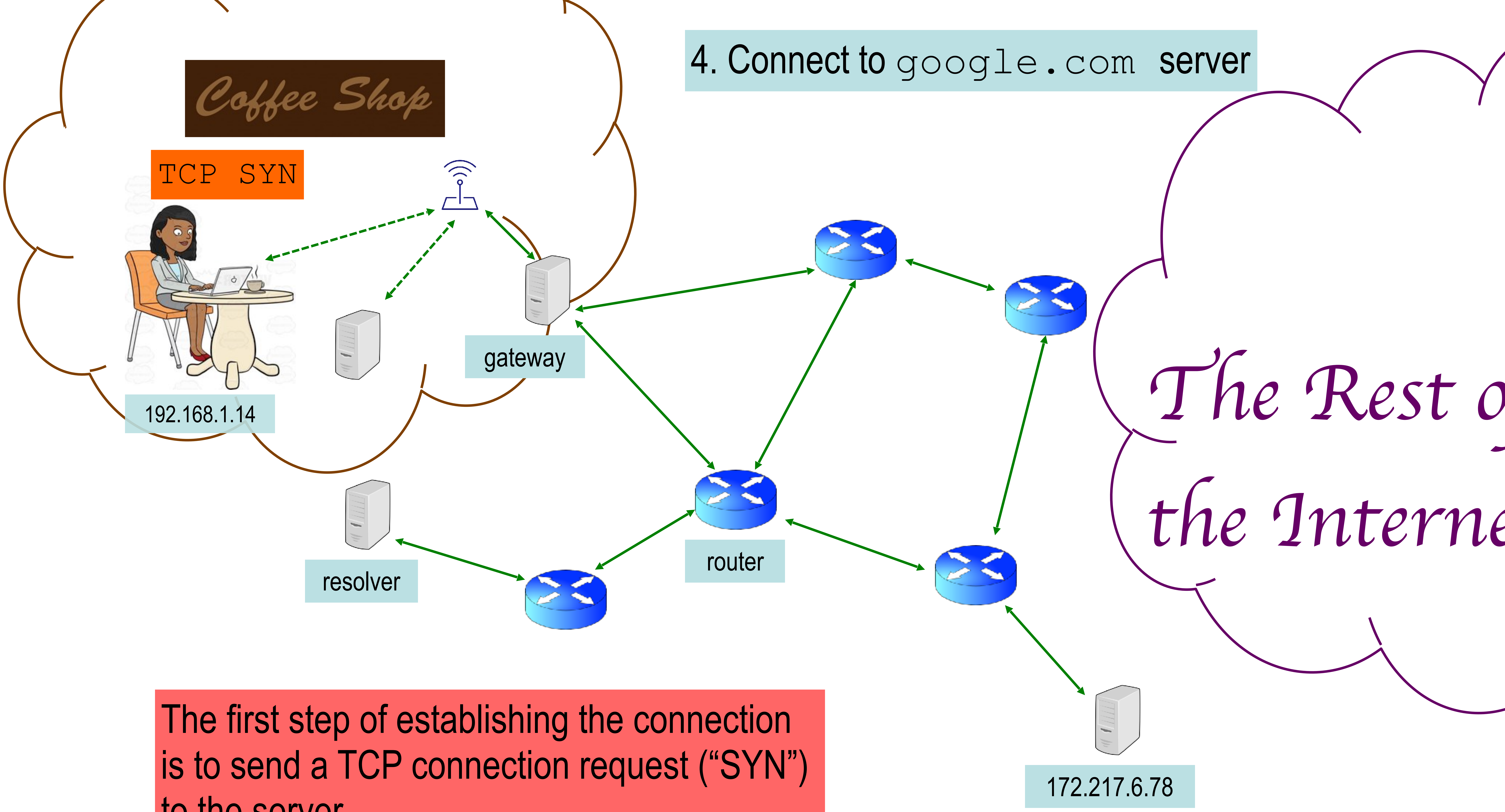


4. Connect to `google.com` server



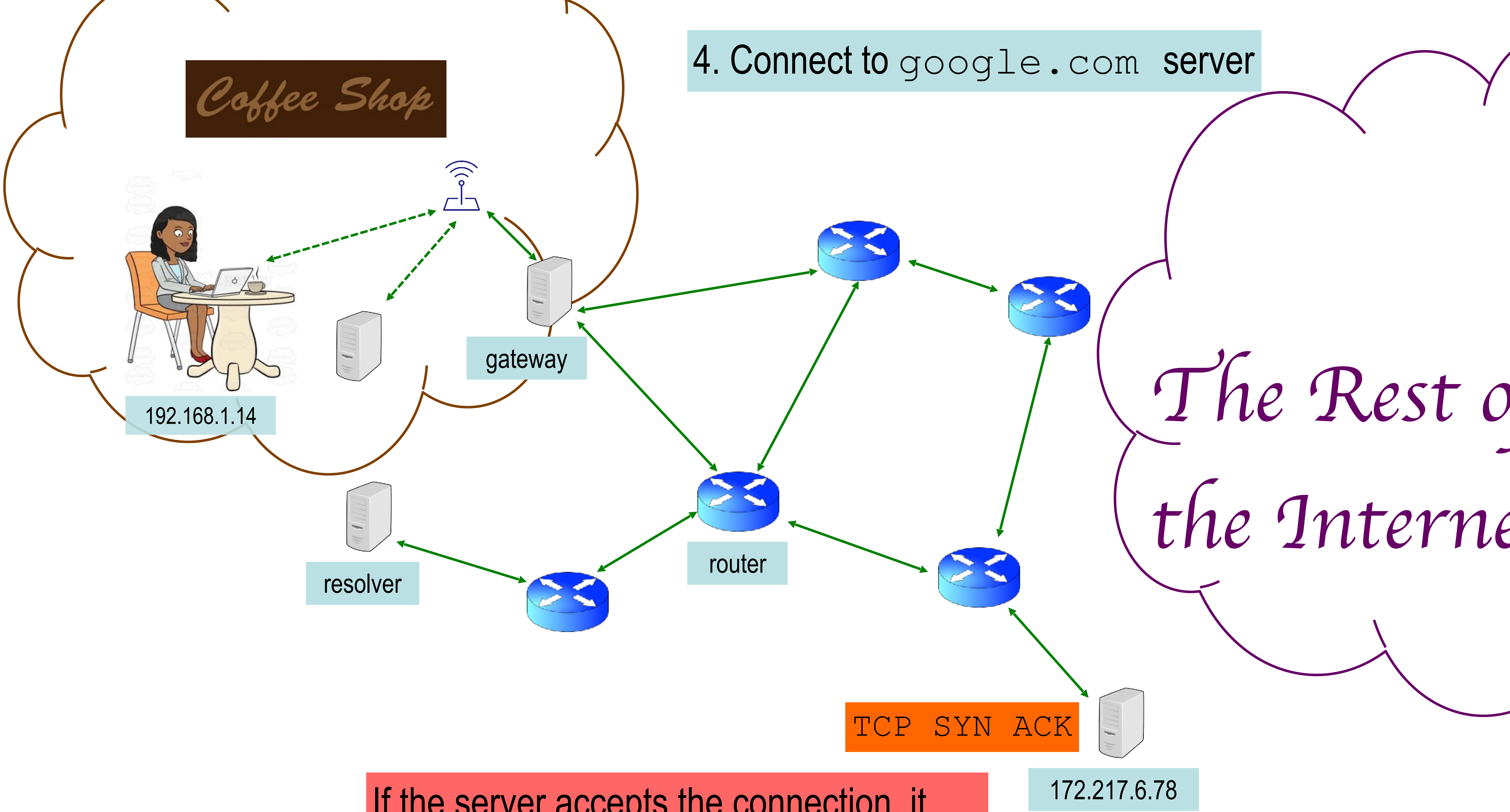
Your laptop now ***establishes a connection*** with the web server at `172.217.6.78`. It uses **TCP** for this rather than UDP, to obtain reliability.

4. Connect to `google.com` server



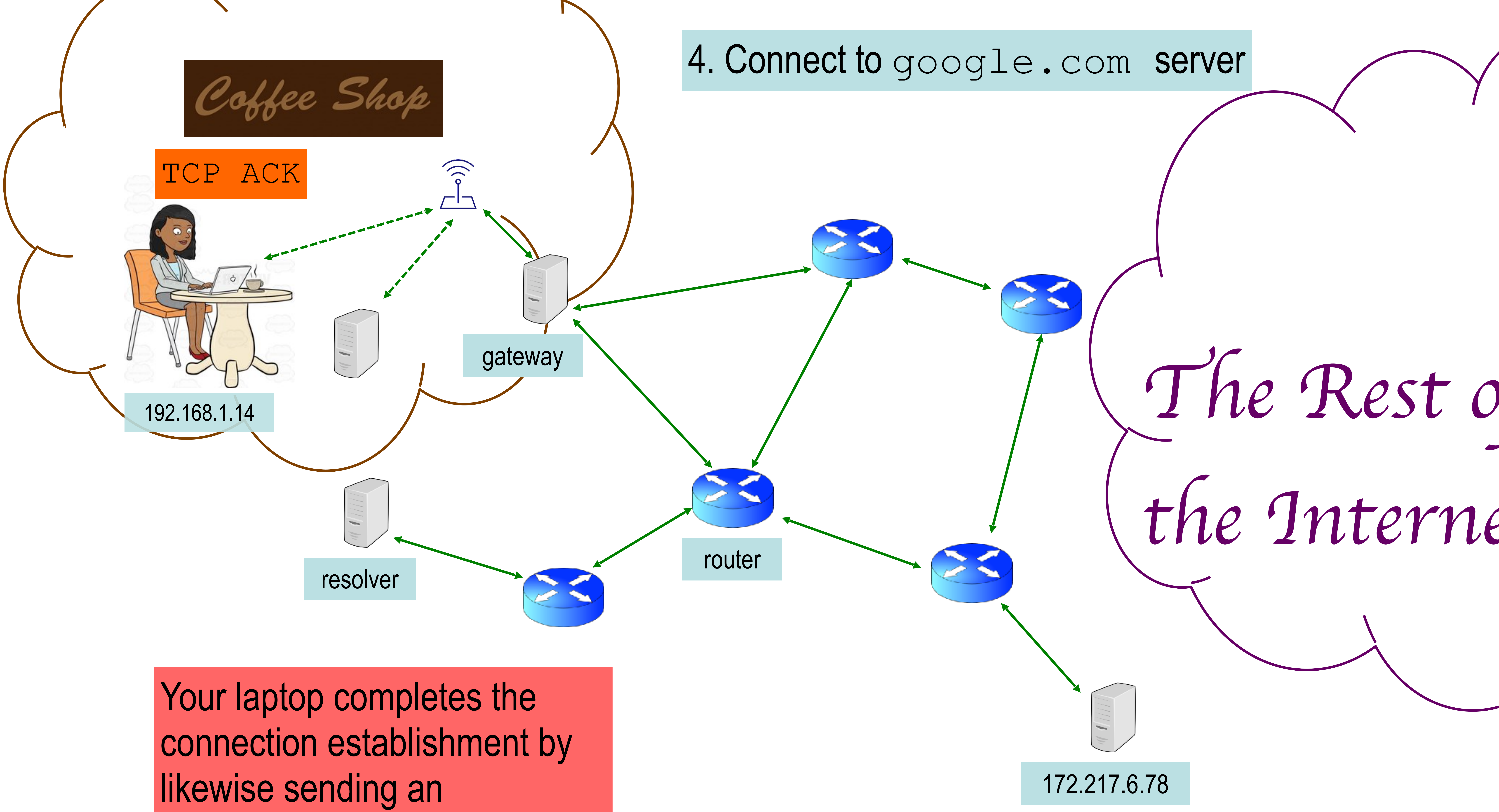
The first step of establishing the connection is to send a TCP connection request ("SYN") to the server.

4. Connect to `google.com` server



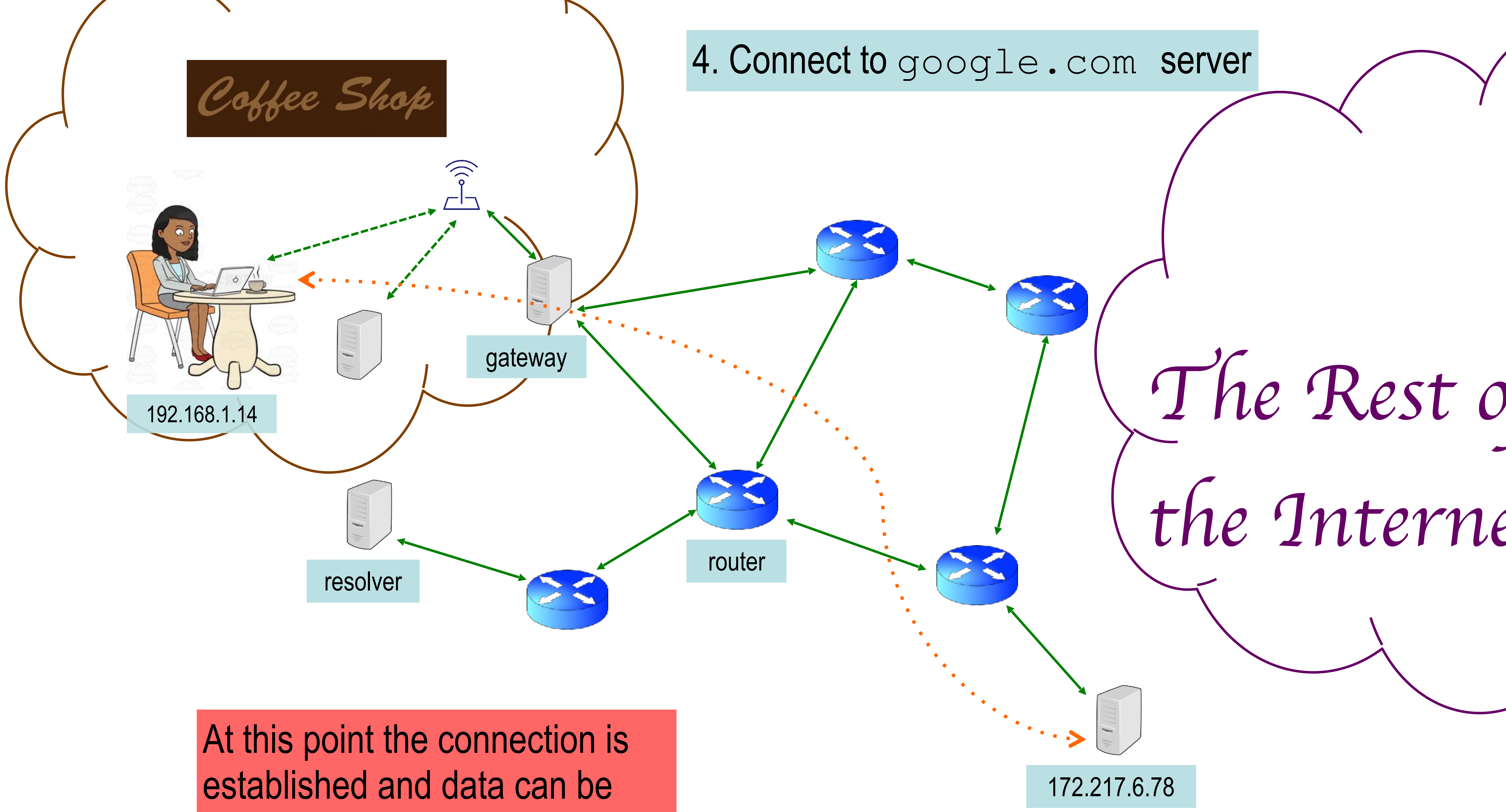
If the server accepts the connection, it replies with a “SYN ACK”.

4. Connect to `google.com` server

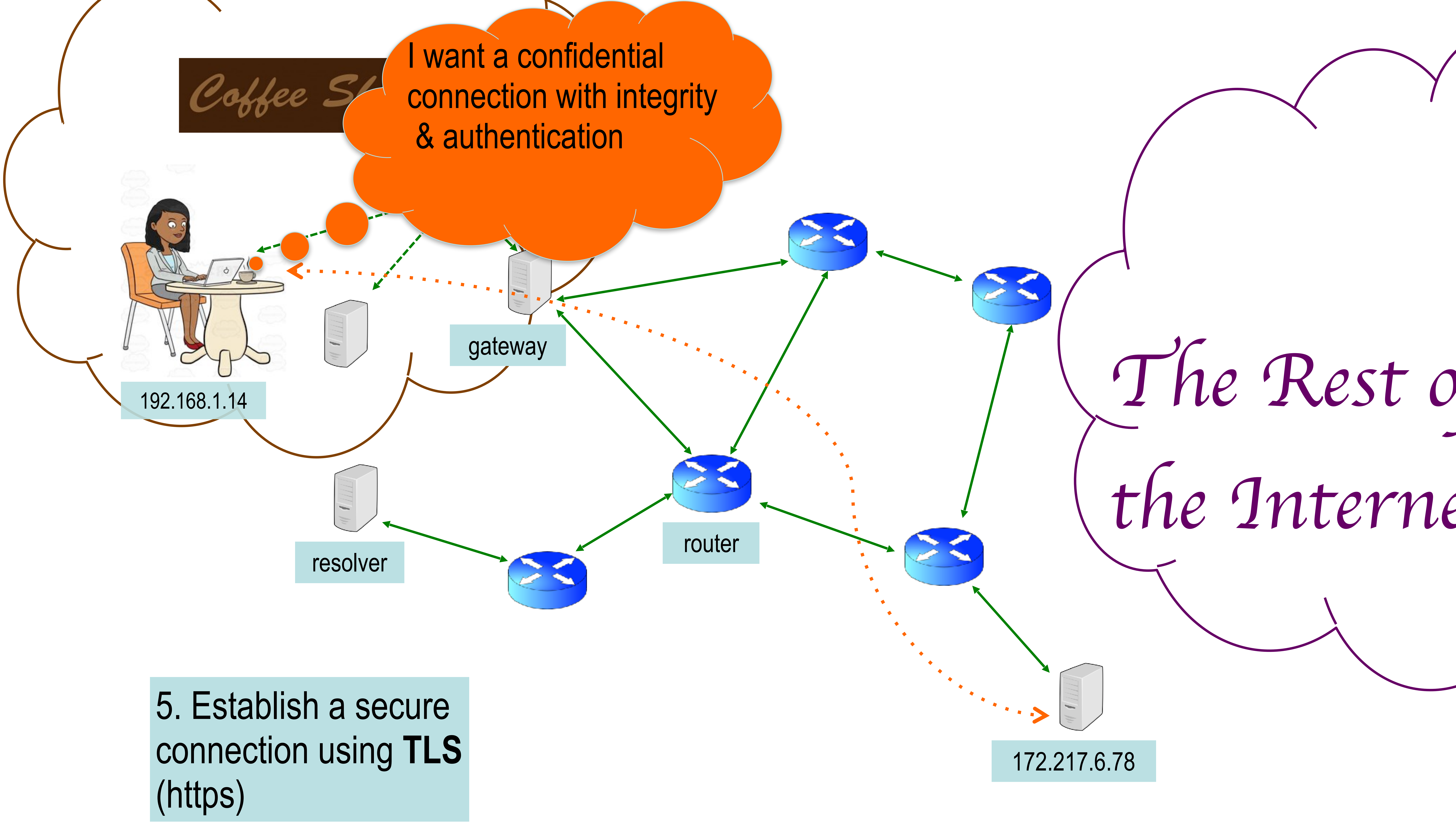


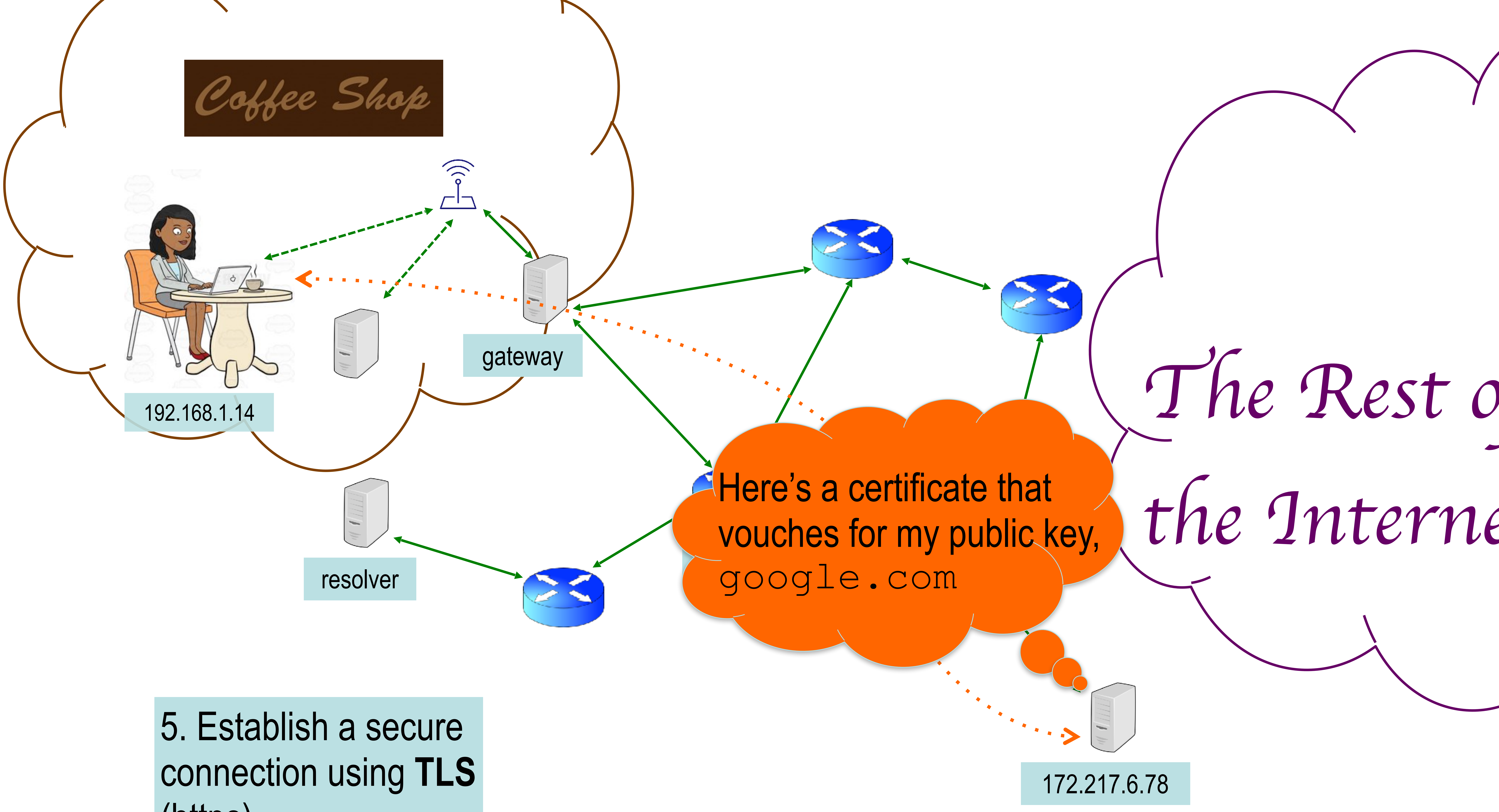
Your laptop completes the connection establishment by likewise sending an acknowledgement.

4. Connect to `google.com` server

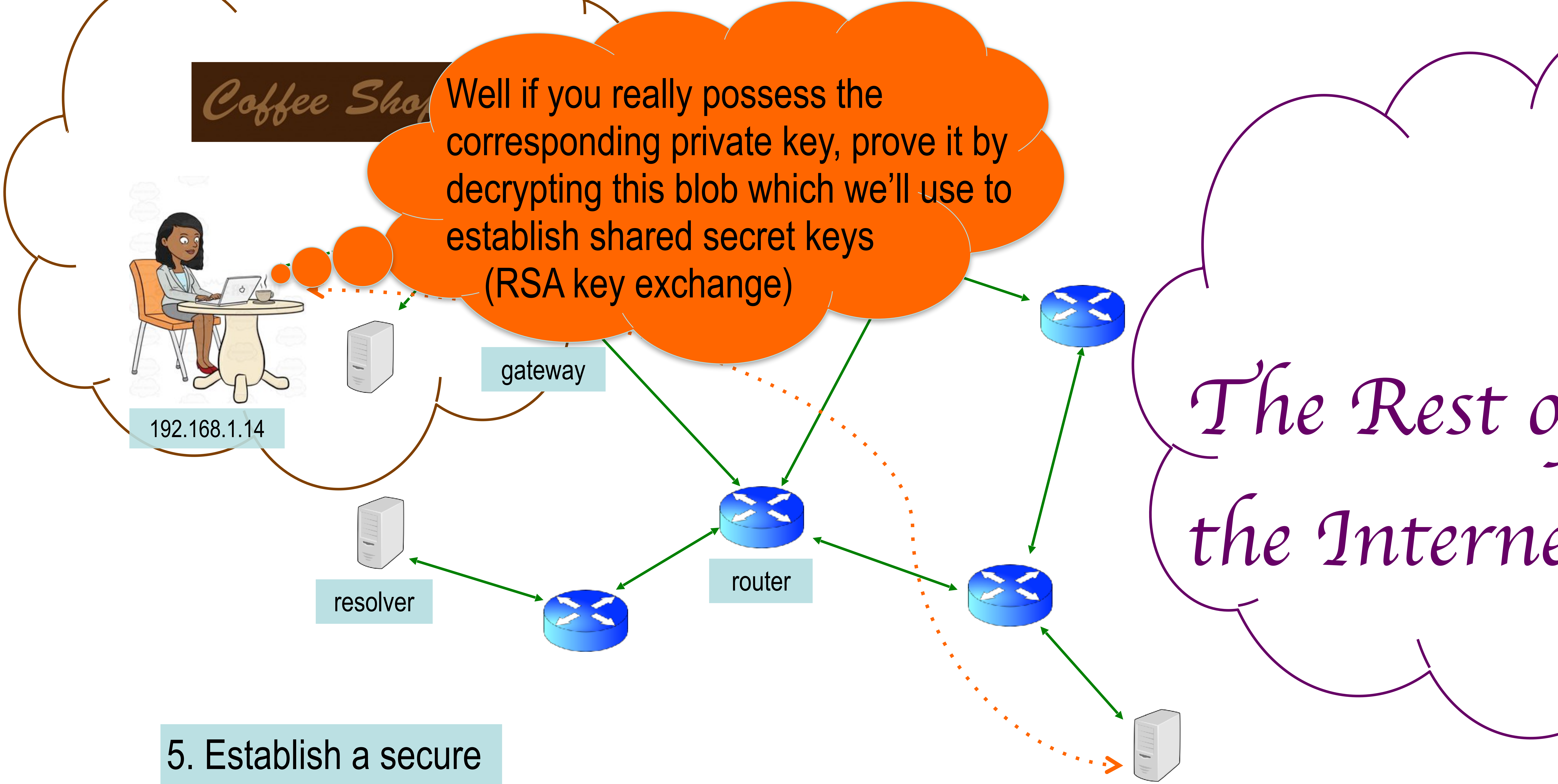


At this point the connection is established and data can be (reliably) exchanged.

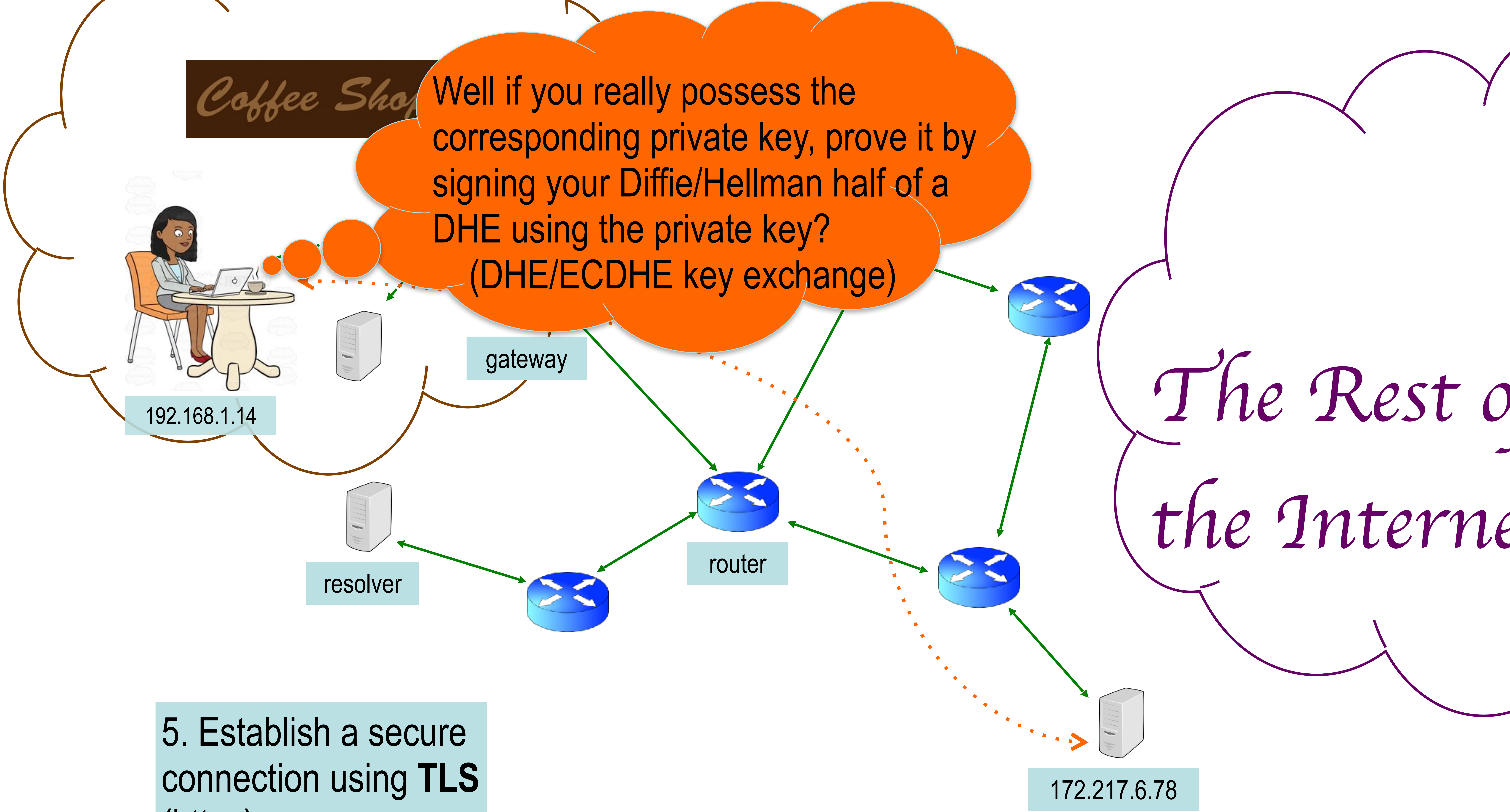

Coffee Sh
I want a confidential connection with integrity & authentication
gateway
192.168.1.14
resolver
router
172.217.6.78
The Rest of
the Interne
5. Establish a secure connection using **TLS** (https)



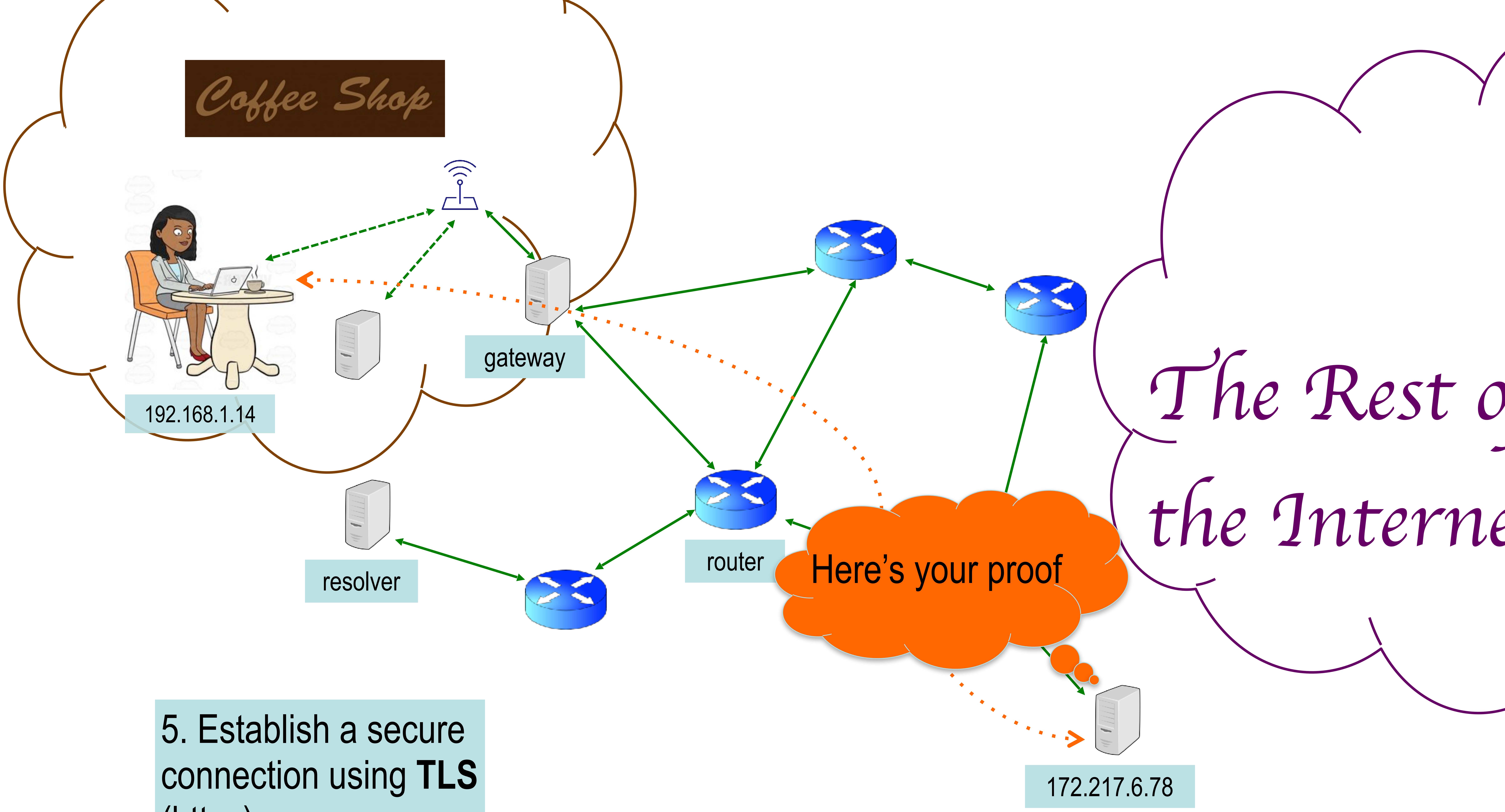
5. Establish a secure connection using **TLS** (https)



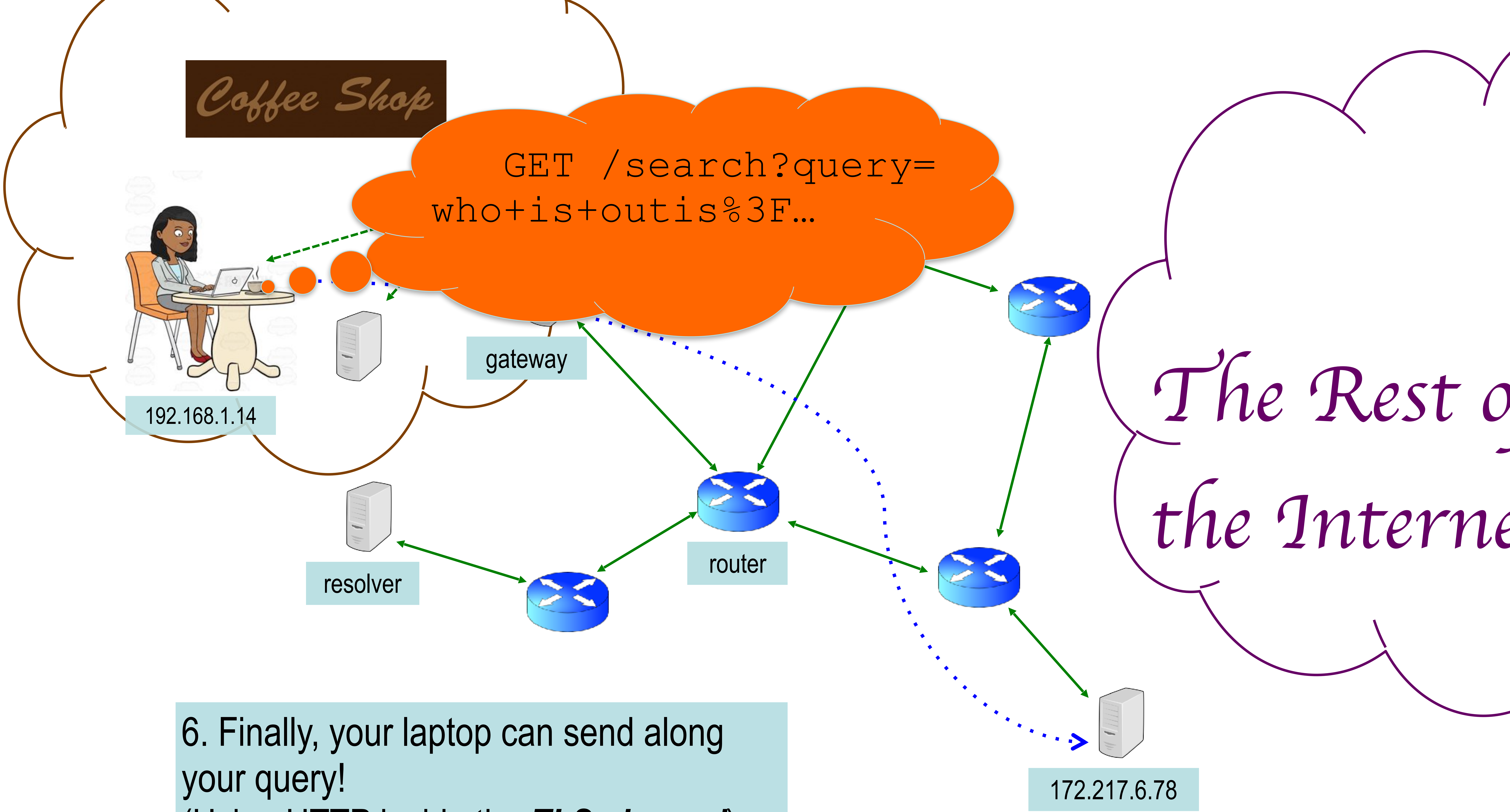
5. Establish a secure connection using **TLS** (https)



5. Establish a secure connection using **TLS** (https)



5. Establish a secure connection using **TLS** (https)



6. Finally, your laptop can send along your query!
(Using HTTP inside the ***TLS channel***)

IP addresses

- IPv4 addresses are 32 bits
  - `aa.bb.cc.dd`
    - Decimal values from 0-255, e.g. `128.32.131.12`
- IPv6 addresses are 128 bits
  - `aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh`
    - Hexadecimal values (can drop leading 0), e.g. `2607:f140:2000:4001:187f:86cc:3dfc:b9c8`
    - A long run of 0s can be replaced with ::
- Subnets (/8, /16, /24...)
  - `128.32/16`
    - All IPv4 between `128.32.0.0` and `128.32.255.255`
  - `2607:f140:2000:4001/64`
    - All IPv6 addresses with the same upper 64 bits

Special IP addresses & Networks

- Localhost: **`127.0.0/24`**
- Broadcast: **`255.255.255.255`**
  - Send to all in the local network
  - Also for subnet, can specify all bits as 1 (e.g. for **`128.32/16`**, **`128.32.255.255`**) to broadcast to that network, but generally ignored these days
- Private: **`10/8`**, **`172.16/12`** (ends up being .16-.32), **`192.168/16`**
  - Not routed on the Internet, can use for internal purposes
  - Commonly used for NAT (more later)
- IPv6 Multicast: **`ff00:/8`**
  - In particular **`ff00::1`** -> all machines on local network

Ethernet

- 6 bytes of destination MAC address
  - In this case, MAC means media access control address, not message authentication code!
- 6 bytes of source MAC address
- Optional 4-byte VLAN tag
- 2-byte length/type field
- 46-1500 bytes of payload

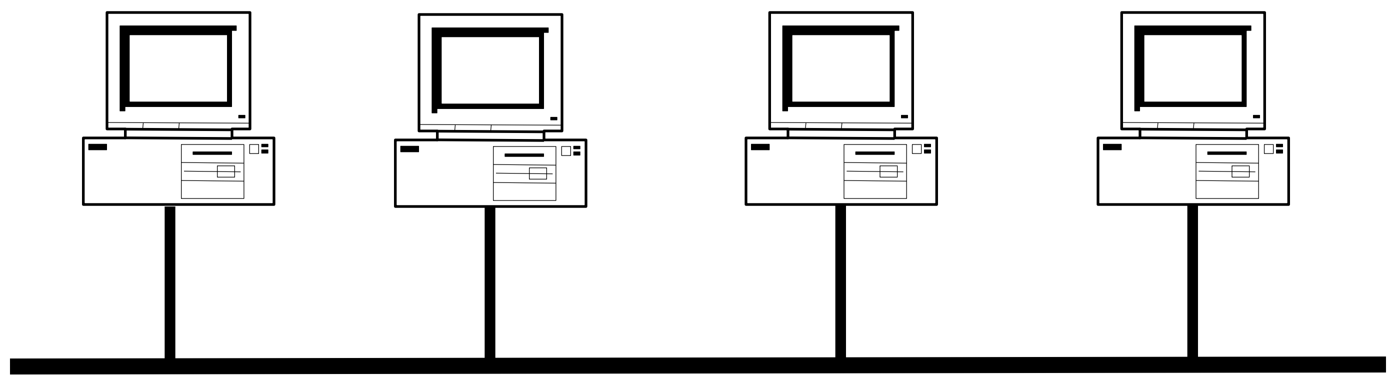
DST MAC	SRC MAC	VLAN	Type	PAYLOAD

The MAC Address

- The MAC address acts as a device identifier
  - Usually written as 6 bytes in hex, e.g. `13:37:ca:fe:f0:0d`
- A device ***should ignore*** all packets that aren't to itself or to the broadcast address (`ff:ff:ff:ff:ff:ff`)
  - But almost all devices can go into ***promiscuous mode***
    - This is also known as "sniffing traffic"
- A device generally should only send with its own address
  - But this is enforced with software and can be trivially bypassed when you need to write "raw packets"

Attacks

Link-layer threats

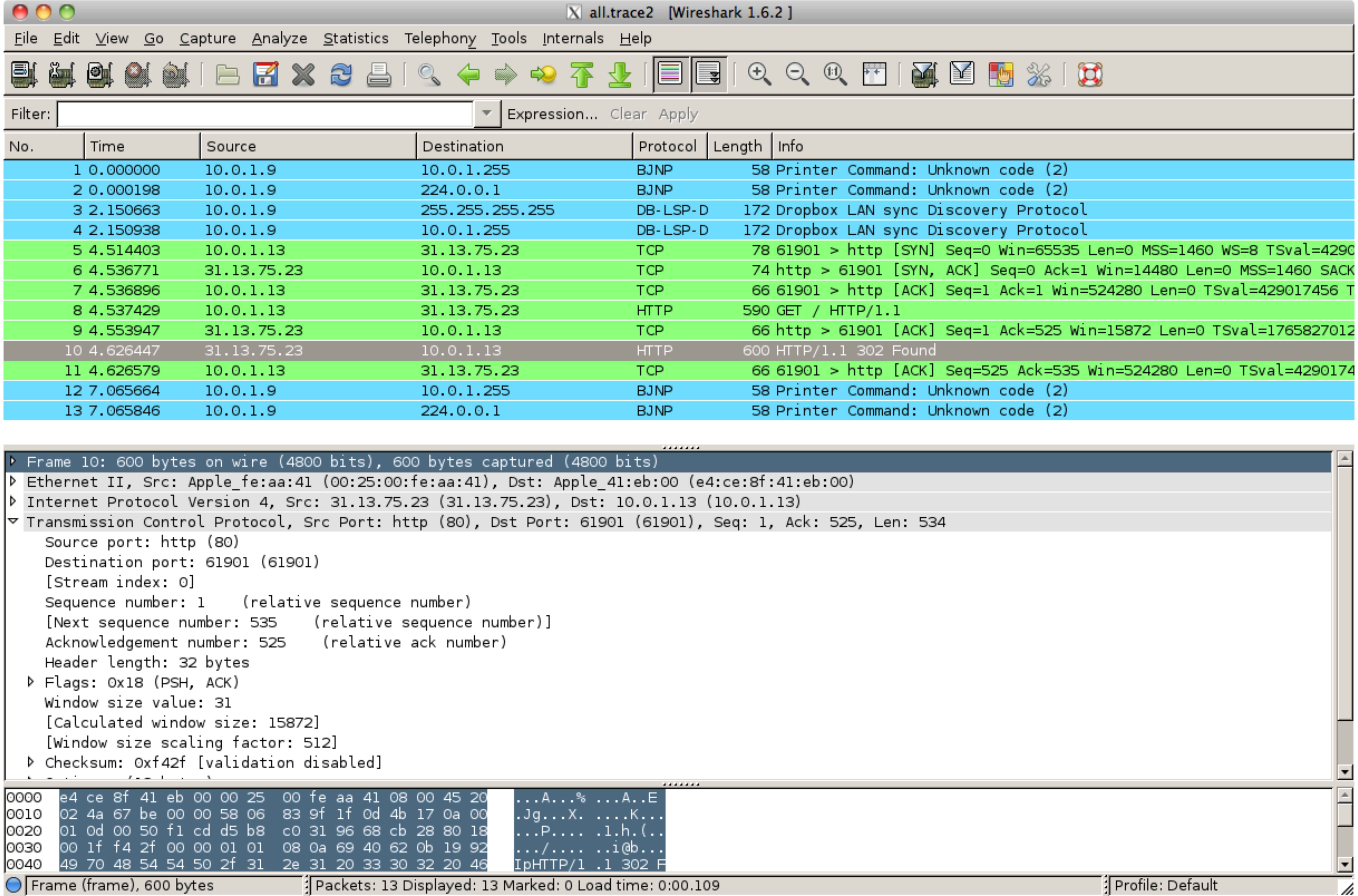


- Confidentiality: eavesdropping (aka sniffing)
- Integrity: injection of spoofed packets
- Availability: delete legit packets (e.g., jamming)

Eavesdropping

- For subnets using broadcast technologies (e.g., WiFi, some types of Ethernet), attacker can eavesdrop
  - Each attached system's NIC (= Network Interface Card) can capture any communication on the subnet
  - Tools: tcpdump / windump (low-level text-based printout), wireshark (GUI)

Wireshark



all.trace2 [Wireshark 1.6.2]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Length Info
1 0.000000 10.0.1.9 10.0.1.255 BJNP 58 Printer Command: Unknown code (2)
2 0.000198 10.0.1.9 224.0.0.1 BJNP 58 Printer Command: Unknown code (2)
3 2.150663 10.0.1.9 255.255.255.255 DB-LSP-D 172 Dropbox LAN sync Discovery Protocol
4 2.150938 10.0.1.9 10.0.1.255 DB-LSP-D 172 Dropbox LAN sync Discovery Protocol
5 4.514403 10.0.1.13 31.13.75.23 TCP 78 61901 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=4290
6 4.536771 31.13.75.23 10.0.1.13 TCP 74 http > 61901 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK
7 4.536896 10.0.1.13 31.13.75.23 TCP 66 61901 > http [ACK] Seq=1 Ack=1 Win=524280 Len=0 TSval=429017456 T
8 4.537429 10.0.1.13 31.13.75.23 HTTP 590 GET / HTTP/1.1
9 4.553947 31.13.75.23 10.0.1.13 TCP 66 http > 61901 [ACK] Seq=1 Ack=525 Win=15872 Len=0 TSval=1765827012
10 4.626447 31.13.75.23 10.0.1.13 HTTP 600 HTTP/1.1 302 Found
11 4.626579 10.0.1.13 31.13.75.23 TCP 66 61901 > http [ACK] Seq=525 Ack=535 Win=524280 Len=0 TSval=4290174
12 7.065664 10.0.1.9 10.0.1.255 BJNP 58 Printer Command: Unknown code (2)
13 7.065846 10.0.1.9 224.0.0.1 BJNP 58 Printer Command: Unknown code (2)
Frame 10: 600 bytes on wire (4800 bits), 600 bytes captured (4800 bits)
Ethernet II, Src: Apple_fe:aa:41 (00:25:00:fe:aa:41), Dst: Apple_41:eb:00 (e4:ce:8f:41:eb:00)
Internet Protocol Version 4, Src: 31.13.75.23 (31.13.75.23), Dst: 10.0.1.13 (10.0.1.13)
Transmission Control Protocol, Src Port: http (80), Dst Port: 61901 (61901), Seq: 1, Ack: 525, Len: 534
Source port: http (80)
Destination port: 61901 (61901)
[Stream index: 0]
Sequence number: 1 (relative sequence number)
[Next sequence number: 535 (relative sequence number)]
Acknowledgement number: 525 (relative ack number)
Header length: 32 bytes
Flags: 0x18 (PSH, ACK)
Window size value: 31
[Calculated window size: 15872]
[Window size scaling factor: 512]
Checksum: 0xf42f [validation disabled]
0000 e4 ce 8f 41 eb 00 00 25 00 fe aa 41 08 00 45 20 ...A...% ...A..E
0010 02 4a 67 be 00 00 58 06 83 9f 1f 0d 4b 17 0a 00 .Jg...X. ....K...
0020 01 0d 00 50 f1 cd d5 b8 c0 31 96 68 cb 28 80 18 ...P.... .1.h.(..
0030 00 1f f4 2f 00 00 01 01 08 0a 69 40 62 0b 19 92 .../.... ..i@b...
0040 49 70 48 54 54 50 2f 31 2e 31 20 33 30 32 20 46 IpHTTP/1 .1 302 F
Frame (frame), 600 bytes
Packets: 13 Displayed: 13 Marked: 0 Load time: 0:00.109
Profile: Default

Operation Ivy Bells

By Matthew Carle
Military.com

At the beginning of the 1970's, divers from the specially-equipped submarine, USS Halibut (SSN 587), left their decompression chamber to start a bold and dangerous mission, code named "Ivy Bells".

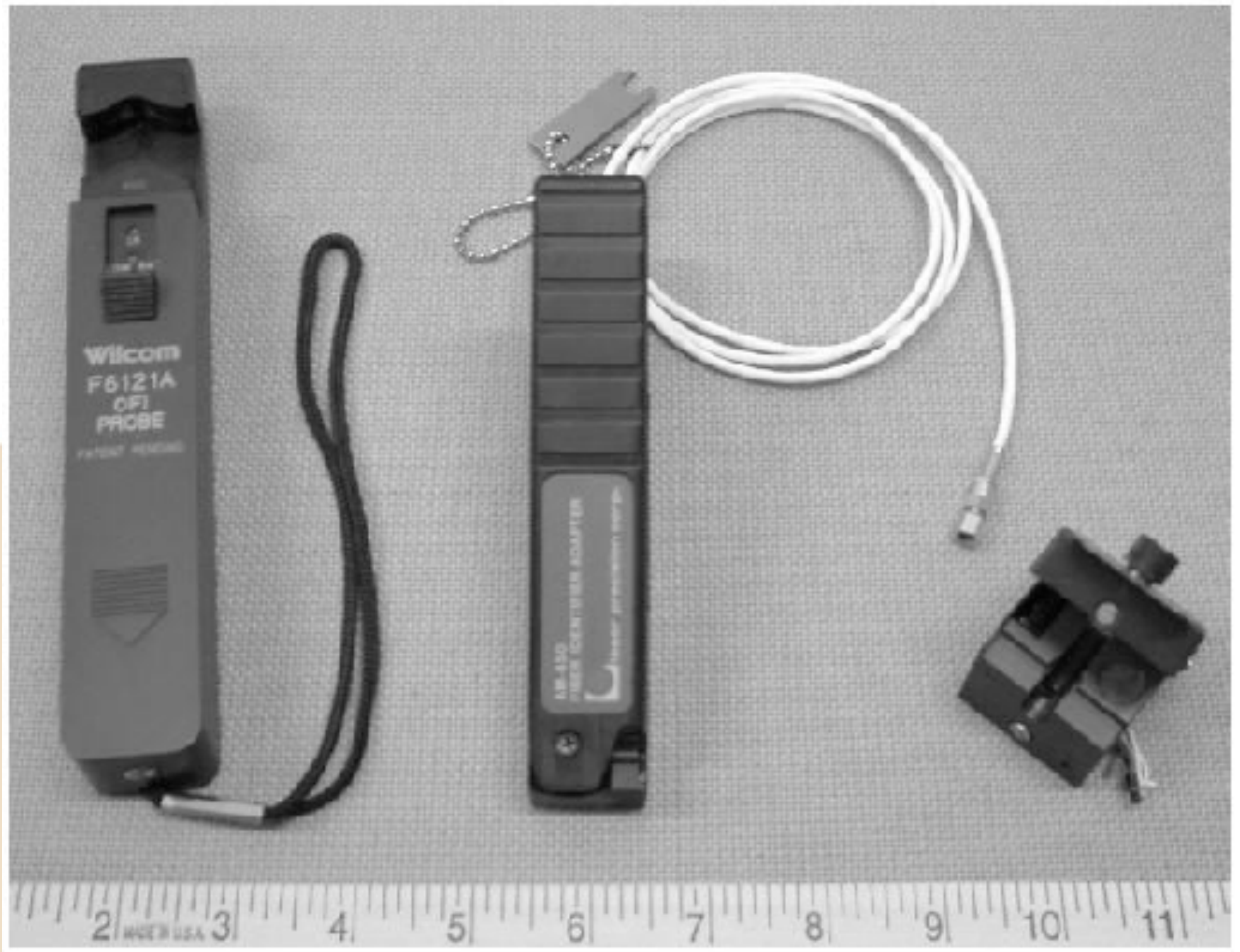
The Regulus guided missile submarine, USS Halibut (SSN 587) which carried out Operation Ivy Bells.

In an effort to alter the balance of Cold War, these men scoured the ocean floor for a five-inch diameter cable carry secret Soviet communications between military bases.

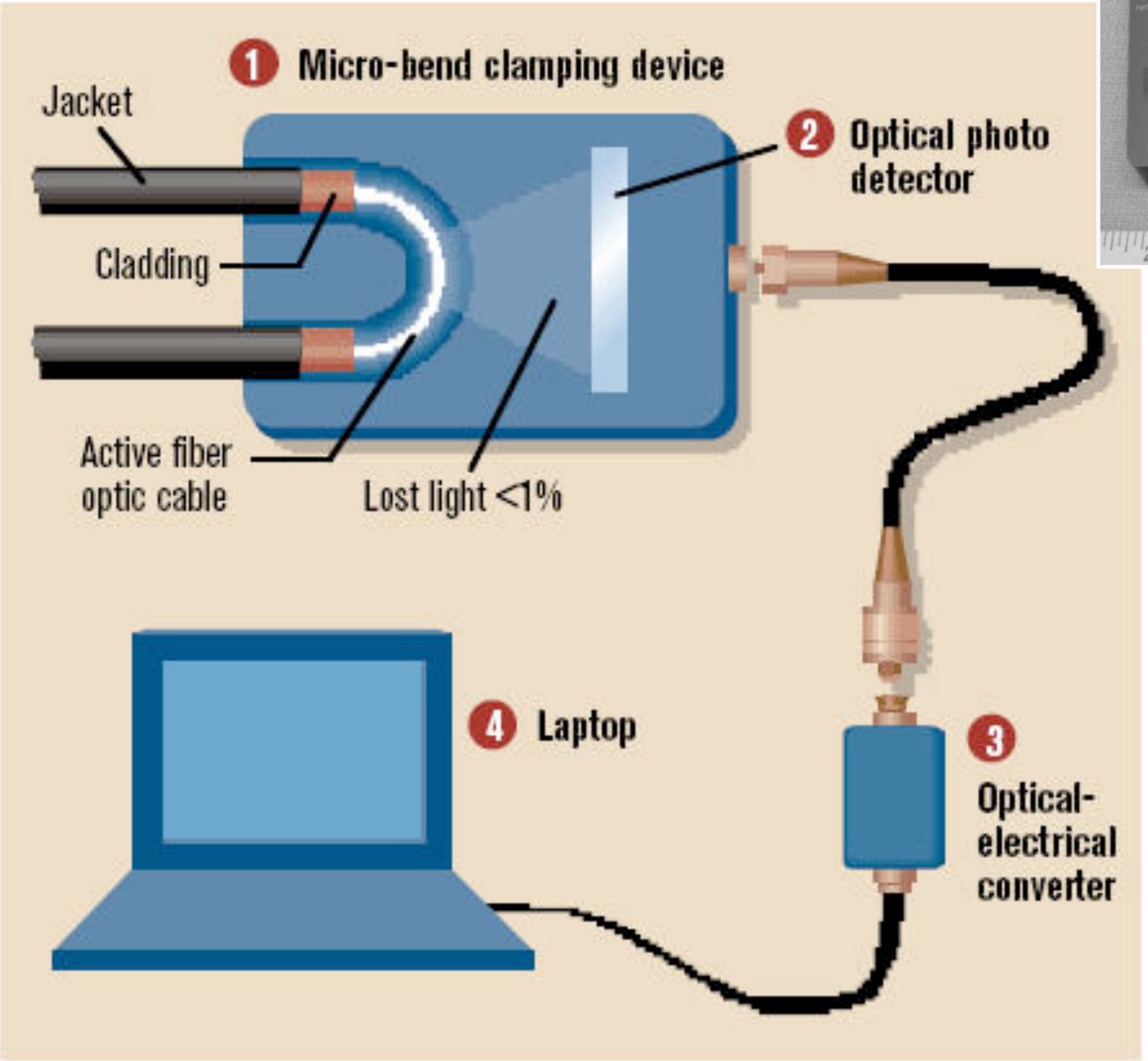
The divers found the cable and installed a 20-foot long listening device on the cable. designed to attach to the cable without piercing the casing, the device recorded all communications that occurred. If the cable malfunctioned and the Soviets raised it for repair, the bug, by design, would fall to the bottom of the ocean. Each month Navy divers retrieved the recordings and installed a new set of tapes.

Upon their return to the United States, intelligence agents from the NSA analyzed the recordings and tried to decipher any encrypted information. The Soviets apparently were confident in the security of their communications lines, as a surprising amount of sensitive information traveled through the lines without encryption.

prison. The original tap that was discovered by the Soviets is now on exhibit at the KGB museum in Moscow.



Wilcom
F6121A
OFI
PROBE



1 Micro-bend clamping device
Jacket
Cladding
Active fiber optic cable
Lost light <1%
2 Optical photo detector
3 Optical-electrical converter
4 Laptop

Link-Layer Spoofing

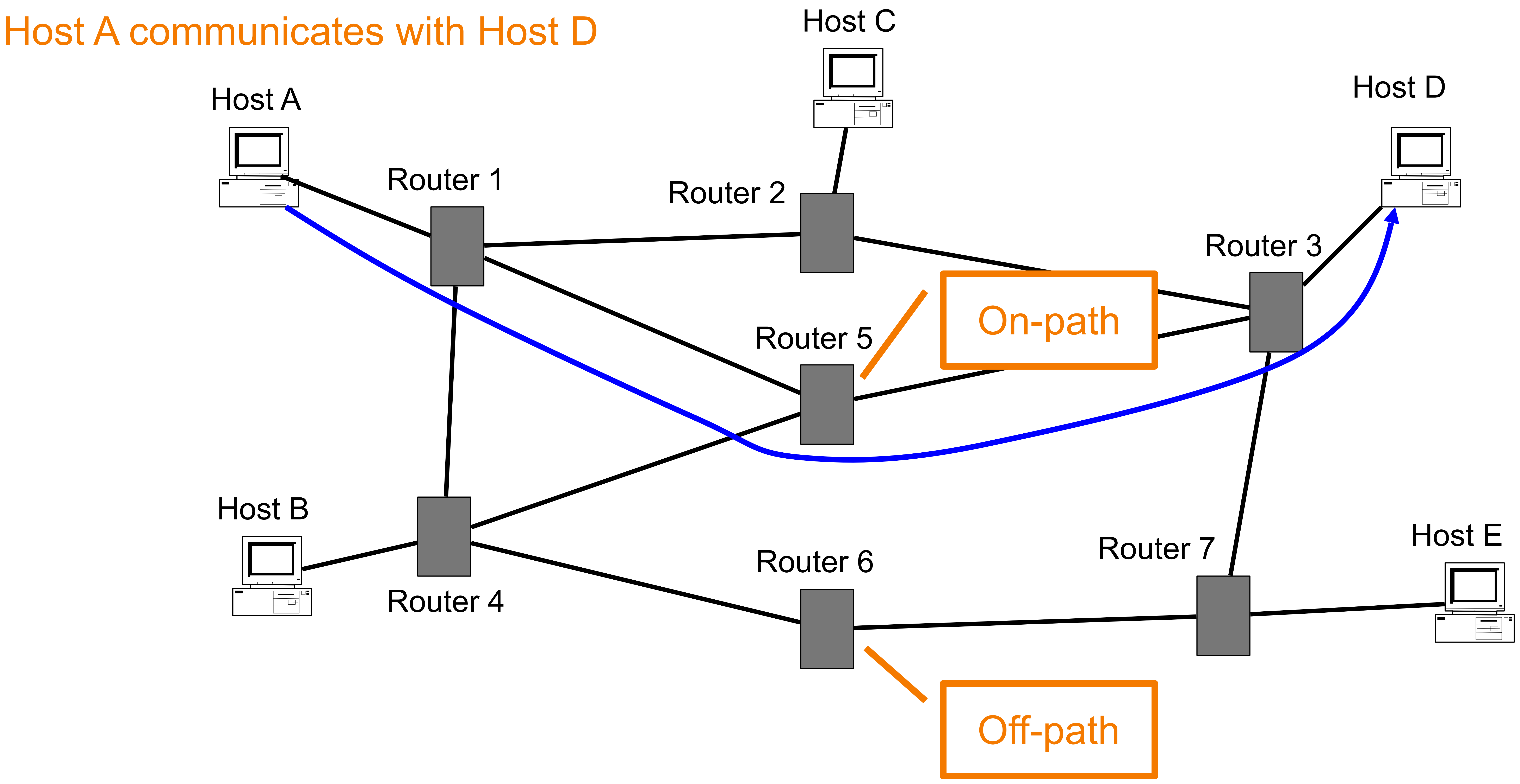
- Attacker can inject spoofed packets, and lie about the source address
-

M	C	Hello world!

Spoofing

- With physical access to a local network, attacker can create any packet they like
  - Spoofing = lie about source address
- Particularly powerful when combined with eavesdropping, because attacker can understand exact state of victim's communication and craft their spoofed traffic to match it
  - Spoofing w/o eavesdropping = blind spoofing

On-path vs Off-path Spoofing



Spoofing on the Internet

- On-path attackers can see victim's traffic ⇒ spoofing is easy
- Off-path attackers can't see victim's traffic
  - They have to resort to blind spoofing
  - Often must guess/infer header values to succeed
    - We then care about work factor: how hard is this
  - But sometimes they can just brute force
    - E.g., 16-bit value: just try all 65,536 possibilities!
- When we say an attacker "can spoof", we usually mean "w/ reasonable chance of success"

DNS Service

- Runs Domain Name Servers
- Translates domain names google.com to IP addresses
- When user browser wants to contact google.com, it first contacts a DNS to find out the IP address for google.com and then sends a packet to that IP address
- More in future lectures..