Network Security: Background

CS 161: Computer Security

Prof. Vern Paxson

TAs: Paul Bramsen, Apoorva Dornadula,
David Fifield, Mia Gil Epner, David Hahn, Warren He,
Grant Ho, Frank Li, Nathan Malkin, Mitar Milutinovic,
Rishabh Poddar, Rebecca Portnoff, Nate Wang

http://inst.eecs.berkeley.edu/~cs161/

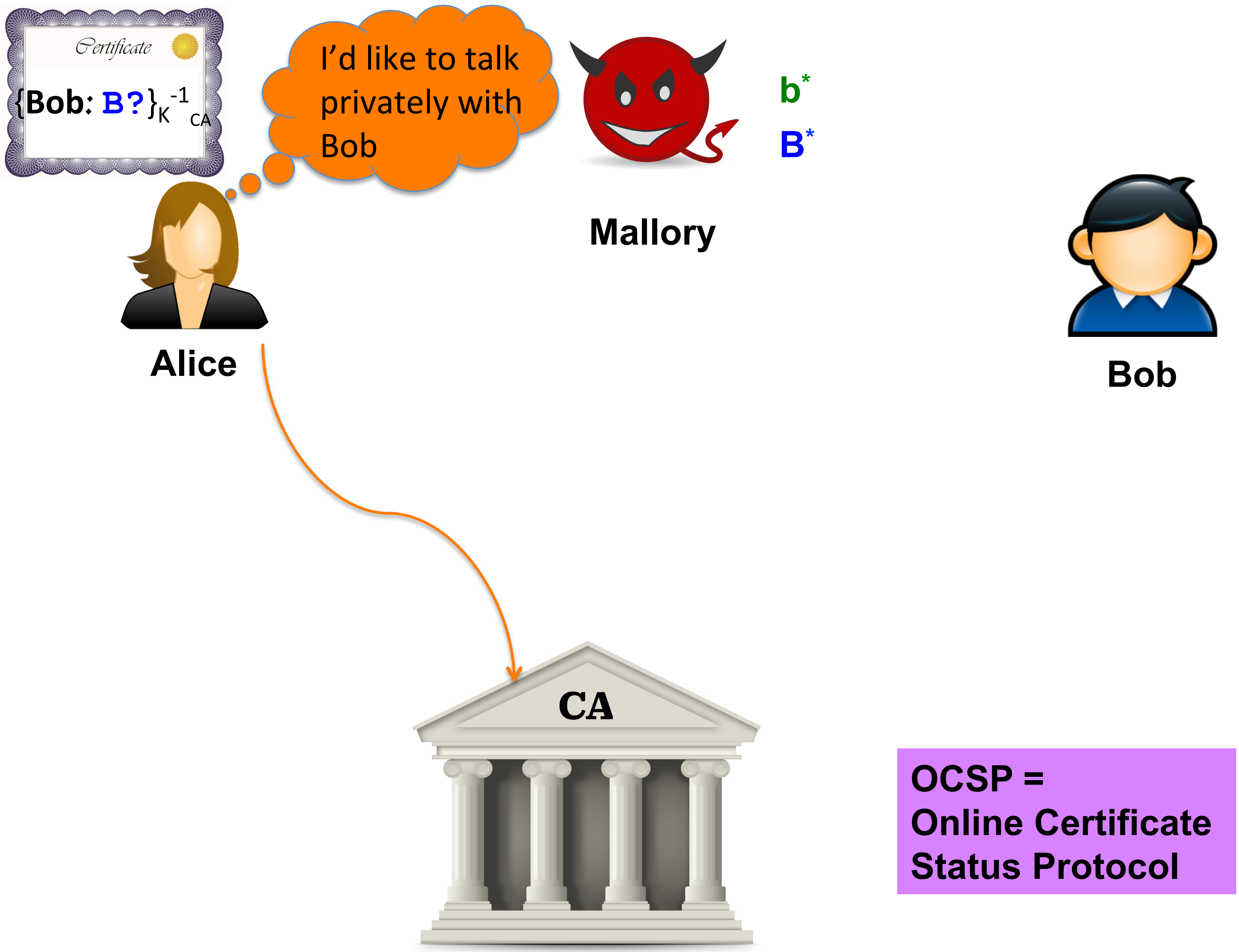
March 7, 2017

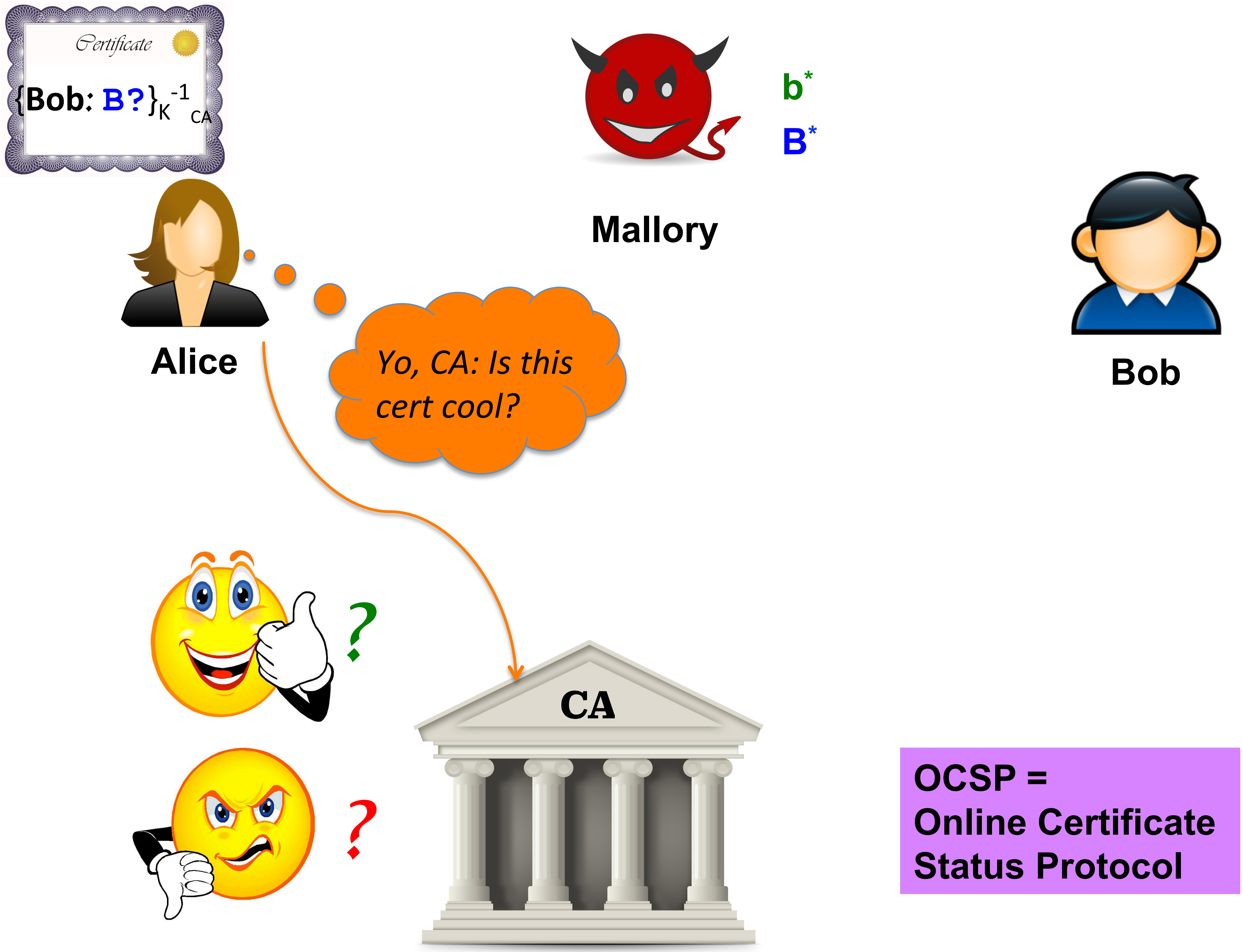
Revocation, con't

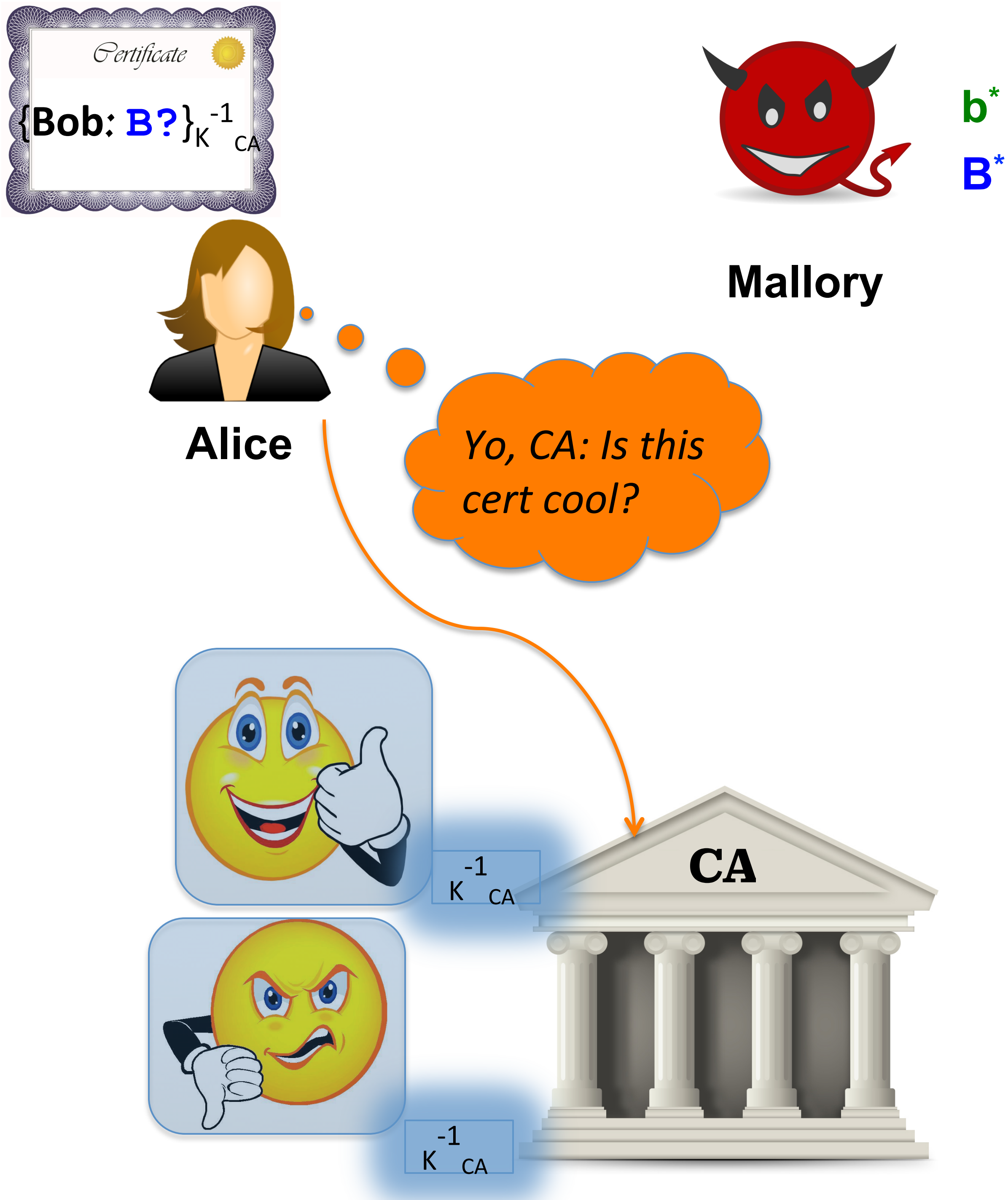
- Approach #2: announce revoked certs
  - Users periodically download *cert revocation list* (CRL)
- Issues?
  - Lists can get large
  - Need to *authenticate the list* itself – how? Sign it!
  - Mallory can exploit download lag
  - What does Alice do if can't reach CA for download?
    1. Assume all certs are invalid (*fail-safe defaults*)
       - Wow, what an unhappy failure mode!
    2. Use old list: widens exploitation window if Mallory can "DoS" CA (DoS = denial-of-service)

Revocation, con't

- Approach #3: CA provides service to query
  - OCSP: *Online Certificate Status Protocol*


Certificate
{Bob: B?}$_{K^{-1}_{CA}}$
I'd like to talk privately with Bob
Mallory
b*
B*
Alice
Bob
CA
OCSP =
Online Certificate Status Protocol


Certificate
{Bob: B?}$_{K^{-1}_{CA}}$
Mallory
b*
B*
Alice
Yo, CA: Is this cert cool?
Bob
?
?
CA
OCSP =
Online Certificate
Status Protocol

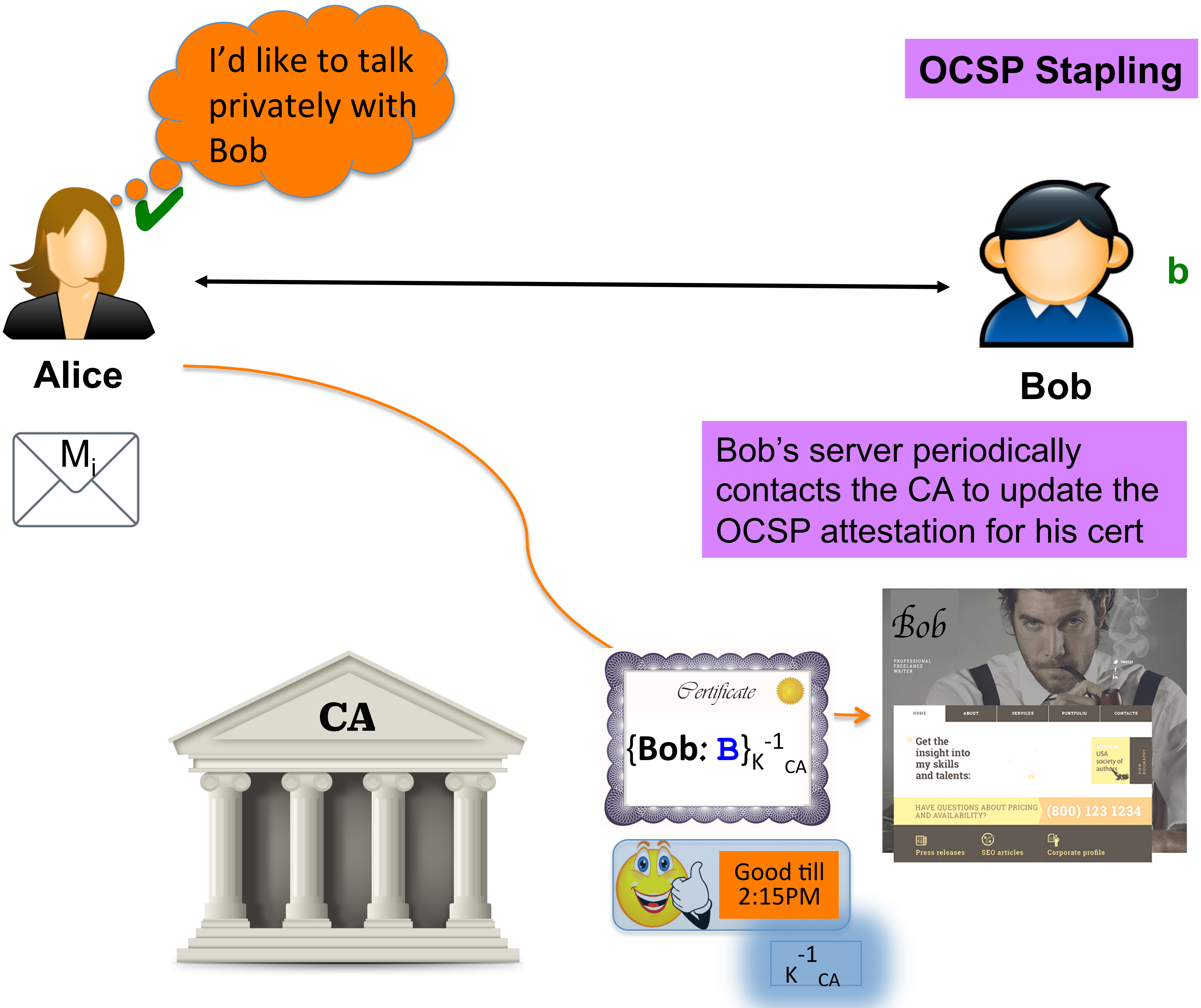


OCSP =
Online Certificate Status Protocol

Revocation, con't

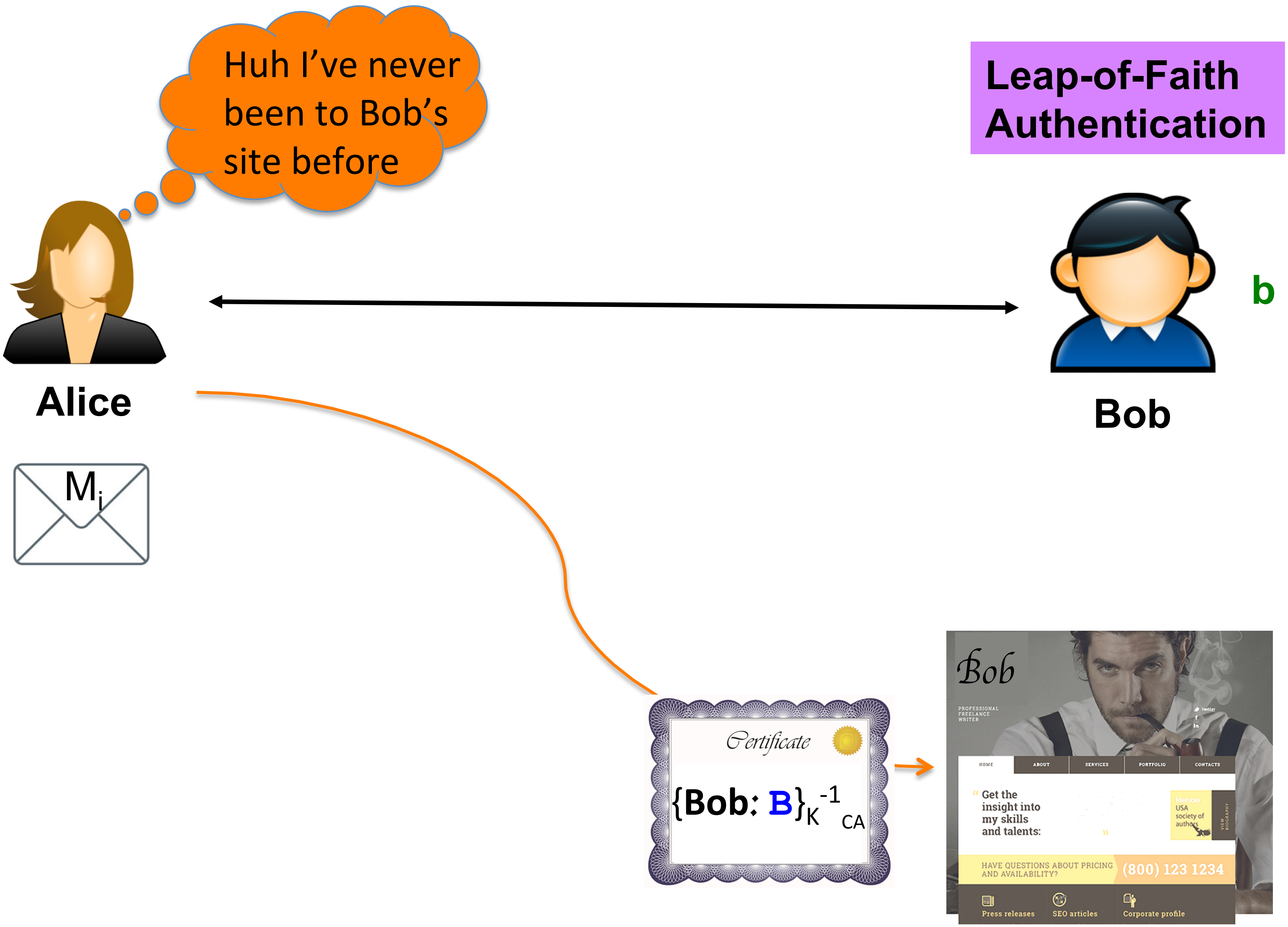
- Approach #3: CA provides service to query
  - OCSP: *Online Certificate Status Protocol*

- Issues?
  - Can't be used if Alice doesn't have connectivity to CA
  - CA learns that Alice talks to Bob
  - CA had better build this in a scalable fashion!
  - CA outages ⇒ big headaches
    - OR: Alice defaults to trusting if OCSP inaccessible
      - Again creates a DoS threat


OCSP Stapling
I'd like to talk privately with Bob
Alice
Bob
b
M_i
Bob's server periodically contacts the CA to update the OCSP attestation for his cert
CA
Certificate
{Bob: B}K-1CA
Good till 2:15PM
K-1CA
Bob
PROFESSIONAL FREELANCE WRITER
HOME
ABOUT
SERVICES
PORTFOLIO
CONTACTS
Get the insight into my skills and talents:
HAVE QUESTIONS ABOUT PRICING AND AVAILABILITY?
(800) 123 1234
Press releases
SEO articles
Corporate profile

Leap-of-Faith Authentication

- A completely different approach leverages key continuity



Leap-of-Faith Authentication
Huh I've never been to Bob's site before
Alice
Bob
b
M_i
Certificate
$\{Bob: B\}_{K^{-1}_{CA}}$
Bob
PROFESSIONAL FREELANCE WRITER
Get the insight into my skills and talents:
HAVE QUESTIONS ABOUT PRICING AND AVAILABILITY?
(800) 123 1234
Press releases
SEO articles
Corporate profile

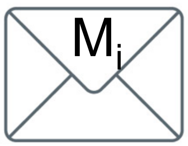
Leap-of-Faith Authentication

Alice

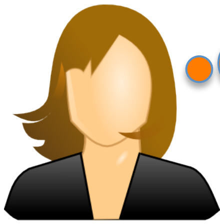
I'm going to **hope** that ***just** this one time*, Mallory didn't show up …

b

Bob



Leap-of-Faith Authentication

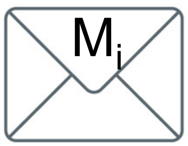


But **now that I have the cert**, any time in the future I'll *refuse a different cert* if offered

b

Alice

Bob



Leap-of-Faith Authentication, con't

- A completely different approach leverages key continuity
  - Also called TOFU: *Trust On First Use*
  - A form of "pinning"
    - Require cert to have specific properties, like particular CA
  - Very popular for SSH
    - Web browsers don't expose an easy equivalent usage model

Leap-of-Faith Authentication, con't

- Properties/Issues?

- Doesn't bug you, just automatically gives you a secure mode of operation
  - Great design property!
- Leverages mental expectations
  - Such as: "hard for attacker to anticipate this'll be my very first visit" (clearly not always true!)
  - Or: "Bob mentioned he'd be upgrading, so the new key is expected"
- Brittle: relies on user to notice and *thoughtfully respond* to key changes

Background on Networking

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: sufficient background in networking to then explore security issues in next ~5 lectures
- Complex topic with many facets
  - We will omit concepts/details that aren't very security-relevant
  - By all means, ask questions when things are unclear
    - (but we may skip if not ultimately relevant for security, or postpone if question itself is directly about security)

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?
  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 You Walk Into A Coffee Shop,

Open Up Your Laptop,

And Issue a Google Query ….

Coffee Shop

Coffee Shop

1. Join the wireless network

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

Coffee Shop

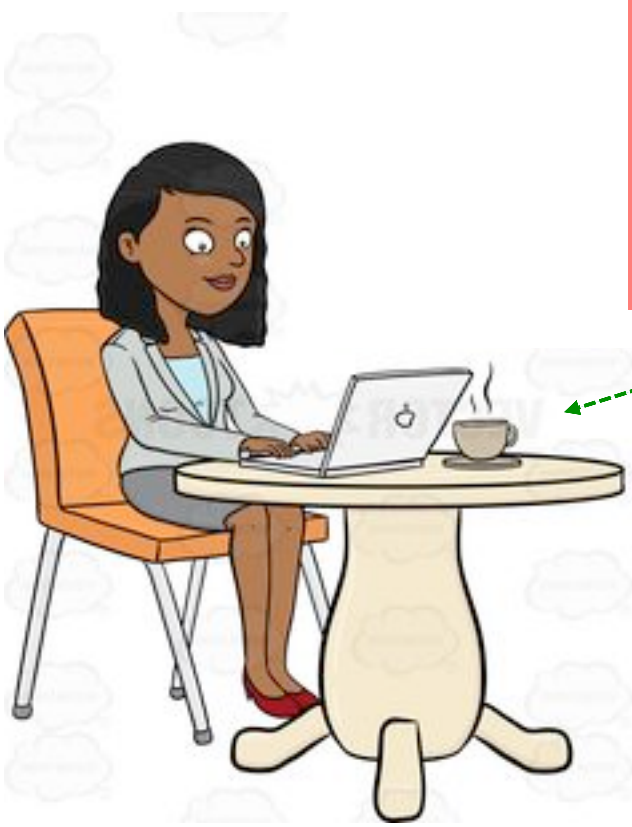
1. Join the wireless network

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.

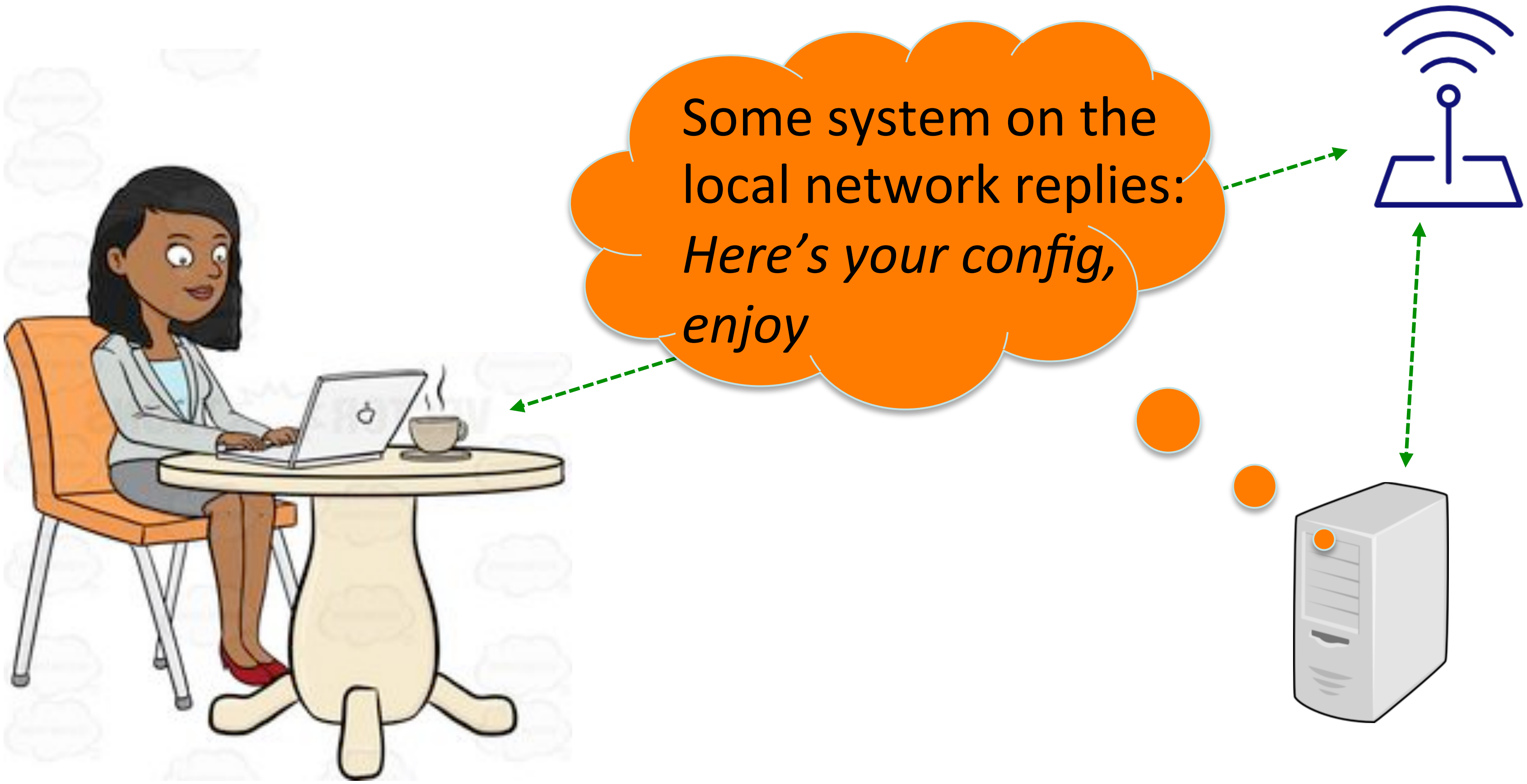


Coffee Shop

2. Configure your connection

Coffee Shop

2. Configure your connection

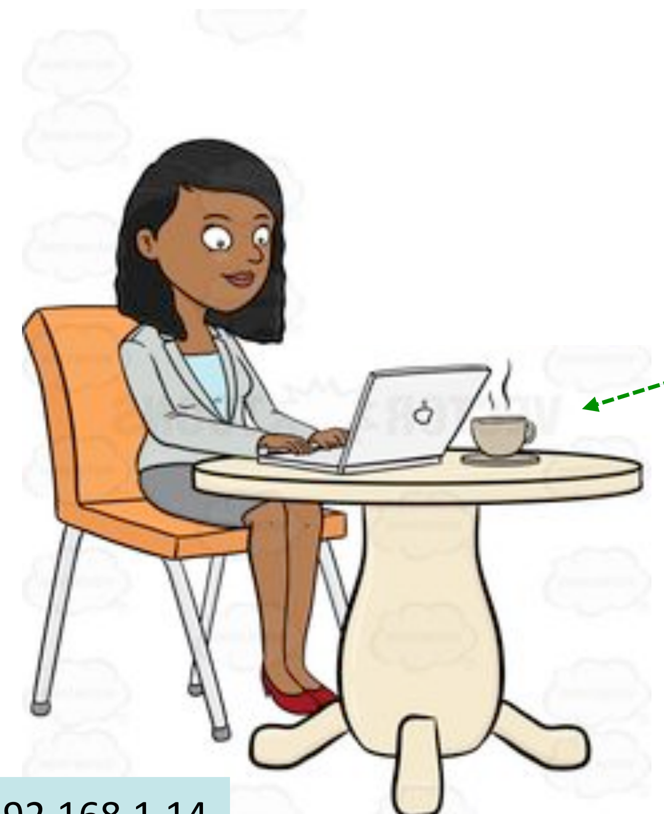


Coffee Shop

2. Configure your connection

The configuration includes:

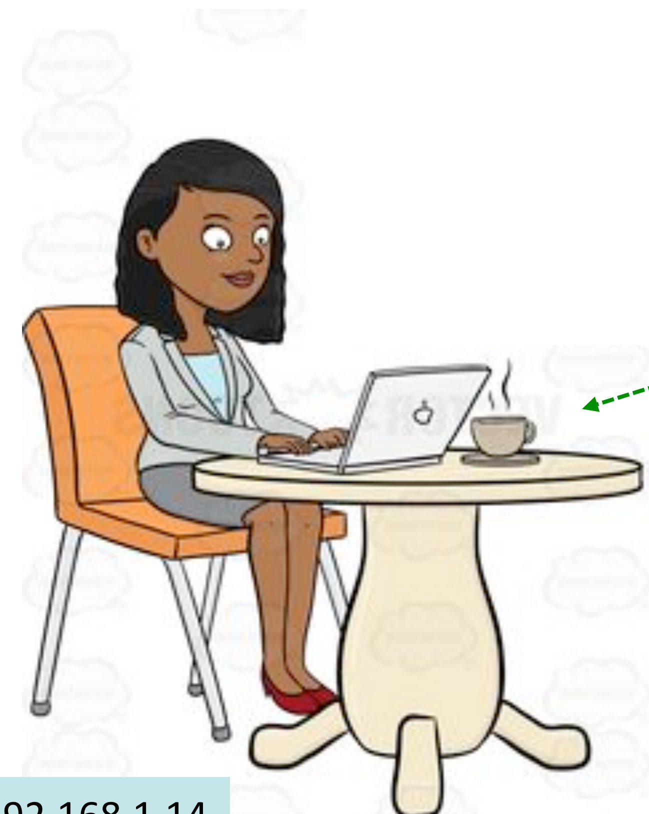
(1) An Internet address (**IP address**) your laptop should use; typ. 32 bits
(2) The address of a "**gateway**" system to use to access *hosts* beyond the local network
(3) The address of a **DNS server** ("*resolver*") to map names like `google.com` to IP addresses



192.168.1.14

Coffee Shop

3. Find the address of google.com

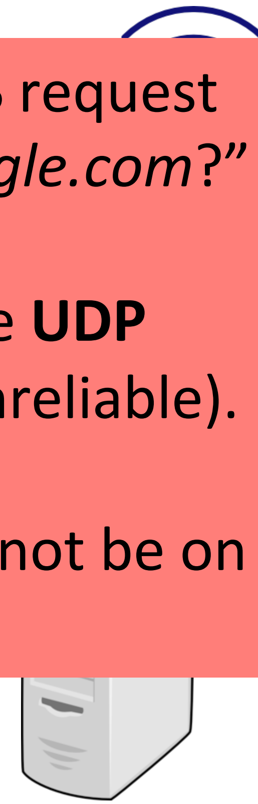


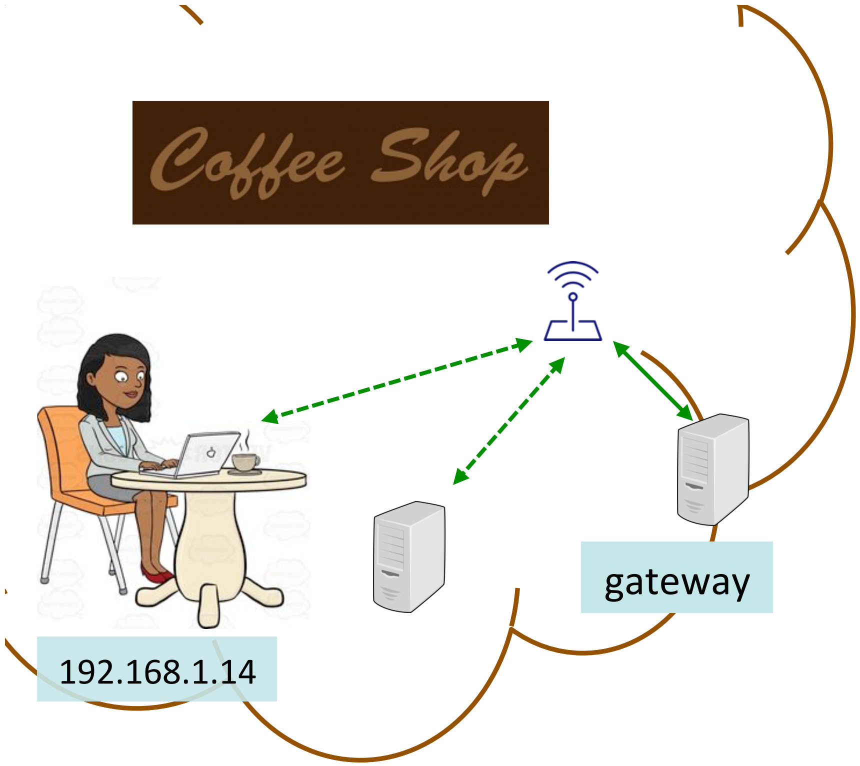
192.168.1.14

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

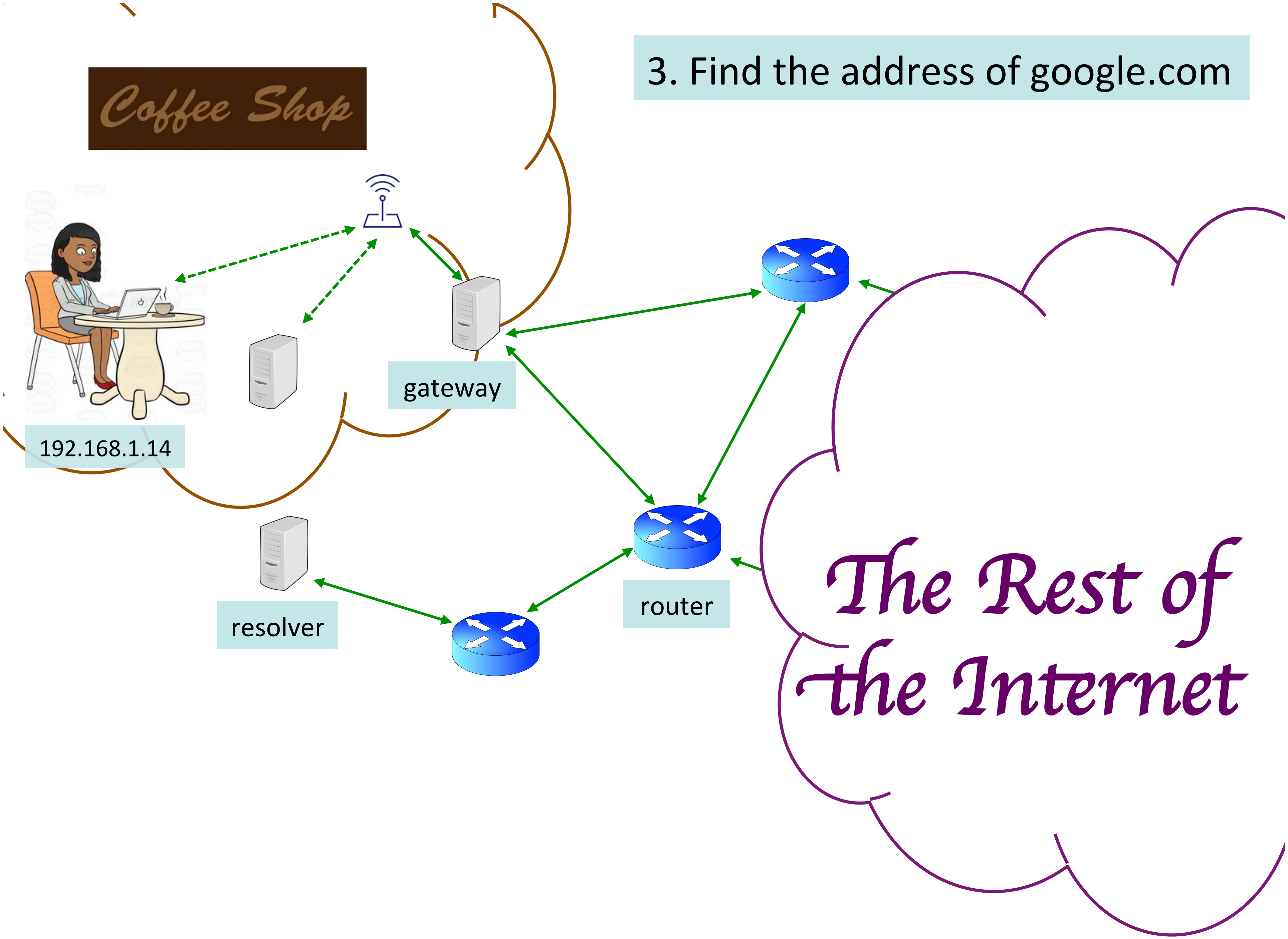
It’s transmitted using the **UDP** protocol (lightweight, unreliable).

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



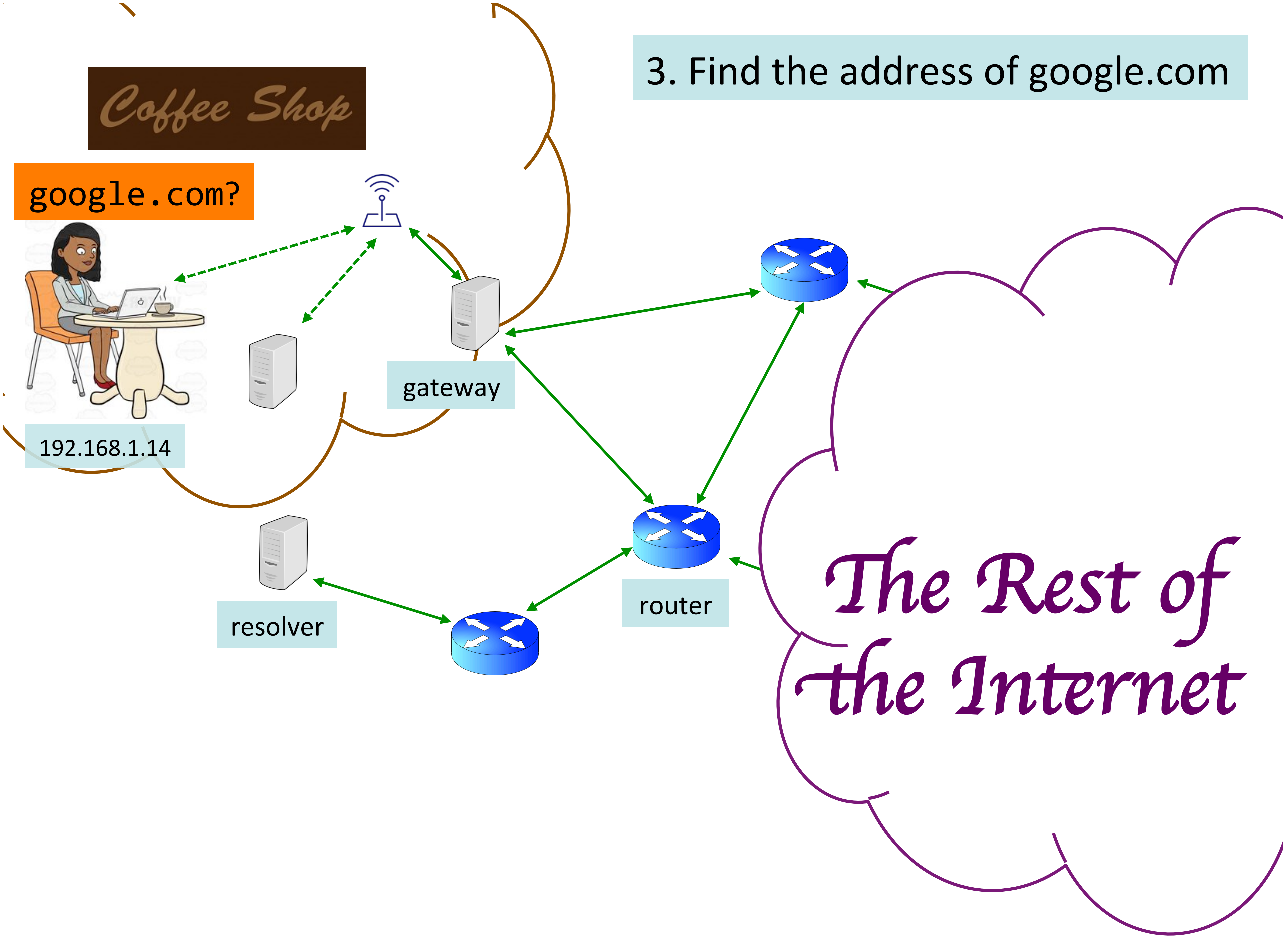


3. Find the address of google.com

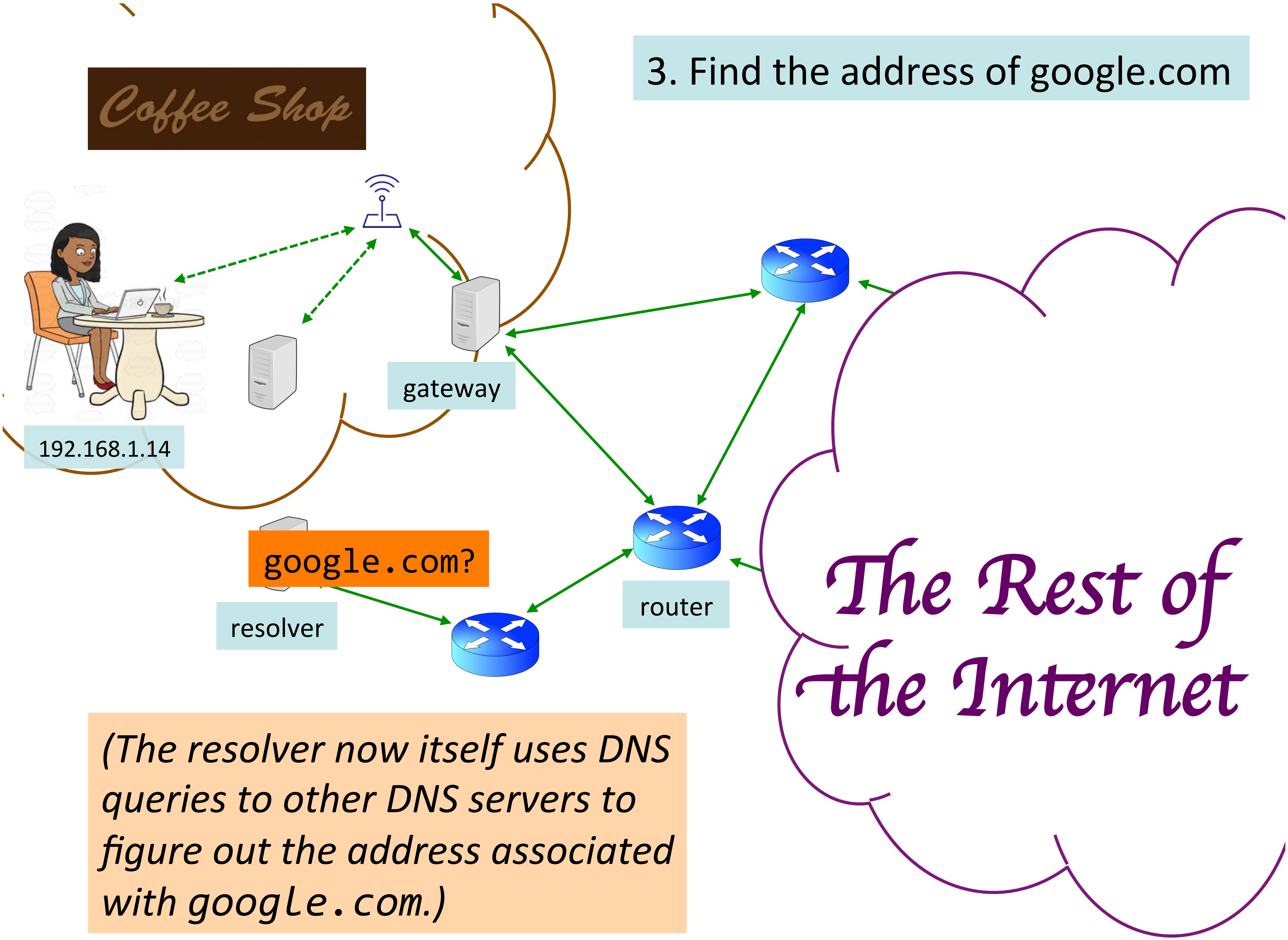


Coffee Shop
3. Find the address of google.com
gateway
192.168.1.14
resolver
router
The Rest of the Internet

3. Find the address of google.com

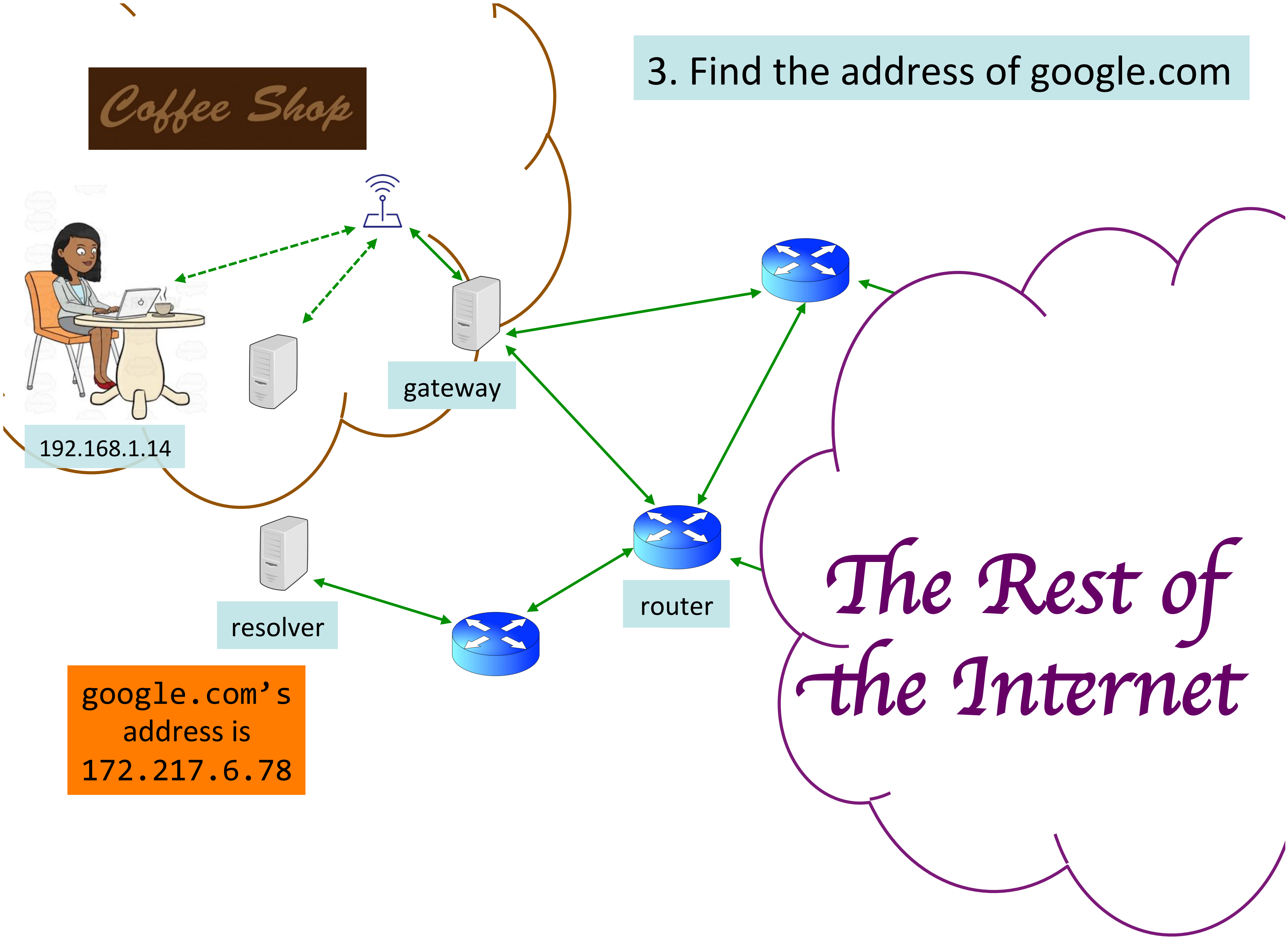


3. Find the address of google.com

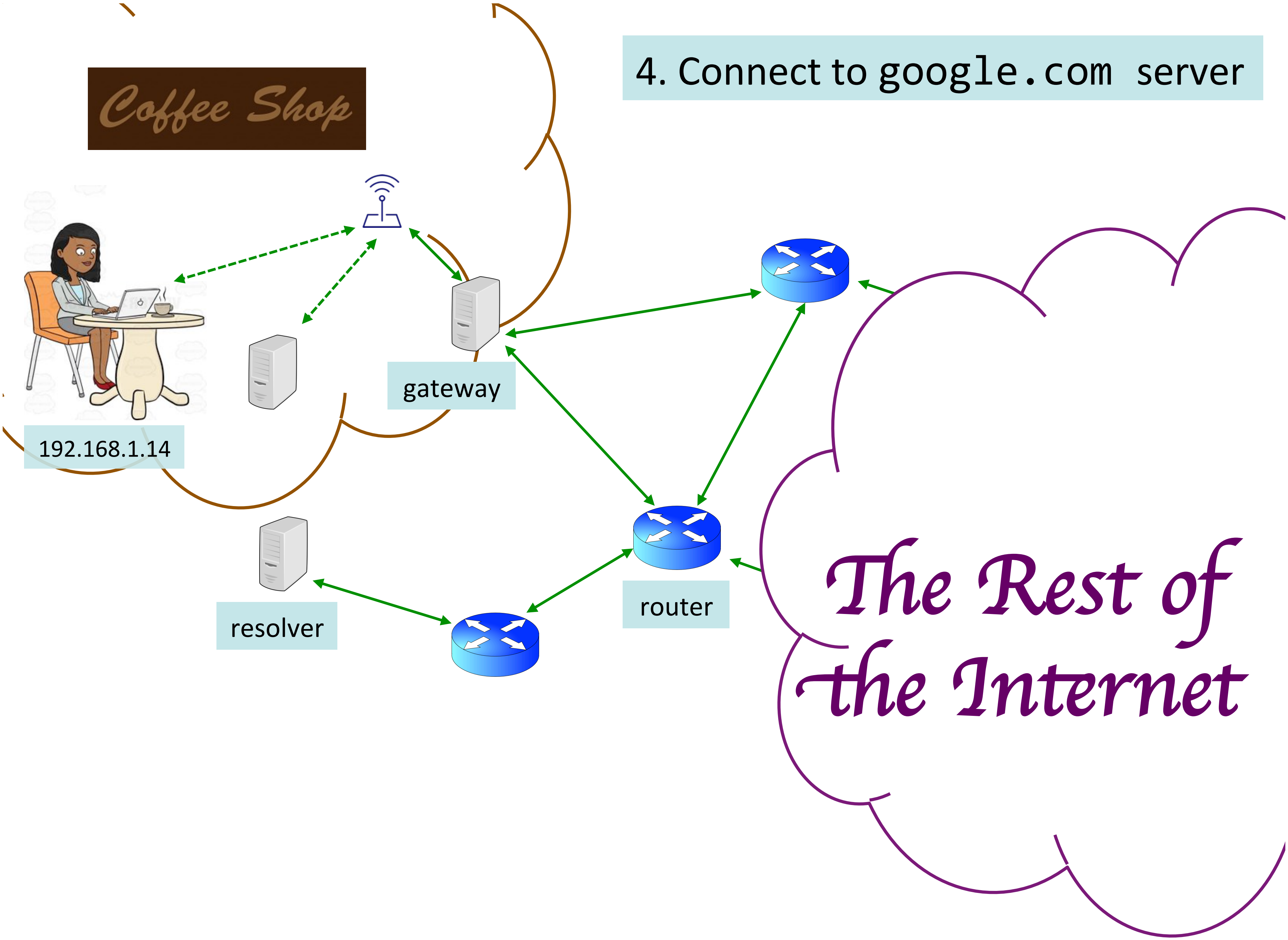


(The resolver now itself uses DNS queries to other DNS servers to figure out the address associated with `google.com`.)

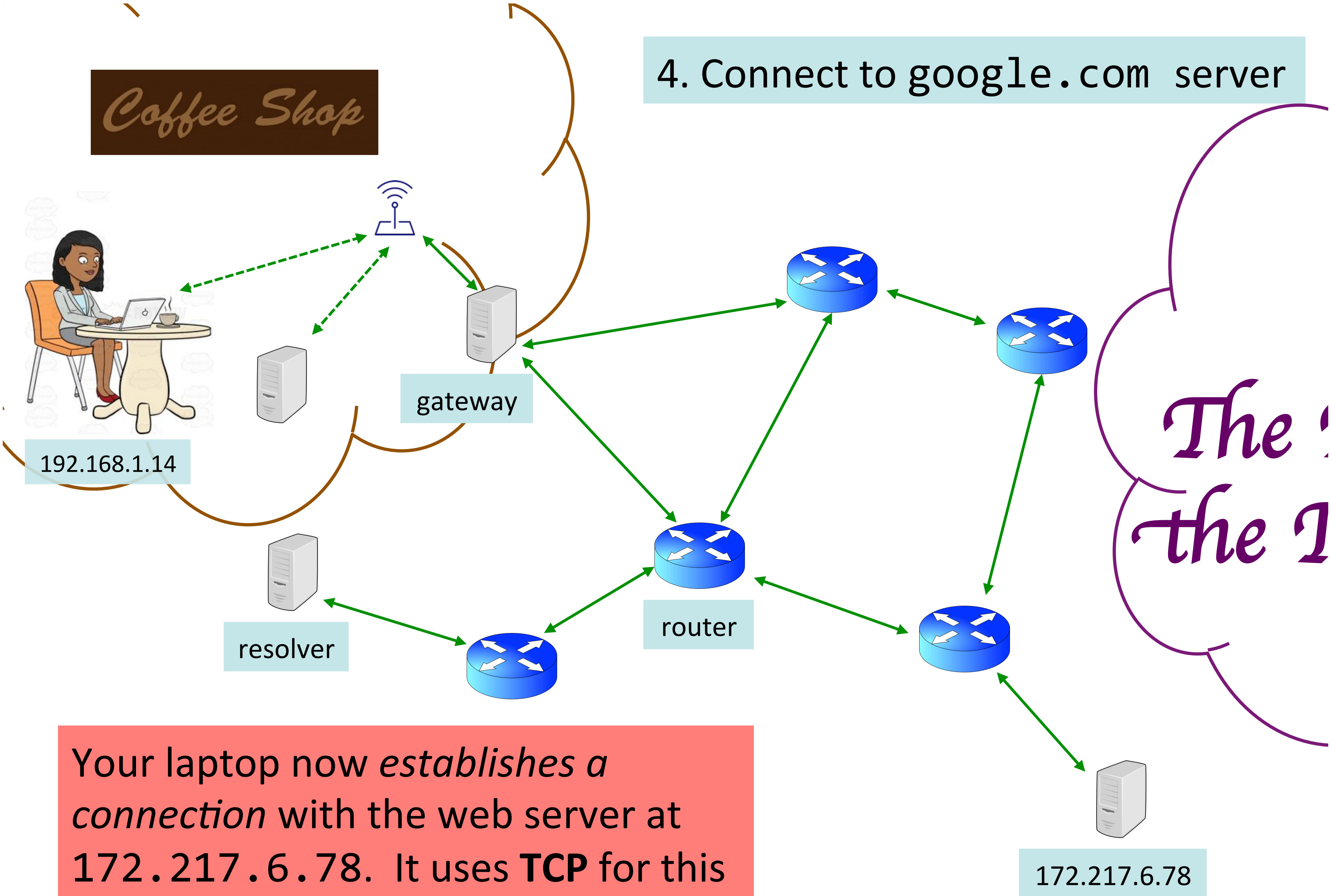
3. Find the address of google.com



4. Connect to google.com server

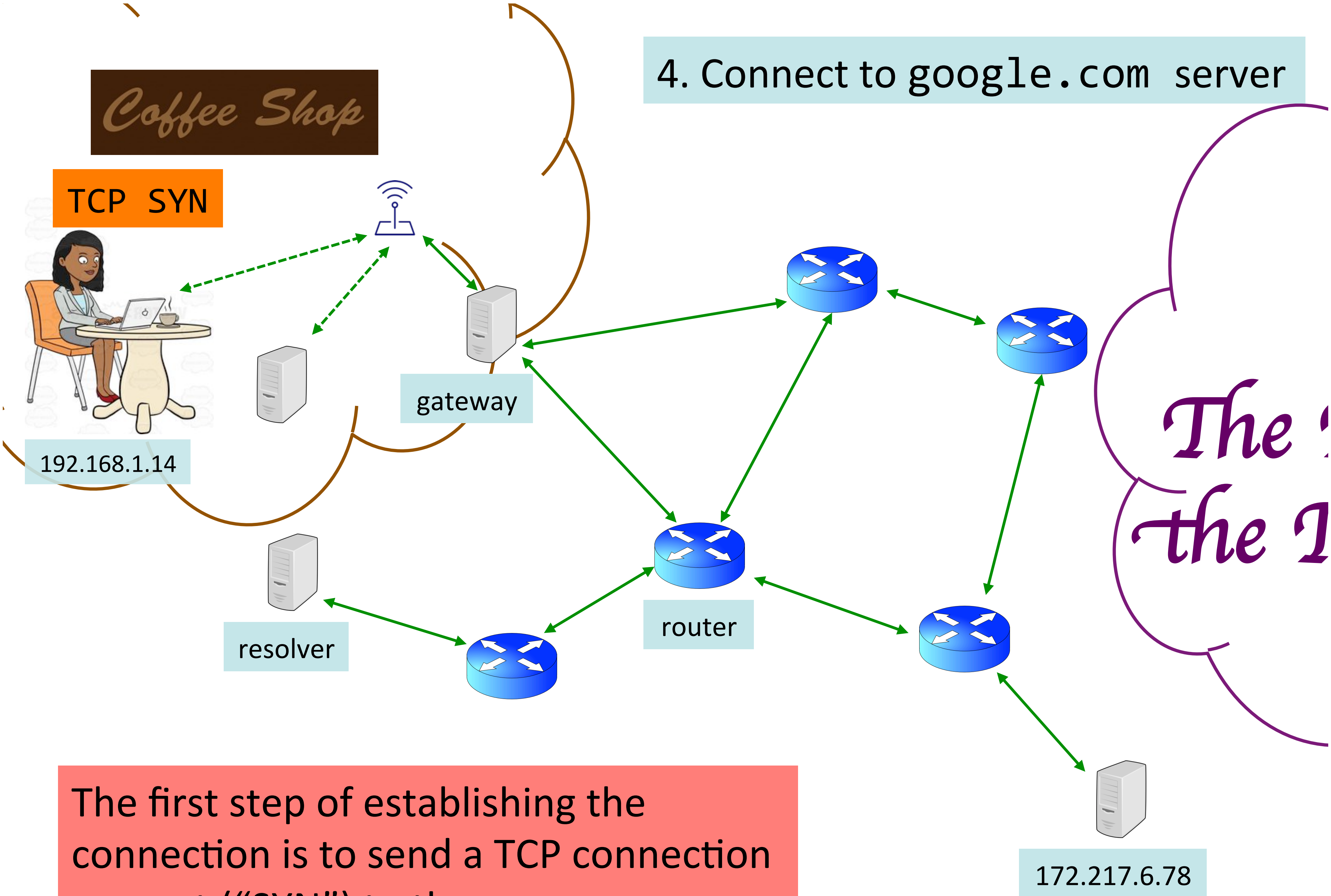


4. Connect to `google.com` server



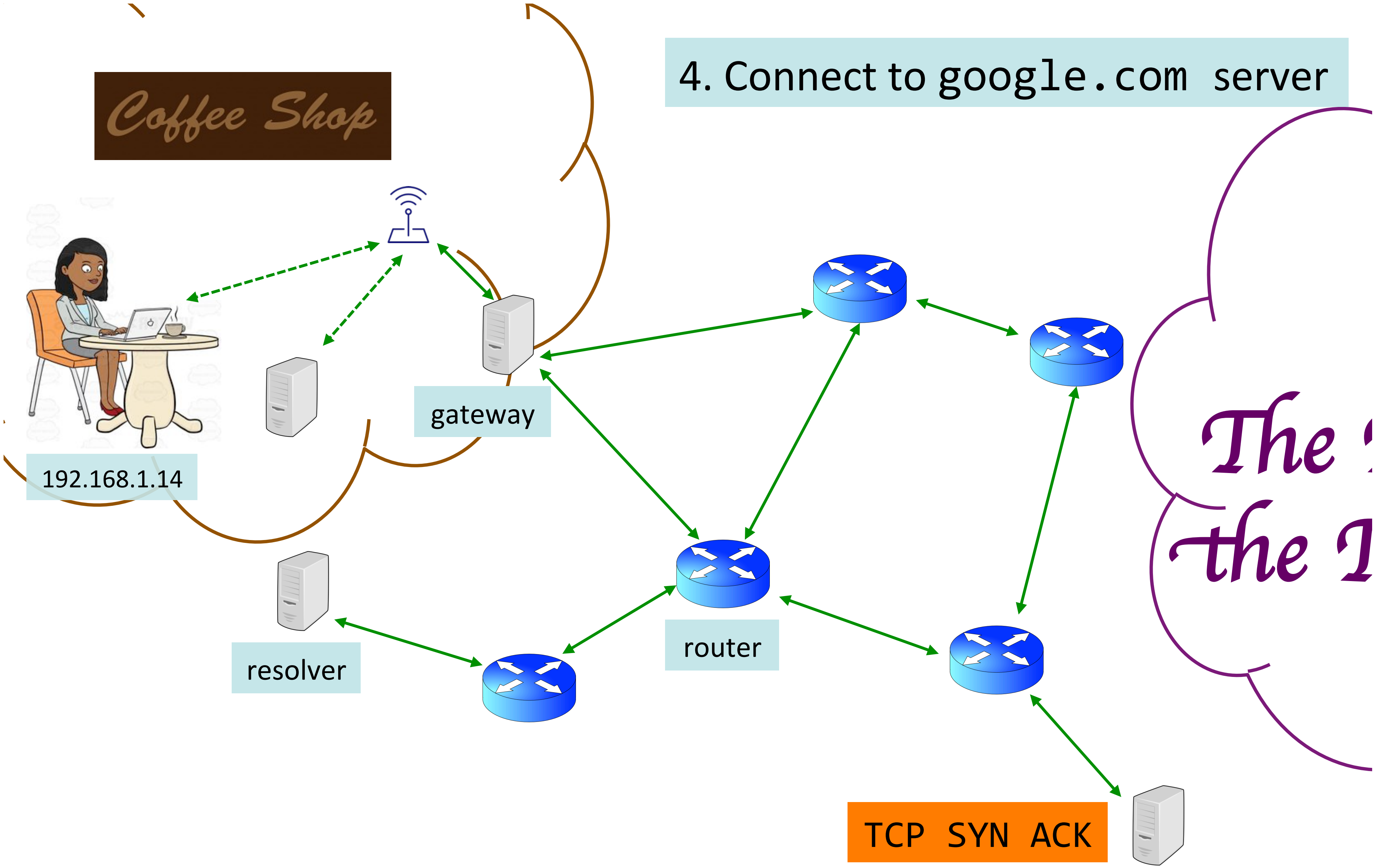
Your laptop now *establishes a connection* with the web server at `172.217.6.78`. It uses **TCP** for this rather than UDP, to obtain reliability.

4. Connect to `google.com` server



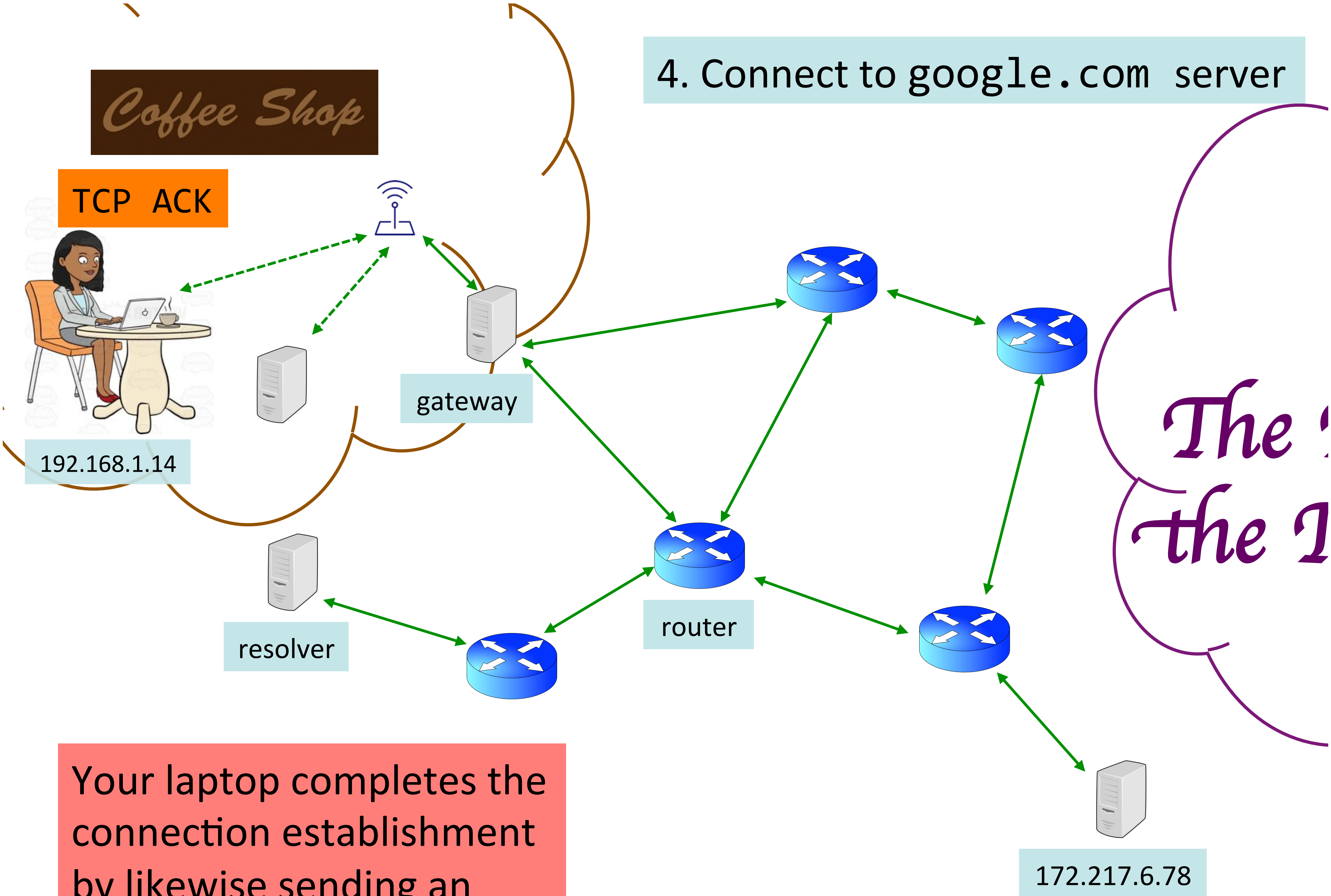
The first step of establishing the connection is to send a TCP connection request ("SYN") to the server.

4. Connect to `google.com` server



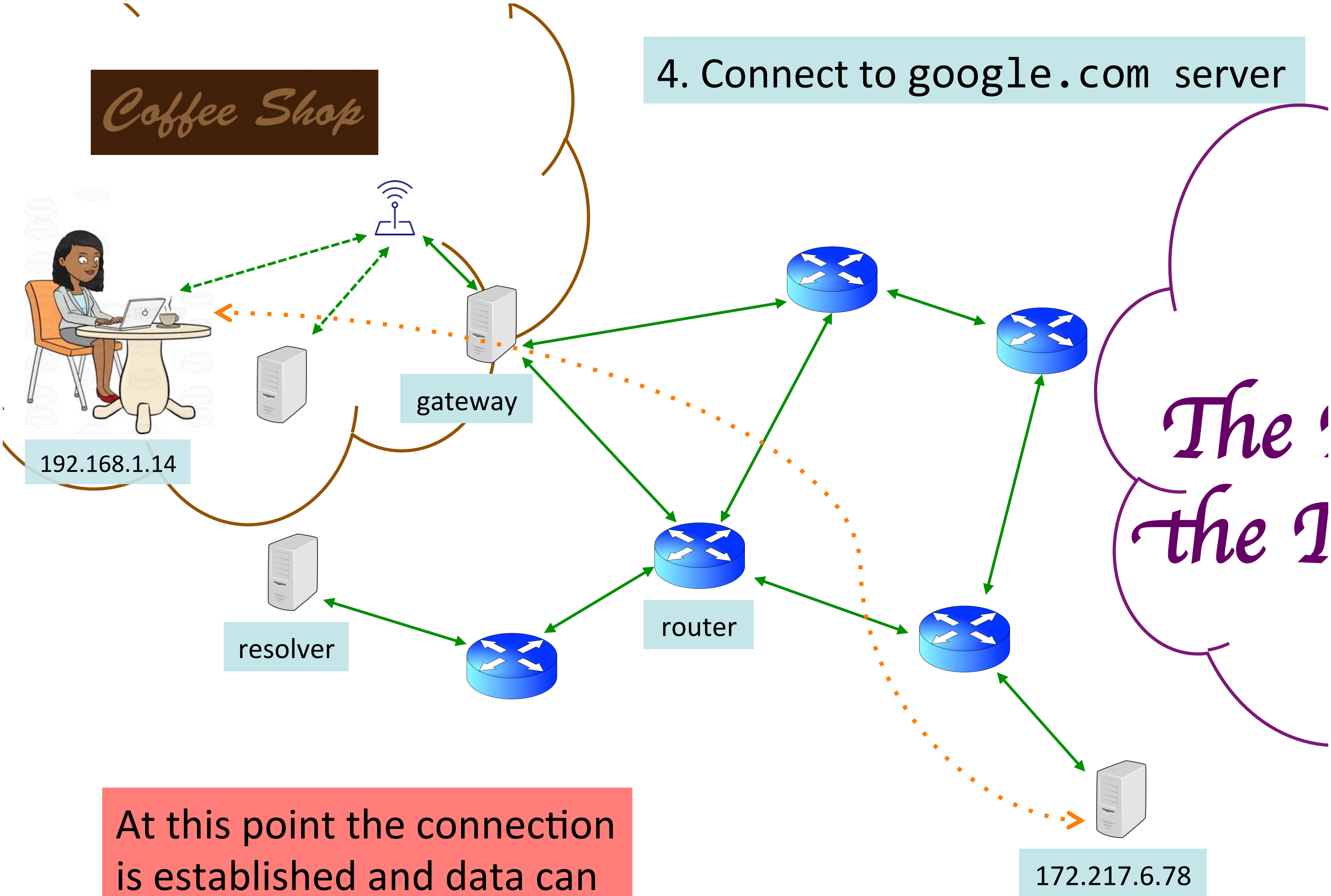
If the server accepts the connection, it replies with a “SYN ACK”.

4. Connect to google.com server

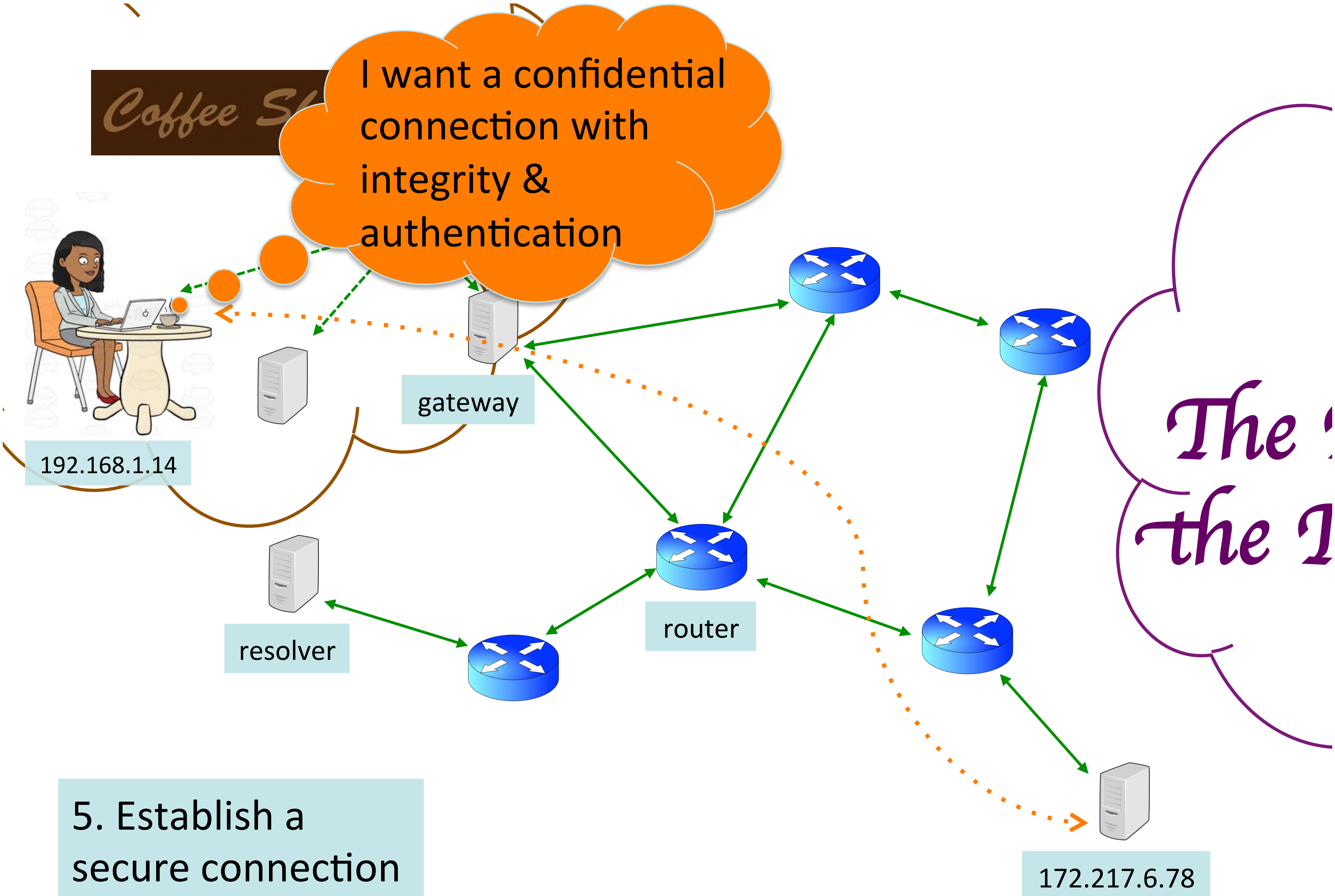


Your laptop completes the connection establishment by likewise sending an acknowledgement.

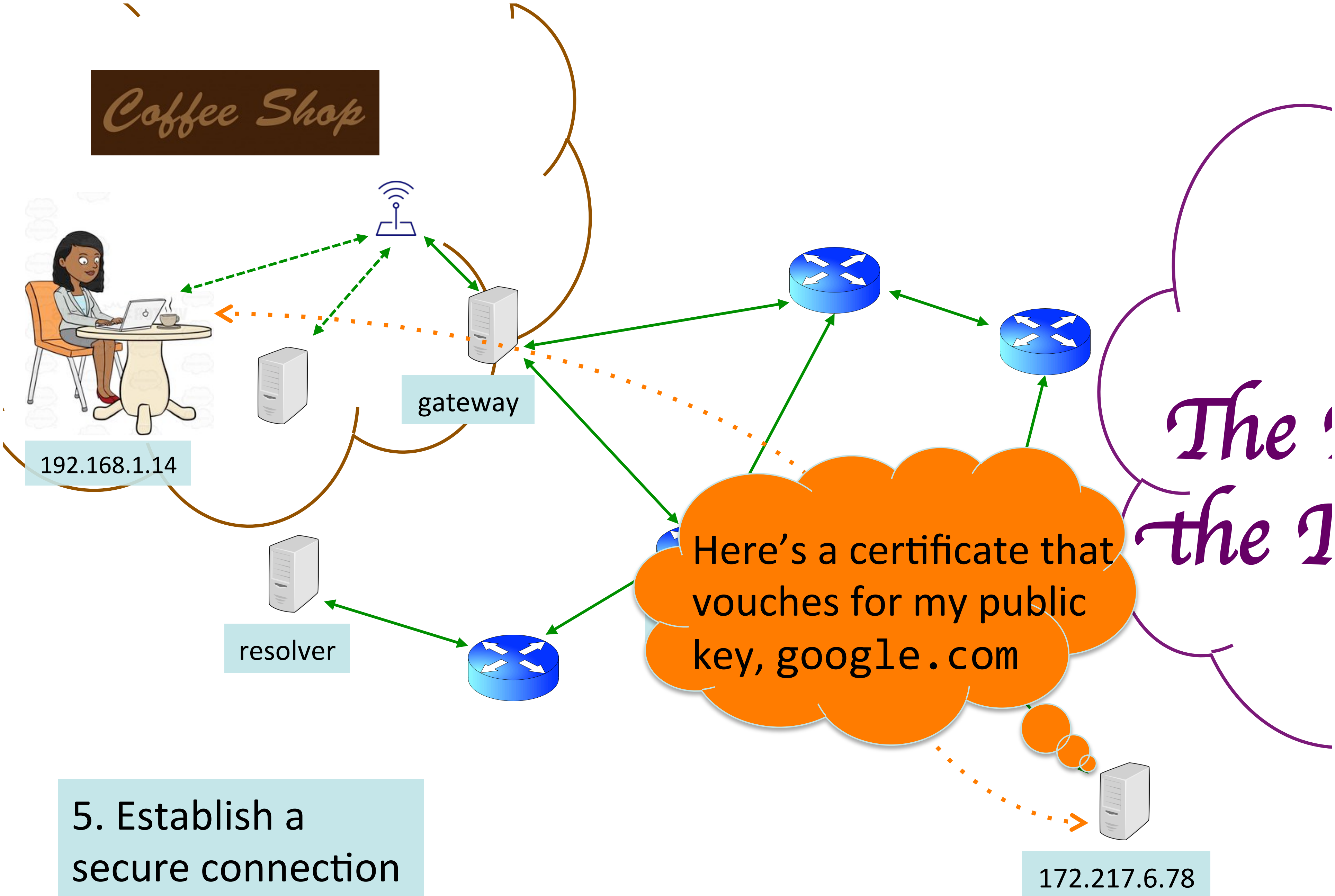
4. Connect to `google.com` server



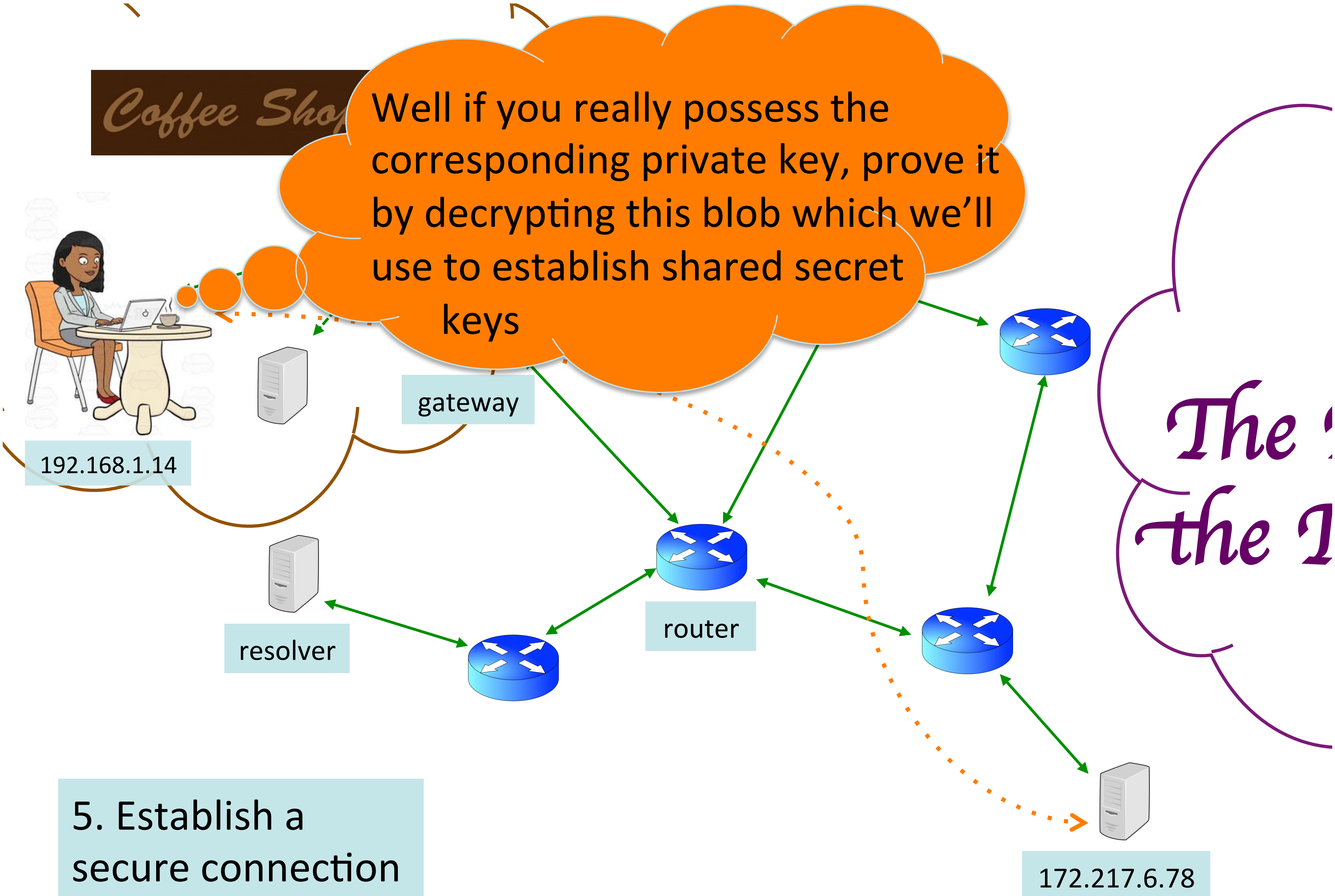
At this point the connection is established and data can be (reliably) exchanged.



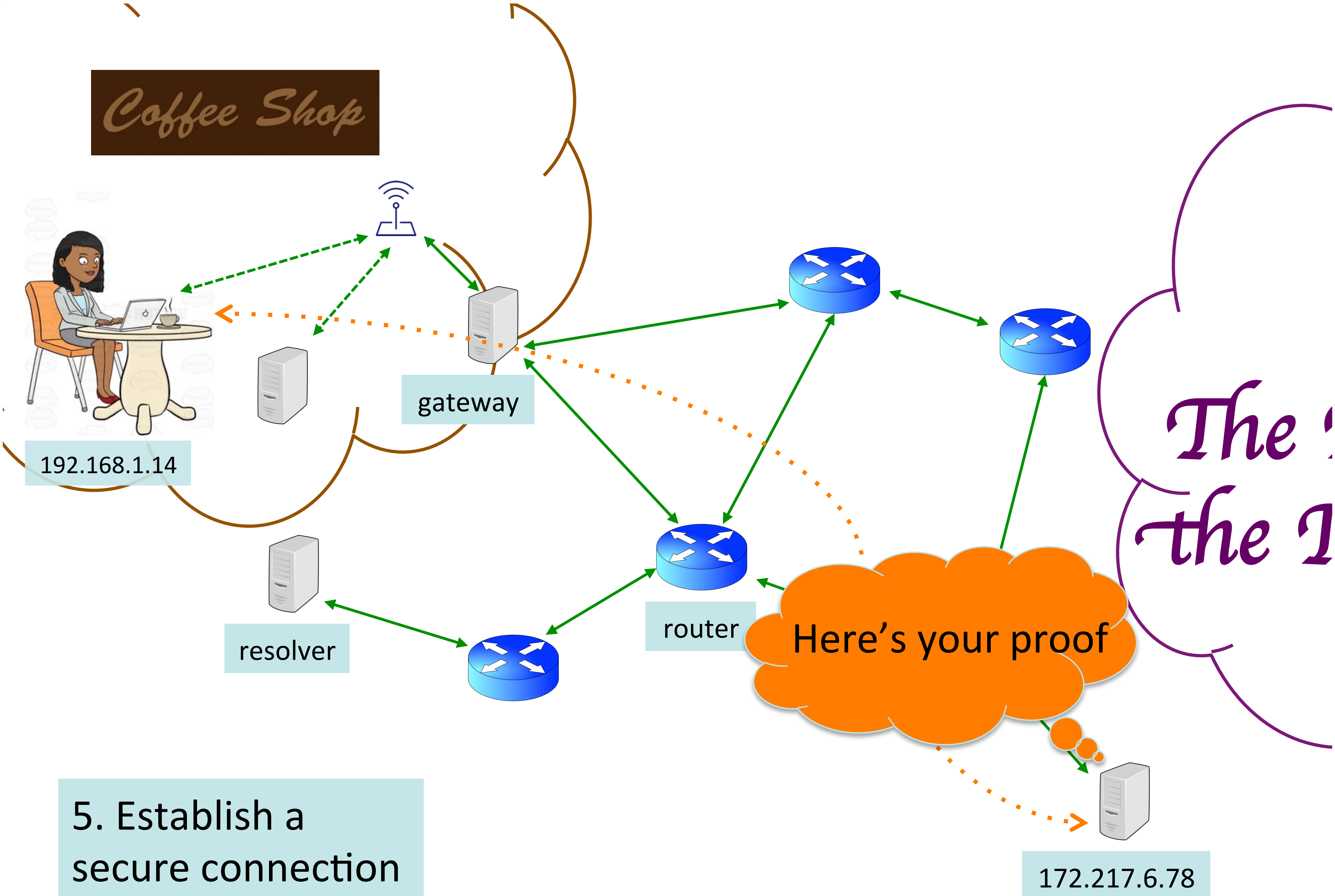
5. Establish a secure connection using **TLS** (https)



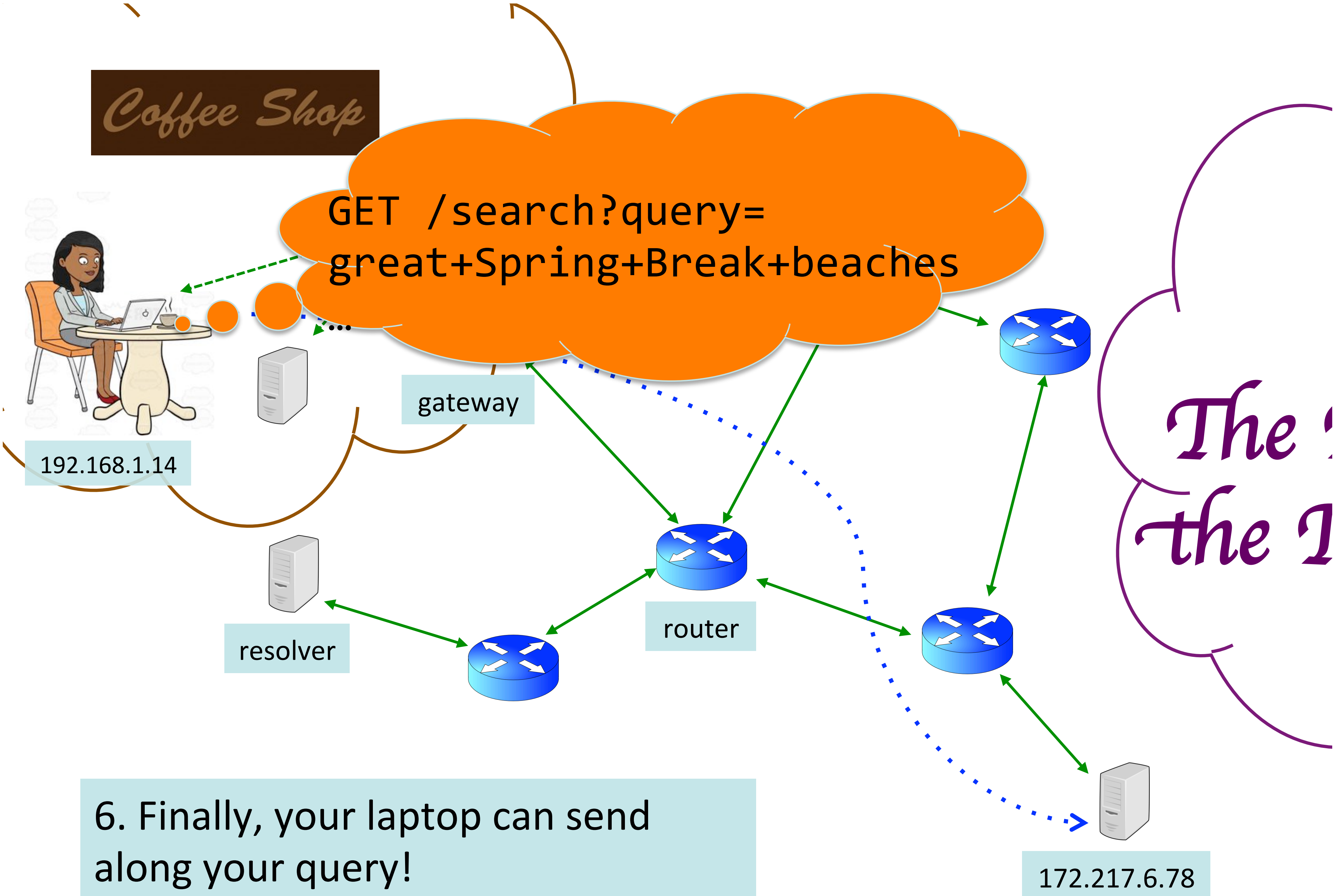
5. Establish a secure connection using **TLS** (https)



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*)

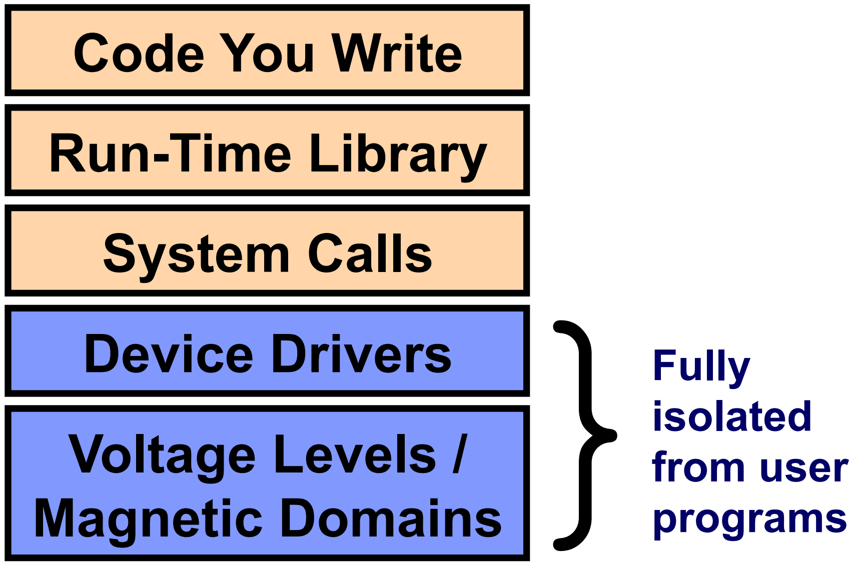
5 Minute Break

Questions Before We Proceed?

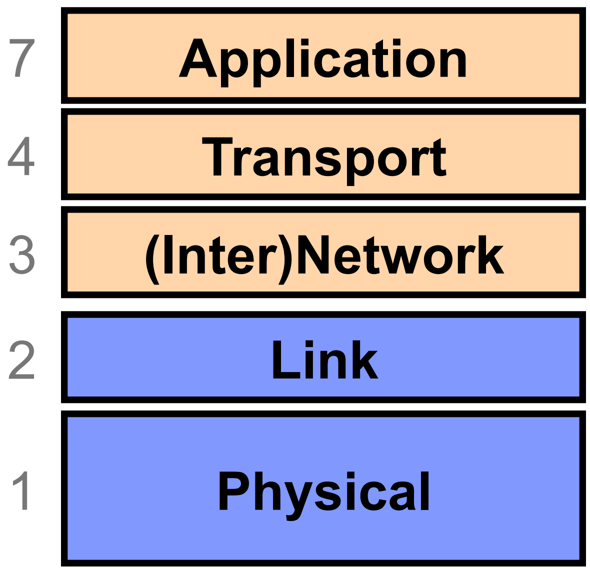
Internet Layering

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



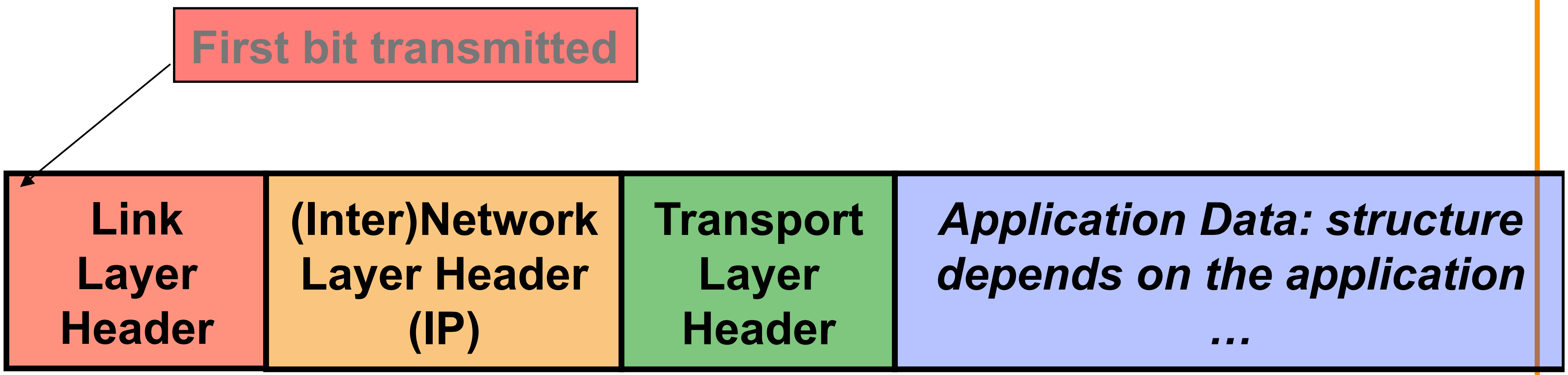
Internet Layering ("Protocol Stack")



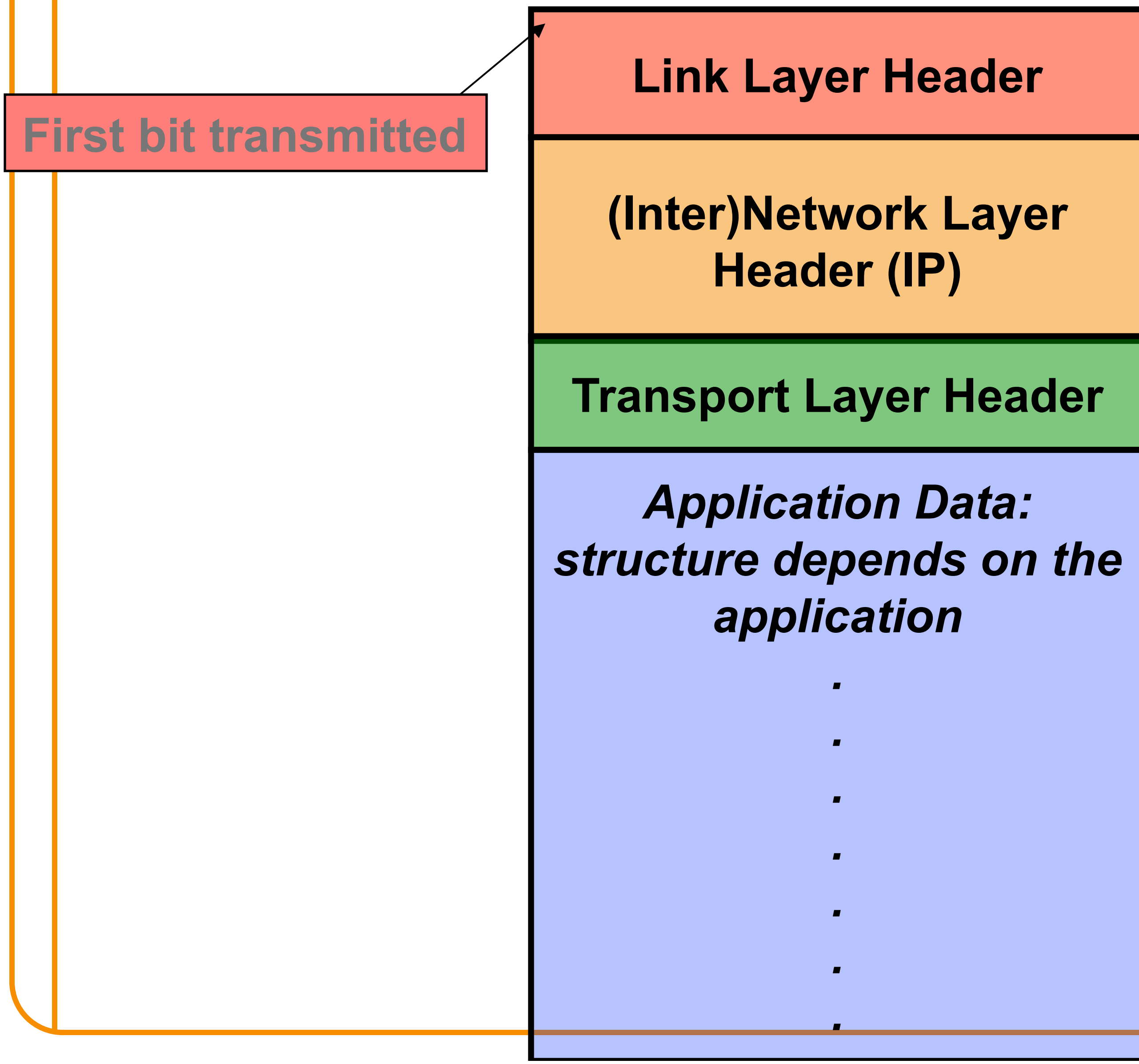
Note on a point of potential confusion: these diagrams are always drawn with lower layers **below** higher layers …

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

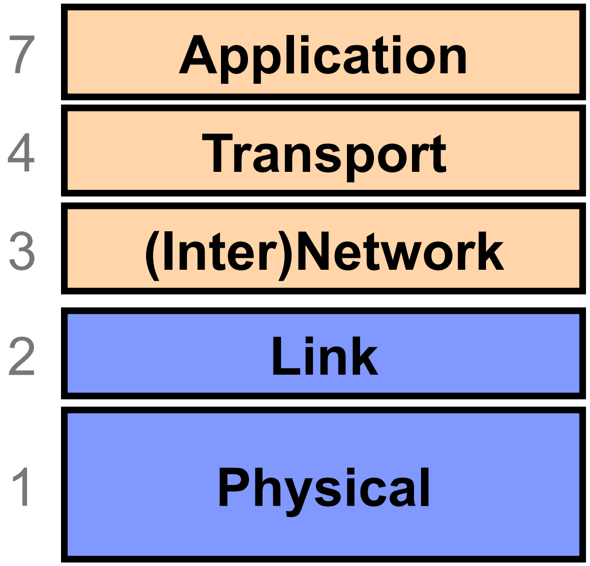
Horizontal View of a Single Packet



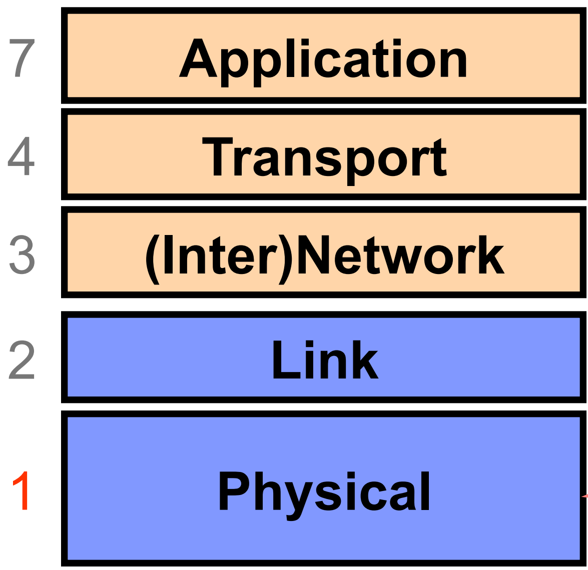
Vertical View of a Single Packet



Internet Layering ("Protocol Stack")



Layer 1: Physical Layer



Encoding bits to send them over a single **physical link**

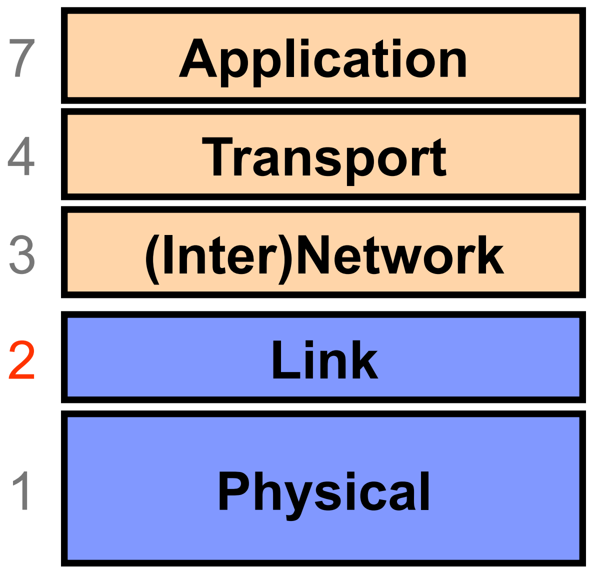
e.g. patterns of

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)

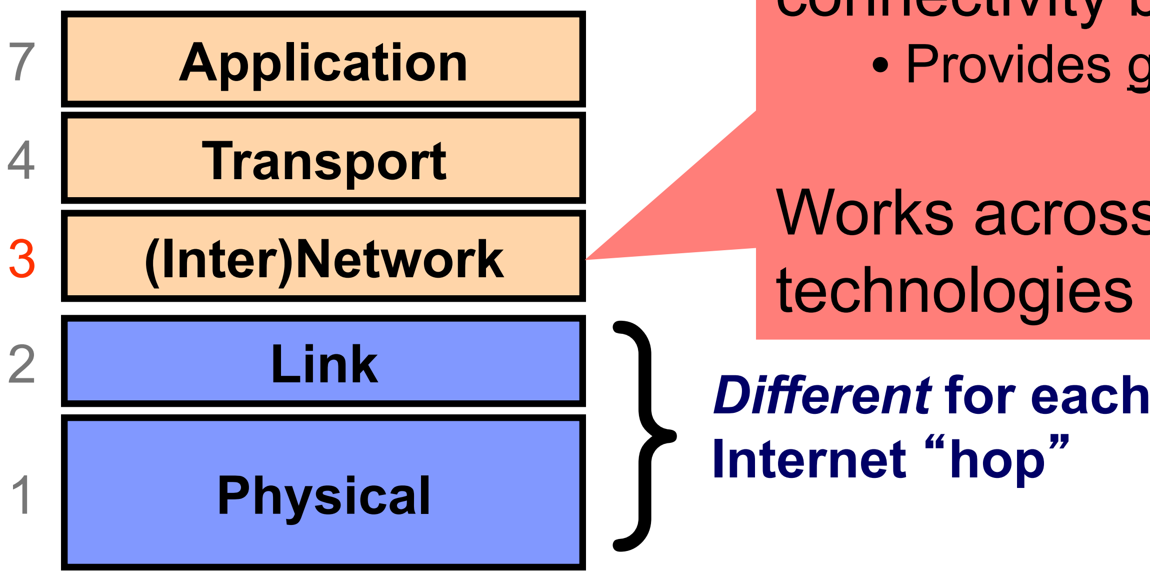
Often technology supports broadcast transmission (**every** "node" connected to subnet receives)

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

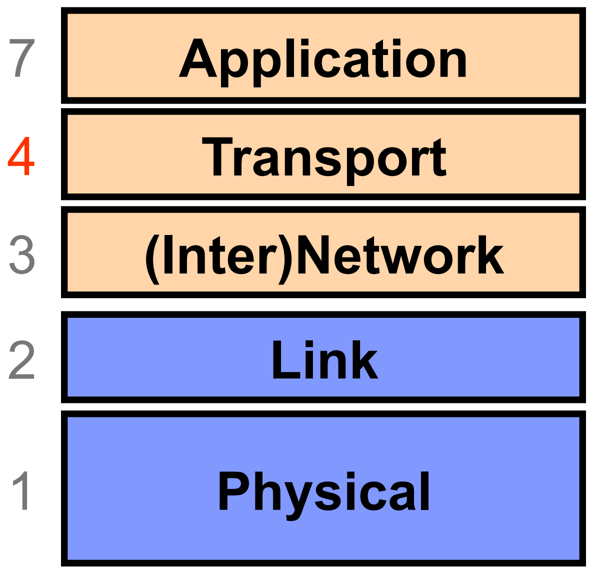
Bridges multiple "subnets" to provide *end-to-end* internet connectivity between nodes

- Provides global addressing

Works across different link technologies



Layer 4: Transport Layer

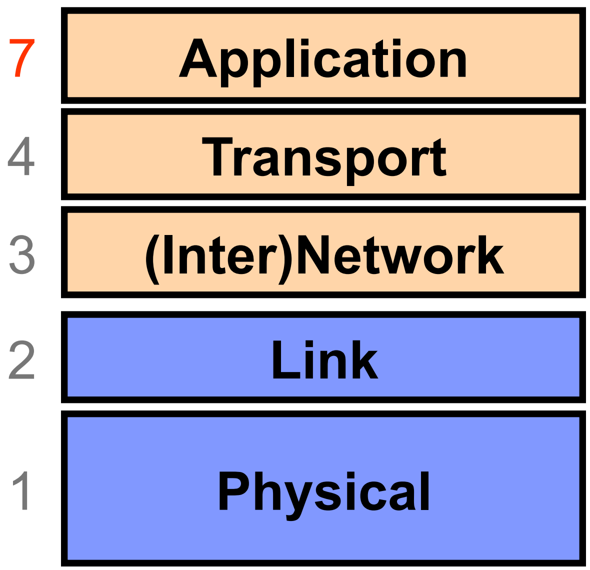


End-to-end communication between processes

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

(Datagram = single packet message)

Layer 7: Application Layer



Communication of whatever you wish

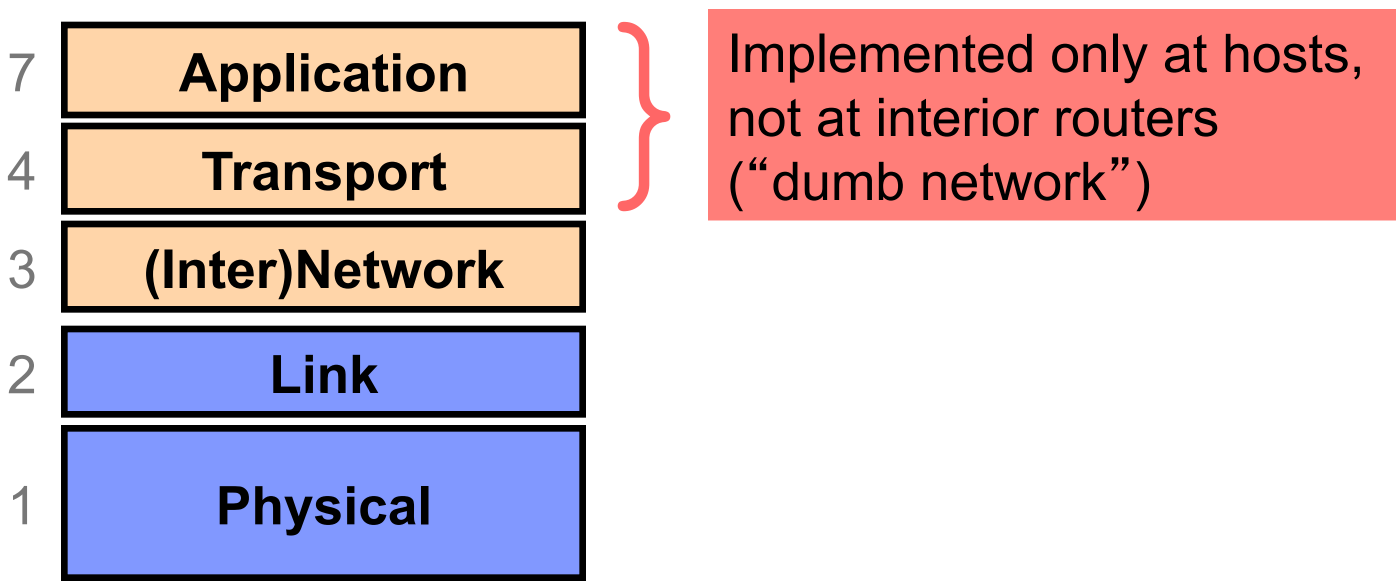
Can use whatever transport(s) is convenient

Freely structured

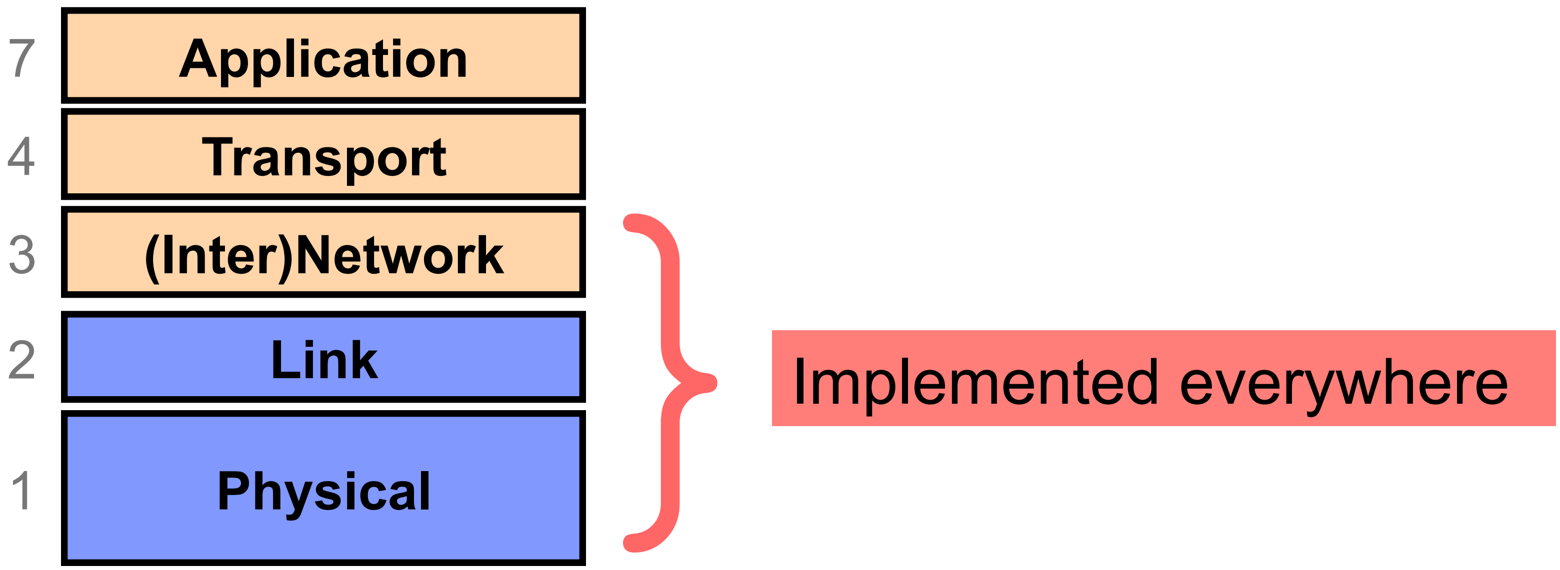
E.g.:

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

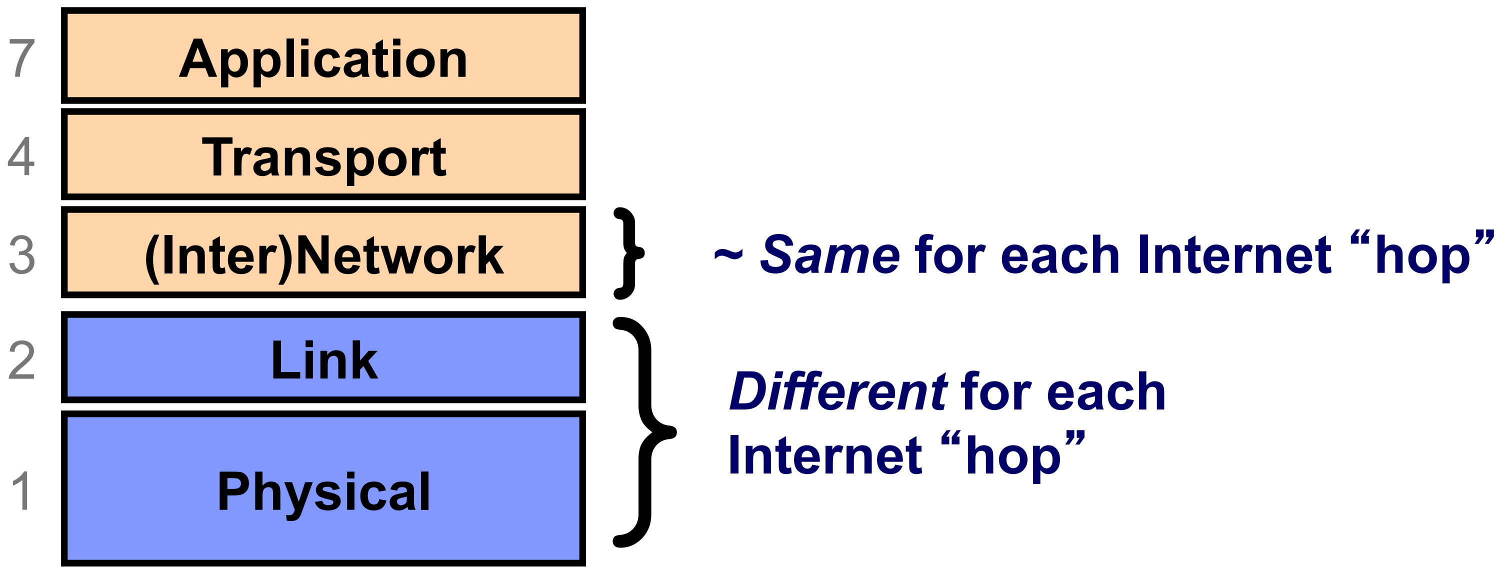
Internet Layering ("Protocol Stack")



Internet Layering ("Protocol Stack")

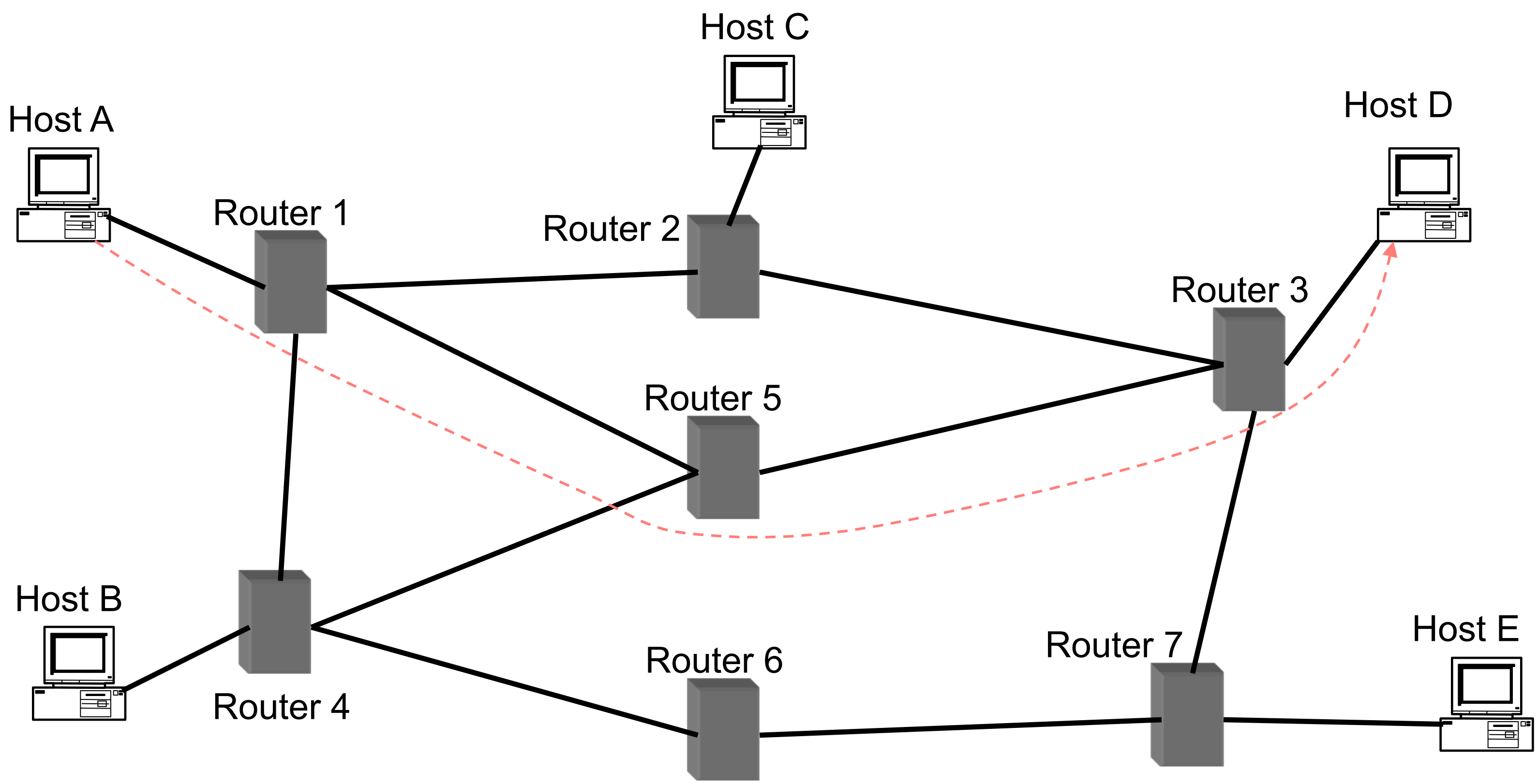


Internet Layering ("Protocol Stack")



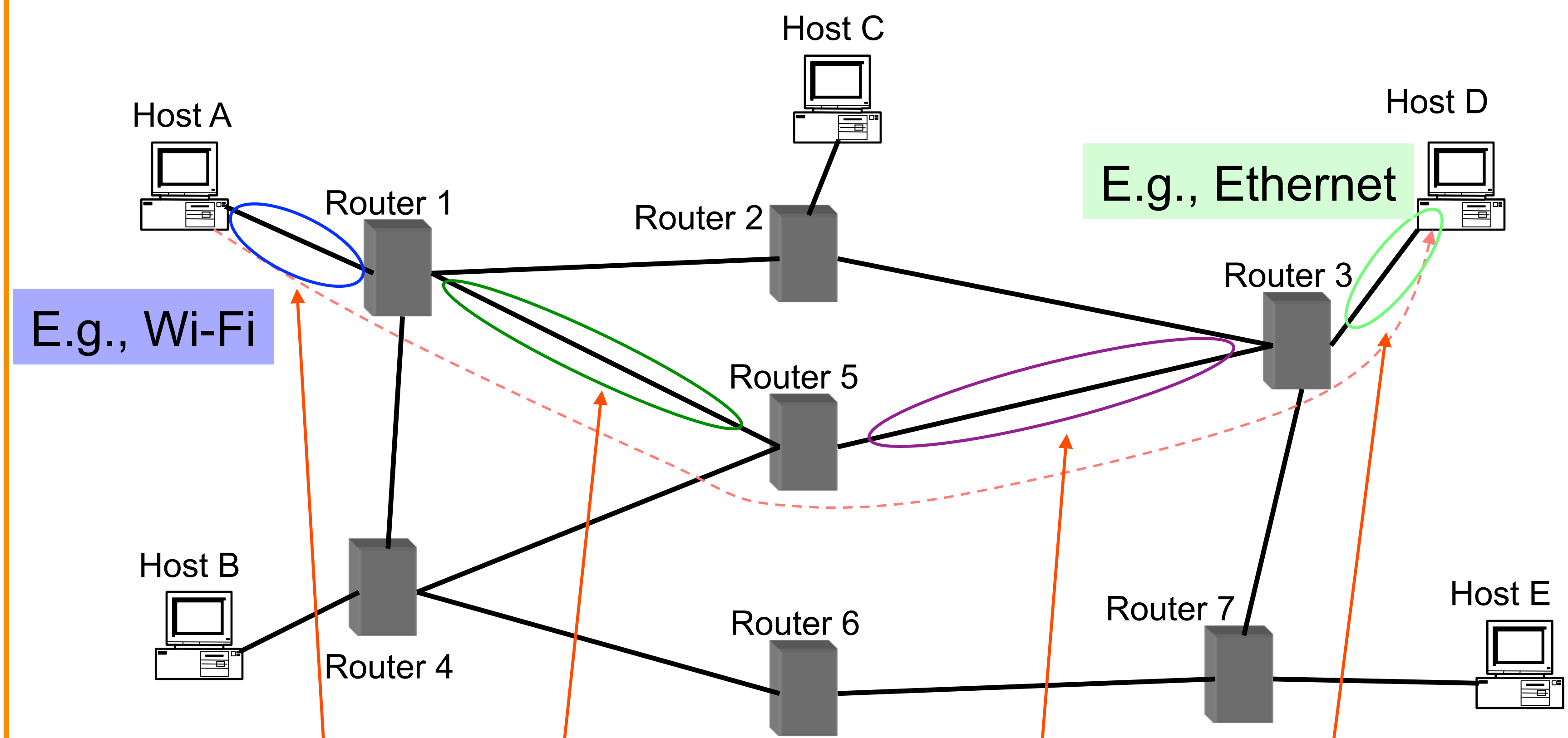
Hop-By-Hop vs. End-to-End Layers

Host A communicates with Host D



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

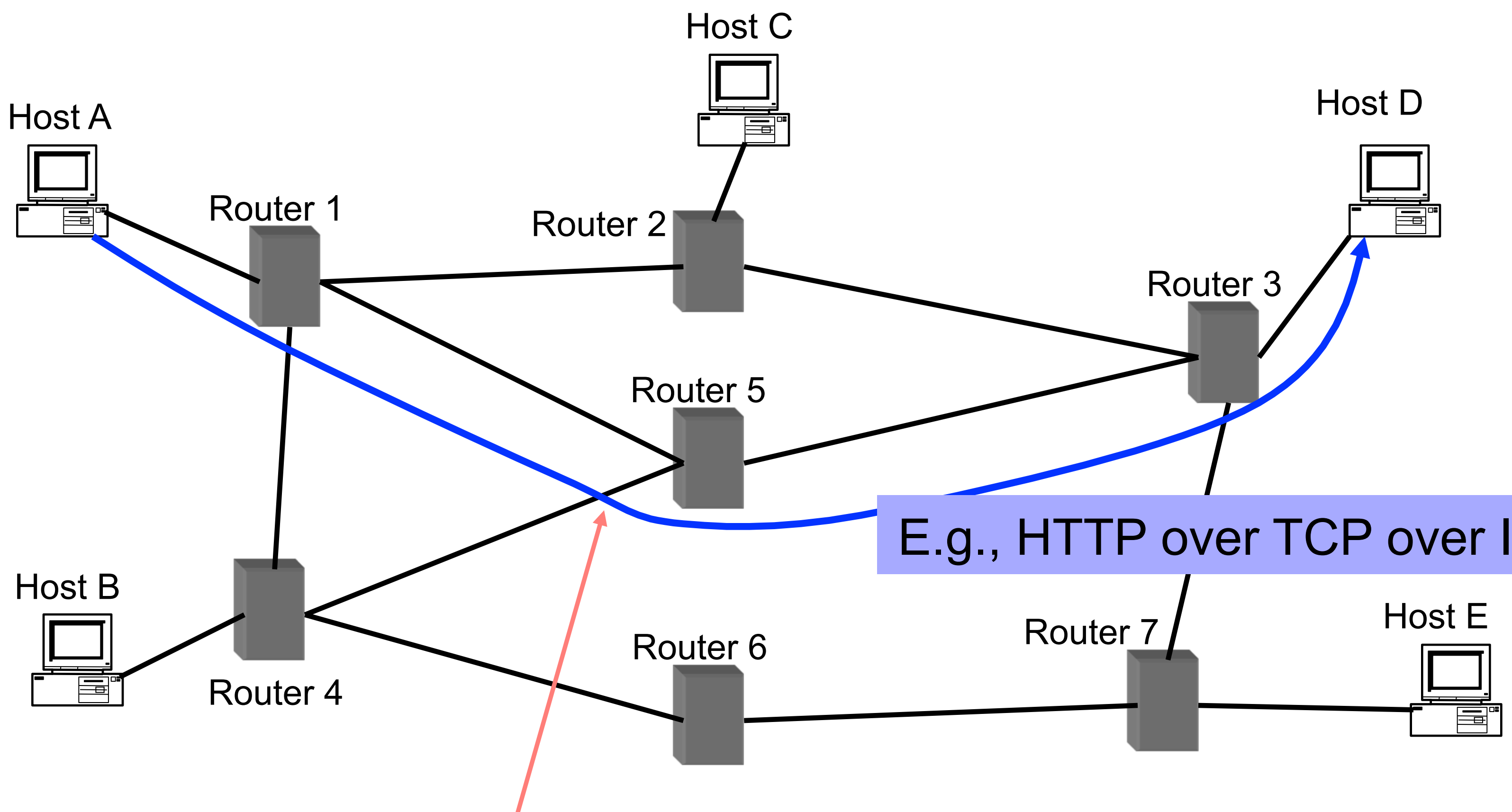
Host A communicates with Host D



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

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

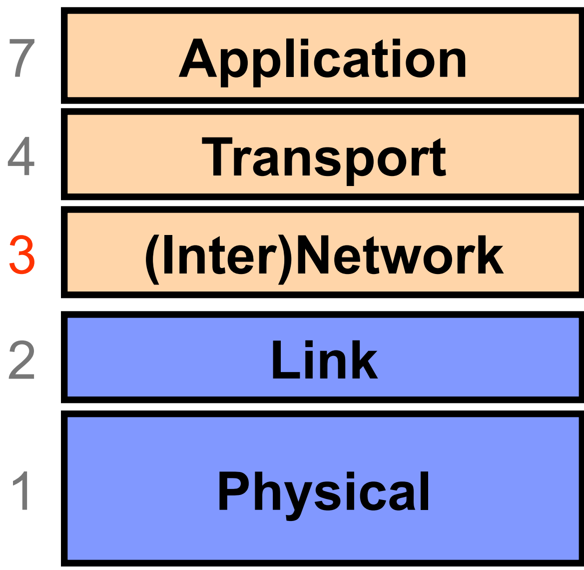
Host A communicates with Host D



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)*

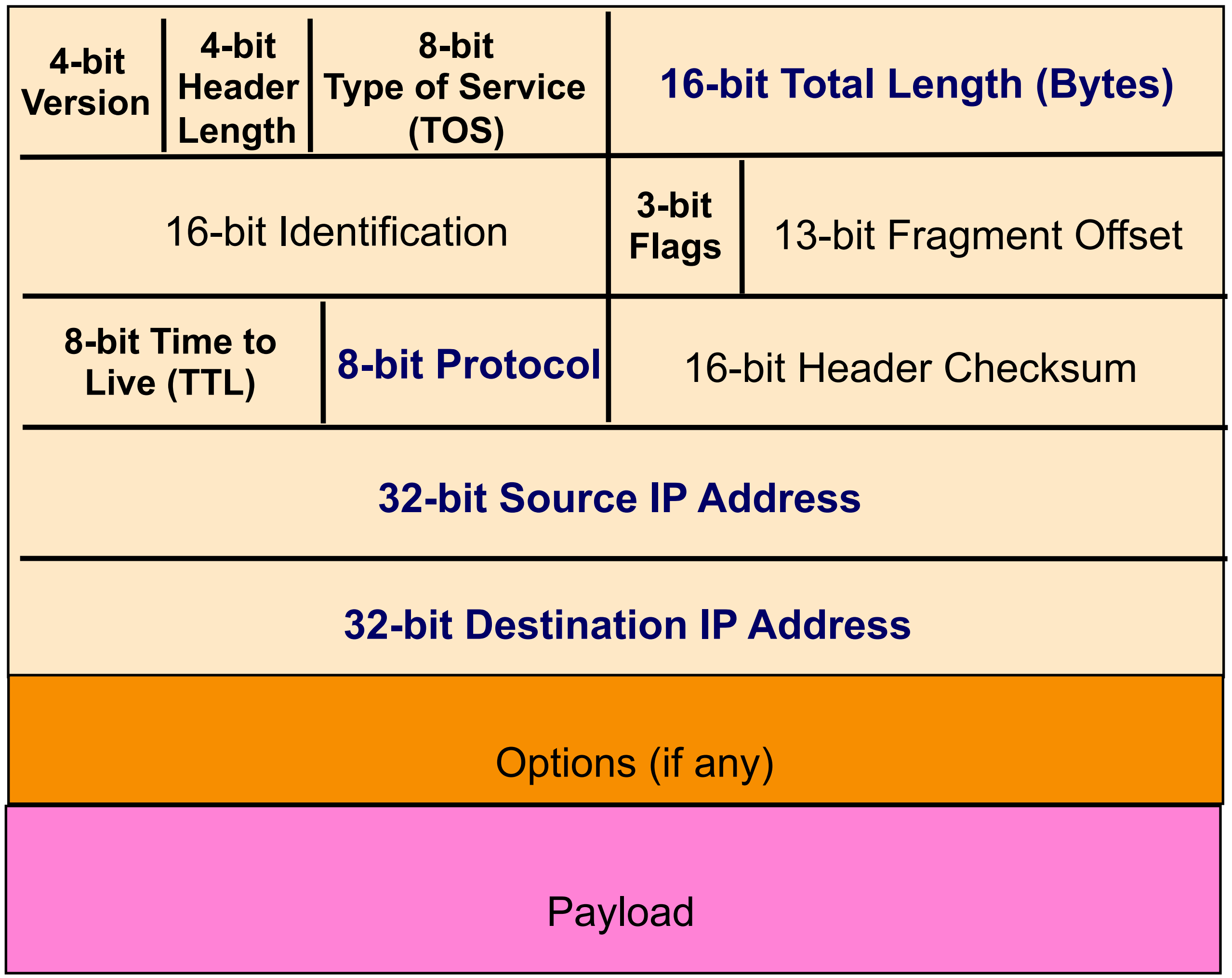


Bridges multiple “subnets” to provide *end-to-end* internet connectivity between nodes

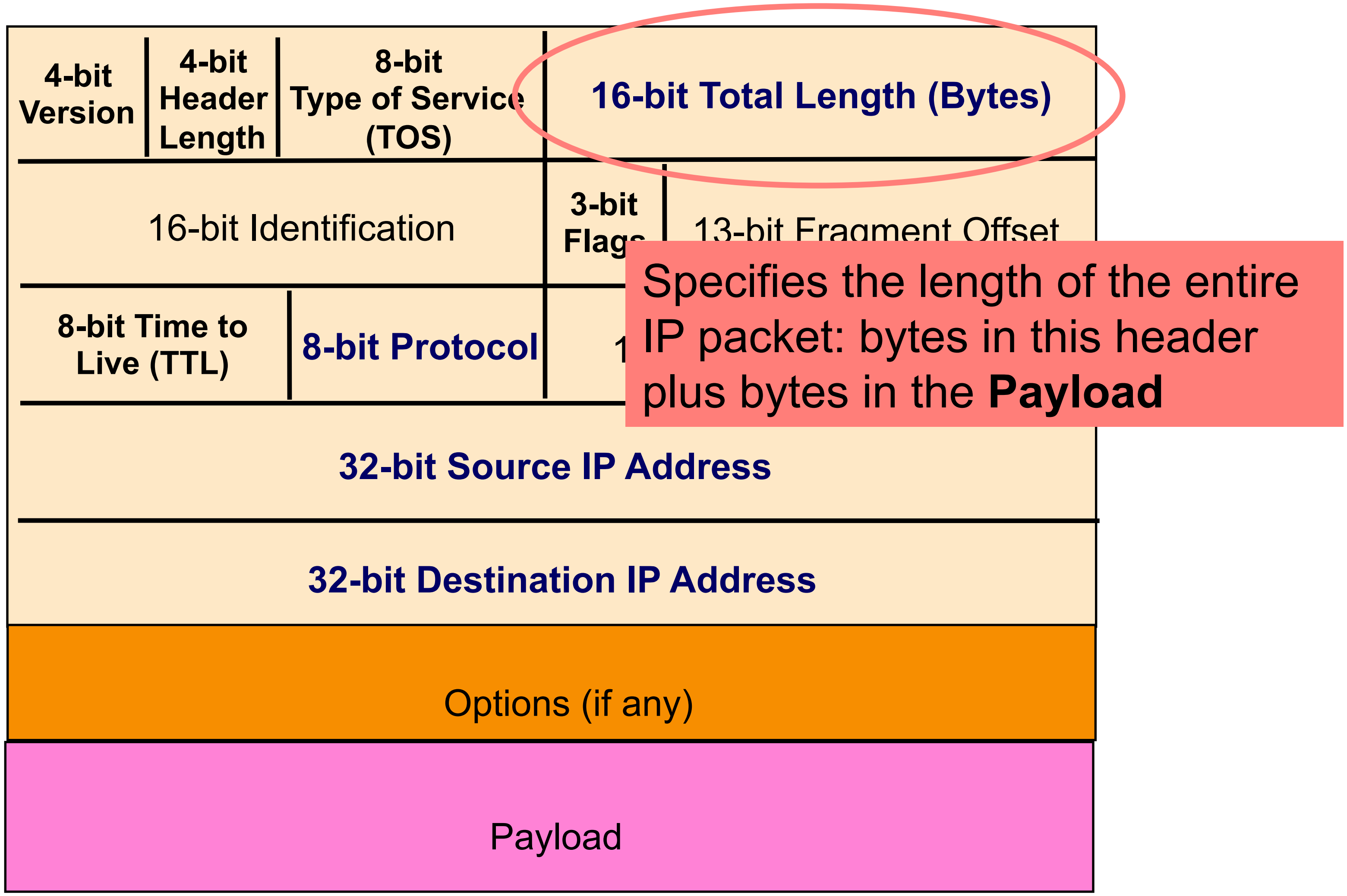
- Provides global addressing

Works across different link technologies

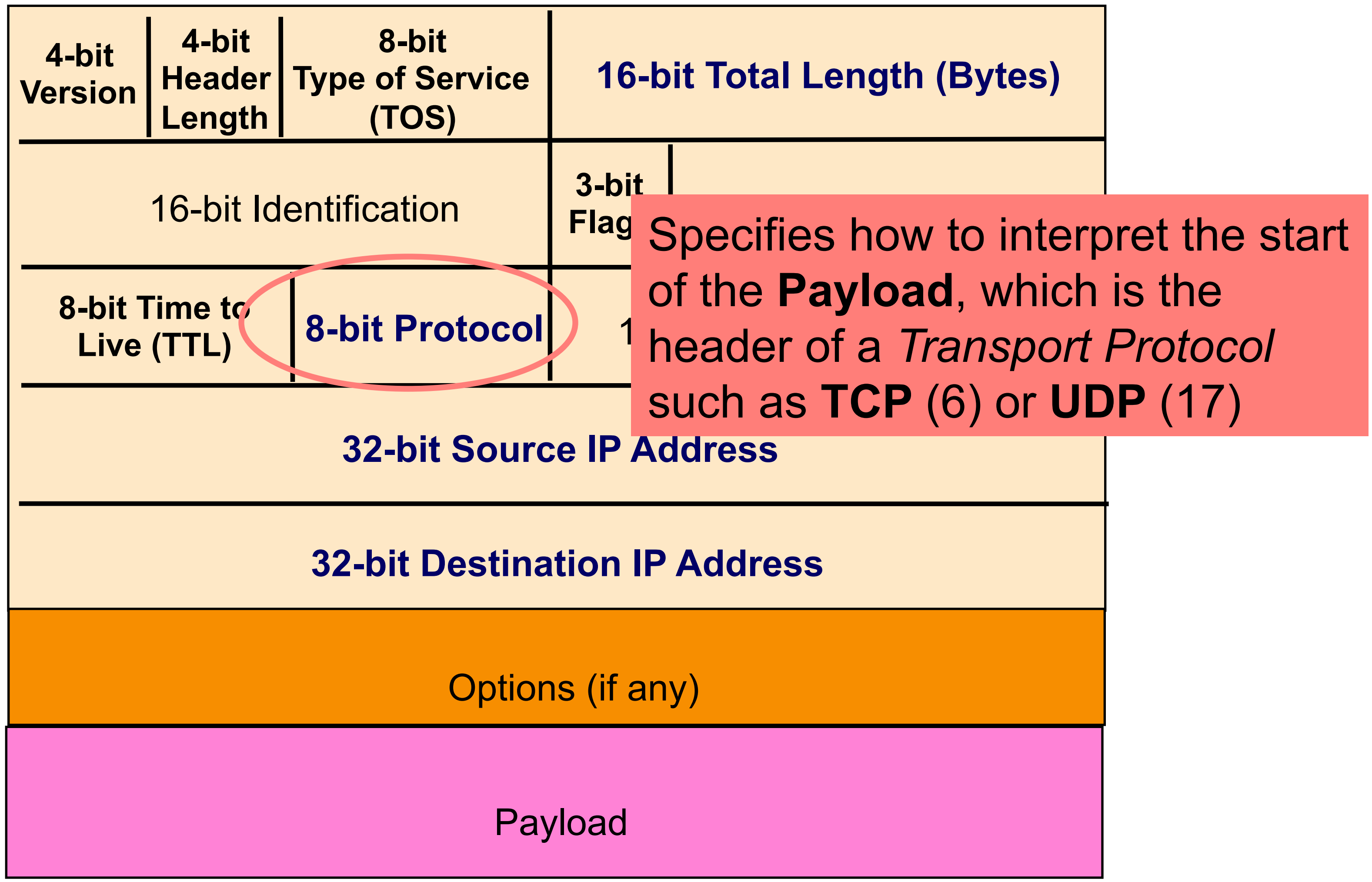
IP Packet Structure



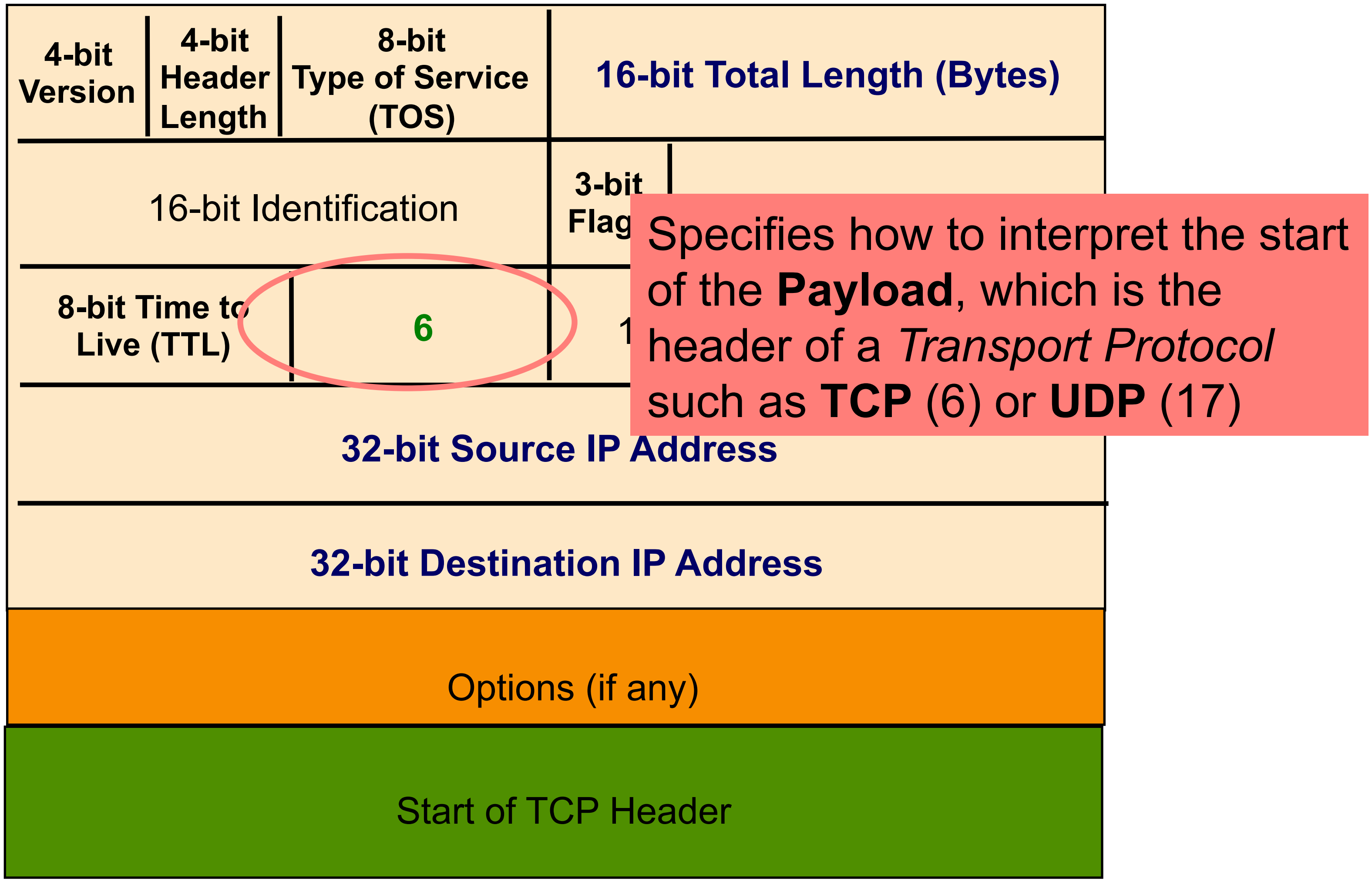
IP Packet Structure



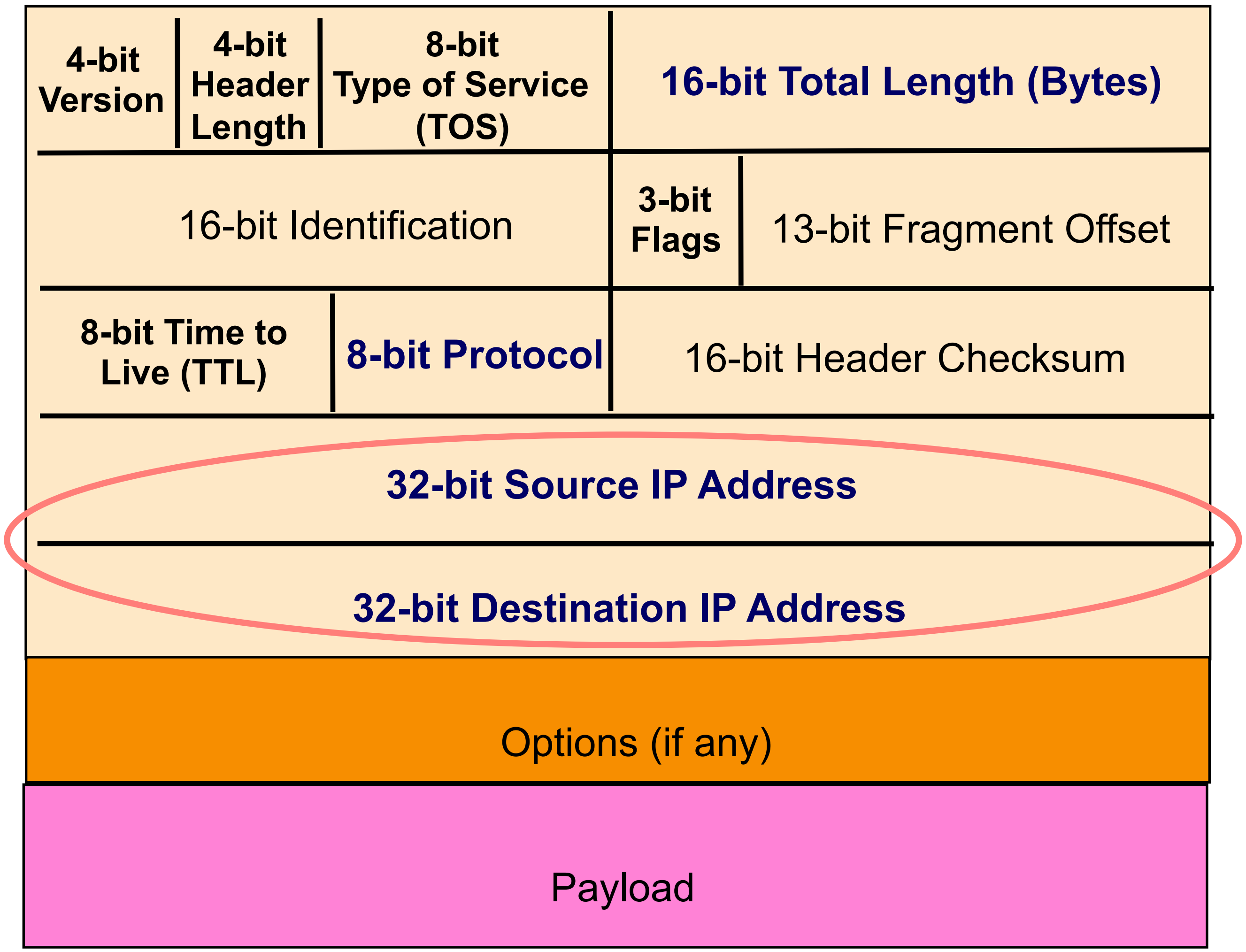
IP Packet Structure



IP Packet Structure



IP Packet Structure



IP Packet Header (Continued)

- Two IP addresses
  - Source IP address (32 bits in main IP version)
  - 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