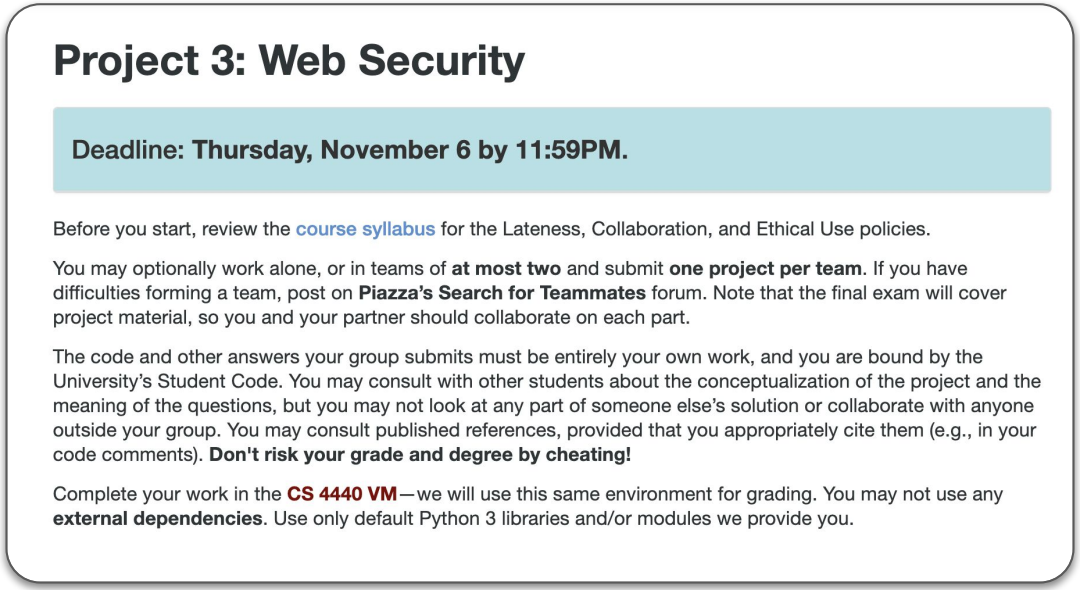
Week 10: Lecture A
Application-layer Network Attacks

Tuesday, October 28, 2025

Announcements

- **Project 3: WebSec** released
  - **Deadline:** Thursday, November 6th by 11:59PM (**next week**)



Project 3: Web Security

Deadline: Thursday, November 6 by 11:59PM.

Before you start, review the course syllabus for the Lateness, Collaboration, and Ethical Use policies.

You may optionally work alone, or in teams of **at most two** and submit **one project per team**. If you have difficulties forming a team, post on **Piazza's Search for Teammates** forum. Note that the final exam will cover project material, so you and your partner should collaborate on each part.

The code and other answers your group submits must be entirely your own work, and you are bound by the University's Student Code. You may consult with other students about the conceptualization of the project and the meaning of the questions, but you may not look at any part of someone else's solution or collaborate with anyone outside your group. You may consult published references, provided that you appropriately cite them (e.g., in your code comments). **Don't risk your grade and degree by cheating!**

Complete your work in the **CS 4440 VM**—we will use this same environment for grading. You may not use any **external dependencies**. Use only default Python 3 libraries and/or modules we provide you.

Project 3 progress

Working on Part 1

0%

Finished Part 1, working on Part 2

0%

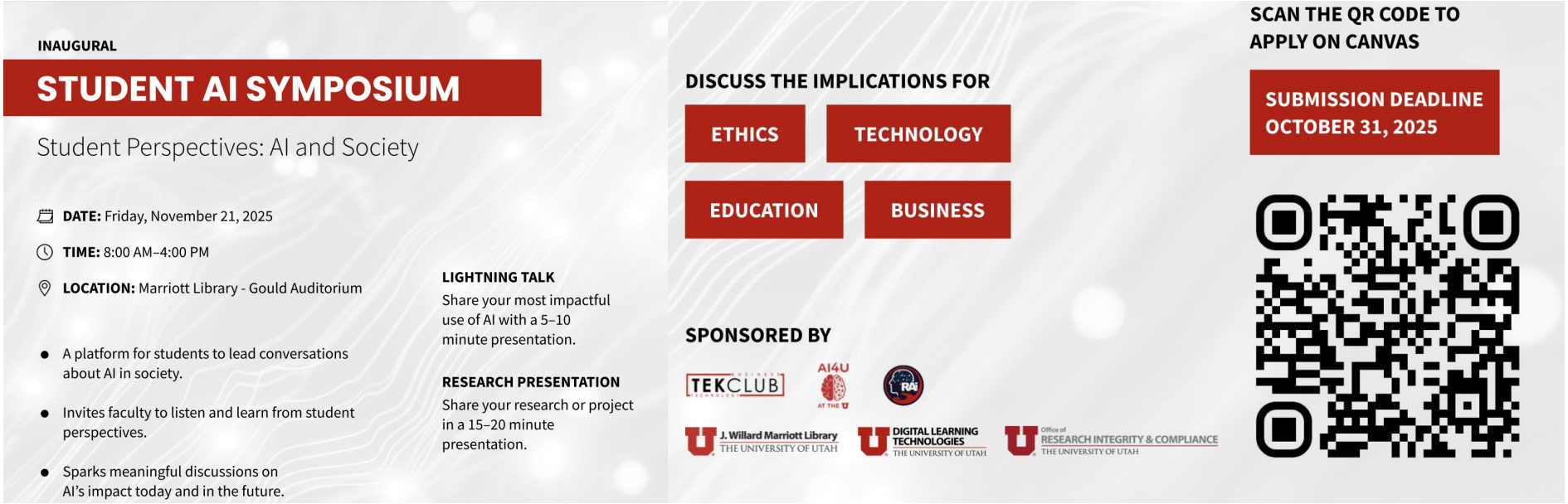
Finished Part 2, working on Part 3

0%

Haven't started :(

0%

Announcements



Announcements

utahsec

See Discord for meeting info!

utahsec.cs.utah.edu

Thanks for a successful hackathon!

Announcements

- **Project 2** regrades now posted
  - Questions? See me after class

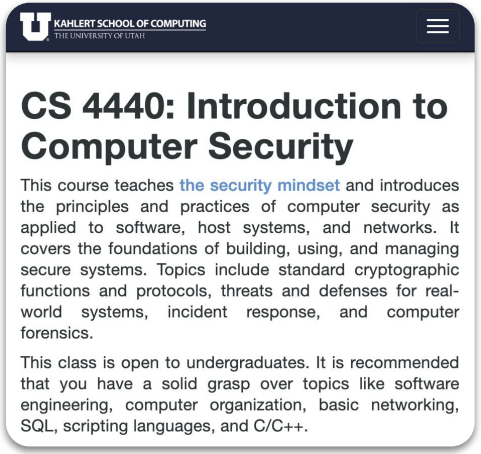
Questions?

Last time on CS 4440...

Introduction to Networking
The Physical, Link, Network,
Transport, and Application Layers

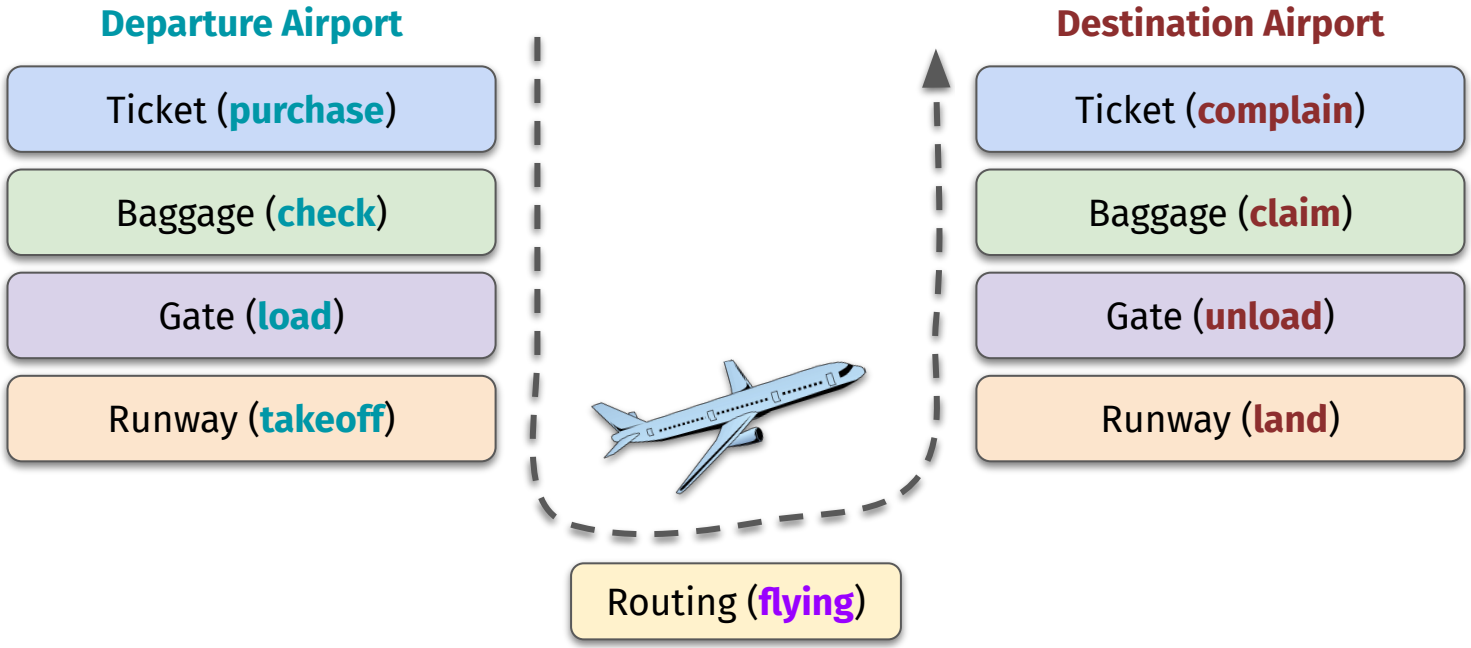
What is the Internet?

- **What is it?**
  - How you trash-talk players in COD game lobbies
  - How Wall Street trades shares faster than you
  - How the CS 4440 website is distributed to you

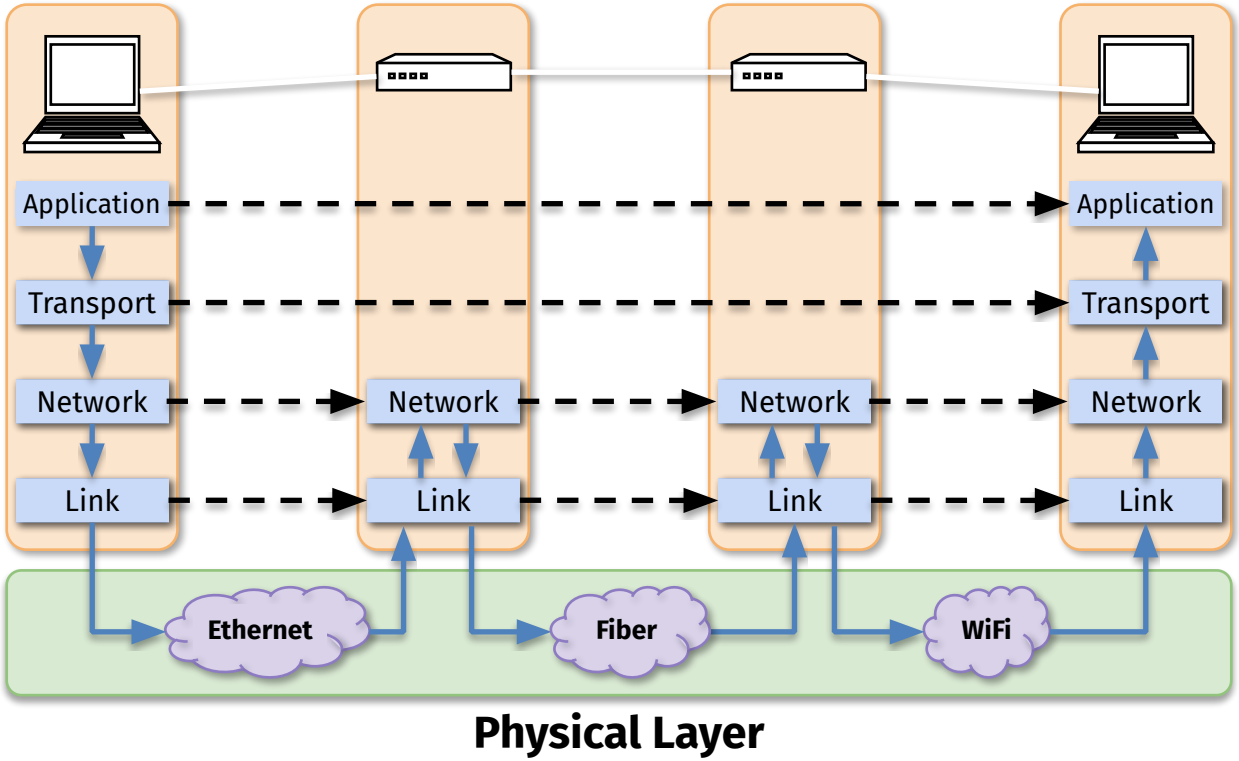


What really is the internet?

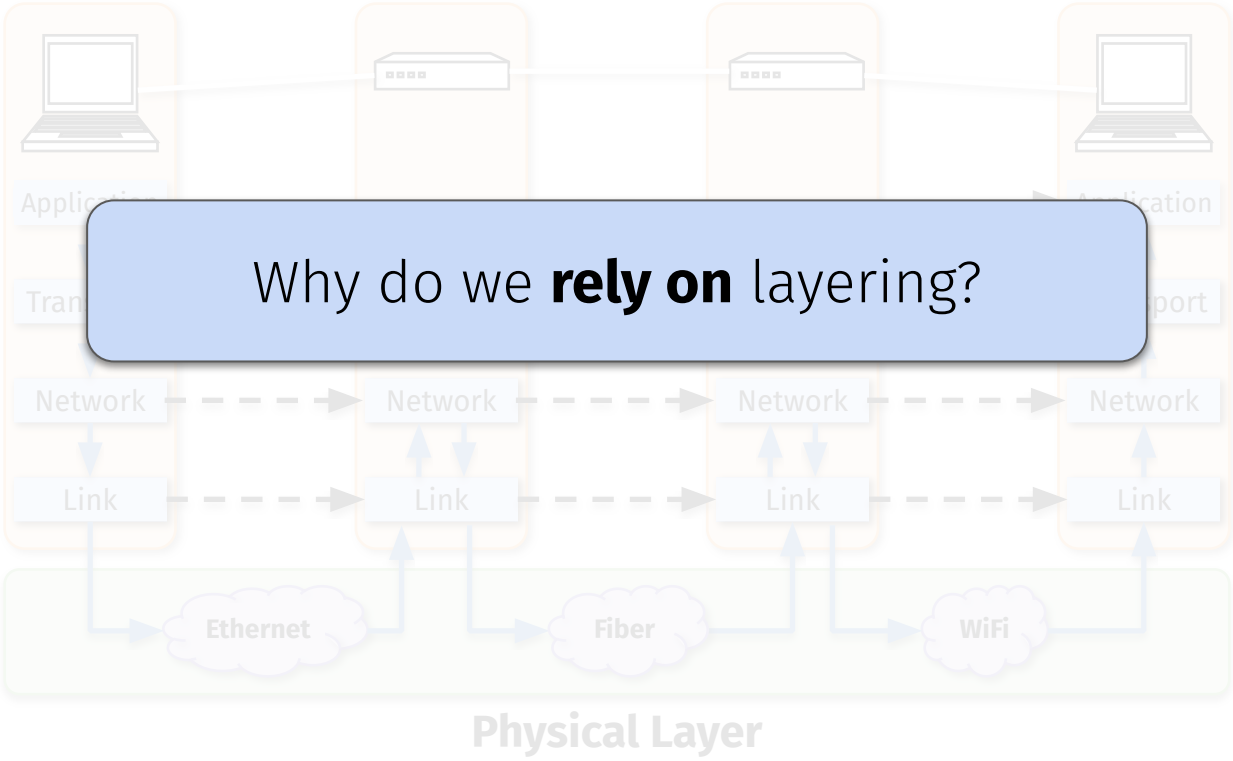
- **A group of layers—each implementing a service**



The 5-layer Internet

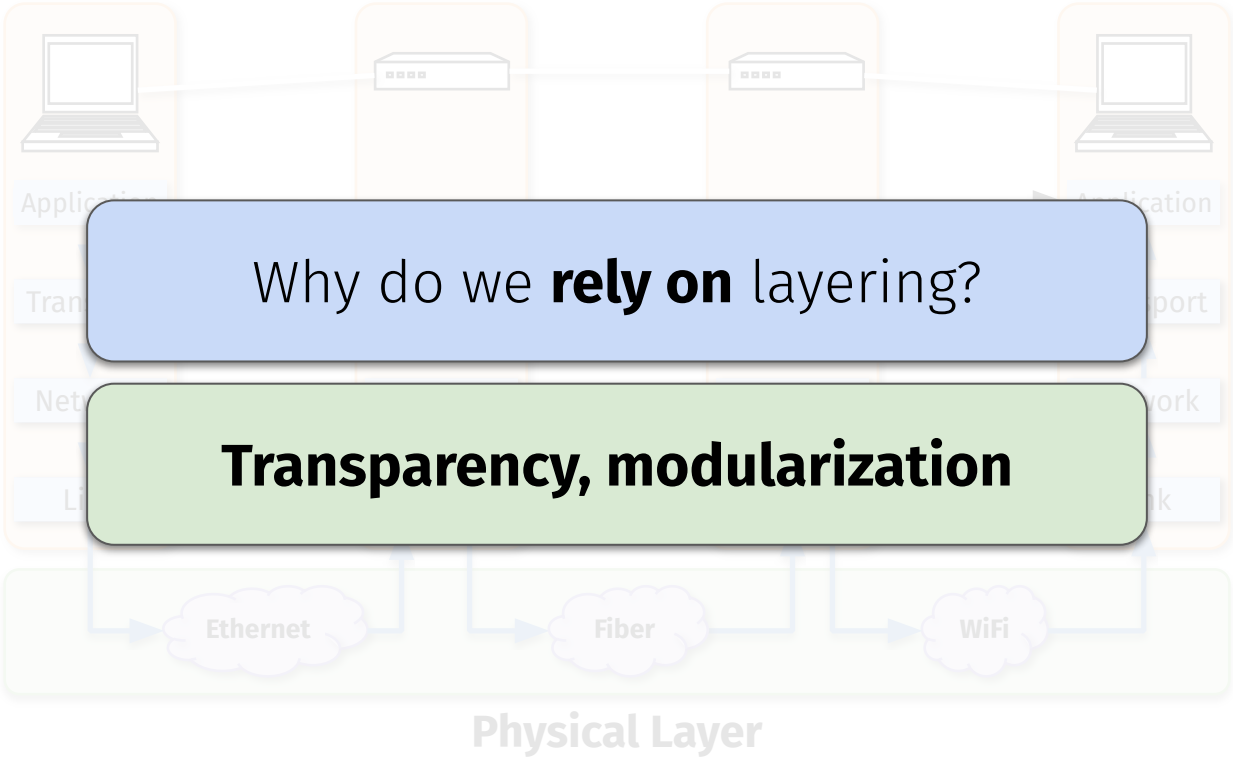


The 5-layer Internet



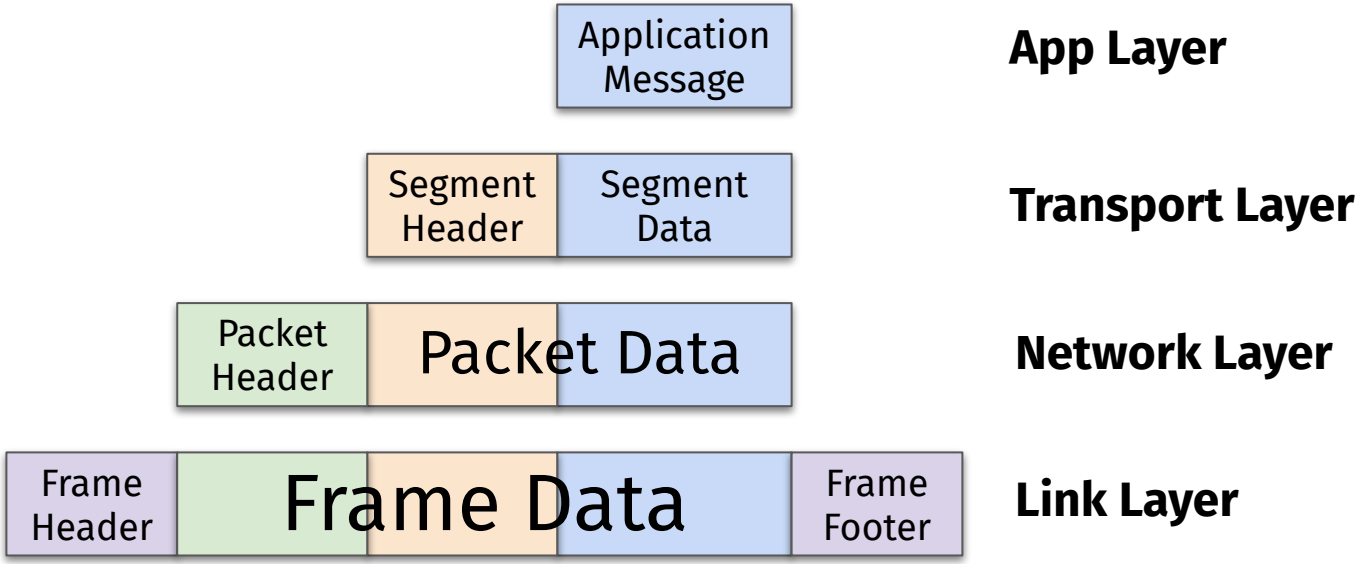
Why do we **rely on** layering?

The 5-layer Internet



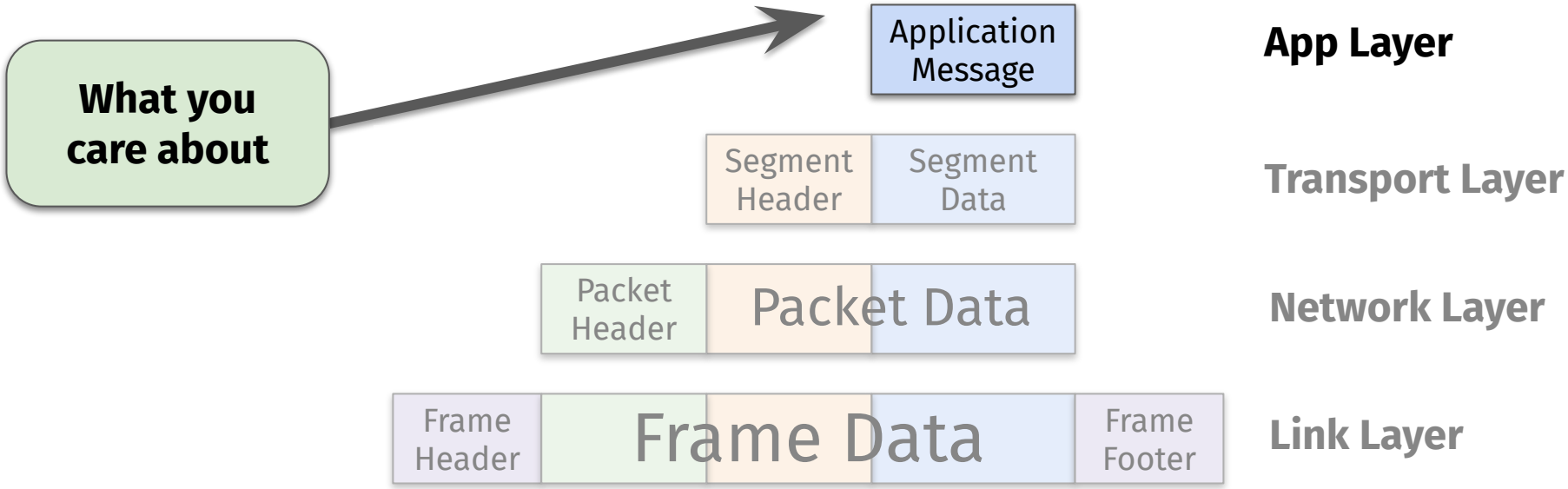
Internet Packet Encapsulation

- How packets are generated and sent



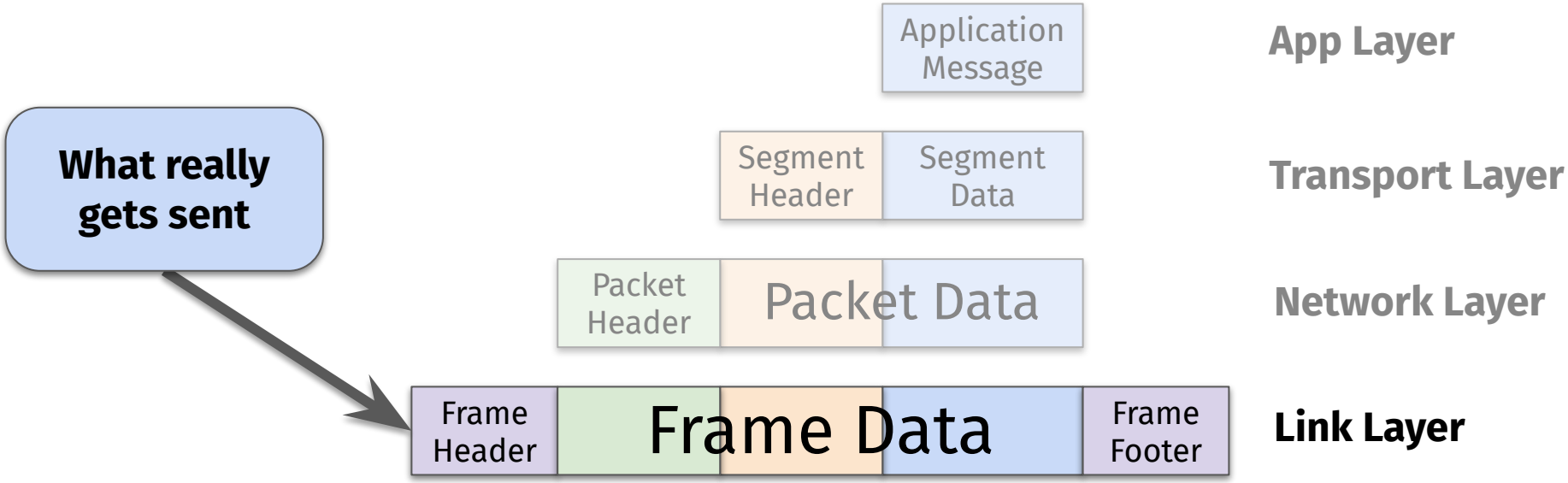
Internet Packet Encapsulation

- How packets are generated and sent



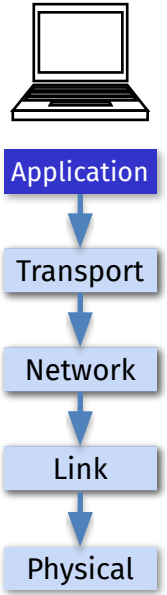
Internet Packet Encapsulation

- How packets are generated and sent



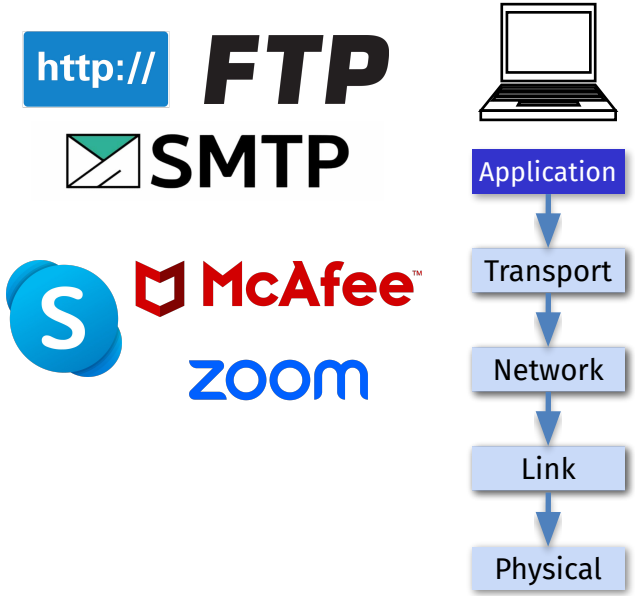
The Application Layer

- **What is it?**
  - ???



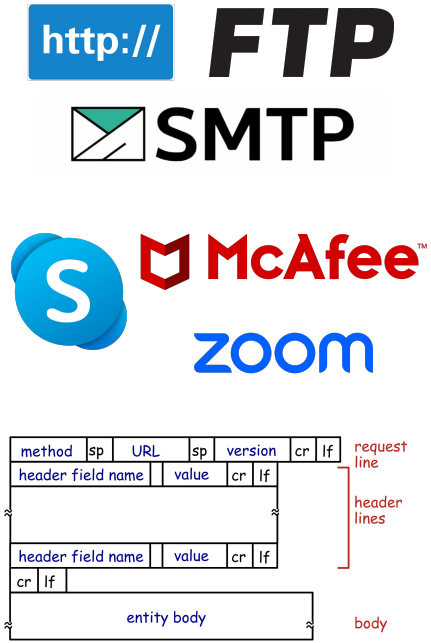
The Application Layer

- **What is it?**
  - The **top-most layer** in the 5-layer network model
  - Where applications send and receive messages
- What does it **define**? **Application protocols**
  - **???**



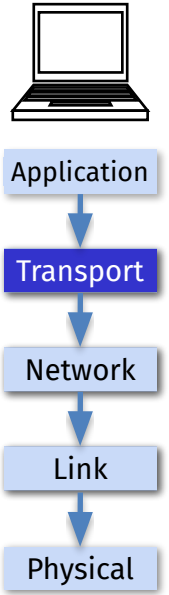
The Application Layer

- **What is it?**
  - The **top-most layer** in the 5-layer network model
  - Where applications send and receive messages
- What does it **define**? **Application protocols**
  - Message **types**
    - What is the purpose of this message?
  - Message **syntax**
    - How should this message be structured?
  - Message **semantics**
    - What does each message field really mean?
  - Rules for **sending/receiving** messages
    - When/how should this application respond?



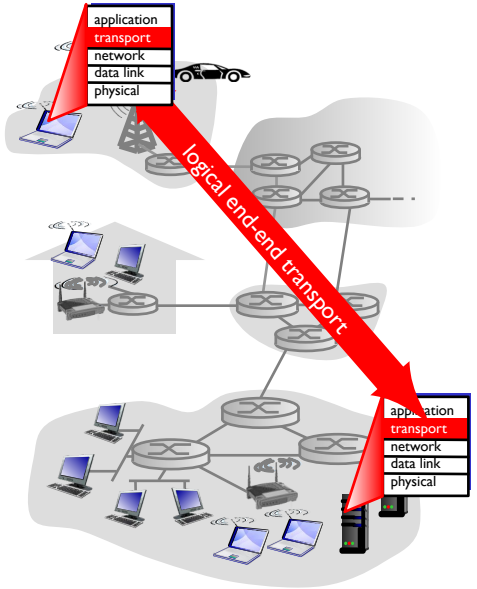
The Transport Layer

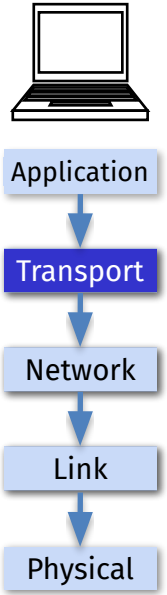
- **What is it?**
  - ???



The Transport Layer

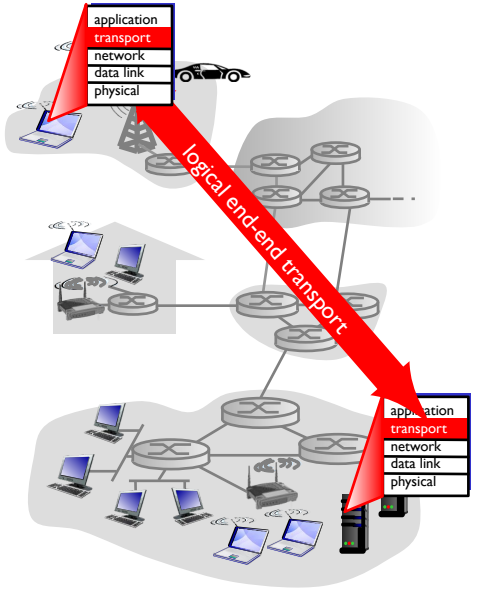
- **What is it?**
  - The **second layer** in the 5-layer network model
  - Communication between apps on different hosts
- What are its two main **protocols**?
  - **???**

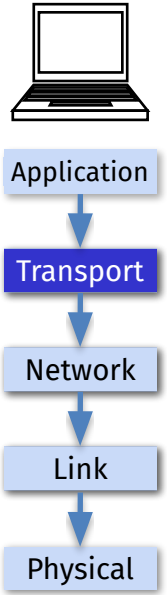




The Transport Layer

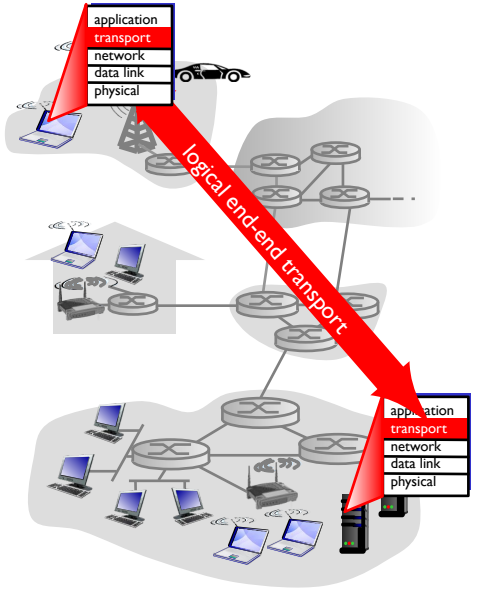
- **What is it?**
  - The **second layer** in the 5-layer network model
  - Communication between apps on different hosts
- What are its two main **protocols**? **TCP**, **UDP**
  - **TCP**—Transmission Control Protocol
    - **Characteristics: slow/complex** but **reliable**
  - **UDP**—User Datagram Protocol
    - **Characteristics: fast/simple** but **unreliable**
- What are **ideal use cases** for TCP and UDP?
  - **???**

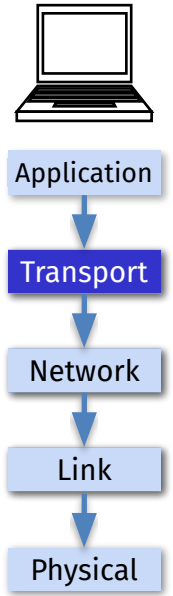




The Transport Layer

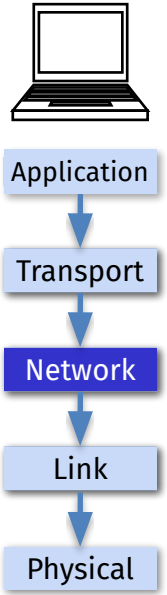
- **What is it?**
  - The **second layer** in the 5-layer network model
  - Communication between apps on different hosts
- What are its two main **protocols**? **TCP**, **UDP**
  - **TCP**—Transmission Control Protocol
    - **Characteristics: slow/complex** but **reliable**
  - **UDP**—User Datagram Protocol
    - **Characteristics: fast/simple** but **unreliable**
- What are **ideal use cases** for TCP and UDP?
  - **TCP:** reliability matters (file transfer, SSH, e-mail)
  - **UDP:** speed matters (video calls, gaming, livestream)





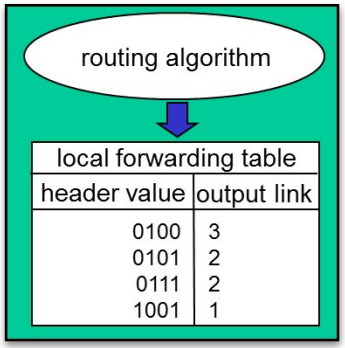
The Network Layer

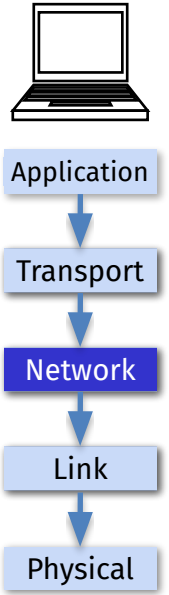
- **What is it?**
  - ???



The Network Layer

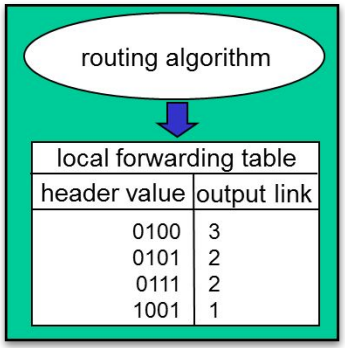
- **What is it?**
  - The **third layer** in the 5-layer network model
  - Sends data from host on one network to another
- What are its **two functions**?
  - **???**

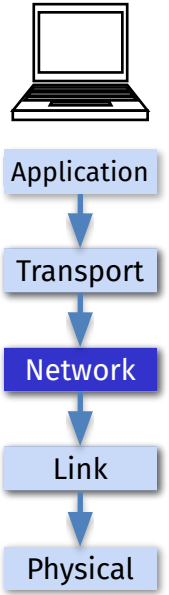




The Network Layer

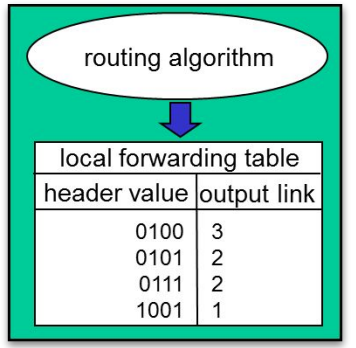
- **What is it?**
  - The **third layer** in the 5-layer network model
  - Sends data from host on one network to another
- What are its **two functions**? **Routing**, **forwarding**
  - **Routing:** find shortest possible path to send a packet
  - **Forwarding:** sending packets on to the next hop
- What is its **addressing** based on?
  - **???**

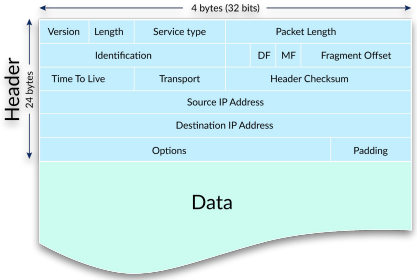


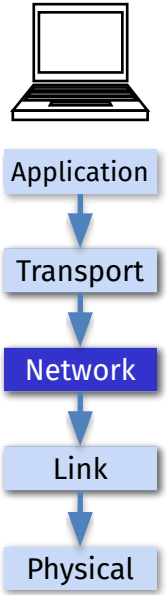


The Network Layer

- **What is it?**
  - The **third layer** in the 5-layer network model
  - Sends data from host on one network to another
- What are its **two functions**? **Routing**, **forwarding**
  - **Routing:** find shortest possible path to send a packet
  - **Forwarding:** sending packets on to the next hop
- What is its **addressing** based on?
  - IP addresses—a logical address
    - Network-internal IP assigned by your router
    - Public IP assigned by Internet Service Provider

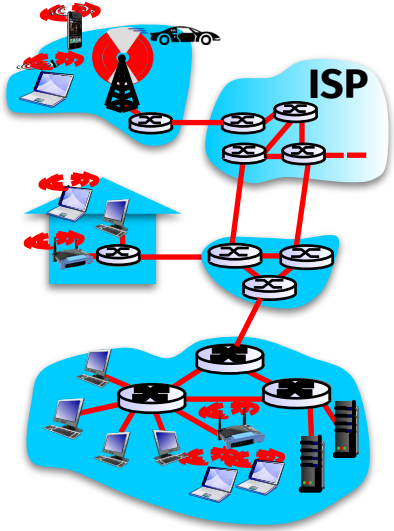


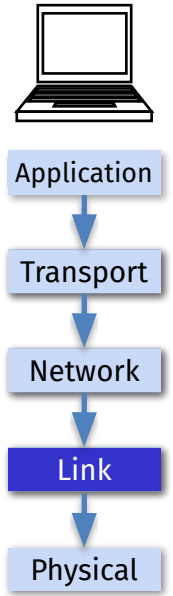




The Data Link Layer

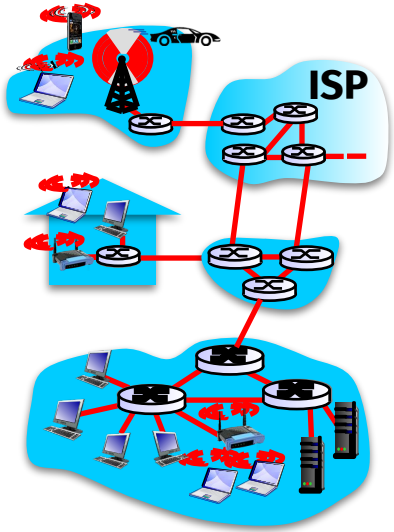
- **What is it?**
  - ???

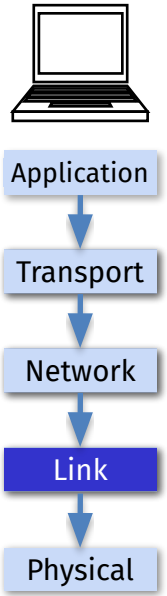




The Data Link Layer

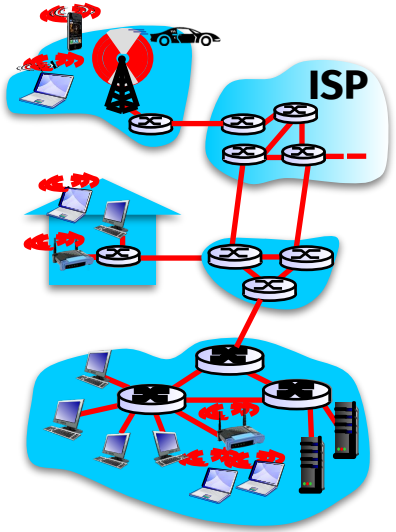
- **What is it?**
  - The **fourth layer** in the 5-layer network model
  - Responsible for the node-to-node delivery of data
- What are **“nodes”**?
  - **???**

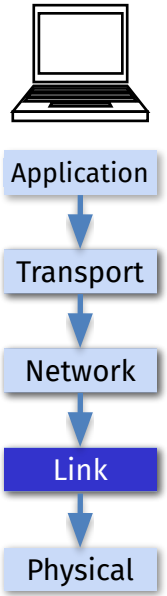




The Data Link Layer

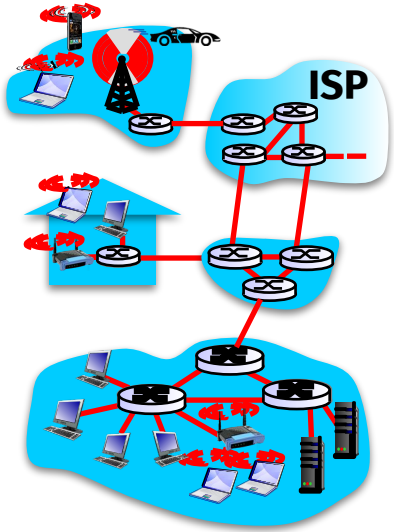
- **What is it?**
  - The **fourth layer** in the 5-layer network model
  - Responsible for the node-to-node delivery of data
- What are **"nodes"**? **Hosts**, **switches**
  - **Hosts:** the physical devices within a network
  - **Switches:** interface to all hosts on the network
- What is its **addressing** based on?
  - **???**

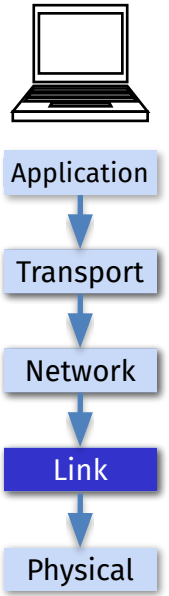




The Data Link Layer

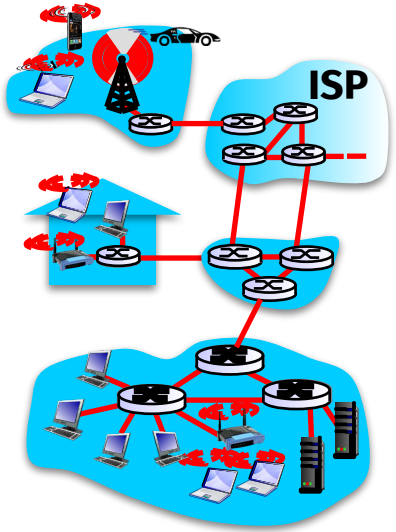
- **What is it?**
  - The **fourth layer** in the 5-layer network model
  - Responsible for the node-to-node delivery of data
- What are **"nodes"**? **Hosts**, **switches**
  - **Hosts:** the physical devices within a network
  - **Switches:** interface to all hosts on the network
- What is its **addressing** based on?
  - MAC addresses—a physical identifier for hardware
- Do MAC addresses guarantee **authenticity**?
  - **???**

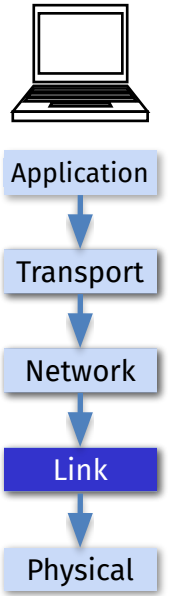




The Data Link Layer

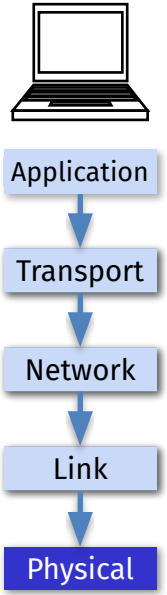
- **What is it?**
  - The **fourth layer** in the 5-layer network model
  - Responsible for the node-to-node delivery of data
- What are **"nodes"**? **Hosts**, **switches**
  - **Hosts:** the physical devices within a network
  - **Switches:** interface to all hosts on the network
- What is its **addressing** based on?
  - MAC addresses—a physical identifier for hardware
- Do MAC addresses guarantee **authenticity**?
  - Reconfigurable via network interface
  - **Attacker-spoofable**





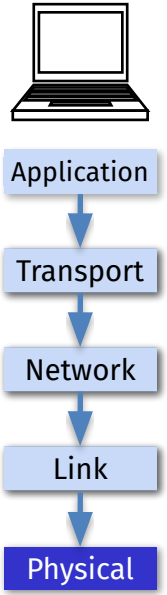
The Physical Layer

- **What is it?**
  - ???



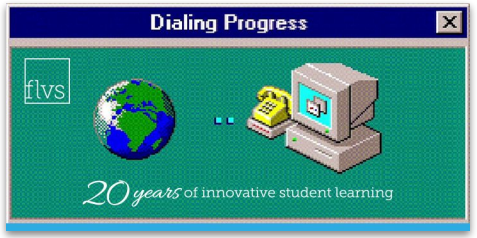
The Physical Layer

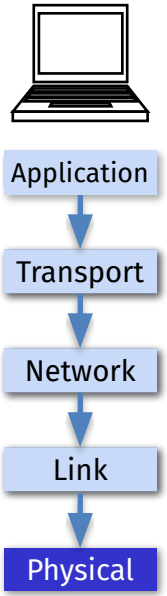
- **What is it?**
  - The **last layer** in the 5-layer network model
  - The physical means of sending/receiving data
- **Examples** of physical layers?
  - ???



The Physical Layer

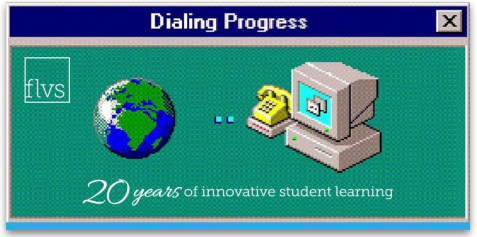
- **What is it?**
  - The **last layer** in the 5-layer network model
  - The physical means of sending/receiving data
- **Examples** of physical layers?
  - Radio waves
  - Telephone lines
  - Fiber optic cables
  - Undersea submarine cables
- Does physical layer guarantee **availability**?
  - **???**

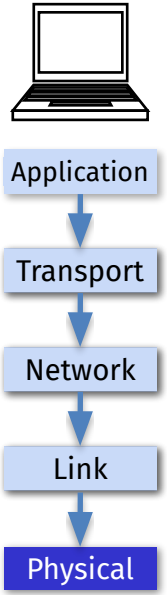




The Physical Layer

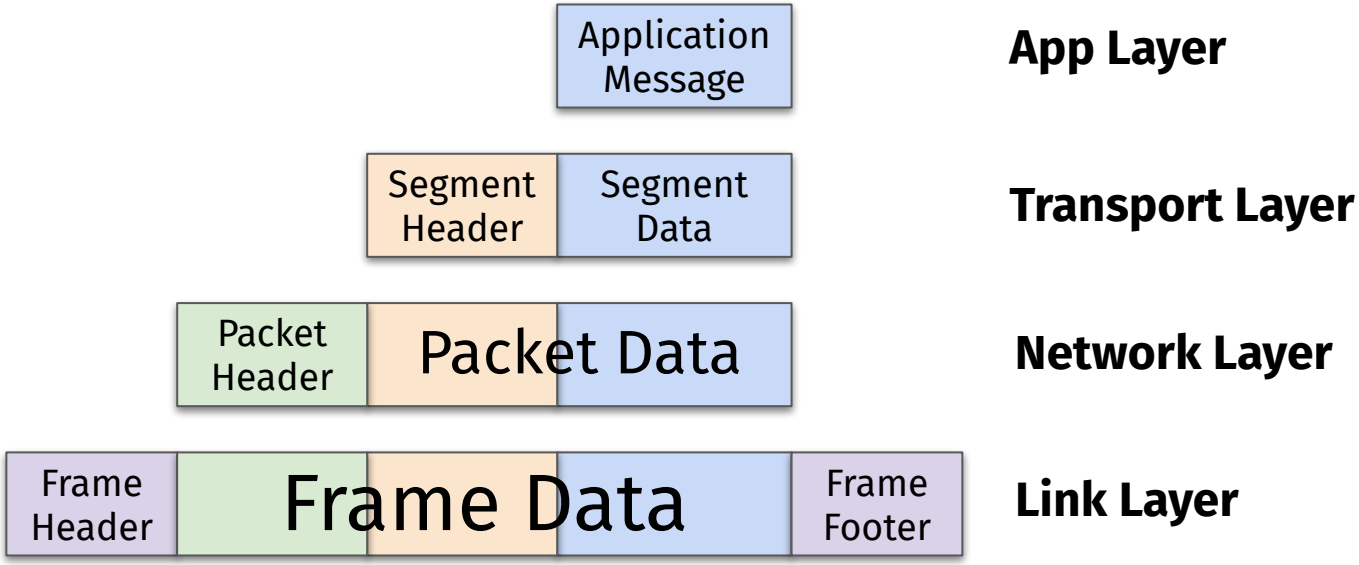
- **What is it?**
  - The **last layer** in the 5-layer network model
  - The physical means of sending/receiving data
- **Examples** of physical layers?
  - Radio waves
  - Telephone lines
  - Fiber optic cables
  - Undersea submarine cables
- Does physical layer guarantee **availability**?
  - **No—tamperable by third parties!**





Food for Thought

- Are any of the five network layers susceptible to **attacks**? If so, **which ones**?



Which network layers are susceptible to attack?

Physical 0%

Link 0%

Network 0%

Transport 0%

Application 0%

None of the above 0%

All of the above 0%

Food for Thought

Every network layer is susceptible to **attack**

Today's focus: **Application Layer** attacks

Thursday's focus: attacking the **other layers**

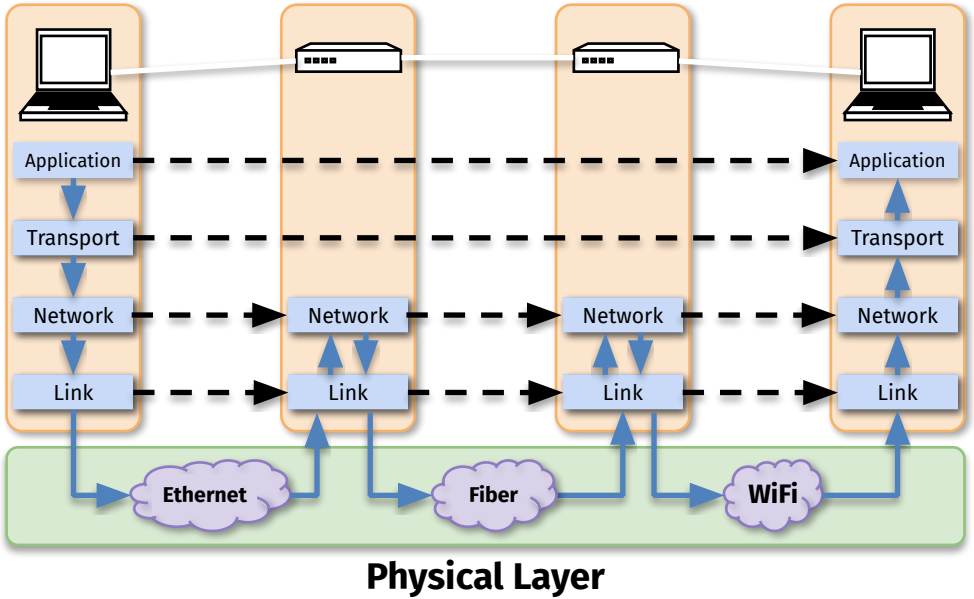
Questions?

This time on CS 4440...

Application Layer Attacks
HTTP Content Injection
SMTP Header Spoofing
DNS Hijacking
Network Packet Analysis

Recap: The 5-Layer Internet

- **Application Layer:**
  - Sends/receives app messages
- **Transport Layer:**
  - Communication between apps
- **Network Layer:**
  - Communication between hosts
- **Data Link Layer:**
  - Node-to-node delivery of data
- **Physical Layer:**
  - Send/receive the physical signals

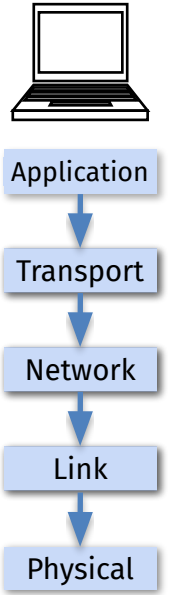


Where is each layer implemented?

- **Application Layer:** ???
- **Transport Layer:** ???
- **Network Layer:** ???
- **Data Link Layer:** ???

Quiz:

1. **Host Device** (e.g., your laptop)
2. **Other Devices** (e.g., switch, router)
3. **Both!**

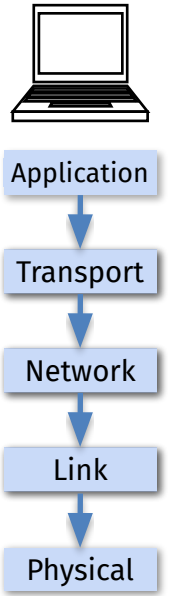


Where is each layer implemented?

- **Application Layer:** **Host**
  - E.g., browser, socket-based app
- **Transport Layer:** **Host**
  - OS library and necessary drivers
- **Network Layer:** **Both**
  - Within every host, router, switch
- **Data Link Layer:** **Host**
  - Network interface card, ethernet

Quiz:

1. **Host Device** (e.g., your laptop)
2. **Other Devices** (e.g., switch, router)
3. **Both!**



Where is each layer implemented?

- **Application Layer:** Host
  - E.g., browser, socket-based app
- **Transport Layer:** Host
  - OS library
- **Network Layer:** Both
  - Within every host, router, switch
- **Data Link Layer:** Host
  - Network interface card, ethernet

Quiz:

1. Host
2. Non-host
   (e.g., switch, router)
3. Both!

Why implement so much in **hosts**?

Where is each layer implemented?

- **Application Layer:** Host
  - E.g., browser, socket-based app
- **Transport L**
  - OS librar
- **Network La**
  - Within ev
- **Data Link Layer:** Host
  - Network interface card, ethernet

Quiz:

Why implement so much in **hosts**?

The End-to-End Principle

The End-to-End Principle

- Based on Paul Baran's 1960's work "reliability from unreliable parts"

- **Key idea:**
  - **Application-specific functions** ought to reside in the **end hosts** of a network
  - Rather than in **intermediary nodes**
  - All this assumes that the end host can implement it "completely and correctly"

The End-to-End Principle

- Based on Paul Baran's 1960's work "reliability from unreliable parts"

- **Key idea:**
  - **Application-specific functions** ought to reside in the **end hosts** of a network
  - Rather than in **intermediary nodes**
  - All this assumes that the end host can implement it "completely and correctly"

Specific

General-purpose

Specific

The End-to-End Principle

- Based on Paul Baran's 1960's work "reliability from unreliable parts"
- **Key idea:**
  - **Application-s**
    reside in the **end hosts** of a network
  - Rather than in **intermediary nodes**
  - All this assumes that the end host can implement it "completely and correctly"

Why not move more functionality to the **intermediate nodes**?

Specific

General-purpose

Specific

The End-to-End Principle

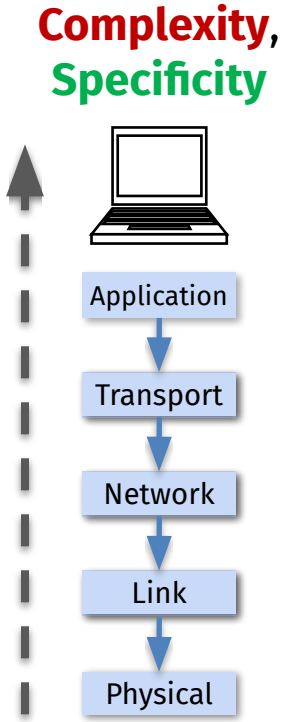
- Based on Paul Baran's 1960's work
  "reliability from unreliable parts"
- **Key idea:**
  - **Application-**
    reside in the **end hosts** of a network
  - Rather than
  - All this assu
    implement it

Why not move more functionality to the **intermediate nodes**?

More **latency**, less **flexibility**, higher-**complexity** midpoints

Implications of The E2E Principle

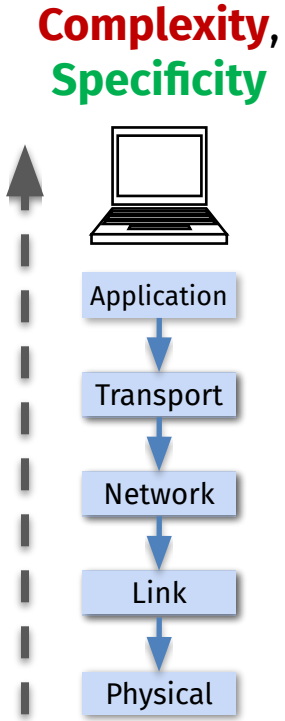
- **Nothing** guaranteed by default
  - No default **integrity**
  - No default **confidentiality**
  - No default **availability**
  - No default **authentication**



Implications of The E2E Principle

- **Nothing** guaranteed by default
  - No default **integrity**
  - No default **confidentiality**
  - No default **availability**
  - No default **authentication**

- **Solution: bolt-on** more security!
  - **IPsec** (Network layer)
    - Integrity, Confidentiality, Authentication
  - **DNSSec** (App layer)
    - Integrity, Authenticity
  - **TLS** (Session layer)
    - Integrity, Confidentiality, Authentication



Application Layer Network Attacks

Application Layer Attacks

- **Application Layer:** where **network-facing apps** send/receive message
  - Application-specific protocols (message semantics, structure, processing rules, etc.)
- **Attacking the application layer:**
  - **???**

Application Layer Attacks

- **Application Layer:** where **network-facing apps** send/receive message
  - Application-specific protocols (message semantics, structure, processing rules, etc.)

- **Attacking the application layer:**
  - **Command Injection**
    - SQL injection, CSRF, XSS
  - **Denial of Service**
    - Crash a remote application
    - Prevent others from using it
  - **Message Tampering / Sniffing**
    - Injecting data into messages
    - Capturing unencrypted data

Application Layer Attacks

- **Application Layer:** where **network-facing apps** send/receive message
  - Application-specific protocols (message semantics, structure, processing rules, etc.)

- **Attacking the application layer:**
  - **Command Injection**
    - SQL injection, CSRF, XSS
  - **Denial of Service**
    - Crash a remote application
    - Prevent others from using it
  - **Message Tampering / Sniffing**
    - Injecting data into messages
    - Capturing unencrypted data
  - **Other protocol-specific attacks**

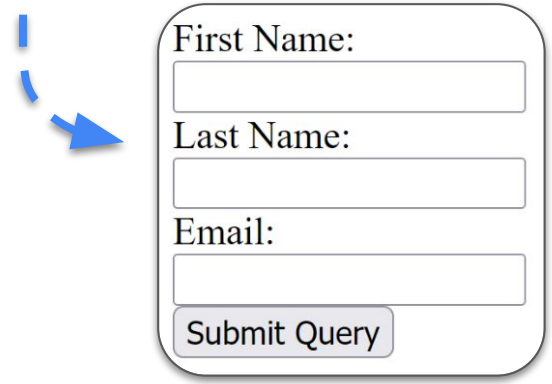
http:// FTP SMTP McAfee zoom

Application Layer Attacks
HTTP Content Injection

Recap: HyperText Markup Language (HTML)

- **What is HTML?**

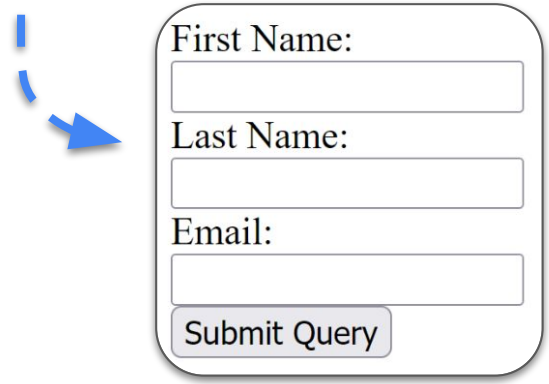
```
<form action="home.html">
    First Name:<br>
    <input type="text" name="first_name">
</br>
    Last Name:<br>
    <input type="text" name="last_name">
</br>
    Email:<br>
    <input type="text" name="email">
</br>
    <input type="submit" name="Submit">

</form>
```



Recap: HyperText Markup Language (HTML)

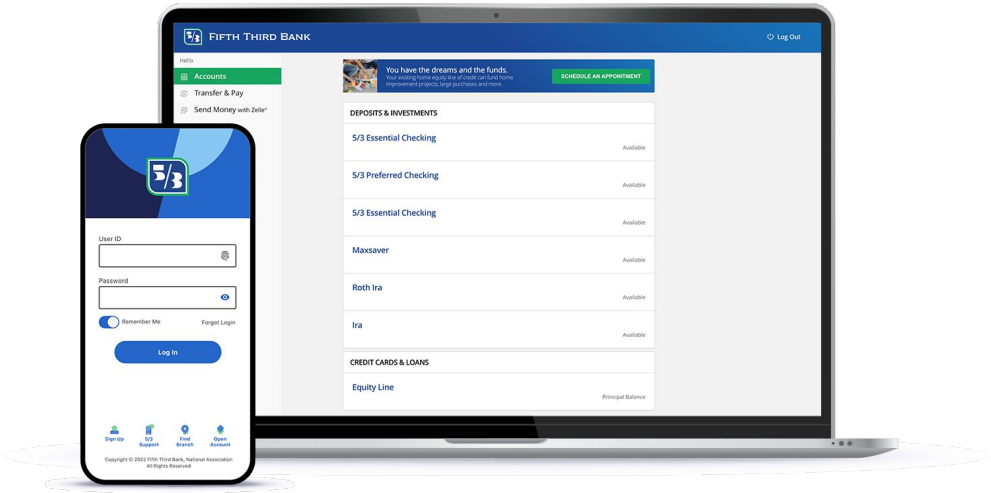
- **What is HTML?**
  - Describes **content** and **formatting** of web pages
  - Rendered within browser window
- **HTML features**
  - **Static** document description language
  - Links to external pages, images by **reference**
  - User input sent to server via **forms**
- **HTML extensions**
  - Additional media (e.g., PDF, videos) via **plugins**
  - Embedding **programs** in other languages (e.g., **Java**) provides **dynamic content** that can**:**
    - Interacts with the user
    - Modify the browser user interface
    - Access the client computer environment

```
<form action="home.html">
    First Name:<br>
    <input type="text" name="first_name">
</br>
    Last Name:<br>
    <input type="text" name="last_name">
</br>
    Email:<br>
    <input type="text" name="email">
</br>
    <input type="submit" name="Submit">

</form>
```



Recap: HyperText Transfer Protocol (HTTP)

- **What is HTTP?**



What is HTTP?

How we transmit hyper-media objects over the web!

0%

An unencrypted, stateless protocol for transmitting information from server to client (and back).

0%

Halloween Toblerone transfer protocol 🎃🍫

0%

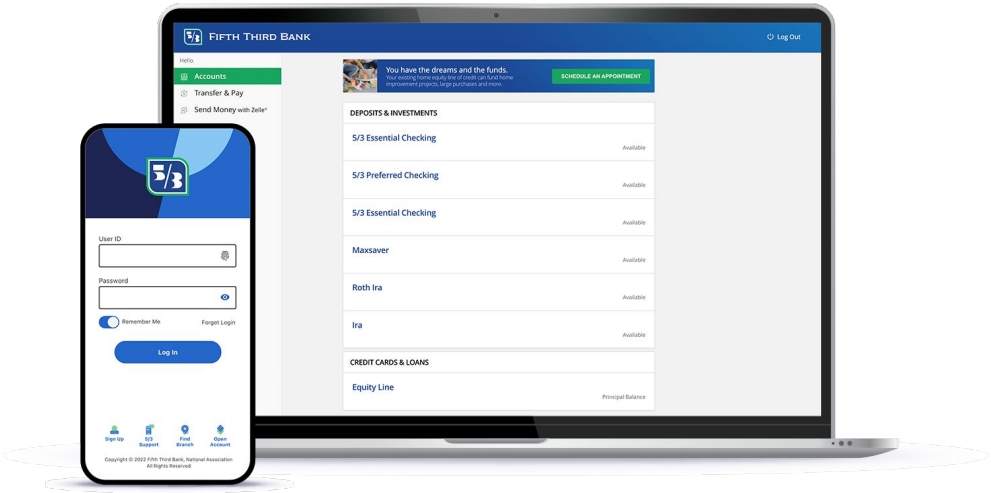
None of the above

0%

Recap: HyperText Transfer Protocol (HTTP)

- **What is HTTP?**
  - Protocol for transmitting **hypermedia documents** (e.g., web pages)

- **HTTP's Characteristics:**
  - Widely used
  - **Simple**
  - **Unencrypted**

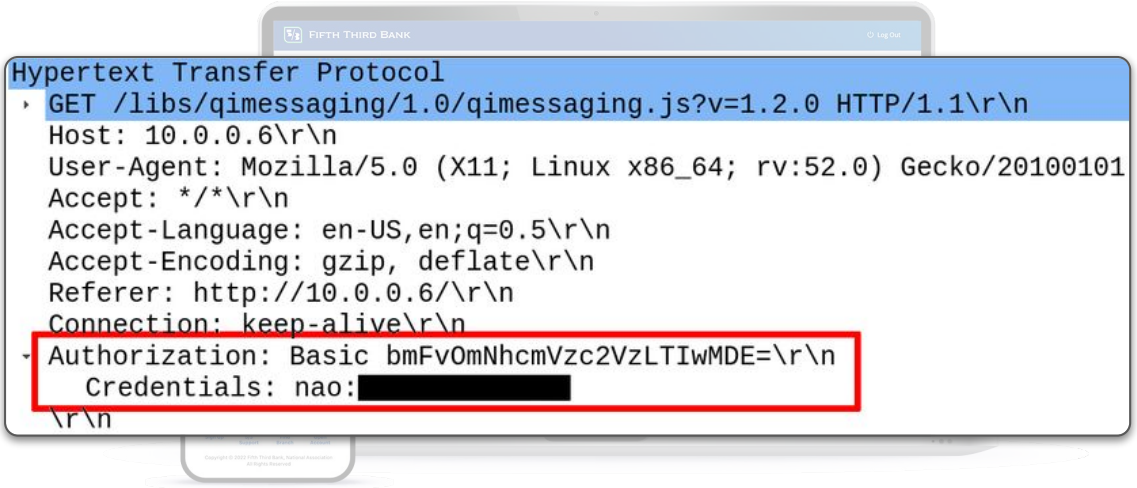


Recap: HyperText Transfer Protocol (HTTP)

- **What is HTTP?**
  - Protocol for transmitting **hypermedia documents** (e.g., web pages)

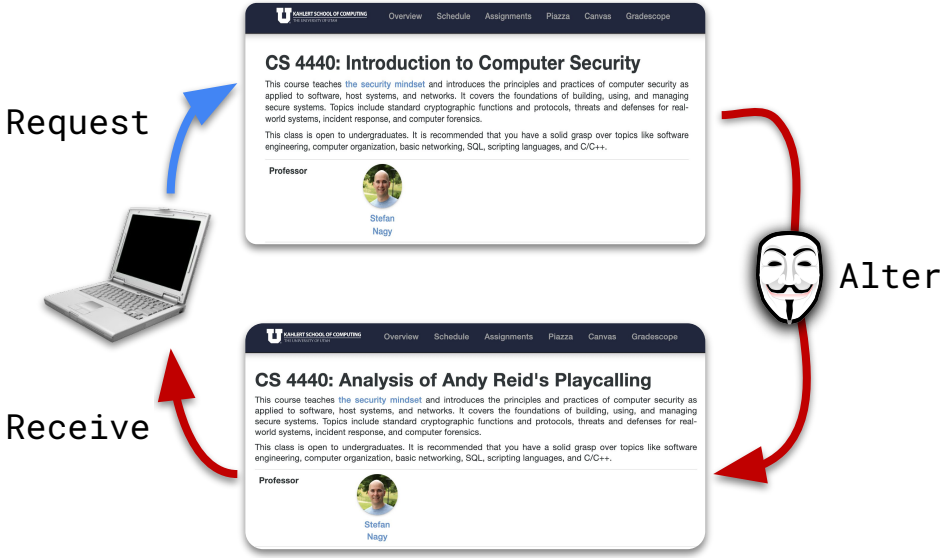
- **HTTP's Characteristics:**
  - Widely used
  - **Simple**
  - **Unencrypted**

Problem: no way of keeping data hidden from **prying eyes**!



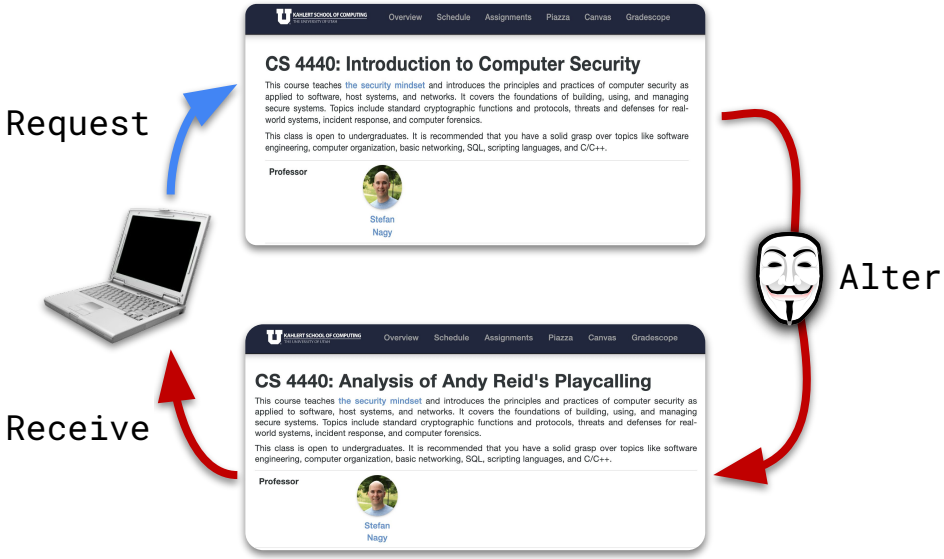
Tampering with HTTP-transmitted HTML

- Capitalizes on **HTTP's** insecurity
  - **Nothing is encrypted!**
- **Attacker intercepts** requested webpage and **modifies it**
  - User receives modified webpage
- **Attacker capabilities?**
  - **???**



Tampering with HTTP-transmitted HTML

- Capitalizes on **HTTP's** insecurity
  - **Nothing is encrypted!**
- **Attacker intercepts** requested webpage and **modifies it**
  - User receives modified webpage
- **Attacker capabilities?**
  - Inject malicious **content**
  - Inject malicious **code**



Case Study: Content Injection

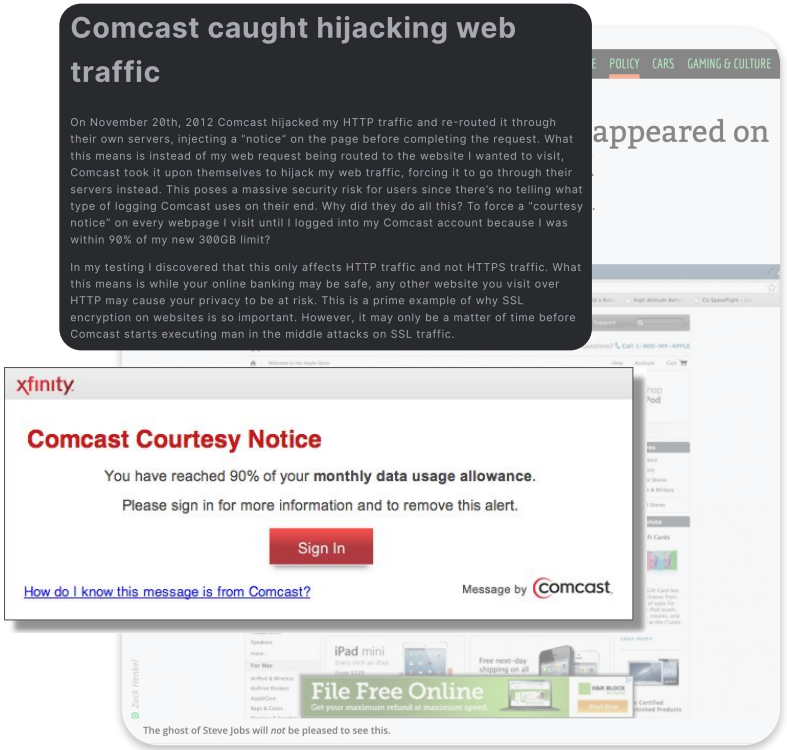
- Do you trust your **ISP**?
  - Could they tamper with data?

Case Study: Content Injection

- Do you trust your **ISP**?
  - Could they tamper with data?
- **Attack:** ISP intercepts HTML pages transmitted via **HTTP**
  - Injects advertisements
    - They make commission

Case Study: Content Injection

- Do you trust your **ISP**?
  - Could they tamper with data?
- **Attack:** ISP intercepts HTML pages transmitted via **HTTP**
  - Injects advertisements
    - They make commission
  - Injects pop-up messages
    - E.g., monthly data usage



Case Study: Content Injection

- Do you trust your **ISP**?
  - Could they tamper with data?
- **Attack:** ISP intercepts HTML pages transmitted via **HTTP**
  - Injects advertisements
    - They make commission
  - Injects pop-up messages
    - E.g., monthly data usage
  - Redirect search engine results
    - More commission!

Case Study: Code Injection

- Do you trust your **government**?
  - Could they tamper with data?

Case Study: Code Injection

- Do you trust your **government**?
  - Could they tamper with data?
- **Attack:** government forces ISPs to **inject code** into HTTP content
  - Steal HTTP-transmitted **passwords**
    - E.g., Facebook, GMail, Twitter

How The Tunisian Government Tried To Steal The Entire Country's Facebook Passwords

Pascal-Emmanuel Gobry
Jan 24, 2011, 10:01 AM

Tunisia is in the midst of what increasingly looks like a happy, democratic revolution. People are wondering about the role social media played in that revolution. It turns out Facebook played a great role -- for good and for bad.

AP

Case Study: Code Injection

- Do you trust your **government**?
  - Could they tamper with data?
- **Attack:** government forces ISPs to **inject code** into HTTP content
  - Steal HTTP-transmitted **passwords**
    - E.g., Facebook, GMail, Twitter
  - **Result: persistent XSS** attack
    - Passes **Same-origin Policy**!

Thwarting HTTP Injection

- How do we prevent code and content injection in HTTP-transmitted data?

Thwarting HTTP Injection

- How do we prevent code and content injection in HTTP-transmitted data?

Answer: *completely* ditch HTTP!

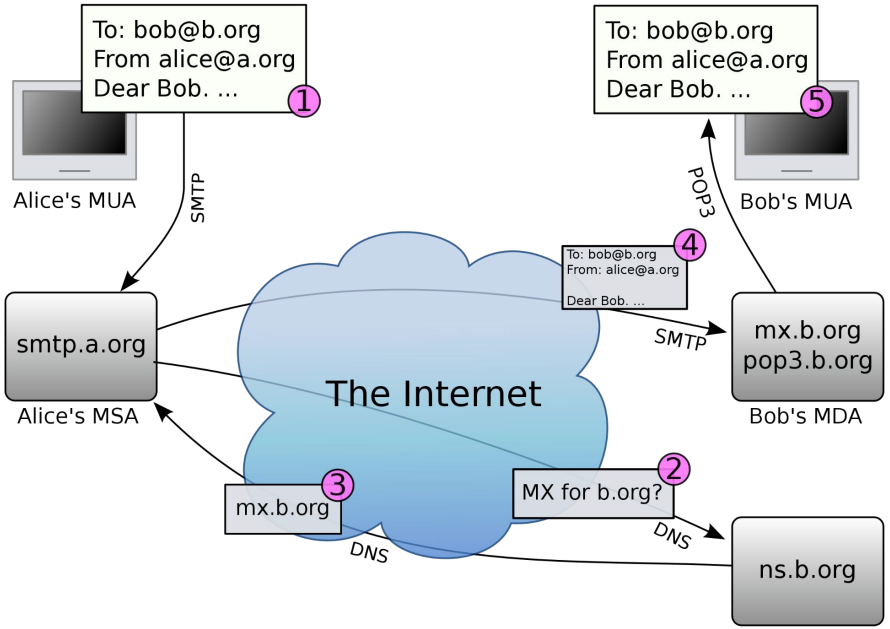
- As web and app developers, enforce strict **HTTPS** compliance
  - Necessary to prevent **HTTPS→HTTP downgrade attacks**

Questions?

Application Layer Attacks
SMTP Header Spoofing

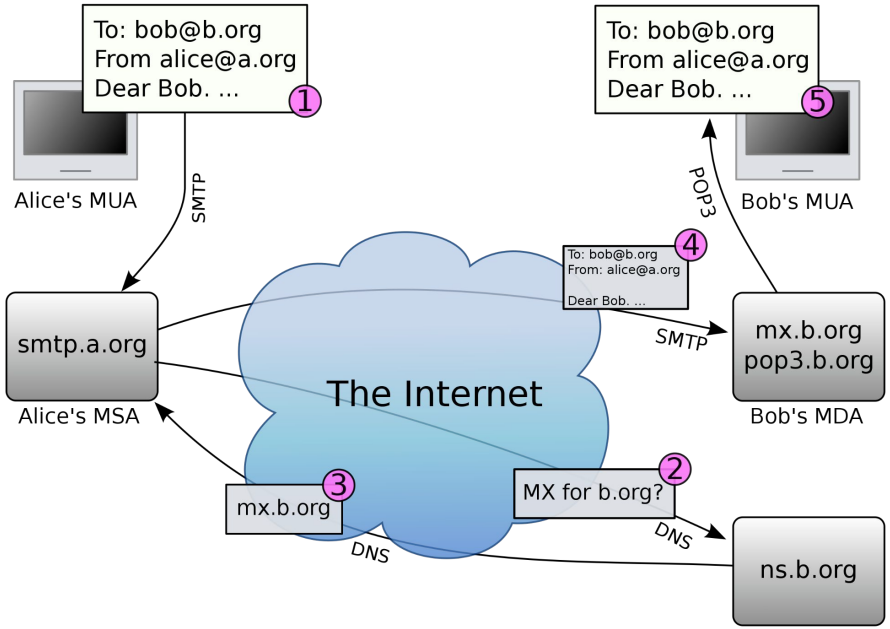
How does sending E-mail work?

- Nigerian Prince writes me a great investment opportunity
- **Sends it!**



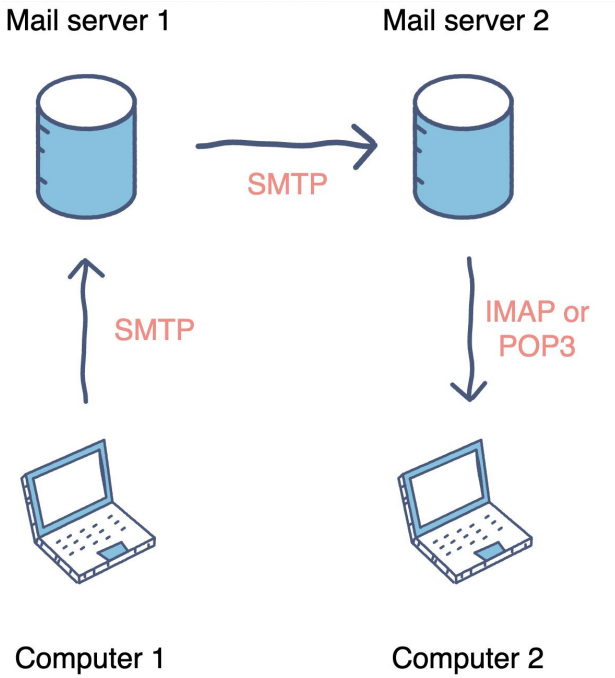
How does sending E-mail work?

- Nigerian Prince writes me a great investment opportunity
- **Sends it!**
  - Protocol **SMTP** (aka the **Simple Mail Transfer Protocol**)
  - **Sender's SMTP server** breaks up the message into body/receiver
  - **Sender's SMTP server** queries DNS to find **receiver's server IP**
  - **Receiver's SMTP server** gets msg, then queries **its POP3 server** to find the **correct user mailbox**



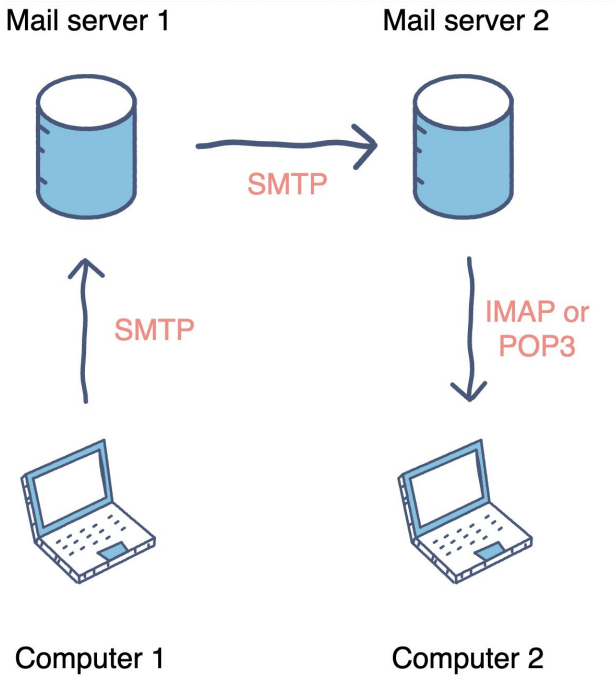
SMTP Protocol

- **SMTP:** Simple Mail Transfer Protocol
  - Implemented in the **application** layer
- **Characteristics:**
  - Text-based
  - Connection-oriented
  - Uses TCP ports 25/587



SMTP Protocol

- **SMTP:** Simple Mail Transfer Protocol
  - Implemented in the **application** layer
- **Characteristics:**
  - Text-based
  - Connection-oriented
  - Uses TCP ports 25/587
- **Security guarantees:**
  - Message integrity—**no!**
  - Confidentiality—**no!**
  - Authentication—**no!**



Example SMTP Connection

- **Plain SMTP:**
  - No encryption whatsoever
- **Key Protocol fields:**
  - `HELO`: setup sender's server
  - `MAIL FROM`: sender address
  - `RCPT TO`: recipient address
  - `DATA`: subject, body, files

```
S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM:<bob@example.org>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Bob Example" <bob@example.org>
C: To: "Alice Example" <alice@example.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 January 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Alice.
C: This is a test message with 5 header fields and 4 lines in the message body.
C: Your friend,
C: Bob
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
{The server closes the connection}
```

Example SMTP Connection

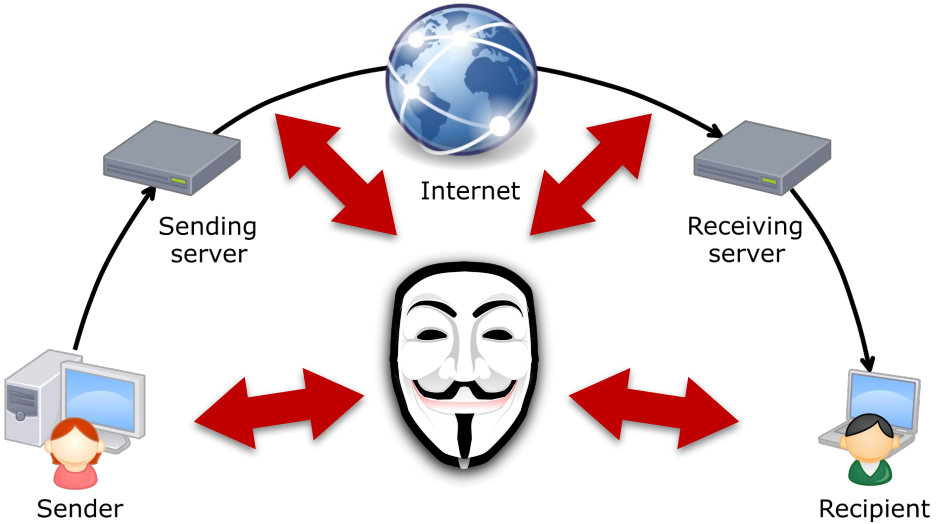
- **Plain SMTP:**
  - No encryption whatsoever
- **Key Protocol fields:**
  - `HELO`: setup sender's server
  - `MAIL FROM`: sender address
  - `RCPT TO`: recipient address
  - `DATA`: subject, body, files

What could an attacker do?

```
S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM:<bob@example.org>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Bob Example" <bob@example.org>
C: To: "Alice Example" <alice@example.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 January 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Alice.
C: This is a test message with 5 header fields and 4 lines in the message body.
C: Your friend,
C: Bob
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
{The server closes the connection}
```

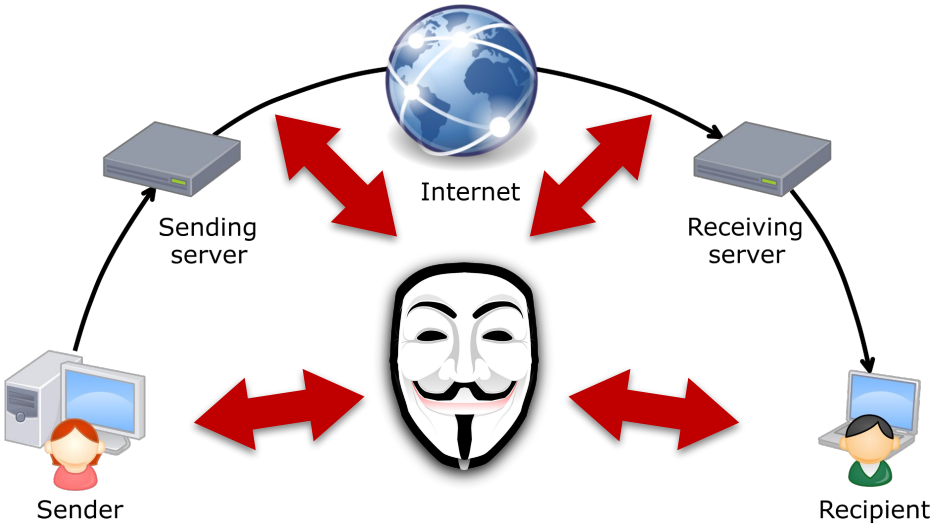
SMTP Attacks

- No **message integrity**
  - Tamper with messages
  - Block messages



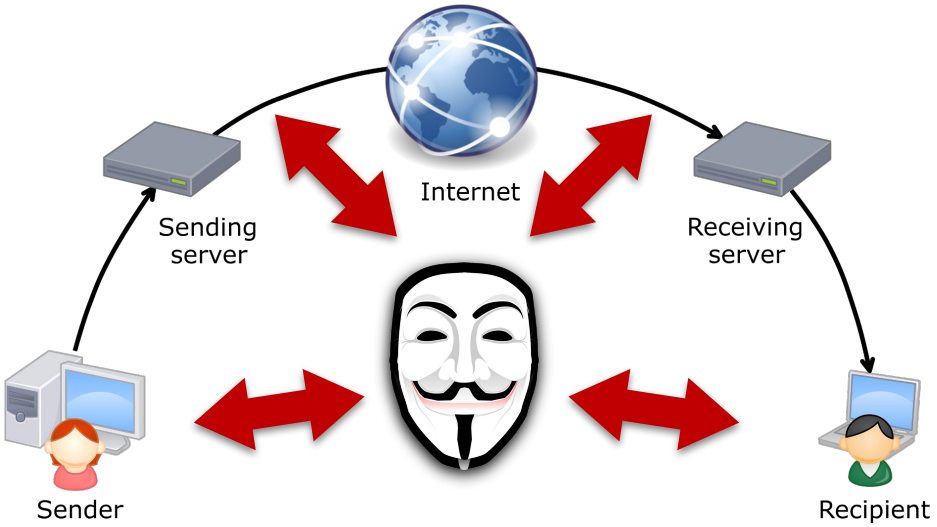
SMTP Attacks

- No **message integrity**
  - Tamper with messages
  - Block messages
- No **confidentiality**
  - Find sender/recipient
  - Read message contents



SMTP Attacks

- No **message integrity**
  - Tamper with messages
  - Block messages
- No **confidentiality**
  - Find sender/recipient
  - Read message contents
- No **authentication**
  - Spoof sender identity



Case Study: Email Header Spoofing

- **Attack:** spoof email header to **mislead recipient** about sender of the email

```
S: 220 attacker.com SMTP Exim
C: HELO attacker.com
S: 250 Hello attacker.com
C: MAIL FROM: <ceo@company.com>
S: 250 Ok
C: RCPT TO: <bob@company.com>
S: 250 Accepted
C: DATA
S: 354 Enter a message, ending with "." on a line by itself
C: Subject: Download this urgently
C: From: ceo@company.com
C: To: bob@company.com
C:
C: Hi Bob,
C: Please download this urgently: https://some-malicious-link.com
C: Regards
C: .
S: 250 OK
C: QUIT
S: 221 attacker.com closing connection
```

```
To: robertbateman@email.com
Subject: Hi There
From: "Mickey Mouse" <m.mouse@disney.com>
X-Priority: 3 (Normal)
Importance: Normal
Errors-To: m.mouse@disney.com
Reply-To: m.mouse@disney.com
Content-Type: text/plain
```

Case Study: Email Header Spoofing

- **Attack:** spoof email header to **mislead recipient** about sender of the email

```
S: 220 attacker.com SMTP Exim
C: HELO attacker.com
S: 250 Hello attacker.com
C: MAIL FROM: <ceo@company.com>
S: 250 Ok
C: RCPT TO: <bob@company.com>
S: 250 Accepted
C: DATA
S: 354 Enter a message, ending with "." on a line by itself
C: Subject: Download this urgently
C: From: ceo@company.com
C: To: bob@company.com
C:
C: Hi Bob,
C: Please download this urgently: https://some-malicious-link.com
C: Regards
C: .
S: 250 OK
C: QUIT
S: 221 attacker.com closing connection
```

Fake Sender

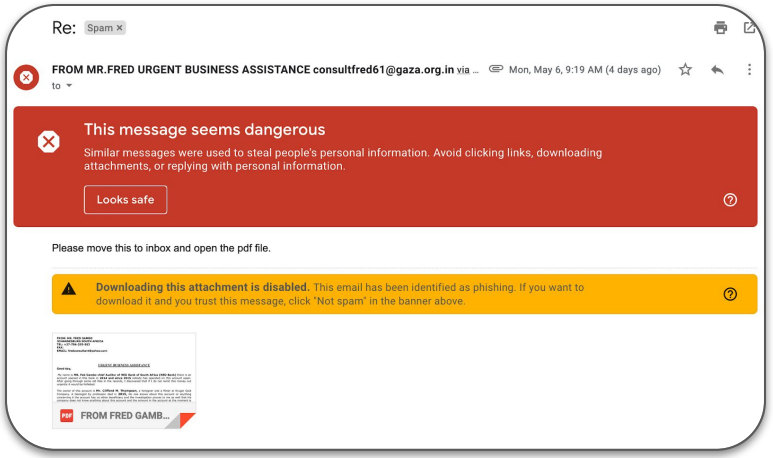
Victim

Malicious Link

```
To: robertbateman@email.com
Subject: Hi There
From: "Mickey Mouse" <m.mouse@disney.com>
X-Priority: 3 (Normal)
Importance: Normal
Errors-To: m.mouse@disney.com
Reply-To: m.mouse@disney.com
Content-Type: text/plain
```

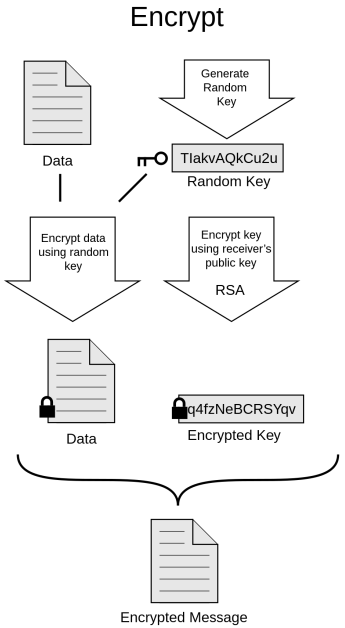
Thwarting Email Spoofing

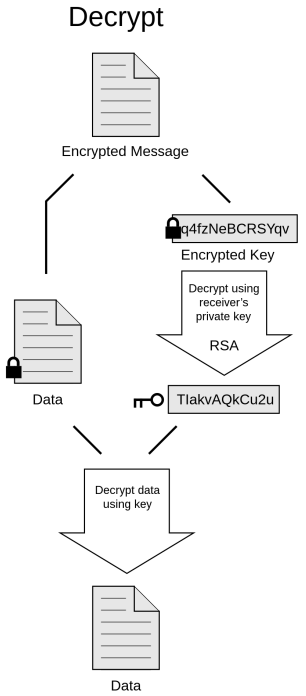
- **Checking email bodies**
  - Included links
  - Attached files
  - Text analysis (e.g., known spam campaigns)



Thwarting Email Spoofing

- **Checking email bodies**
  - Included links
  - Attached files
  - Text analysis (e.g., known spam campaigns)
- **Checking email headers**
  - Egress server domain registration
    - Check that sender is who it says it is
  - Pretty Good Privacy (PGP)
    - Sender and Receiver authentication
    - Confidentiality
    - Integrity





Questions?

Application Layer Attacks
DNS Hijacking

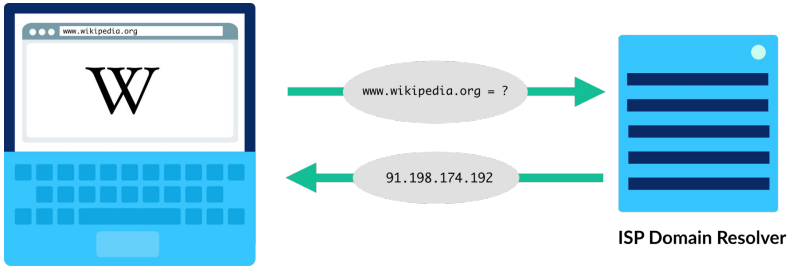
Identification on the Web

- How do we identify **people**?
  - **???**

Identification on the Web

- How do we identify **people**?
  - Social security numbers
  - Passports, drivers licenses
  - Their unique fingerprints

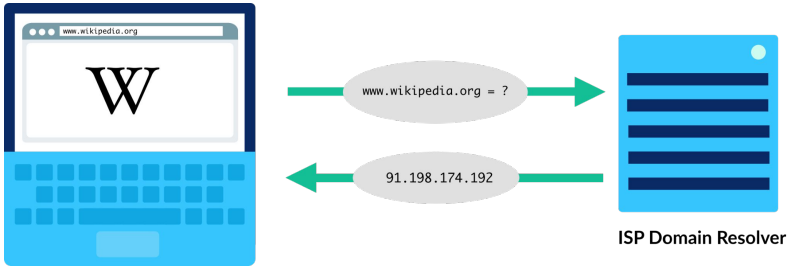
- How can we identify **internet hosts**?
  - **Network layer:** location via **IP addresses**
    - A logical addressing system
    - 32-bit (IPV4) addressing datagrams



Identification on the Web

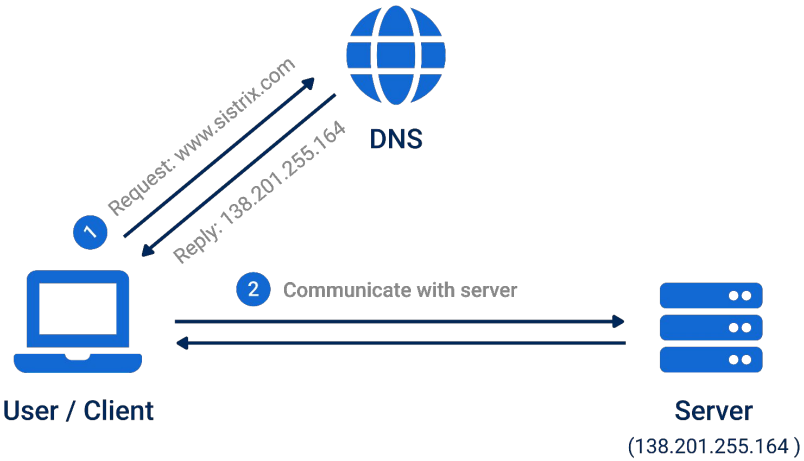
- How do we identify **people**?
  - Social security numbers
  - Passports, drivers licenses
  - Their unique fingerprints

- How can we identify **internet hosts**?
  - **Network layer:** location via **IP addresses**
    - A logical addressing system
    - 32-bit (IPV4) addressing datagrams
  - **What you care about:** the domain name
    - E.g., `www.wikipedia.org`



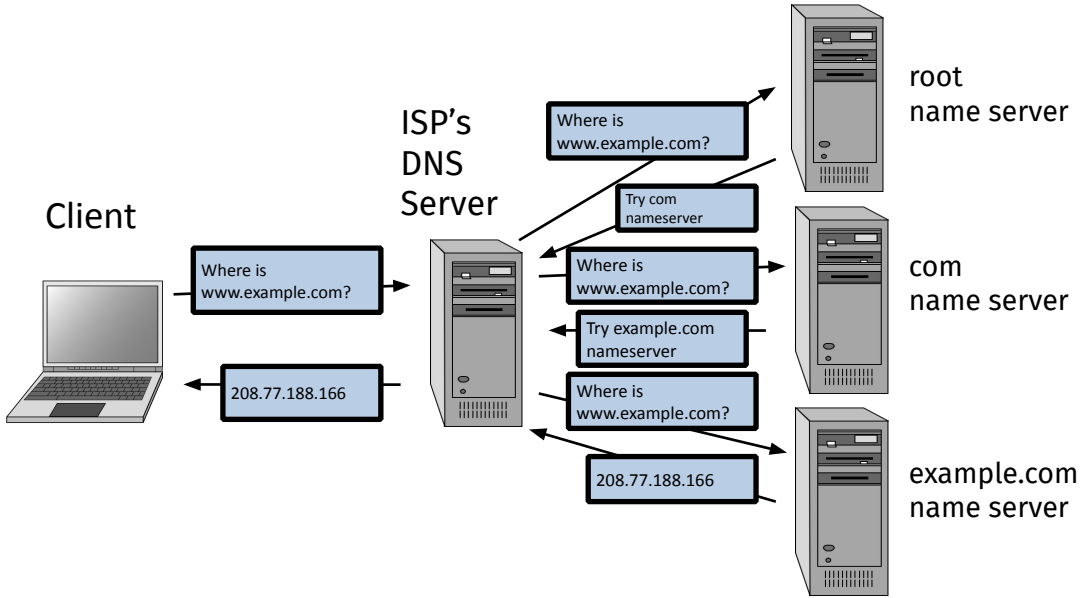
The Domain Name System

- **Distributed database** implemented in hierarchy of many name servers
- **Application-layer protocol:**
  - Hosts and domain name servers communicate to resolve **domain names**
    - Address–name translation
- **Result:** user requests **domain name**
  - But their host really gets its **IP address**
  - Convenient!



DNS Name Resolution

- **Zone:** collection of connected nodes with the same authoritative DNS server
- Resolution method when answer **not in cache:**

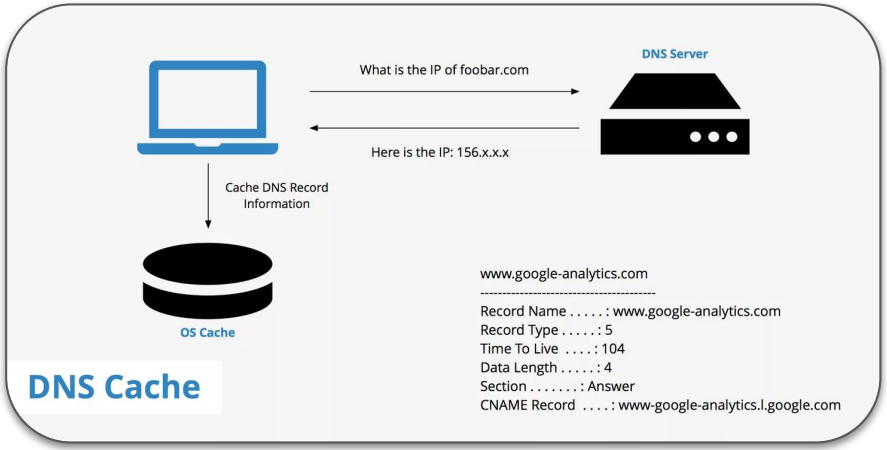


Reducing Resolution Latency

- **How can we optimize DNS resolution?**
  - **???**

Reducing Resolution Latency

- **How can we optimize DNS resolution?**
  - **Cache look-ups** to **amortize initial look-up**, reduce system load
- Temporal locality of requests:
  - `www.espn.com/page1`
  - `www.espn.com/page2`
- Popular destinations:
  - `google.com`
  - `Facebook.com`



Reducing Resolution Latency

```
stefan@cs4440:~$ time nslookup facebook.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   facebook.com
Address: 31.13.70.36
Name:   facebook.com
Address: 2a03:2880:f10d:83:face:b00c:0:25de


real    0m0.474s
user    0m0.000s
sys     0m0.015s
```

First Lookup (**non-cached**)

Reducing Resolution Latency

```
stefan@cs4440:~$ time nslookup facebook.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   facebook.com
Address: 31.13.70.36
Name:   facebook.com
Address: 2a03:2880:f10d:83:face:b00c:0:25de

real    0m0.474s
user    0m0.000s
sys     0m0.015s
```

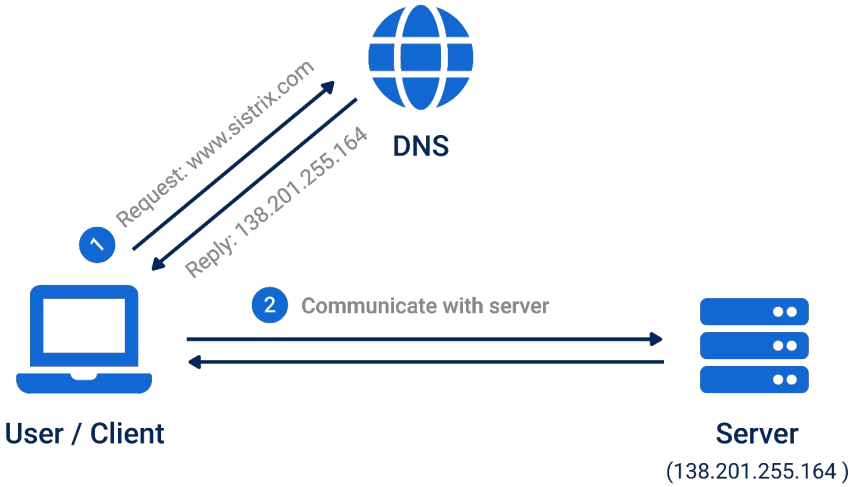
First Lookup (**non-cached**)

```
stefan@cs4440:~$ time nslookup facebook.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   facebook.com
Address: 31.13.70.36
Name:   facebook.com
Address: 2a03:2880:f10d:83:face:b00c:0:25de

real    0m0.023s
user    0m0.000s
sys     0m0.011s
```

Second Lookup (**cached**)

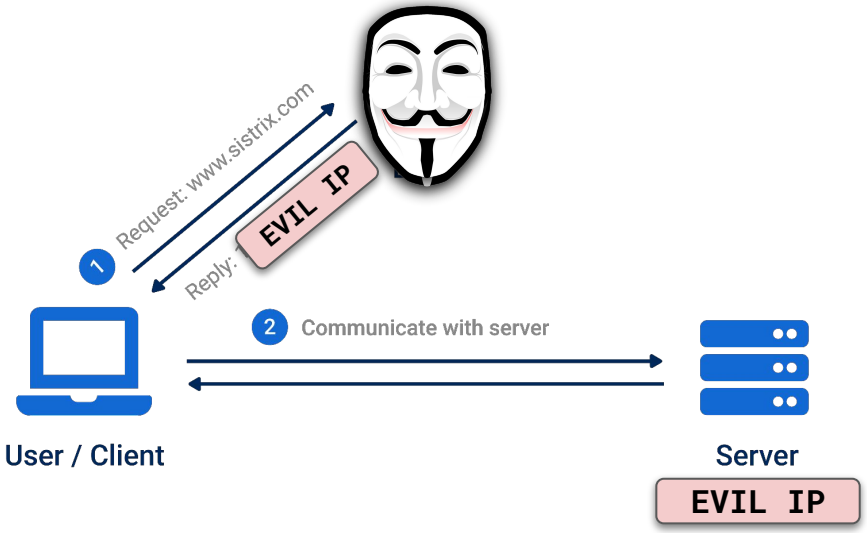
Attacking DNS

- What can an attacker do if they **control a DNS server**?



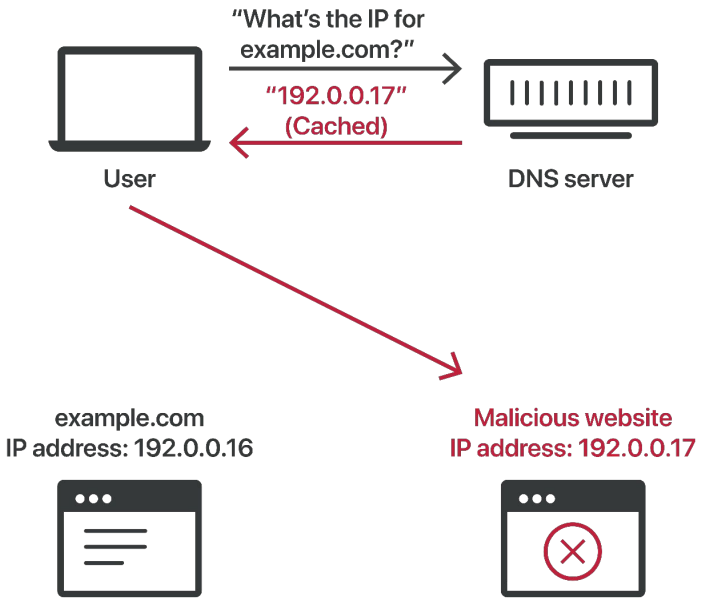
Attacking DNS

- What can an attacker do if they **control a DNS server**?
  - **Control how users of that DNS server view the internet!**
    - Assuming they use domain names



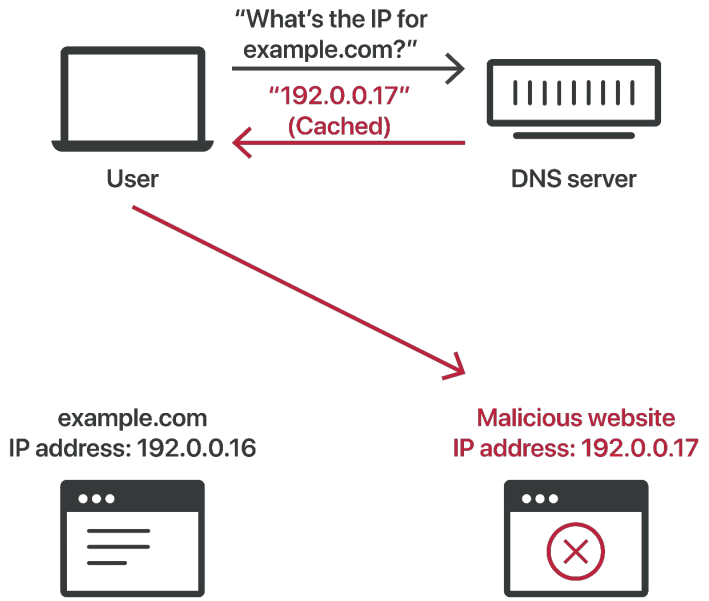
Case Study: DNS Cache Poisoning

- **Attack:** pre-empt DNS lookup by **injecting malicious cache** contents
  - Exploits DNS lookup optimization!



Case Study: DNS Cache Poisoning

- **Attack:** pre-empt DNS lookup by **injecting malicious cache** contents
  - Exploits DNS lookup optimization!
- Victim performs cache lookup, instead gets malicious domain IP
  - Attacker can **redirect** the victim's browser to the malicious website



Case Study: DNS Cache Poisoning

- **Attack:** pre-empt DNS lookup by **injecting malicious cache** contents
  - Exploits DNS lookup optimization!
- Victim performs cache lookup, instead gets malicious domain IP
  - Attacker can **redirect** the victim's browser to the malicious website
- **A massive vulnerability in 2008!**

The Great DNS Vulnerability of 2008 by Dan Kaminsky

The Internet was never designed to be secure. The Internet was designed to move pictures of cats.

In 2008, Security Researcher Dan Kaminsky presented on the massively widespread and critical Domain Name System (DNS) vulnerability that allowed attackers to send users to malicious sites and hijack email at Black Hat, the information security conference. The exploit would allow attackers to impersonate any legitimate website and steal data.

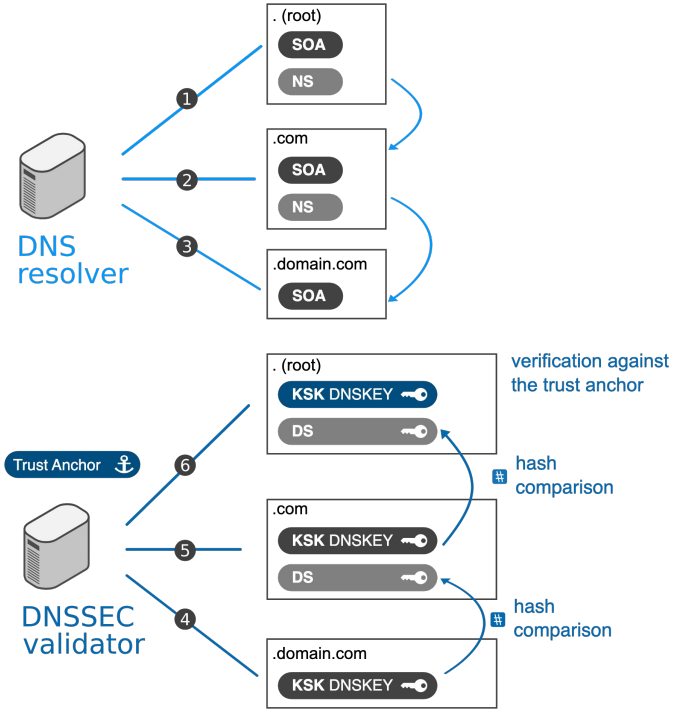
This fundamental design flaw allowed for arbitrary DNS cache poisoning - affecting nearly every DNS server on the planet, including vendors and products that worked with DNS. To explain what that is - here's some background on DNS:

Thwarting DNS Hijacking

- **Attack points:**
  - Local host
  - Router
  - ISP

Thwarting DNS Hijacking

- **Attack points:**
  - Local host
  - Router
  - ISP
- **DNS-level** authentication
  - DNSSec
  - Public-key crypto to “sign” DNS records
- **Endpoint** authentication
  - Certify that what I am seeing really is bank.com
  - Transport Layer Security (TLS)

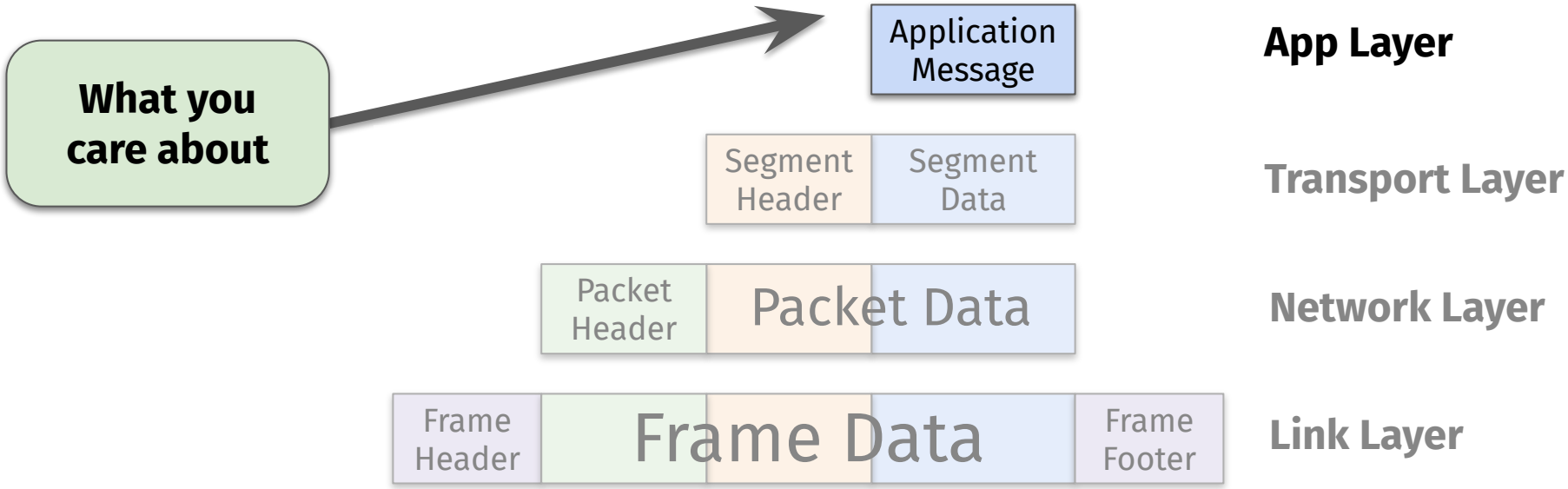


Questions?

Analyzing Network Packets

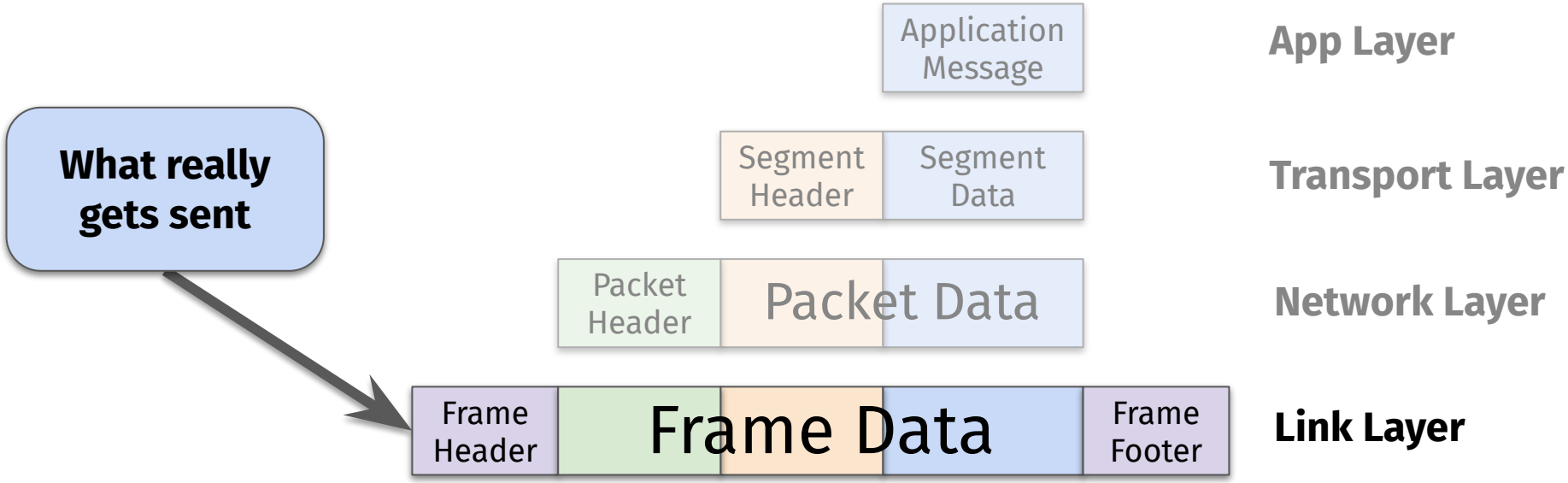
Recap: Internet Packet Encapsulation

- How packets are generated and sent



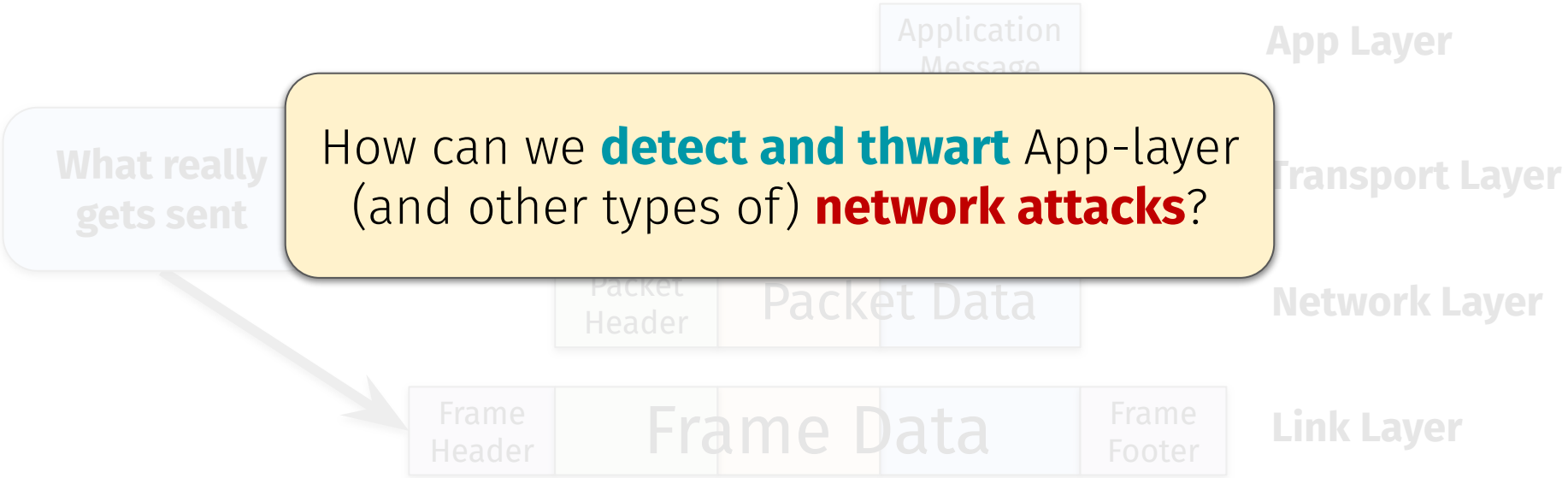
Recap: Internet Packet Encapsulation

- How packets are generated and sent



Recap: Internet Packet Encapsulation

- How packets are generated and sent



Tools of the Trade

- **Packet Analyzers:**
  - Tools for dissecting network packets
- Packet Analyzers allow you to:
  - Identify unusual packets
  - Characterize network activity
  - Pinpoint **malicious traffic**
- The basis of modern-day network security (e.g., firewalls, antivirus)

Wireshark

Familiarity with packet analysis tools?

I eat NetSec CTF challenges like a kid eats candy on Halloween. 🧙

0%

Some (e.g., Wireshark, DPKT, Scapy, or something else)

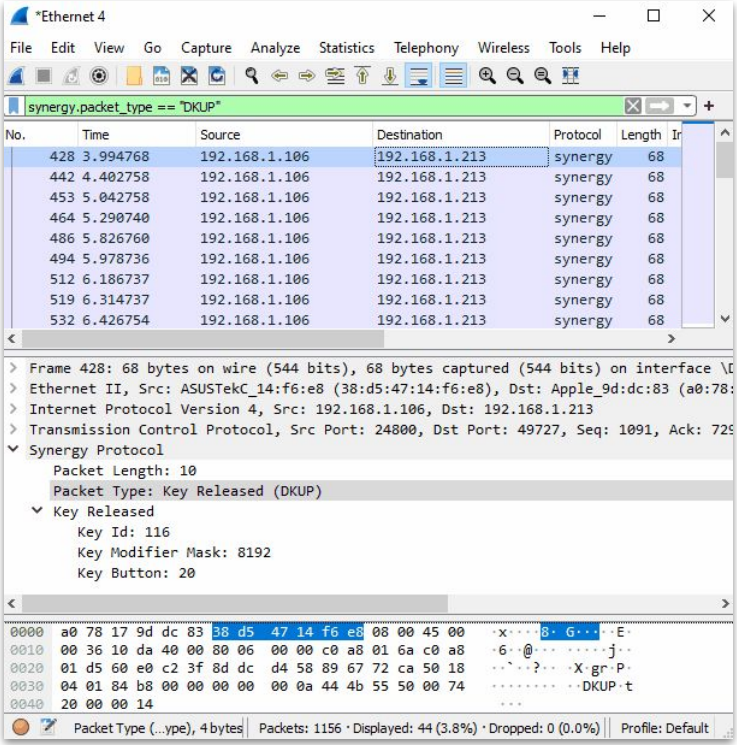
0%

None (but that's totally okay!)

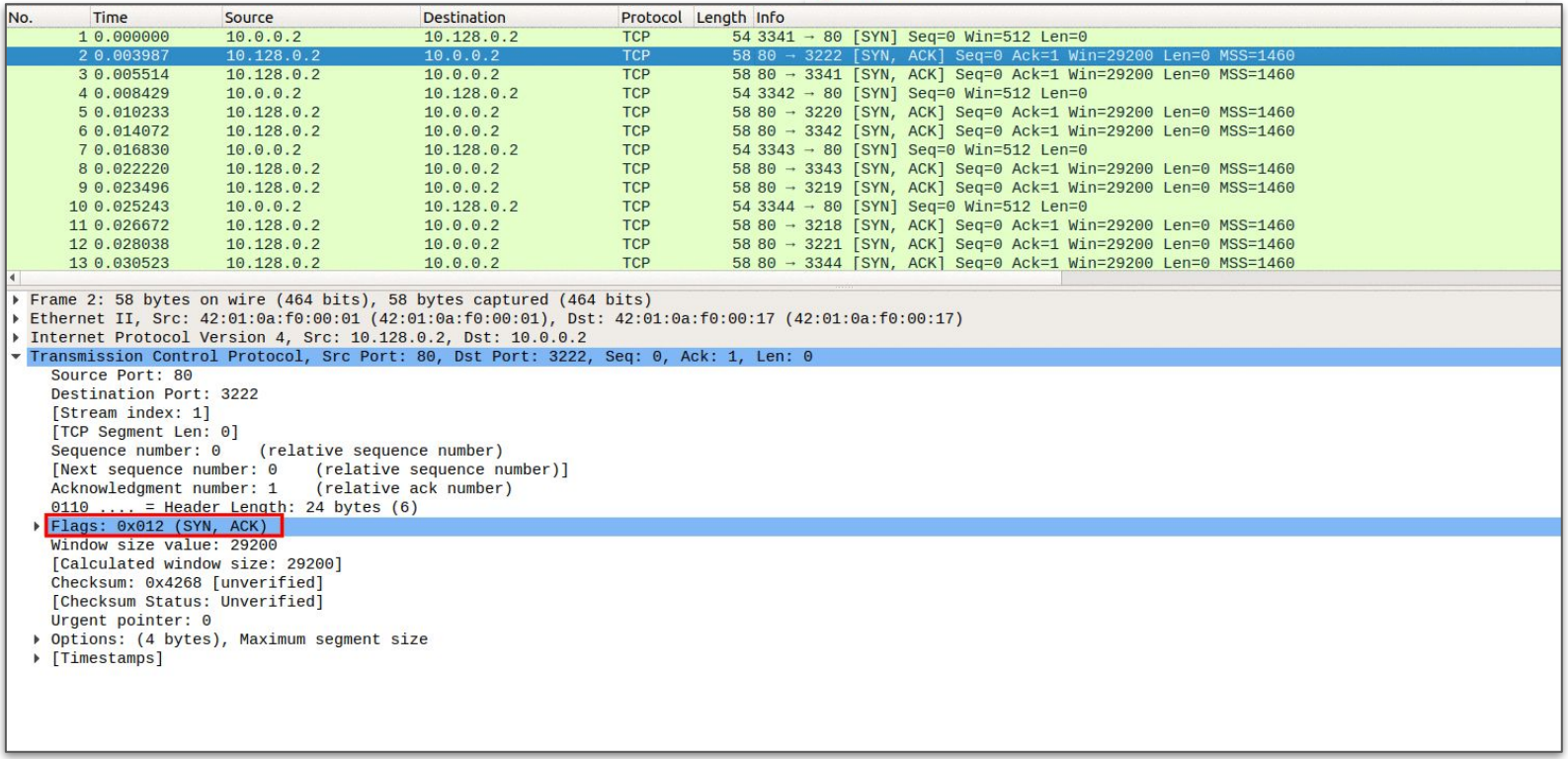
0%

Tools of the Trade: Wireshark

- A “graphical interface” for manual packet analysis
  - Completely **open-source and free**
- General workflow:
  - Load up a PCAP (packet capture)
  - Wireshark will display each packet
  - Inspect particular fields of interest



Tools of the Trade: Wireshark

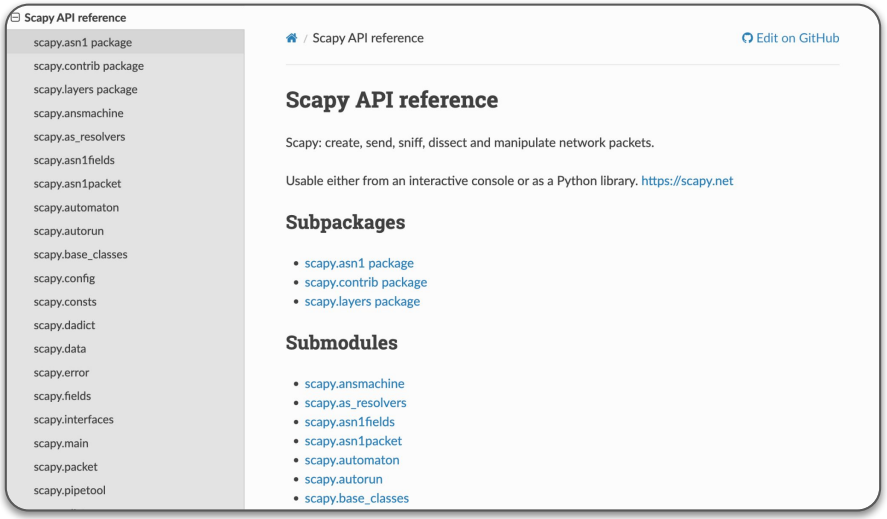


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.2	10.128.0.2	TCP	54	3341 → 80 [SYN] Seq=0 Win=512 Len=0
2	0.003987	10.128.0.2	10.0.0.2	TCP	58	80 → 3222 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
3	0.005514	10.128.0.2	10.0.0.2	TCP	58	80 → 3341 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
4	0.008429	10.0.0.2	10.128.0.2	TCP	54	3342 → 80 [SYN] Seq=0 Win=512 Len=0
5	0.010233	10.128.0.2	10.0.0.2	TCP	58	80 → 3220 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
6	0.014072	10.128.0.2	10.0.0.2	TCP	58	80 → 3342 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
7	0.016830	10.0.0.2	10.128.0.2	TCP	54	3343 → 80 [SYN] Seq=0 Win=512 Len=0
8	0.022220	10.128.0.2	10.0.0.2	TCP	58	80 → 3343 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
9	0.023496	10.128.0.2	10.0.0.2	TCP	58	80 → 3219 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
10	0.025243	10.0.0.2	10.128.0.2	TCP	54	3344 → 80 [SYN] Seq=0 Win=512 Len=0
11	0.026672	10.128.0.2	10.0.0.2	TCP	58	80 → 3218 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
12	0.028038	10.128.0.2	10.0.0.2	TCP	58	80 → 3221 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
13	0.030523	10.128.0.2	10.0.0.2	TCP	58	80 → 3344 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460

```
▸ Frame 2: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
▸ Ethernet II, Src: 42:01:0a:f0:00:01 (42:01:0a:f0:00:01), Dst: 42:01:0a:f0:00:17 (42:01:0a:f0:00:17)
▸ Internet Protocol Version 4, Src: 10.128.0.2, Dst: 10.0.0.2
▾ Transmission Control Protocol, Src Port: 80, Dst Port: 3222, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 3222
    [Stream index: 1]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    [Next sequence number: 0    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    0110 .... = Header Length: 24 bytes (6)
  ▸ Flags: 0x012 (SYN, ACK)
    Window size value: 29200
    [Calculated window size: 29200]
    Checksum: 0x4268 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (4 bytes), Maximum segment size
  ▸ [Timestamps]
```

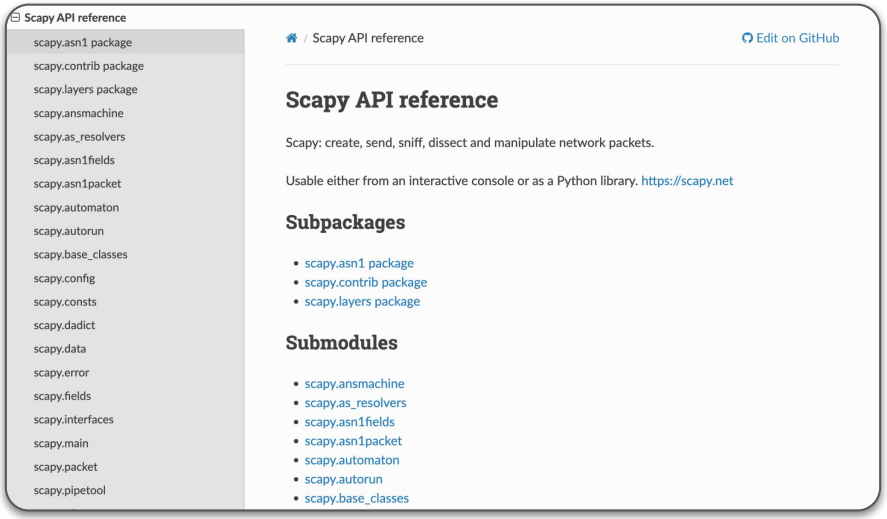
Tools of the Trade: Scapy

- Python API for programmatic packet capture and analysis
  - Think of it as “Wireshark in API form”
  - **Project 4:** you will use Scapy to write your own packet analysis scripts



Tools of the Trade: Scapy

- Python API for programmatic packet capture and analysis
  - Think of it as “Wireshark in API form”
  - **Project 4:** you will use Scapy to write your own packet analysis scripts
- We’ll provide the PCAP traces…
  - You’ll write code to analyze them!
  - **Examples:**
    - **Detecting attacks** on a network
    - Finding **user credentials**
    - Sniffing a user’s **browsing history**



Questions?

Next time on CS 4440...

Transport, Network, Link, and Physical Layer Attacks