Code Fellows

# Class 16
# TCP Server

seattle-javascript-401n14

# Lab 15 Review

## What is an event?

## What is a listener?

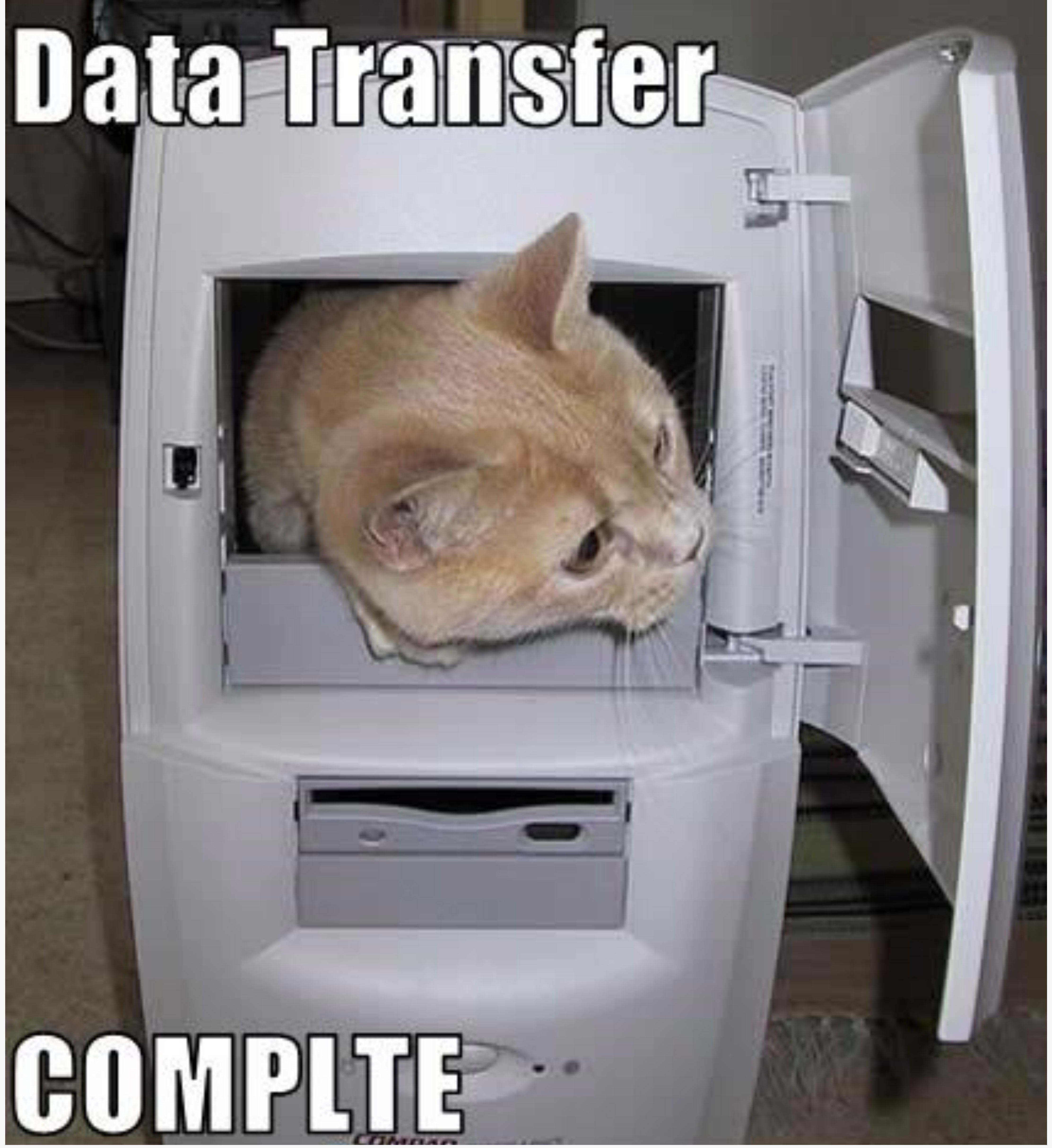
## What is a handler?

## What is a protocol?

## What is HTTP?

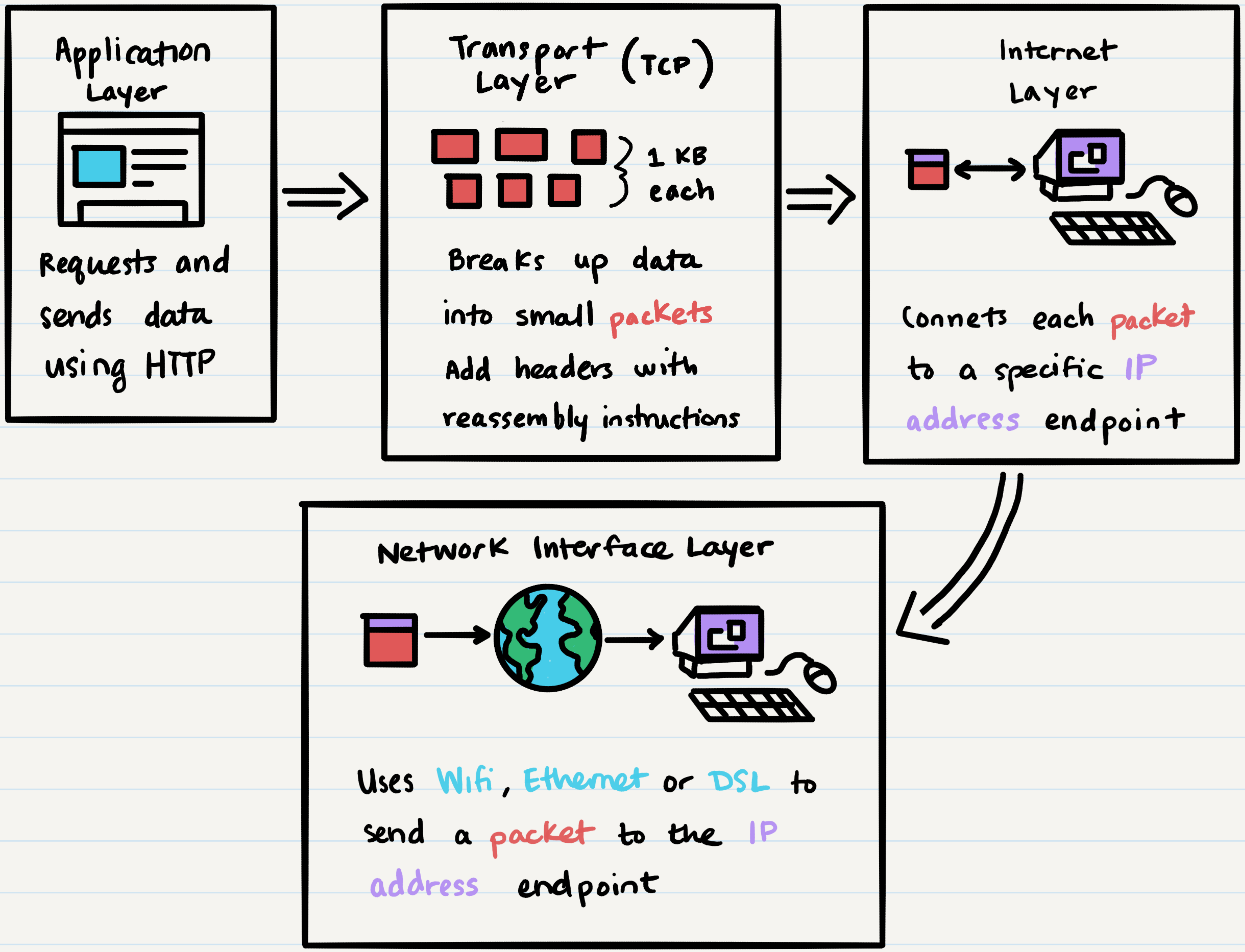
## Data Transfer

- We've done data transfer between two web applications using **HTTP** (`GET`, `POST`, `PUT`, `DELETE`)
- What happens behind the scenes?
- Multiple layers of operation
- All layers follow some **protocol** - a collection of rules



## Protocol Suites

- When multiple protocols are executed step-by-step
- We care about two in particular:
    - TCP/IP - Very old, very commonly used, 4 steps
    - OSI - More generic / thorough, 7 steps
- Both share the idea of Application Layer >> Transport Layer >> Network Layer



**Application Layer**

Requests and sends data using HTTP

**Transport Layer (TCP)**

Breaks up data into small packets
Add headers with reassembly instructions

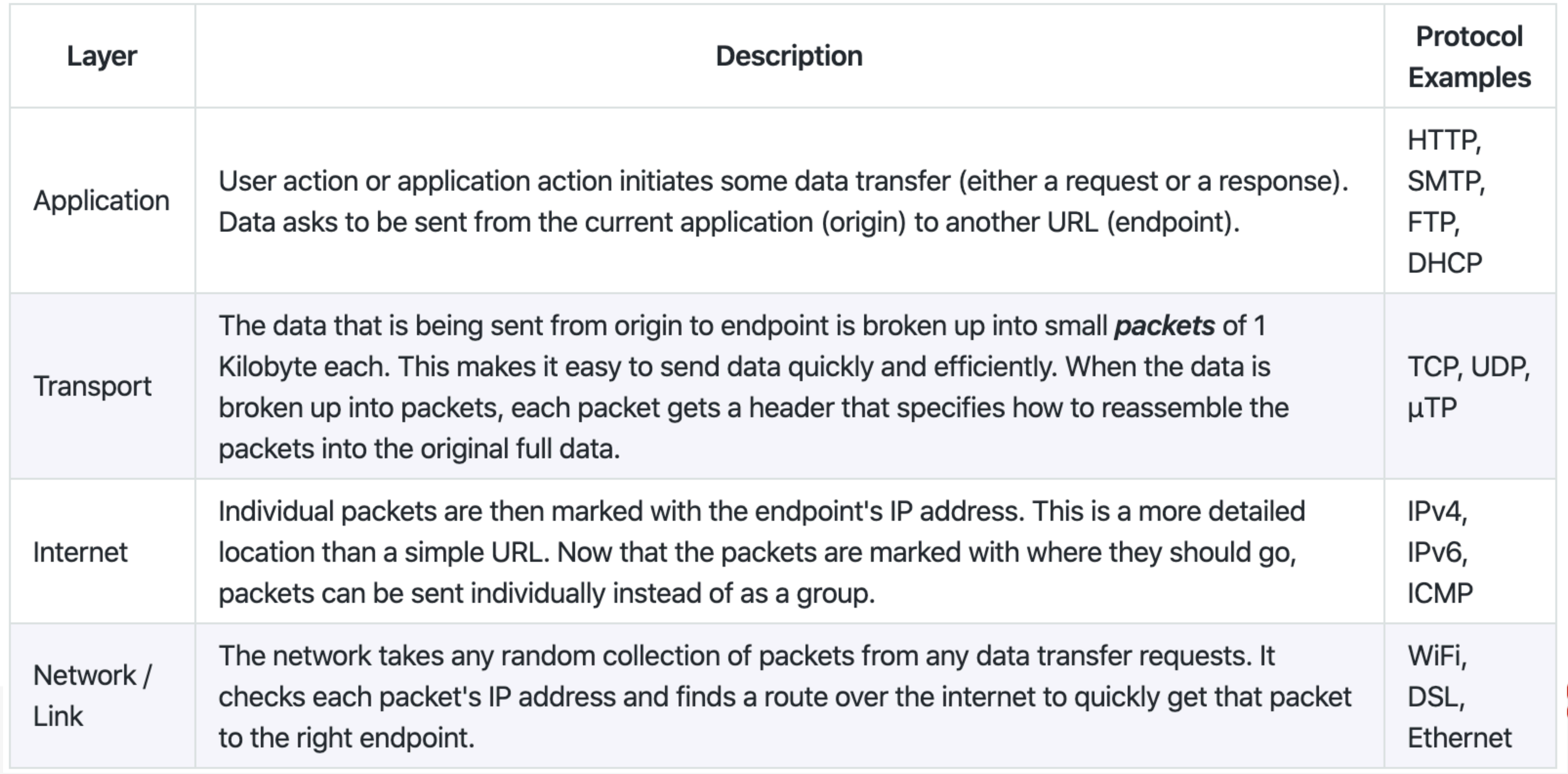
**Internet Layer**

Connets each packet to a specific IP address endpoint

**Network Interface Layer**

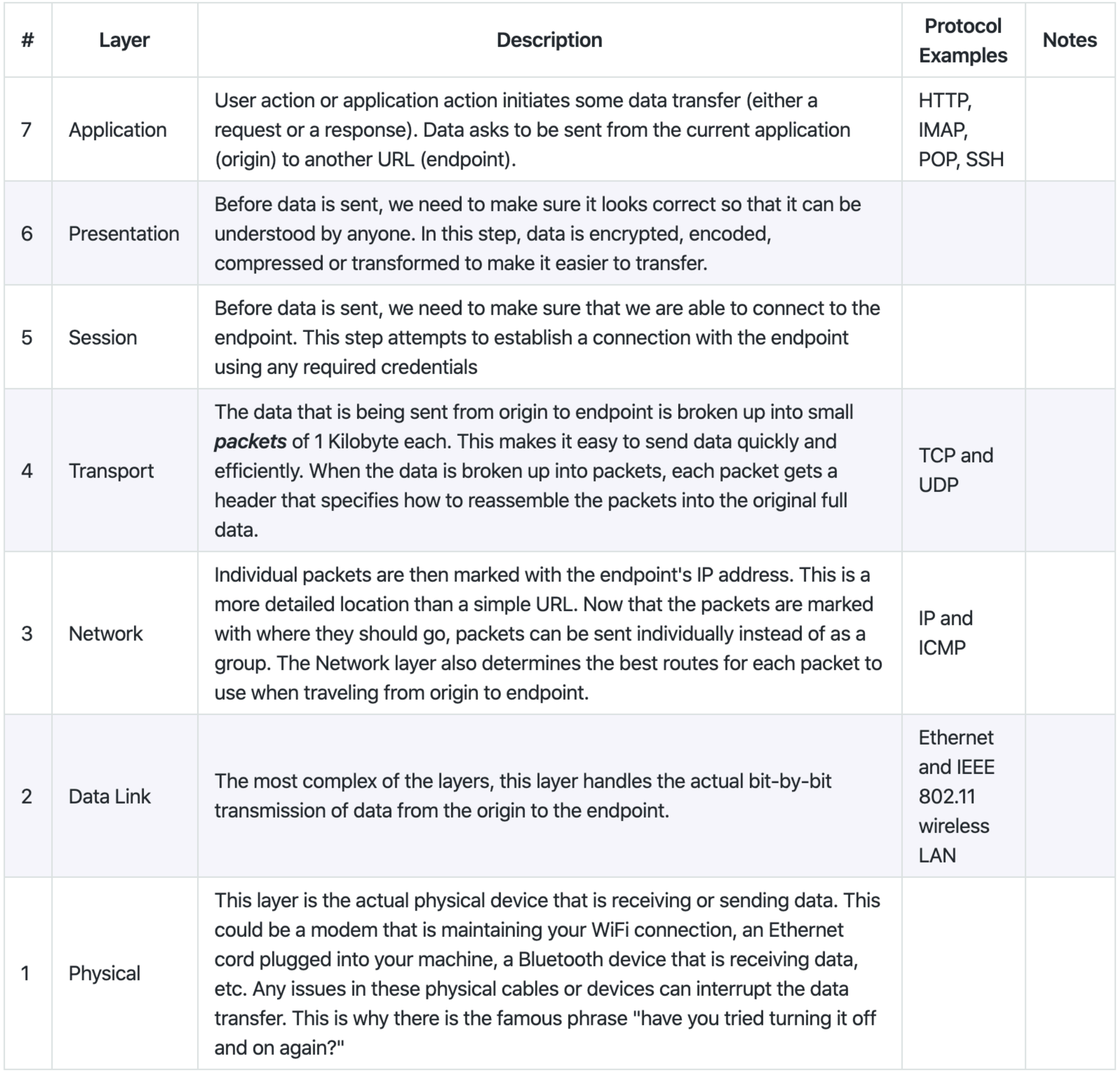
Uses Wifi, Ethernet or DSL to send a packet to the IP address endpoint

# TCP/IP



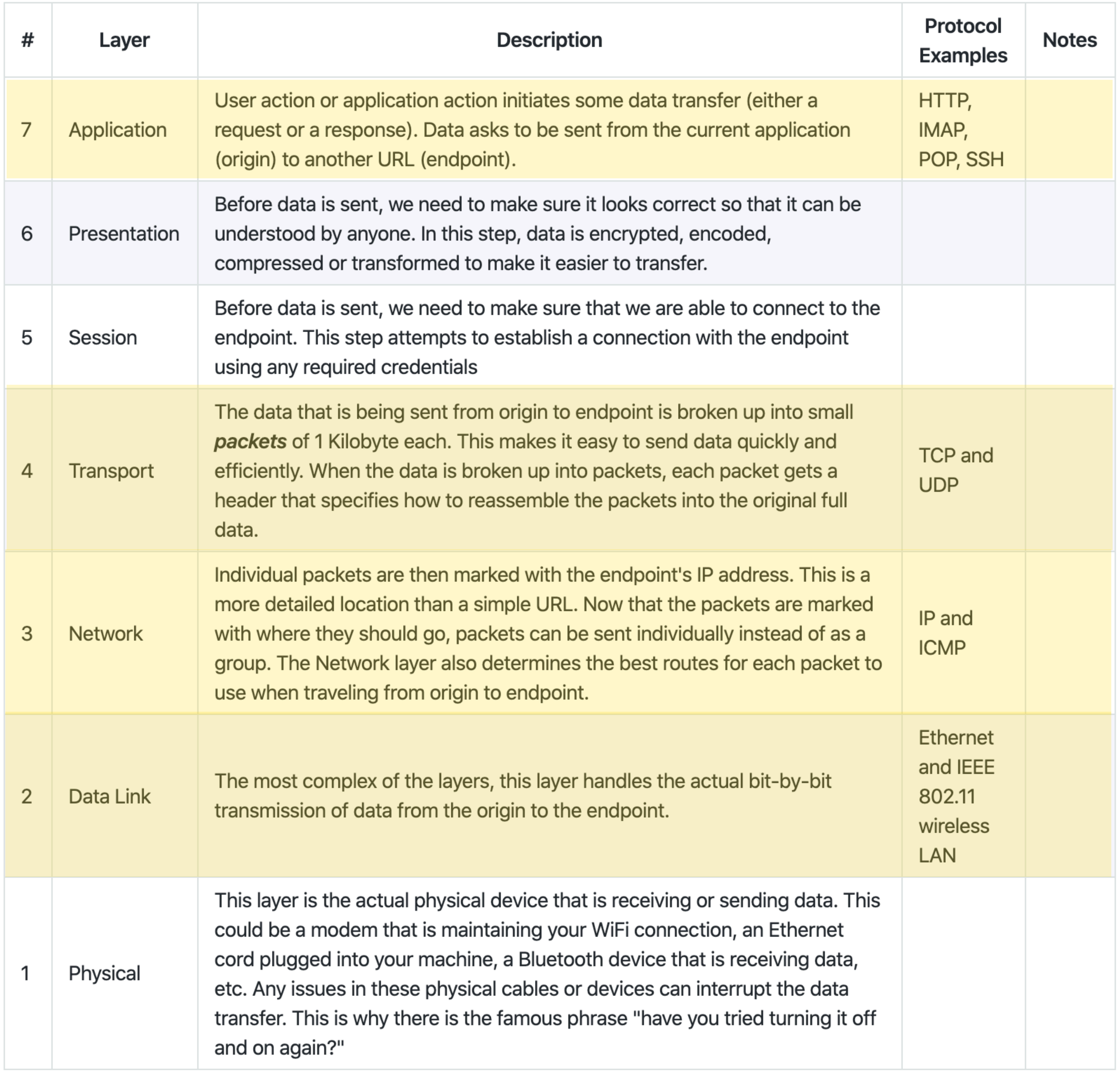
| Layer | Description | Protocol Examples |
|---|---|---|
| Application | User action or application action initiates some data transfer (either a request or a response). Data asks to be sent from the current application (origin) to another URL (endpoint). | HTTP, SMTP, FTP, DHCP |
| Transport | The data that is being sent from origin to endpoint is broken up into small ***packets*** of 1 Kilobyte each. This makes it easy to send data quickly and efficiently. When the data is broken up into packets, each packet gets a header that specifies how to reassemble the packets into the original full data. | TCP, UDP, µTP |
| Internet | Individual packets are then marked with the endpoint's IP address. This is a more detailed location than a simple URL. Now that the packets are marked with where they should go, packets can be sent individually instead of as a group. | IPv4, IPv6, ICMP |
| Network / Link | The network takes any random collection of packets from any data transfer requests. It checks each packet's IP address and finds a route over the internet to quickly get that packet to the right endpoint. | WiFi, DSL, Ethernet |

# OSI



| # | Layer | Description | Protocol Examples | Notes |
|---|---|---|---|---|
| 7 | Application | User action or application action initiates some data transfer (either a request or a response). Data asks to be sent from the current application (origin) to another URL (endpoint). | HTTP, IMAP, POP, SSH | |
| 6 | Presentation | Before data is sent, we need to make sure it looks correct so that it can be understood by anyone. In this step, data is encrypted, encoded, compressed or transformed to make it easier to transfer. | | |
| 5 | Session | Before data is sent, we need to make sure that we are able to connect to the endpoint. This step attempts to establish a connection with the endpoint using any required credentials | | |
| 4 | Transport | The data that is being sent from origin to endpoint is broken up into small ***packets*** of 1 Kilobyte each. This makes it easy to send data quickly and efficiently. When the data is broken up into packets, each packet gets a header that specifies how to reassemble the packets into the original full data. | TCP and UDP | |
| 3 | Network | Individual packets are then marked with the endpoint's IP address. This is a more detailed location than a simple URL. Now that the packets are marked with where they should go, packets can be sent individually instead of as a group. The Network layer also determines the best routes for each packet to use when traveling from origin to endpoint. | IP and ICMP | |
| 2 | Data Link | The most complex of the layers, this layer handles the actual bit-by-bit transmission of data from the origin to the endpoint. | Ethernet and IEEE 802.11 wireless LAN | |
| 1 | Physical | This layer is the actual physical device that is receiving or sending data. This could be a modem that is maintaining your WiFi connection, an Ethernet cord plugged into your machine, a Bluetooth device that is receiving data, etc. Any issues in these physical cables or devices can interrupt the data transfer. This is why there is the famous phrase "have you tried turning it off and on again?" | | |

# OSI



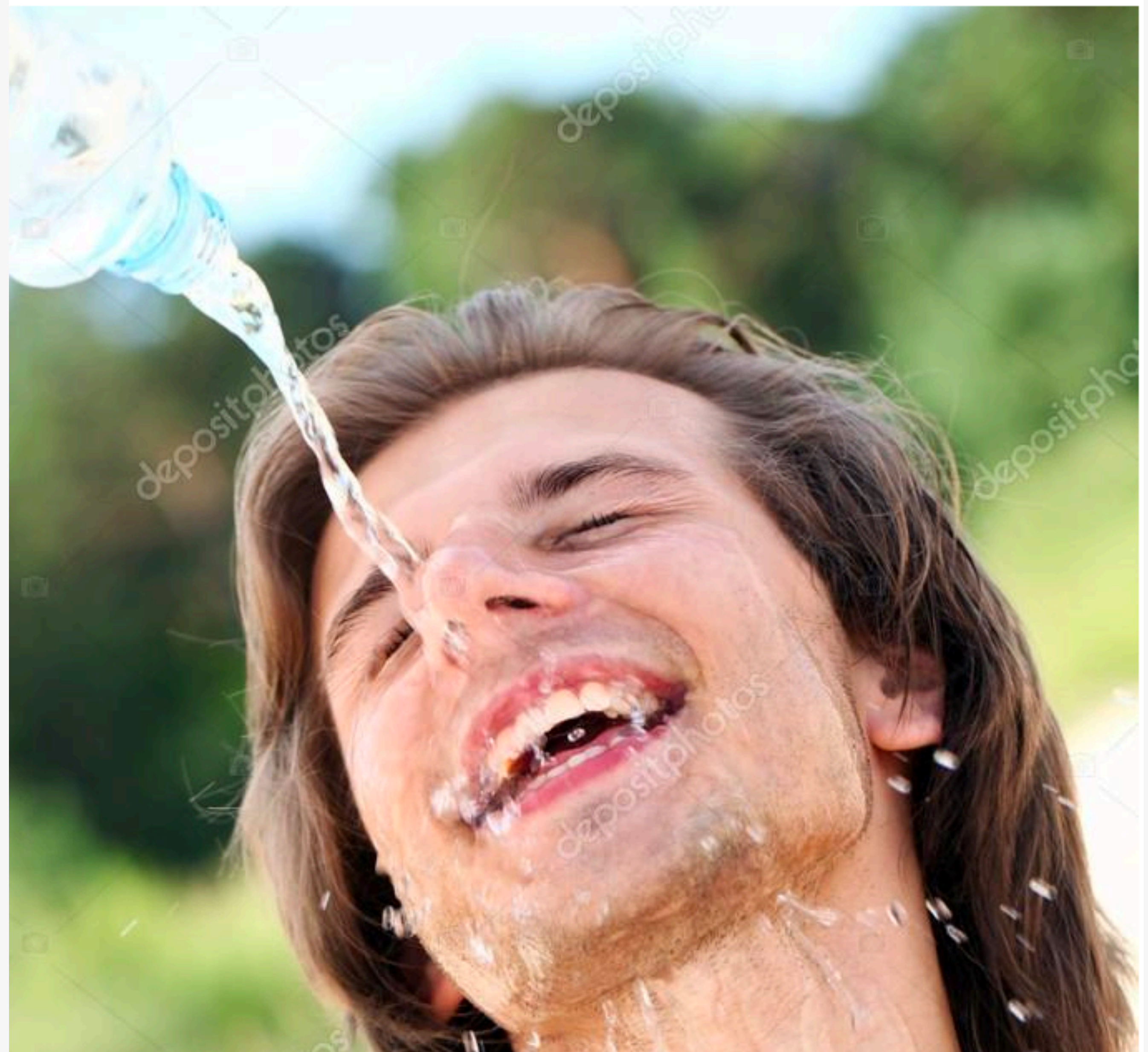
| # | Layer | Description | Protocol Examples | Notes |
|---|---|---|---|---|
| 7 | Application | User action or application action initiates some data transfer (either a request or a response). Data asks to be sent from the current application (origin) to another URL (endpoint). | HTTP, IMAP, POP, SSH | |
| 6 | Presentation | Before data is sent, we need to make sure it looks correct so that it can be understood by anyone. In this step, data is encrypted, encoded, compressed or transformed to make it easier to transfer. | | |
| 5 | Session | Before data is sent, we need to make sure that we are able to connect to the endpoint. This step attempts to establish a connection with the endpoint using any required credentials | | |
| 4 | Transport | The data that is being sent from origin to endpoint is broken up into small ***packets*** of 1 Kilobyte each. This makes it easy to send data quickly and efficiently. When the data is broken up into packets, each packet gets a header that specifies how to reassemble the packets into the original full data. | TCP and UDP | |
| 3 | Network | Individual packets are then marked with the endpoint's IP address. This is a more detailed location than a simple URL. Now that the packets are marked with where they should go, packets can be sent individually instead of as a group. The Network layer also determines the best routes for each packet to use when traveling from origin to endpoint. | IP and ICMP | |
| 2 | Data Link | The most complex of the layers, this layer handles the actual bit-by-bit transmission of data from the origin to the endpoint. | Ethernet and IEEE 802.11 wireless LAN | |
| 1 | Physical | This layer is the actual physical device that is receiving or sending data. This could be a modem that is maintaining your WiFi connection, an Ethernet cord plugged into your machine, a Bluetooth device that is receiving data, etc. Any issues in these physical cables or devices can interrupt the data transfer. This is why there is the famous phrase "have you tried turning it off and on again?" | | |

# Transport Layer

- There are two main ways to handle transport: **TCP** and **UDP**
- Both create **packets** with **headers**
- TCP headers are very complex and rigorous
- UDP doesn't really care about a strong connection with the destination, it just sends data out
- TCP is more common because it's more careful with its data

TCP

UDP



# TCP Server and Sockets

- We can use the package called `net` to create a **TCP server**
- This removes the need for an application layer
- A TCP server has multiple **sockets** that are connected to it
- Sockets have access to connect, close, data events, and a function write that emits a data event
- Servers have access to connection and close events, and maintain a **socket pool**

## Demo

demo/tcp-events

# What’s Next:

- Due by Midnight Wednesday:
    - **Learning Journal 16**
- Due by Midnight Thursday:
    - **Code Challenge 16**
- Due by Saturday 9am
    - **Lab 16**
    - **Reading Class 17**
- Next Class: **Class 17 - Socket.io**

## Questions?