Your Next Week

Saturday May 9 *9:00 AM* **— DUE Class 16 Code Challenge** **— DUE Class 16 Lab** **— DUE Class 17 Reading** — Class 17 — Interview Prep 03 *MIDNIGHT* **— DUE Class 17 Learning Journal**	Sunday May 10 *MIDNIGHT* — Class 16-17 Feedback	Monday May 11	Tuesday May 12 *6:30 PM* **— DUE Class 17 Code Challenge** **— DUE Class 17 Lab** **— DUE Class 18 Reading** — Class 18A
Wednesday May 13 *6:30 PM* — Class 18B *MIDNIGHT* **— DUE Class 18 Learning Journal**	Thursday May 14 *6:30 PM* — Co-working	Friday May 15	Saturday May 16 *9:00 AM* **— DUE Class 18 Code Challenge** **— DUE Class 18 Lab** **— DUE Class 19 Reading** — Class 19 — Interview Prep 04 *MIDNIGHT* **— DUE Class 19 Learning Journal**

What We've Covered

Module 01

Javascript Fundamentals and Data Models

C01 — Node Ecosystem, TDD, CI/CD
C02 — Classes, Inheritance, Functional Programming
C03 — Data Modeling & NoSQL Databases
C04 — Advanced Mongo/Mongoose
C05 — DSA: Linked Lists

Module 02

API Servers

C06 — HTTP and REST
C07 — Express
C08 — Express Routing & Connected API
C09 — API Server
C11 — DSA: Stacks and Queues

Module 03

Auth/Auth

C10 — Authentication
C12 — OAuth
C13 — Bearer Authorization
C14 — Access Control (ACL)
C15 — DSA: Trees

Module 04

Realtime

C16 — Event Driven Applications
C17 — TCP Server
C18 — Socket.io
C19 — Message Queues
C20 — Midterms Prep

Midterms

Module 05

React Basics

C21 — Component Based UI
C22 — React Testing and Deployment
C23 — Props and State
C24 — Routing and Component Composition
C25 — DSA: Sorting and HashTables

Module 06

Advanced React

C26 — Hooks API
C27 — Custom Hooks
C28 — Context API
C29 — Application State with Redux
C30 — DSA: Graphs

Module 07

Redux State Management

C31 — Combined Reducers
C32 — Asynchronous Actions
C33 — Additional Topics
C34 — React Native
C35 — DSA: Review

Module 08

UI Frameworks

C36 — Gatsby and Next
C37 — JavaScript Frameworks
C38 — Finals Prep

Finals

Lab 16 Review

Code Challenge 16
Review

Class 17

TCP Server

seattle-javascript-401n16

Data Transfer

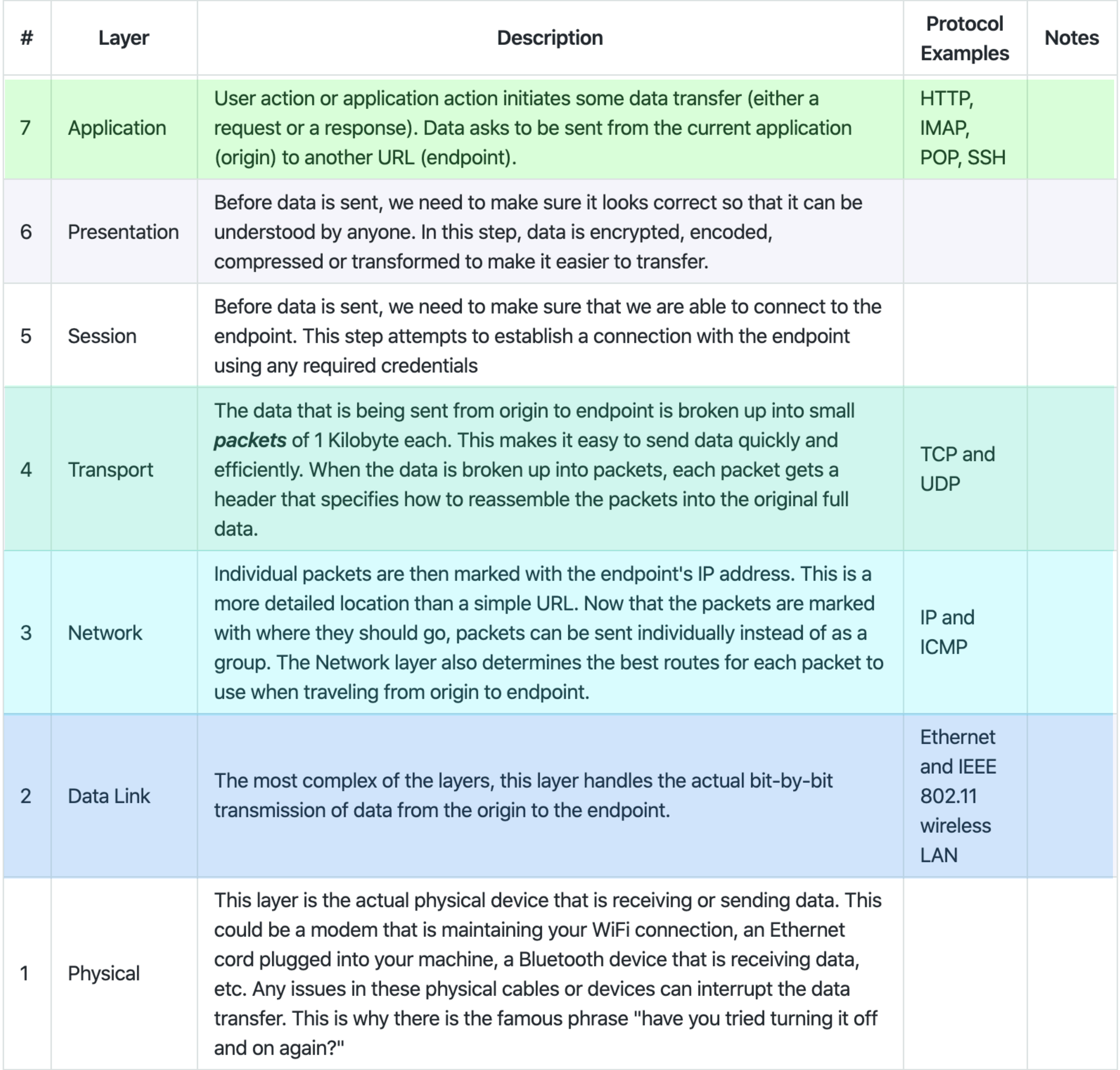
- We've seen one layer of data transfer - the HTTP **layer**
- What are the other layers?
- Each follows a **protocol**
- Some layers aren't always needed (HTTP layer is for client web app to server app)

Protocol Suites

- A collection of protocols and layers
- What layers should execute
  - Which protocols are available for each layer
- Two major ones we focus on:
  - **Open Systems Interconnection Reference Model (OSI)**
  - **Internet Protocol Suite (TCP/IP)**

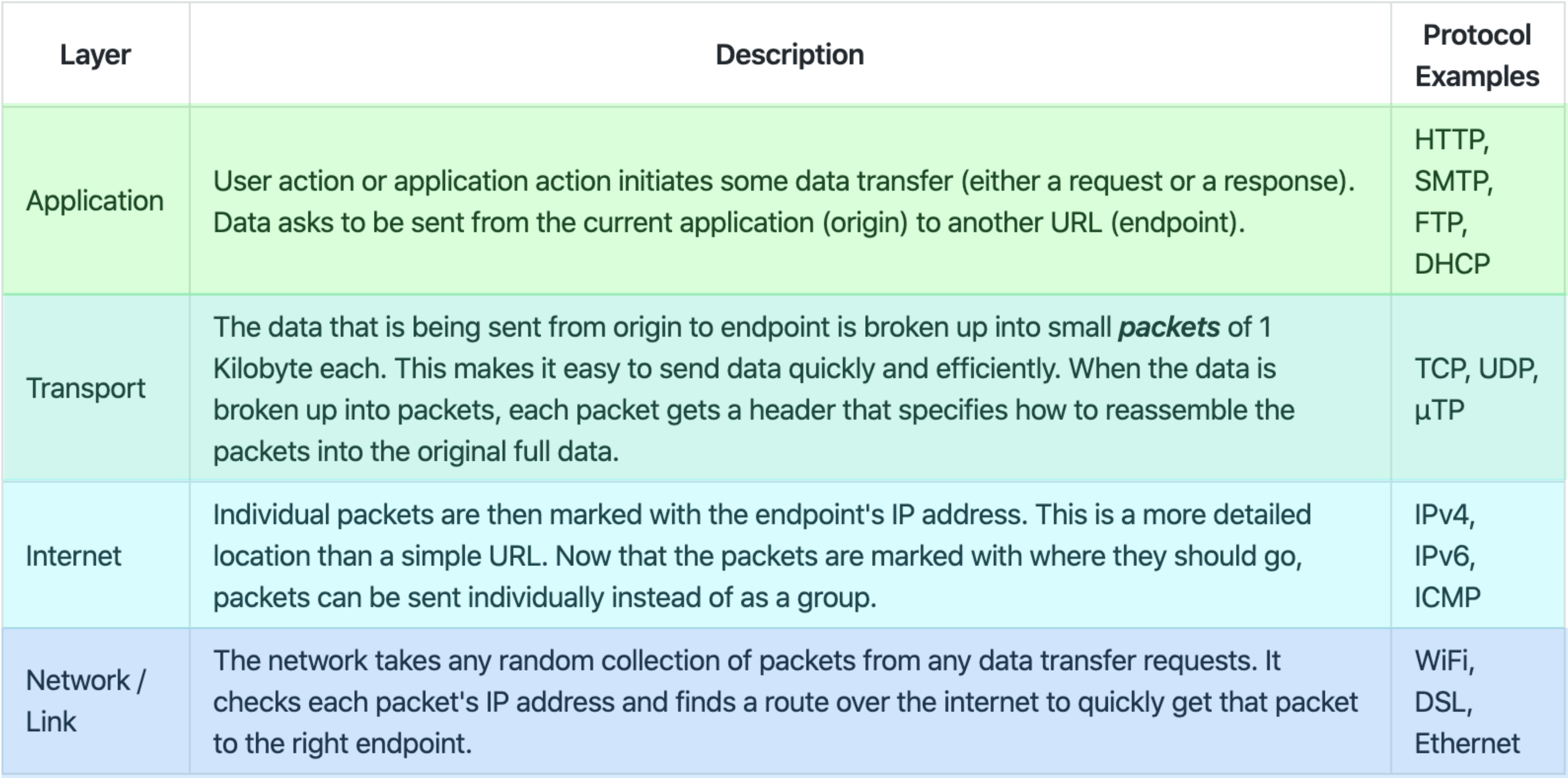
OSI

- Seven layers
- Very detailed guidelines on how to transfer data
- We can simplify to TCP/IP by only using **four fundamental steps**

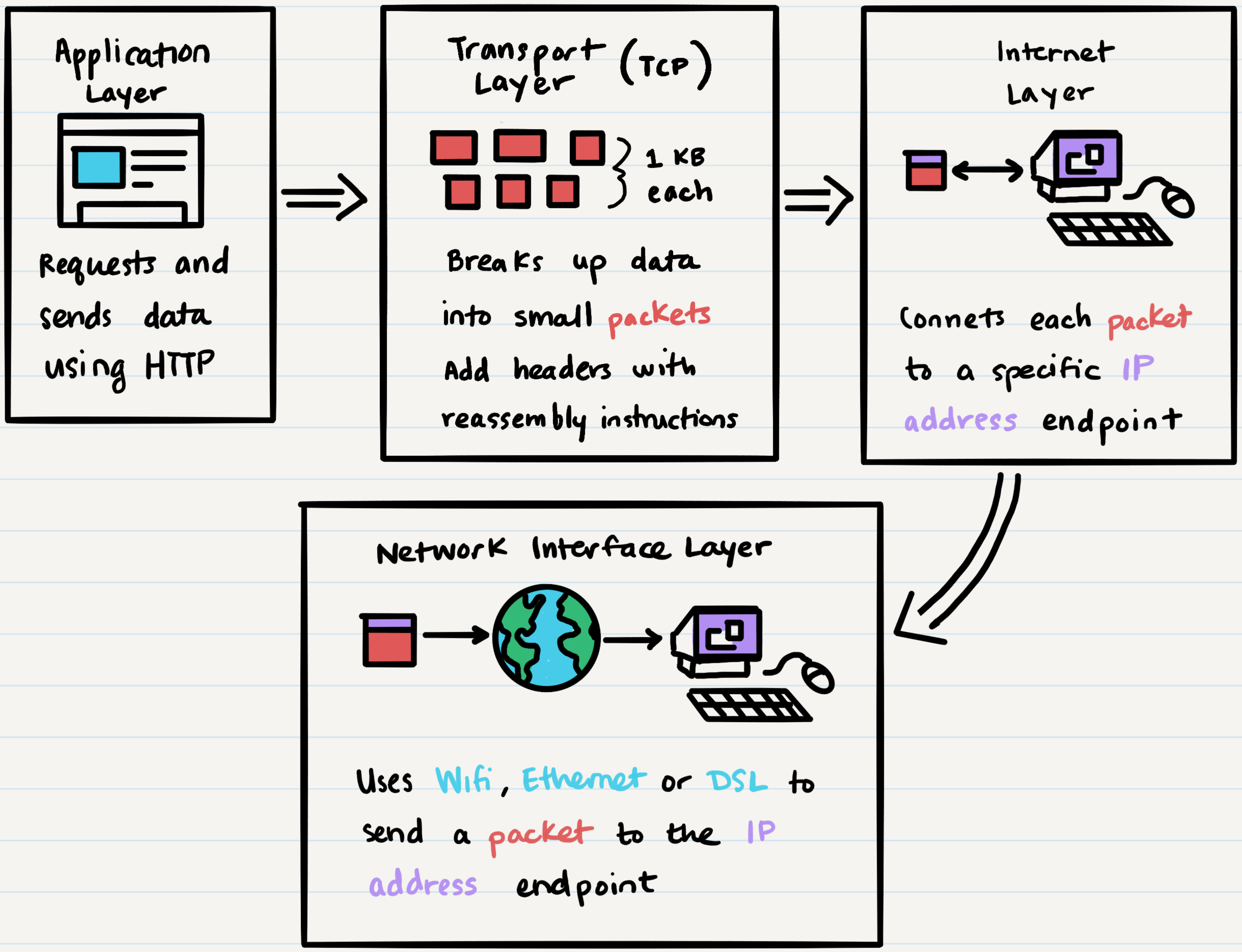


#	Layer	Description	Protocol Examples	Notes
7	Application	User action or application action initiates some data transfer (either a request or a response). Data asks to be sent from the current application (origin) to another URL (endpoint).	HTTP, IMAP, POP, SSH	
6	Presentation	Before data is sent, we 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?"		

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



Application Layer

Requests and sends data using HTTP

Transport Layer (TCP)

Breaks up data into small packets Add headers with reassembly instructions

Internet Layer

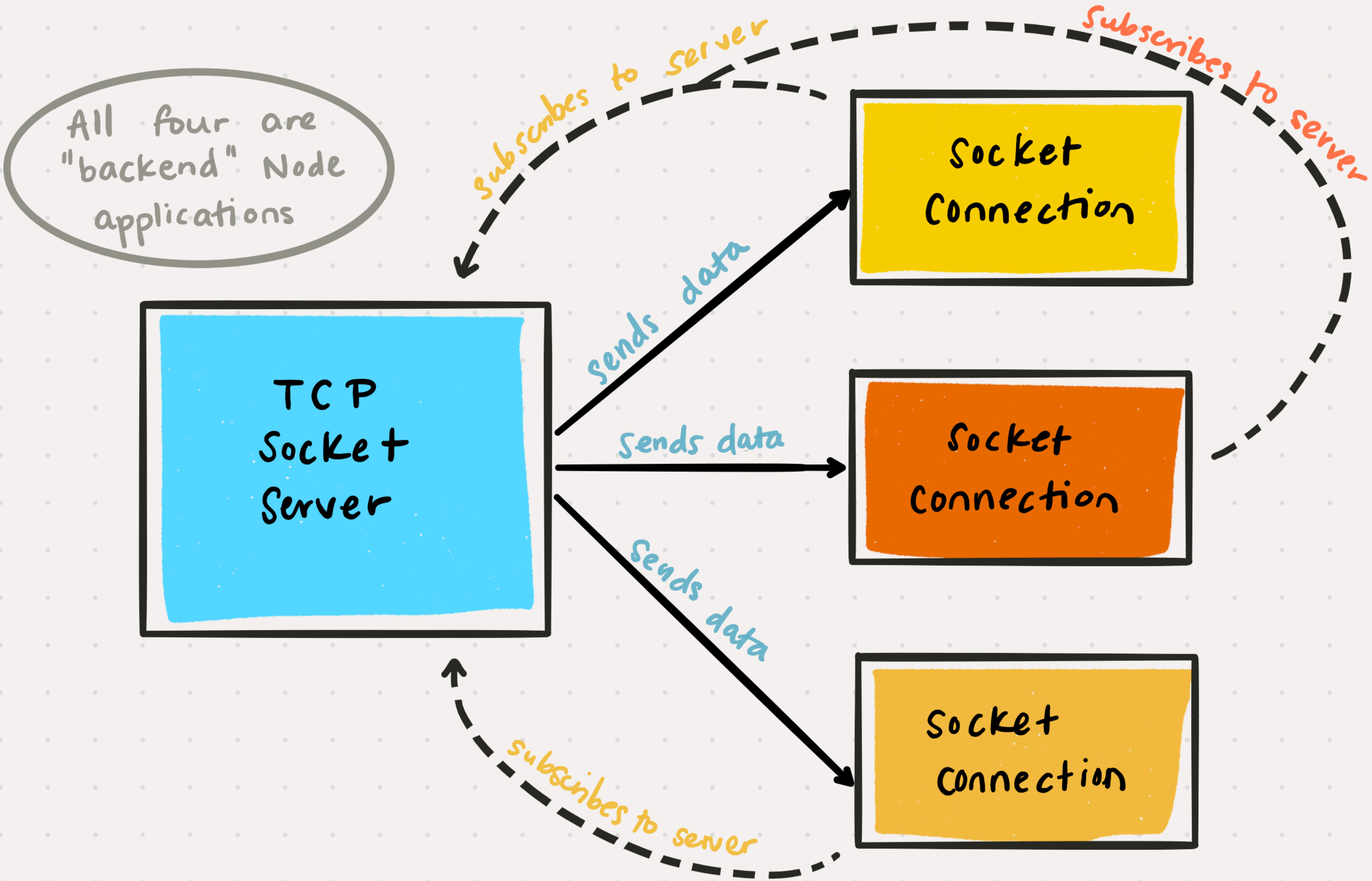
Connets each packet to a specific IP address endpoint

Network Interface Layer

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

TCP and UDP

- The two major protocols for the **transport layer**
- Both create packets with headers
- **Transmission Control Protocol (TCP)** headers are very complex and rigorous
    - More common and more secure
- **User Datagram Protocol (UDP)** doesn't care about a strong connection with the destination, it just sends data out


All four are "backend" Node applications
subscribes to server
Subscribes to server
TCP Socket Server
sends data
Sends data
Sends data
Socket Connection
Socket Connection
Socket Connection
subscribes to server

Lab 17 Overview