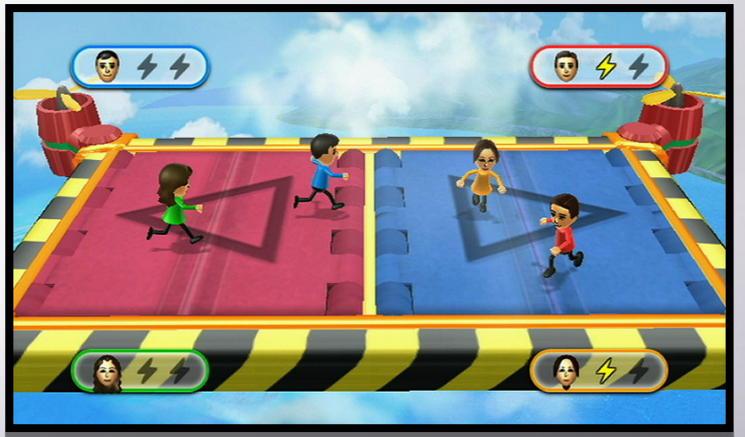
Networking lecture 1

- Course intro
- Networking in general
- Getting started with TCP
- Introduction to assignment 1

Overall course goal

Learn about programming simple (turnbased) network games using low level network technologies (Sockets, TCP & UDP protocols).



Learning goals for this course

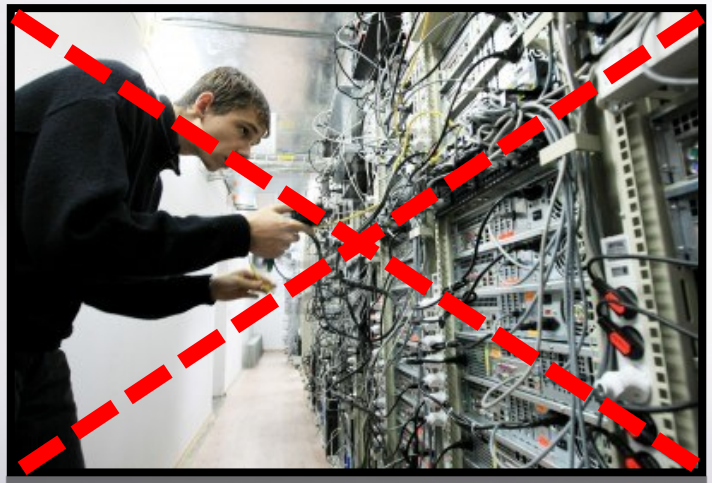
- Learn about the networking stack (TCP, UDP, IP & Sockets)
- Learn to communicate over a network using TCP
- Learn about (de)serialization of objects & error handling
- Learn about possible turn-based network game architectures
- And if there is time:
  - UDP
  - Threads
  - Real-time games
  - Other topics

Why low level?

- Less prone to aging, no chance of being deprecated any time soon
- Not limited to a specific tool:
  - Build a commandline network game
  - Build a GXPEngine network game
  - Build a Unity network game
  - Build an Unreal network game
- Allows you to understand high level API's much faster (Photon, Mirror, ...)
- Looks good on your portfolio

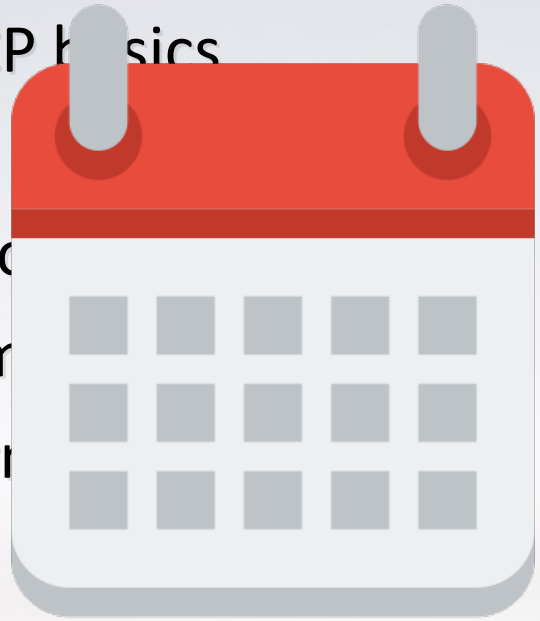
(Game) networking

- Our perspective is that of the network user, not the network administrator
- We assume the network is up and running, so that we can build a game for it…



Course schedule & time investment

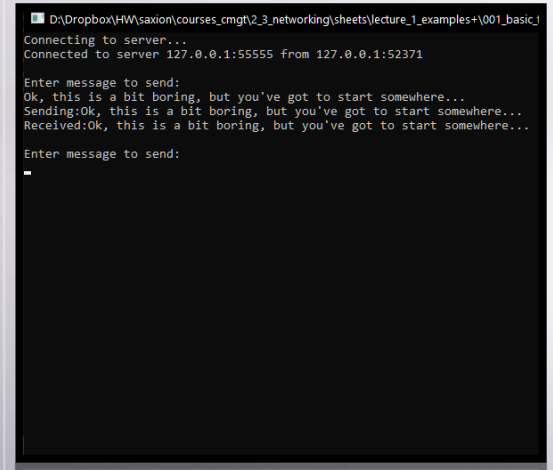
- Lecture 1 Networks & TCP b sics
- Lecture 2 TCP challenges
- Lecture 3 Object based co
- Lecture 4 Turn based gam
- Lecture 5 <... to be deter
- *(Week 6 No lecture)*

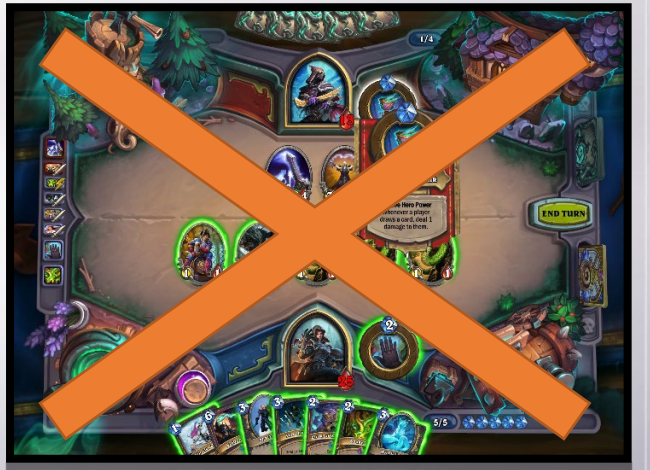


- 3 ECTS = 84 hours = 14 hours per week for a 6 week course
- In practice 6/7 weeks, 8-12 hours per week, so somewhere between 48-84

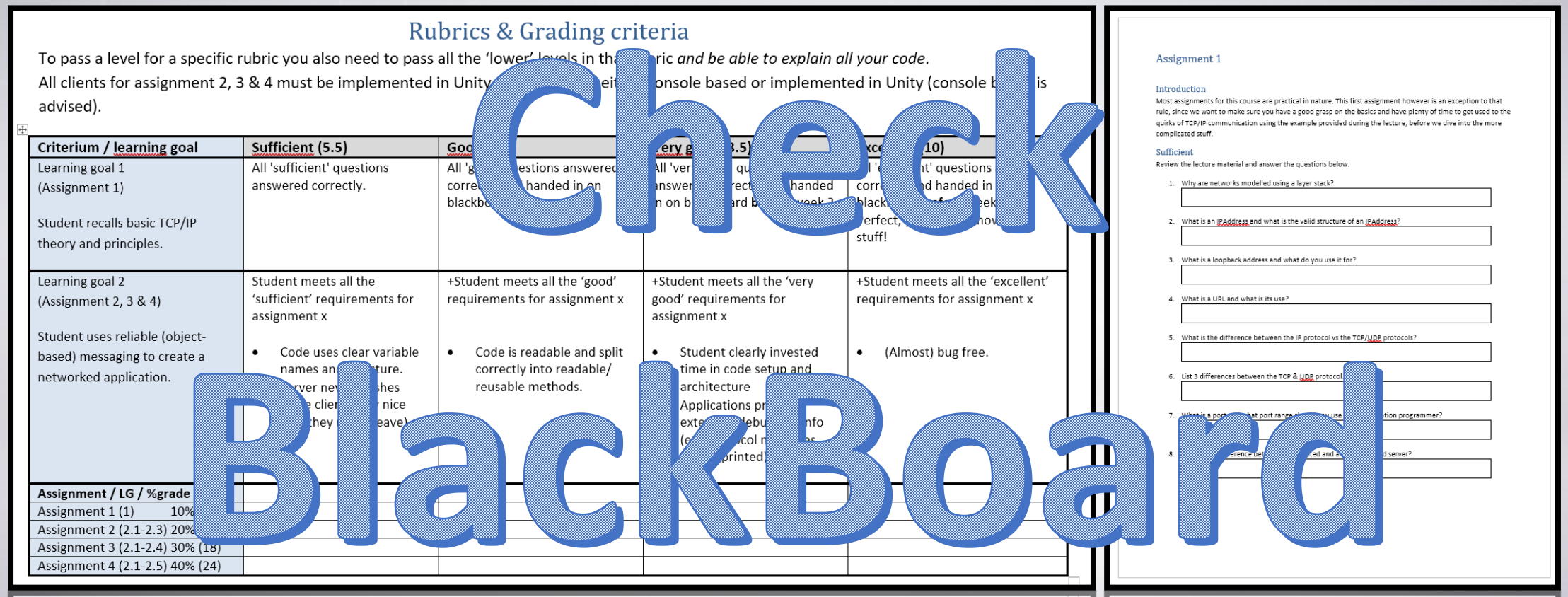
Exam format

- Assessment: 4 assignments (base code provided)
  - Week 1: A1 - Getting started with TCP (1 point)
  - Week 2: A2 - Text based chat client/server (2 points)
  - Week 3: A3 - 3D chat lobby client/server (3 points)
  - Week 4: A4 - Turn based network game (4 points)





Assignment instructions & grading info



Rubrics & Grading criteria

To pass a level for a specific rubric you also need to pass all the 'lower' levels in tha ric *and be able to explain all your code*.
All clients for assignment 2, 3 & 4 must be implemented in Unity ei onsole based or implemented in Unity (console b is advised).

Criterium / learning goal	Sufficient (5.5)	Goo	ery g .5)	xce 10)
Learning goal 1 (Assignment 1) Student recalls basic TCP/IP theory and principles.	All 'sufficient' questions answered correctly.	All 'g estions answered corre handed in on blackbo	All 'ver qu answer rect handed n on b ard b	l ' nt' questions corr d handed in blacki eek erfect, ho stuff!
Learning goal 2 (Assignment 2, 3 & 4) Student uses reliable (object-based) messaging to create a networked application.	Student meets all the 'sufficient' requirements for assignment x • Code uses clear variable names an ture. rver ne shes e clie nice they eave)	+Student meets all the 'good' requirements for assignment x • Code is readable and split correctly into readable/ reusable methods.	+Student meets all the 'very good' requirements for assignment x • Student clearly invested time in code setup and architecture Applications pr exte ebu nfo (col r printed)	+Student meets all the 'excellent' requirements for assignment x • (Almost) bug free.
Assignment / LG / %grade				
Assignment 1 (1) 10%				
Assignment 2 (2.1-2.3) 20%				
Assignment 3 (2.1-2.4) 30% (18)				
Assignment 4 (2.1-2.5) 40% (24)				

Check

BlackBoard

Assignment 1

Introduction

Most assignments for this course are practical in nature. This first assignment however is an exception to that rule, since we want to make sure you have a good grasp on the basics and have plenty of time to get used to the quirks of TCP/IP communication using the example provided during the lecture, before we dive into the more complicated stuff.

Sufficient

Review the lecture material and answer the questions below.

1. Why are networks modelled using a layer stack?
2. What is an IPAddress and what is the valid structure of an IPAddress?
3. What is a loopback address and what do you use it for?
4. What is a URL and what is its use?
5. What is the difference between the IP protocol vs the TCP/UDP protocols?
6. List 3 differences between the TCP & UDP protocol
7. ...
8. ...

Assignment notes

- The assignments build on top of each other!
- Make sure to finish at least the ***sufficient*** of ***each*** assignment, before the next lecture !
  - Assignment 1 Sufficient before lecture 2 (but try Excellent, A1 is really small)
  - Assignment 2 Sufficient before lecture 3
  - Assignment 3 Sufficient before lecture 4
- After that you can go back to previous assignments and improve them
- Assignment 1, 2 & 3 (sufficient) **need to be signed off during lab**, ***before*** the assessment in week 3.9

Networks & the network stack

What is a network?

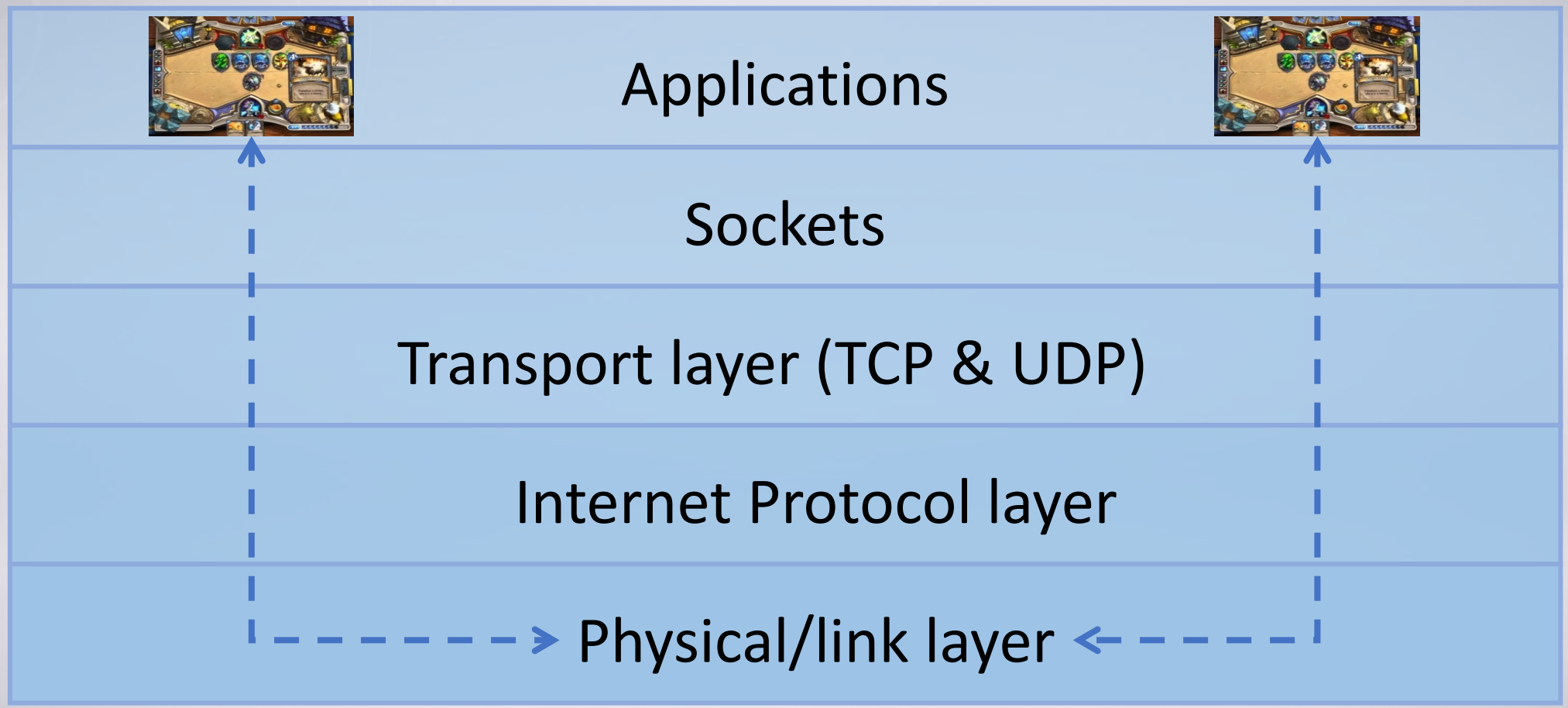
In this course: two or more application instances (on different computers/devices) exchanging data (byte[]'s) in real-time

The Internet

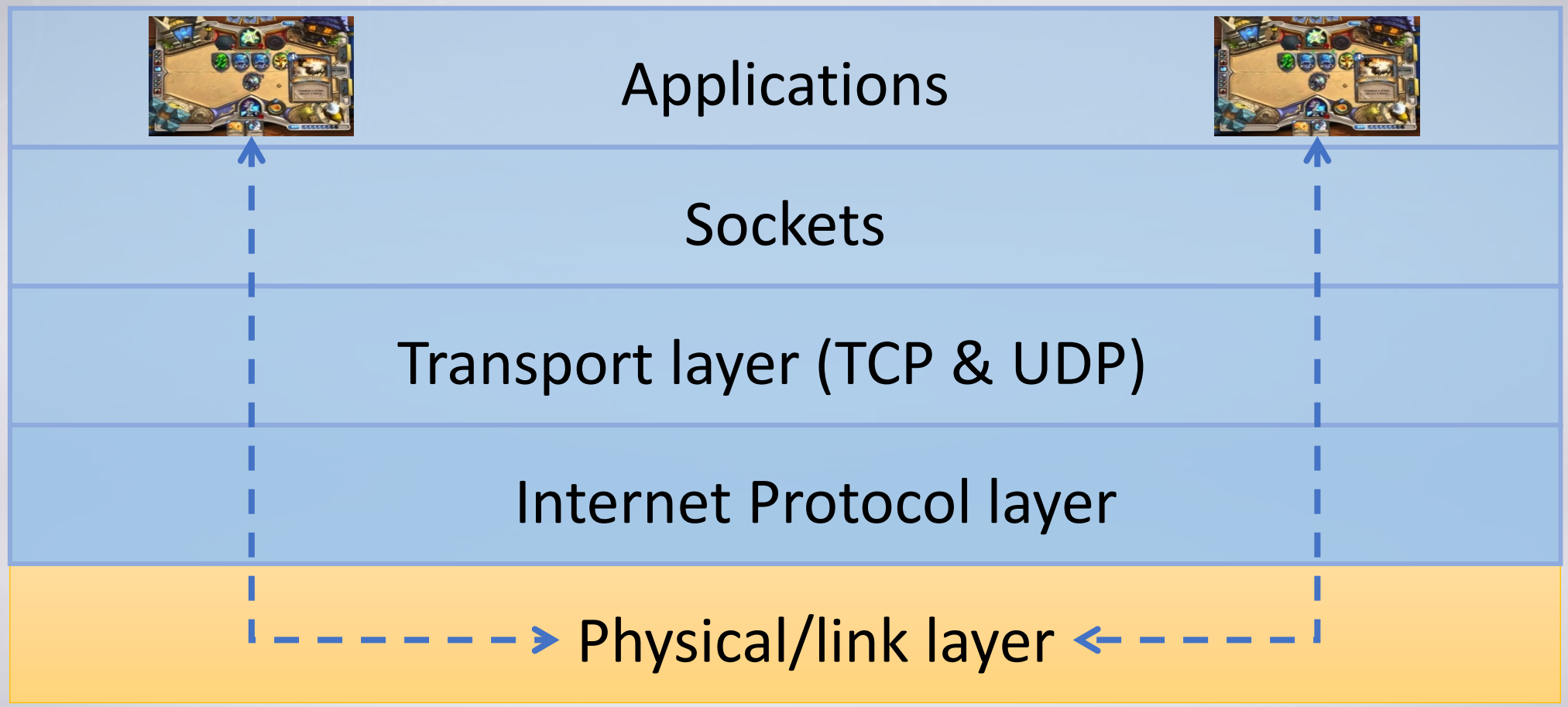
Networks are a complicated beast

- Different media types (WiFi, 4G, Coax, twisted-pair, fiber-optic cable)
- Different devices (pc's, mobile devices, televisions, IoT, etc)
- Different applications with different demands

Network Stack



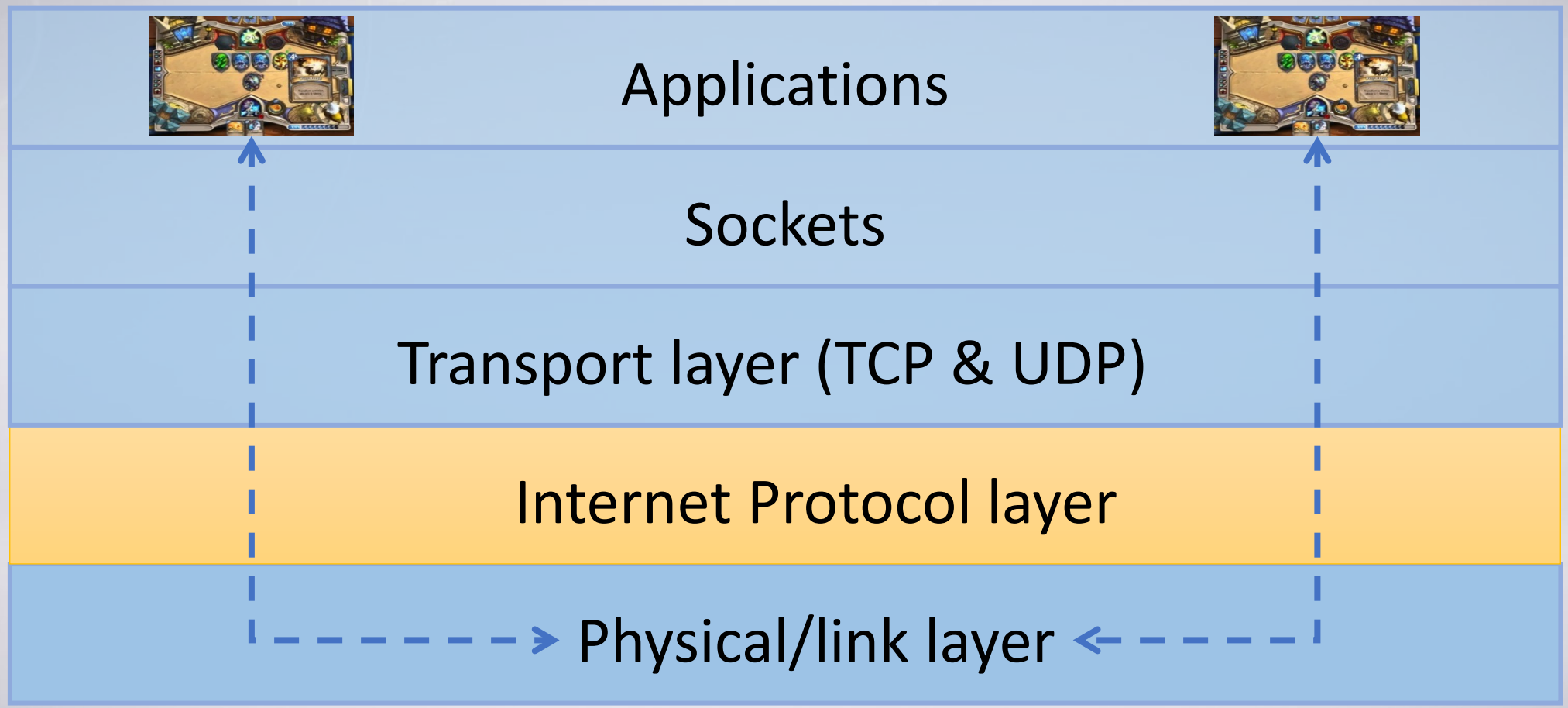
Network Stack



Physical/link layer

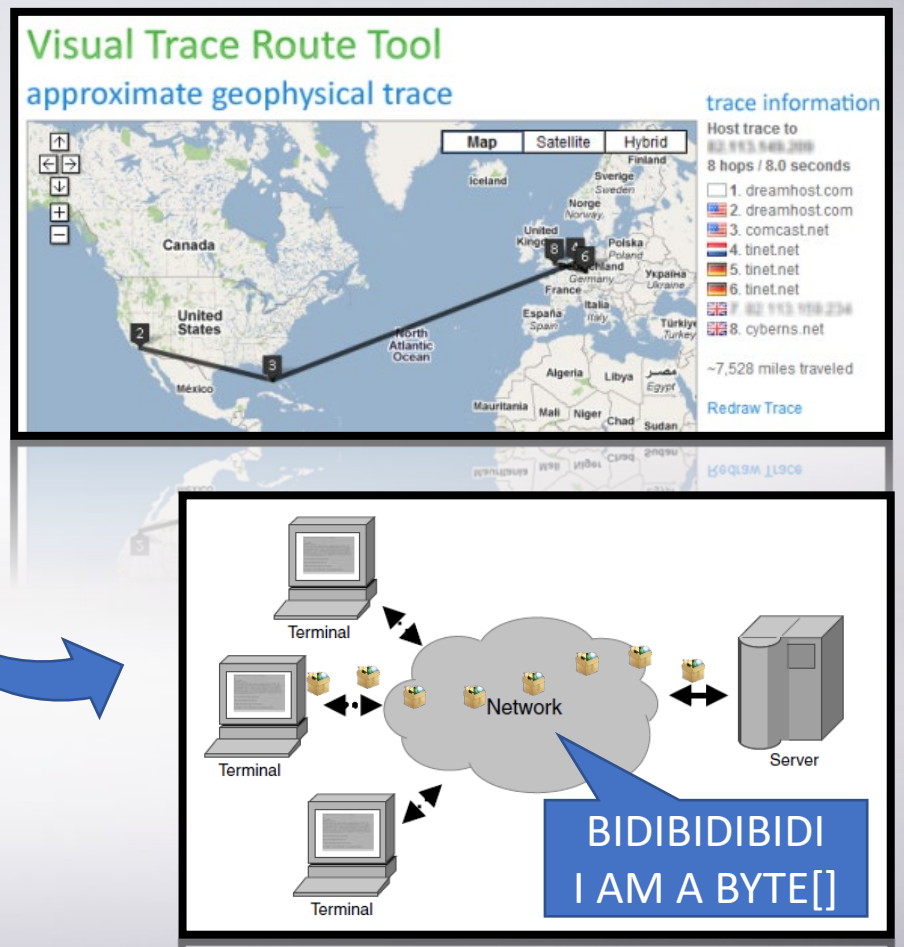
- Transmits data across a specific medium in 'frames'
  - WiFi, 4G, twisted pair, fiber optic, etc
- Makes no promises about reliability of any sort
- Defines things like:
  - ways to identify a host
  - the frame format
  - how to convert bits into signals and vice versa

Network Stack



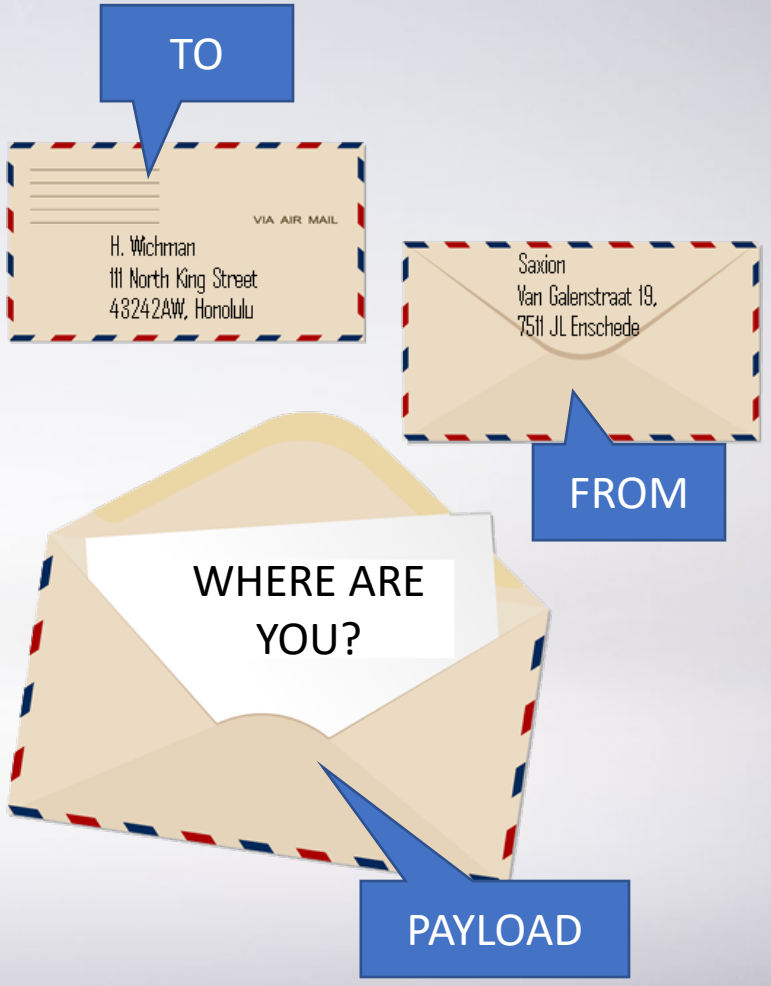
Internet Protocol Layer : host to host

- The IP Layer takes a complicated process of moving packets through a network...
- ... and makes it look like a single end to end channel:

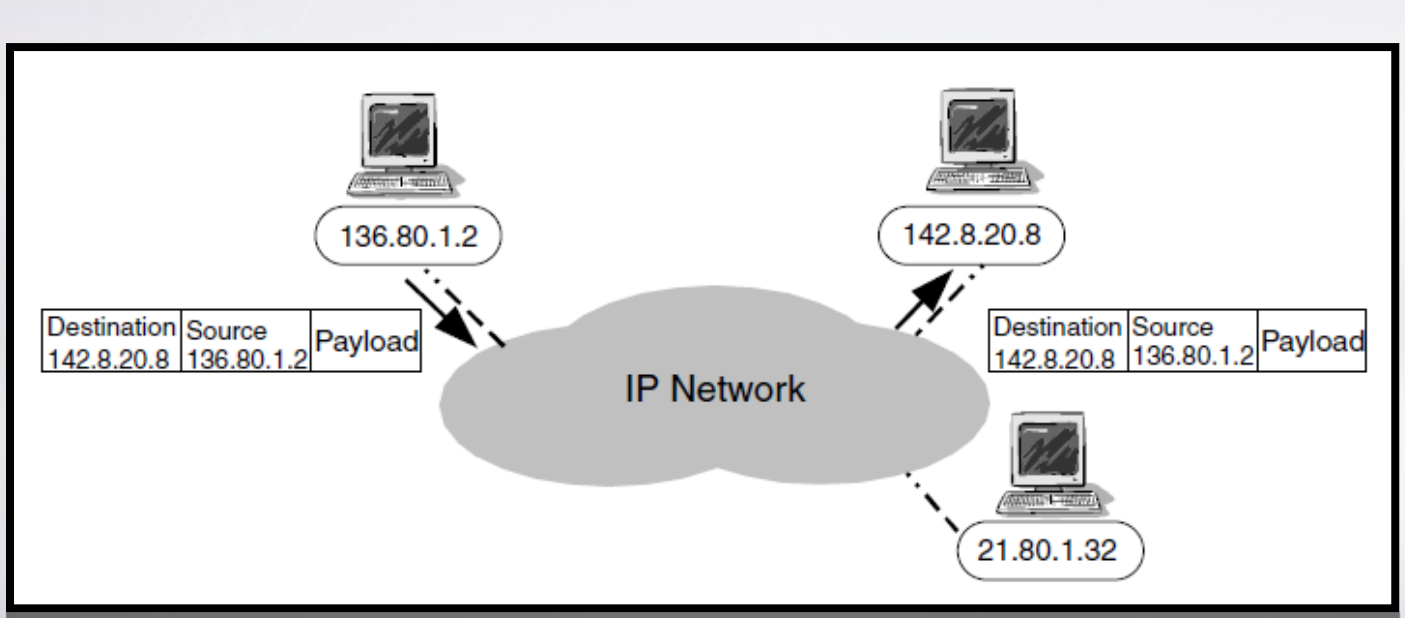


Internet Protocol is a like a postal service…

- To send a packet we need:
  - a ***to*** address
  - a ***from*** address
  - the ***data*** to send (aka payload)



…but the addresses used by the Internet Protocol just look slightly different from what your regular postal service expects…



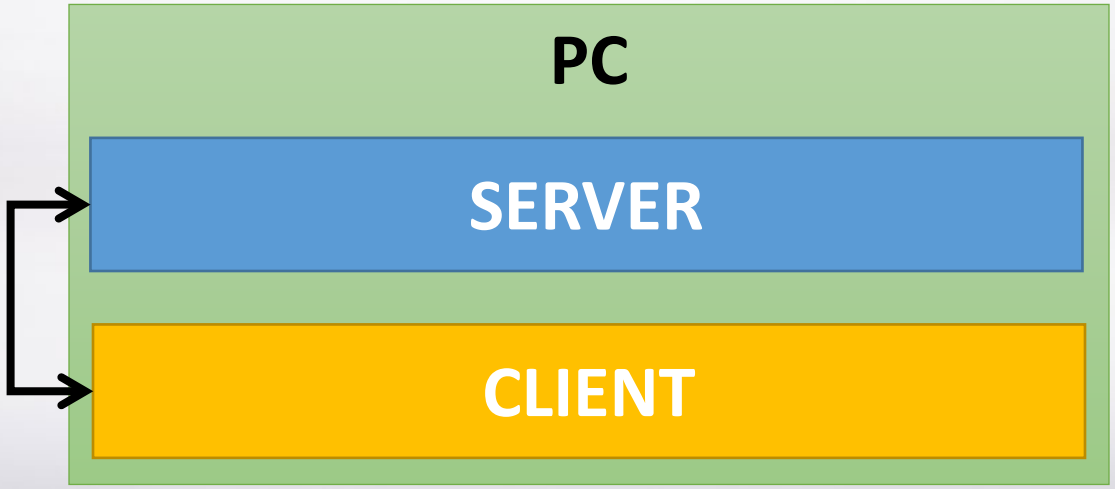
IP Address (v4)

- A 32-bit number, eg 0xC0A90001
- We know it as 4 dotted decimals eg. 192.168.0.1
- “Unique” ID for your individual machine
- In **theory** 2^{32} internet addresses
- Check your IP address using **ipconfig**:

```
C:\Users\hans>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : dynamic.ziggo.nl
   Link-local IPv6 Address . . . . . : fe80::b052:f159:14c:4933%10
   IPv4 Address. . . . . . . . . . . : 192.168.1.10
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

Some addresses are “special”

- For example: the localhost address: 127.0.0.1
- “Loopback” to your own system
- Can be used for testing purposes or running a local server



URLs

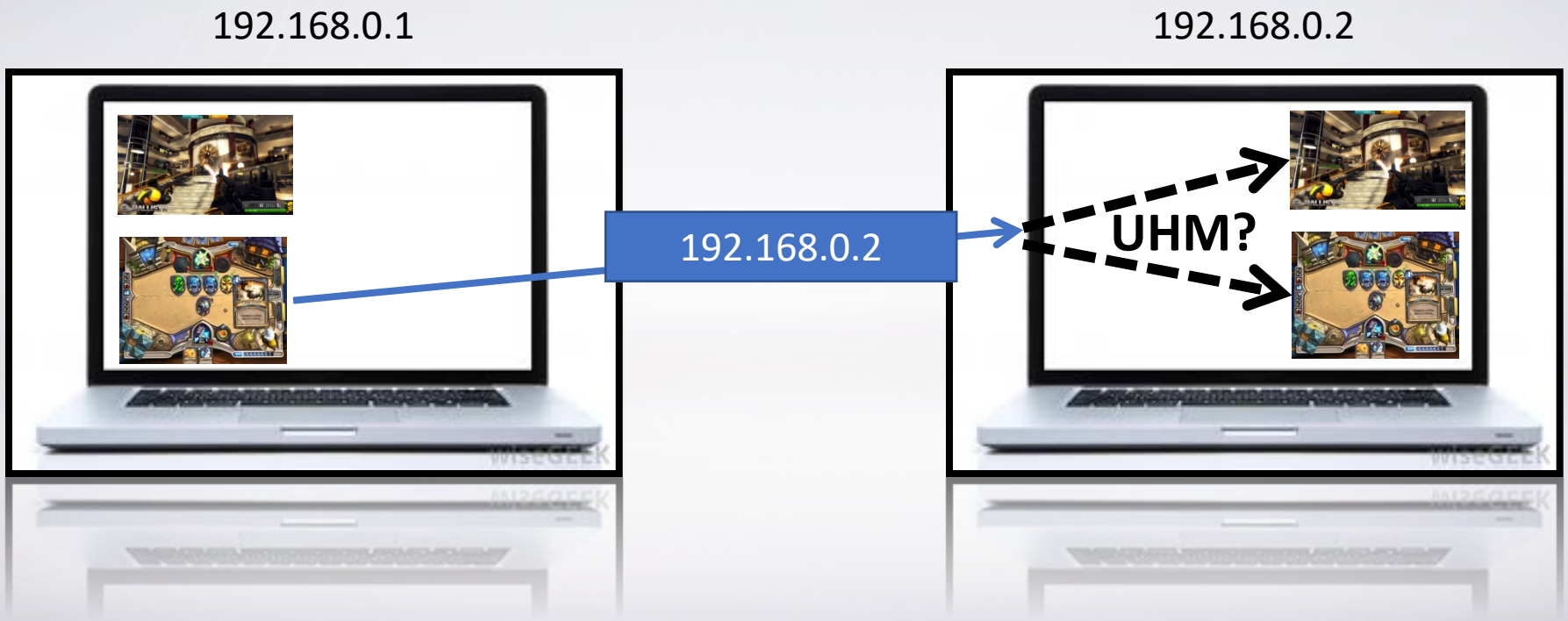
- IpAddresses are usually aliased using URL's (Uniform Resource Locator):
  - e.g. instead of 216.58.213.36, we use www.google.com
- Easier to remember, prevents failure when an IpAddress changes
- Translation is done by DNS server:
  - Ipconfig /all
- We will stick with IpAddresses for now…

Internet Protocol is a datagram service

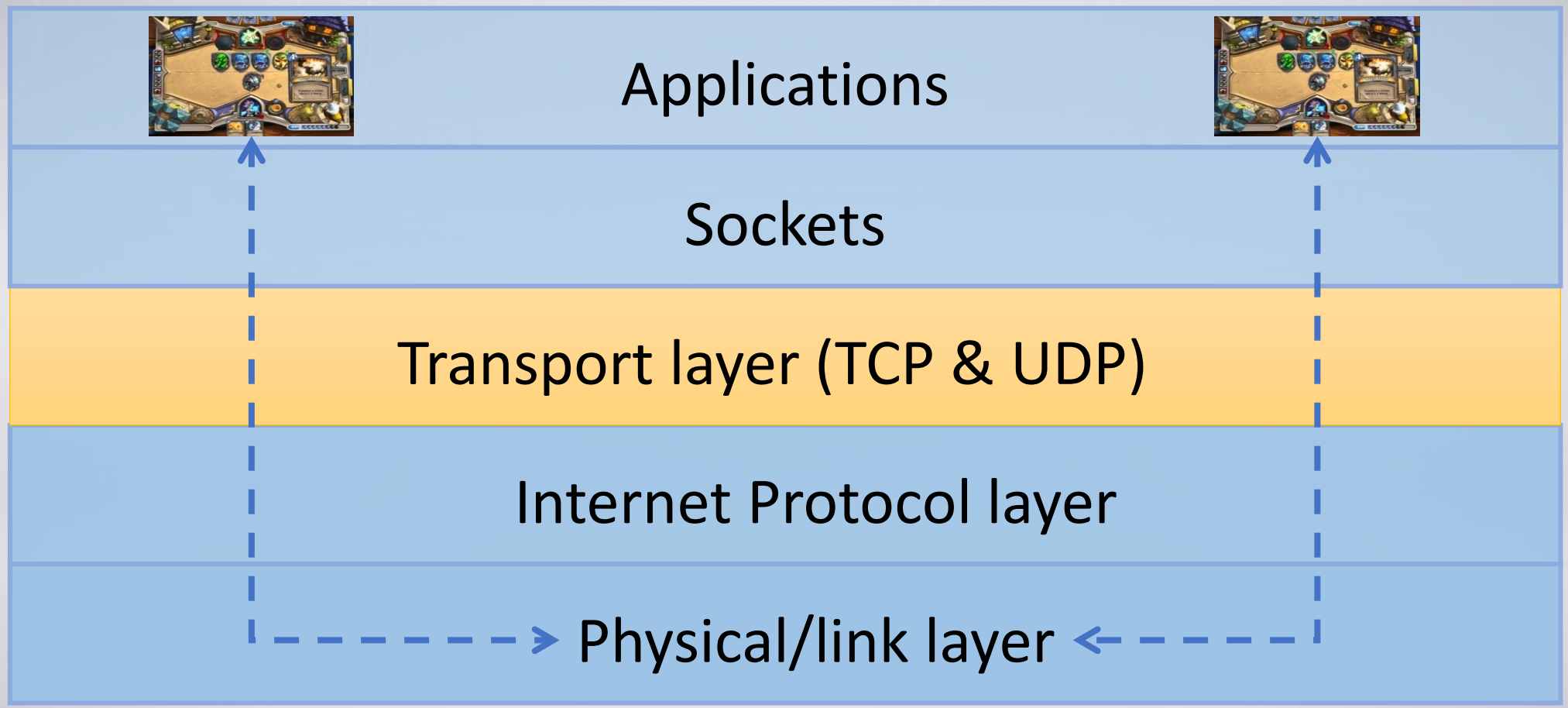
- Post → Tele-gram…. Network → Data-gram
- Each packet is handled independently
- No guarantees:
  - Packets might be **lost**
  - Packets might arrive out of order
  - Packets might be duplicated
- Handles the most basic communication between network **devices**
- Best-effort protocol → Really sir, we tried!

The internet protocol by itself is not enough

Internet protocol identifies **hosts** not **applications**



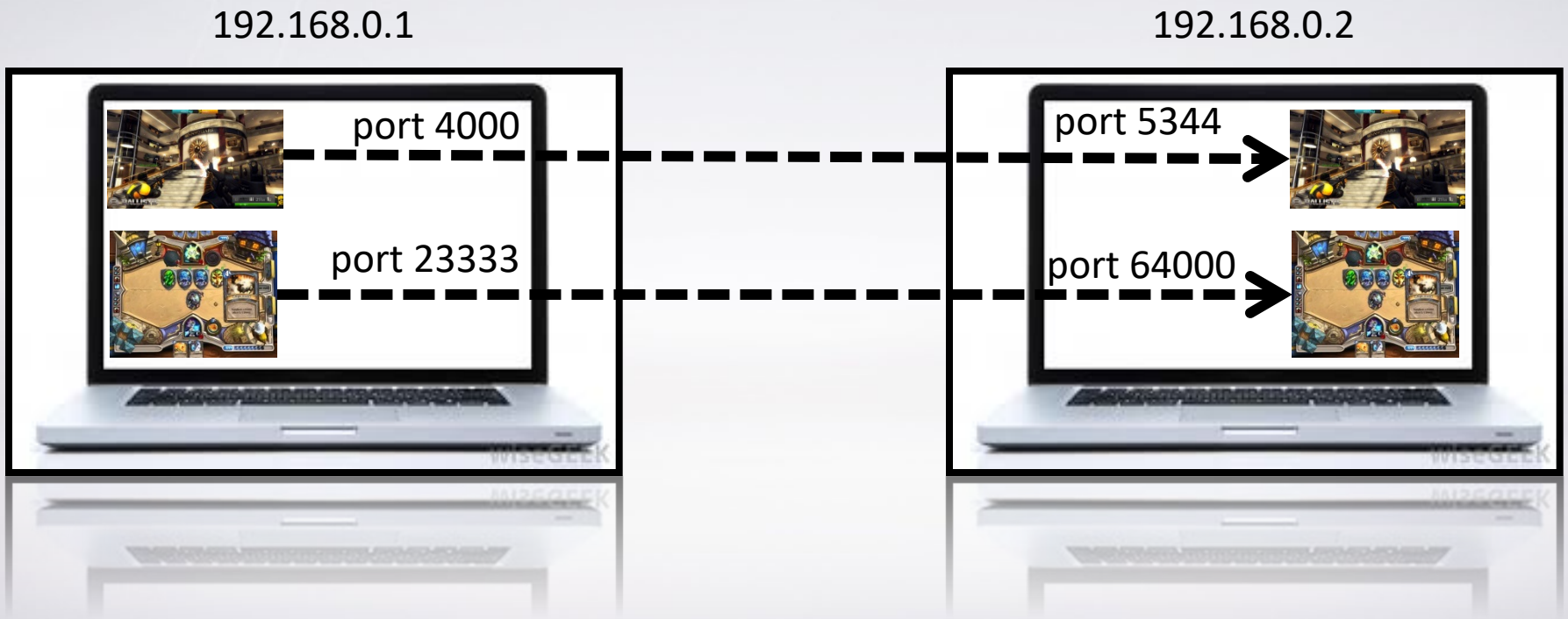
Network Stack



Transport layer : app to app

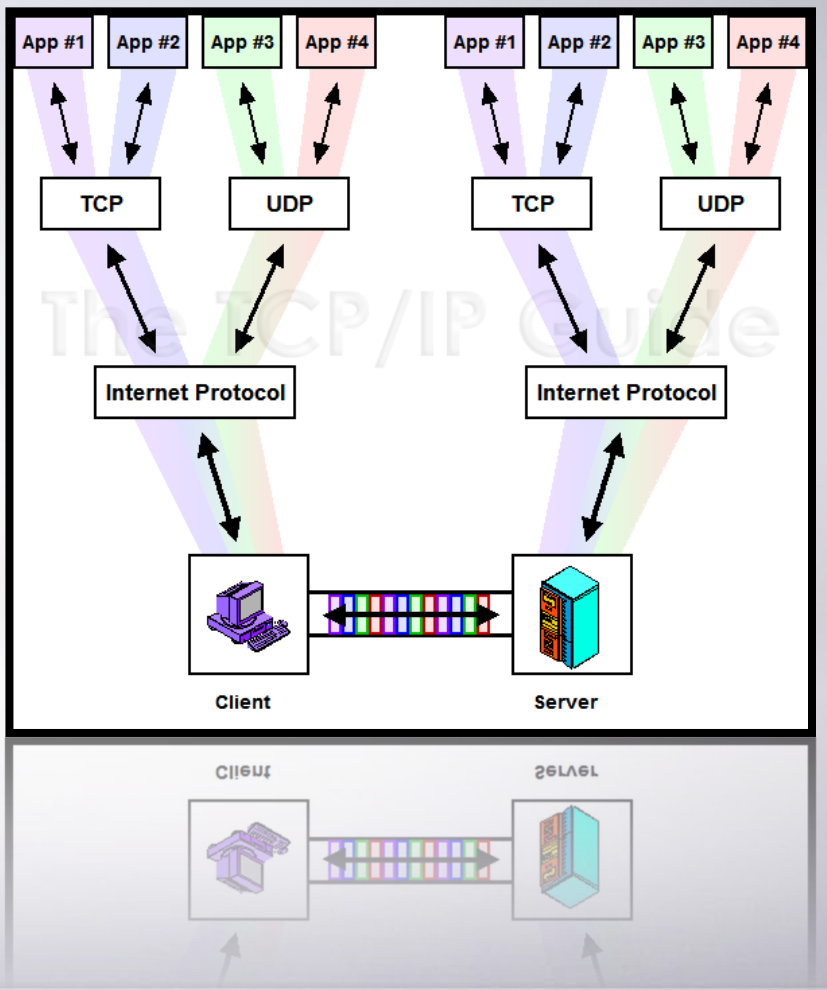
- Contains two **end-to-end** (app to app) protocols
  - UDP (User Datagram Protocol)
    - Adds nothing to IP except **application addressing**
    - Lossy, out of order, duplicated
    - Connectionless
  - TCP (Transmission Control Protocol)
    - Adds **application addressing** to IP just like UDP
    - Guaranteed delivery
    - Ordered delivery
    - No duplication
    - Connection oriented

Both transport protocols define **ports**



What exactly is a port??

- Port numbers are nothing more than an ID to (de)multiplex information streams
- Port numbers range from 0 – 65535 (16 bits)
- Each protocol (TCP/UDP) has its own range



Transport protocol: postal service analogy?

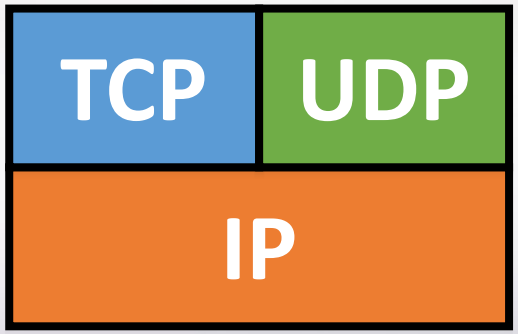
- An address gets us to the building (system)
- A room number gets us to the room (application)

IP + Port == unique endpoint

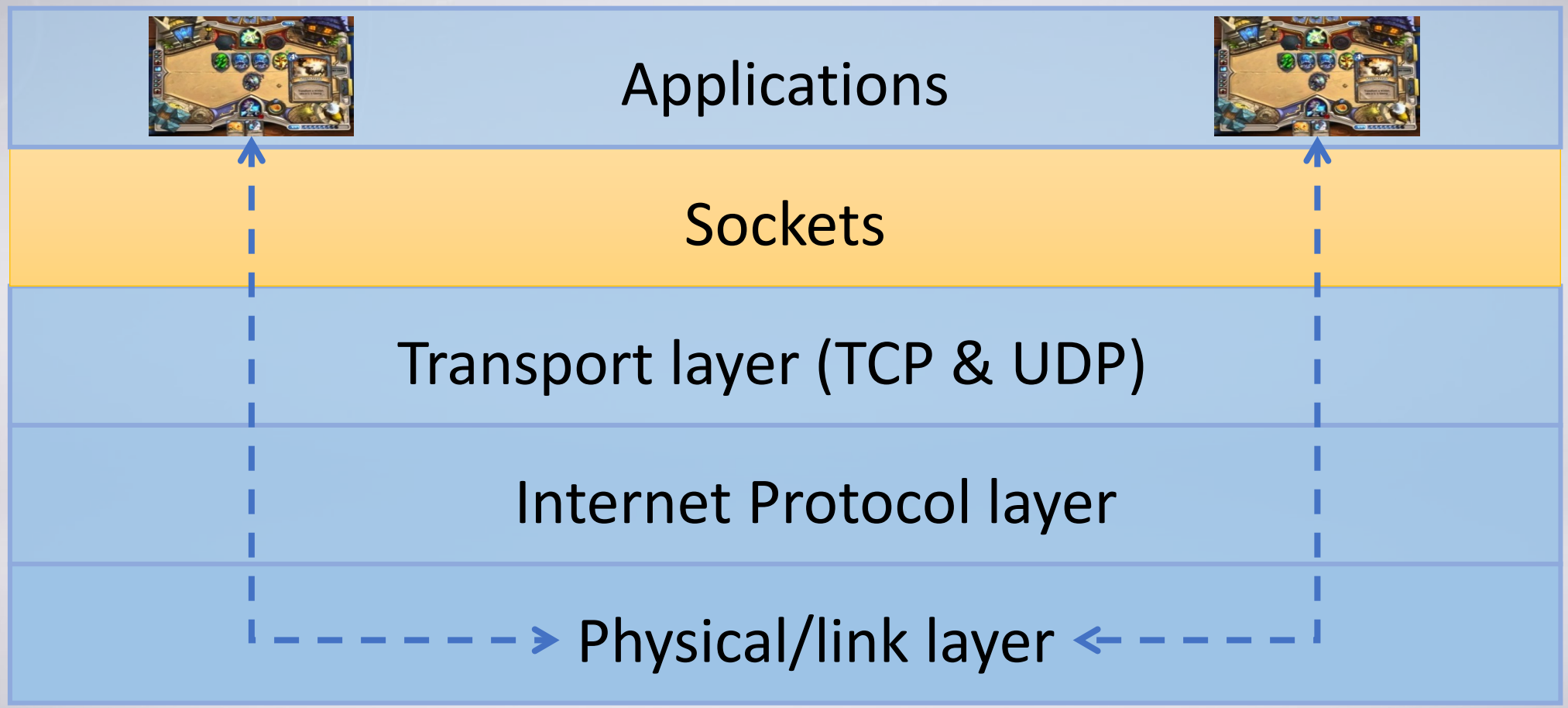
- **IP address** identifies a **host,** TCP/UDP **port** identifies an **application**
- The IPAddress & Port together are often written as x.y.z.w:port
- Port restrictions: 16 bits → 65535 ports, buuut:
  - 0-1023 → system ports
  - 1024-49151 → user/registered ports
  - **49152-65535** → dynamic ports
- You should only use the dynamic port range!

TCP/IP Protocol suite

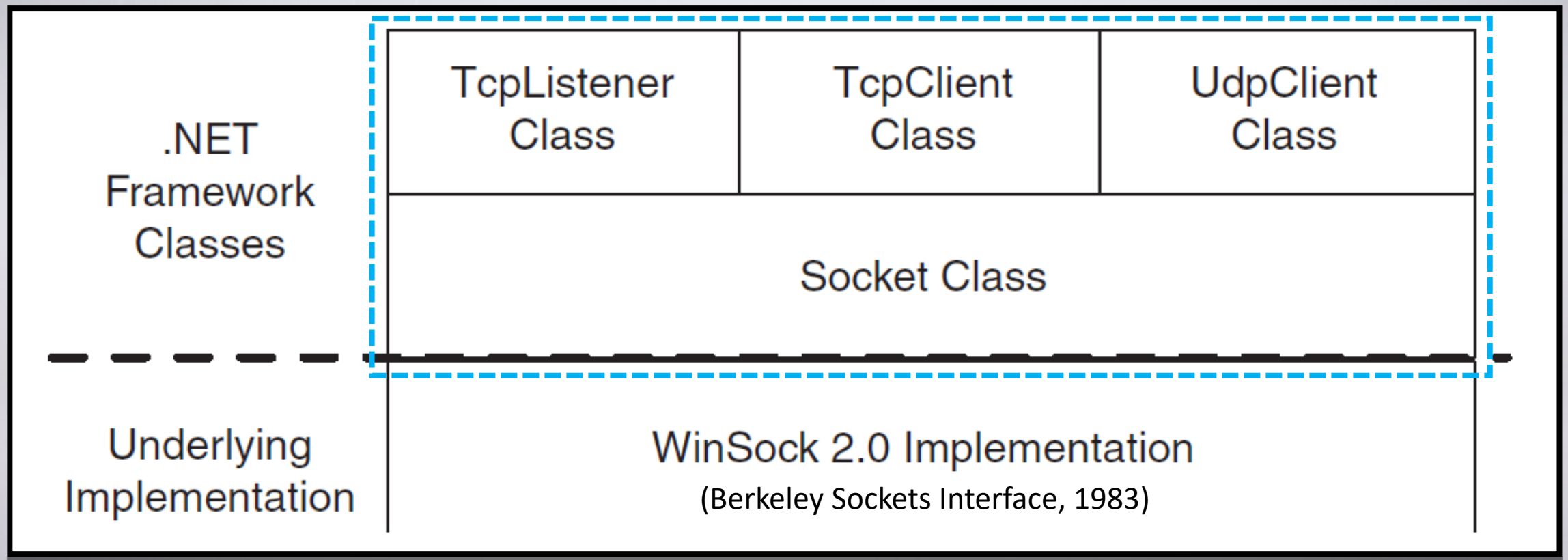
- IP + TCP/UDP is called the TCP/IP Protocol suite
- Set of related technologies aimed at transferring data over a network
- Used on both internet & standalone networks
- One out of many possible protocols
- But how do we actually access this TCP/UDP goodness?



Network Stack



Socket API in C#



Configuring sockets “manually”

```
Socket sock = null;

try {
  // Create a TCP socket instance
  sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
                    ProtocolType.Tcp);


 // Create a UDP socket instance
  sock = new Socket(AddressFamily.InterNetwork,
                    SocketType.Dgram, ProtocolType.Udp);
```

Using socket helper classes

- Much easier:

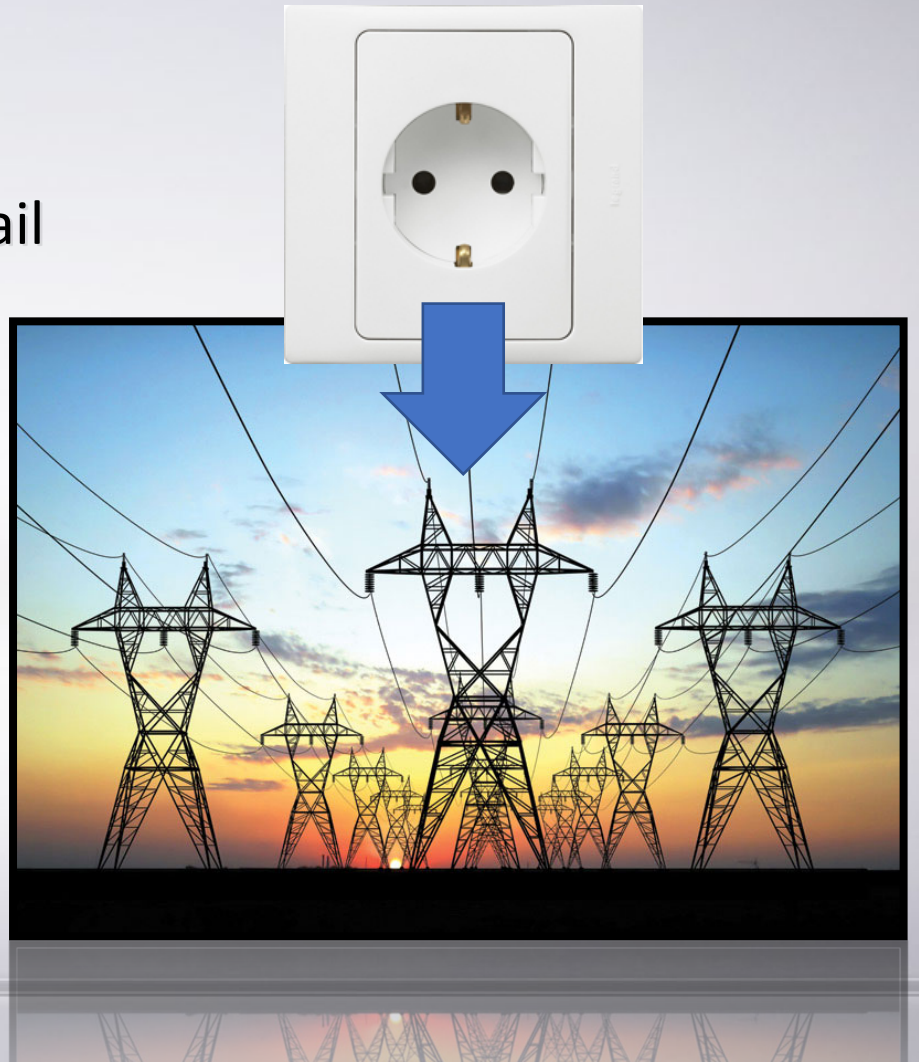
```
TcpClient client = new TcpClient (serverIpAddress, serverPort);

UdpClient client = new UdpClient();
```

- The bad news is that you have less options, but..
- The good news is that you have less options!

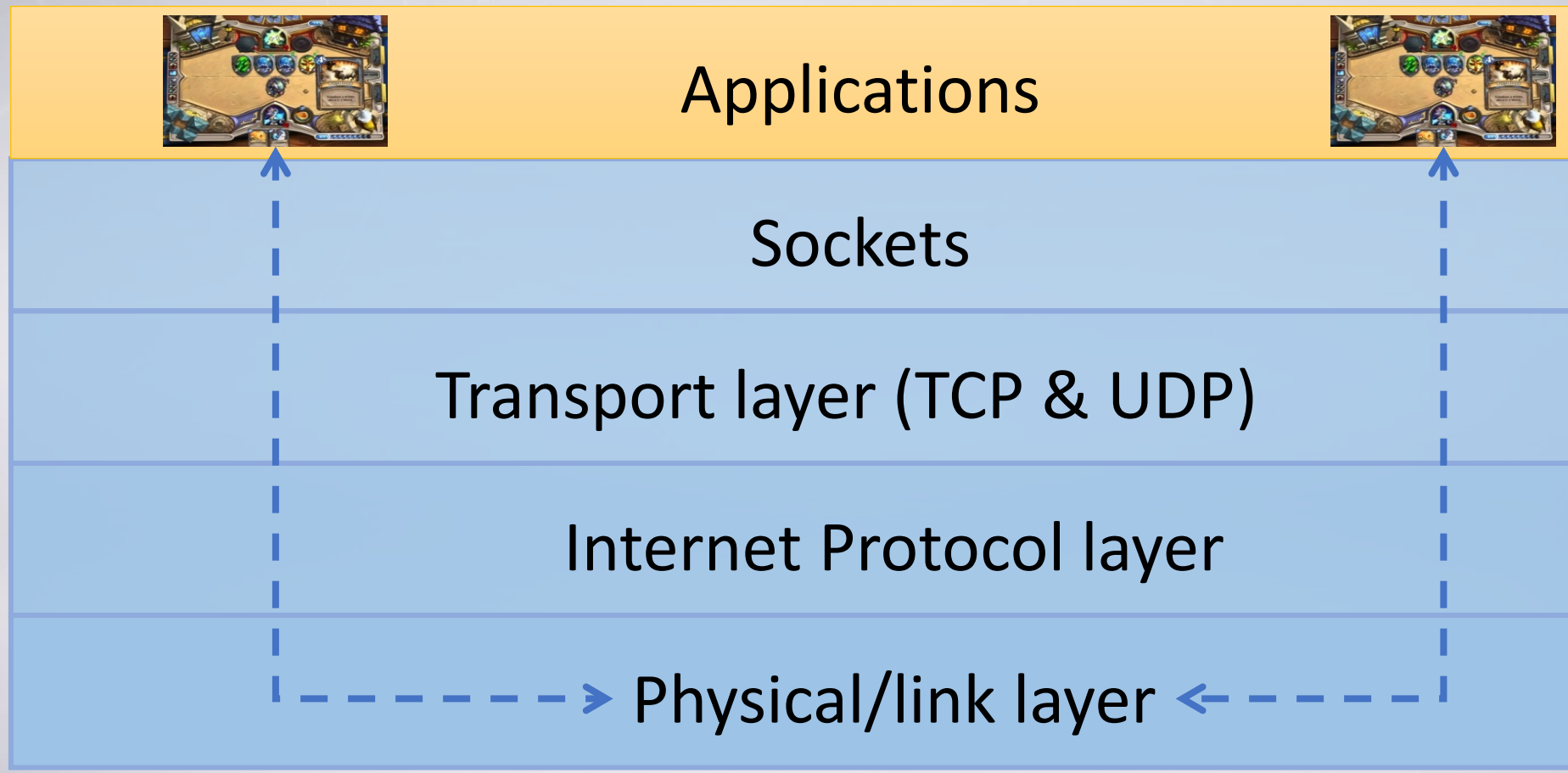
- More on how to actually use these classes later!

Networking sockets are like electricity sockets in a way

- Similar in the sense that they are:
  - "Simple" interface to a complicated thing
  - Usable without knowing every underlying detail
  - Plug and p(l/r)ay
  - Limited amount of different socket types



Network Stack



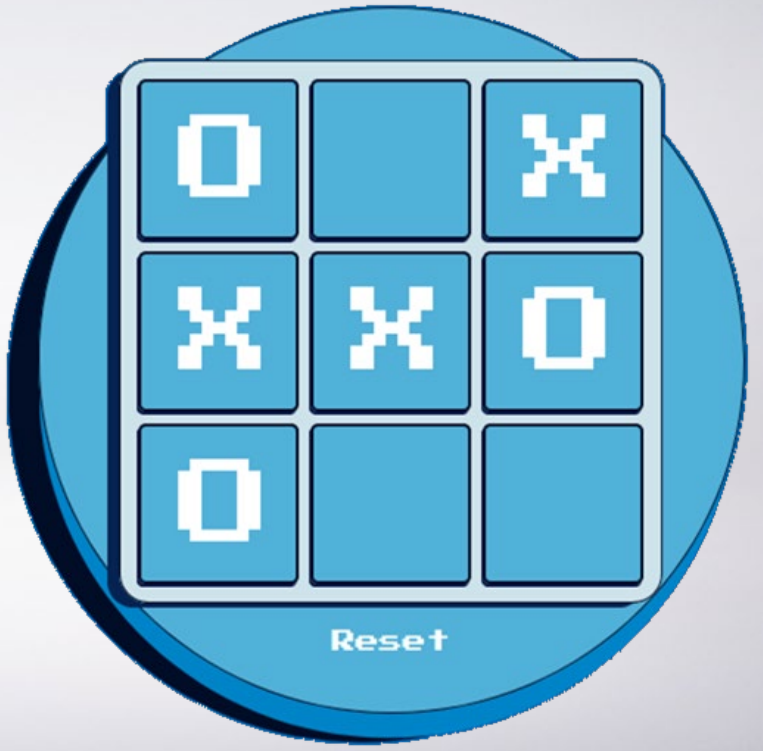
Application protocol

- **IP Protocol** was…
  - an agreement on how packets are routed between systems
- **TCP / UDP Protocol** was…
  - an agreement on how packets are routed between applications
- **Application protocol** is…
  - an agreement on what sort of messages your game/application requires/is able to process etc

(So in other words: "protocol" means "agreement")

Application protocol example

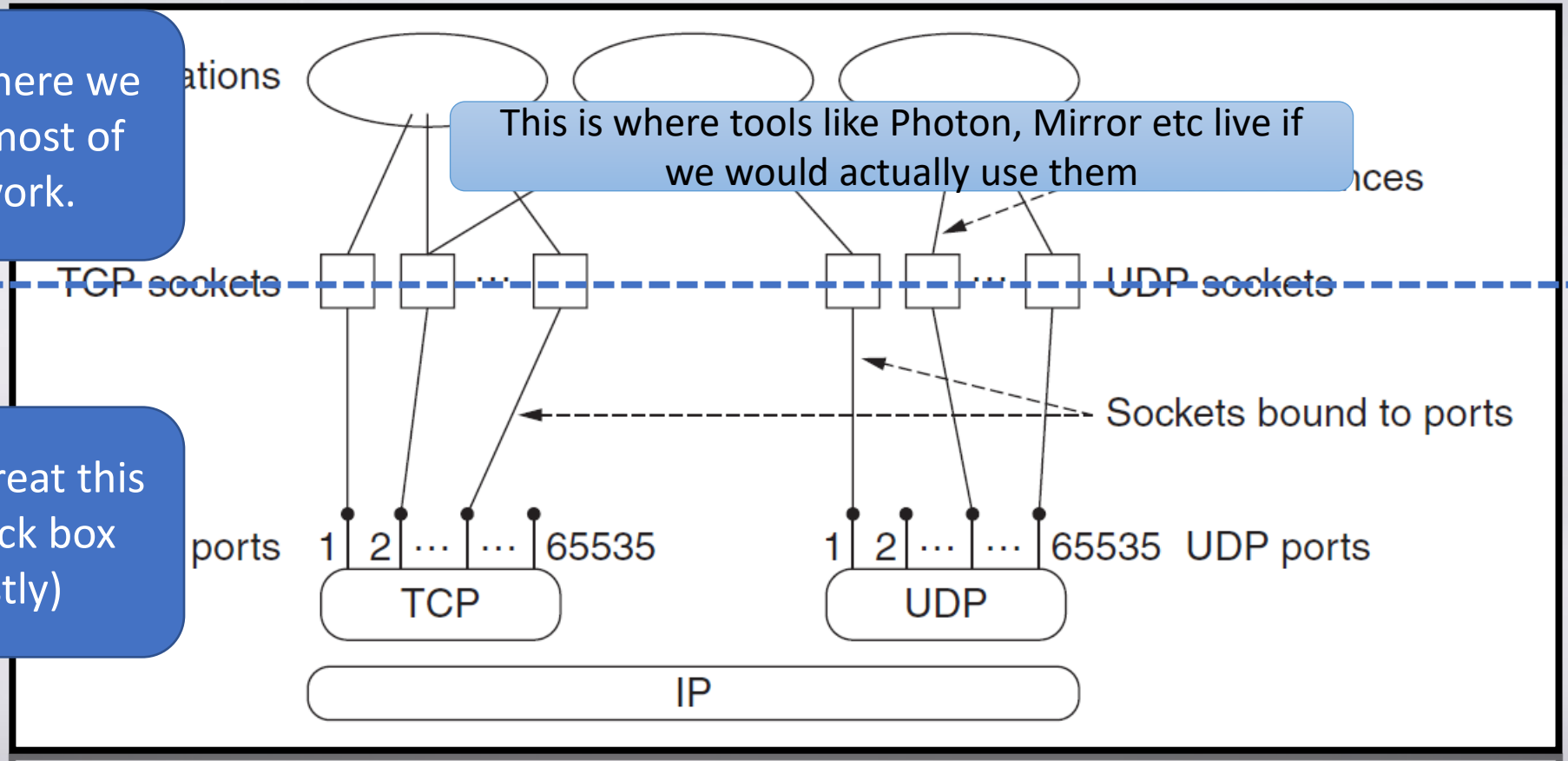
- A networked game of Tic Tac Toe might require:
  - a "make move" message
  - a "board update" message
  - a "game over" message
- You won't find these 'protocol objects' in a description of the TCP/IP stack…
- How these kind of messages are actually implemented is the topic of lecture 3



Applications, sockets, ports

This is where we will do most of our work.

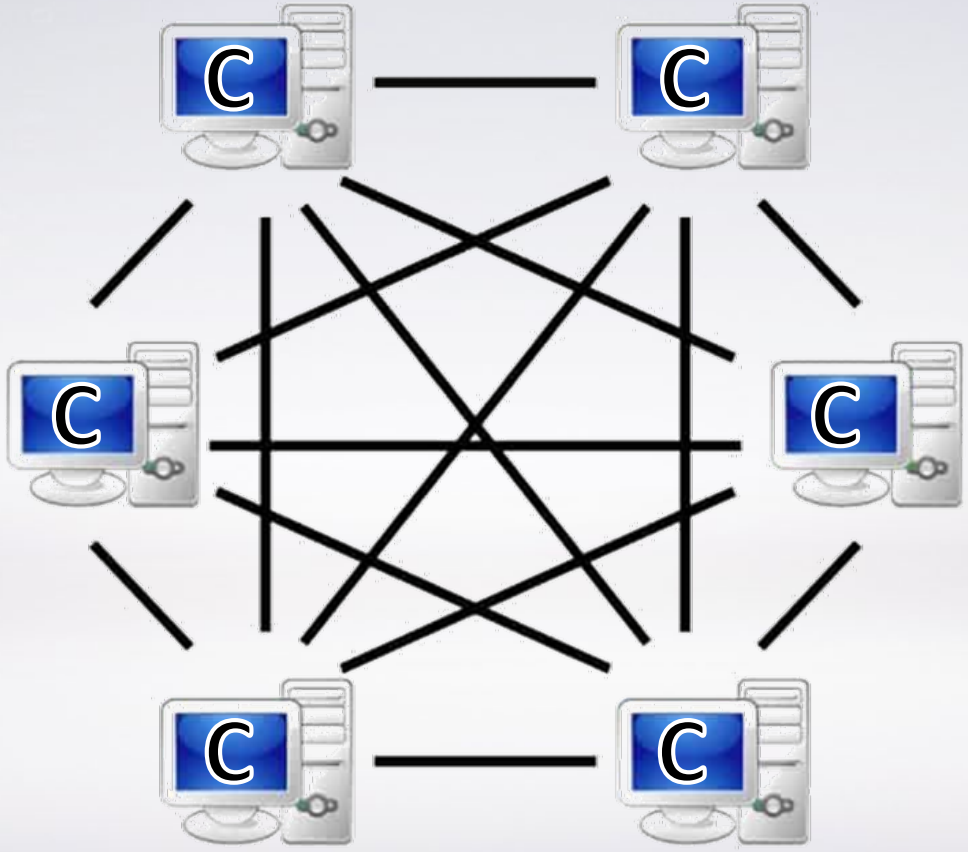
We will treat this as a black box (mostly)



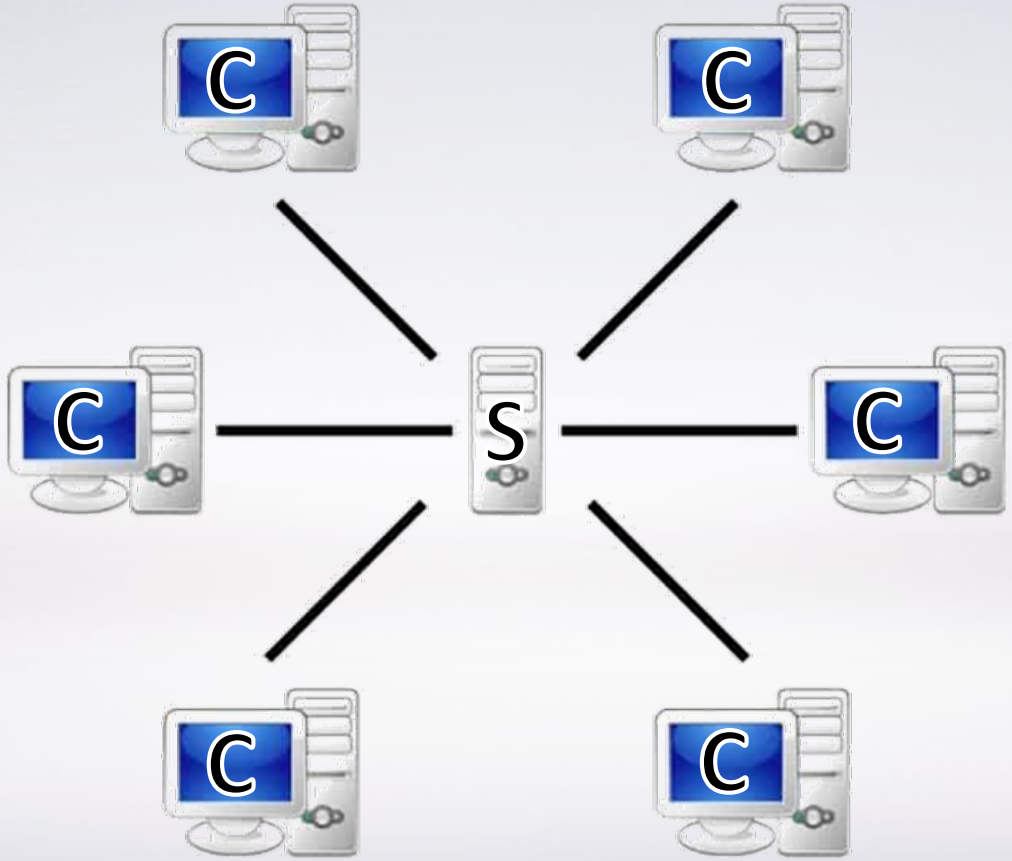
Let's get practical!

Step 1. Deciding on a setup

Application setup 1 - Peer 2 peer



Application setup 2 Client-Server (*our* setup)



Our setup

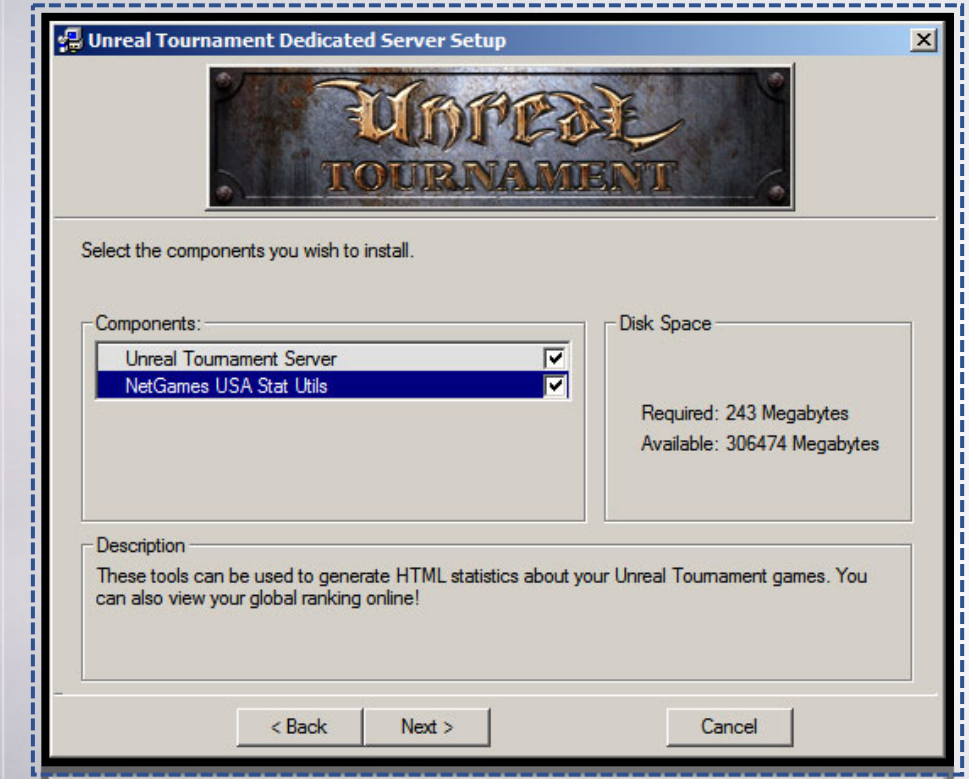
Server

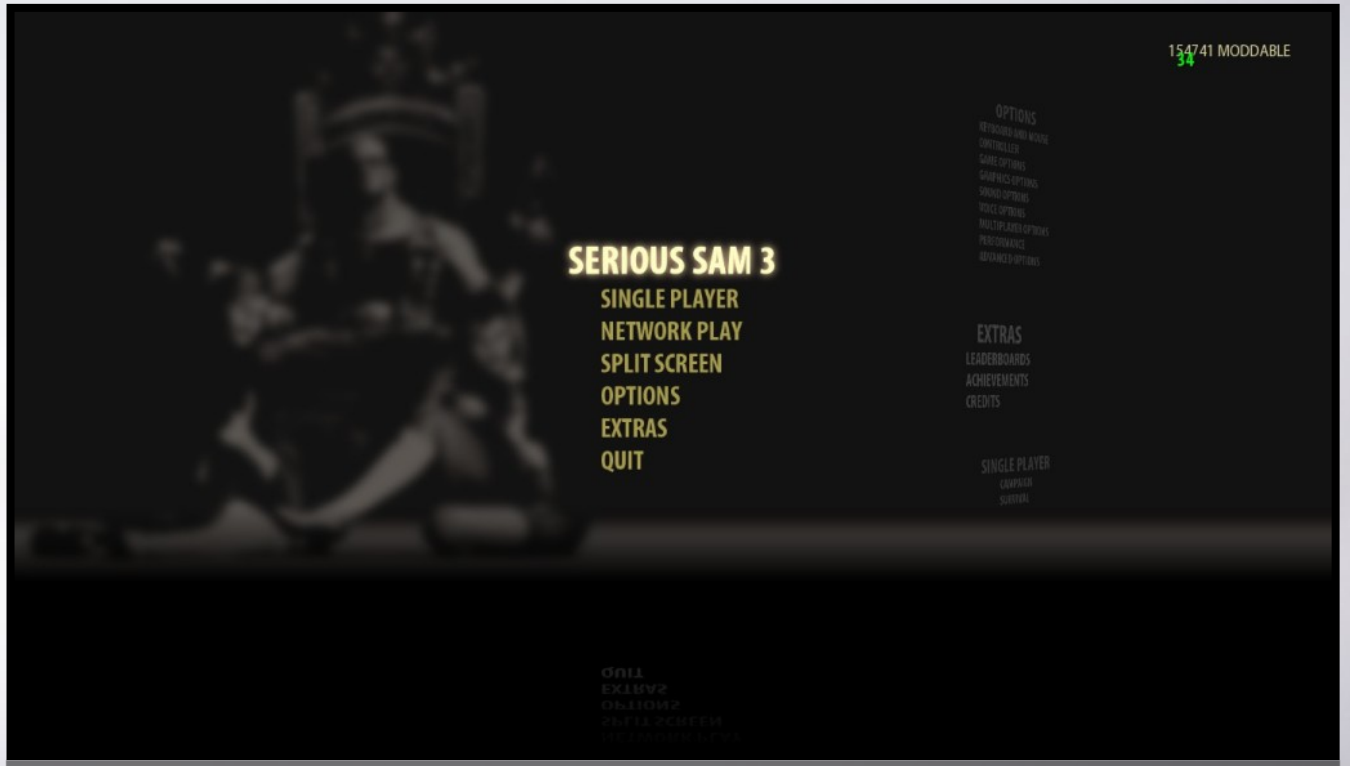
- Waits for connection
- Can connect to x clients
- Long lived?
- ***Dedicated (not a player)***
- ***Authoritative***

Client

- Initiates connection
- Connected to 1 server
- Short lived?
- 1 client = 1 player
- Dumb terminal

Dedicated vs non dedicated





Authoritative vs non-authoritative

NON AUTHORITATIVE: Client decides

C: “Hey server I found 10000 gold in this **empty** bag”

S: “Great! Wanna buy stuff?”

C: “My mage has bought full **plate**!”

S: “Unbeatable man, unbeatable!”

AUTHORITATIVE: Server decides

C: “Hey server I clicked on this bag”

S: “You have found 0 gold! GZ!”

C: “Hey server, GIEV full plate!”

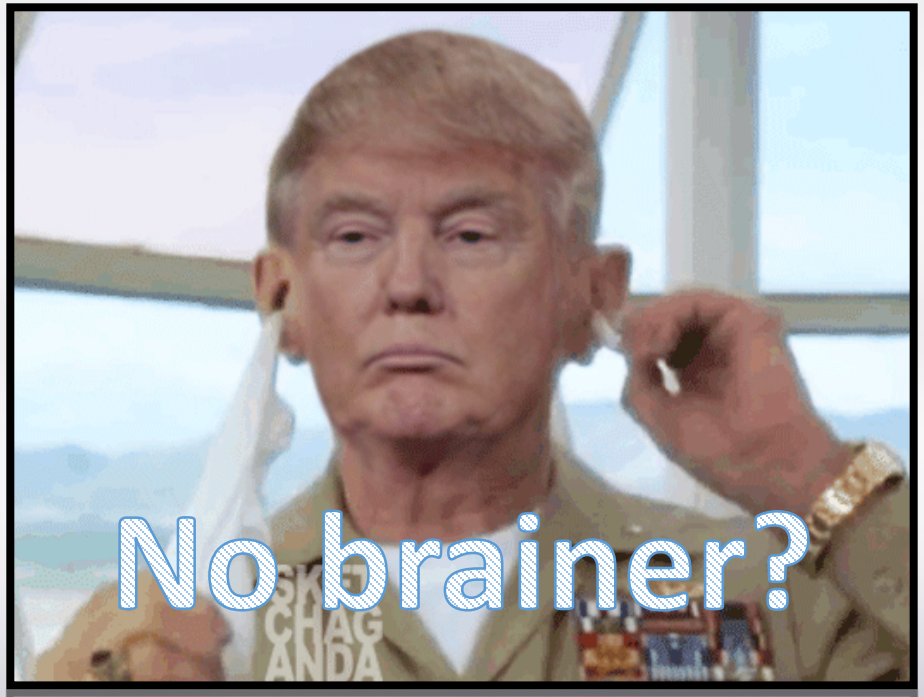
S: “Dude get real, you are a mage!”

Turn based vs real time setup

Step 2. UDP or TCP?

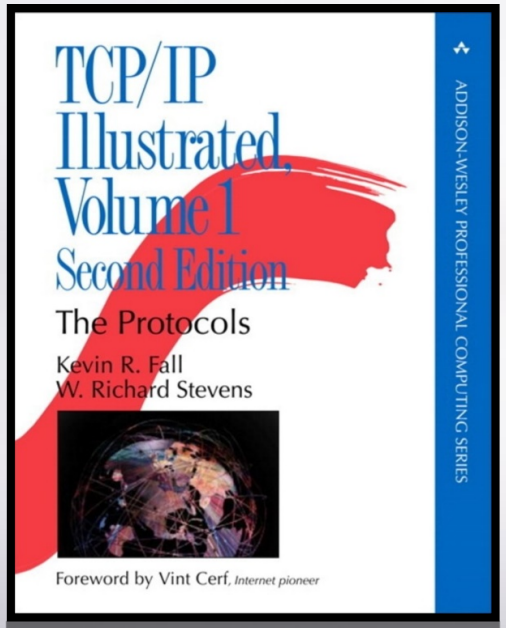
UDP or TCP?

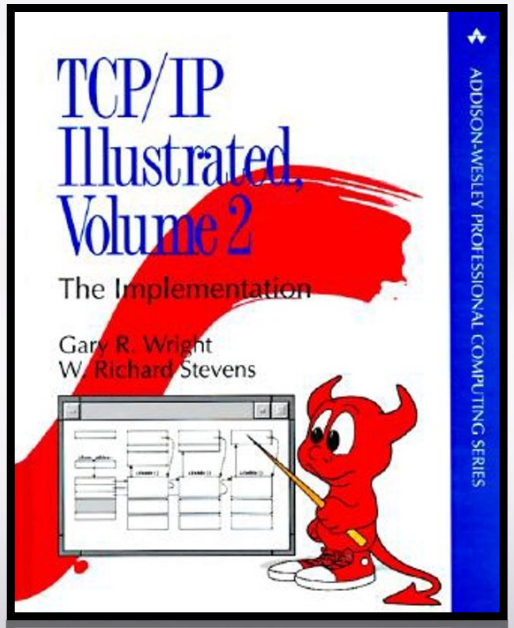
- UDP: Unreliable, duplicates, out of order
- TCP: Reliable, no duplicates, in order

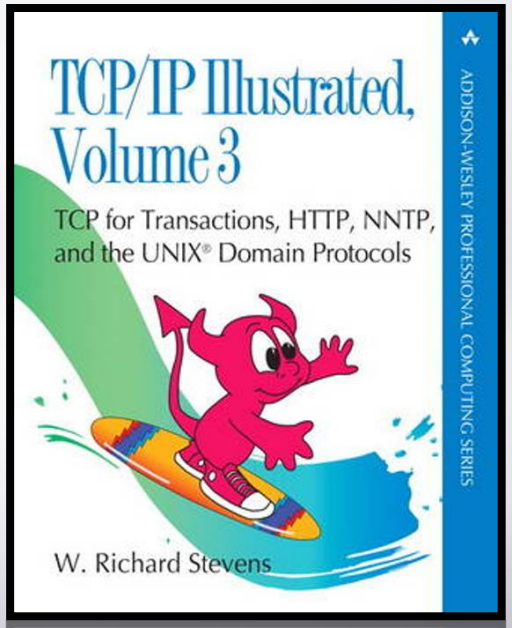


In reality, the choice is not that easy …

- TCP is a much more complicated protocol
- Enforcing reliability requires a lot of complicated control mechanisms
- A lot of complicated control mechanisms imply a lot of overhead







So what do we choose?

- TCP *reliable*, but a lot of *overhead* (i.e. latency)
- UDP *unreliable*, but much *faster* than TCP
- The choice depends on what you want to build:
  - Fast paced non critical data: Unreliable/UDP is fine
    - Video, player positions, etc (streams)
  - Data that should not be lost: TCP or custom reliable protocols
    - Bank transfer, player hit, etc (events)

- This course:
  - Turn based, limited amount of data, 6 weeks
  - TCP is the better choice (also for our sanity)

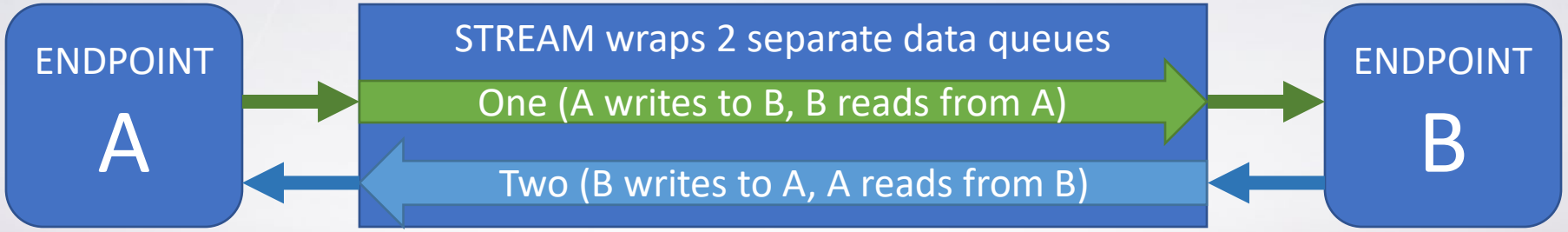
Step 3. Getting started with TCP

Example 001_basic_tcp_echo_server_commented

- Read server first, then client
- Things to note/understand:
  - Use of the "localhost" loopback address
  - Relation between TCPListener/TCPClient
  - Relation between TCPClient/NetStream
  - NetStream Read/Write (& streams in general)
  - Blocking calls
  - Use of inBytes buffer
  - String – byte conversion (and back)
  - Closing / Exception handling / ...

(Net)stream in a picture

- (Net)stream abstracts a bidirectional connection between 2 (fixed) endpoints:



- Write (byte[] buffer, int offset, int count)
- Read (byte[], int offset, int count)

Note that we HAVE to pass in an amount of bytes to read/write…

That's it! .... for now ....

Plan for the lab tomorrow

- Download lecture 1 + assignment 1 (and try to finish it right away!)
- Assignment 1 is all about:
  - reviewing the basic theory from this lecture
  - doing experiments with the given example/TCP
  - understanding the issues/limitations of the current setup before next week
  - it's an easy start, so don't waste it!
- To do ***now***: install the latest Unity 2019 LTS version: ***2019.4.21f1*** (Use this version for the whole networking course!)

Important lab announcements (will post to bb)

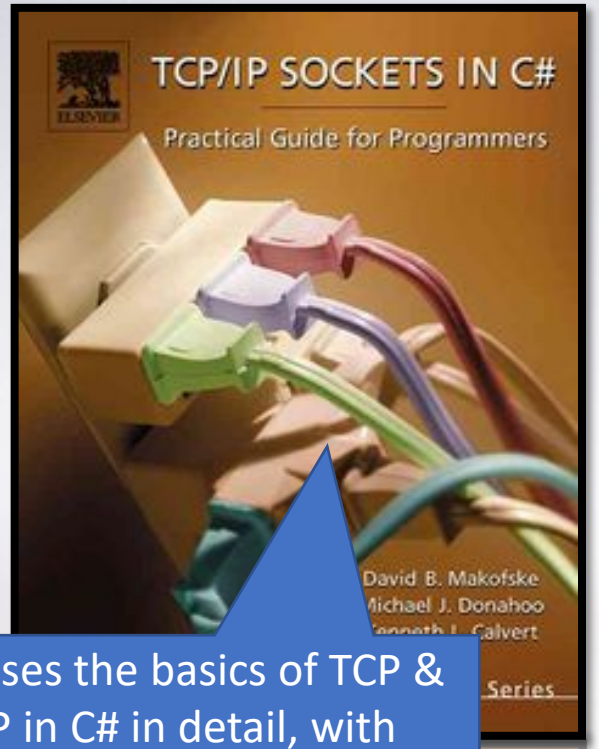
- ECM2V.Ea - Tuesday 9:15 - Hans
- ECM2V.Ec - Tuesday 9:15 - Paul
- ***ECM2V.Ed - Tuesday 14:00 - Hans***

- ***Schedule will be updated, but probably not before tomorrow.***
- *I will post this including the class list to blackboard after the lecture.*

Plan for next week

- Working with networking/TCP has some challenges, such as:
  - Dealing with messages boundaries
  - Dealing with blocking while serving multiple clients
  - Error handling
  - Debugging
  - Centralizing code
  - Protecting against evil clients etc
- These are the topics for next week, so we can start building our ....

Some book recommendations …



Discusses the basics of TCP & UDP in C# in detail, with different approaches & solutions, while discussing a lot of API calls

Discusses a lot of details of the networking stack & realtime multiplayer game programming in C++