SOCKETS

Client and Server Sockets

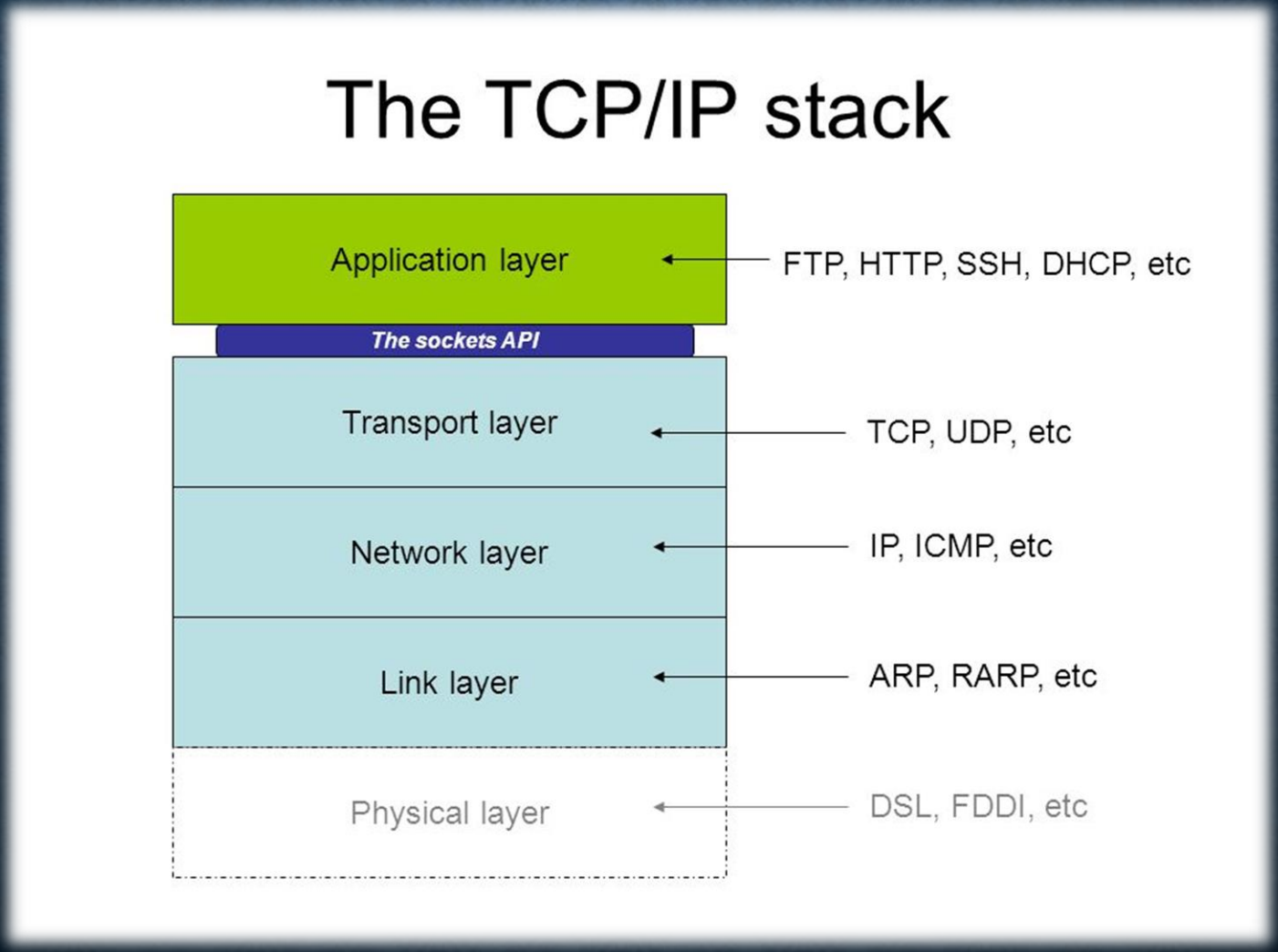
COS216
AVINASH SINGH
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF PRETORIA

https://www.youtube.com/watch?v=XiFkyR35v2Y

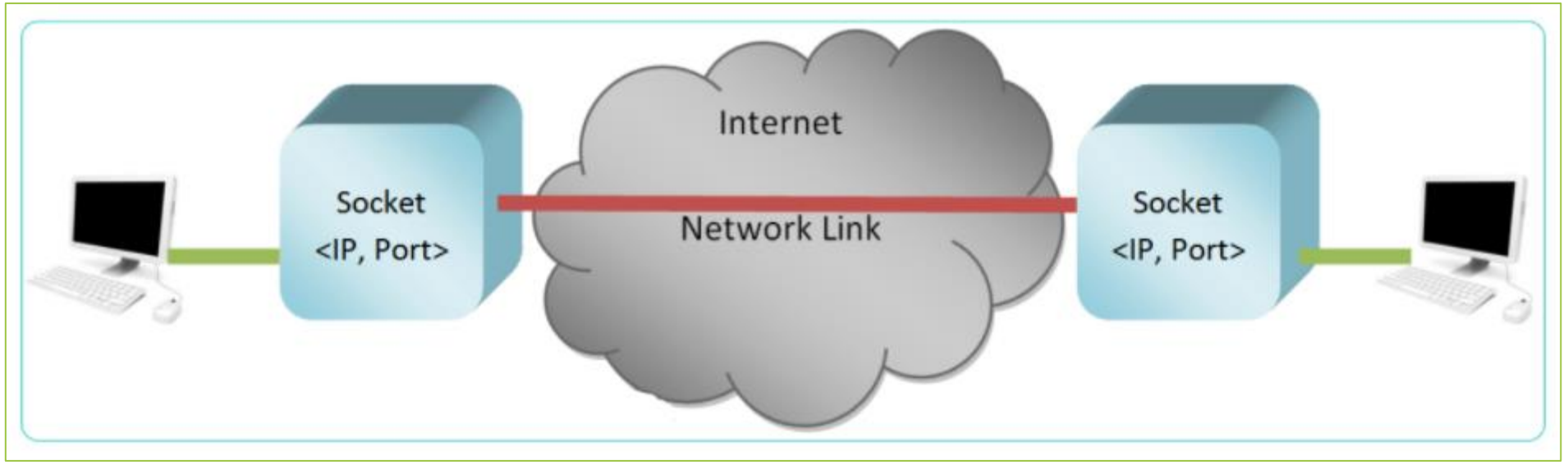
SOCKETS – OVERVIEW

- Network sockets are endpoints for sending or receiving data over a network
- Provides lower level data transmission facility
- HTTP and FTP are applications build on top of sockets
  - So are most other application protocols
- Sockets are implemented between the application/session layer and the transport layer
- Many programming languages have built-in socket functionality

SOCKETS – STACK



Sockets



SOCKETS – CONNECTION

- HTTP can only pull data
  - HTTP request is sent to the server every X seconds asking for the latest data
  - Unnecessary requests are made, because most of the time there is no new data
  - Example: If cryptocurrency prices change, the server should push the new prices to the client, instead of waiting for the client to pull the data
  - Some degree of pushing can be done with the new HTML5 Server-Sent Events API

https://www.tutorialspoint.com/html5/html5_server_sent_events.htm

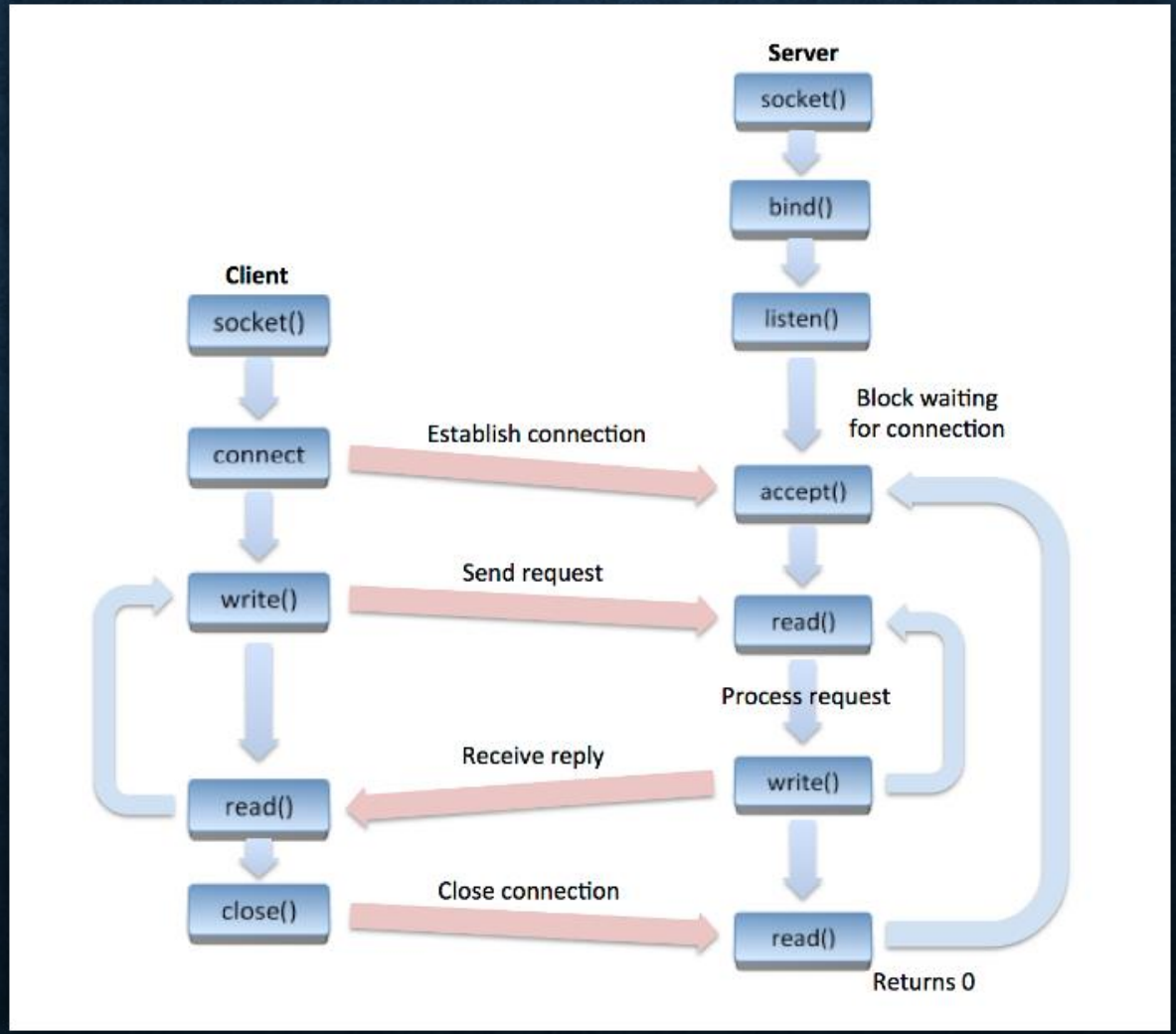
SOCKETS – CONNECTION

- Sockets can pull or push data
  - Can pull or push, but not at the same time, requires a previous pull/push to be finished before starting a next one
  - Requires the implementation of the socket server (webserver) and the socket client (website)
  - Example: If the new cryptocurreny prices are available on the server, push them to the client's website the moment they become available

SOCKETS – CONNECTION

- Socket servers should have:
  - Address
    - The address that clients connect to
    - An IP address or FQDN
  - Port
    - The port that clients connect to
    - Must be a port that is available/open
    - First 1024 ports are well-known ports used by common applications (FTP, HTTP, email, etc)
    - Don't use well-known ports or any higher port that is used by another application
    - Use something like port 10000

SOCKETS – PROCESS



SOCKETS – PROCESS

- Server starts and wait for incoming connections
- At some point a client connects and establishes a connection with the server
- Data is now transmitted between the client and server
- After communication is done, the client disconnects from the server
- The server is now listening again for any new incoming connections

SOCKETS – PROTOCOLS

- TCP is connection-oriented
  - State and connection is maintained
  - Establishes a connection and then transmits data as a stream while staying connected
  - TCP is typically used by socket libraries

- UDP is connection-less
  - State and connection is not maintained
  - Packets are sent in any order without maintaining a continues connection
  - UDP can be used (or rather abused) for sockets, but most libraries have not implemented UDP support (since it is counter productive)

SOCKETS – PROTOCOLS

- TCP is connection-oriented
  - State and connection is maintained
  - Establishes a connection and then transmits data as a stream while staying connected
  - TCP is typically used by socket libraries

- UDP is connection-less
  - State and connection is not maintained
  - Packets are sent in any order without maintaining a continues connection
  - UDP can be used (or rather abused) for sockets, but most libraries have not implemented UDP support (since it is counter productive)

SOCKETS – DATA

- Any kind of data can be send over socket
  - Text or binary
- To send structured data, a common convention is to encode it as a JSON string

SOCKETS – BUFFER

- Sockets uses buffers for communication
  - A small block of memory where incoming data is temporarily stored
  - The data has to be retrieved from the buffer in order to clear it and make room for new data
- Buffers have a fixed size
  - A good size is between 1MB – 10MB
  - Don't make buffers too large, since they might consume too much memory, especially if there are multiple simultaneous connections
  - Don't make buffers too small, since it might slow down the connection
  - If you have a 1MB buffer and want to send 3MB of JSON data, it will arrive in 3 x 1MB chunks. You have too reassemble the chunks at the receiver, JSON data can be simply concatenated as a string before decoding/parsing it

SOCKETS – PHP SERVER

- Sockets are natively supported in PHP

- PHP has a default timeout of typically 30 seconds (might vary)
  - Scripts that take longer to execute are killed by the webserver
  - Avoids possible attacks and bugs in your code
  - Ensures that the server stays responsive in case there are too many scripts that do not exit
  - You have to overwrite this timeout, since sockets must continue running forever (or until the server closes the connection)

SOCKETS – PHP SERVER

- TCP Socket Server
  - In PHP sockets just provide the connection, and are TCP sockets
  - That means they will work with TCP Socket Clients
  - Transport Layer (Layer 4)

- Websocket
  - A JS Client using Application layer protocols (Layer 7)
  - Has its own set of rules defined by the RFC https://www.rfc-editor.org/rfc/rfc6455
  - Used in the browser
  - In PHP we use TCP sockets and then cater for Websockets protocol

SOCKETS – PHP SERVER

- Example Code

SOCKETS – JS CLIENT

- Sockets are not natively supported in JS

- There exists a W3C standard called WebSockets
  - Supported by most browsers
  - Can be used for client or server

SOCKETS – JS CLIENT

```
// Create a new socket and connect
var socket = new WebSocket("ws://127.0.0.1:10000");
```

SOCKETS – JS CLIENT

```
// Receive data
socket.onmessage = function(event)
{
        var json = JSON.parse(event.data);
        console.log(json.message);
}
```

SOCKETS – JS CLIENT

```
// Send data
function send(command)
{
        var json = { "command" : command };
        socket.send(JSON.stringify(json));
}
```

SOCKETS – JS CLIENT

```
// Quit by sending a command
function quit()
{
        send("quit");
}
```

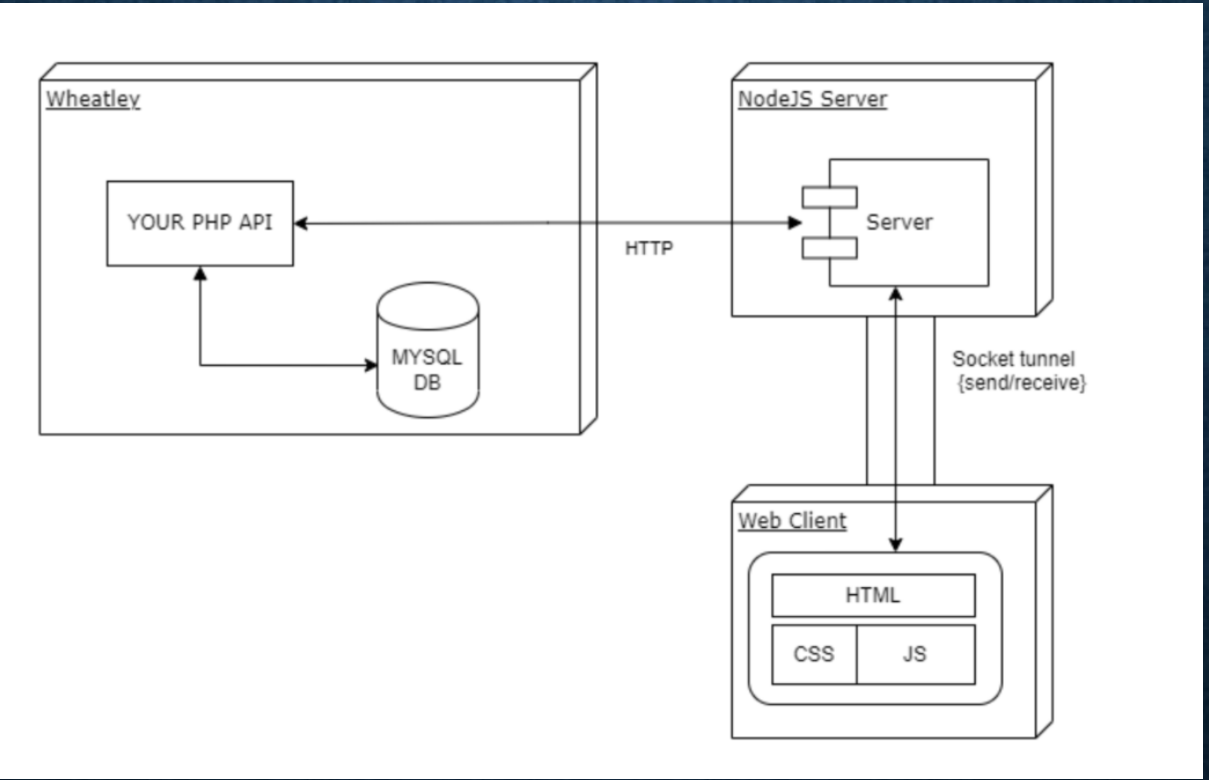
SOCKETS – SOCKET.IO

- WebSockets have been around for very long without being official adopted
  - Many JS libraries have popped up to provide socket functionality

- Socket.io has become the de facto JS library for sockets
  - Just like any other JS library, must be added to your site
  - Provides more advanced functionality compared to WebSockets
  - Actively maintained and expanded
  - Works in all browsers
  - https://socket.io

- Many other JS socket libraries exist if you want to use them

SOCKETS – HOMEWORK

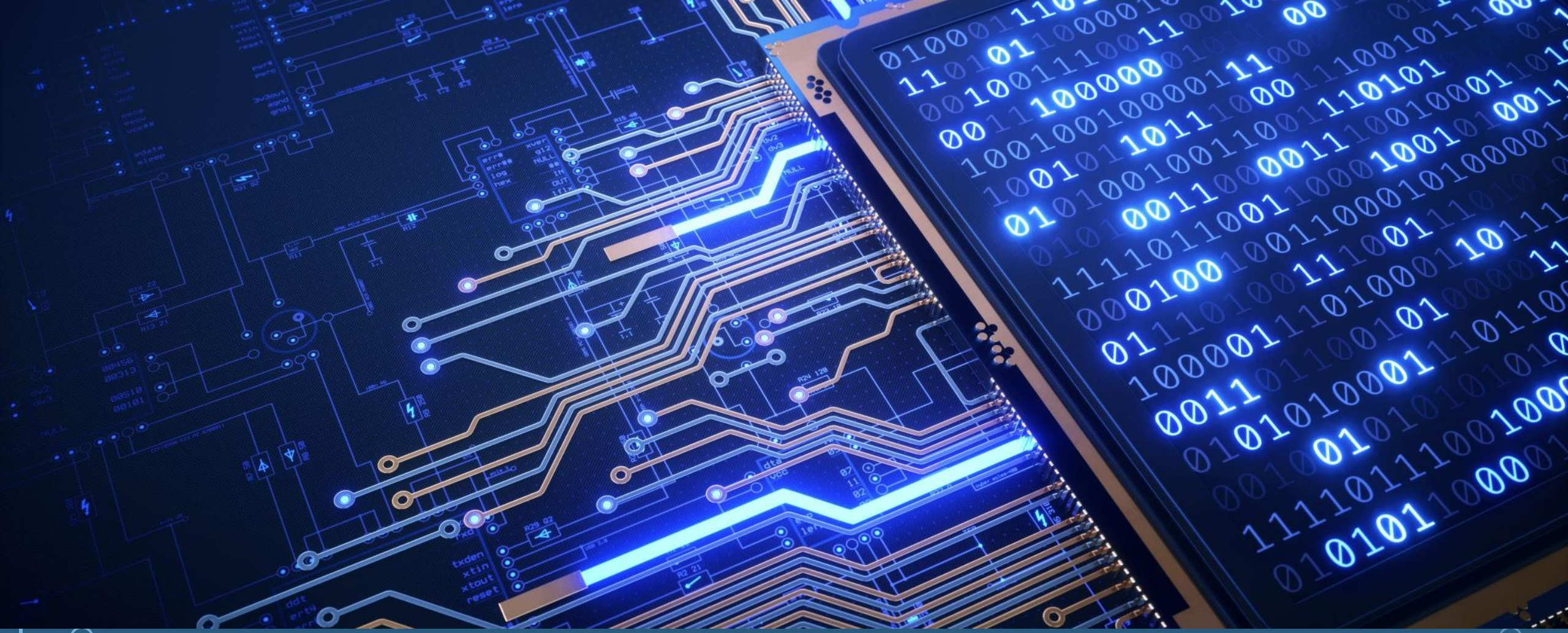


SOCKETS – HOMEWORK

- Optimally, a socket server should be implemented in PHP on Wheatley
- This is not possible due to:
  - Scripts need an infinite timeout and we cannot allow this for all students
  - Fixed ports have to be used and this might create clashes if each student opens up a port
  - Wheatley requires HTTP login which is difficult to do over sockets
- Instead:
  - Create a local socket server over which you have full control (ports, login, etc)
  - Let your local socket server use AJAX request to get data from your PHP API
    - This is a bit counter productive. You want to push data via sockets as it comes in, however you are still using a pull-based requests in the background. This is clearly not optimal, but due to the restrictions on wheatley the only way it can be done

SOCKETS – HOMEWORK

- Socket server has to be implemented in NodeJS
- Read up on that now, or wait until the NodeJS lectures



SOCKETS