Robot Networking

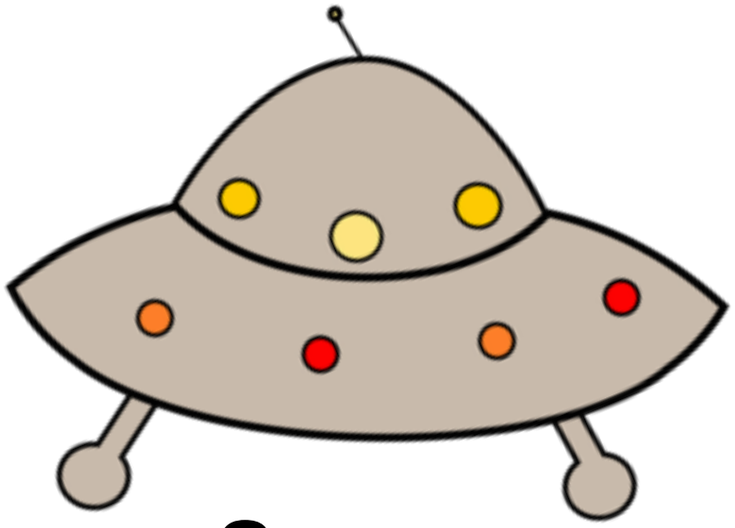
socket.io

and the

ESP8266

Network Terms
Sockets
WebSockets
Socket.io
Demos

Networking Terms



Server

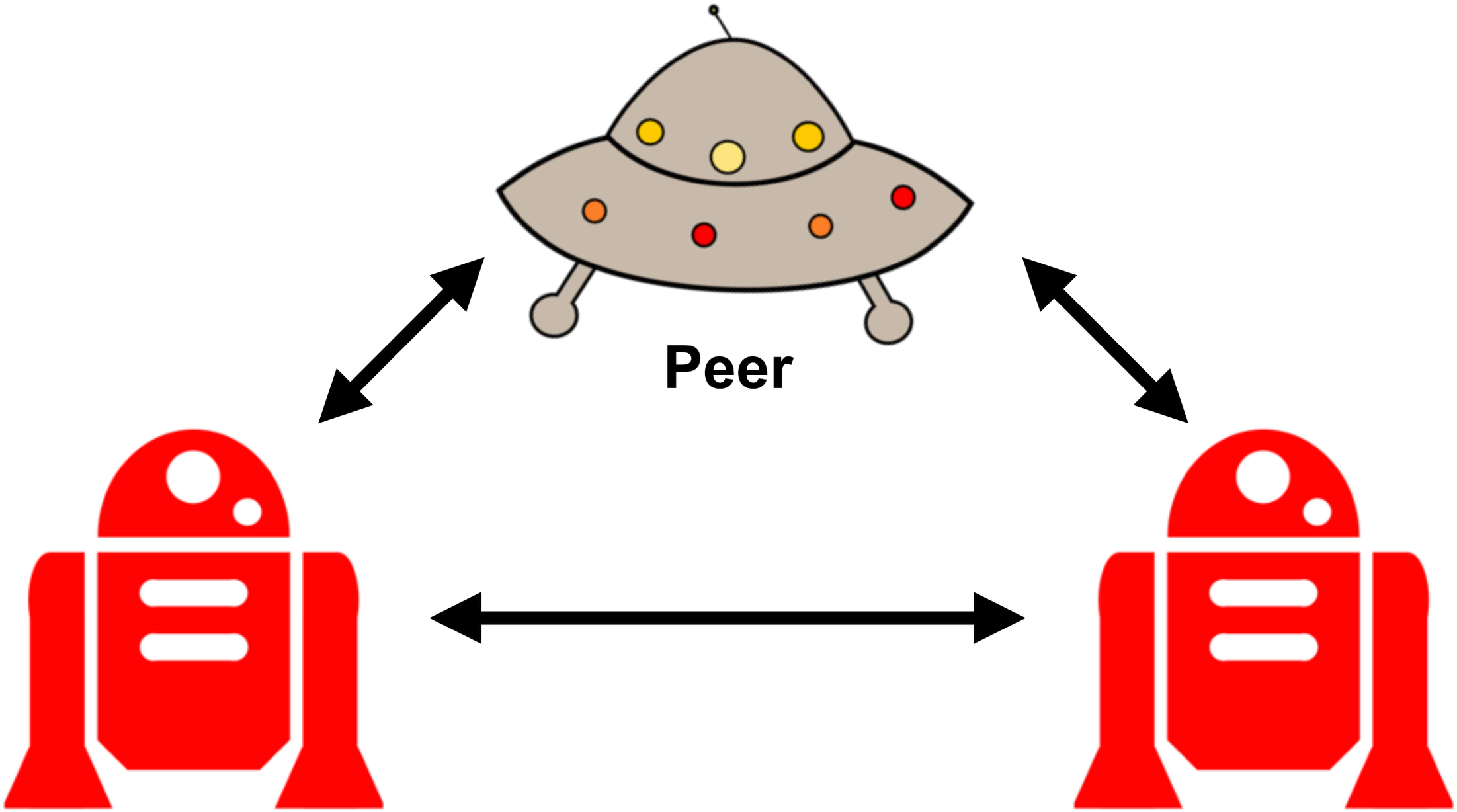
Client

Client

Client / Server Network

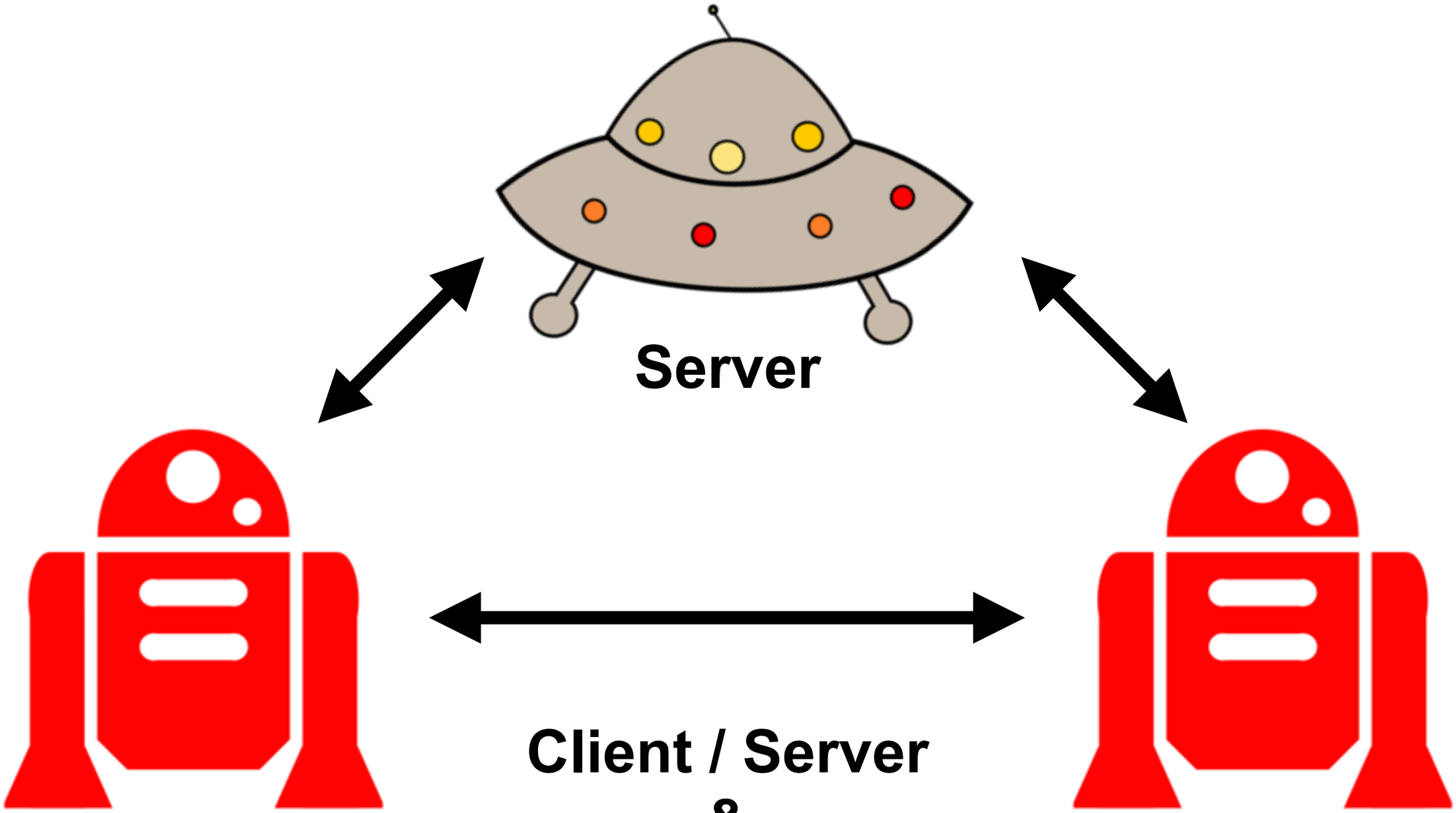
https://openclipart.org

Networking Terms



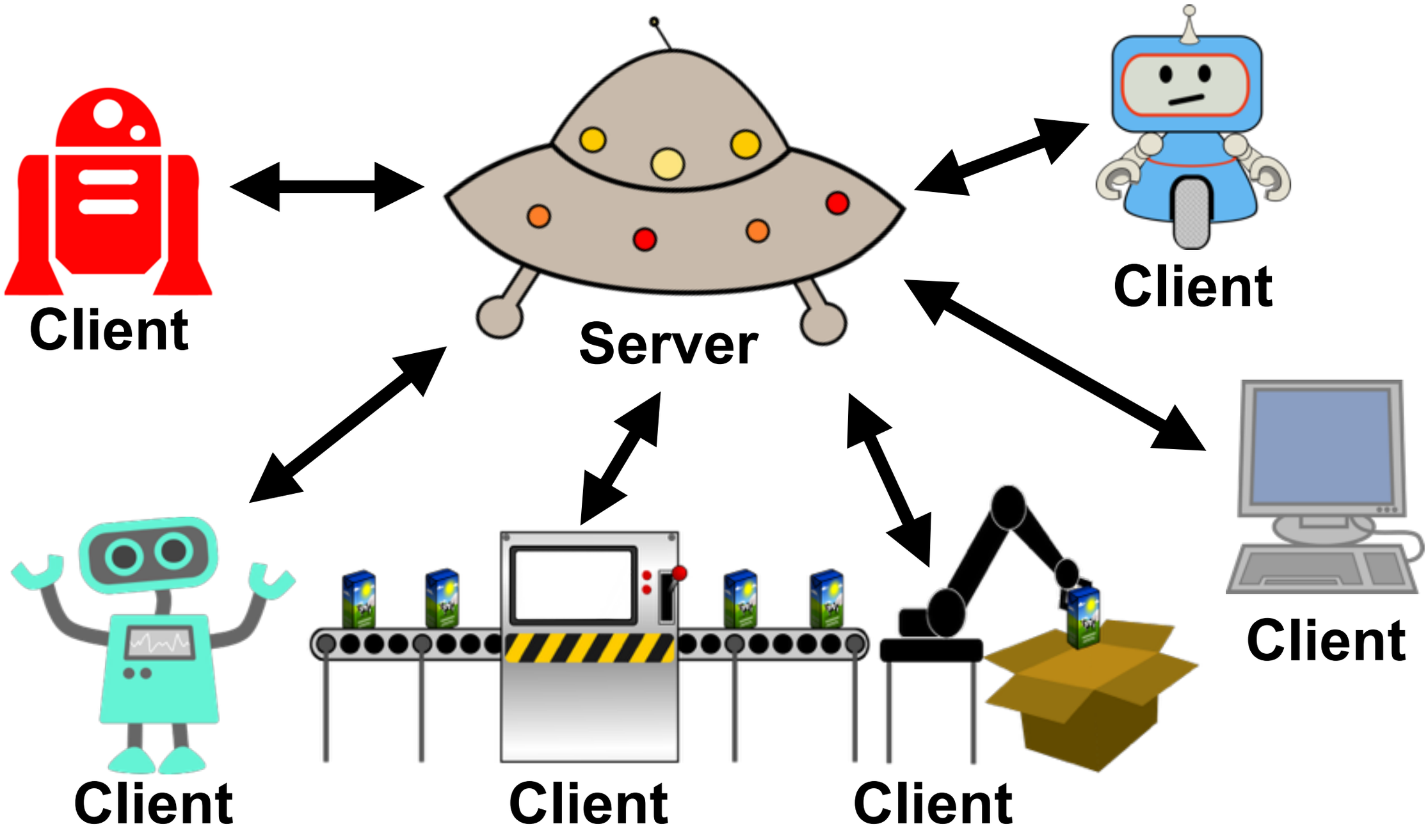
https://openclipart.org

Networking Terms



https://openclipart.org

Networking Terms

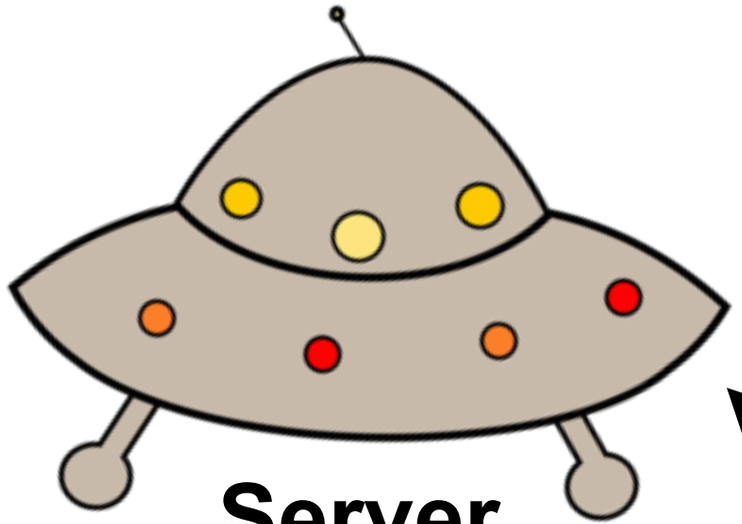


Client / Server Network

Buzzword Alert!
Internet of Things
IoT

https://openclipart.org

Today's Focus



Server

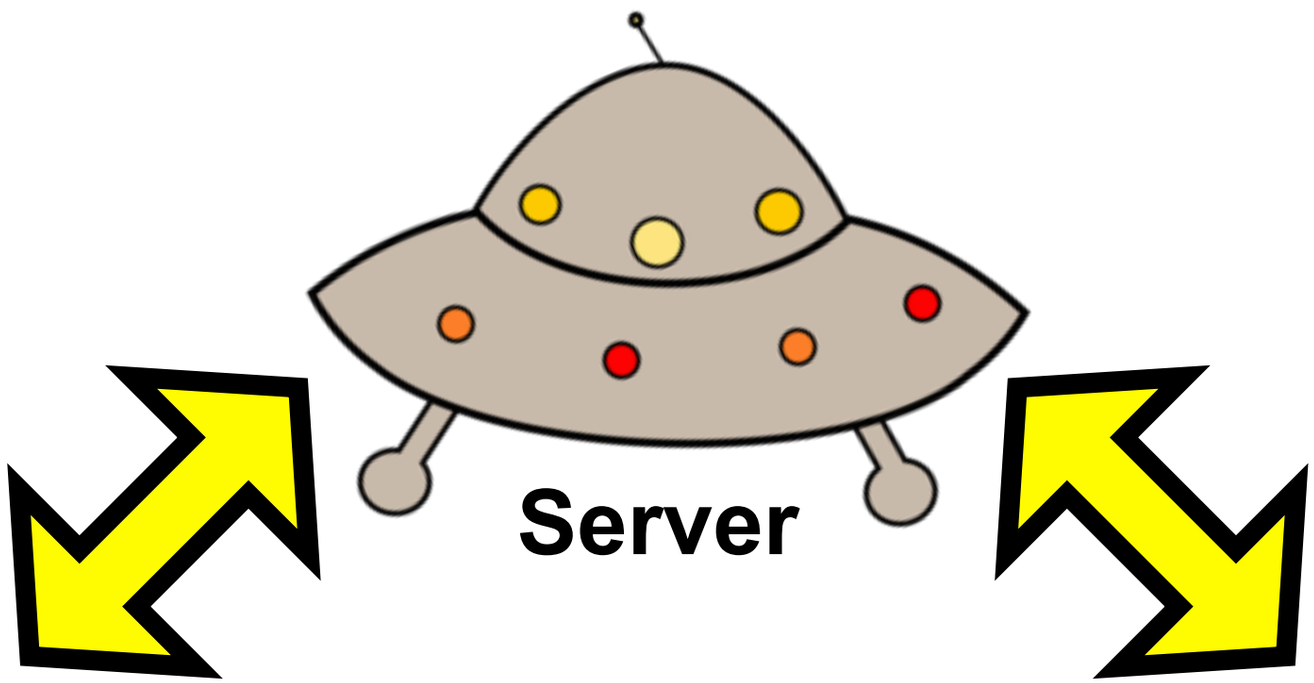
Client

Client

Client / Server Network

https://openclipart.org

Today's Focus



Client

Client

Client / Server Network

https://openclipart.org

Sockets

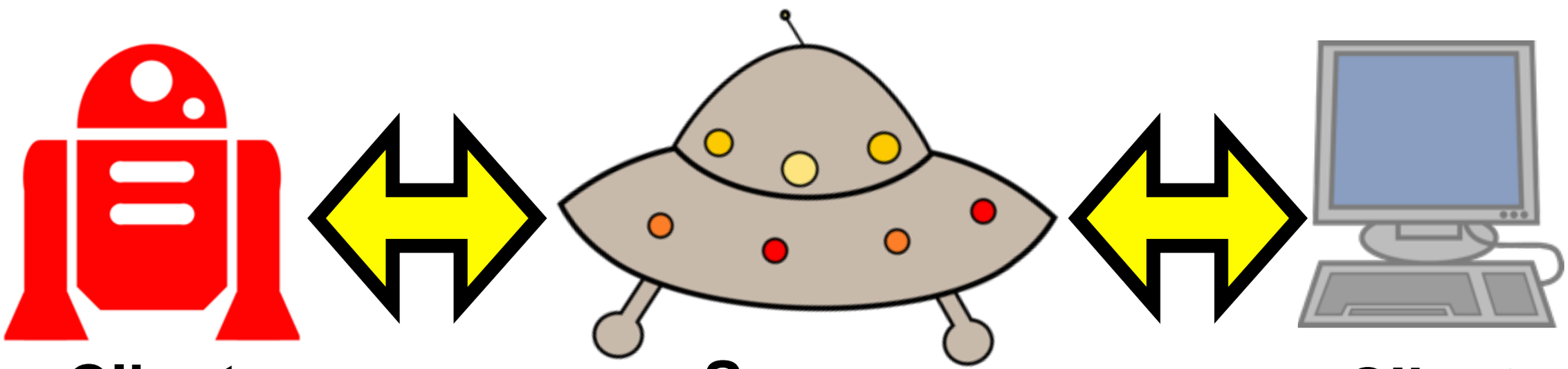
Assumption: We're doing TCP/IP networking

Each network node (Servers and Clients) has an IP **Address**.

Logical **Ports** are used to organize communications.

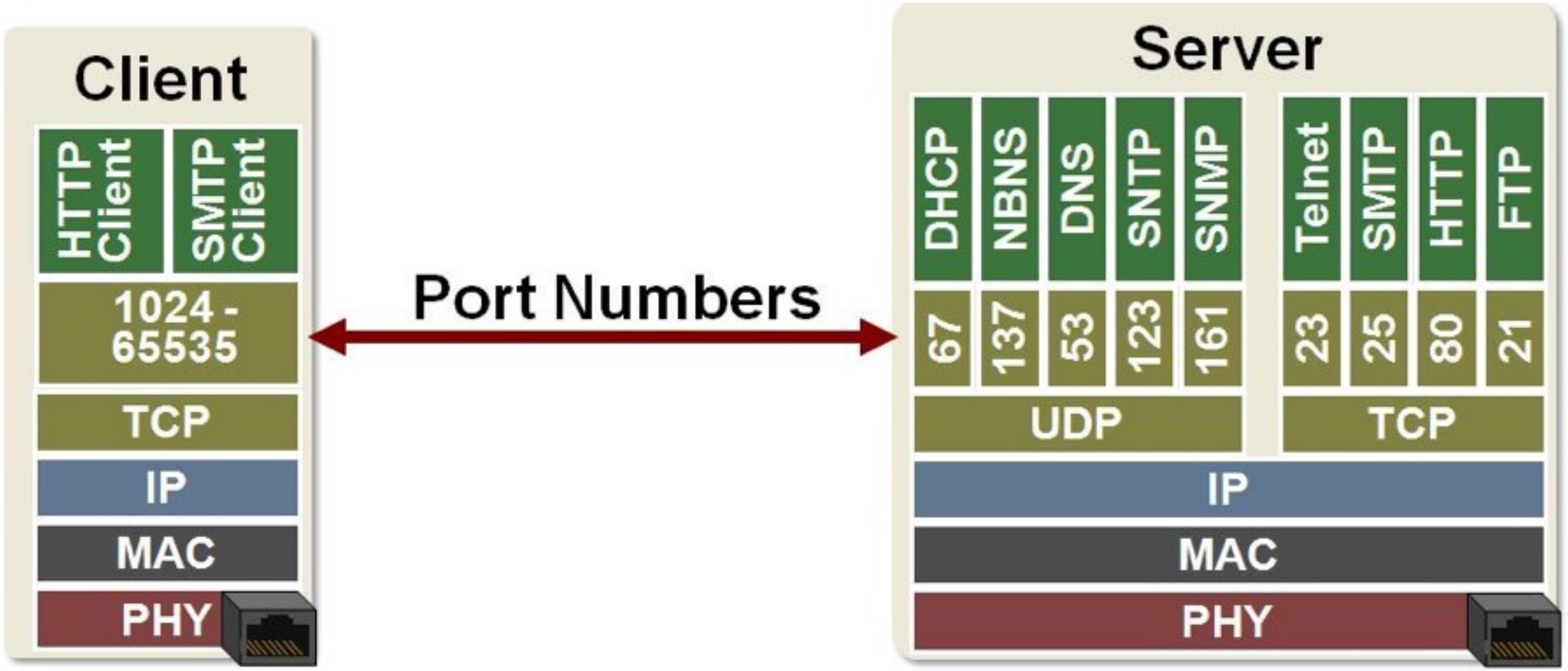
Programs running on nodes may **Listen** on specific ports.

Sockets are the logical combination of an address and port that are **Opened** by nodes (usually clients) to communicate with other nodes (usually servers).



https://openclipart.org

Sockets



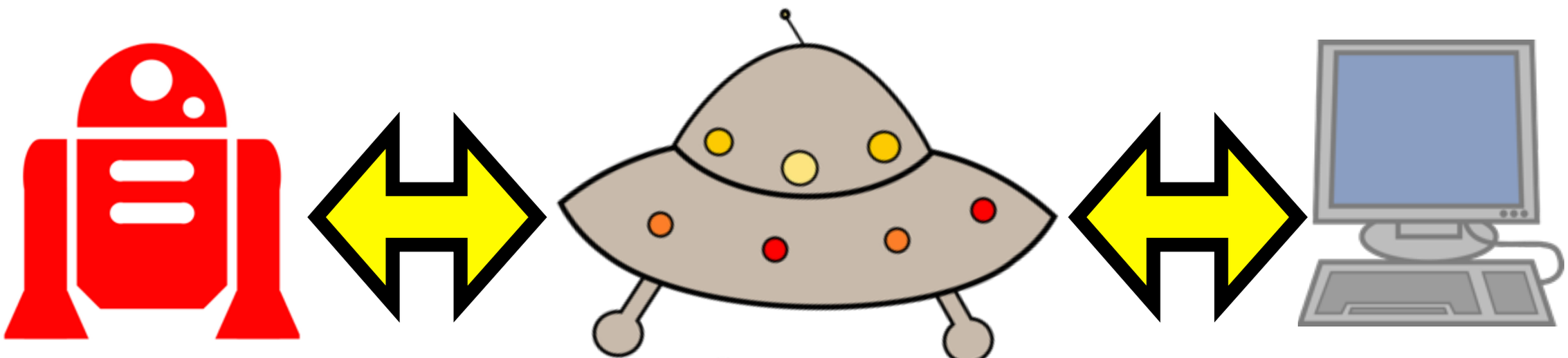
http://microchip.wikidot.com/tcpip:tcp-ip-ports

Sockets

Assumption: We're talking TCP Sockets now.

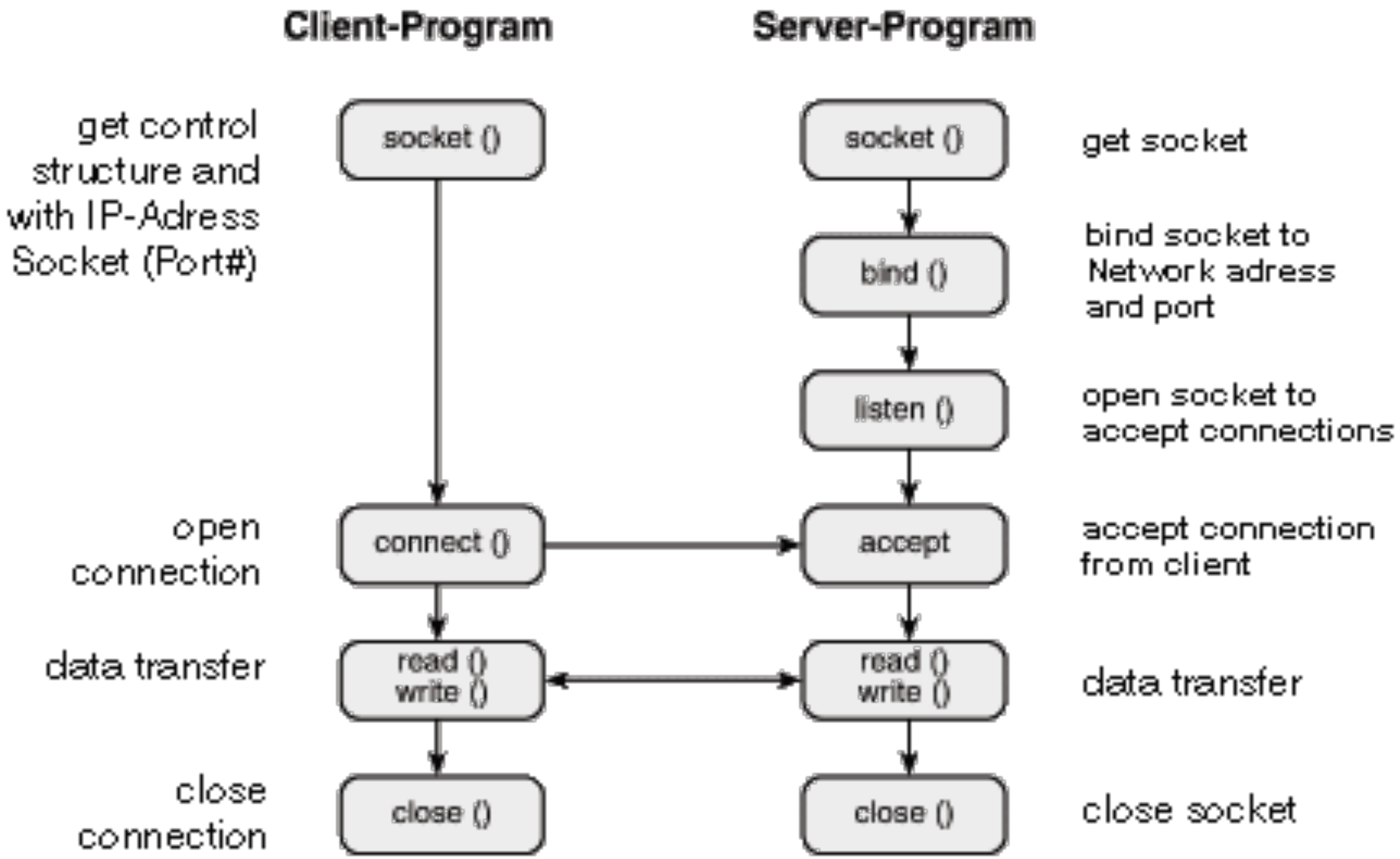
Once a Socket has been opened, bidirectional communication can take place.

With a TCP Socket, as long as an error condition doesn't occur, you can assume that the Socket is a transparent communication channel.



https://openclipart.org

Sockets



https://www.wut.de/e-58www-16-apus-000.php

Sockets

Arduino Library

https://www.arduino.cc/en/Reference/Ethernet

Arduino Example

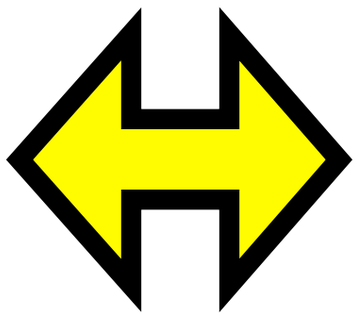
https://www.arduino.cc/en/Tutorial/ChatServer

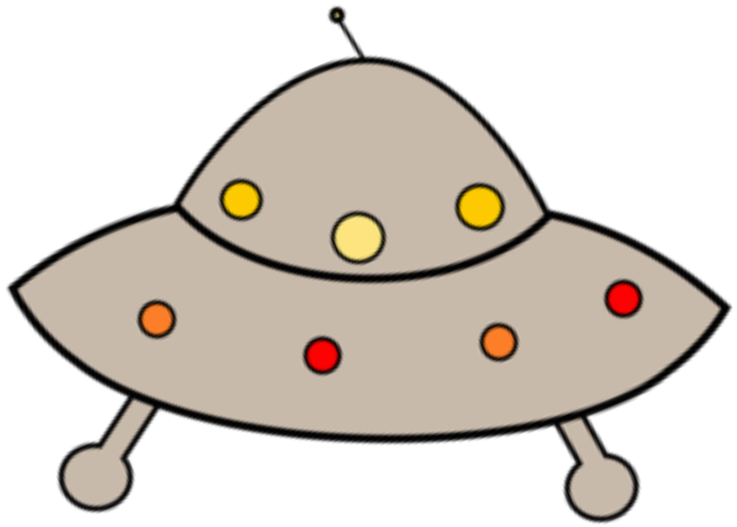
https://www.arduino.cc/en/Tutorial/ChatClient

WebSockets

Web Client

(Browser)





Web Server

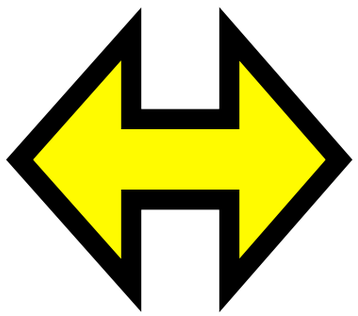
For security reasons, JavaScript running within Web Browsers can not establish TCP sockets.

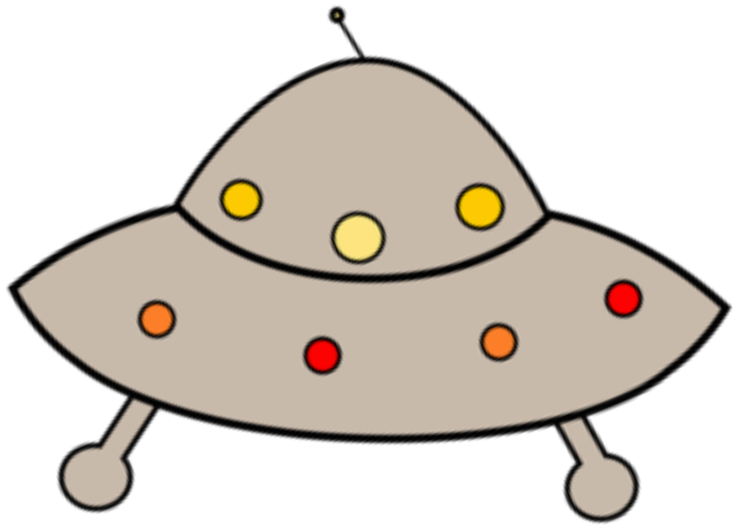
Web Browsers resorted to various hacks to receive data updates from servers (e.g. HTTP constant polling, long polling, etc.).

WebSockets

Web Client

(Browser)

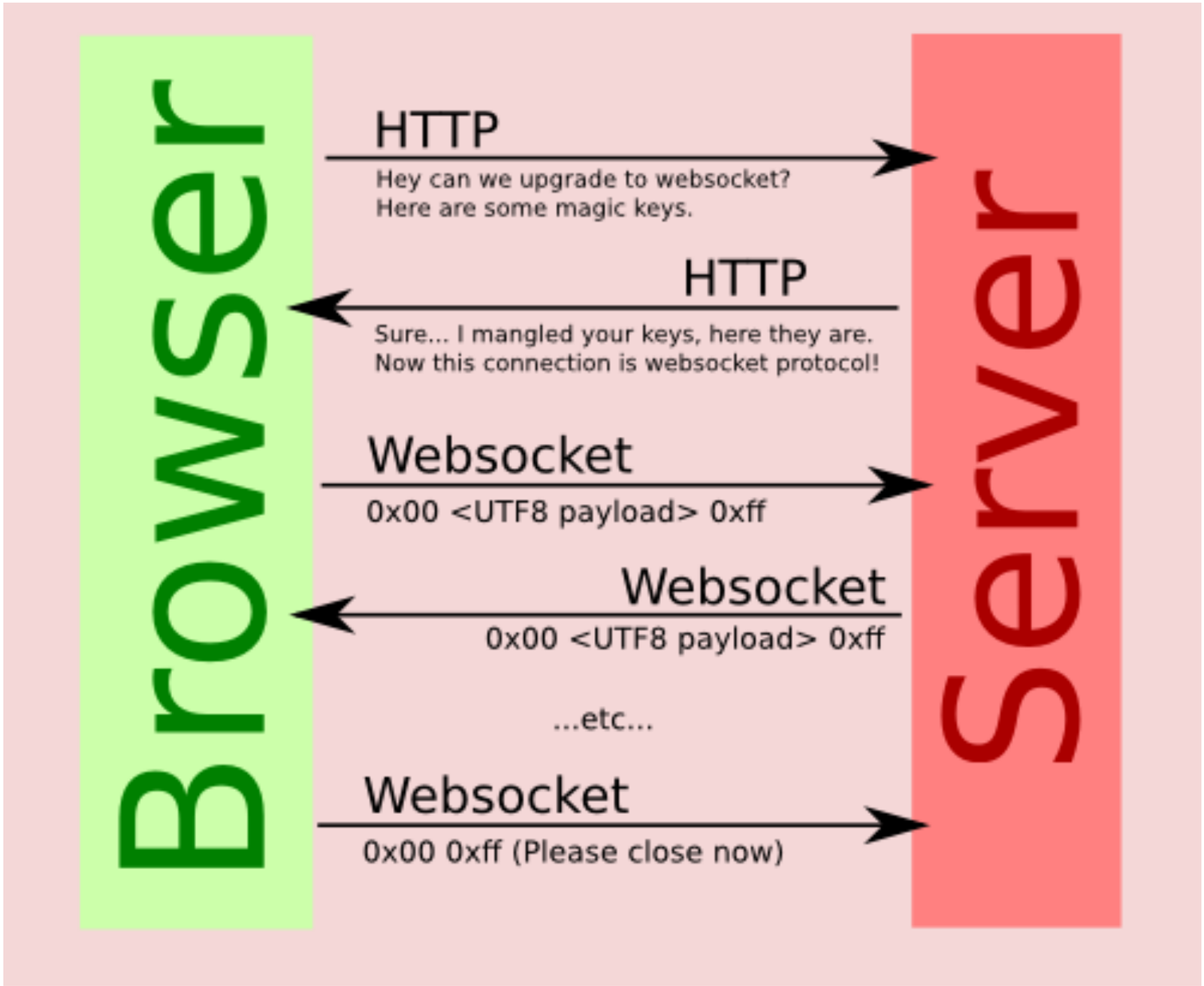




Web Server

The WebSocket protocol was created to allow efficient bidirectional communication between the client and the server while maintaining security. It can operate over any TCP Port, but typically resides on Port 80 (HTTP) or Port 443 (HTTPS) along with the Web Server.

WebSockets



http://www.cuelogic.com/blog/php-and-html5-websocket-server-and-client-communication/

WebSockets

Arduino Library

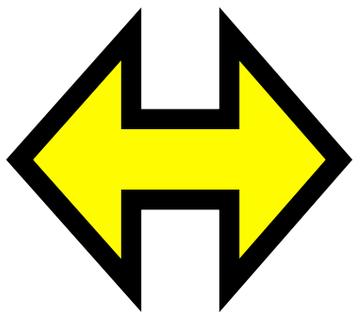
https://github.com/Links2004/arduinoWebSockets

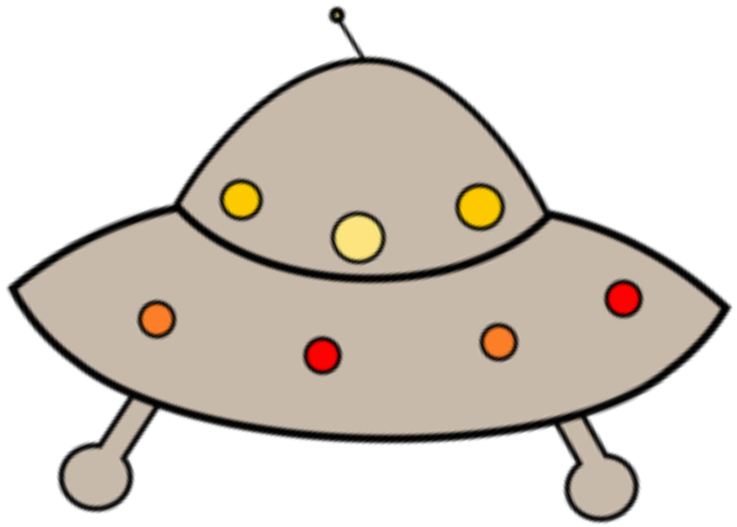
Arduino Examples

https://github.com/Links2004/arduinoWebSockets/tree/master/examples

Socket.io

Web Client (Browser)





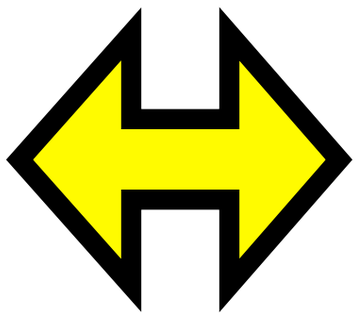
Web Server

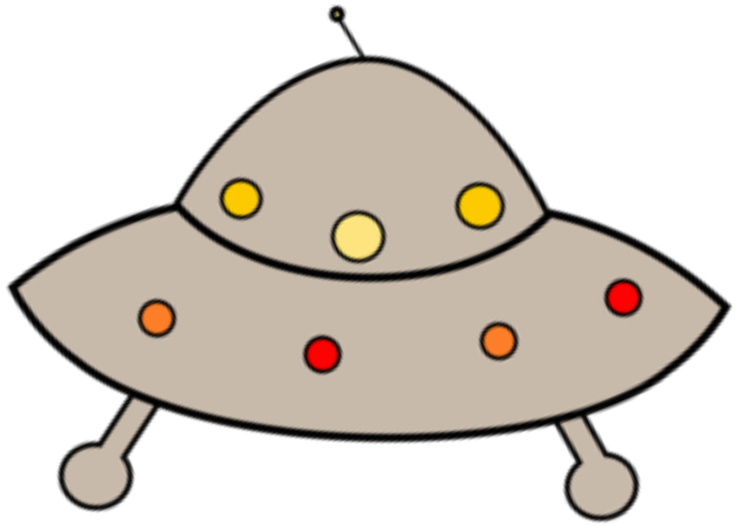
Socket.io is a protocol designed for **real-time** bidirectional **event-based** communication.
It uses WebSockets, but can fall back to other mechanisms if necessary (e.g. the Browser does not support WebSockets).

Socket.io

Web Client

(Browser)





Web Server

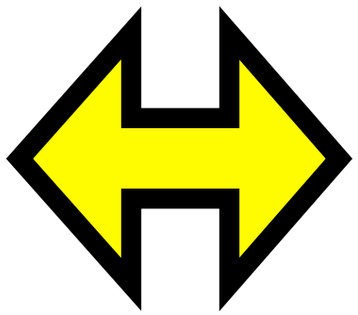
JavaScript programming interface very similar between Server and Client.

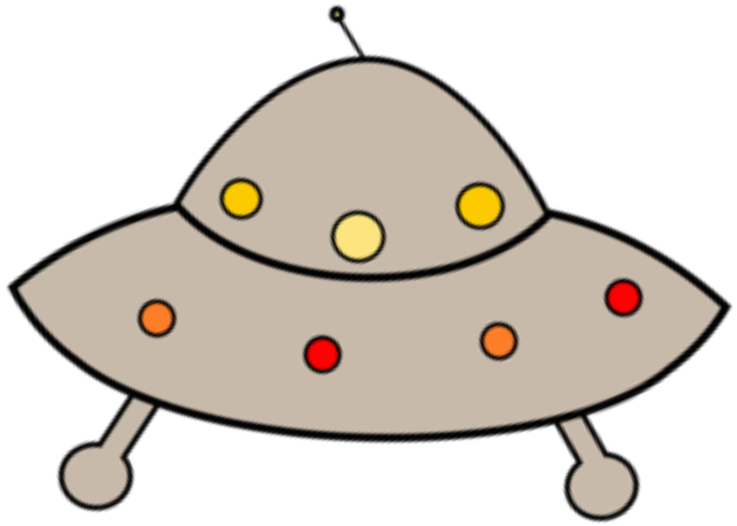
Register callback functions that are triggered by specific events (e.g. connect, disconnect, message, and custom events).

Socket.io

Web Client

(Browser)





Web Server

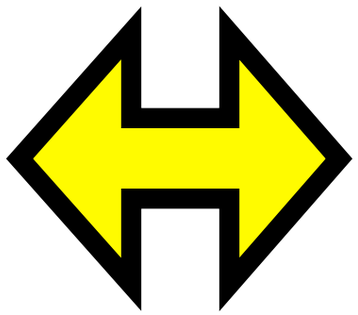
Client and Server can “send” messages or
“emit” event data

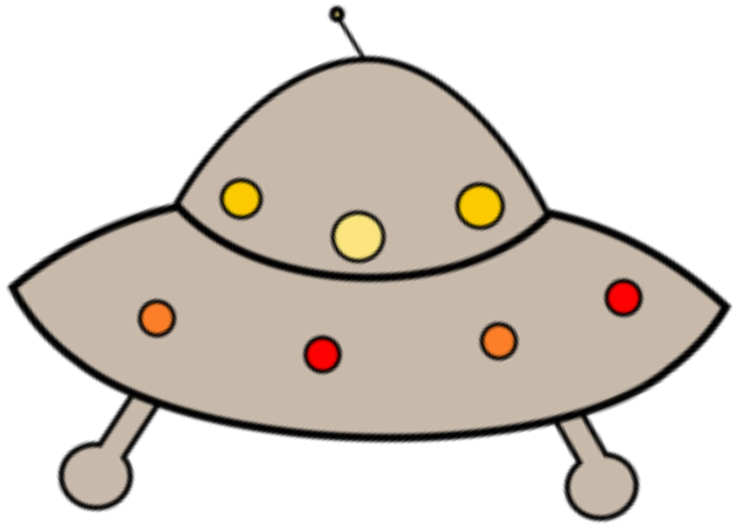
Server can also choose to broadcast messages
and events to all connected Clients

Socket.io

Web Client

(Browser)





Web Server

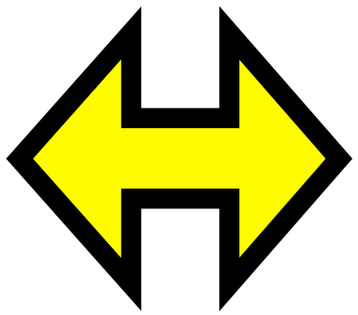
Also has the concept of “name spaces” and “rooms” to further segregate network traffic.

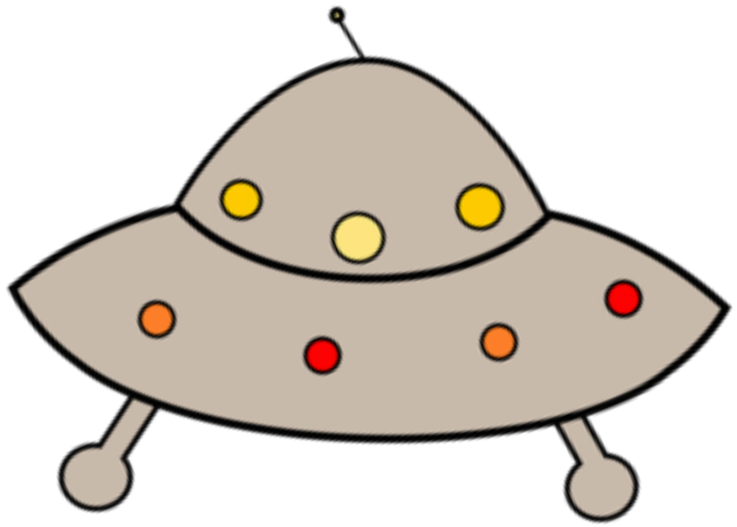
The classic example for using these features is a chat application with multiple “rooms” for conversations and “name spaces” for administrative versus chat data.

Socket.io

Web Client

(Browser)

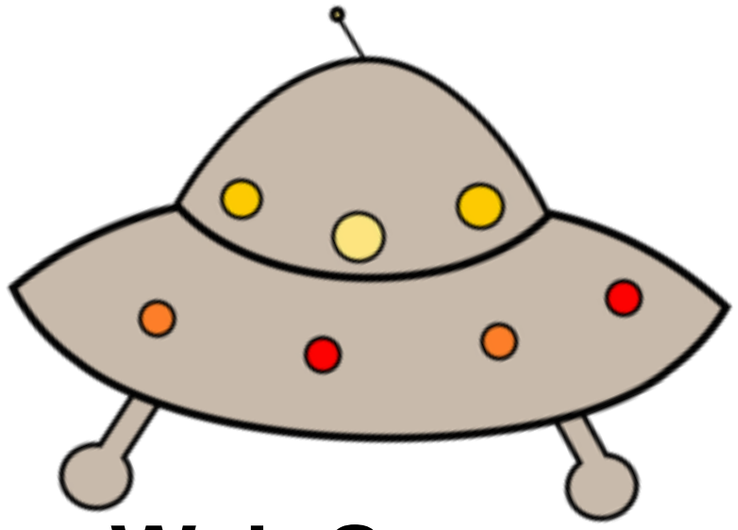




Web Server

We will first go over a simple Web Server / Browser multi-way light switch application, and then add an ESP8266 into the mix.

Socket.io



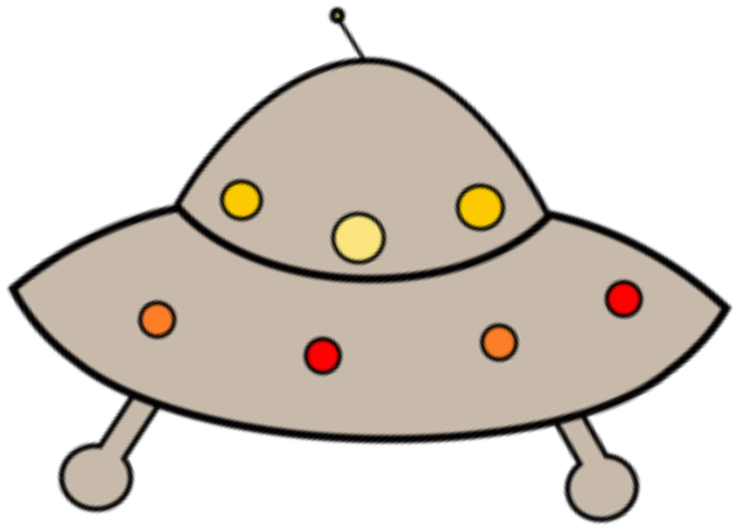
Web Server
Node.js
JavaScript

Web Client
Browser
HTML + JavaScript

Web Client
Browser
HTML + JavaScript

https://openclipart.org

Socket.io



Wait for a connection
“emit” the light state to the Client
Set up callback functions for:
“disconnect” and “toggle”

When a “toggle” event occurs
changes the light state
“broadcast emit” the new state to all Clients

Socket.io

Create the text for the light state
Create the toggle button
Initialize socket and callback functions for:
button “click” and “light” event

When the toggle button is clicked,
“emit” a “toggle” event

When a “light” event occurs,
update the light state text

https://openclipart.org

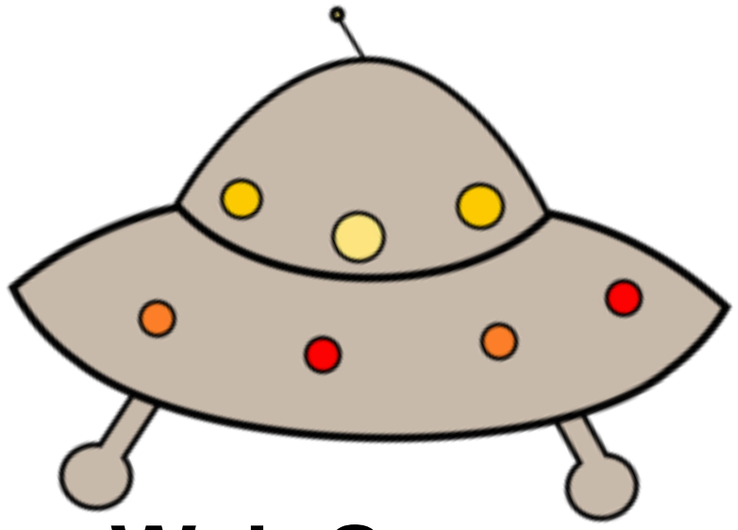
Socket.io - Server

```
var app = require('express')();
var http = require('http').Server(app);
var io = require('socket.io')(http);
var light = {state:false};

app.get('/', function(req, res) {
  res.sendFile(__dirname + '/index.html');
});

io.on('connection', function(socket) {
  console.log('User connected: ' + socket.id);
  socket.emit('light', light);
  socket.on('disconnect', function(){
    console.log('User disconnected: ' + socket.id);
  });
  socket.on('toggle', function(state) {
    light.state = !light.state;
    console.log('id: ' + socket.id + ' light: ' + light.state);
    io.sockets.emit('light', light);
  });
});

http.listen(3000, function() {
  console.log('listening on *:3000');
});
```

Socket.io - Client

```
<!doctype html>
<html>
  <head>
    <title>Socket.io Simple Example</title>
  </head>
  <body>
    <div id="light">Unknown</div>
    <button id="toggleButton">Toggle</button>
    <script src="/socket.io/socket.io.js"></script>
    <script src="http://code.jquery.com/jquery-1.11.1.js"></script>
    <script>
      var socket = io();
      $('#toggleButton').on('click', function() {
        socket.emit('toggle', {state:true});
      });
      socket.on('light', function(light) {
        console.log(light);
        if  (light.state) {
         $('#light').text('ON');
        }
        else {
         $('#light').text('off');
        }
       });
    </script>
  </body>
</html>
```

Socket.io - Demo



Web Server
Node.js
JavaScript

Web Client
Browser
HTML + JavaScript

Web Client
Browser
HTML + JavaScript

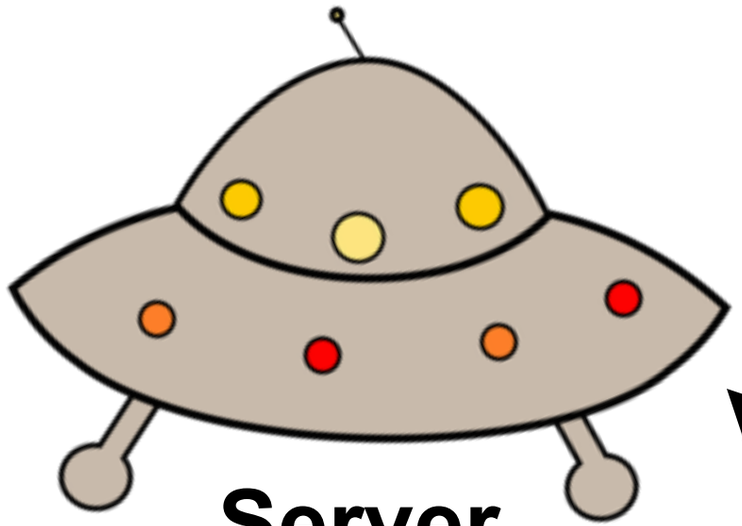
https://openclipart.org

Socket.io - Demo

Demo Source

https://github.com/robojay/simple-socket-io-example

Socket.io



Server
Node.js

Client
ESP8266
Arduino C++

Client
Browser
JavaScript

https://openclipart.org

Socket.io

WARNING!!!

This is Bleeding Edge…

https://openclipart.org

Socket.io

Arduino Library

https://github.com/robojay/Socket.io-v1.x-Library

Arduino Socket.io library was forked and
then modified by Jay and Paul.

This is ***DEFINITELY*** a work in progress and
certainly has bugs and “features”.

Socket.io - ESP8266

```
#include <SocketIOClient.h>

#define LedPin 2
#define ButtonPin 0

#define SOFTAP_MODE

#ifdef SOFTAP_MODE
const char* password = "myMinion";
#else
const char* ssid = "SkyNet";
const char* password = "myMaster";
#endif

const char HexLookup[17] = "0123456789ABCDEF";

String host = "192.168.4.2";
int port = 3000;
bool clicked = false;

SocketIOClient socket;
```

Socket.io - ESP8266

```
//
// This code runs only once
//
void setup() {

  // set up our pins
  pinMode(LedPin, OUTPUT);
  pinMode(ButtonPin, INPUT);

  digitalWrite(LedPin, LOW);

  Serial.begin(115200);

  setupNetwork();

  attachInterrupt(digitalPinToInterrupt(ButtonPin), click, FALLING);

  socket.on("light", light);

  socket.connect(host, port);
}
```

Socket.io - ESP8266

```
void setupNetwork() {

  #ifdef SOFTAP_MODE
    WiFi.disconnect();
    byte mac[6];
    WiFi.macAddress(mac);
    char ssid[14] = "Minion-000000";
    ssid[7] = HexLookup[(mac[3] & 0xf0) >> 4];
    ssid[8] = HexLookup[(mac[3] & 0x0f)];
    ssid[9] = HexLookup[(mac[4] & 0xf0) >> 4];
    ssid[10] = HexLookup[(mac[4] & 0x0f)];
    ssid[11] = HexLookup[(mac[5] & 0xf0) >> 4];
    ssid[12] = HexLookup[(mac[5] & 0x0f)];
    ssid[13] = 0;
    WiFi.softAP(ssid, password);
  #else
    WiFi.begin(ssid, password);
    uint8_t i = 0;
    while (WiFi.status() != WL_CONNECTED && i++ < 20) delay(500);
    if(i == 21){
      while(1) delay(500);
    }
  #endif

}
```

Socket.io - ESP8266

```
void click() {
  clicked = true;
}

void clickCheck() {
  if (clicked) {
    Serial.println("[click]");
    socket.emit("toggle", "{\"state\":true}");
    clicked = false;
  }
}

void light(String state) {
  Serial.println("[light] " + state);
  if (state == "\"state\":true") {
    Serial.println("[light] ON");
    digitalWrite(LedPin, HIGH);
  }
  else {
    Serial.println("[light] off");
    digitalWrite(LedPin, LOW);
  }
}
```

Socket.io - ESP8266

```
//
// This code runs over and over again
//
void loop() {
  socket.monitor();
  clickCheck();
}
```

Socket.io - ESP8266

ESP8266 Source

https://github.com/robojay/simple-socket-io-example-esp8266