Socket.io

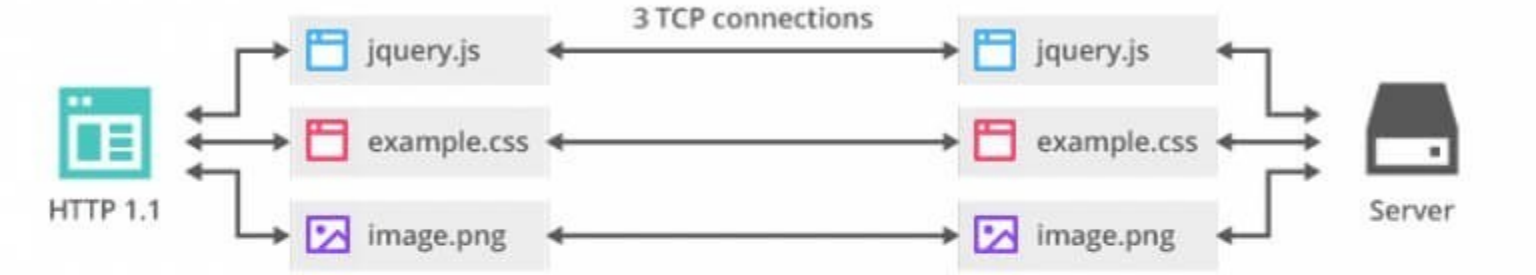
Prof.: Álvaro Pagliari

Sockets

- Sockets são uma forma de comunicação que permite que dois processos diferentes possam se conversar e trocar informações entre si via uma rede
- O Sistema Operacional é o responsável por abrir e manter um socket em operação
- Os programas usam uma API do Sistema Operacional para manipular um socket
- Ou seja, um socket é um canal de comunicação entre dois processos através de uma rede de computadores

Modelo de Conexão HTTP/1.x

- Cada requisição feita para o servidor inicia um socket novo
- No HTTP 0.9 e 1.0, após a resposta do servidor o socket é finalizado
- No HTTP 1.1 usando o cabeçalho Connection: “Keep-Alive”, pode se reutilizar o socket TCP por um número limitado de vezes
- Outras técnicas usadas foram Pipeling e Sharding



UNOESC

Modelo de Conexão HTTP/2

- O HTTP/2 é um protocolo binário, ao contrário do HTTP 1.x que é textual
- Por padrão é obrigatório a compressão gzip
- Não necessita de sharding ou pipelining porque é feito multiplexação de dados em um socket TCP
- Evita o problema de Head of Line Blocking (HOLB) a nível do protocolo, porém não a nível do TCP

UNOESC

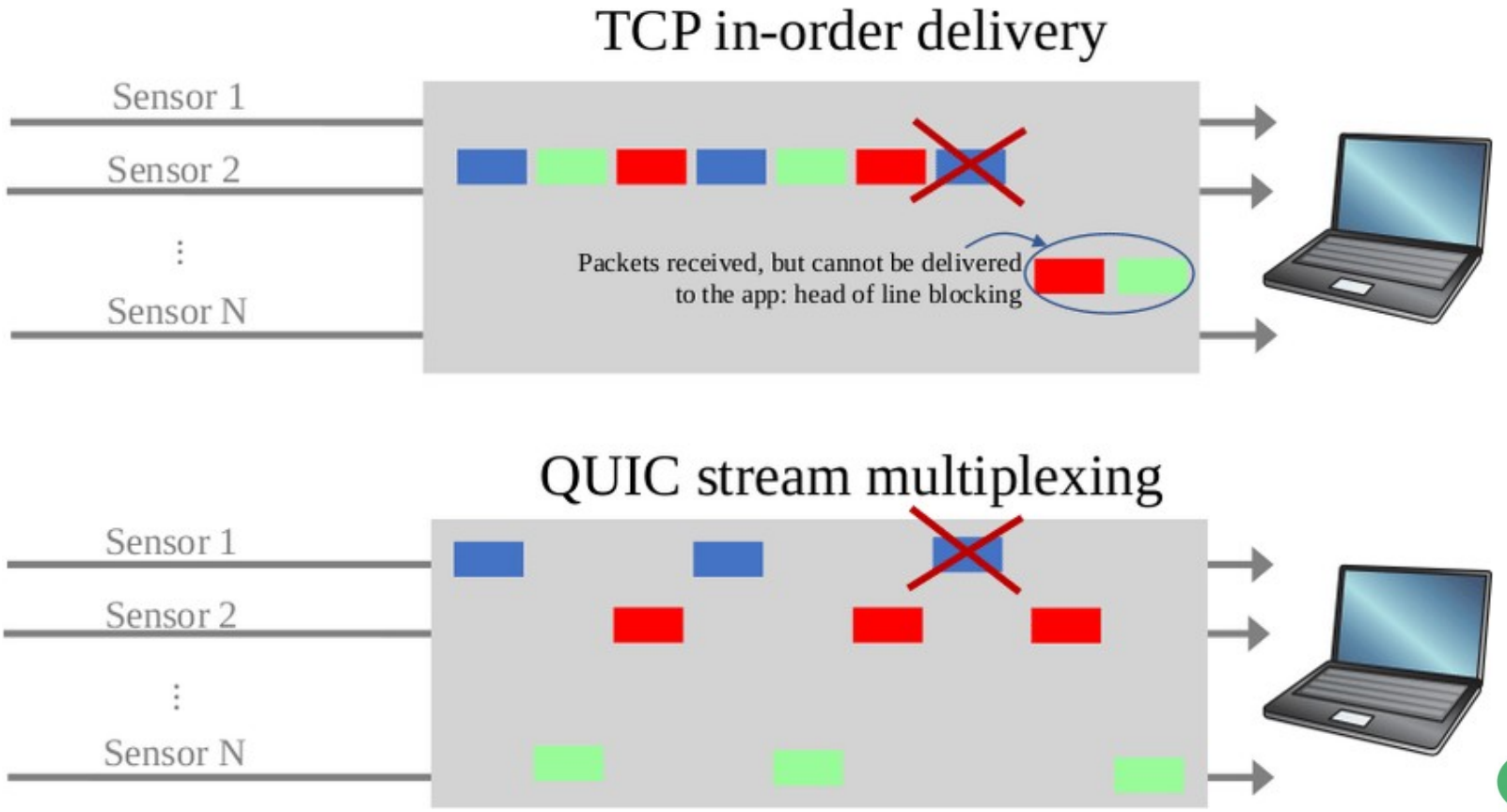
Modelo de Conexão HTTP/3

- O HTTP/3 é a última versão do protocolo HTTP
- Tem todos os benefícios do HTTP/2, como ser binário, suportar multiplexing e gzip
- Resolveu o problema de Head of Line Block do HTTP/2 em nível TCP
- Utiliza um novo protocolo chamado QUIC (Quick UDP Internet Connections) executado em cima do UDP
- Suporta somente conexões criptografadas

UNOESC

Modelo de Conexão HTTP/3

Head of Line Blocking



UNOESC

HTTP + Socket = WebSocket

- O WebSocket é um protocolo de comunicação bidirecional que executa sobre o TCP com um handshake feito via HTTP upgrade
- O padrão WebSocket veio para resolver o problema de comunicação unidirecional do HTTP
- Os métodos utilizados antes do WebSocket eram conhecidos como Comet, que incluiam técnicas de long polling via Hidden iFrame, XMLHttpRequest, etc

Socket.IO

- Socket.IO é uma biblioteca JS de comunicação de baixa latência, bidirecional e baseada em eventos, entre um cliente e um servidor
- As principais funcionalidades do Socket.IO são:
  - HTTP long-polling fallback
  - Reconexão automática
  - Buffer de pacotes (somente no cliente)
  - Acknowledgements
  - Broadcasting
  - Multiplexing
- Atenção: o Socket.IO não é uma implementação do WebSocket, ou seja, não é possível conectar via outros clientes que não o do Socket.IO

UNOESC

Socket.IO – Server - Conexão

```
const express = require('express');
const app = express();
const http = require('http');
const server = http.createServer(app);
const { Server } = require("socket.io");
const io = new Server(server);

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

io.on('connection', (socket) => {
  console.log('a user connected');
});

server.listen(3000, () => {
  console.log('listening on *:3000');
});
```

UNOESC

Socket.IO – Server - Eventos

“Ouvindo eventos”

```
io.on("connection", (socket) => {
  socket.on("hello", (arg) => {
    console.log(arg); // world
  });
});
```

Emitindo eventos

```
io.on("connection", (socket) => {
  socket.emit("hello", 1, "2", { 3: '4', 5: Buffer.from([6]) });
});
```

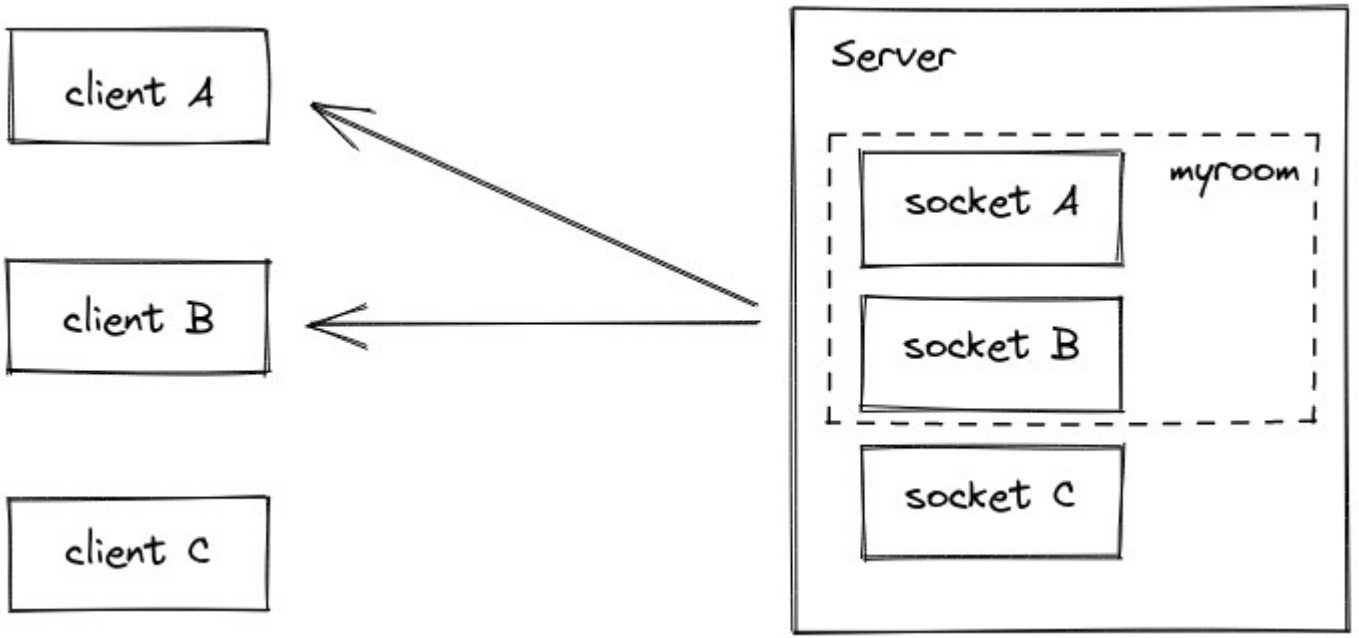
Acknowledgement de um evento através de callback

```
io.on("connection", (socket) => {
  socket.on("update item", (arg1, arg2, callback) => {
    console.log(arg1); // 1
    console.log(arg2); // { name: "updated" }
    callback({
      status: "ok"
    });
  });
});
```

UNOESC

Socket.IO – Server - Rooms

- É um canal arbitrário que o socket pode entrar (join) ou sair (leave)



Socket.IO – Server - Rooms

Entrando em uma room

```
io.on("connection", (socket) => {
  socket.join("some room");
});
```

Enviando uma mensagem para todos no room

```
io.to("some room").emit("some event");
```

Saindo de uma sala

```
socket.on('unsubscribe',function(room){
  try{
    console.log('[socket]','leave room :', room);
    socket.leave(room);
    socket.to(room).emit('user left', socket.id);
  }catch(e){
    console.log('[error]','leave room :', e);
    socket.emit('error','couldnt perform requested action');
  }
})
```

UNOESC

Socket.IO – Client

Carregar a biblioteca na página

```
<script src="/socket.io/socket.io.js"></script>
```

Abrir a conexão com o servidor

```
var socket = io();

socket.emit('joining msg', name);
```

Recebendo uma mensagem do servidor

```
socket.on('chat message', function(msg){
        $('#messages').append($('<li>').text(msg));
});
```

Enviando uma mensagem para o servidor

```
socket.emit('chat message', (name + ':  ' + $('#m').val()));
```

UNOESC

Socket.IO – Exemplo

Vamos criar um exemplo de chat!

Socket.IO – Exercícios

1. Adapte o chat do exemplo anterior para que tenha várias salas
2. Permita que o cliente troque de sala
3. Crie um botão de “BUZZ” que trema a tela de todos os clientes que estão em uma sala

UNOESC