WebSockets

Wallace da Silva Ribeiro

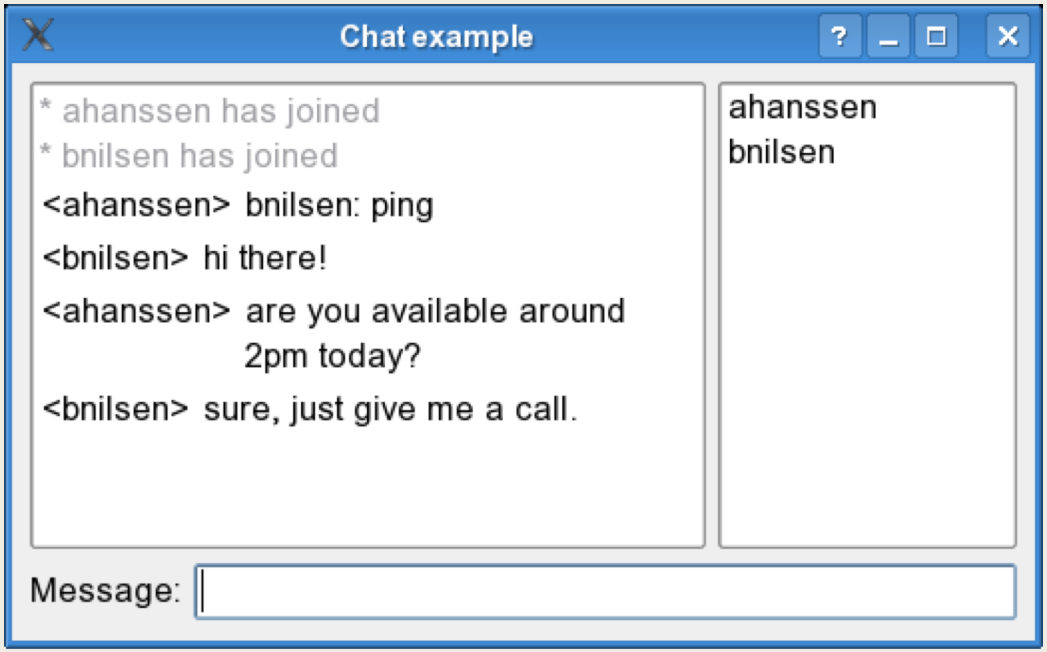
Comunicação servidor - cliente

- Em determinadas situações o servidor necessita enviar informações ao navegador Web de forma assíncrona.

- Desta maneira, não apenas o cliente é ativo, mas também é passivo, no sentido de ser atualizado por eventos externos.

Exemplo - Chat

- Mensagens podem ser enviadas de um usuário para todos os outros usuários do Chat.
- Mensagens novas são recebidas (em tempo real) por todos os usuários.



Dúvida

- Neste exemplo o servidor deve ser ativo e avisar todos os participantes quando uma nova mensagem surgir.

- De que maneira o servidor pode “encontrar” os clientes e encaminhar mensagens?

Soluções

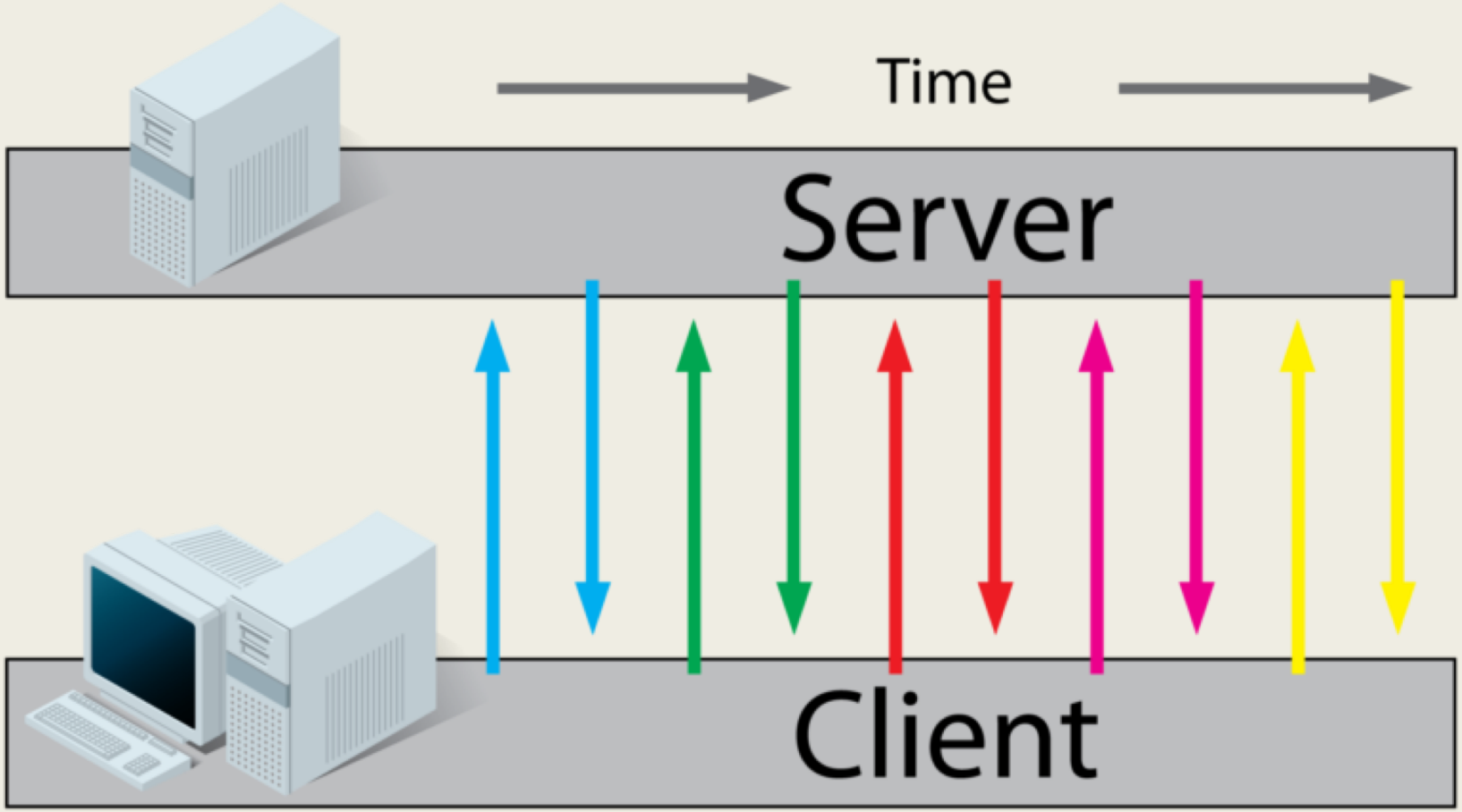
- Polling

- WebSocket

Polling

- Consiste em manter chamadas (notadamente http) constantes, de modo que o cliente se mantenha atualizado sobre qualquer tipo de intenção do servidor em enviar qualquer notificação ao mesmo.

Polling



Vantagens

- Escalabilidade: *Stateless*.
- Fácil implementação do lado do servidor. Simplesmente mais um endpoint do lado do servidor.
- Fácil implementação do lado do cliente. Um loop para formar o polling cycle.

Desvantagens

- Mantem uma sobrecarga de processamento na página cliente. A realização de uma chamada http pode ser bastante custosa tendo que ser realizada a cada fração de tempo.

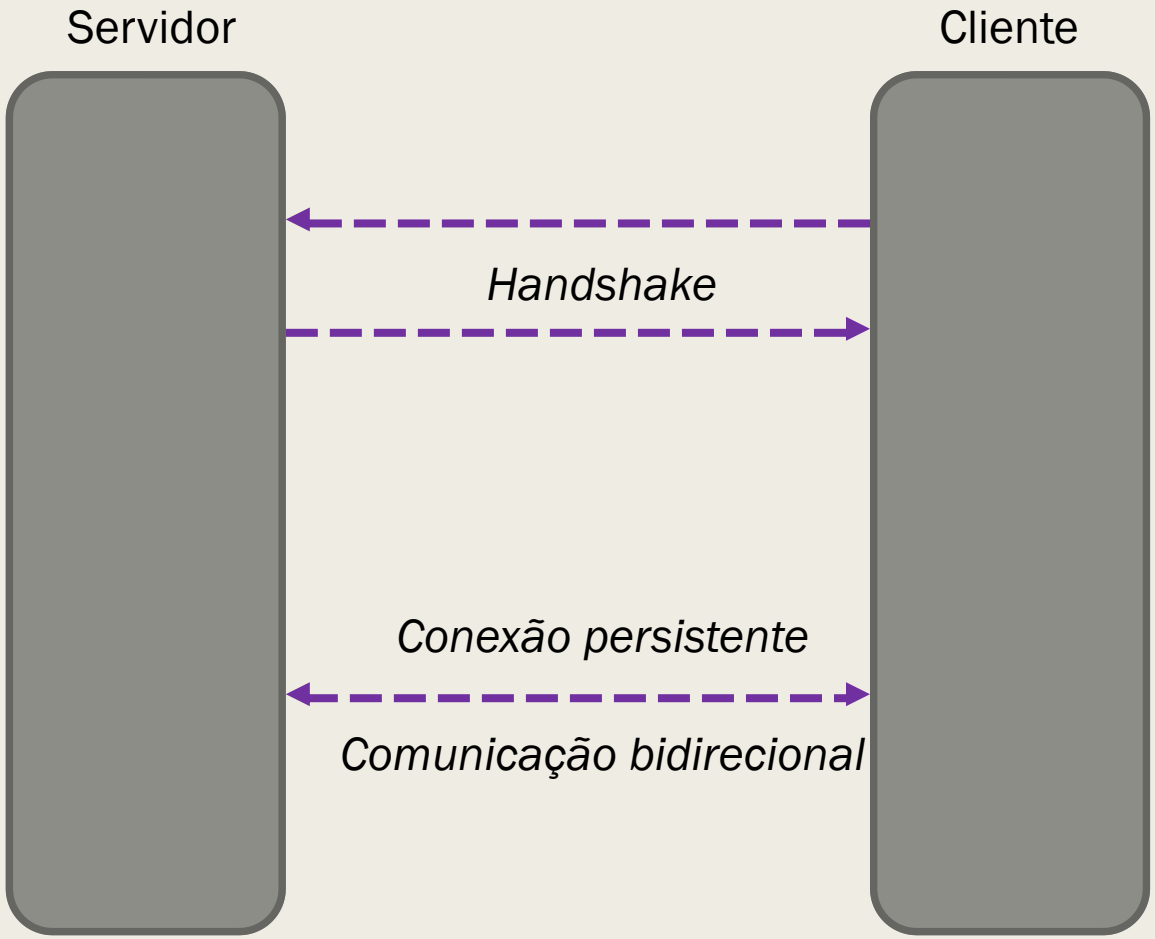
- Sobrecarga da rede. Toda a pilha do protocolo http é informação suficientemente grande para sobrecarregar a rede.

WebSocket

- Mantém uma conexão de socket aberta com o servidor, com um payoad de controle mínimo.

- Desta maneira mantém um canal de comunicação aberto para que o servidor possa notificar o cliente.

WebSocket


Servidor
Cliente
Handshake
Conexão persistente
Comunicação bidirecional

Vantagens

- Pouco processamento no lado cliente. Relativamente leve a manipulação de sockets.
- Pouca sobrecarga de rede.
- Comunicação bidirecional é mais intuitiva para várias soluções.

Desvantagens

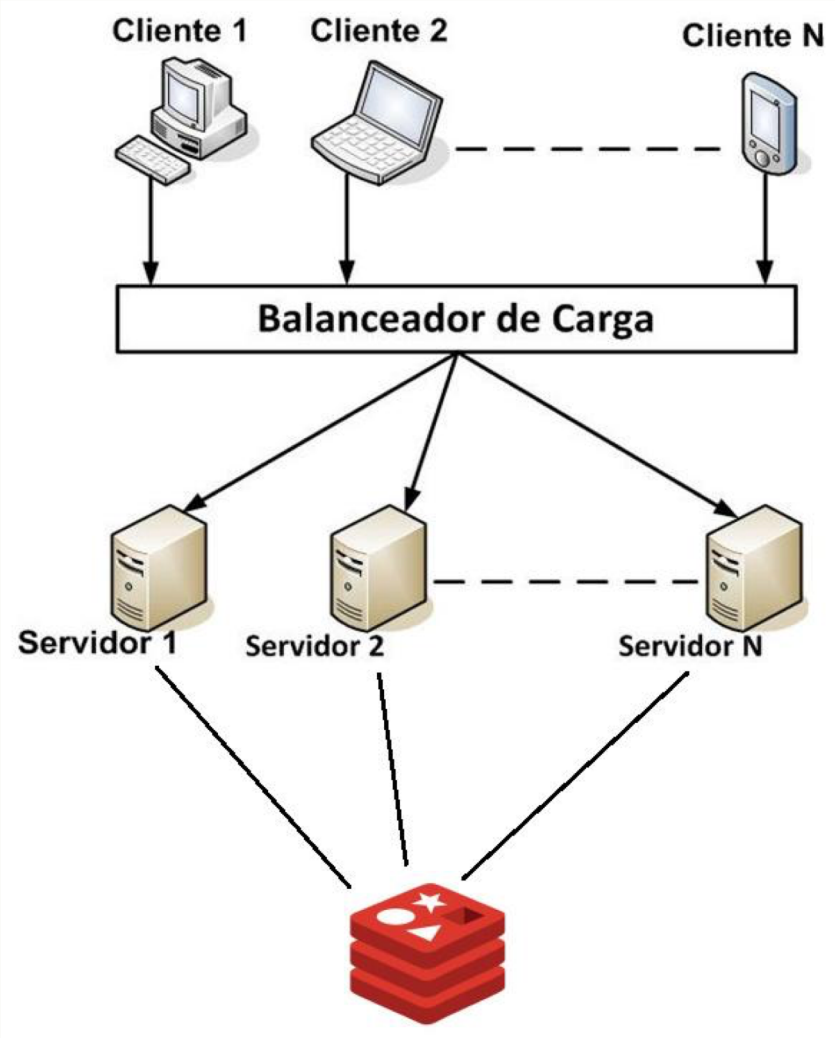
- Escalabilidade. O socket é mantido aberto com uma determinada instância do servidor. O número de conexões abertas é limitada.

- Por outro lado um rebalanceamento de carga do servidor irá causar perdas de conexão.

Formas de contornar o problema

- Existem técnicas que combinam alocação de sockets em banco de dados em memória (Redis por exemplo) que faz com que a realocação das instancias seja transparente para o cliente.

Utilizando Redis



Utilizando REDIS - Desvantagens

- Não resolve caso o pool de maquinas de servidores estejam distribuídas geograficamente.

Socket.IO - Servidor

```
const users = [];

server.listen(8080);

io.on('connection', function(socket){
      console.log('a user connected!');

      users.push(socket);

      socket.on('msg', function(msg){
         broadcast({ author: getName(socket), msg });
     });

      socket.on('set_name', function(name){
         console.log('set name: ',name);
         setName(socket, name);
    });

});
```

Socket.IO - Servidor

```
const users = [];

server.listen(8080);

io.on('connection', function(socket){
      console.log('a user connected!');

      users.push(socket);

      socket.on('msg', function(msg){
         broadcast({ author: getName(socket), msg });
     });

      socket.on('set_name', function(name){
         console.log('set name: ',name);
         setName(socket, name);
    });

});
```

Socket.IO - Servidor

```
const users = [];

server.listen(8080);

io.on('connection', function(socket){
      console.log('a user connected!');

      users.push(socket);

      socket.on('msg', function(msg){
         broadcast({ author: getName(socket), msg });
     });

      socket.on('set_name', function(name){
         console.log('set name: ',name);
         setName(socket, name);
    });

});
```

Socket.IO - Servidor

```
const users = [];

server.listen(8080);

io.on('connection', function(socket){
      console.log('a user connected!');

      users.push(socket);

      socket.on('msg', function(msg){
         broadcast({ author: getName(socket), msg });
     });

      socket.on('set_name', function(name){
         console.log('set name: ',name);
         setName(socket, name);
    });

});
```

Socket.IO - Servidor

```
const users = [];

server.listen(8080);

io.on('connection', function(socket){
      console.log('a user connected!');

      users.push(socket);

      socket.on('msg', function(msg){
         broadcast({ author: getName(socket), msg });
      });

      socket.on('set_name', function(name){
         console.log('set name: ',name);
         setName(socket, name);
    });

});
```

Socket.IO - Servidor

```
const users = [];

server.listen(8080);

io.on('connection', function(socket){
      console.log('a user connected!');

      users.push(socket);

      socket.on('msg', function(msg){
         broadcast({ author: getName(socket), msg });
     });

      socket.on('set_name', function(name){
         console.log('set name: ',name);
         setName(socket, name);
    });

});
```

Socket.IO - Cliente

```
myApp.controller('ChatController', ['$scope',
function($scope) {

    var socket = io.connect('http://localhost:8080');

    socket.on('msg', function(msg){
        msgs.push(msg);
    });

    $scope.setName = function(name) {
        socket.emit('set_name', name);
    };

    $scope.emitMsg = function(msg) {
        socket.emit('msg', msg);
    };
}]);
```

Socket.IO - Cliente

```
myApp.controller('ChatController', ['$scope',
function($scope) {

    var socket = io.connect('http://localhost:8080');

    socket.on('msg', function(msg){
        msgs.push(msg);
    });

    $scope.setName = function(name) {
        socket.emit('set_name', name);
    };

    $scope.emitMsg = function(msg) {
        socket.emit('msg', msg);
    };
}]);
```

Socket.IO - Cliente

```
myApp.controller('ChatController', ['$scope',
function($scope) {

    var socket = io.connect('http://localhost:8080');

    socket.on('msg', function(msg){
        msgs.push(msg);
    });

    $scope.setName = function(name) {
        socket.emit('set_name', name);
    };

    $scope.emitMsg = function(msg) {
        socket.emit('msg', msg);
    };
}]);
```

Socket.IO - Cliente

```
myApp.controller('ChatController', ['$scope',
function($scope) {

    var socket = io.connect('http://localhost:8080');

    socket.on('msg', function(msg){
        msgs.push(msg);
    });

    $scope.setName = function(name) {
        socket.emit('set_name', name);
    };

    $scope.emitMsg = function(msg) {
        socket.emit('msg', msg);
    };
}]);
```

Socket.IO - Cliente

```
myApp.controller('ChatController', ['$scope',
function($scope) {

    var socket = io.connect('http://localhost:8080');

    socket.on('msg', function(msg){
        msgs.push(msg);
    });

    $scope.setName = function(name) {
        socket.emit('set_name', name);
    };

    $scope.emitMsg = function(msg) {
        socket.emit('msg', msg);
    };
}]);
```

WebSockets - FIM

Wallace da Silva Ribeiro