César Augusto Furriela Nero - 170221080
David Miguel Gregório Afonso - 170221081
João Pedro Pires da Silva - 170221099

1. **Deteção de métodos inexistentes**

O nome dos métodos a serem utilizados/existentes é passado pelo servidor no início da sessão do cliente ou quando o comando "\help" é aplicado, todos os outros métodos remetem para uma exceção que indica ao utilizador como proceder.

2. **Deteção de parâmetros inválidos**

Caso os parâmetros sejam inválidos, o resultado por parte do servidor é retornado como "error".

3. **Mecanismos para "registo" de funções no servidor**

Na criação do servidor, é passado como argumento um tuplo de funções, que internamente irão ser adicionadas com recurso a um dicionário.

Quando o cliente se conecta com o servidor, são pedidos os nomes das funções existentes. Deste modo, quando uma função não reconhecida é chamada do lado do cliente, é feita uma comparação com a lista recebida para ver se a mesma pertence ao servidor.

4. **Várias invocações de funções num único pedido**

Optámos por não fazer esta opção. Abordou-se a possibilidade de acumular pedidos no lado do cliente e apenas mandar tudo de uma vez só quando toda a linha fosse processada. No entanto, isto não nos possibilitava fazer algo do género add(2,add(2,2)) pois a primeira função iria estar dependente do resultado da segunda. Havia também a possibilidade de enviar todo o input do cliente para o servidor, para ai ser processado "do outro lado", mas não achámos que isso fosse o pretendido neste exercício.

5. **Implementação de notificações json-rpc**

Existe uma lista em ambos os lados (cliente/servidor) com os métodos pertencentes às notificações, se estes forem invocados, o cliente não esperará resposta e o servidor não a irá enviar.

6. **Reutilização de ligações tcp/ip**

Utilizando o módulo asyncore cada cliente é trabalhado de forma independente, o que possibilita a continuação da ligação de cada um e a reutilização do IP.

7. **Múltiplos clientes em paralelo**

Devido à criação de um novo thread para os clientes o servidor permanece responsivo na própria consola do servidor ou a qualquer pedido dos seus clientes.

8. **Respostas assíncronas**

O servidor está sempre a aguardar novas ligações, tendo a habilidade de aceitar múltiplos clientes.

9. **Múltiplos parâmetros**

Através da utilização de tuplos na obtenção de variáveis e interação das mesmas, é possível a criação de métodos com um número dinâmico de variáveis.

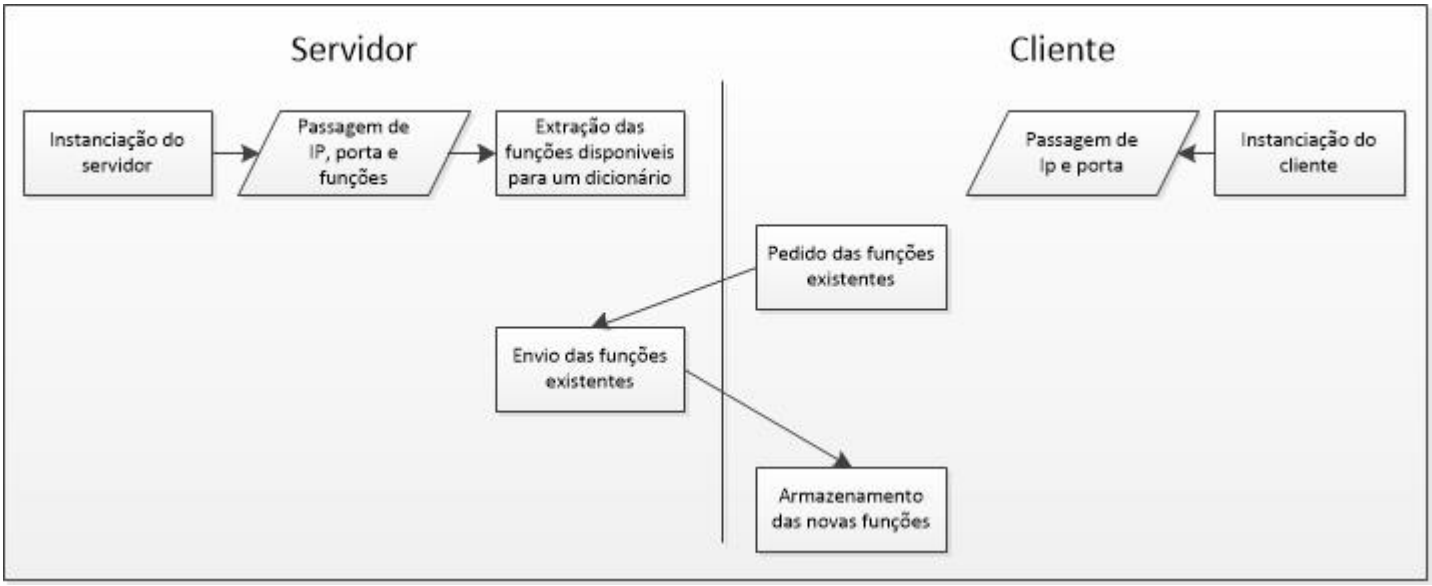
César Augusto Furriela Nero - 170221080
David Miguel Gregório Afonso - 170221081
João Pedro Pires da Silva - 170221099

10. Cliente e servidor como bibliotecas

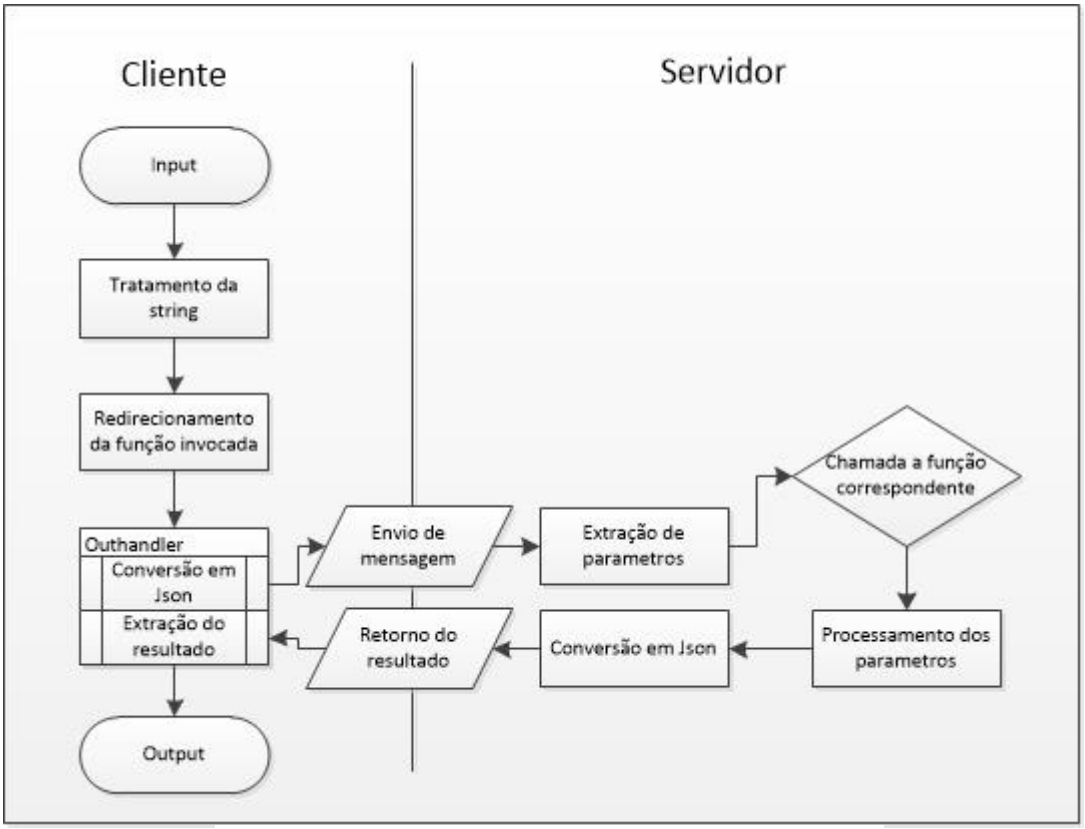
O cliente servidor estão construídos de modo a poderem ser instanciados e reutilizados.

11. Uso com e sem shell

No lado do cliente, é possível escolher se queremos ou não usar shell (True ou False nos parâmetros).



Processo de inicialização do cliente e servidor



Processo de comunicação do cliente com o servidor