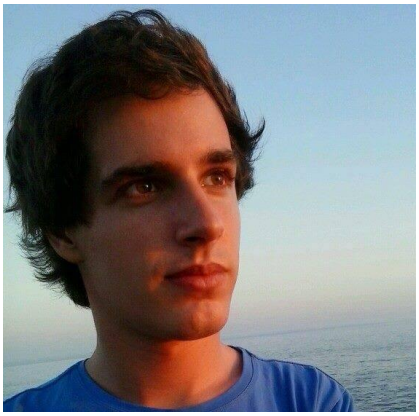
Sistemas Distribuídos

Relatório da 2º Entrega

Grupo A38

Luís Santos

Nº75964



Filipa Costa

Nº75888

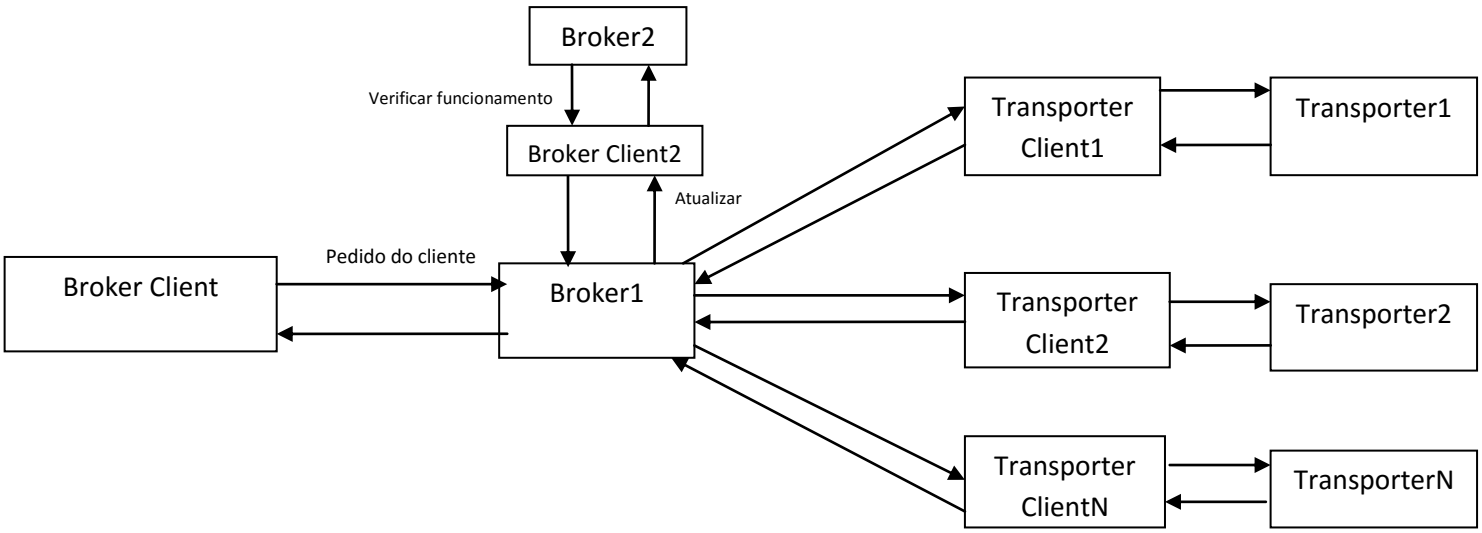
Ana Galvão

Nº75312

URL do repositório no GitHub:

https://github.com/tecnico-distsys/A_38-project.git

Replicação do Broker



Descrição:

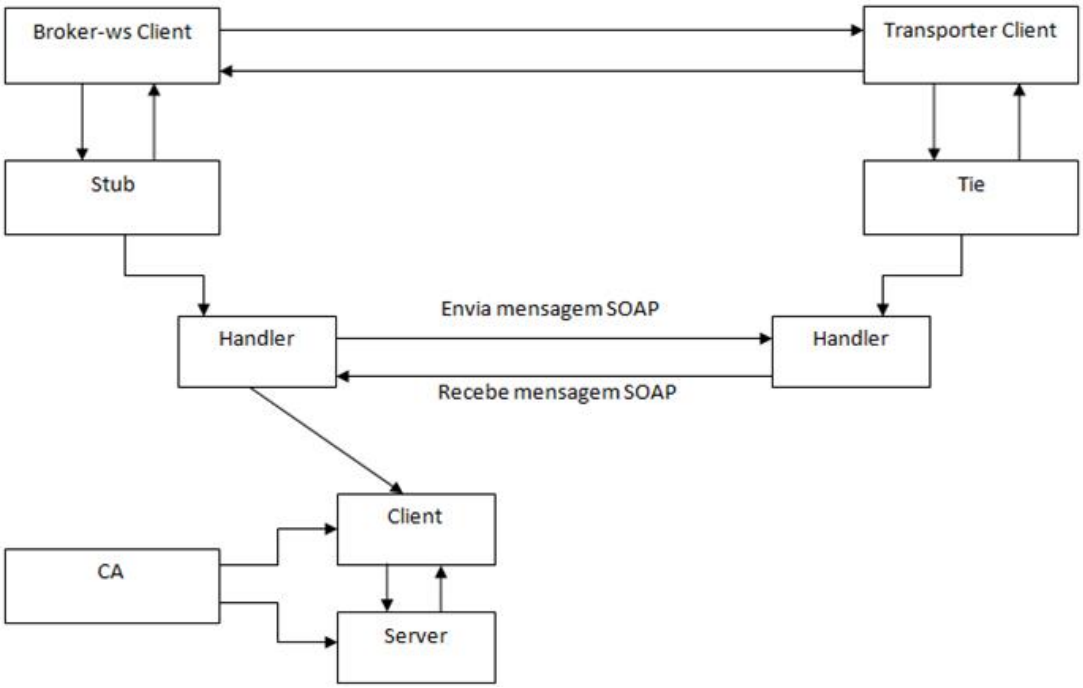
Nesta figura representa-se o estado de funcionamento normal do sistema, com a replicação em curso. Tendo o Broker1 a fazer a comunicação com os N Transportadores, qualquer operação pedida pelo BrokerClient (que na prática seria o utilizador do sistema) que necessite de guardar alguma informação no Broker1 (novos pedidos, atualizações de estado, limpeza de pedidos) irá desencadear uma operação de atualização do Broker1 para o Broker2, de maneira a que este último tenha uma cópia de tudo o que o Broker1 irá estar a guardar no decorrer do seu funcionamento.

Explicação da solução:

A solução baseou-se em 3 partes: Primeiro, o Broker2 teria que se ligar mas sem publicar o seu respetivo url (apenas o Broker1 saberá, nesse ponto). De seguida, O Broker1 liga-se e, sabendo que é o primário, vai criar um BrokerClient com ligação para o Broker2, e irá utilizá-lo para efetuar as operações de atualização sobre o Broker2. A última parte será quando o Broker1 se desligar: O Broker2 ficou , durante o tempo de execução do Broker1, a enviar pings para o Broker1, de modo a apanhar a exceção de ligação quando o último fosse desligado. Apanhando essa exceção, o Broker2 sabe que o Broker1 se desligou (foi implementado um booleano para certificar que se espera primeiro que o Broker1 se ligue), e coloca-se então no lugar do antigo Broker1, alterando o seu url para o do antigo Broker1. Desse modo, os clientes só terão que aguardar alguns segundos, os necessários para que o Broker2 perceba que o Broker1 se desligou e fazer a mudança de url.

Foi implementado, no BrokerClient, um tempo de espera para cada chamada ao Broker no caso de haver uma WebServiceException, voltando de seguida a tentar a ligação (dando assim tempo para o Broker2 assumir o lugar do Broker1). De notar que o tempo de espera do método ping é ligeiramente menor que os restantes, por ser o método utilizado pelo Broker2 para saber do funcionamento do Broker1.

Segurança



Descrição:
No nosso sistema ligamos o broker-ws, no papel de cliente, ao transporter-ws-cli no papel de servidor.
Ambos contêm um handler que lhes permite enviar mensagens SOAP de um lado para o outro, com o conteúdo pretendido.
Temos também a Certificate Authority (CA), que contém um cliente e um server, e que se ligam ao handler do broker-ws.

Explicação da solução:

Tínhamos como intenção os handlers do broker e do transporter conseguirem aceder ao CAClient - que iria utilizar um método do ca-ws, o getCertificate - para retirar a chave pública de todos os elementos e, assim, poderem assinar as suas mensagens - após cifra com SHA-1. Depois, tendo cada um conhecimento da sua chave privada, cada handler iria fazer o processo contrário, e o CAClient iria confirmar que as mensagens eram iguais antes e após a assinatura.
Contudo tivemos diversos problemas de implementação. Antes de mais, os handlers emitiam excepções que impediam o nosso broker de enviar correctamente as mensagens ao transporter-ws-cli, e não conseguimos detectar onde. Isto resultou em termos de comentar os handler, sendo que a assinatura não chegou a ser feita porque também não conseguimos aceder correctamente ao CA e, por isso, à chave pública de todos. Portanto, apesar de termos os handlers implementados, não os temos a funcionar no projecto.
Da mesma forma, testámos “hard-coded”, passando as chaves públicas e privadas de cada um para os seus resources, cifrar e decifrar mensagens, sendo que o processo estava bem feito, mas no seu conjunto não resultou.