T25

TÉCNICO LISBOA

Sistemas Distribuídos

2.º Semestre 2015/2016

SD-Upa

https://github.com/tecnico-distsys/T_25-project/

Relatório de Projeto

77900 Luis Santos

77966 Pedro Filipe

78030 Constantin Zavgorodnii

1 Segurança

1.1 Figura Descritiva

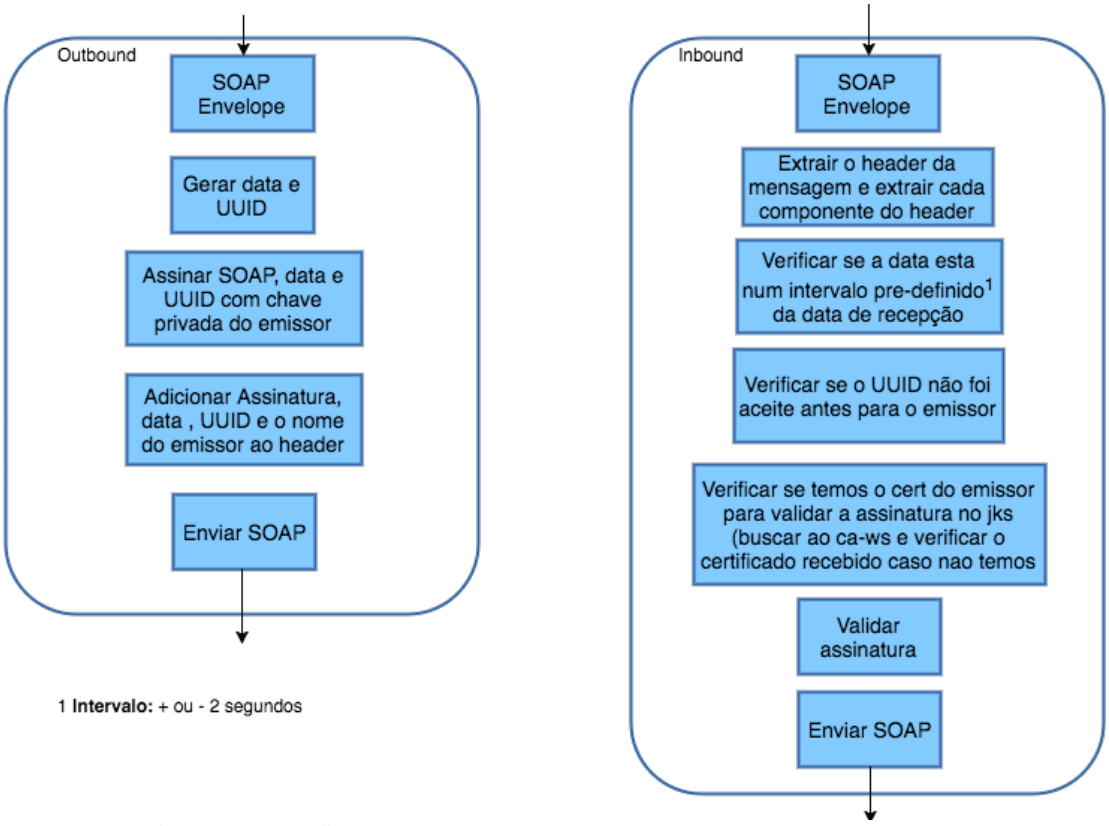


Figura 1 - Authentication Handler

1.2 Racional

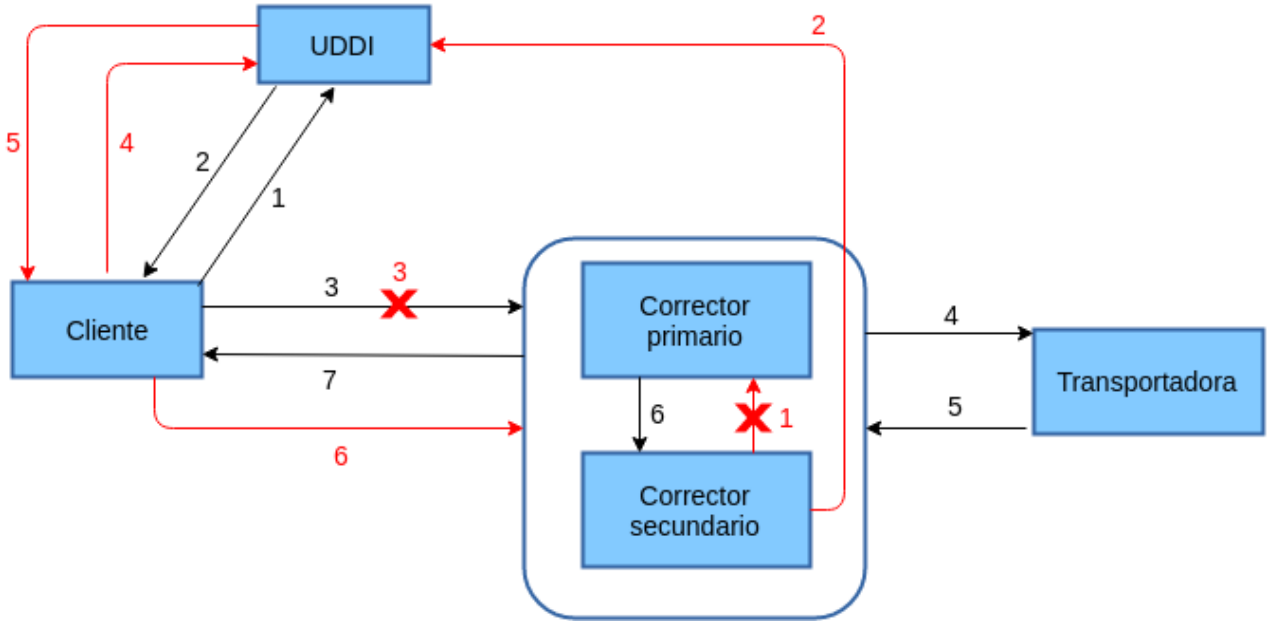
Os objetivos de garantir o não repudio, a autenticidade e a "frescura" das mensagens foram atingidos da seguinte forma:

1. **Não Repudio e Autenticidade:** no cabeçalho das mensagens SOAP mandamos um digest assinado que contem o corpo da mensagem SOAP, a data em que foi criada a mensagem e do UUID da mensagem. Incluímos também no cabeçalho da mensagem o nome do emissor, e.g. "UpaBroker", "UpaTransporter1".
   Quando o recetor recebe a mensagem compara a assinatura da mensagem com os elementos correspondentes da mensagem.
2. **Frescura:** a frescura é garantida pela data enviada na mensagem e pelo UUID, ou seja, quando uma mensagem está de saída adicionamos a data e o UUID e quando está de entrada verificamos se a data enviada com a mensagem esta dentro de um intervalo de +- 2 segundos do tempo de receção da mensagem. Se não tiver é rejeitada. Também verificamos se o UUID para o emissor já foi tratado anteriormente ou não, se já foi tratado então a mensagem é rejeitada.

2 Replicação Primário/Secundário

Replicação do UpaBroker em modo passivo com um corretor primário e uma réplica.

2.1 Figura Descritiva



2.2 Descrição da imagem

Funcionamento normal:	**Funcionamento em caso da falha do corretor primário**:
1, 2 : Cliente obtém o URL do corretor **3** : Cliente estabelece conexão com o corretor **4, 5** : Corretor envia pedidos e recebe respostas da Transportadora **6** : Corretor primário envia cópia de dados para corretor secundário **7** : Corretor responde para cliente	**1** : Corretor secundário descobre a falha do corretor primário **2** : Corretor secundário atualiza o URL do corretor **3** : Cliente descobre a falha do corretor **4, 5** : Cliente obtém novo URL do corretor **6** : Cliente continua a iteração com corretor

2.3 Racional

Para implementar a replicação foi necessário adicionar duas operações ao WSLD do corretor: alive():boolean e updateTransport(TransportData):void. Foi também criado um tipo de dados para transportar o os dados da transportadora.

Na nossa solução, o corretor secundário é quem vai perguntar ao corretor primário se está disponível em vez de ser ao contrário, ou seja, como temos o caso dos web services e os pacotes são enviados via html e temos uma ligação TCP que é bidirecional está abordagem é possível. Se apenas tivéssemos uma ligação unidirecional em que seria necessário enviar mensagem do primário ao secundário. Nós também não implementamos uma thread independente no secundário para a verificar se o primário está disponível visto que os web services são por natureza threaded.