Sistemas Distribuídos 2015-2016

Grupo T67

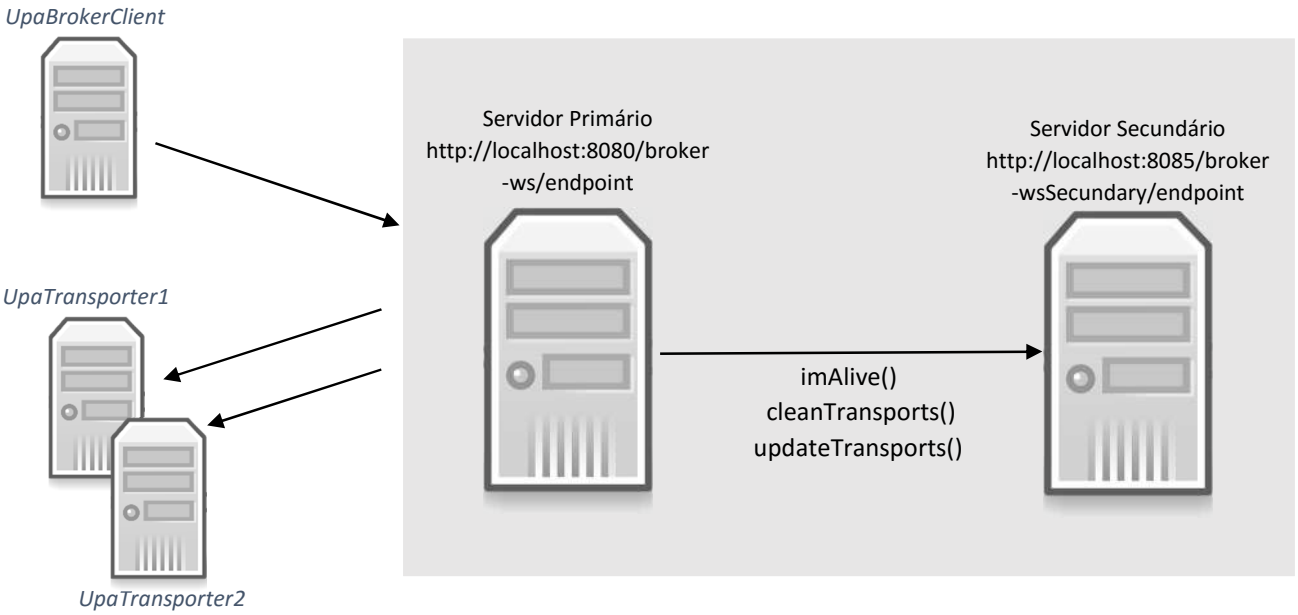
https://github.com/tecnico-distsys/T_67-project

Miguel João Gil Catarino
78084

Tiago Costa Santos
77936

Francisco Miguel Garcia Dias
70612

1. Replicação



O contrato WSDL foi alterado para incluir mais três WebMethods para separar completamente os métodos usados para comunicação entre Cliente-Servidor e Servidor-Servidor são eles:

imAlive() – serve para enviar as provas de vida do servidor primário para o servidor secundário, ao receber esta chama o servidor secundário altera o valor de uma variável para true;

cleanTransports() – para limpar todos os transportes que o servidor tem guardados;

updateTransports() – envia um objeto TransportView e um ID para o servidor secundário para este o adicionar á sua lista de transportes com o ID dado.

Foram implementadas duas classes Runnable, uma para ser executada no Servidor Primário e outra no Servidor Secundário:

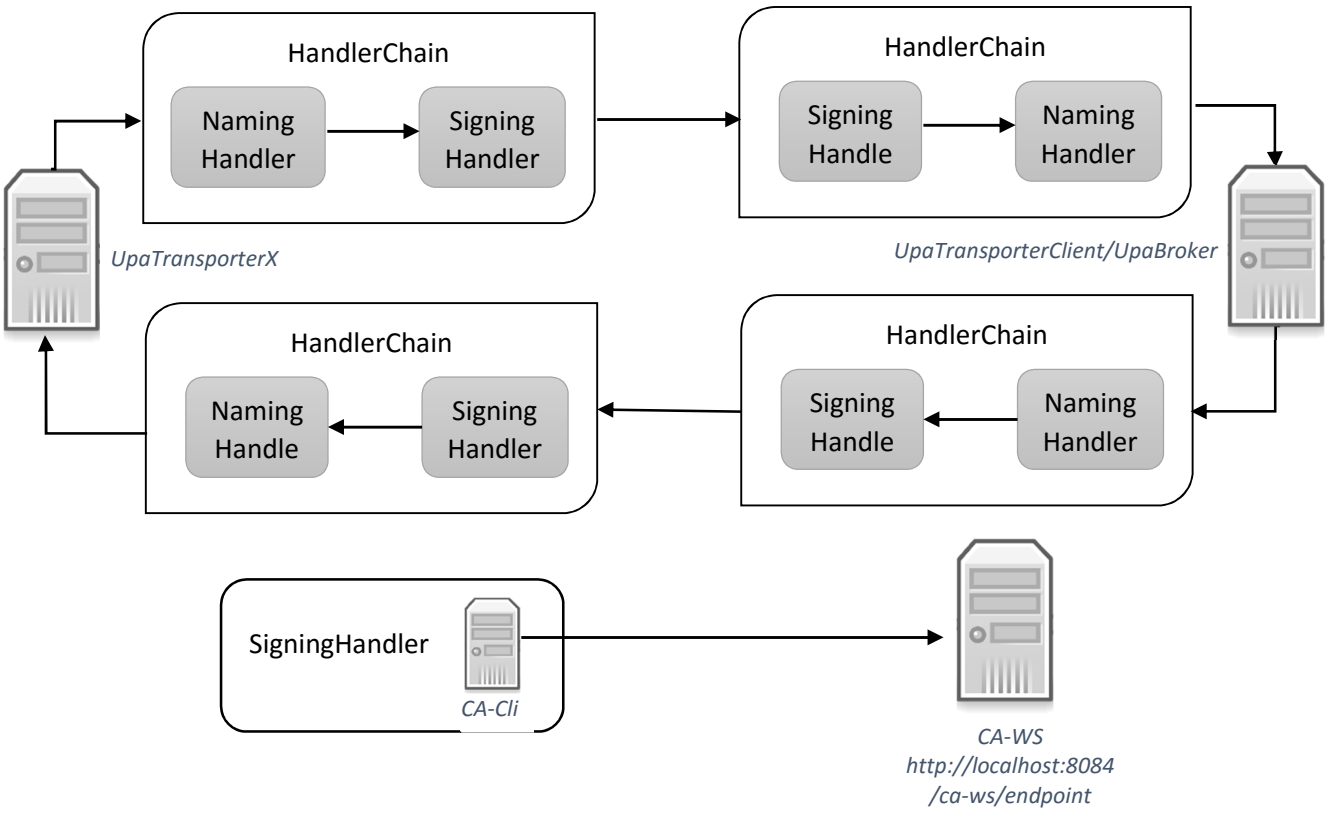
Notifier (Primário) – Em ciclo infinito, envia provas (imAlive) de vida de 3 em 3 segundos;

Checker (Secundário) – Verifica se foi recebida uma prova de vida (se a variável foi alterada) nos últimos 3,2 segundos e volta a alterar o valor da variável para false. (3,2 segundos para compensar alguma possível demora na rede ou outro tipo de problema que cause um atraso de forma a evitar que o servidor tente tomar o lugar do primário sem razão).

Ao iniciar a execução o servidor primário liga-se ao servidor secundário, previamente registado no servidor de nomes e inicia então o envio das provas de vida.

O servidor secundário ao detetar que não foi recebida uma prova de vida no intervalo referido, substitui o servidor primário, faz isto removendo o seu próprio registo do servidor de nomes UDDI voltando então a registar-se com as propriedades do servidor primário

2. Segurança



Para garantir a segurança da comunicação entre Transportadoras e Corretor foram criados dois handlers que processam as mensagens como mostrado na figura.

NamingHanlder – Regista, no cabeçalho da mensagem, quem enviou a mensagem e para quem a mensagem foi enviada, este handler foi desenvolvido principalmente para não ser necessário ter código repetido de para o SigningHandler nas diferentes componentes do projeto.

SigningHandler – Handler responsável por assegurar a segurança na comunicação. Este handler lê os nomes das entidades em comunicação registados na mensagem pelo Naming Handler e usa os certificados das respetivas para produzir, assinar o Resumo e escreve-lo no cabeçalho da mensagem caso seja uma mensagem em saída, caso contrário verifica se assinatura que veio na mensagem que chegou é válida. No caso de o handler não conseguir aceder aos certificados, este irá ligar-se á Entidade de Certificação e pedir o certificado necessário, ao recebe-lo verifica se o certificado é válido verificando se foi devidamente assinado pela entidade de certificação.

Para a distribuição dos certificados foi criado um simples WebService que apenas tem um WebMethod que recebe o nome da entidade e devolve o certificado da mesma. O cliente deste WebService é instanciado no SigningHandler como demonstrado na figura.