INSTITUTO SUPERIOR TÉCNICO

Sistemas Distribuídos 2015-2016

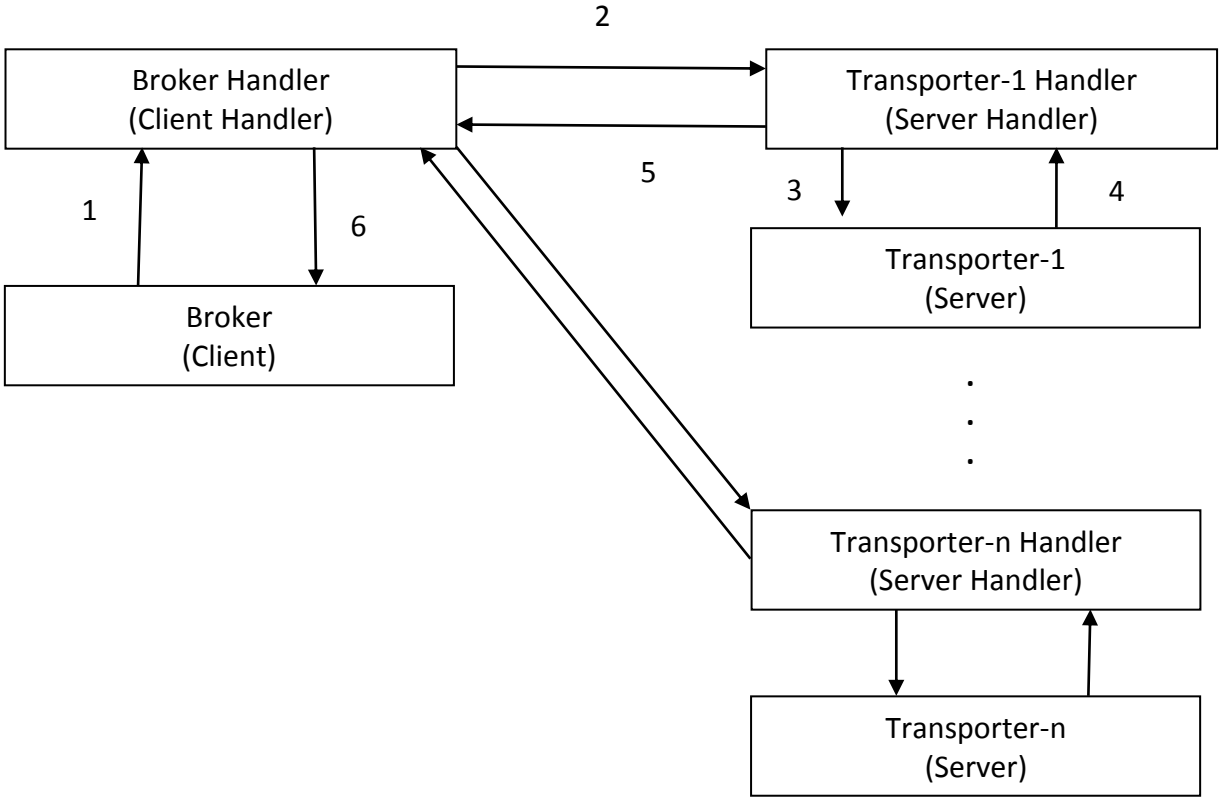
A-64

https://github.com/tecnico-distsys/A_64-project

ANDRÉ ÁGUAS - 78854
JOÃO NETO - 78745
RUI SÁ - 78324

SEGURANÇA

A abordagem tomada em relação à segurança contempla a autenticidade, a frescura e o não repúdio das mensagens (não é necessário garantir confidencialidade).
A comunicação ocorre entre o Broker (que corresponde ao cliente) e as várias Transporter (cada uma das quais corresponde a um servidor).
Por Contexto, o Broker envia ao seu handler (Client Handler) o nome da Transporter com que vai comunicar. O seu handler intercepta a SOAP Message Context e vai alterar SOAP Message. Em primeiro lugar, o handler recebe o número da Transporter com que o Broker quer comunicar e mete-o no Header da mensagem. De seguida, gera o nonce e coloca-o no Header. A seguir, obtém o Body, aplica digest sobre o Body, concatena-o com o nonce e usa a chave privada do Broker (que foi buscar à sua KeyStore) e para assinar a concatenação anterior. Todo este resultado é colocado no Header e enviado ao handler da Transporter (Server Handler) em questão.
O handler da Transporter intercepta a SOAP Message e obtém o seu Header. Primeiro, tira o nome da Transporter e guarda-o. Depois, retira o nonce e verifica se é único, de modo a garantir a frescura da mensagem e assim evitar replay attacks. De seguida, pega no Body, aplica-lhe digest e concatena-o com o nonce que recebeu. A seguir vai buscar o certificado da CA e, com a chave pública da CA, verifica se a fonte é fidedigna. Depois, vai buscar o certificado do Broker à CA e, a partir do certificado, obtém a chave pública do Broker. Por fim, usa a chave pública do Broker para verificar (verify) a assinatura digital.
A mensagem chega à Transporter e envia o orçamento, que segue como mensagem pelo processo inverso.



REPLICAÇÃO

A abordagem tomada em relação à replicação baseia-se na existência de dois servidores Broker: o servidor Broker principal e o servidor Broker secundário. O servidor principal é responsável por garantir que existe um servidor secundário pronto a ser lançado em caso de necessidade. O servidor secundário é responsável por detectar uma falha no servidor primário e, nesse caso, assume o papel de servidor principal. Por simplificação do problema, considerou-se que um servidor só pode falhar uma vez, isto é, quando um servidor inicialmente secundário passa a primário, este não precisa de garantir a existência de um servidor alternativo.
O servidor principal envia duas mensagens ao servidor secundário. Uma delas é o "i'm alive", enviado em intervalos de tempo constantes, que notifica o servidor secundário que está operacional. Esta é a medida que permite ao servidor secundário detectar uma falha silenciosa no servidor principal. A outra mensagem é a de "update", que notifica o servidor secundário sempre que há uma actualização do seu estado. O servidor secundário é responsável por receber estes novos dados e actualizar imediatamente o seu estado, de modo a garantir a replicação de dados em qualquer instante.
O servidor secundário tem de ter um timeout que é necessariamente superior ao intervalo de tempo entre duas mensagens "i'm alive" consecutivas provenientes do servidor principal. Quando o servidor principal falha, o servidor secundário dá timeout e percebe que ocorreu uma falha. O que o servidor secundário tem de fazer é simplesmente fazer "rebind", que consiste em alterar o nome de "UpaBroker2" para "UpaBroker". O objectivo é fazer com que o cliente apenas procure por "UpaBroker" no servidor de nomes.
Também o cliente necessita de ter timeout na comunicação com o Broker. Se Broker demorar muito tempo a responder a um pedido, o cliente tem de ir ao servidor de nomes procurar por "UpaBroker", pois nesse momento pode ter sido alterado (pode estar num endpoint diferente).