Sistemas Distribuídos

2015 - 2016

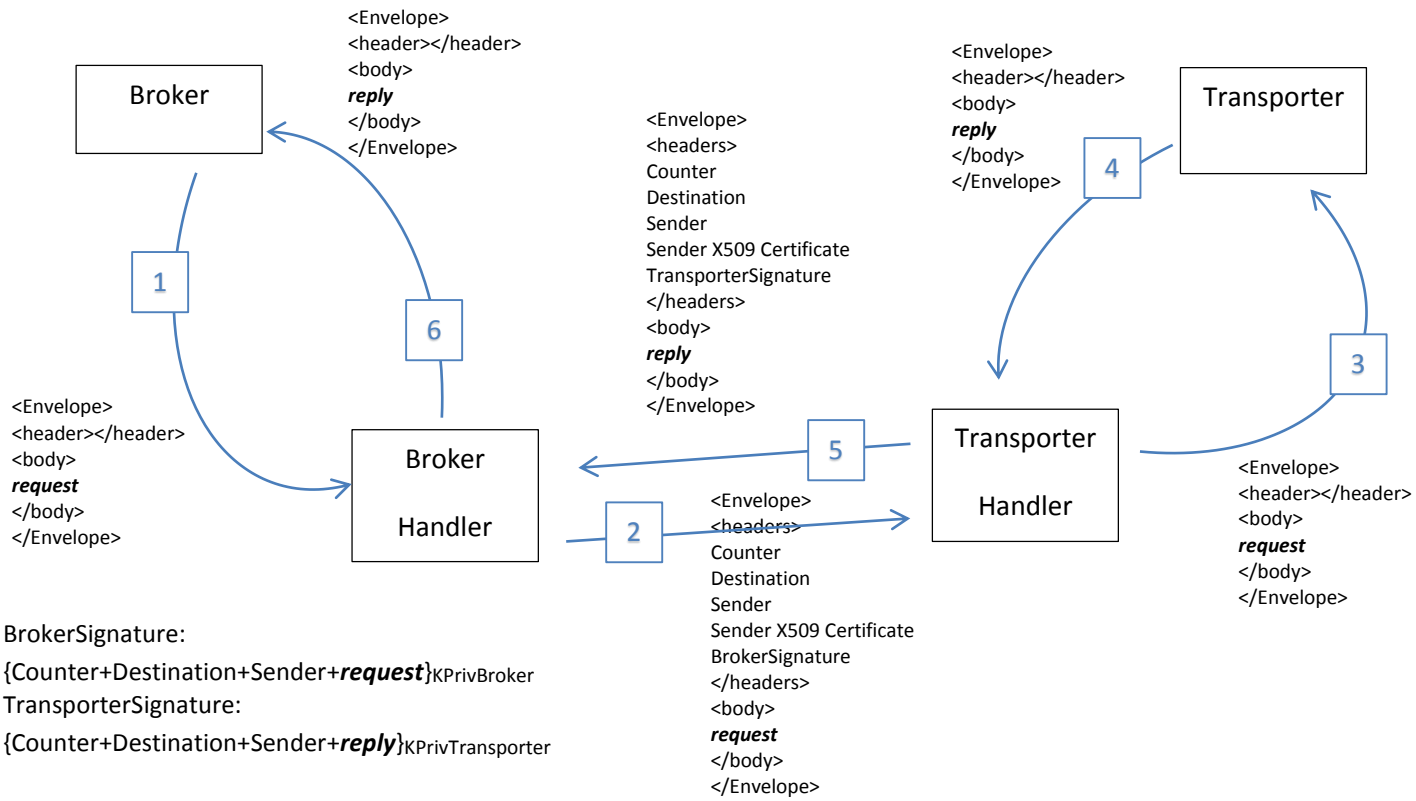
Grupo T70

https://github.com/tecnico-softeng-distsys-2015/T_70-project

Daniel Reigada
82064

Diogo Mesquita
81968

João Plancha da Silva
51355



BrokerSignature:

{Counter+Destination+Sender+***request***}$_{KPrivBroker}$

TransporterSignature:

{Counter+Destination+Sender+***reply***}$_{KPrivTransporter}$

Descrição

Na figura acima estão descritas, de forma simplificada, as trocas de mensagens SOAP entre o Broker e um Transporter. São usados *Handlers* nos dois extremos, para garantir a autenticidade, integridade, não repúdio e frescura das mensagens.

A mensagem SOAP com o pedido inicial, enviada pelo broker e tendo como destino o transporter, é primeiramente interceptada pelo *handler* do lado do broker. Este coloca-lhe 4 novos *headers*: Counter, Destination, Sender, Sender X509 Certificate e Signature, que se descrevem em seguida.

Em Counter é colocado um contador geral, controlado pelo broker, que juntamente com a informação colocada em Destination (*endpoint* do transporter), se destina a garantir a frescura das mensagens enviadas.

Em Sender é colocado o *endpoint* do broker, para garantir o não repúdio.

Em Sender X509 Certificate, é colocado o certificado público do broker, que contém a chave pública do mesmo, assinada pela autoridade de certificação, para o receptor a possa decifrar o conteúdo da assinatura digital.

Finalmente, em Signature, coloca-se a assinatura digital (resumo+cifra) da concatenação do Counter, Destination, Sender e o conteúdo do *body* do SOAP, que consiste na mensagem a transmitir. Este campo é devidamente cifrado com a chave pivada do broker.

Seguidamente a mensagem é transmitida ao *handler* do lado do transporter, conforme indicado na figura.

Num primeiro momento, o *handler* verificará o valor do Counter e do Destination, validando assim que a mensagem foi recebida pela entidade correcta. Seguidamente usa a chave pública do broker, incluída na mensagem SOAP, para fazer a verificação da assinatura digital contida no *header* Signature (usando também os headers e o texto do *body*, tal como descrito anteriormente). Se a assinatura for válida, significa que todos os pressupostos foram verificados, e a mensagem pode seguir para o transporter, para que este a possa processar e dar a resposta necessária.

Na resposta do transporter ao broker, os papéis dos *handlers* invertem-se e a mensagem seguirá o caminho inverso ao descrito, obedecendo exactamente aos mesmos passos de segurança e verificação de validade da mensagem, tal como descrito na figura acima.

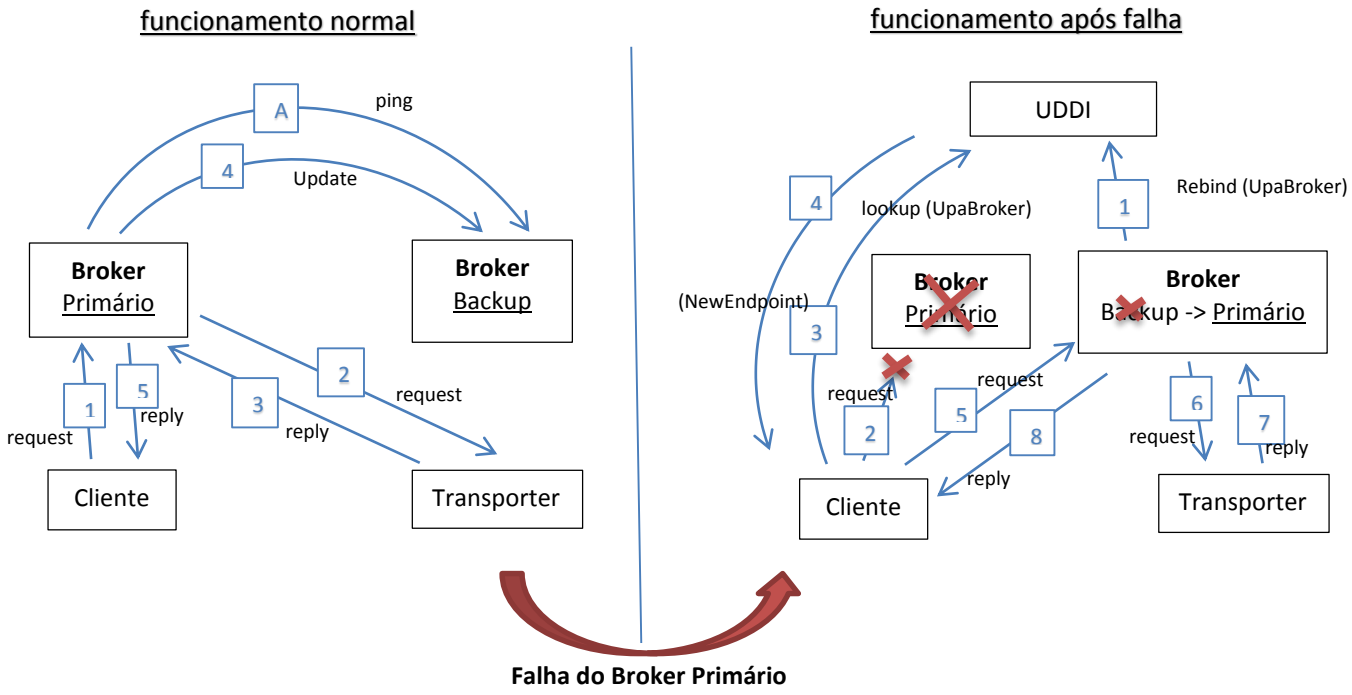
Racional

A assinatura é o elemento central que garante a autenticidade, integridade e não repúdio da mensagem. Para isso, usa o conteúdo da mensagem e as referências das entidades de origem e de destino. O facto de ser cifrada com a chave privada da entidade de origem, garante que só esta a pode ter enviado. O contador (que também faz parte da assinatura) e a entidade de destino garantem a frescura da mensagem.

Na solução apresentada, não foi contemplada a existência de um webservice para a autoridade de certificação (CA), embora todos os certificados públicos sejam assinados pela CA. Note-se que qualquer verificação de assinatura é sempre precedida pela verificação da validade do certificado de chave pública, recorrendo à chave pública da CA.

Considerou-se que a existência de um webservice para a CA (para distribuição de certificados) acrescentaria um nível de complexidade extra, incluindo mais um possível ponto de ataque e falha, centralizando a distribuição de certificados. No caso de um ataque ao CA, ou de uma falha que a levasse a deixar de poder responder a pedidos, todo o sistema ficava impossibilitado de funcionar.

Assim, optou-se por responsabilizar as entidades pelo envio dos próprios certificados (assinados pela CA), tornando o sistema autónomo para a distribuição de certificados, e acrescentando uma maior descentralização e tolerância a falhas.



Descrição

Na figura acima estão representados: a sequência de funcionamento normal, em que o broker primário e o broker backup se encontram a funcionar em pleno (à esquerda), e a sequência de funcionamento do sistema imediatamente após a falha do broker primário (à direita).

No caso de funcionamento normal, assume-se que ambos os brokers e o transpoter se encontram registados no UDDI, daí este elemento não estar representado no esquema. A sequência 1-2-3-5 corresponde a um pedido-resposta comum, entre o cliente e o transporter, sendo intermediado pelo broker. O passo 4, corresponder ao update do broker backup, por parte do broker primário. De cada vez que é executada uma operação e o broker primário altera o seu estado interno, este update é enviado ao broker backup, para que os seus estados permaneçam idênticos. Note-se que o Counter (descrito na secção anterior e que garante a frescura das mensagens), também é mantido síncrono com o broker backup, para que o sistema continue a funcionar em pleno e de forma transparente para os intervenientes, em caso de falha do broker primário.

Finalmente, observe-se que o passo A, que não faz parte da sequência descrita, corresponde à prova de vida que o broker primário faz ao broker backup. Esta mensagem é enviada em intervalos regulares, e dá conhecimento ao broker backup do estado do primário. No caso de o tempo entre envios de mensagem ser excedido (caso em que o broker primário falha), o broker backup é responsável por assumir-se como broker primário, registando-se no UDDI com o mesmo nome, e assegurando a continuidade do serviço de forma transparente para clientes e transporters.

No caso de funcionamento após falha do broker primário, o broker backup não recebe a prova de vida em tempo útil, assumindo-se assim como primário e fazendo o rebind no UDDI (passo 1), com o mesmo nome (UpaBroker).

No momento em que o cliente tenta enviar um request para o broker (passo 2), a entrega da mensagem falha, uma vez que a o endpoint que o cliente tem como referência para o broker não está a apontar para o novo servidor. Assim, o cliente efectua um novo lookup junto do UDDI, procurando por UpaBroker (passo 3), recebendo como resposta o endpoint do novo broker primário (ex-backup).

Por fim, a sequência 5-6-7-8, representa novamente um pedido-resposta comum (5 é a repetição de 2). Nesta situação, o sistema encontra-se no modo normal de funcionamento, aparte do envio de updates e provas de vida, uma vez que deixou de existir um servidor de backup.

Racional

A prova de vida (ping) é unidirecional (do primário para o backup), uma vez que não é necessário obter resposta do backup. Por outro lado, a implementação de um sistema inverso, em que o backup pede uma prova de vida e aguarda resposta do primário, seria menos eficiente, obrigando o servidor de backup a executar uma operação desnecessária, e o servidor primário a aguardar por chamadas do servidor secundário.

Em relação ao update, apenas as novas alterações ao estado interno do broker são enviadas ao backup (uma de cada vez), como por exemplo, um novo transporte ter sido decidido. Desta forma, minimiza-se a quantidade de informação transmitida e a dimensão da operação a realizar.

É ainda importante reiterar que o contador geral de mensagens, que garante a frescura das mesmas (Counter - descrito na secção anterior), é também objecto de update no broker backup.