

Sistemas Distribuidos 2015-2016

Relatório Projecto Upa Transportes



Gonçalo Gaspar

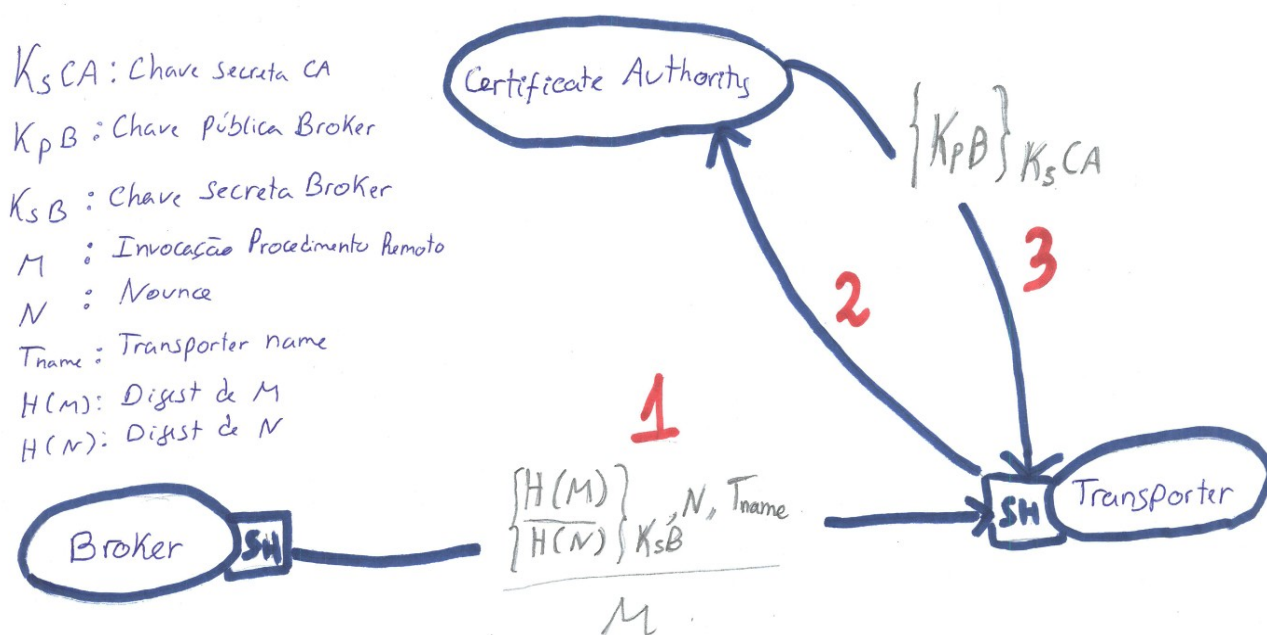


António Silva

Grupo A49

URL repositório: https://github.com/tecnico-distsys/A_49-project

1. Segurança



O Diagrama ilustra uma invocação remota feita pelo *Broker* a um *Transporter*. No caso da resposta a esta invocação, iniciada pelo Transporter, este passa a ser o emissor. O processo é portanto, inverso, invertendo-se os papéis e a estrutura do diagrama, mas mantendo-se a estrutura das mensagens.

A ordem sequencial das mensagens é indicada pela numeração a vermelho.

Os handlers representados (SH - Security Handler) são entidades que interceptam todas as **SOAP** messages incoming e outgoing tanto no *UpaBroker* como nos *UpaTransporters*, respectivamente.

Sempre que existe um **envio** de mensagem, o handler à saída do emissor faz o seguinte:

1º Gera um número aleatório (*Nounce*), faz o seu digest (*digestedNounce*) e cifra este com a sua chave privada (*cipheredDigestedNounce*). Insere **cipheredDigestedNounce** e **Nounce** no header da mensagem SOAP. O *Nounce* tem como objectivo garantir a **frescura** das mensagens, e o envio destes dois elementos (*DigestedNounce* e *Nouce*) tem como objectivo garantir a **integridade** do *Nounce* gerado.

2º Captura o conteúdo da mensagem SOAP *outgoing* (*soapMsg*), faz o seu digest (*digestedSoap*) e cifra-a com a sua chave privada (*cipheredSoap*). Insere **cipheredSoap** e **soapMsg** no header da mensagem SOAP, tendo como objectivo garantir a **integridade** da mensagem enviada.

Sempre que existe uma **recepção** de mensagem, o que o handler que a intercepta faz, é:

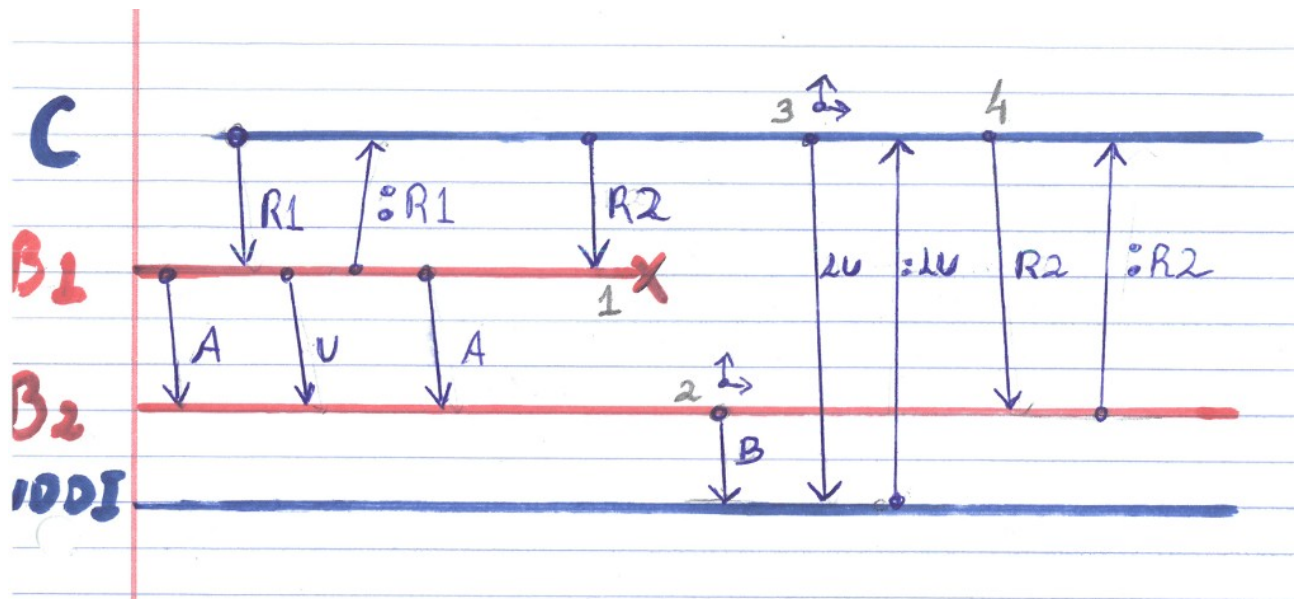
1º Se não tem em sua posse o certificado de chave pública da entidade que enviou a mensagem, pede-o à Central Authority (**CA**). Este passo tem como objectivo garantir a **autenticidade** e **não-repúdio** da mensagem recebida. Verificar se o certificado do emissor é decifrável com a chave pública da CA de confiança, valida a entidade emissora pois este certificado foi assinado com a chave privada da CA. Se estiver tudo validado, o receptor

procede a decifrar os conteúdos do *SOAP header* com a chave pública obtida da CA. Se houver erros na verificação da chave, envia uma SOAPFault pois não se garante a autenticidade do remetente.

2º Decifra *cipheredDigestedNounce* e verifica se é igual ao digest de *nounce*. Se sim, verifica se já recebeu alguma mensagem com esse *nounce*, garantindo a **frescura** da mensagem. Caso algum passo tenha resultado inesperado é lançada uma SOAPFault.

- Decifra *cipheredSoap* e faz o *digest* da **SOAP message** recebida sem os seus headers. Se são iguais a **integridade** da mensagem não foi comprometida, caso contrário é lançada uma SOAPFault.

2. Replicação



Entidades	
C	Cliente
B1	Broker Primário
B2	Broker Secundário
UDDI	Servidor de Nomes

Mensagens	
A	Proof of life
(R1, :R1)	(Request 1, Reply to Request 1)
U	Update state
(R2, :R2)	(Request 2, Reply to Request 2)
B	Bind
(LU, :LU)	(LookUp, LookUp Reply)

Eventos	
1	Paragem do Servidor Primário
2	TimeOut na espera da prova de vida. Bind no Uddi-Naming com o nome do Primário.
3	TimeOut no cliente na espera de resposta ao pedido. Consulta ao Uddi-Naming pelo endereço do serviço.
4	Reenvio de novo pedido para a Replica (agora a funcionar como principal).

A implementação do sistema de tolerância a faltas do Corretor (Upa Broker) foi realizada através de :

- Expansão da interface partilhada de UpaBroker através da introdução das operações no **contrato WSDL**. Foram introduzidas e implementadas as operações que fazem o update de estado do broker secundário e fazem o envio de provas de vida, chamaremos **udpateBroker** e **sendAlive**.
- Introdução de *timeouts* para a conexão e recepção de mensagens na comunicação BrokerClient -> UpaBroker Principal. Foram definidos 2000 *ms* para timeout conexão e 4000 *ms* para timeout de recepção de mensagem.
- Foi implementado um **TimerTask** em UpaBroker Principal com um intervalo cíclico de 5000 *ms* para o envio de provas de vida.
- Foi implementado um **TimerTask** em UpaBroker Secundário com um intervalo cíclico de 5000 *ms* para a verificação de provas de vida enviadas por Upa Broker Principal.
- Todas as operações realizadas por UpaBroker Principal que mudam o estado de “interesse comum” dos dois brokers, passam a actualizar imediatamente o estado de UpaBroker Secundário.

Este Sistema de tolerância a faltas com replicação passiva, tolera **no máximo uma falta** e assume que esta é **silenciosa**. Funciona da seguinte maneira:

UpaBroker Principal e Secundário começam por registar os seus serviços no servidor de nomes com identificadores diferentes. BrokerClient utiliza o endereço de UpaBroker Principal para realizar os seus pedidos (obtido através do servidor de nomes). No decorrer dos seus pedidos, caso aconteça algum *timeout* (tanto de conexão como de recepção de mensagem), é lançada uma *WebServiceException* que faz com que o Cliente vá ao servidor de nomes novamente pedir o endereço do serviço “UpaBroker”. Entretanto, no decorrer das operações entre BrokerClient -> UpaBrokerPrincipal, o Broker principal foi enviando mensagens de *update* de estado e de provas de vida ao broker secundário. Caso este último não as receba por um determinado período de tempo, auto proclama-se como primário no Servidor de Nomes, cancela o seu *TimerTask* e ignora as operações de actualização de estado, passando a contactar os serviços Transporter em resposta aos pedidos do Cliente.