UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Krishnan Lage Pontes

PROPOSTA DE UM MODELO DE QUALIDADE DE SERVIÇO E SEGURANÇA PARA A TECNOLOGIA DE *WEB SERVICES*

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de mestre em Ciência da Computação.

Prof. Carlos Westphall, Dr.
Orientador

westphal@lrg.ufsc.br

Florianópolis, Março de 2003

PROPOSTA DE UM MODELO DE QUALIDADE DE SERVIÇO E SEGURANÇA PARA A TECNOLOGIA DE *WEB SERVICES*

Krishnan Lage Pontes

Esta Dissertação foi julgada adequada para a obtenção do título de mestre em Ciência da Computação, área de concentração Sistemas de Computação, e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____
Prof. Fernando Ostuni Gauthier, Dr.
Coordenador do Curso

Banca Examinadora

_____
Prof. Carlos Westphall, Dr.
Orientador

_____
Prof. Paulo Freitas, Dr.

_____
Prof. Ricardo Felipe Custódio, Dr.

_____
Prof. Carla Merkle Westphall, Dra.

Ofereço este trabalho a meus pais, que sempre tiveram em minha educação, e na de meus irmãos, o mais alto grau de dedicação e prioridade.

AGRADECIMENTOS

A meu orientador, Professor Carlos Westphall, por todo o direcionamento, tranqüilidade, e metodologia empreendidos ao longo do curso e deste trabalho.

À Professora Carla Merkle Westphall, que muito contribuiu e esclareceu sobre importantes tópicos de sistema distribuídos.

À minha namorada, Paula, que tem sido a pessoa mais presente a meu lado, fazendo críticas inteligentes, e me dando suporte emocional.

A Deus, por esta oportunidade maravilhosa e única.

ÍNDICE

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

API – *Application Programming Interface*

ASP – *Active Server Pages*

ATM – *Assyncronous Transfer Mode*

BTP – *Business Transaction Protocol*

CORBA – *Common Object Request Broker Arquitecture*

DBMS – *DataBase Management System*

DCOM – *Distributed Component Object Model*

DLL – *Dynamic Link Library*

DSA - *Digital Signature Algorithm*

DTD – *Document Type Definition*

ECF – Emissor de Cupom Fiscal

HTTP – *Hyper Text Transfer Protocol*

IETF – *Internet Engineering Task* Force

IP – Internet Protocol

MTU - *Maximum Transmission Unit*

OASIS - *Organization for the Advancement of Structured Information Standards*

QoS – *Quality of Service*

RPC – *Remote Procedure Call*

RSA - *Rivest, Shamir e Adleman*

RSVP – *ReSource Reservation Protocol*

SIAC – Sistema de Automação Comercial

SLA – *Service Lavel Agreement*

SOAP – *Simple Object Access Protocol*

SSL – *Secure Sockets Layer*

TCP – *Transport Control Protocol*

TEF – Transferência Eletrônica de Fundos

UDDI - *Universal Description, Discovery and Integration*

W3C – *World Wide Web Consortiou*

WSDL – *Web Services Description Language*

WSRP – *Web Services Routing Protocol*

X509 – Padrão para certificado de chave pública

XKMS – XML *Key Management Specification*

XML – *eXtensible Markup Language*

XML Dsig – XML *Digital Signature*

RESUMO

Este trabalho apresenta um modelo para avaliação de *Web Services* em relação à utilização de aspectos de Qualidade de Serviço (QoS) e Segurança, e à aderência aos padrões XML (*eXtensible Markup Language*) de comunicação definidos para o tratamento destes aspectos.

São apresentados os conceitos, e protocolos que fazem parte da tecnologia de *web services*, e feitas comparações com outras tecnologias de propósito semelhante, como CORBA e DCOM.

É apresentado um estudo sobre o BTP *(Business Transaction Protocol)*, protocolo que visa prover integridade transacional aos serviços.

É implementada uma proposta que trata o aspecto de desempenho em *web services*, com as requisições SOAP sendo priorizadas por um "Filtro de QoS", e o controle de banda feito por RSVP *(ReSource Reservation Protocol)*.

Aspectos de segurança são abordados no sentido de fazer um estudo dos recém criados padrões XML para troca de informações seguras, com criptografia e assinatura digital, esclarecer sua utilização, e avaliar, com testes, seu impacto sobre o desempenho do serviço.

Aspectos de segurança, desempenho, e integridade transacional, devem poder ser avaliados no modelo, a fim de mostrar que a tecnologia de *web services* pode constituir-se em uma proposta alternativa e vantajosa ao modelo atual de Transferência Eletrônica de Fundos (TEF), no qual existe a integração de Sistemas de Automação Comercial (SIAC) e Operadoras de Cartão de Crédito.

ABSTRACT

This work presents a framework for evaluating web services regarding Quality of Service and Security, and its adhesion to related XML standards.

It presents the base concepts and protocols on web services technology, and compares web services with CORBA and DCOM, technologies with the same purpose.

It also presents a review on BTP, a protocol for keeping integrity on web service transactions.

It is implemented a proposition on Performance aspect of QoS, with SOAP requests beeing priorized through a “QoS Filter”, and band control beeing done by RSVP (ReSource Reservation Protocol).

Security issues are studied on the new XML standards for providing confidencialty and digital signature on web services, and evaluate, with tests, their impact on service performance.

Secutity, Performance, and Transaction Integrity, must be able to be verified on the framework, to show that web services can be an alternative and profitable technology for current TEF (*Transferência Eletrônica de Fundos*) model.

1 INTRODUÇÃO

O comércio apresenta, anualmente, uma tendência de crescimento dos meios de pagamento eletrônicos em detrimento dos tradicionais pagamentos em dinheiro ou cheque. O governo, a fim de aumentar sua receita com impostos e facilitar o trabalho da fiscalização, aprovou leis que obrigam as transações de cartão de crédito ou débito a serem integradas com os sistemas de automação comercial (SIAC), que por sua vez, devem ser integrados com uma impressora fiscal.

Esta exigência da fiscalização implicou em um grande esforço das operadoras de cartão no sentido de disponibilizar uma plataforma simples de integração com os diversos SIAC, para os mais diferentes tipos de comércio.

A fiscalização de Santa Catarina vem se destacando como a mais severa a nível nacional no cumprimento desta lei.

Embora o ambiente Windows seja dominante como plataforma da maioria dos SIAC, é desejável, senão imprescindível, em médio prazo, que o SIAC possa utilizar qualquer Sistema Operacional.

A fim de garantir a qualidade de serviço das transações eletrônicas, diferentes requisitos de qualidade de serviço (QoS) e segurança devem ser atendidos para que o sucesso da integração com o SIAC não venha a constituir uma ameaça à segurança da operadora de cartão ou a seus clientes, usuários do cartão. Qualidade de Serviço é um conceito bastante amplo e este trabalho propõe um modelo de avaliação abrangendo alguns aspectos de QoS: desempenho, integridade transacional, e segurança.

1.1 Motivação

Com o crescimento das transações eletrônicas e da Transferência Eletrônica de Fundos (TEF), e, paralelamente, o surgimento da tecnologia de *web services*, pode-se vislumbrar para um futuro não muito distante, uma plataforma de integração entre Sistemas de retaguarda de Operadoras de Cartão e Sistemas de Automação Comercial (SIAC), baseados no protocolo SOAP.

No entanto, alguns aspectos de Segurança e Qualidade de Serviço em *Web Services* ainda se encontram em fase embrionária de definição e de utilização.

Eventuais falhas de segurança ou desempenho das operações, no estabelecimento comercial, ao longo do processo de TEF, podem resultar em danos irreparáveis na credibilidade e nas contas das operadoras de cartão, dos estabelecimentos comerciais que utilizam o sistema, das empresas de software desenvolvedoras dos SIAC e dos usuários de cartão de crédito ou débito. Portanto, o desempenho e a confiabilidade de sistemas de TEF são decisivos para a credibilidade das instituições e a segurança dos usuários, e devem ser tratados em qualquer modelo TEF que venha a se tornar o padrão de mercado.

1.2 Problemas do Modelo TEF

A fiscalização do governo, com o objetivo de evitar a sonegação de impostos, determina, com a lei federal 9532/97, a implantação de máquinas Emissoras de Cupom Fiscal (ECF), em substituição às antigas caixas registradoras. O cupom fiscal, desde que revestido das formalidades legais exigidas pelo fisco, substitui a nota fiscal para vendas ao consumidor [WAR 03].

A lei 9532/97 não discrimina espécie de pagamento, sendo assim obrigatório a impressão do comprovante de venda na ECF mesmo para pagamentos em cartão de crédito ou débito. Alguns exemplos de leis estaduais mostram que o modelo de Transferência Eletrônica de Fundos integrada ao software de automação do comércio é o que deseja a fiscalização, como as exemplificadas no quadro 1.1:

PORTARIA Nº 336, DE 6 DE JUNHO DE 2002 (http://www.sefp.df.gov.br/Legislacao/ecf/Port336.htm)
LEI Nº 7.601 DE 11 DE JUNHO DE 2001 (DOE 21.06.01)

Quadro 1.1 - Leis estaduais estabelecendo regras para o comércio eletrônico.

Estabelece prazos para implementação da impressão do comprovante de pagamento com uso de Transferência Eletrônica de Fundos (TEF) no Emissor de Cupom Fiscal (ECF) e dá outras providências.

A solução mais adotada nos dias de hoje para integração de aplicativos comerciais com sistemas de cartão de crédito ou débito, é baseada em troca de arquivos entre as partes da comunicação, com informações dentro de um layout pré-definido pela operadora de cartão. Esta troca de arquivos ocorre em um nível local, na própria estação do PDV (Ponto de Venda), entre o SIAC (Sistema de Automação Comercial) e o software Gerenciador TEF.

Uma transação de venda normal segue o seguinte fluxo de procedimentos:

1. Venda é registrada no SIAC até o ponto de escolha da espécie de pagamento;
2. Ao escolher a espécie cartão, o SIAC cria um arquivo, no layout da operadora, com um chamado para o Gerenciador TEF e passando os parâmetros necessários à consolidação da transação de venda;
3. O Gerenciador TEF permite ao cliente passar o cartão no Pin Pad (dispositivo para leitura de cartão e digitação de senha, conectado a uma porta serial do computador) e conecta com a operadora para transferir as informações e receber uma autorização ou uma negação para a transação solicitada;
4. O Gerenciador TEF devolve ao SIAC um arquivo, segundo layout da operadora, contendo os detalhes da autorização;
5. Se a transação não foi autorizada, o SIAC informa a mensagem ao usuário;

6. Se a transação foi autorizada, o SIAC termina seus procedimentos e a impressão do Cupom Fiscal e do Cupom não Fiscal vinculado à transação;
7. Se tudo ocorreu bem, o SIAC cria um arquivo de confirmação para o Gerenciador TEF, caso contrário, cria um arquivo de não confirmação da transação para o gerenciador TEF;
8. Quando ocorrer uma próxima transação, o Gerenciador TEF envia para a operadora o status final da transação anterior.

A solução atual possui méritos, dentre os quais podemos destacar a simplicidade de integração com o SIAC. Esta é a solução hoje adotada pelo BanriSul, Visa, RedeCard e AMEX. Nesta solução são atendidos requisitos de confidencialidade e integridade pois o software Gerenciador TEF pode cifrar e gerar resumos das mensagens trocadas com o servidor central da operadora.

Alguns problemas encontrados nesta solução são:

- Não é definido um modelo de autenticação do PDV;
- Falta uma proposta de Não Repudiação;
- O Gerenciador TEF é um *software* que deve ser instalado em cada PDV por toda a rede de lojas credenciadas com a operadora. Isto gera uma grande necessidade de manutenção e atualizações *on site*, a ponto de as operadoras terem que terceirizar este serviço para outras empresas (Instaladoras) nas regiões de abrangência;
- Atualmente o Gerenciador TEF só é encontrado para ambiente Windows. Esta ainda é a plataforma dominante, porém o advento de novas linguagens de desenvolvimento RAD (Rapid Application Development) para LINUX, como o Kylix, somente tende a acelerar a expansão do uso deste sistema operacional no comércio.

Com base nos problemas apresentados nesta seção, pode-se concluir que seria altamente desejável a possibilidade de integração do comércio eletrônico de TEF por meio de *web services*. No entanto, precisamos nos certificar de que esta tecnologia atenderá a todos os requisitos de segurança e QoS necessários a tal implementação.

1.3 Objetivos

O objetivo geral deste trabalho é propor um modelo de QoS na utilização de *Web Services*. Este modelo contempla aspectos de desempenho, integridade transacional, e segurança, a fim garantir a qualidade de serviço necessária ao comércio eletrônico baseado em *web services*.

Podemos destacar ainda os seguintes objetivos específicos, relacionados ao tema principal:

1. Estudar as tecnologias de segurança de dados, criptografia e protocolos criptográficos hoje utilizados em transações eletrônicas, com ênfase no SSL;
2. Mostrar como SSL e as estruturas padrão XML para assinatura digital e criptografia sobre requisições SOAP podem se complementar.
3. Identificar requisitos de Qualidade de Serviço em *web services* necessários para o desenvolvimento de um novo modelo para TEF;
4. Identificar as diferenças e vantagens do *Web Services* em relação a outros modelos de integração de computação distribuída, como CORBA e DCOM;
5. Propor soluções para aspectos de desempenho em *Web Services*, com implementação, e validação com testes, de um serviço *web* com filas de prioridade e com estabelecimento de conexão RSVP;
6. Avaliar, com implementação e testes, o impacto da utilização de SSL e de XML *Digital Signature* no desempenho dos serviços;
7. Mostrar, no protocolo BTP (*Business Transaction Protocol*), como podem ser tratados aspectos de segurança.

1.4 Trabalhos Correlatos

A especificação do protocolo SOAP, base da tecnologia de *Web Services*, encontrada no sítio http://www.w3.org/TR/SOAP, não determina extensões para as áreas de segurança e QoS. Padrões estão em fase de definição por organismos como W3C, IEEE e OASIS.

[BAN 97] procura classificar um perfil do usuário baseado em seu histórico de acesso, a fim de determinar o nível de serviço a que ele terá direito no servidor *web*. Neste caso, as regras para definição de prioridades estão todas no servidor *Web*.

[CHA 00] mostra que se pode atender requisitos de QoS, não somente pela classificação e priorização do usuário, mas também, pela transcodificação de conteúdo multimídia a ser trafegado na rede.

[RON 99] define uma proposta em nível de corporação para *Security Service Level Agreement* (SLA).

O projeto BTP (Business Transaction Protocol), cuja primeira implementação é o Hewlett-Packard Web Services Transactions (HP-WST) [KAL 02], é um excelente projeto, porém aborda somente o aspecto de integridade transacional.

No aspecto de segurança, [EVA 02] discute se o SSL é capaz de tratar os diversos requisitos de segurança necessários a um *web service* entre dois nós SOAP.

[DAM 02] apresenta uma proposta de controle de acesso e autorização baseado em papéis para web services.

1.5 Organização do Texto

O texto, dividido em capítulos, está organizado da seguinte maneira:

No capítulo 2 são apresentados conceitos sobre sistemas distribuídos, é feita uma comparação entre as soluções Web Services, CORBA e DCOM, e a seguir, são apresentados conceitos de segurança no comércio eletrônico e os novos padrões XML para tratar segurança em *web services*.

O capítulo 3 mostra problemas de QoS que devem ser contemplados na tecnologia *web services*.

No capítulo 4 é apresentado o modelo proposto por este trabalho e detalhada a proposta para o aspecto de desempenho em *web services*.

O capítulo 5 mostra o ambiente de desenvolvimento, a implementação, e os resultados dos testes realizados do módulo de desempenho.

O capítulo 6 apresenta a conclusão do trabalho, comentando sobre as considerações finais, comparando com trabalhos correlatos, e destacando as contribuições científicas e os trabalhos futuros.

CAPITULO 2
REVISÃO BIBLIOGRÁFICA

2.1 Serviços Distribuídos

2.1.1 Breve Histórico

A indústria de software deseja, há muito tempo, uma linguagem que Windows, Unix, Macintosh e mainframes possam utilizar para comunicarem entre si pela internet. Empresas precisam disponibilizar suas aplicações para parceiros, fornecedores e clientes. Protocolos como o HTTP permitem apenas tarefas simples como mostrar um hipertexto requerido pelo usuário. Mas, e se você desejar ir a um nível mais alto e permitir iteração entre duas aplicações distintas rodando em diferentes plataformas?

Aplicações distribuídas exigem um protocolo que defina um mecanismo de comunicação entre dois processos concorrentes.

Nos anos 80, modelos de protocolos de comunicação enfocavam no nível de rede, como o NFS (*Network File System*) que permitia utilização do sistema de arquivos de máquinas UNIX via rede. Já nos anos 90, a programação orientada a objetos determinou a tendência aos protocolos ORPC (*Object Remote Procedure Call*) para ligar aplicações a protocolos de rede. Microsoft DCOM, CORBA, Internet Inter-ORB Protocol (IIOP) eram protocolos dominantes nesta década.

Acreditar em padrões largamente adotados é apenas uma parte da solução. É necessário também ter certeza de que a solução oferece um alto nível de interoperabilidade e que implementações do protocolo estejam facilmente disponíveis. CORBA e DCOM possuem deficiências que serão abordadas na seção 2.1.4. Gisolfi [GIS 01] ressalta que a necessidade de um novo modelo para computação distribuída, com interoperabilidade comprovada e simplicidade para os programadores desenvolverem os lados cliente e servidor das aplicações utilizando o protocolo, indicam a direção de um modelo baseado nos padrões abertos da internet.

Existindo uma linguagem tão universal como o XML, o mais natural foi o surgimento de SOAP.

2.1.2 Simple Objects Access Protocol (SOAP)

SOAP é um protocolo, baseado em XML, para troca de informações em ambientes distribuídos [BOX 01]. SOAP é simples, extensível, independente de plataforma, e feito em XML, uma linguagem largamente difundida e orientada para definição de conteúdo. SOAP consiste em três partes:

- Um envelope que define um modelo padrão para descrever o conteúdo de uma mensagem e como processa-la. É o elemento XML de mais alto nível, representando a mensagem;
- Um conjunto de regras (Cabeçalho ou *Header*) para expressar instâncias de tipos de dados definidos por uma aplicação, de uma maneira descentralizada e sem acordo prévio entre as partes que estarão em comunicação. No *Header* são definidos características e atributos da mensagem. SOAP também define alguns atributos que podem ser utilizados para definir quem deve usar uma determinada característica da mensagem e se isto é opcional ou obrigatório. Um exemplo de atributo deste tipo é o <mustUnderstand>, que significa que aquele elemento XML deve ser, obrigatoriamente , entendido, conforme exemplifica o quadro 2.1.
- Uma convenção para representar chamadas a procedimentos remotos (RPC) e respostas. O Corpo (*Body*) é um elemento XML cujo conteúdo são informações obrigatórias para o destino da mensagem.

O elemento <*Body*> é codificado como um filho imediato do elemento XML SOAP Envelope. Se existe um elemento <*Header*>, então o elemento <*Body*> deve ser imediatamente subseqüente ao elemento <*Header*>. Senão ele deve ser um filho imediatamente subseqüente ao elemento <Envelope>. Todos os elementos filhos imediatos do elemento <*Body*> são chamados entradas *Body* e cada entrada é codificada como um elemento independente dentro do elemento <*Body*>.

A codificação de tipos SOAP é baseada em um sistema simples que é uma generalização das características mais comuns encontradas em linguagens de programação ou bancos de dados. Existem tipos simples (escalares) e tipos compostos, construídos como uma composição de diversos tipos simples e/ou compostos;

O SOAP pode, potencialmente, ser encapsulado para ser transportado por diversos protocolos. No entanto, a ligação mais utilizada tem sido com o HTTP, que é um protocolo cuja porta TCP normalmente não é filtrada em *firewalls*.

SOAP define um modelo para codificar mensagens (requisições e respostas), mas não define regras de programação ou semântica de implementação específica para os serviços.

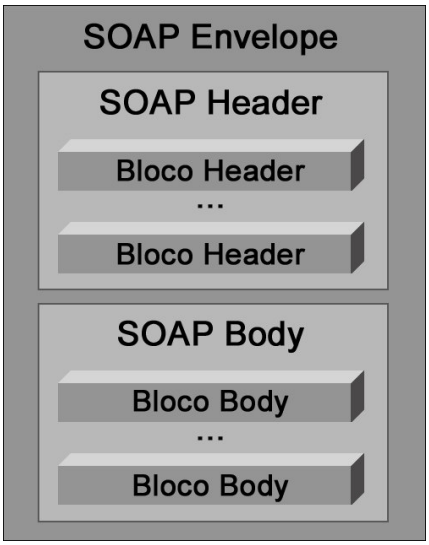

Figura 2.1 – Blocos de uma mensagem SOAP.

O quadro 2.1 mostra um exemplo de requisição SOAP sobre HTTP. Neste exemplo, uma requisição a *ConsultaSaldo* é enviada para um serviço. A requisição tem um código como parâmetro *string*, e retorna um *Float* na resposta SOAP. O elemento SOAP Envelope é o elemento do topo do documento XML que representa a mensagem SOAP. O exemplo mostra um *header* com um elemento identificado por "Transação" com valor “5” e um

"*mustUnderstand*" value de "1". *NameSpaces* XML são utilizados para não haver ambigüidades entre identificadores SOAP e identificadores específicos da aplicação. Este exemplo ilustra uma amarração do SOAP com o HTTP. A diretiva SOAPAction no cabeçalho HTTP indica que é uma mensagem SOAP. É importante notar que as regras que governam o formato do conteúdo XML no SOAP, são independentes do protocolo de camada inferior, neste caso, o HTTP.

```
POST /BuscaSaldo HTTP/1.1
Host: www.operadora.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "www.operadora.com/ConsultaSaldo"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Header>
       <t:Transacao
           xmlns:t="www.operadora.com/transacao"
           SOAP-ENV:mustUnderstand="1">
               5
       </t:Transacao>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
       <m:ConsultaSaldo xmlns:m="www.operadora.com/ConsultaSaldo">
           <m:cartao>1234567890123456</m:cartao>
       </m: ConsultaSaldo>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Quadro 2.1 - requisição SOAP

Maiores detalhes sobre o protocolo SOAP podem ser encontrados em http://www.w3.org/TR/SOAP .

2.1.3 Web Services

Segundo Darakhvelidze [DAR 02], *Web Service* é uma aplicação servidora disponível na *internet*, cuja interface é desenvolvida e publicada de acordo com o padrão SOAP.

A linguagem utilizada para descrever um web service é chamada *Web Services Description Language* (WSDL).

Com a crescente demanda por transações eletrônicas e aplicações sobre *internet* ou *intranets*, integrar *web services* de diversos web servers se torna essencial. Como a maioria dos web services entrega seus serviços aos usuários usando HTTP, e este é um protocolo sem marcação de estado, criado com o objetivo de apenas mostrar informações, se faz necessário um protocolo de padrões mais abertos. Ele deve permitir a troca de informações entre *web servers* independente de suas plataformas de sistema operacional, modelo de objetos, ou linguagens de programação. SOAP provê um mecanismo padrão baseado em XML para integração de web services através de HTTP ou outro protocolo padrão da internet, que possibilita a comunicação entre aplicações.

Web Services possibilita a aplicações, fazerem requisições e utilizarem serviços de outras aplicações, sem preocupação de como ou em que plataforma o serviço foi desenvolvido; um mecanismo de transporte para enviar as requisições, e a capacidade de receber resultados. Por serviços disponibilizados por outras aplicações, entenda-se, métodos de objetos disponibilizados pelas mesmas.

Para que este novo paradigma possa ser efetivamente utilizado, diversas partes precisam trabalhar em conjunto:

1. Ambiente de desenvolvimento de aplicações depende de uma capacidade de descoberta de serviços que permita aos desenvolvedores encontrarem os serviços disponíveis. UDDI é uma tecnologia com este objetivo;
2. Um serviço de endereço a fim de manter as aplicações já desenvolvidas atualizadas com eventuais mudanças no endereço dos serviços utilizados, e com eventuais atualizações dos tipos e serviços disponíveis. WSDL trata esta finalidade;
3. Segurança, com autenticação e integridade dos serviços disponíveis;

2.1.4 Comparando *Web Services*, CORBA e DCOM

A arquitetura CORBA *(Common Object Request Broker*) é uma especificação do *Object Management Group* com o intuito de prover interoperabilidade entre computadores

distribuídos. Seu objetivo era definir uma arquitetura que permitisse a plataformas heterogêneas comunicarem-se em nível de objeto, independente de quem programou os pontos finais da aplicação distribuída.

Um ORB recebe uma mensagem invocando um método específico de um objeto registrado. O ORB intercepta a mensagem e procura o objeto da requisição. Passa os parâmetros para o método chamado e retorna os resultados. Em teoria, o nó que fez a requisição não precisa saber onde localizar o objeto, nem sua plataforma ou quaisquer aspectos que não sejam parte da interface do objeto.

DCOM é uma extensão distribuída do Microsoft COM (*Component Object Model*). Um cliente COM interage com um objeto COM apontando para uma das interfaces do objeto e invocando métodos através daquele ponteiro, como se o objeto residisse no espaço de memória do cliente.

A arquitetura básica RPC (*Remote Procedure Call*) em ambos, CORBA e DCOM, para iteração entre um processo cliente e um objeto servidor, são implementadas como uma comunicação RPC orientada a objeto. A figura 2.2 mostra a típica estrutura. Para invocar uma função remota, o cliente faz uma chamada para um módulo cliente, chamado *Stub*. O *Stub* empacota os parâmetros da chamada em uma mensagem de requisição e a entrega ao protocolo de transporte para entrega-la ao servidor. No lado do servidor, uma camada de software chamada Server Stub recebe a mensagem do protocolo de transporte, desempacota a requisição da mensagem e chama a função pedida no objeto. Em DCOM os dados da requisição são escritos em um formato chamado DR (*Data Representation*). Em CORBA os dados são escritos em CDR (*Common Data Representation*). Pequenas diferenças entre os dois formatos os tornam incompatíveis.

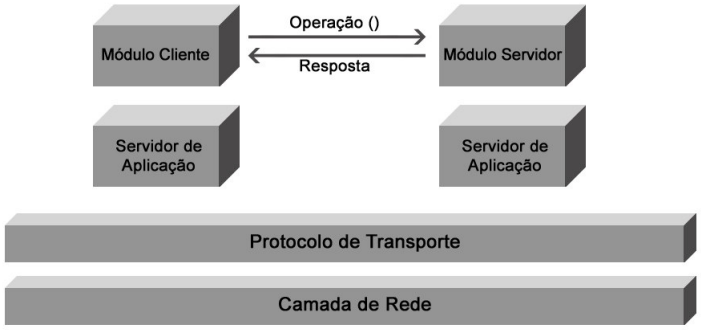

Figura 2.2 – Arquitetura RPC no DCOM e CORBA. Na realidade os módulos *Stub* cliente e servidor possuem nomenclatura diferente em ambas as arquiteturas.

E porque o sucesso de CORBA e DCOM foi limitado?

Porque, embora tenham sido implementados em diversas plataformas, na realidade, as soluções, sempre dependem de um único fabricante. Para uma aplicação DCOM todos os nós participantes devem ser alguma versão de Windows. No caso de CORBA, cada nó participante da comunicação deveria executar o mesmo produto ORB do mesmo fabricante. Interoperabilidade entre ORBs de diferentes fabricantes existe, porém em um nível limitado. Em um ambiente como este cada fabricante tenta vender sua solução em detrimento das outras. Ambos os protocolos não interagem entre si. Além disso, programadores têm que lidar com regras do protocolo de formatação das mensagens, alinhamento e tipos de dados.

Estas deficiências são a grande dificuldade para a comunicação cliente-servidor, especialmente quando máquinas clientes estão espalhadas pela *internet*.

A tecnologia Web Services é baseada em padrões já consagrados na internet e em especificações abertas que tendem a se tornar padrões. A pilha básica é composta por HTTP, XML, SOAP, WSDL, e UDDI.

Na base da pilha encontra-se o HTTP, um protocolo largamente utilizado e depurado, apropriado para RPC, e que passa sem problemas por firewalls.

HTTP é um modo elegante de carregar conteúdo (leia-se SOAP) na tecnologia web services. Seus cabeçalhos são texto plano, o que facilita o acesso à maioria dos programadores. HTTP utiliza a infra-estrutura TCP/IP e suporta modelos de comunicação

baseados em requisição/resposta [COM 00]. Ele também faz uso de URLs para referência a objetos, o que coincide com IORs e OBJREFs, encontrados em CORBA e DCOM, respectivamente.

A seguir tem-se uma linguagem simples, independente de plataforma e largamente adotada para representação de dados, como o XML, que é base de todos os outros componentes da pilha.

SOAP é um protocolo de codificação de mensagens baseado em XML, e, portanto, também independente de linguagem ou plataforma. Ele suporta simples envio de mensagens ou modelos de comunicação baseados em requisição/resposta. Como CORBA e DCOM, SOAP também precisa de uma IDL (*Interface Definition Language*). WSDL é um serviço IDL baseado em XML que define a interface do serviço bem como características de implementação. No entanto, ao contrário de DR e CDR, XML é simples de usar e oferece um formato de dados extensível, simples e flexível. Além de ser largamente adotado na maioria das plataformas.

Um *endpoint* SOAP é apenas uma URL HTTP que identifica um método a ser invocado. Como CORBA, o objeto não precisa estar ligado ao *endpoint*. O programador do serviço decide como mapear o identificador do *endpoint* ao objeto no servidor. O *namespace* da URL que chama um método em SOAP é equivalente ao ID da interface que referencia um método em DCOM e CORBA.

Tendo apresentado similaridades e diferenças nas tecnologias, alguns pontos podem resumir o porquê acreditar no sucesso da tecnologia *Web Services* em detrimento de SOAP ou DCOM.

- Utiliza HTTP, que utiliza portas TCP não filtradas por firewalls, além de possuir área de dados de tamanho variável;
- Emprega XML, um esquema de codificação mais difundido do que DR e CDR;
- Ambiente de servidor gratuito, baseado em HTTP/SOAP, ao contrário dos modelos ORB, comercializados por diversos fabricantes;
- Utiliza o conceito de URLs para endereçar e identificar objetos;

- Oferece mais do que uma simples promessa de interoperabilidade. Fabricantes de software estão trabalhando fortemente no sentido de provar que suas implementações são compatíveis com o padrão e se comunicam entre si.

2.2 Segurança no Comércio Eletrônico

O comércio eletrônico e a comunicação, em ambiente de rede, podem ser alvos de diferentes tipos de ataque. Segundo [STA 00], os ataques podem ser classificados em ativos ou passivos. Ataques ativos são divididos em quatro categorias: personificação, repetição, modificação e negação de serviço. Ataques passivos são divididos em interceptação e análise de tráfego. Em contra-partida a estes ataques [STA 00] define os requisitos de segurança abaixo:

Confidencialidade – consiste na proteção dos dados transmitidos de ataques passivos. Garante aos participantes da comunicação que somente eles terão conhecimento do conteúdo das informações que estão sendo trocadas;

Autenticação – propriedade que assegura que os participantes da comunicação são, de fato, quem eles dizem ser. Impede que algum atacante possa se fazer passar por algum participante legítimo da comunicação;

Integridade – assegura aos participantes da comunicação que as mensagens trocadas entre eles chegam ao destinatário exatamente como foram enviadas, sem modificação, reordenação, resequenciamento ou duplicação;

Não repudiação – previne que ambos, o emissor ou o recipiente, possam negar que uma mensagem foi emitida ou recebida em uma comunicação;

Controle de acesso – é a propriedade de limitar e controlar acesso aos recursos da rede através dos meios de comunicação. Cada usuário ou programa deve ser primeiro identificado e autenticado a fim de receber os recursos de rede e software pertinentes;

Disponibilidade – assegura a disponibilidade dos recursos de rede. Caracteriza-se por técnicas para evitar falhas no sistema e ataques de negação de serviço;

2.2.1 Assinatura Digital

Segundo Ghisleri [GHI 02], a assinatura digital objetiva, para o meio eletrônico, o mesmo propósito que a assinatura em papel: garantir que uma mensagem recebida por um recipiente foi realmente enviada por um emissor A de forma tal que o emissor A não possa negar que enviou a mensagem, e não exista maneira de um atacante falsificar uma mensagem se fazendo passar pelo emissor A..

A assinatura digital deve possuir as seguintes propriedades:

- Pode-se verificar o autor e a data e hora da assinatura;
- Pode-se autenticar o conteúdo na hora da assinatura;
- A assinatura é verificável por terceiros para resolver disputas;

A assinatura digital emprega o certificado digital do emissor da mensagem e uma função de resumo, e consiste em:

Emissor

- gerar um resumo do texto plano P a ser enviado;
- cifrar este resumo com a chave privada do emissor, gerando um resultado R;
- concatenar o resultado R com o texto plano P;
- enviar ambos para o recipiente

Recipiente

- decifrar o resultado R com a chave pública do emissor e comparar o resultado com um novo resumo gerado para o texto plano P recebido;
- se os resultados forem iguais significa que a mensagem não foi alterada durante o envio, o que garante a sua integridade;

- se os resultados forem iguais significa que a assinatura do emissor está garantida pois somente ele poderia cifrar uma mensagem com sua chave privada;

A adição de números aleatórios ou datas às mensagens trocadas serve para impedir ataques de repetição.

O RSA (Rivest, Shamir e Adleman) e o DSS (Digital Signature Standard) são exemplos de algoritmos de assinatura digital e são abordados em detalhes em [STA 99]

2.2.2 Secure Sockets Layer (SSL)

O SSL é um protocolo que funciona em uma camada acima do TCP (*Transmission Control Protocol*) e abaixo de protocolos de alto nível em camada de aplicação, como o HTTP (*Hyper Text Transport Protocol*) ou o IMAP (*Internet Messaging Access Protocol*). O objetivo do SSL é mutuamente autenticar cliente e servidor com a finalidade de abrir um canal de comunicação segura entre ambos [ALA 96]. A figura 2.3 mostra a inserção do SSL na pilha de protocolos.

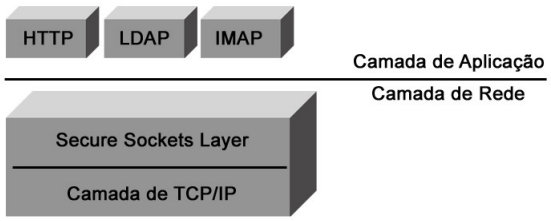

Figura 2.3 – SSL na pilha de protocolos.

Na realidade, para troca de informações seguras entre um navegador e um servidor Web, é preciso que ambos sejam habilitados a usar SSL. O mesmo acontece, por exemplo, entre um cliente e um servidor de e-mail que queiram habilitar troca de mensagens cifradas.

O SSL inclui dois sub protocolos: o SSL *Record protocol* e o SSL *handshake protocol*. O SSL Record protocol define formatos para a transferência dos dados. O SSL handshake protocol envolve utilizar o SSL *record protocol* para trocar uma série de mensagens entre um servidor habilitado com SSL e um cliente habilitado com SSL. Esta troca de mensagens é para:

- Autenticar o servidor junto ao cliente;
- Permitir ao cliente e ao servidor selecionar algoritmos criptográficos que ambos suportem;
- Opcionalmente autenticar o cliente junto ao servidor;
- Utilizar técnicas de critografia de chave pública para gerar chaves secretas;
- Estabelecer uma conexão SSL segura;

De acordo com Schneier [SCH 00], decisões sobre que algoritmos aplicar levam em consideração o grau de segurança necessário aos dados envolvidos, a velocidade do algoritmo e aspectos legais de exportação de determinados algoritmos.

O protocolo SSL utiliza uma combinação de técnicas de criptografia simétrica e de criptografia de chave pública. Algoritmos de criptografia simétrica executam muito mais rápido, ao passo que criptografia de chave pública provê melhores técnicas de autenticação [SCH 00]. O *HandShake Protocol* permite ao cliente e ao servidor se autenticarem mutuamente utilizando criptografia de chave pública. Permite ainda que criem uma chave simétrica de sessão que será utilizada para cifrar e decifrar as mensagens da sessão subseqüente mais rapidamente.

Uma sucinta descrição dos passos envolvidos no *HandShake protocol* antes do estabelecimento de uma sessão SSL:

1. O cliente envia ao servidor sua versão SSL, configurações de cifras disponíveis e número randômico e outras informações;
2. O servidor responde ao cliente com as informações do passo 1 (pertinentes ao servidor) mais o seu certificado digital;

3. O cliente valida o certificado digital do servidor (figura 2.4) com sua lista de Autoridades Certificadoras confiáveis;
4. Caso o certificado digital do servidor seja aceito, o cliente gera uma pré-chave secreta e a envia para o servidor, cifrada com a chave pública do servidor. Neste ponto, se tiver sido requerido, o cliente também envia seu certificado digital para o servidor;
5. Se necessário o servidor valida o certificado digital do cliente. A seguir, o servidor usa sua chave privada para decifrar a pré-chave secreta e, a partir desta, executa uma série de passos (que também são executados pelo cliente) a fim de gerar uma chave de sessão;
6. Ambos, cliente e servidor utilizarão a chave mestra para gerar chaves simétricas de sessão que serão utilizadas para cifrar e decifrar as mensagens trocadas durante a sessão e para verificar a integridade da informação, ou seja, detectar se houve alguma modificação nos dados entre o momento do envio e o momento do recebimento através da conexão SSL;

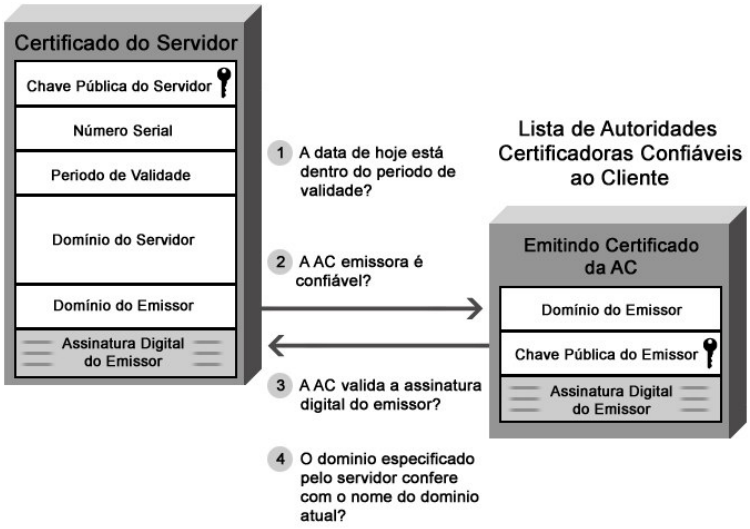

Figura 2.4 - mostra como um cliente habilitado com SSL autentica a identidade (certificado digital) de um servidor;

2.2.3 Segurança em Web Services

Sendo Web Services uma tecnologia baseada no protocolo SOAP, aspectos de segurança devem ser aplicados sobre este protocolo. O que tem sido feito, ao longo dos anos 2001 e 2002, é adaptar conceitos, protocolos, e algoritmos de segurança já consagrados no mercado, ao protocolo SOAP. Algumas tecnologias padrão XML foram recentemente definidas por órgãos como W3C e OASIS, bem como por empresas como IBM, Microsoft e RSA Security:

- XML *Digital Signature* (XML DSig) [BAR 02], é um padrão XML para troca de mensagens SOAP, com o objetivo de permitir a troca de informação necessária aos nós participantes de uma comunicação, visando conseguir assinatura digital, integridade, confidencialidade, e não repudiação na comunicação.
- XML *Encryption* [IMA 01], é um padrão XML para troca de mensagens SOAP, com o objetivo de permitir a troca de informação cifrada com criptografia simétrica e resumo de mensagens SOAP, aos nós participantes de uma comunicação.
- XML *Key Management Specification* (XKMS) [FOR 01], define um padrão XML para intercâmbio de mensagens SOAP visando a distribuição de chaves públicas, e certificados digitais, de maneira a serem usados em conjunto com XML *Digital Signature*
- SSL, que pode ser utilizado para transferência de SOAP sobre HTTP seguro.

É importante salientar que os padrões XML são apenas um protocolo para troca de informações necessárias para que se possa requisitar determinados aspectos de segurança a objetos nos servidores distribuídos. Os objetos responsáveis pela implementação, de fato, dos algoritmos criptográficos, devem estar implementados nos servidores *web*, e devem expor suas propriedades e métodos via WSDL, segundo o padrão XML pertinente.

2.2.4 XML *Digital Signature* e XML *Encryption*

XML Digital Signature [BAR 02] e XML Encryption [IMA 01] são padrões XML do W3C para permitir assinatura digital e criptografia em uma comunicação SOAP.

Com XML Encryption pode-se obter confidencialidade e integridade.

Com XML Digital Signature pode-se obter assinatura digital, com autenticação e integridade. XML Digital Signature possui a seguinte estrutura (onde ? significa zero ou uma ocorrência, + significa uma ou mais ocorrências, e * zero ou mais ocorrências) :

```
<Signature ID?>
     <SignedInfo>
       <CanonicalizationMethod/>
       <SignatureMethod/>
       (<Reference URI? >
         (<Transforms>)?
         <DigestMethod>
         <DigestValue>
       </Reference>)+
     </SignedInfo>
     <SignatureValue>
    (<KeyInfo>)?
    (<Object ID?>)*
</Signature>
```

Quadro 2.2 – Elementos básicos do XML-Digital Signature

Exemplo de código SOAP para uma requisição assinada digitalmente:

```
POST /order HTTP/1.1
Host: www.onlinetrade.com
Content-Type: text/xml; charset="UTF-8"
Content-Length: nnnn
SOAPAction: "http://www.operadora.com.br/pedido#compra"
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header>
        <SOAP-SEC:Signature xmlns:SOAP-
SEC="http://schemas.xmlsoap.org/soap/security/2000-12"        SOAP-
ENV:mustUnderstand="1">
          <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
               <ds:SignedInfo>
             <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-20001026"/>
```

```
            <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
            <ds:Reference URI="#Body">
                <ds:Transforms>
                <ds:Transform Algorithm="http://www.w3.org/TR/2000/CR-
xml-c14n-20001026"/>
                </ds:Transforms>
                <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
      <ds:SignatureValue>MC0CFFrVLtRlk=...</ds:SignatureValue>
      <ds:KeyInfo>
          <ds:KeyName>EDM Software LTDA</ds:KeyName>
             </ds:KeyInfo>
         </ds:Signature>
      </SOAP-SEC:Signature>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body xmlns:SOAP-
SEC="http://schemas.xmlsoap.org/soap/security/2000-12" SOAP-
SEC:id="Body">
    <ped:autorizacao xmlns:ped="http://www.operadora.com.br/autorizacao">
    <ped:nrterminal>123456</ped:nrterminal>
    <ped:nome>Leandro Pontes</ped:nome>
    <ped:nrcartao>8356264897263865</ped:nrcartao>
    <ped:expiracao>25062005</ped:expiracao>
    <ped:valor>85,00</ped:valor>
    </ped: autorizacao>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Quadro 2.3 – Requisição SOAP, seguindo padrão DSig, com pedido de autorização de venda.

No código do quadro 2.3 acima podemos verificar a assinatura digital no elemento <ds:SignatureValue>. Esta assinatura consiste em cifrar o hash da mensagem identificada pelo id=”Body”, com a chave privada de EDM Software LTDA, usuário identificado pelo elemento <ds:KeyInfo>. [BAR 02] detalha os namespaces [BRA 99] para cada algoritmo necessário à implementação no processamento da requisição acima. Neste caso, parte-se do princípio de que o recebedor da mensagem possui, ou tem como encontrar, a chave pública de EDM Software LTDA. Também deve-se observar que os elementos XML do pedido de autorização devem ser especificados em um namespace pela operadora de cartão em questão, neste exemplo, http://www.operadora.com.br/autorizacao.

É importante ressaltar que o elemento <ds:KeyInfo> também pode identificar a chave pública de forma não nominativa, mas sim por meio de um certificado X509.

O quadro 2.4 apresenta uma requisição que faz uso do padrão XML *Encryption*, e o elemento <EncryptedData> indica a referência de que é um elemento que está sendo cifrado.

O elemento CipherValue, contém o corpo da mensagem cifrado em sequências de octetos codificados Base64.

```
<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
      Type='http://www.w3.org/2001/04/xmlenc#Element'/>
  <EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#3des-cbc '/>
  <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
    <ds:KeyName>
      EDM Software LTDA<!—identifica o emissor da mensagem→
    </ds:KeyName>
  </ds:KeyInfo>
     <SessionId
xmlns:sid='http://lrg.ufsc.br/kpontes/wsmodelo#sessionid> 6542SD
     </SessionId><!—cifrado com a chave pública do recipiente→
  <CipherData><CipherValue>
     DEADBEEF…
  </CipherValue></CipherData>
</EncryptedData>
```

Quadro 2.4 – Cifrando o corpo da mensagem com criptografia simétrica.

2.3 Conclusão

Neste capítulo foram apresentados importantes conceitos sobre tecnologias para desenvolvimento de sistemas distribuídos, passando por CORBA, DCOM e *Web Services*. Em seguida foram abordados aspectos de segurança no comércio eletrônico, e apresentados o protocolo SSL e os novos padrões XML para tratamento de segurança em web services.

No próximo capítulo, serão apresentados conceitos de qualidade de serviço em *web services*, com ênfase nos aspectos desempenho e integridade transacional. Sobre o aspecto desempenho, serão levantados problemas, cujas soluções estão na proposta do capítulo 4.

CAPITULO 3
O MODELO ATUAL

3.1 Qualidade de Serviço (QoS)

A cada ano, mais empresas procuram efetivar contratos de nível de serviço com fornecedores e clientes. Gerenciamento de Nível de Serviço (SLM) é uma metodologia disciplinada e proativa de procedimentos que visam assegurar que os níveis adequados de serviço sejam prestados a todos os usuários de tecnologia da informação (TI), de acordo com as prioridades empresariais e a um custo razoável [STU 01].

Qualidade de Serviço é a capacidade de uma rede prover serviços diferenciados a tráfegos de rede selecionados de acordo com suas necessidades, sobre diversas tecnologias, incluindo *Frame Relay, Asynchronous Transfer Mode* (ATM), *Ethernet*, SONET, e, em uma camada de rede acima, redes roteadas por IP (*Internet Protocol*), que podem utilizar qualquer uma destas tecnologias como protocolo de nível inferior. O principal objetivo do QoS é prover prioridades, largura de banda dedicada, controle de atrasos e da freqüência dos pacotes, e garantia de entrega dos mesmos. Também deve ser certificado, de que priorizar um fluxo não implique na falha dos demais fluxos.

Ferramentas de QoS podem aliviar a maioria dos problemas de congestionamento de pacotes, porém algumas vezes simplesmente existe tráfego em demasia para a largura de banda fornecida. Neste caso QoS é somente um retalho mas não a solução para seu tráfego de rede. Uma possível solução para diminuição de tráfego é a transcodificação de conteúdo multimídia, que hoje, representa quase 70% do tráfego da internet [CHA 00].

Em outros casos, quedas em indicadores de desempenho podem estar inerentes aos serviços ou a protocolos utilizados, como, por exemplo, é apresentado em [KAN 00], que mostra que o uso do protocolo SSL incrementa o custo computacional das transações em um fator de 5/7.

Uma arquitetura básica introduz três partes fundamentais, como mostra a figura 3.1, para a implementação de QoS:

1. Técnicas de identificação e marcação do fluxo para possibilitar a coordenação de QoS entre os elementos de rede por onde o fluxo trafega;
2. Técnicas para garantir qualidade de serviço dentro de um mesmo elemento da rede. Por exemplo, enfileiramento e agendamento;
3. Funções de Gerência e Contabilização a fim de administrar o tráfego ponto a ponto através da rede.

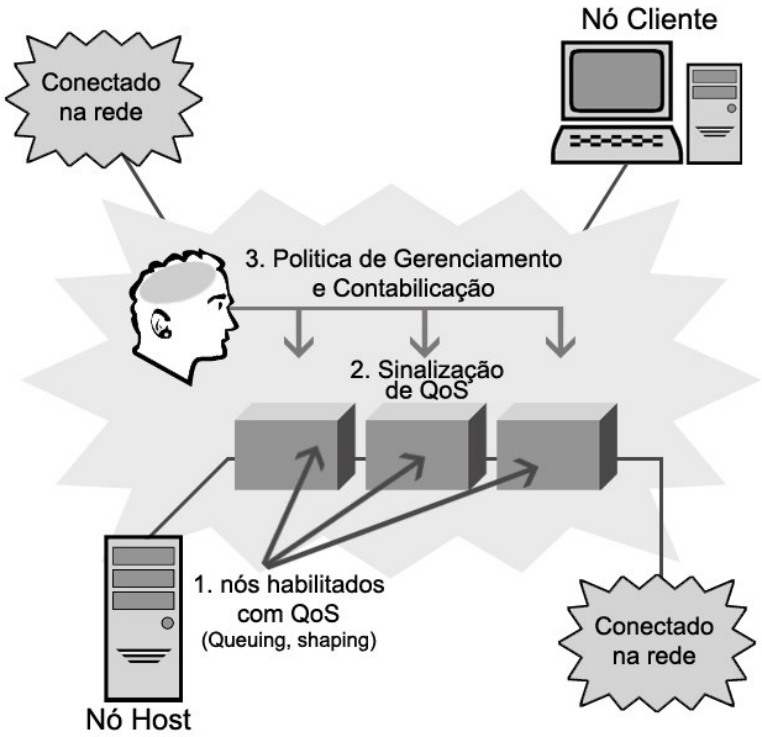

Figura 3.1 – Componentes de uma arquitetura para prover QoS.

3.2 Resource Reservation Protocol (RSVP)

"RSVP é utilizado para que um computador possa requerer, de uma rede, um nível específico de QoS, em favor de um fluxo de dados de uma aplicação" [BRD 97].

O RSVP atua sobre o protocolo TCP, e exige que todos os dispositivos de rede, pelos quais vai trafegar um fluxo, entendam RSVP.

Uma aplicação, para iniciar uma sessão RSVP, cria os objetos SENDER_TSPEC e ADSPEC, que são incluídos numa mensagem RSVP PATH, gerada pela aplicação. O ADSPEC é modificado em cada elemento de rede, conforme a mensagem RSVP PATH trafega até o nó destinatário. Esta modificação é feita em função da capacidade do elemento de rede, de prover QoS, segundo os serviços de controle QoS (*Controled Load* [RFC 2211] ou *Guaranteed* [RFC 2212]) daquele elemento. Quando a mensagem PATH chega ao destinatário, os dados dos objetos SENDER_TSPEC e ADSPEC são encaminhados, através da API RSVP, para a aplicação. A aplicação pode, então, prover o seu RSVP local com os parâmetros de reserva. Dentre os parâmetros, estão o PATH_MTU (com o tamanho máximo de um pacote), o serviço de controle QoS desejado (*Guaranteed* ou *Controled Load*), o objeto TSPEC, descrevendo o nível de tráfego para o qual recursos devem ser reservados, e o objeto RSPEC, descrevendo o nível de serviço desejado. Estes parâmetros são compostos em um objeto RSVP FLOWSPEC e transmitido de volta ao emissor em uma mensagem RSVP RESV. Em cada elemento de rede, a mensagem RESV, contendo o objeto FLOWSPEC, é utilizada para reservar o recurso necessário ao controle de serviço QoS, até chegar ao nó iniciador da conexão.

Os objetos do RSVP, conforme especificados na RFC 2210, são:

- Objeto SENDER TSPEC, gerado no iniciador de uma sessão RSVP, e trafega sem modificações, sendo entregue aos nós intermediários e ao nó destinatário da sessão;
- Objeto ADSPEC, gerado no iniciador da sessão ou em elementos de rede intermediários, pode ser atualizado a cada nó, e segue um fluxo em direção ao nó destinatário da conexão. Possui parâmetros descrevendo o caminho *(Data*

Path), disponibilidade de controle de serviço QoS, e parâmetros requeridos pelos controles de serviço QoS para operar corretamente;

- Objeto RSVP FLOWSPEC, gerado com informações providas pelo nó destinatário, trafega em direção ao iniciador da sessão, e pode ser modificado em elementos de rede intermediários.

No *Windows* 2000 e *Windows* XP, para invocar QoS, deve-se utilizar a GQoS API, parte da DLL (*Dynamic Link Library*) Winsock2.

3.3 Aspectos de QoS em *Web Services*

Com as principais empresas da indústria de software adotando o padrão SOAP, WSDL, UDDI, um grande número de serviços *Web* vem sendo desenvolvido áreas tão diversas quanto finanças e entretenimento.

Qualidade de serviço abrange técnicas que, baseadas nos recursos de rede disponíveis, tentam solucionar as necessidades de serviços de usuários com ofertas de serviços de provedores. Segundo Anbazhagan [ANB 02], os maiores requerimentos para suportar QoS em *web services* são:

Disponibilidade – é a medida do aspecto de qualidade do quanto um serviço está presente e disponível para uso;

Acessibilidade – é a medida de qualidade do quanto um serviço é capaz de servir uma requisição. Pode ser expresso pela probabilidade de uma requisição obter uma resposta. Podem haver situações em que um serviço está disponível porém não acessível, por exemplo, devido a um volume de requisições acima da escalabilidade prevista para o sistema;

Integridade Transacional – é o aspecto de qualidade que garante atomicidade e consistência às transações. Sequências de atividades oriundas de uma requisição devem ser tratadas como uma unidade única de serviço. São todas executadas com sucesso ou são desfeitas (*rollback*) em caso de falha em qualquer etapa da seqüência.

Desempenho – Desempenho é o aspecto de qualidade medido em termos de capacidade de atendimento e latência. Alta capacidade de atendimento e baixa latência são desejáveis. Capacidade representa o número de requisições que podem utilizar um serviço em um dado período de tempo. Latência é o tempo entre enviar uma requisição e receber uma resposta;

Confiabilidade – Confiabilidade se refere à capacidade de manter um serviço e sua qualidade de serviço. O número de falhas por mês ou por ano representa a medida de confiabilidade de um Web Service. Pode ser representada também pela razão entre o número de requisições enviadas e o número de requisições atendidas por uma aplicação servidora em um dado período de tempo;

Aderência aos padrões – É o aspecto de qualidade referente à conformidade com as leis, ao acordo de nível de serviço estabelecido, e aos padrões utilizados. Web Services utiliza muitos padrões, como SOAP, WSDL, HTTP, etc. Estrita aderência na implementação às versões atualizadas por parte dos provedores de serviços é necessária para a correta invocação dos serviços pelas aplicações web service cliente;

Segurança - É o aspecto de qualidade responsável por prover requisitos como controle de acesso, confidencialidade, autenticação, integridade e não repudiação.

Dos aspectos acima apresentados, desempenho, integridade transacional e segurança em *web services* serão os pontos principais abordados na proposta deste trabalho, pois são estas as questões fundamentais a serem resolvidas para possibilitar um novo modelo TEF baseado em SOAP.

3.4 Integridade Transacional em *Web Services*

Muhammad Kaleem [KAL 02], explica que, dada a natureza distribuída e independente do ambiente de web services, é seguro dizer que sistemas baseados em transações tradicionais, com certeza incorreriam em falhas. No entanto, é de desejável que serviços compostos possuam propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Bancos de dados resolvem este problema com a alocação dos recursos que

estão sendo utilizados pela transação. Esta abordagem não é totalmente adequada em um ambiente de web services, no qual serviços não pertencem ao mesmo dono, e estão espalhados pela internet. O tempo de resposta entre os serviços pode ser grande, e um serviço não deve poder ser alocado e controlado indeterminadamente por um terceiro participante da comunicação, que pode ser não confiável.

Publicado em junho de 2002 pelo OASIS, BTP (Business Transaction Protocol) tem o objetivo de prover propriedades ACID a serviços compostos em transações. No BTP existem os seguintes papéis:

- Iniciador – Cliente que inicia uma requisição composta a diversos serviços;
- Finalizador – Cliente que determina o término de uma sessão;
- Decisor – Serviço que faz o papel de coordenador e decide se os serviços participantes devem confirmar ou cancelar suas ações.

O Decisor faz uso de um protocolo de terminação de transação em duas fases. A primeira é a fase "Prepara", na qual os serviços participantes irão votar um status de confirmação ou cancelamento. Baseado nestas respostas, o coordenador passa para a segunda fase do protocolo, a "Finaliza", e ordena a confirmação ou o cancelamento da transação em cada serviço participante. Ao contrário de transações em banco de dados, BTP permite o *commit* seletivo psrvisl de transações. Esta propriedade é denominada Coesão.

BTP resolve bem o problema de transações em *web services*, porém, não trata nenhum aspecto de segurança. Não define como as requisições SOAP podem trafegar cifradas ou define quaisquer regras de autenticação e controle de acesso.

3.5 Problemas de Desempenho em *Web Services*

O SOAP, por ser um protocolo situado na camada de aplicação da pilha TCP/IP, possui limitações inerentes à sua camada em se tratando de alocação de mensagens em filas de prioridade adequada ao nível de serviço necessário. SOAP não define um modelo para QoS em *Web Services*, dependendo de especificações de trabalhos correlatos. Pode-se

considerar que seria altamente desejável a existência de uma forma padrão de priorizar requisições, independente do serviço requerido, a fim de respeitar acordos de nível de serviço.

Como o roteamento atua na camada de rede da pilha TCP/IP, a garantia efetiva de acordos de nível de serviço não deveria se limitar a um modelo para o protocolo SOAP, mas procurar, simultaneamente, fazer uso da infra-estrutura existente em camadas inferiores visando prover QoS. E este é, de fato, o maior problema, a necessidade de implementação de uma solução de QoS para *web services,* abrangente, que atue em diferentes camadas da pilha TCP/IP.

3.6 Conclusões do Capítulo

Neste capítulo foram apresentados conceitos sobre qualidade de serviço em web services, sobre RSVP, um protocolo utilizado para garantir banda a fluxos de tráfego que precisem de QoS, e sobre BTP, um protocolo utilizado para garantir integridade a transações em *web services*. Também foram apresentadas as necessidades de desempenho em um projeto de TEF baseado em web services, para que contratos de nível de serviço possam ser respeitados.

No próximo capítulo, será apresentado um modelo que englobe, e permita validar a utilização das diversas tecnologias apresentadas até aqui para os aspectos de segurança e integridade das transações. Em seguida será apresentada a proposta deste trabalho para o aspecto de desempenho em *web services*.

CAPITULO 4

MODELO PROPOSTO DE QoS E SEGURANÇA PARA WEB SERVICES

4.1 Introdução

Nesta seção será descrito um modelo de segurança e qualidade de serviço (nos requisitos de desempenho e integridade transacional) em *web services*. Este modelo visa uma forma simples de avaliação de aspectos a serem implementados em um ambiente de *web services.* Para os aspectos de segurança e integridade transacional, a idéia é apresentar as soluções tecnológicas e padrões XML mais recentes.

Para a área de desempenho do *web service*, a seção 4.3 apresenta as propostas deste trabalho.

Na seção 4.4, é feita uma sugestão de como utilizar SSL e os novos padrões XML de segurança, para assinatura digital, troca de chaves, e criptografia em *web services*. O capítulo 6 apresenta Implementação e Testes, faz uma avaliação de desempenho sobre a proposta da seção 4.3. Outra avaliação é realizada ao ser agregada a utilização de SSL e XML Digital Signature no sistema.

Na seção 4.5 é sugerida a utilização de BTP para integridade transacional. Esta sugestão não consta da implementação deste trabalho.

4.2 Arquitetura do Modelo Proposto

O modelo proposto neste capítulo aborda os principais requisitos de segurança e QoS que foram considerados para possibilitar a construção de um sistema de integração TEF x SIAC. A figura 4.1 expressa uma visão geral da arquitetura. O sistema todo deve poder ser avaliado em relação à implementação de módulos de QoS e segurança. Dentro de cada um destes módulos, deve-se avaliar a aderência aos padrões de aspectos inerentes. No módulo de QoS, deve-se avaliar o sistema quanto a aspectos de desempenho e Integridade

Transacional. No módulo de Segurança, deve-se avaliar o sistema quanto à utilização dos padrões XML *Digital Signature*, XML *Encryption*, e XKMS. Este modelo não tem por objetivo ser conclusivo, mas sim, funcionar como uma ferramenta extensível de avaliar a adequação de um projeto de *web services*, a padrões de qualidade e segurança necessários ao sistema em questão.

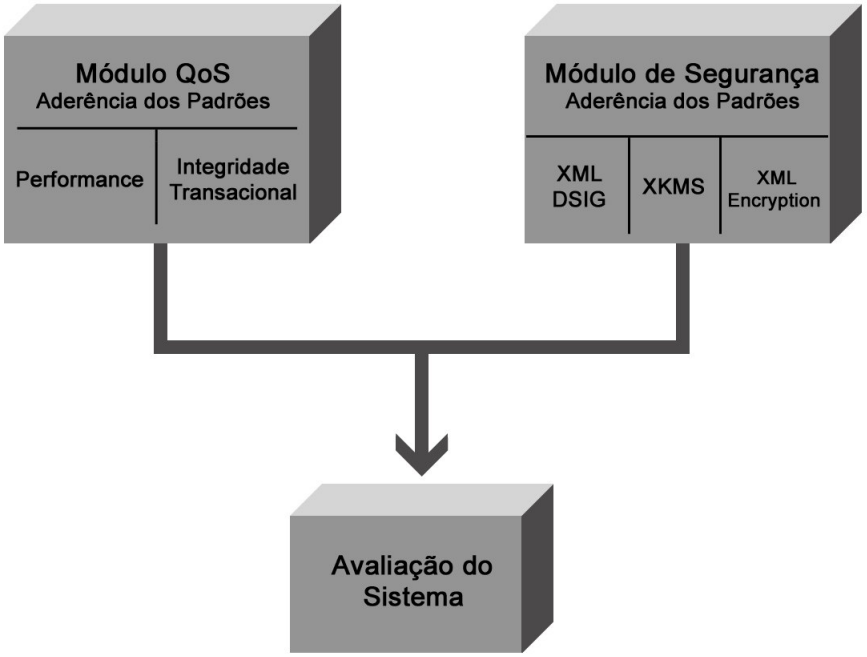

Figura 4.1 – Avaliação da arquitetura.

A tabela 4.1 fornece um bom exemplo do que pode ser um módulo de Avaliação do Sistema, mostrando como quantificar a importância de cada requisito de QoS e Segurança, a um ambiente específico. O Peso determina o grau de importância da métrica dentro do sistema. O Grau de Implementação deve determinar, detalhadamente, a implementação da métrica e qual a sua extensão e pontos fracos (ex: somente implementa assinatura digital do servidor). Para se chegar ao Custo de uma métrica, devem ser feitos testes a fim de avaliar o impacto da implementação daquela métrica no desempenho do sistema.

Módulo QoS			
Métrica	Peso	Grau de Implementação	Custo
DESEMPENHO			
Tratamento de Desempenho em SOAP			
Tratamento de Desempenho RSVP			
Integração entre as camadas de protocolo			
INTEGRIDADE TRANSACIONAL			
Uso do protocolo BTP			
Com *Digital Signature*			
Com XML *Encryption*			
Com controle de autorização			
Módulo Segurança			
XML *Digital Signature*			
XML *Encryption*			
XML *Key Management Specification*			
Secure Sockets Layer			

Tabela 4.1 – *Framework* para analise de tópicos envolvidos na construção de um serviço da web com garantia de qualidade e segurança

4.3 Desempenho

Esta seção propõe um modelo de QoS que permita diferenciar e privilegiar o processamento de determinadas requisições em relação a outras. Uma requisição SOAP deseja que sua mensagem seja tratada com o devido nível de serviço ao longo de sua trajetória até o destino final, incluindo servidores SOAP que executarão algum processo intermediário. A informação da mensagem SOAP é usada para selecionar o apropriado nível de serviço. A seleção do nível de serviço deve ser baseada em políticas de QoS e Contratos de Nível de Serviço (SLA) .

Este modelo propõe um cabeçalho de QoS na mensagem SOAP. O primeiro cliente a requisitar a mensagem deve inserir este elemento, denominado <Qos_Header>. Segundo a especificação do WS *Routing Protocol* [FRY 01], os intermediários podem ser referenciados na requisição SOAP original através de elementos “via” no cabeçalho da

mensagem. O intermediário deve analisar o Qos_Header e determinar como atender ao nível de serviço requerido.

Se este cabeçalho estiver marcado como “mustUnderstand”, então o intermediário deve ser capaz de entender o Qos_Header ou uma mensagem de erro deve ser gerada.

Um servidor deve redirecionar todas as requisições para um módulo de tratamento de prioridades, que chamaremos de “Filtro QoS”. Este módulo será responsável pelas priorizações de cada requisição. Nesta implementação é utilizado um algoritmo *Round Robin* com quatro filas de prioridade.

Se o intermediário for capaz de entender o Qos_Header mas não for capaz de honrar o nível de serviço requerido, uma mensagem de erro SOAP deve ser retornada ao emissor da requisição. Neste caso o emissor pode optar por baixar o nível de serviço exigido ou desistir da requisição.

O protocolo SOAP atua na camada de aplicação, sobre o HTTP, e, portanto, não tem controle sobre o roteamento de pacotes IP, função esta da camada de rede.

Este modelo de desempenho permite a cada nó SOAP a opção de criar chamadas à API de protocolos de nível inferior. Neste trabalho estas chamadas são feitas ao RSVP, a fim de classificar o tráfego da requisição conseqüente. Portas TCP e endereços IP da conexão HTTP são requeridos para esta classificação do tráfego.

Naturalmente, é obrigatório, na implementação deste modelo, que todos os hosts e dispositivos de rede envolvidos na comunicação do web service entendam RSVP. Isto é fácil de se conseguir em uma *intranet* ou em uma rede privada, porém, se os serviços forem requisitados através da *internet*, esta é uma exigência de difícil implementação, uma vez que os pacotes irão trafegar por roteadores de terceiros, que podem não entender RSVP. Os resultados nos nós SOAP intermediários, advindos da implementação do Qos_Header, continuam válidos sob quaisquer circunstâncias, mesmo com requisições via *Internet* e sem chamadas a RSVP.

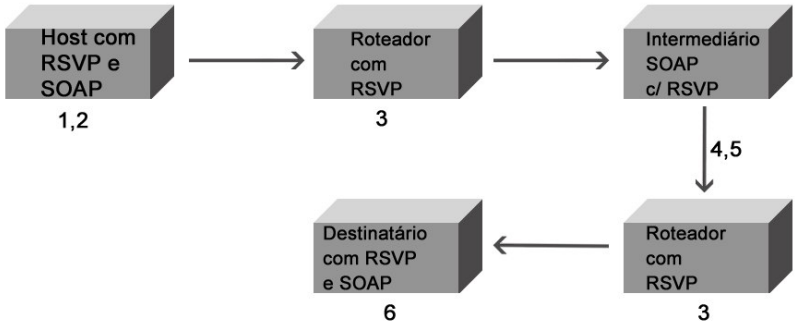

Figura 4.2 – Modelo para desempenho

Descrição das etapas do modelo, pressupondo que todo o tráfego de rede sempre passa por dispositivos que entendam RSVP, conforme mostra o fluxo da figura 4.2.

1. Aplicação Host (Iniciador) prepara requisição a serviços da *web*. Nesta fase o aplicativo monta o Qos_Header dentro do elemento *Header* da requisição SOAP;
2. Aplicação Host faz chamada à API para abrir conexão RSVP passando por todos os servidores intermediários dos serviços *web* necessários, até o computador destinatário da requisição. Nesta etapa são negociados itens de qualidade de serviço do protocolo RSVP;
3. Ao trafegar pelos roteadores, o protocolo RSVP garante a QoS negociada para aquele tráfego;
4. Os servidores intermediários podem prover o serviço requisitado segundo algum algoritmo de QoS e em função do elemento XML <Priority>, do Qos_Header. Vasiliou [VAS 00] apresenta exemplos de algoritmos de QoS. Neste trabalho utilizamos o algoritmo *Round Robin*, conforme demonstrado na figura 4.3;
5. O roteamento SOAP é feito segundo a especificação do WS-*Routing Protocol* [FRY 01], protocolo proposto em outubro de 2001 pela Microsoft e IBM. O roteamento SOAP é importante quando queremos que uma requisição passe necessariamente por determinados servidores

intermediários. Estes servidores intermediários também devem respeitar o nível de serviço especificado no Qos_Header;

6. O servidor SOAP destinatário final da requisição também processa a mensagem segundo algoritmo de QoS, e em função do elemento XML Priority, do Qos_Header.

Na resposta, a conexão RSVP já está aberta, e, portanto, a mensagem trafega até o nó iniciador como tráfego de rede com nível de serviço garantido pelo RSVP; Os servidores SOAP intermediários devem proceder da mesma maneira que na requisição, procurando atender o nível de serviço demandado.

A figura 4.3 mostra que o número de requisições atendidas aumenta conforme o peso da fila de prioridade. Assim, quatro requisições são servidas na fila de prioridade 4, para cada 3 requisições servidas na fila 3, e assim sucessivamente.

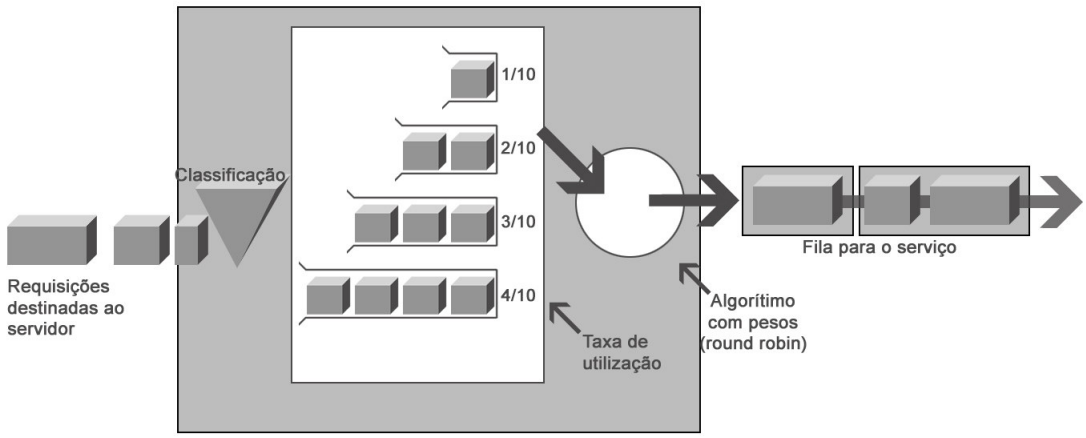

Figura 4.3 – Modelo de algoritmo de QoS adotado, com numero de requisições atendidas pelo sistema *Multi-Thread* em função da fila de prioridade.

Exemplo de requisição SOAP, ao chegar no nó intermediário soap://C.com :

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2001/09/soap-envelope">
  <env:Header>
    <t:TrackingHeader    xmlns:t=”http://lrg.ufsc.br/kpontes/wsmodelo#
tracking">
      <t:Via>
```

```
        <t:Intermediary>soap://A.com/servicoA</t:Intermediary>
        <t:Timestamp>2003-01-09 08:00:00 </t:Timestamp>
      </t:Via>
      <t:Via>
        <t:Intermediary>soap://B.com/servicoB</t:Intermediary>
        <t:Timestamp>2003-01-09 08:01:00 </t:Timestamp>
      </t:Via>
      </t:TrackingHeader>
      <wsrp:path xmlns:wsrp="http://schemas.xmlsoap.org/rp">

<wsrp:action>http://www.operadora.com/pedido_autorizacao</wsrp:action>
        <wsrp:to>          http://www.operadora.com/pedido_autorizacao
</wsrp:to>
        <wsrp:fwd>
          <wsrp:via>soap://C.com</wsrp:via>
        </wsrp:fwd>
        <wsrp:from>soap://B.com/ServicoB</wsrp:from>
        <wsrp:id>uuid:84b9f5d0-33fb-4a81-b02b-5b760641c1d6</wsrp:id>
      </wsrp:path>
     <qos:QoS_Header
xmlns:qos='http://lrg.ufsc.br/kpontes/wsmodelo#QoS'>
        <qos:priority mustUnderstand> 5 </qos:priority>
     </qos:QoS_Header >
  </env:Header>
  <env:Body>
    <ped:autorizacao
xmlns:ped="http://www.operadora.com.br/autorizacao">
    <ped:nrterminal>123456</ped:nrterminal>
    <ped:nome>Leandro Pontes</ped:nome>
    <ped:nrcartao>8356264897263865</ped:nrcartao>
    <ped:expiracao>25062005</ped:expiracao>
    <ped:valor>85,00</ped:valor>
    </ped: autorizacao>
  </env:Body>
</env:Envelope>
```

Quadro 4.1 – Requisição SOAP, segundo padrão do WS-Routing Protocol, e utilizando o QoS_Header.

Alguns detalhes devem ser observados no código do quadro 4.1 acima:

- O elemento <TrackingHeader> é utilizado para guardar um histórico, para fins de auditoria, do trajeto feito pela mensagem. Ele guarda o endereço do nó SOAP visitado e a data e hora deste registro e vai sendo preenchido ao longo do trajeto da mensagem.

- Os elementos <via> do WS-Routing Protocol (WSRP), vão sendo retirados da mensagem conforme ela passa pelos nós SOAP intermediários.
- O elemento <from> guarda o endereço do nó imediatamente anterior, segundo especificação do WSRP. Neste exemplo, o nó imediatamente anterior é soap://B.com .
- O elemento <to>, indica o destino final da mensagem, segundo especificação do WSRP.
- O elemento <priority>, proposto neste trabalho, e o respectivo namespace com sua definição.
- No corpo <Body> da mensagem, a requisição endereçando um pedido de autorização de transação para alguma operadora de cartão.
- Nesta proposta o *software* aplicativo cliente é quem determina o nível de prioridade da requisição, em função do SLA acordado.

O código do quadro 4.1 apresenta, desta forma, um bom exemplo de uma requisição SOAP feita de acordo com as especificações do Filtro de QoS e do WSRP.

4.4 Aspectos de Segurança

4.4.1 Como Aplicar os Padrões de Segurança

SOAP é um protocolo baseado em XML e transportado por HTTP, que, por sua vez, pode ser seguro via SSL. No entanto, em web services, tecnologias de segurança em XML atuam na camada de aplicação da pilha TCP/IP, e usam XML para entregar segurança.

Nesta seção é apresentado um estudo de quando utilizar, ou XML Encryption, ou SSL, associado a XML Digital Signature, para troca segura de informações em web services.

Segundo Sarah Evans [EVA 02], desde sua criação, pela Netscape, em março de 1995, SSL se tornou o protocolo “de facto” para cifrar dados entre requisições e servidores HTTP, atuando na camada de sessão da pilha TCP/IP. A seção 2.2.2 detalha fundamentos

do SSL e podemos constatar que SSL consegue prover autenticação, integridade, e confidencialidade entre dois nós participantes da comunicação cliente x servidor. O problema é que nem todos os requisitos de segurança são preenchidos pelo protocolo SSL. Particularmente, ele não é capaz de prover assinatura digital. Para tratar este aspecto, XML DSig, conforme apresentado na seção 2.2.4, é a tecnologia adequada.

SSL cria um túnel de comunicação segura entre cliente e servidor. Quando a comunicação envolve somente dois nós SOAP, SSL deve ser utilizado, sendo um padrão já bastante difundido no mercado. *Web Services*, no entanto, pode envolver uma comunicação com mais de dois nós participantes. Neste caso, de serviços compostos, XML DSig deve ser utilizado para autenticar os nós participantes da comunicação. Controle de acesso baseado em papéis pode ser feito com um "Filtro de Autorização", como em [DAM 02].

Para troca segura de mensagens, se faz necessário o estabelecimento de uma chave de sessão e um acordo sobre os algoritmos de criptografia e resumo a serem utilizados entre os nós participantes da comunicação. XML Key Management Specification (XKMS) [FOR 01] define um padrão para distribuição de chaves públicas, e certificados digitais, possibilitando então, a distribuição de chaves secretas de sessão.

De posse de uma chave de sessão, XML Encryption deve ser a tecnologia responsável pela troca segura de mensagens, provendo ambos, confidencialidade e integridade na comunicação entre os diversos nós.

4.5 Integridade Transacional

O Business Transaction Technical Committee deferiu integrar BTP 1.0 com padrões de segurança. Nesta versão BTP assume que todos os participantes pertencem a um domínio confiável.

Para um projeto de integração SIAC x TEF, integridade transacional é um aspecto bastante importante. Uma compra com cartão de débito, por exemplo, deve validar informações da base de dados da operadora de cartão, e, também, da instituição bancária. Neste caso, a atomicidade da transação é decisiva, pois qualquer falha na validação deve implicar na rejeição da transação.

BTP carece de diversos aspectos de segurança: autenticação, autorização, confidencialidade, integridade (dos dados) e assinatura digital.

Em relação a confidencialidade e integridade dos dados, nossa proposta é de que todos os participantes da transação devem estabelecer uma chave de sessão e algoritmos de criptografia e de resumo. XKMS deve ser a tecnologia utilizada para criação segura de uma chave de sessão, com a qual XML *Encryption* deve ser utilizado pra cifrar as mensagens daquela sessão BTP. XML *Signature* deve ser utilizado para garantir a assinatura digital dos clientes e serviços participantes. Caso exista necessidade de controle de acesso, [DAM 02] propõem um modelo de um "Filtro de Autorização" baseado em papéis. Desta maneira, BTP pode garantir integridade das transações trabalhando, paralelamente, com tecnologias XML que garantem os requisitos de segurança necessários.

4.6 Conclusão

Neste capítulo foi apresentada a proposta de um modelo extensível para avaliar a de utilização de padrões de segurança e qualidade de serviço. Foi apresentada também uma proposta para o aspecto de desempenho, com um objeto Filtro de QoS responsável pela priorização de requisições SOAP, e RSVP garantindo largura de banda. Em seguida foi feito um estudo para determinar quando utilizar SSL ou os padrões XML para segurança, e em que estas tecnologias podem se complementar. Finalmente, é proposta uma extensão ao protocolo BTP mostrando aspectos de segurança que devem ser tratados.

No próximo capítulo será apresentada a rede utilizada para os testes e, a seguir, serão detalhados a implementação e os testes realizados a fim de comprovar a eficiência do Filtro de QoS, e como os resultados podem ser afetados com a utilização de RSVP, de SSL e de XML *Signature*.

CAPITULO 5

IMPLEMENTAÇÃO E TESTES

5.1 Introdução

Este capítulo apresenta o ambiente de desenvolvimento e detalha a implementação e os testes efetuados. Na seção 5.2 é descrito o ambiente de rede utilizado.

A implementação da proposta de QoS para o aspecto de desempenho foi dividida em duas etapas. Na seção 5.3 é explicada como foi feita a implementação do Filtro de QoS, a fim de para prover diferenciação de serviço sobre requisições SOAP, e quais os testes feitos para sua validação. Em seguida, os testes são estendidos para um ambiente seguro com SSL e os resultados são comparados. Na seção 5.4 são apresentados testes feitos com XML *Signature*, e na seção 5.5 é agregada a complexidade de estabelecimento de uma conexão RSVP antes da requisição SOAP.

5.2 Apresentação do Ambiente de Desenvolvimento e Testes

O ambiente utilizado para desenvolvimento do modelo de Desempenho consiste em uma LAN (Local Área Network) TCP/IP Ethernet, com saída de 128 Kb para a *internet* por uma linha LPCD da Brasil Telecom. Esta rede é composta por três sub-redes IP. A figura 5.1 mostra uma máquina cliente, as máquinas que fazem roteamento, e um servidor *web* com um *web service*. Outras máquinas pertencentes a cada sub-rede não estão representadas na figura.

- Estação Pentium III 800 MHz, *Windows 2000 professional*, com RSVP instalado, *software* cliente do *web service*, objetos *ActiveX* do produto *SecureXML* da InfoMosaic *Corporation*, e Certificado Digital;
- Host *Windows 2000 Server* com RSVP e duas placas de rede ethernet, fazendo roteamento ;
- Linux *Red Hat* 7.2 com RSVP e três placas de rede ethernet, fazendo roteamento, e com saída para a *internet*;

- *Host* Pentium III 1 GHz*, Windows 2000 Server*, com RSVP, *software* servidor do *web service*, objetos *ActiveX* do produto *SecureXML* da InfoMosaic *Corporation*, e Certificado Digital;

Os módulos de software cliente e servidor do *web service* foram desenvolvidos na linguagem Delphi 6.0, para plataforma *Windows*.

As páginas para validação do XML *Signature* foram desenvolvidas em ASP (*Active Server Pages*)

Para verificação do código das mensagens SOAP em tráfego foi utilizada a ferramenta *Trace* do Microsoft SOAP *ToolKit* 2.0, que pode ser encontrado, gratuitamente, em www.microsoft.com.

Na estação cliente *Windows 2000 professional*, o estabelecimento de chamadas com QoS foi habilitado a partir da instalação do serviço QoS *Packet Scheduler*.

No roteador *Windows 2000 Server* foram instalados os serviços:

- *Active Directory;*
- *Routing and Remote Access Service;*
- QoS RSVP e QoS *Packet Scheduler;*
- *QoS Admission Control*.

No servidor do *web service Windows 2000 Server* foram instalados os serviços:

- *MS-SQL SERVER 2000;*
- *Internet Information Server 5.0 (IIS);*
- QoS RSVP e QoS *Packet Scheduler;*
- Objetos *ActiveX* do produto *SecureXML* da InfoMosaic *Corporation*;
- Certificado digital;

No Linux, foi instalado um *deamon* RSVP encontrado em ftp://ftp.isi.edu

Para monitoração do tráfego no servidor SOAP e no cliente, foi utilizada a ferramenta TCMON (*Traffic Control Monitor*), encontrado no *Windows 2000 Resource Kit*.

Para gerar “ruído” na rede, ou seja, pacotes em excesso, destinados a congestionar a rede para os testes, foi utilizado o *freeware* MNOISE. Os parâmetros de carga do Mnoise são explicados na seção 5.5.

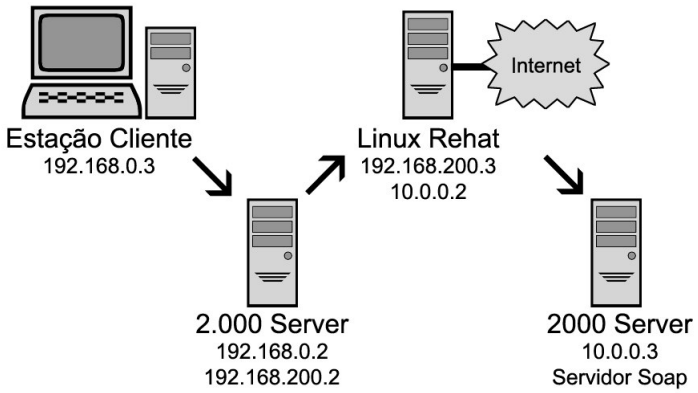

Figura 5.1 – Ambiente de Rede.

O ambiente de rede, descrito nesta seção 5.1, permite a realização dos testes da proposta de desempenho deste trabalho, considerando-se uma intranet, com ethernet como meio físico. Para ambientes que tenham a internet como meio, esta avaliação deve ser refeita, pois o tempo gasto em tráfego, pelas requisições e respostas, será bem maior. Também será maior o tempo gasto para estabelecimento de conexões SSL ou RSVP.

5.3 Filtro de QoS

A implementação proposta do módulo servidor consiste na criação de um objeto “FiltroQos”, o qual é responsável pela priorização das filas, segundo algum critério. A idéia principal é de que todas as requisições para um servidor, devem passar, antes, pelo FiltroQoS, que é o módulo responsável por liberar e endereçar as requisições aos serviços inicialmente pedidos, conforme mostra a figura 5.2. Ao receber uma requisição, o serviço a redireciona para o FiltroQoS, que cria uma Identificação única (Id) da requisição, que será anexada aos outros parâmetros da requisição, para que ela entre em uma fila antes de ser atendida por um serviço. O FiltroQoS é responsável por manter uma estrutura de dados

com Id´s em fila, em execução, e seus serviços. Quando um determinado serviço S termina de ser executado, ele retorna ao FiltroQoS a Id finalizada, e o FiltroQoS pode então liberar a próxima requisição da fila.

Esta abordagem é bastante flexível, pois permite que o objeto FiltroQoS seja customizado segundo as necessidades de QoS dos serviços de um determinado servidor. A implementação do FiltroQoS pode ser feita usando-se filas para cada serviço, ou com uma fila genérica. Outra possível abordagem é implementar o FiltroQoS para liberar as requisições segundo algum critério de tempo, personalizado para o servidor em questão. Por exemplo, pode-se chegar à conclusão que um determinado servidor é capaz de atender a 20 requisições a cada 5 segundos, para os *web services* por ele disponibilizados, e configurar o FiltroQoS para trabalhar nesta taxa de utilização.

Alguns aspectos devem ser ressaltados:

- Para que um módulo cliente possa, ele mesmo, determinar o nível de prioridade de uma requisição, ele deve ser "QoS consciente", ou seja, deve prover o elemento XML *Priority*, que será utilizado no algoritmo do objeto FiltroQoS;
- Caso a requisição cliente não possua o elemento *Priority*, o FiltroQoS deve prover a requisição com algum nível de serviço, a critério de implementação. Na especificação DTD desta proposta, algumas outras propriedades são expostas, como mostra o quadro 5.1. Estas propriedades poderiam ser utilizadas por algum critério de priorização do FiltroQoS. Neste trabalho, no entanto, somente a propriedade *Priority* é utilizada;
- Neste trabalho é utilizado o algoritmo *round robin,* com 4 filas de prioridade, conforme a figura 4.3.
- Outras propriedades, como a identificação do usuário, ou a identificação do serviço, podem ser utilizados, conforme necessidade, em outras implementações do FiltroQoS;
- Por questão de segurança, o filtro QoS pode ser um objeto interno e seus métodos não precisam ser expostos via WSDL. Para fins de ilustração, neste trabalho ele foi implementado como um serviço chamado via SOAP, e o

quadro 5.1 mostra o WSDL do objeto implementado. Observe que, embora não seja utilizado no algoritmo deste trabalho, são definidos os elementos XML IdUser e IdMetodo, que, em uma implementação diferente da realizada neste trabalho, poderiam fazer parte do critério de priorização e seleção de uma requisição por outro algoritmo;

- Em uma situação ideal, a implementação de um QoSFilter deveria estar interna a um *web server*. Neste caso o QoSFilter poderia atuar sobre quaisquer solicitações HTTP, e não somente sobre as que acessam serviços via SOAP. Neste trabalho, o *web server* utilizado é o IIS, de código fechado e o QoSFilter atende somente a serviços endereçados via SOAP;

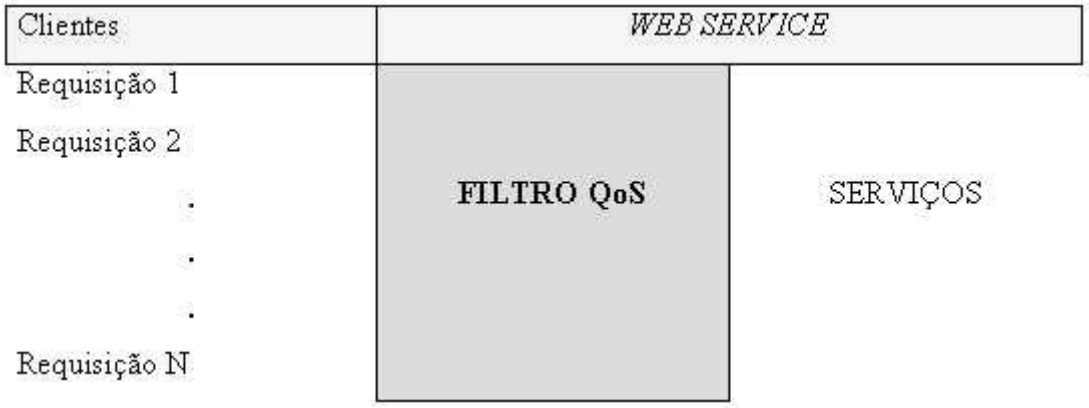

Figura 5.2 – Arquitetura proposta para desempenho.

```
  <?xml version="1.0" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
name="IQoSFilterResponseservice" targetNamespace="http://tempuri.org/"
xmlns:tns="http://tempuri.org/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
-     <message name="RetIdTransacaoRequest">
          <part name="Priority" type="xs:int" />
          <part name="IdMetodo" type="xs:string" />
          <part name="IdUser" type="xs:string" />
          <part name="DtHor" type="xs:datetime" />
      </message>
-     <message name="RetIdTransacaoResponse">
          <part name="return" type="xs:int" />
      </message>
```

```
-     <portType name="IQoSFilterResponse">
-         <operation name="RetIdTransacao">
              <input message="tns:RetIdTransacaoRequest" />
          <output message="tns:RetIdTransacaoResponse" />
          </operation>
      </portType>
-     <binding name="IQoSFilterResponsebinding"
type="tns:IQoSFilterResponse">
          <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http" />
-             <operation name="RetIdTransacao">
                  <soap:operation soapAction="urn:UQoSFilterResponseIntf-
IQoSFilterResponse#RetIdTransacao" style="rpc" />
-                 <input>
<soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:UQoSFilterResponseIntf-IQoSFilterResponse" />
                  </input>
-                 <output>
  <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:UQoSFilterResponseIntf-IQoSFilterResponse" />
                </output>
            </operation>
      </binding>
-     <service name="IQoSFilterResponseservice">
-         <port name="IQoSFilterResponsePort"
binding="tns:IQoSFilterResponsebinding">
              <soap:address location="http://edm-
ntserver/scripts/QoSFilterSoapServer.exe/soap/IQoSFilterResponse" />
          </port>
      </service>
  </definitions>
```

Quadro 5.1 – WSDL, gerado pelo meu programa Delphi servidor, extraída do servidor de minha *intranet* no endereço: http://edm-ntserver/scripts/QoSFilterSoapServer.exe/wsdl/IQoSFilterResponse.

Os três testes executados, sobre o ambiente acima descrito, foram baseados em quatro filas de prioridade. Requisições na fila 1 eram servidas à taxa de uma por vez. Na fila 2, à taxa de duas por vez, etc.

No primeiro teste foram disparadas cem requisições simultâneas para um determinado serviço que retornava uma consulta a um banco de dados MS SQL Server 2000, e, logo a seguir, uma centésima primeira requisição, que foi monitorada. Este teste foi repetido quatro vezes, com a requisição número 101 entrando em cada fila de prioridade. O segundo teste foi semelhante ao primeiro, porém com somente cinqüenta

requisições simultâneas e a requisição de número cinqüenta e um sendo monitorada. O terceiro teste foi repetido para dez requisições e uma décima primeira monitorada. Os resultados estão na tabela 5.1 a seguir:

Prioridade/Carga	Carga de 100 req. tempo (s)	Carga de 50 req. tempo (s)	Carga de 10 req. Tempo (s)
Prioridade 01	118	63	21
Prioridade 02	62	35	11
Prioridade 03	43	25	5
Prioridade 04	36	20	5

Tabela 5.1 – Tempo, em segundos, para a próxima requisição ser atendida, em cada fila de prioridade, para cada faixa de carga de requisições no servidor.

Importante ressaltar que a alocação de prioridades pelo cliente neste teste foi aleatória. Estatisticamente a probabilidade é de que cada fila tenha tido um quarto das requisições. Para melhor validar esta experiência, os testes acima foram repetidos três vezes e a tabela 5.1 apresenta valores médios. Notar que, com uma carga pequena de requisições, as filas de prioridade 3 e 4, atenderam no mesmo tempo, visto que somente uma rodada do algoritmo pode já ter sido suficiente para chegar a vez da requisição monitorada.

Em seguida, foi instalado um certificado digital de teste, fornecido pela CertiSign (www.certisign.com.br), no servidor Web IIS, habilitando-se o SSL. Todos os testes foram repetidos, para as quatro filas de prioridade, e com o mesmo volume de requisições, seguindo o mesmo procedimento explicado para os resultados da tabela 5.1. Porém, neste teste, as requisições SOAP trafegam em ambiente seguro por SSL. Os resultados se encontram na tabela 5.2:

Prioridade/Carga	Carga de 100 req. tempo (s)	Carga de 50 req. tempo (s)	Carga de 10 req. Tempo (s)
Prioridade 01	122	66	22
Prioridade 02	65	37	11
Prioridade 03	45	26	5
Prioridade 04	38	21	5

Tabela 5.2 – Tempo, em segundos, para a próxima requisição ser atendida, em cada fila de prioridade, para cada faixa de carga de requisições no servidor, em ambiente seguro por SSL.

Para efeito de avaliação do ambiente, foi feito um teste de carga para uma aplicação cliente fazendo uma única requisição a um web service que acessa um banco de dados MS-Sql-Server 2000, e retorna dados de um cadastro de produtos em formato XML. A requisição foi feita para 5.000 produtos, 2.500 produtos. 500 produtos, e 10 produtos, e foram medidos os tempos de retorno. Em seguida o teste foi repetido em ambiente seguro por SSL. A tabela 5.3 mostra os resultados.

Carga/Requisição	Sem SSL tempo (s)	Com SSL tempo (s)
5000 produtos	17	19
2500 produtos	15	14
500 produtos	4	3
10 produtos	2	2

Tabela 5.3 – Tempo, em segundos, para uma requisição ser atendida, variando o volume de informação trafegado.

Analisando os resultados dos testes feitos nesta seção, pode-se concluir que o preço pago, em tempo de atraso, pela segurança de trafegar as requisições SOAP em ambiente seguro por SSL, em uma *intranet*, não é muito alto. Este atraso não cresce na mesma proporção que o número de requisições, e, tampouco, na mesma proporção que o tamanho

das requisições. Desta maneira, o Filtro de QoS continua a ser o principal determinante pelo tempo de resposta de uma requisição, mesmo em ambiente seguro por SSL, conforme retrata o gráfico da figura 5.3.

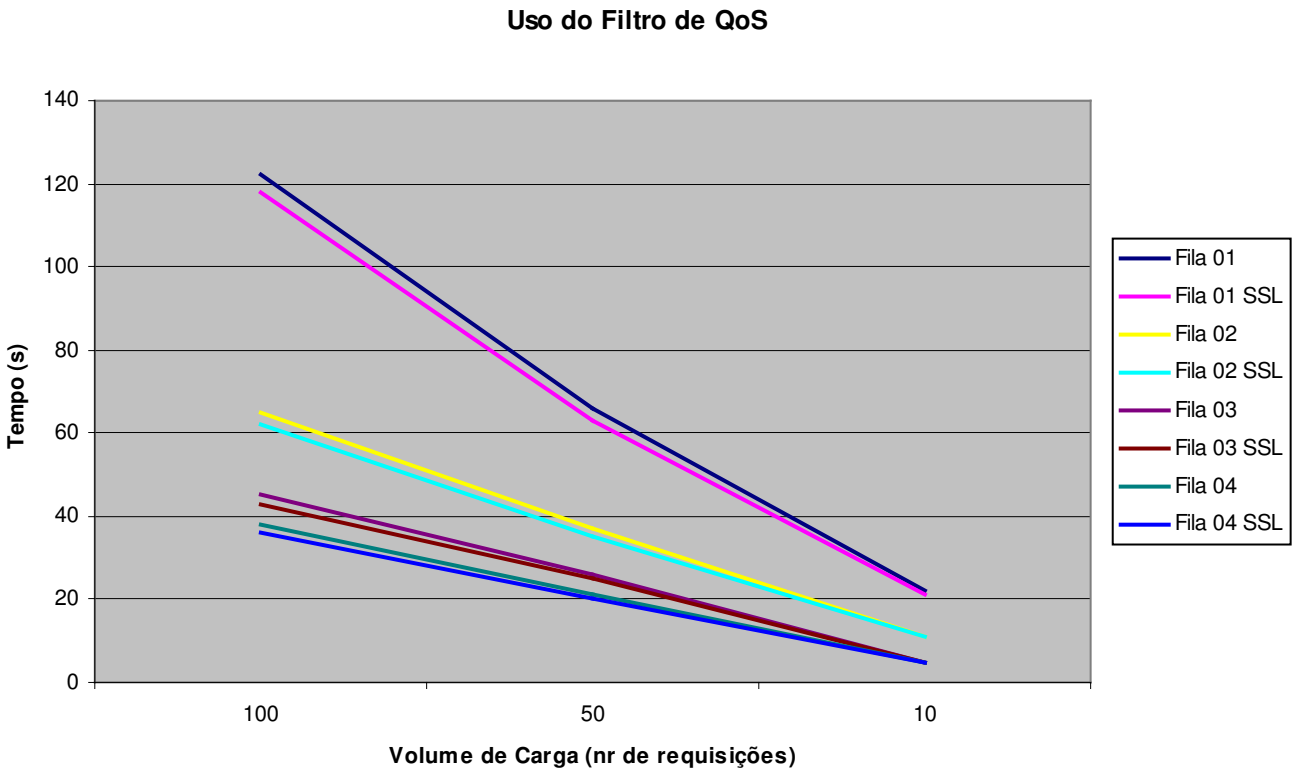

Figura 5.3 – Comparação dos resultados dos testes feitos do Filtro de QoS, com uso, e sem uso de SSL.

5.4 XML *Digital Signature*

Para a implementação de testes com XML Digital Signature, foi instalado em um navegador cliente, e no servidor *web*, certificados digitais, da InfoMosaic *Corporation* (www.infomosaic.net). Em seguida, foi instalado em ambas as máquinas, a versão de testes do produto *SecureXML*, também da InfoMosaic, como objetos *ActiveX*. As páginas do servidor foram desenvolvidas em ASP (*Active Server Pages*) com JavaScript.

Os testes realizados foram baseados em validar a assinatura digital do cliente emissor da mensagem. Arquivos XML de diferentes tamanhos foram assinados digitalmente, na origem, pelo acesso aos componentes *ActiveX* do *SecureXML*, através de JavaScript. Ao assinar um arquivo digitalmente, estes componentes concatenam a assinatura digital e a chave pública do emissor ao arquivo. Em seguida o usuário pode fazer uma requisição ao servidor para validação da assinatura digital, concatenada ao arquivo XML.

No lado do servidor, ao receber a requisição com o conteúdo do arquivo XML assinado digitalmente, a assinatura digital é decifrada com a chave pública do emissor, e o resultado é comparado com o resumo da mensagem feito no servidor. Se ambos os procedimentos forem validados, a assinatura digital é considerada válida, e uma resposta é emitida para o cliente, conforme mostra a figura 5.4. Na figura 5.4, também se observa que, além da assinatura com a chave privada, o *secureXML* também possibilita a assinatura por mouse.

Figura 5.4 – Imagem retornada ao navegador cliente com a verificação da assinatura digital.

Os testes realizados foram para avaliação de desempenho da validação da assinatura digital pelo servidor. Para isto, foram assinados digitalmente, arquivos XML de diferentes tamanhos, e submetidos à validação. Os testes foram repetidos para uma e para três requisições simultâneas. Os resultados, em segundos, são medidos pelo programa servidor, para validar a assinatura digital. A tabela 5.4 mostra os resultados.

Tamanho/Requisição	**1 requisição tempo (s)**	**3 requisições tempo (s)**
300 kb	0,9	0,9
600 kb	1,5	1,5
1500 kb	2,2	2,3
1800 kb	3,1	3,2

Tabela 5.4 – Tempo, em segundos, para a assinatura digital de uma requisição ser validada.

Os resultados da tabela 5.4 indicam uma pequena variação no desempenho quando requisições simultâneas de arquivos grandes devem ser validadas. Para o caso prático de transações TEF, no entanto, o tamanho da requisição será bem menor do que o dos testes, o que torna os benefícios da assinatura digital bastante atraentes, comparados com um baixo custo de desempenho. Pode-se prever que uma futura integração de *secureXML* com o Filtro de QoS não implicará em sensíveis quedas de desempenho, mas poderá agregar um grande valor em aspectos de segurança da arquitetura.

5.5 Chamada a RSVP

Neste teste o objetivo é mostrar a utilidade de, antes de chamar o *web service*, ser aberta uma conexão RSVP com o servidor. O serviço chamado por este teste, retorna uma grande quantidade de informação ao cliente, composta de duzentas linhas com: um número seqüencial, um *time stamp* do servidor, uma frase de tamanho fixo e os caracteres *Carriage Return* (CR) e *Line Feed* (LF), como no exemplo abaixo:

Exemplo: 150-0,000198437497601844-Enviada linha teste de numero CRLF

O módulo cliente, ao encontrar um CRLF, o exclui, junto com os 31 caracteres à sua esquerda (que sempre deveriam corresponder à frase de tamanho fixo), e gravam o restante em uma nova linha em um arquivo. Desta forma, o conteúdo do arquivo no final deveria ser composto de números seqüenciais, seguidos do caractere "-", seguidos de um número *time stamp* do servidor.

Utilizou-se o programa *freeware* chamado Mnoise, com o objetivo de gerar ruído, ou seja, congestionar o tráfego na rede. Em seguida, foram feitos dois testes com uma aplicação cliente fazendo uma chamada para o serviço descrito acima, o primeiro sem, e o segundo com abertura de conexão RSVP. Os parâmetros da conexão foram configurados como:

Token Bucket size = 64 bytes;
Token Rate = 200 kbytes;
O que resulta em 200 *1024 /64 = 3200 pacotes transmitidos por segundo.

Token Rate é a média de transmissão de dados, utilizado para controlar o intervalo de transmissão entre os pacotes no nó emissor. *Token Bucket Size* é o maior tamanho de pacote que o controle de tráfego enviará para a rede. Deve ser menor que a MTU *(Maximum Transmission Unit).*

Os resultados, que podem ser encontrados no Apêndice A, mostram claramente o efeito do congestionamento na resposta do *web service* chamado sem abertura de conexão RSVP, com perda significativa de pacotes, principalmente partir da numeração 133 (cento e trinta e três). Por outro lado, o resultado na resposta do *web service* chamado com

abertura de conexão RSVP, mostra um comportamento sem perda de informação, com todas as linhas sendo gravadas, pelo cliente, na seqüência correta. A monitoração dos fluxos foi feita com a ferramenta TCMON (Traffic Control Monitor), para verificação da utilização de controle de banda, conforme mostra a figura 5.5.

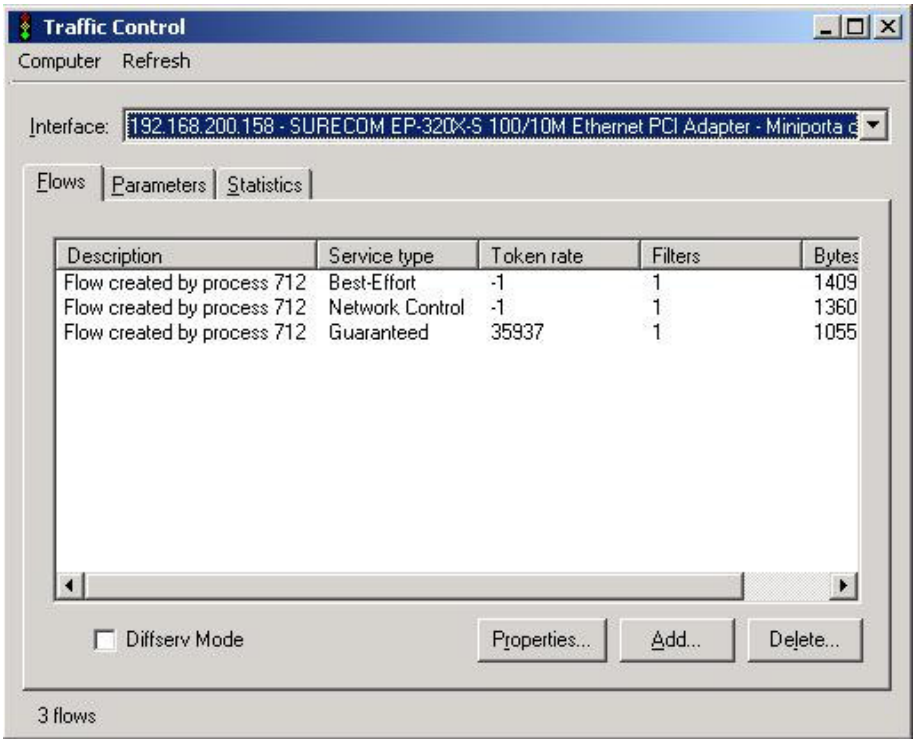

Figura 5.5 – Tela do TCMON, mostrando a utilização de controle de banda.

5.6 Conclusão

Este capítulo apresentou o ambiente de rede utilizado e detalhou a implementação do Filtro de QoS e como foram realizados os testes para sua validação. Em seguida foram feitos testes com a utilização de SSL e de XML *Signature*, e foi avaliado o impacto destas tecnologias de segurança sobre o serviço. Finalmente, foi testada a utilização simultânea de RSVP, e foi avaliado o impacto deste protocolo sobre o serviço.

O próximo capítulo apresenta as conclusões finais do trabalho, comparando com trabalhos correlatos, e destacando as principais contribuições científicas.

CAPITULO 6

CONCLUSÃO

6.1 Introdução

O modelo proposto neste trabalho constitui uma forma simples de documentar e avaliar o estágio em que se encontra um projeto de *web services*, contemplando tópicos relacionados a segurança, desempenho, e integridade transacional. Estes pontos são fundamentais na ambição de propor uma alternativa ao modelo TEF usando a tecnologia de *web services*.

Este trabalho elucidou as tecnologias e padrões XML, para as áreas de segurança e integridade das transações, necessários a um projeto de tal envergadura. Também foi comprovado, com implementação e testes, o funcionamento do "Filtro de Qos" na proposta para desempenho. E, finalmente, foi feita uma avaliação de desempenho considerando-se o impacto da segurança por SSL e XML *Signature*, e também o uso de controle de banda por RSVP.

Foram atingidos todos os objetivos da seção 1.2:

- Foram realizados estudos sobre SSL e sobre as estruturas padrão XML Encryption, XML Signature e XKMS, e foi esclarecido como e quando utilizar cada uma destas tecnologias, e em que se complementam (objetivos 1 e 2 da seção 1.2);
- Foram identificados os principais requisitos de Qualidade de Serviço em *web services,* necessários ao desenvolvimento de um novo modelo para TEF, e apresentados sob um modelo extensível de avaliação (objetivo 3 da seção 1.2);
- Foram esclarecidas as principais diferenças e vantagens do *Web Services* em relação a outros modelos de integração de computação distribuída, como CORBA e DCOM (objetivo 4 da seção 1.2);

- Foi feita a proposta e implementação de um Filtro de QoS para tratar desempenho em *Web Services*, com implementação, e validação com testes, de um serviço *web* com filas de prioridade e com estabelecimento de conexão RSVP simultânea (objetivo 5 da seção 1.2);
- Foi avaliado, com implementação e testes, o impacto da utilização de SSL e de XML *Digital Signature* no desempenho dos serviços (objetivo 6 da seção 1.2);
- Foi esclarecido, no protocolo BTP (*Business Transaction Protocol*), que padrões XML para segurança devem ser utilizados (objetivo 7 da seção 1.2);

6.2 Comparação com Trabalhos Correlatos

[BAN 97] procura classificar um perfil do usuário baseado em seu histórico de acesso, a fim de determinar o nível de serviço a que ele terá direito no web server. Neste caso, as regras para definição de prioridades estão todas no servidor *Web*. [CHA 00] mostra que se pode atender requisitos de QoS, não somente pela classificação e priorização do usuário, mas também, pela transcodificação de conteúdo multimídia a ser trafegado na rede. Segundo Chandra, 70% do conteúdo trafegado na web nos dias de hoje, constituem tráfego multimídia, e, em se trocando qualidade por velocidade, pode-se transcodificar arquivos multimídia por outros de qualidade inferior, porém menores, exigindo menos largura de banda. Assim, um arquivo de imagem colorida, pode ser enviado em um formato menor, em preto e branco, em função da largura de banda disponível no momento e do contrato de SLA com o usuário que fez a requisição. Assim como [BAN 97], [CHA 00] estabelece regras de negócio no servidor que determinam a priorização de um usuário em relação a outro, bem como se um determinado conteúdo deve ou não ser transcodificado. Em nossa proposta, como nosso cliente não é um usuário, mas sim um *software* aplicativo, e que normalmente deverá passar por um processo de homologação, as regras de negócio para definirem o nível de serviço podem estar no aplicativo cliente, respeitando os devidos contratos de SLA. Em relação a transcodificação de conteúdo, neste trabalho não

abordamos o assunto, tendo em vista que os serviços citados são textuais e voltados para resolver problemas de transações com cartão. Transcodificação de conteúdo também implicaria em problemas com algoritmos de consistência de integridade das mensagens.

[RON 99] define uma proposta em nível de corporação para *Security Service Level Agreement* (SLA). Ronda estabelece diversas categorias de segurança que devem ser endereçadas em um contrato de SLA ao se terceirizar a infra-estrutura de rede da corporação. Esta dissertação pode se enquadrar na categoria "Segurança do *Web Server*", visto que os diversos aspectos de segurança abordados neste trabalho deverão ser implementados em objetos a serem chamados pelo servidor SOAP. Neste sentido, pode-se concluir que esta dissertação e a parte de segurança do *framework* aqui definido, ajudam a montar um *Security* SLA num sentido corporativo mais amplo.

O projeto BTP (Business Transaction Protocol), cuja primeira implementação é o Hewlett-Packard Web Services Transactions (HP-WST) [KAL 02], cria um modelo de integridade transacional pra *web services* baseado na lógica do modelo de transação dos Sistemas Gerenciadores de Banco de Dados (DBMS). O protocolo BTP, no entanto, aborda somente o aspecto de qualidade integridade. BTP carece de autenticação e de assinatura digital. Este trabalho propõe uma solução baseada na utilização dos novos padrões de definição de mensagens XML para assinatura digital e criptografia, como XML DSig e XML Encryption.

No aspecto de qualidade, segurança, [EVA 02] discute se SSL é capaz de endereçar os diversos requisitos de segurança necessários a um web service entre dois nós SOAP. Neste trabalho, conclui-se que SSL resolve somente para serviços em uma estrutura cliente x servidor. Neste trabalho é detalhado como atender a serviços com três ou mais nós. Deve ser estabelecida uma chave de sessão entre os nós participantes para troca de mensagens com confidencialidade. Neste processo, devem ser utilizados os padrões XML *Signature*, XML *Encryption* e XKMS.

[DAM 02] apresenta uma proposta de controle de acesso e autorização baseado em papéis para web services. Damianni cria a figura de um "filtro de acesso", que consiste em um módulo que funciona como um *firewall* para as requisições dos clientes. Este filtro pode

autorizar, negar, ou autorizar a requisição com modificações em determinados parâmetros. Para o processo autorização de venda TEF baseado em *web services*, não seria necessário tal complexidade. No entanto, para que a operadora possa prover outros serviços a seus clientes (p.ex. Consulta extrato, Cancelamento de cartão, etc), a proposta de [DAM 02] pode ser bastante relevante, ainda mais tendo em vista sua compatibilidade sintática com a definição de XML Digital Signature. [DAM 02] também serve como inspiração da implementação do Filtro QoS.

6.3 Principais Contribuições Científicas

Podemos enumerar como principais contribuições científicas deste trabalho:

- Elucidação dos principais requisitos de QoS e segurança necessários a um projeto de *web services* para que possa atender às necessidades da Transferência Eletrônica de Fundos, e mapeamento destes requisitos em um modelo simples e extensível;
- Estudo comparativo entre as tecnologias *web services*, CORBA e DCOM;
- Estudo de padrões XML para tratar segurança em web services: XML *Digital Signature*, XML *Encryption*, e XKMS. Este estudo relacionou suas aplicações, e mostrou sua relação com SSL e como podem se complementar;
- Desenvolvimento e testes de um modelo de desempenho baseado na implementação de um "Filtro de QoS", proposto neste trabalho, e com atuação não somente na camada de aplicação, onde se encontra o protocolo SOAP, mas também com chamada de API simultânea para o protocolo RSVP. Neste tópico ainda se conclui que:
  - O Filtro de QoS funciona e prioriza as requisições de acordo com o parâmetro "*Priority*", afetando diretamente o tempo de resposta das requisições;
  - O Filtro de QoS é genérico e pode servir a diferentes serviços;
  - O Filtro de QoS é configurável em função da capacidade de atendimento de um serviço ou do servidor;

- O RSVP, garantindo largura de banda, consiste em importante protocolo para garantia da integridade dos pacotes em trânsito, em redes muito congestionadas.

- Avaliação de desempenho de web services com a utilização de SSL e XML Digital Signature, chegando-se às seguintes conclusões:
  - O atraso causado pela comunicação segura com SSL não cresce na mesma proporção que o número de requisições, e, tampouco, na mesma proporção que o tamanho das requisições. Desta maneira, o Filtro de QoS continua a ser o principal determinante pelo tempo de resposta de uma requisição;
  - Para requisições pequenas, o atraso causado por XML *Signature* é muito pequeno, e, desta maneira, o Filtro de QoS continua a ser o principal determinante pelo tempo de resposta de uma requisição;
  - O custo da segurança por SSL ou XML *Signature* é baixo, em relação aos benefícios que proporciona, para um ambiente de integração TEF sobre *web services*;

6.4 Trabalhos Futuros

Pode-se definir como trabalhos futuros:

- No módulo de QoS, estender o modelo de desempenho a outros protocolos de classificação de tráfego e reserva de banda;
- Definição de novas métricas para QoS, estendendo o modelo;
- Implementação do modelo de prioridade internamente, diretamente no código de um *web server;*
- Detalhar integração do protocolo BTP com XML Digital Signature, com XML Encryption, e com o modelo proposto de controle de acesso baseado em papéis, como em [DAM 02]; e
- Integração do *SecureXML* ao Filtro de QoS, em Delphi, fechando um sistema com priorização de requisições SOAP assinadas digitalmente.

REFERÊNCIAS BIBLIOGRÁFICAS

[ALA 96] Alan O. F, Philip K, **The SSL Protocol version 3.0**, Netscape Corporation, http://wp.netscape.com/eng/ssl3/draft302.txt , 1996.

[ANB 02] Anbazhagan M e Arun N , **Understanding quality of service for Web services**, IBM developer works 2002.

[BAN 97] Banatre M, Isaarny V, Charpiou B, et all, **Providing Quality of Service over the Web: A Newspaper-based Approach**. Proceedings of the Sixth International World Wide Web Conference, Set 1997, pp 1457-1465.

[BAR 02] Bartel M, Boyer J, Fox B, et all, **XML-Signature Syntax and Processing**, W3C Recommendation, Fev 2002.

[BOX 01] D. Box, D Ehnebuske, Kakivaya, **Simple Object Access Protocol 1.1**, http://www.w3.org/TR/SOAP, W3C, mai 2000.

[BRA 99] Bray T, Hollander D, Laymann A, et all, **Namespaces in XML**, W3C Recommendation, Jan 1999.

[BRD 97] Braden R., Zhang, L., Berson, S., Herzog S., Jamin, S., **Resource ReSerVation Protocol (RSVP)** - Version 1 Functional Specification, Internet Engineering Task Force RFC 2205, Set 1997.

[CHA 00] Chandra S, Schlatter C, Vahdat A. **Application level differentiated multimedia web services using quality aware transcoding**. IEEE Journal on Selected Areas in Communications - Special Issue on QOS in the Internet – Dez 2000, Volume 18, Nr 2, pp. 2544-2565

[COM 00] Comer D, **Interligação em redes com TCP/IP**, Editora Campus 2000.

[DAM 02] Damiani E, Vimercati S, Paraboshi S, et all, **Fine Grained Access Control for SOAP E-Services,** ACM Transactions on Information and System Security (TISSEC), Volume 5, pags 169 – 202, Mai 2002.

[DAR 02] Darakhvelidze P, Markov E, **Web Services Development with Delphi**, 1a ed. Alist 2002.

[EVA 02] Evans S, Olwyn D, **Is SSL enough security for first-generation Web services?** , WebServices.org , Jul 2002.

[FOR 01] Ford W, Baker P, Fox B, et all, **XML Key Management Specification**, W3C Note, Mar 2001.

[FRY 01] Frystyk H, Thatte S, **Web Services Routing Protocol**, MSDN Out 2001 http://msdn.microsoft.com/library/default.asp? url=/library/en- us/dnglobspec/html/wsroutspecindex.asp.

[GHI 02] Ghisleri A, **Sistema Seguro de Atendimento ao Cliente Garantia da Qualidade de Serviço**, Universidade Federal de Santa Catarina, Dissertação de Mestrado, 2002.

[GIS 01] Gisolfi, D; **Is Web services the reincarnation of CORBA?** , IBM developer works 2001.

[IMA 01] Imamura T, Dillaway B, Schaad J, et all, **XML Encryption Syntax and Processing**, W3C Draft, Jun 2001.

[KAL 02] Kaleem M, Transactions over Web Services **- An Introduction to the Business Transaction Protocol** , WebServices.Org , Mai 2002.

[KAN 00] Kant K, Iyer R, Mohapatra P, **Arquitectural Impact of Secure Socket Layer on Internet Servers**, IEEE International Conference on Computer Design, Set - 2000, pp 7.

[RON 99] Ronda R H, **Security Service Level Agreements: Quantifiable Security for the Enterprise?,** Proceedings of the 1999 workshop on New security paradigm, ACM Press , pp 54-60, 1999.

[SCH 00] Schneier B, **Applied Cryptography**, Wiley 2000.

[SCU 97] Schumacher, H. J.; Ghosh, Sumit. **A fundamental framework for network security**. Journal of Network and Computer Applications 1997, Volume 20, pp 305–322.

[SOA 95] Soares, L. F. G, Lemos G, Colcher S, **Redes de Computadores: Das LANs,MANs e WANs às Redes ATM**. 7a. ed. Campus, Rio de Janeiro, RJ, 1995.

[STA 00] Stallings W - **Cryptography and network security**, Prentice Hall 2000.

[STU 01] Sturm R, Morris W, **Service Level Management**, Editora Campus 2001.

[TAN 97] Tanenbaum A, **Redes de Computadores**, Editora Campus 1997.

[VAS 00] Vasiliou N, Lutfiyya H. **Providing a Differentiated Quality of Service in a World Wide Web Server**, Desempenho Evaluation Review (ACM Sigmetrics), Volume 28, number 2, pags 22-27, 2000.

[WAR 03] Warnecke E, **G-DEF-Protocolo Criptográfico para Geração de Documento Eletrônico Fiscal nas Operações entre Empresas**, Universidade Federal de Santa Catarina, Dissertação de Mestrado, 2003.

APÊNDICE A

Este apêndice apresenta os resultados dos testes explicados na seção 5.5, com e sem o uso de RSVP. Pode-se constatar que, a partir da linha 133, na coluna de testes sem o uso de RSVP, os resultados sofrem alterações devido à perda de pacotes.

Resultado com o uso de RSVP	Resultado sem o uso de RSVP
0-0,000196944441995583	0-0,000197974535694811
1-0,000196944441995583	1-0,000198437497601844
2-0,000197060180653352	2-0,000198784720851108
3-0,000197060180653352	3-0,000199143520148937
4-0,000197060180653352	4-0,000199490743398201
5-0,000197060180653352	5-0,000199837959371507
6-0,000197060180653352	6-0,000200185182620771
7-0,000197060180653352	7-0,00020532405870035
8-0,000197060180653352	8-0,00020995367777068
9-0,000197060180653352	9-0,00021342591026332
10-0,000197060180653352	10-0,00021689814275596
11-0,000197060180653352	11-0,00020203703752486
12-0,000197175926587079	12-0,00020615738089662
13-0,000197175926587079	13-0,00020354166917968 5
14-0,000197175926587079	14-0,00020400463108671 8
15-0,000197291665244848	15-0,00020435185433598 2
16-0,000197291665244848	16-0,00020469907030928 9
17-0,000197291665244848	17-0,00020504629355855 3
18-0,000197407403902616	18-0,00020539351680781 7
19-0,000197407403902616	19-0,00020574074005708 1
20-0,000197407403902616	20-0,00020631944062188 3
21-0,000197407403902616	21-0,00020666666387114 7
22-0,000197523149836343	22-0,00020702502546316 8
23-0,000197523149836343	23-0,00020737268641823 9
24-0,000197523149836343	24-0,00020771990966750 3
25-0,000197523149836343	25-0,00020818287157453 6
26-0,000197638888494112	26-0,00020876157213933 8
27-0,000197638888494112	27-0,00020910879539860 2
28-0,000197638888494112	28-0,00020945601863786 6
29-0,000197638888494112	29-0,00020980324188713
30-0,00019775462715188	30-0,00021015046513639 4
31-0,00019775462715188	31-0,00021072916570119 6
32-0,00019775462715188	32-0,00021110763889504 6
33-0,00019775462715188	33-0,00021142361219972 4
34-0,000197870365809649	34-0,00021177083544898 8
35-0,000197870365809649	35-0,00021211805869825 2
36-0,000197870365809649	36-0,00021246527467155 8
37-0,000197870365809649	37-0,00021281249792082 2
38-0,000197986111743376	38-0,00021315972117008 6
39-0,000197986111743376	39-0,00021350694441935
40-0,000197986111743376	40-0,00021396990632638 3
41-0,000197986111743376	41-0,00021431712957564 7
42-0,000198101850401144	42-0,00021467592887347 6
43-0,000198101850401144	43-0,00021502314484678 2

44-0,000198101850401144	44-0,000215486114029773
45-0,000198217589058913	45-0,000215833330003079
46-0,000198217589058913	46-0,000216180553252343
47-0,000198217589058913	47-0,000216527776501607
48-0,000198217589058913	48-0,000216874999750871
49-0,00019833333499264	49-0,000217222223000135
50-0,00019833333499264	50-0,00021685184907168
51-0,00019833333499264	51-0,000218032408156432
52-0,00019833333499264	52-0,000218379631405696
53-0,000198449073650409	53-0,00021872685465496
54-0,000198449073650409	54-0,000219074070628267
55-0,000198449073650409	55-0,000219537039811257
56-0,000198449073650409	56-0,000219884255784564
57-0,000198449073650409	57-0,000220231479033828
58-0,000198564812308177	58-0,000220694440940861
59-0,000198564812308177	59-0,000221041664190125
60-0,000198564812308177	60-0,000221388887439389
61-0,000198680550965946	61-0,000221851849346422
62-0,000198680550965946	62-0,00022221064864425
63-0,000198680550965946	63-0,000222557871893514
64-0,000198680550965946	64-0,000222905095142778
65-0,000198796296899673	65-0,000223252311116084
66-0,000198796296899673	66-0,000223599534365349
67-0,000198796296899673	67-0,000224062496272381
68-0,000198796296899673	68-0,000224409719521645
69-0,000198912035557441	69-0,00022475694277091
70-0,000199837959371507	70-0,000225104166020174
71-0,000205636570171919	71-0,000225451389269438
72-0,00021143518097233	72-0,000225798612518702
73-0,000217222223000135	73-0,000226493051741272
74-0,000223020833800547	74-0,00022776620608056
75-0,000228819444600958	75-0,000233564816880971
76-0,00023461805540137	76-0,000239363427681383
77-0,000240405090153217	77-0,00024515046243323
78-0,000246203700953629	78-0,000250949073233642
79-0,00025200231175404	79-0,000256631945376284
80-0,000257800922554452	80-0,000262430556176696
81-0,000263587964582257	81-0,000268229166977108
82-0,000269386575382669	82-0,000274131940386724
83-0,00027518518618308	83-0,000279930558463093
84-0,000280983796983492	84-0,000285729169263504
85-0,000286782407783903	85-0,000291527780063916
86-0,000292569442535751	86-0,000297199076157995
87-0,000298368053336162	87-0,000303113425616175
88-0,000304166664136574	88-0,000308912036416586
89-0,000309965274936985	89-0,000314710647216998
90-0,000315752309688833	90-0,000320381943311077
91-0,000321550920489244	91-0,000326296292769257
92-0,000327349538565613	92-0,000332094903569669
93-0,000333148149366025	93-0,000337893521646038
94-0,000338935184117872	94-0,000343680556397885
95-0,000344733794918284	95-0,000349363428540528
96-0,000350532405718695	96-0,000355277777998708

97-0,000356331016519107	97-0,000360960650141351
98-0,000362118051270954	98-0,000366863423550967
99-0,000367916662071366	99-0,000372893518942874
100-0,000373715272871777	100-0,000378576391085517
101-0,000379513883672189	101-0,000384490740543697
102-0,000385300925699994	102-0,000390162036637776
103-0,000391099536500406	103-0,000395960647438187
104-0,000396898147300817	104-0,000401759258238599
105-0,000402696758101229	105-0,00040755786903901
106-0,000408483792853076	106-0,000413344903790858
107-0,000414282403653488	107-0,000419143514591269
108-0,000420081014453899	108-0,000424942132667638
109-0,000425879625254311	109-0,00043074074346805
110-0,000431678236054722	110-0,000436643516877666
111-0,000437465278082527	111-0,000442442127678078
112-0,000443263888882939	112-0,000448240738478489
113-0,00044906249968335	113-0,000454039349278901
114-0,000454861110483762	114-0,000459942129964475
115-0,000460648145235609	115-0,000465740740764886
116-0,000466446756036021	116-0,000471655090223067
117-0,000472245366836432	117-0,000477453701023478
118-0,000478043977636844	118-0,00048325231182389
119-0,000483831019664649	119-0,000489039353851695
120-0,00048962963046506	120-0,000494837964652106
121-0,000495428241265472	121-0,000500752314110287
122-0,000501226852065884	122-0,000506550924910698
123-0,000507013886817731	123-0,000512337959662545
124-0,000512812497618143	124-0,000518136570462957
125-0,000518611108418554	125-0,000523935181263369
126-0,000524409719218966	126-0,000530081015313044
127-0,000530196753970813	127-0,000536215280590113
128-0,000535995372047182	128-0,00054212963004829 4
129-0,000541793982847594	129-0,000548159718164243
130-0,000547592593648005	130-0,000554074074898381
131-0,000553379628399853	131-0,00056010416301433
132-0,000559178239200264	132-0,000566122682357673
133-0,000564976850000676	133-0,000589074072195217
134-0,000570775460801087	te de numero 135-0,000589305556786712
135-0,000576574071601499	0,00058953703410225
136-0,000582361106353346	136-0,000591631942370441
137-0,000588159717153758	137-0,000596261575992685
138-0,000593958327954169	138-0,00060194444085937
139-0,000599756946030539	139-0,000607743051659781
140-0,000605543980782386	140-0,000613530093687586
141-0,000611342591582797	141-0,000619444443145767
142-0,000617141202383209	142-0,000625243053946178
143-0,000622939813183621	143-0,000634629628621042
144-0,000628726847935468	144-0,000637071760138497
145-0,00063452545873588	145-0,000642627317574807
146-0,000640324069536291	146-0,000648425928375218
147-0,000646122680336703	147-0,00065422453917563
148-0,000651909722364508	148-0,000660254627291579
149-0,000657708333164919	149-0,000666157407977153

150-0,000663506943965331	150-0,000680995370203163
151-0,000669189816107973	0,000681111108860932
152-0,000674976850859821	152-0,000683784724969883
153-0,000680775461660232	153-0,000689108797814697
154-0,000686574072460644	154-0,000704062498698477
155-0,000692372683261055	0,000704293983289972
156-0,000698159718012903	156-0,000718437499017455
157-0,000703958328813314	te de numero 158-0,00071866898360895
158-0,000709756939613726	0,000718784722266719
159-0,000715555550414138	159-0,000742314812669065
160-0,000721354161214549	te de numero 161-0,00074254629726056
161-0,000727141203242354	linha teste de numero 162-0,000742662035918329
162-0,000732939814042766	0,000742893520509824
163-0,000738738424843177	163-0,000757962960051373
164-0,000744537035643589	0,000758194444642868
165-0,000750324070395436	165-0,000760983799409587
166-0,000756122681195848	166-0,000764571756008081
167-0,000761921291996259	167-0,000770254628150724
168-0,000767719902796671	168-0,000775821761635598
169-0,000773506944824476	169-0,000781724535045214
170-0,000779305555624887	170-0,000787523145845626
171-0,000785104166425299	171-0,000793437502579764
172-0,00079090277722571	172-0,000799236113380175
173-0,000796689811977558	173-0,000805254632723518
174-0,000802488422777969	174-0,000810821758932434
175-0,000808287033578381	175-0,000828090276627336
176-0,000814085644378792	0,000828321761218831
177-0,000819872686406597	177-0,000851388889714144
178-0,000825671297207009	te de numero 179-0,000851504628371913
179-0,000831469908007421	linha teste de numero 180-0,000851631943078246
180-0,000837268518807832	0,000851863427669741
181-0,00084305555355968	181-0,00085197916632751
182-0,000848854164360091	182-0,000857418977830093
183-0,000854652775160503	183-0,000863101849972736
184-0,000860451385960914	184-0,000868784722115379
185-0,000866249996761326	185-0,000874699071573559
186-0,000872037038789131	186-0,000880486113601364
187-0,000877835649589542	187-0,000891851850610692
188-0,000883634260389954	188-0,000905763889022637
189-0,000889432871190365	te de numero 190-0,000906111112271901
190-0,000895219905942213	0,00090622685092967
191-0,000901018516742624	191-0,000909467591554858
192-0,000906817127543036	192-0,000915266202355269
193-0,000912615738343447	193-0,000921064813155681
194-0,000918402773095295	194-0,000926851847907528
195-0,000924201383895706	195-0,000932650465983897
196-0,000929999994696118	196-0,000938564815442078
197-0,000935798612772487	197-0,000944594903558027
198-0,000941585647524334	198-0,000950150460994337
199-0,000947384258324746	199-0,000956643518293276
200-0,000953182869125158	200-0,00096174768259516