

**3^a Escola Regional de Redes de Computadores
ERRC 2005**

**18 a 20 de agosto de 2005
Santa Cruz do Sul, RS – Brasil**

Edição
Cristiano Bonato Both
Valter Roesler

Promoção
SBC – Sociedade Brasileira de Computação

Organização
Universidade de Passo Fundo – UPF
Universidade de Santa Cruz do Sul – UNISC
Universidade do Vale do Rio dos Sinos - UNISINOS
Universidade Federal do Rio Grande do Sul - UFRGS

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Escola Regional de Redes de Computadores (3.: 2005 Agosto 18 a 20: Santa Cruz do Sul, RS, Brasil).

Anais/Editores Cristiano Bonato Both, Valter Roesler, Santa Cruz do Sul: UNISC: Departamento de Informática, 2005.

ISBN 85-7669-040-3

ERRC 2005

1. Redes de Computadores. I. Both, Cristiano Bonato. II. Roesler, Valter. III. ERRC (3.: 2005: Santa Cruz do Sul).

APRESENTAÇÃO

Bem-vindo à 3^a ERRC!

É com grande satisfação que apresentamos a terceira edição da Escola Regional de Redes de Computadores – ERRC, um evento anual que proporciona o encontro de estudantes, professores e profissionais que atuam na área de redes de computadores no Estado do Rio Grande do Sul.

Cada edição a escola vem se firmando como uma excelente oportunidade para a qualificação de estudantes e profissionais, para a apresentação de trabalhos e discussões de idéias acerca da área de redes de computadores nas universidades gaúchas.

Em 2005, a ERRC está sendo organizada pela Universidade de Santa Cruz do Sul (UNISC) e possuindo a colaboração da Universidade Federal do Rio Grande do Sul (UFRGS), Universidade de Passo Fundo (UPF), Centro Universitário La Salle (UNILASALLE), Universidade do Vale do Rio dos Sinos (UNISINOS), Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS) e Fundação Universidade Federal do Rio Grande (FURGS). Como nas demais edições, conta com o apoio da Sociedade Brasileira de Computação (SBC).

O presente volume contém os trabalhos técnicos aceitos pelo comitê de avaliação da terceira Escola de Regional de Redes de Computadores, o resumo dos minicursos e oficinas.

A comissão organizadora agradece a todos os participantes, divulgadores e aos patrocinadores (TechDec, TellFree e Microsoft).

Desejamos a todos uma excelente e proveitosa escola.

Cristiano Bonato Both
Coordenador geral da 3^a ERRC
Santa Cruz do Sul, agosto de 2005

COMITÊ

Coordenação Geral

Cristiano Bonato Both (UNISC)

Coordenação do Comitê de Programa

Valter Roesler (UNISINOS)

Coordenação de Minicursos

Marco Antônio Trentin (UPF)

Coordenação de Palestras

Juergen Rochol (UFRGS)

Cristina Moreira Nunes (UNILASALLE/PUCRS)

Coordenação de Salão de Ferramentas

Weldson Queiroz de Lima (UFRGS)

Clarissa Cassales Marquezan (UFRGS)

Comitê de Organização

Alisson Ceolin (UNISC)

Alexandra Aguiar (UNISC)

Cristiano Bonato Both (UNISC)

Fábio Schwengber (UNISC)

Jorge Luis Staub (UNISC)

Juergen Rochol (UFRGS)

Lisandro Zambenedetti Granville (UFRGS)

Rafael Ramos dos Santos (UNISC)

Rita de Cássia Rocha (UNISC)

Comitê de Programa

Ana Cristina Silva (PUC)

Avelino Zorzo (PUCRS)

Carlos Schaeffer (UPF)

Cristian Koliver (UCS)

Cristiano Both (UNISC)

Cristina Nunes (UNILASALLE/PUCRS)

Eduardo Leivas Bastos (FEEVALE)

Fabio Zanin (URI)

Fernando Dotti (PUCRS)

Gaspare Bruno (UFRGS)

Gerson Battisti (UNIJUI)

João Netto (UFRGS)

José Valdeni de Lima (UFRGS)

COMITÊ

Juergen Rochol (UFRGS)
Katia Saikoski (HP Brasil)
Liane Tarouco (UFRGS)
Lisandro Zambenedetti Granville (UFRGS)
Luciano Gaspary (UNISINOS)
Luís Felipe Balbinot (SIM Telecom e PRAV/UNISINOS)
Marco Antônio Trentin (UPF)
Marcos Barreto (UNILASALLE)
Marinho Barcellos (UNISINOS/University of Manchester)
Nelson Duarte Filho (FURG)
Taisy Weber (UFRGS)
Tasso Faria (UNILASALLE)
Valter Roesler (UNISINOS)
Vinicius Ribeiro (UNILASALLE/UNIRITTER/FACENSA)

Comitê de Mini-cursos

André Luís Fávero (UFRGS)
Carlos Adriani Lara Schaeffer (UPF)
Roseclea Duarte Medina (UFSM)
Vinícius Serafim (UFRGS)

SUMÁRIO

SESSÃO TÉCNICA 1

Uma Proposta de um Sistema para a Prevenção de Intrusões em Redes de Computadores

Guilherme Linck, Diego Luís Kreutz (UFSM) 3

Distribuições de Probabilidade na Descrição de Carga de Falhas para Injetores de Falhas de Comunicação

Juliano C. Vacaro, Gabriela Jacques-Silva, Taisy SilvaWeber, Ingrid Jansch-Pôrto (UFRGS) 9

Geração de logs de Experimentos de Injeção de Falhas para Análise de Dependabilidade de Aplicações Distribuídas

Joana M. F. Trindade, Gabriela Jacques-Silva, Taisy SilvaWeber, Ingrid Jansch-Pôrto (UFRGS) 15

Tolerância a falhas no modelo MultiCluster

Marcos Locateli Gontarski, Marcos Ennes Barreto (UNILASALLE) 21

Modelagem de Requisitos de QoS Utilizando UML-RT

Edison Pignaton de Freitas, Elias Teodoro da Silva Jr, Fabiano Costa Carvalho (UFRGS) 27

SESSÃO TÉCNICA 2

Segurança na troca de informações em uma rede sem fio no modo Ad Hoc

Floriano Ferreira Dos Reis Filho, Héctor Dave Orrillo Ascama, Sergio Takeo Kofuji (USP) 35

Definindo e Usando Contexto Derivado de Multi-Sensores

Felipe Weber Fehlberg, Cláudio Fernando Resin Geyer, Iara Augustin, Adenauer Corrêa Yamin, Luciano Cavalheiro da Silva (UFRGS) 41

Uma Proposta de Mecanismo para Controle de Admissão para a Rede Backbone

UMTS Empregando Serviços Diferenciados

Paulo Dias de Alecrim, Paulo Roberto Guardieiro (UFU) 47

Design and Simulation of the IEEE 802.16a physical layer

K. Jalandhar Reddy, S.Srikanth (ANNA UNIVERSITY) 53

Utilização de Agentes em um Modelo de Autorização para Acessos a Dados Médicos em Ambiente de Computação Móvel

Erico H. do Amaral, Gerson A. Soares, Raul C. Nunes, Roger C. Machado (UFSM) 61

SESSÃO TÉCNICA 3

Integração de Sistemas Embutidos utilizando Web Services

Guilherme Bertoni Machado, Frank Siqueira (UFSC) 69

TCP Reno, TCP Vegas e TCP Westwood: Uma Comparação de Desempenho

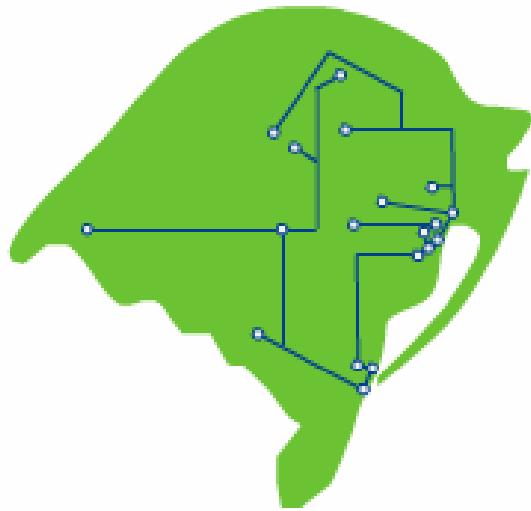
Juliana de Santi, Michele Mara de Araújo Espíndula Lima (UNIOESTE) 75

Análise do desempenho dos protocolos UDP e TCP entre os Kernels 2.4.27 e 2.6.9 do Linux em Redes Gigabit Ethernet

Carlos Brites, Gaspare Bruno (UNILASALLE) 81

SUMÁRIO

Uma Proposta de Escalonamento Descentralizado de Tarefas para Computação em Grade	
<i>Lucas Alberto Souza Santos (UFRGS), Patrícia Kayser Vargas (UNILASSALLE/COPPE), Cláudio F. R. Geyer (UFRGS)</i>	89
Projeto de Implementação dos Protocolos de Transporte no Modelo TCP/IP em Placas FPGAs	
<i>Alisson Roggia Ceolin, Cristiano Bonato Both (UNISC)</i>	95
Desenvolvimento de Agente SNMP Embocado para Microprocessador Rabbit	
<i>Elvis Lopes Monteiro, Westter José da Silva Santos, Dr. Cláudio Afonso Fleury (CEFET-GO)</i>	101
 MINICURSOS	
Dotando serviços de segurança em Windows e Linux	
<i>Vinicius G. Ribeiro e Fernando Patzlaff (UNILASALLE)</i>	109
VoIP - Transmissão de Voz em Redes IP	
<i>João Netto (UFRGS)</i>	109
Redes Sem-fio e Sistemas Celulares	
<i>Juergen Rochol (UFRGS)</i>	110
 OFICINAS	
Segurança em ambientes Linux	
<i>Vinicius G. Ribeiro e Fernando Patzlaff (UNILASALLE)</i>	113
Analise de segurança em uma rede wireless	
<i>Evandro Franzen e Roberto Ely Scherer (UNISC)</i>	113
Programando Web Services para Gerenciamento de Redes	
<i>Lisandro Zambenedetti Granville, Clarissa Cassales Marquezan, Ricardo Lemos Vianna, Weldson Q. de Lima (UFRGS)</i>	114



SESSÃO TÉCNICA 1

Uma Proposta de um Sistema para a Prevenção de Intrusões em Redes de Computadores

Guilherme Linck, Diego Luís Kreutz

¹Núcleo de Ciência da Computação — NCC
Universidade Federal de Santa Maria — UFSM

{linck,kreutz}@inf.ufsm.br

Abstract. This article presents the architecture of a simple system, that can easily be distributed, for intrusions prevention on computer networks. The basic idea is to adapt monitoring modules, to resources as package filters, that are able to identify and block probable attackers.

Resumo. Este artigo apresenta a proposta de um sistema simples, que pode ser facilmente distribuído, para a prevenção de intrusões em redes de computadores. A idéia básica é adaptar módulos de monitoramento, a recursos como filtros de pacotes, que sejam capazes de identificar e barrar prováveis atacantes.

1. Introdução

Atualmente existem diversas ferramentas que auxiliam na detecção de intrusão, porém, a área de prevenção de intrusões é relativamente nova e carece de soluções [Desai 2003, Holland 2004]. Neste contexto, este trabalho apresenta uma proposta de um sistema que busca auxiliar de forma simples e prática a prevenção de intrusões.

A idéia básica é fazer uso dos filtros de pacotes existentes, acrescentando monitores de *log* que irão analisar os registros armazenados e tomar as medidas preventivas cabíveis para interromper uma ação suspeita. Outro objetivo é permitir um controle integrado de diversos sistemas de uma rede local ou corporativa, possibilitando a identificação e prevenção distribuída de prováveis tentativas de ataque contra sistemas.

O artigo está estruturado da seguinte forma: a seção seguinte contém alguns trabalhos relacionados. A arquitetura do proposta é apresentada na seção 3. Na seção 4. é abordado o estado atual. Finalizando, seguem a conclusão e os trabalhos futuros.

2. Trabalhos Relacionados

Existem basicamente duas grandes classes de sistemas para a segurança em uma rede de computadores. A primeira delas, mais antiga, é composta pelos sistemas de detecção de intrusão (IDS¹). Estes sistemas permitem um monitoramento e análise de uma vasta gama de dados e ações em sistemas ou contra a própria rede. Os dados coletados permitem a detecção de anomalias na rede ou em sistemas locais.

Uma segunda técnica que vem crescendo nos últimos anos é a prevenção de intrusões [Holland 2004]. Segundo uma das definições encontradas para essa classificação mais recente [Desai 2003], um sistema de prevenção de intrusões (IPS²) pode ser a combinação de um *firewall* e um IDS especializado na análise de pacotes de rede. A

¹Intrusion Detection Systems

²Intrusion Prevention System

idéia básica é prever ataques conhecidos ou desconhecidos e tentar impedir a realização dos mesmos.

Ferramentas como o *firestarter* [Junnonen 2000], e *portsentry* [Rowland 2002] costumam ser utilizadas para a proteção de estações de trabalho contra possíveis ataques pela rede. Algumas destas ferramentas possuem interface gráfica, facilitando a manipulação de regras e controle geral de acesso por parte do usuário. Outros sistemas como o *firestorm* [Leach and Tedesco 2004], *Snort* [Caswell and Roesch 2004] e *prelude-nids* [Prelude Trac 2005] são destinados ao monitoramento de serviços e detecção de intrusões e anomalias em máquinas ou na rede. A funcionalidade básica é identificar situações conhecidas e desconhecidas (anormais/suspeitas), gerando relatórios e alertas de estado da rede ou tomando medidas preventivas baseadas em políticas pré-definidas. Dentro deste contexto, algumas dessas ferramentas possuem funcionalidades como bloqueio e restrição de acesso, impedindo ações posteriores de máquinas suspeitas. Esse é o caso do *portsentry* e o *Snort-inline*.

O sistema aqui proposto tem como abordagem principal a prevenção, de forma centralizada ou distribuída/colaborativa, de intrusões à redes de computadores ou sistemas locais.

3. A Arquitetura Proposta

A idéia básica é criar um sistema prático e útil para a prevenção de intrusões em redes de computadores. Uma das metas é manter a arquitetura tradicional de sistemas de segurança, como filtros de pacotes, e acrescentar módulos auxiliares que irão incrementar funcionalidade nesses sistemas, de modo a impedir, proteger, ou desviar o trabalho (que pode ser uma tentativa de ataque) de endereços suspeitos.

A figura 1 apresenta uma visão geral da base do sistema. Esta é composta por: (1) um filtro de pacotes; (2) um arquivo de registros de atividade sobre as políticas de acesso ou tráfego na rede; e (3) um monitor de atividades registradas no arquivo de log e gerenciador de políticas de controle de acesso e roteamento do filtro de pacotes.

Os três componentes básicos executam na mesma máquina. Esta pode ser uma simples estação de trabalho, um servidor ou um *firewall* da rede.

O objetivo do sistema é tentar identificar e bloquear máquinas suspeitas, que possam ser prováveis atacantes à rede. A primeira função cabe ao monitor de registros do filtro de pacotes. Em uma primeira instância, através da análise das tentativas de conexões não

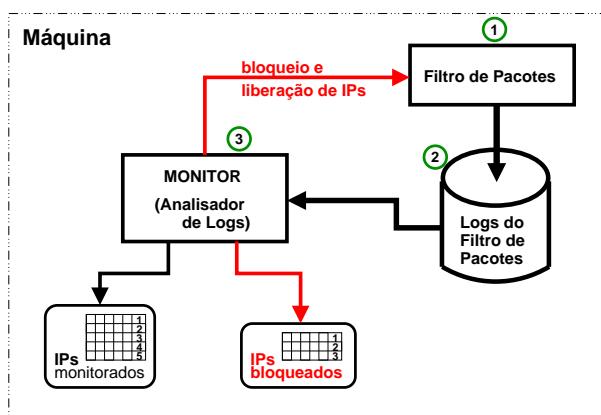


Figure 1. Visão geral do sistema em uma máquina

autorizadas [Kreutz 2004] o monitor irá classificar os computadores, tanto da rede externa quanto da rede interna. Todos os IPs das máquinas que infringiram alguma regra do filtro de pacotes são colocadas em uma tabela geral de monitoramento. A partir desse ponto o agente monitor começa a analisar o número e a freqüência de conexões não autorizadas de cada endereço de origem. A meta é identificar endereços com uma alta probabilidade de serem fontes de futuros ataques e bloqueá-los, ou desviá-los a um *honeypot*.

Uma máquina é considerada suspeita quando atingir o limite máximo de infrações permitidas [Kreutz 2004]. Neste caso, infrações estão relacionadas às entradas do arquivo de *logs* que registram tentativas de conexão, a serviços ou máquinas, negadas (não autorizadas). Outra forma de um IP ser enquadrado como uma possível ameaça à rede é quando a probabilidade é bastante grande de o endereço de origem estar realizando uma varredura de portas ou máquinas da rede. Uma terceira maneira de um endereço ser considerado suspeito é quando sua atividade denota freqüências ou comportamentos de tempos de conexão (segundo regras autorizadas) constantes, anômalos, ou reconhecidas como táticas de invasão (a partir da análise de casos anteriores).

Quando uma máquina entra para a lista de prováveis atacantes (ou lista negra do sistema) ela é automaticamente colocada em estado de observação, que pode ser: bloqueio no filtro de pacotes, ou redirecionamento dos pacotes para um *honeypot*. Com isso, espera-se impedir que muitos ataques, em relação a redes sem nenhum mecanismo de prevenção desse gênero, sejam iniciados, prossigam ou tenham sucesso.

Um endereço IP é retirado da lista negra quando satisfizer a seguinte condição: não ter cometido novas infrações nos últimos X intervalos de tempo. Esses intervalos de tempo são parâmetros de configuração do sistema, determinados pelo administrador da rede.

Para o funcionamento dessa arquitetura base da proposta são necessários os seguintes requisitos: (1) número mínimo de infrações para que um endereço seja enquadrado como suspeito; (2) intervalo de tempo de monitoramento³; (3) número mínimo de conexões negadas à n diferentes máquinas ou portas em um determinado intervalo tempo (normalmente reduzido); (4) valor de n contido no requisito anterior; (5) intervalo de tempo contido no requisito 3; (6) tempo mínimo de bloqueio⁴;

O primeiro requisito define o número mínimo de infrações para considerar uma máquina como uma possível ameaça à rede. Todas as máquinas que venham a cometer mais infrações que o número mínimo definido, no intervalo de tempo estipulado, entrarão para o estado de observação. Além disso, o administrador do domínio do endereço de origem (quando definido no DNS) será notificado por e-mail do fato de uma máquina sob sua jurisdição ter sido considerada suspeita e estar em observação. Isso possibilita que administradores tomem providências e verifiquem o fato.

A definição do número mínimo de infrações pode ser feita de acordo com análises do tráfego e ações contra a rede local ou sistema em monitoramento[Kreutz 2004]. Cada grupo de máquinas pode possuir seu próprio número mínimo de infrações. Onde, por exemplo, servidores da rede tem um nível de tolerância maior, evitando bloqueios por problemas de má configuração ou estado de teste dos mesmos.

As infrações de cada endereço IP são cumulativas e verificadas a cada intervalo de

³Tempo durante o qual o número mí nimo de infrações deve acontecer

⁴Tempo em que o fi caí bloqueado ou terá seu tráfego desviado a uma máquina isca

monitoramento. A definição destes intervalos pode ser determinada e programada pelo nível de atividade do sistema [Kreutz 2004]. Com excessão dos casos que se enquadram nos requisitos 3 e 4. Estes casos são verificados com uma periodicidade maior, dinâmica.

O terceiro requisito remete ao número mínimo de conexões negadas oriundas de um endereço IP qualquer, cujo destino é um determinado número de máquinas ou portas diferentes em um espaço de tempo reduzido. É o caso comum de escaneamentos de endereços da rede ou portas de máquinas. Um escaneamento é normalmente encarado como uma primeira premissa de um possível ataque ou ação mal intencionada na rede.

O quarto requisito consiste em definir o número de portas ou máquinas diferentes necessário para verificar se um endereço IP infringe ou não o terceiro requisito. Esta variável poderá ter um valor reduzido, visto que um escaneamento é comumente realizado em um período curto de tempo. Por exemplo, é estabelecido que 10 tentativas de conexão não autorizadas com portas ou máquinas distintas em um período de 0 à 60 segundos é enquadrado como uma infração às políticas do sistema. Assim que as 10 tentativas forem detectadas, em um período de no máximo 1 minuto, o endereço irá para o estado de observação. O intervalo de 1 minuto é definido pelo próximo requisito do sistema, que pode também ser infinito.

O último requisito remete ao tempo mínimo que um endereço IP ficará em observação no sistema. Após transcorrido esse tempo e nenhuma nova infração ter sido registrada, o endereço é liberado. No entanto, entra para um segundo estado de observação. Neste estado, o limite de infrações mínimas, e o período de tempo para verificação destas, é reduzido em $Y\%$ (definidos pelo usuário). Essa medida tem como objetivo evitar casos em que o endereço em questão está mais cauteloso, tentando reduzir sua periodicidade, número de varreduras ou tentativas de ataque a sistemas da rede. A cada período de tempo (requisito 2) essa maior sensibilidade de monitoramento irá sendo reduzida $Z\%$ (parâmetro de configuração do sistema), até atingir o estado normal.

3.1. Controle Distribuído

Outra meta é permitir uma prevenção distribuída e integrada. A simples detecção de um endereço suspeito deverá ser rapidamente propagada para as demais instâncias do sistemas, possibilitando uma ação conjunta e mais efetiva, evitando possíveis futuros ataques e reduzindo a eficácia de ataques em andamento. A figura 2 ilustra essa arquitetura distribuída de prevenção e controle de intrusões.

Em uma rede local, por exemplo, essa estrutura pode ser utilizada para impedir ações maliciosas internas à rede. Sabe-se que os ataques mais perigosos comumente são os internos [Holland 2004]. Logo, mecanismos que auxiliam no controle cooperativo das diretrizes de segurança de servidores e máquinas da rede podem ser uma boa política para aumentar a efetividade na prevenção e combate desses ataques.

A inclusão de um IP na lista de observação, em um dos pontos de prevenção de intrusões, irá implicar o monitoramento deste endereço em todos os pontos que fazem parte do sistema distribuído de prevenção de ataques. Ao mesmo tempo, a liberação de um determinado endereço será propagada aos demais membros da rede de segurança preventiva. No entanto, a liberação só ocorrerá mediante a satisfação dos pré-requisitos de liberação do sistema, em cada ponto de controle, vistos anteriormente.

3.2. Exemplo de Aplicação

A figura 3 apresenta um caso de uso da arquitetura distribuída. É ilustrada uma empresa com filiais espalhadas geograficamente.

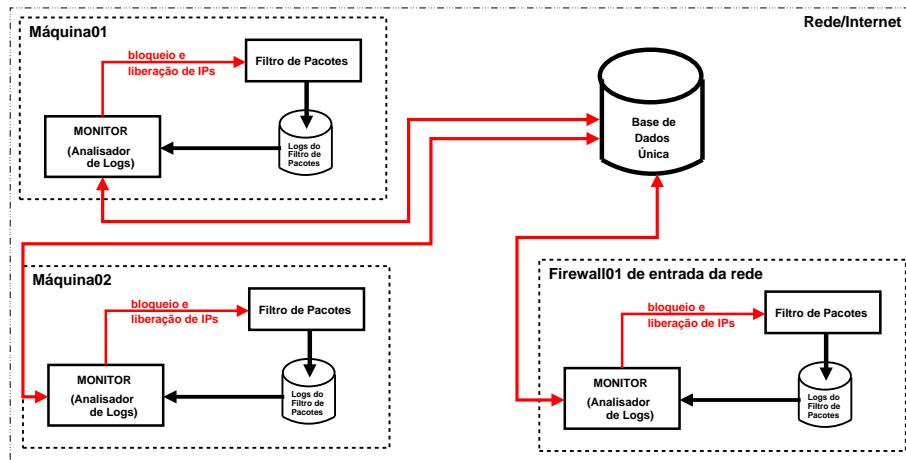


Figure 2. Visão geral do sistema em uma rede

Cada unidade da empresa contém um ponto central de controle de acesso à rede privada. O monitoramento é realizado sobre os *logs* do filtro de pacote dos pontos de acesso. Uma cópia atualizada das tabelas de infratores é mantida na base de dados distribuída, permitindo o compartilhamento de informações entre as filiais.

A detecção de um provável ataque em uma das unidades da empresa irá ser propagada para as demais. Com isso, tentativas de ataques subsequentes podem ser evitadas. Os sistemas de segurança podem ser facilmente colocados em estado de alerta mesmo antes de qualquer monitoramento suspeito local ter sido identificado. Com isso, um ataque inicial, que visasse a propagação para as demais unidades da rede privada da corporação, seria rapidamente denunciado, tendo suas possibilidades de continuidade ou sucesso bastante reduzidas.

4. Estado Atual

A versão inicial do sistema (segundo a proposta básica - figura 1) foi implementada e testada na rede do Núcleo de Ciência da Computação (NCC) [Kreutz 2004]. Os resultados se mostraram promissores.

A implementação atual funciona apenas com o IPTables, devidamente configurado [Kreutz 2004]. Esta escolha deve-se ao fato deste ser o filtro de pacotes mais difundido e utilizado por administradores de segurança.

A versão completa do sistema, segundo a proposta aqui apresentada, está em

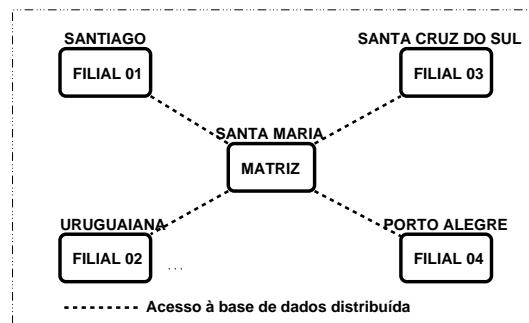


Figure 3. Ilustração de aplicação distribuída do sistema

fase de desenvolvimento. A linguagem de programação Perl está sendo utilizada para a codificação, enquanto que na primeira versão havia sido utilizado *shell scripts* [Kreutz 2004], e o gerenciador de diretórios *online* OpenLDAP para o acesso distribuído e transparente às informações compartilhadas pelas diversas instâncias do sistema. O uso da linguagem Perl tem demonstrado uma melhora significativa no desempenho do sistema, comparado a versão original.

5. Conclusão

A prevenção de intrusões é uma área recente e que ainda demanda pesquisa e soluções. O sistema aqui proposto vem contribuir para o desenvolvimento dessa área.

A arquitetura proposta é simples e prática para o controle preventivo de possíveis invasões. O funcionamento distribuído também torna a ferramenta interessante para uma variedade diversificada de cenários.

Os resultados obtidos até o momento tem se mostrado promissores e satisfatórios. A aplicação e teste na rede do NCC apresentou uma demonstração inicial de sua viabilidade e utilidade. Novos resultados são esperados com os trabalhos futuros.

Trabalhos Futuros. Finalização da implementação da versão completa do sistema, apresentada no decorrer do texto. Inclusão de mecanismos, de identificação de máquinas suspeitas, baseados em freqüência e comportamentos de conexões bem sucedidas (autorizadas). Testes do sistema em um ambiente distribuído. Melhoramentos na arquitetura e implementação. Desenvolvimento de ferramentas para a visualização de estatísticas dos índices do sistema.

References

- Caswell, B. and Roesch, M. (2004). Snort - The Open Source Network Intrusion Detection System. <http://www.snort.org/>. Último acesso: maio de 2005.
- Desai, N. (2003). Intrusion Prevention Systems: the Next Step in the Evolution of IDS. <http://www.securityfocus.com/infocus/1670>. Último acesso: maio de 2005.
- Holland, T. (2004). Understanding IPS and IDS: Using IPS and IDS together for Defense in Depth. <http://www.sans.org/rr/whitepapers/detection/1381.php>. Último acesso: maio de 2005.
- Junnonen, T. (2000). Firestarter. <http://www.fs-security.com/>. Último acesso: maio de 2005.
- Kreutz, D. (2004). Controle Independente de Restrições de Acesso a Redes Locais em Firewalls que Utilizam o IPTables. In *III Simpósio de Informática da Região Centro do RS*. UNIFRA.
- Leach, J. and Tedesco, G. (2004). Firestorm NIDS. <http://www.scaramanga.co.uk/firestorm/>. Último acesso: maio de 2005.
- Prelude Trac (2005). Prelude 0.9 Handbook. <https://trac.prelude-ids.org/wiki/PreludeHandbook>. Último acesso realizado em maio de 2005.
- Rowland, C. H. (2002). Sentry Tools: portsentry. <http://sourceforge.net/projects/sentrytools/>. Último acesso: maio de 2005.

Distribuições de Probabilidade na Descrição de Carga de Falhas para Injetores de Falhas de Comunicação *

Juliano C. Vacaro, Gabriela Jacques-Silva, Taisy Silva Weber, Ingrid Jansch-Pôrto

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{jcvacaro, gjsilva, taisy, ingrid}@inf.ufrgs.br

Abstract. *Fault injection is an efficient technique for evaluating systems and fault tolerance mechanisms. In order to obtain accurate results from fault injection based experiments, it is important to reproduce real conditions in the test environment. The aim of this paper is to extend FIONA to provide the use of probabilistic distributions and obtain faultloads that can simulate the behavior observed in real system environments.*

Resumo. *Injeção de falhas é uma técnica eficiente na validação do funcionamento de sistemas computacionais, bem como dos mecanismos de tolerância a falhas. Para que os resultados de experimentos baseados nesta técnica sejam válidos, é importante que o ambiente de teste reproduza as condições observadas na realidade. Este artigo tem por objetivo expandir a ferramenta de injeção de falhas FIONA para prover o uso de distribuições de probabilidade e assim obter cenários de falhas capazes de simular o comportamento observado na realidade.*

1. Introdução

Aplicações devem ser projetadas para operar em ambientes onde falhas podem ocorrer. Mecanismos de tolerância a falhas são a melhor garantia para prover dependabilidade (habilidade de prover serviços que possam ser justificadamente confiáveis [Avizienis et al. 2004]) a estes sistemas. A validação de tais aplicações não é tarefa trivial. Essa tarefa se torna ainda mais crítica em sistemas distribuídos onde, além do comportamento sob falhas de cada nó do sistema, também a sua reação a falhas no subsistema de troca de mensagens deve ser determinada. Deste modo, procedimentos convencionais de teste de cada nó isolado não são suficientes para garantir propriedades como confiabilidade e disponibilidade de todo o sistema ou estabelecer medidas como cobertura de falhas e queda de desempenho sob falhas.

Ferramentas de injeção de falhas mostram-se um método eficiente para a validação de sistemas. A abordagem visa a emulação de falhas e a análise do comportamento do sistema na presença destas falhas. FIONA é uma ferramenta de injeção de falhas de comunicação que suporta a criação de cenários de falhas para condução de experimentos. A manifestação de falhas pode ser definida baseada em distribuição uniforme. Porém, em ambientes de rede, nem todas as aplicações seguem este padrão de manifestação de erros

*Financiado pelos projetos ACERTE/CNPq (472084/2003-8) e DepGriFE/HP P&D Brasil

de comunicação [Yoma et al. 2005]. O uso de modelos probabilísticos capazes de reproduzir tais características é relevante para a criação de cenários de falhas mais significativos na condução de experimentos.

Este artigo tem por objetivo descrever a expansão da ferramenta FIONA para possibilitar a injeção de falhas baseadas em distribuições de probabilidade, proporcionando maior flexibilidade e generalidade na elaboração de cenários de falhas. Esta expansão permite que experimentos sejam conduzidos de modo a refletir a ocorrência de falhas observadas em ambientes reais.

A Seção 2. mostra as principais características dos injetores de falhas. A Seção 3. descreve a arquitetura da ferramenta FIONA. Detalhes da implementação para a realização do trabalho proposto são mostrados na Seção 4. Considerações finais são feitas na Seção 5.

2. Injetores de Falhas

Injeção de falhas é uma técnica eficiente para o teste de estratégias de tolerância a falhas implementadas em sistemas. Nesta técnica, falhas são inseridas no ambiente sob teste e o comportamento do sistema alvo na presença de falhas é observado. As técnicas de injeção de falhas são classificadas em injeção de falhas por simulação, por *hardware* e por *software* [Hsueh et al. 1997]. Na injeção por *software* falhas são inseridas no ambiente de execução das aplicações em teste. Esta abordagem tem como característica a flexibilidade e o baixo custo, porém, o nível de perturbação gerado ao ambiente de teste deve ser levado em consideração.

Na próxima Seção será descrita a ferramenta FIONA, que utiliza a técnica de injeção de falhas por *software* para inserir falhas de comunicação em sistemas distribuídos.

3. FIONA - *Fault Injector Oriented to Network Applications*

FIONA (*Fault Injector Oriented to Network Applications*) é uma ferramenta baseada em JVMTI desenvolvida para a condução de experimentos de injeção de falhas de comunicação em aplicações Java. FIONA pode ser usado para a verificação do funcionamento correto de mecanismos de tolerância a falhas em aplicações Java distribuídas de maneira fácil e flexível [Jacques-Silva et al. 2004].

A ferramenta FIONA permite a execução em ambientes locais e distribuídos. Neste trabalho será focada a arquitetura local de FIONA, devido a facilidade na condução de experimentos para a validação da implementação proposta. É importante salientar que as alterações realizadas na ferramenta são válidas tanto no ambiente de operação local quanto no ambiente de operação distribuído.

Nas próximas Seções são descritas as características da ferramenta FIONA. Na Seção 3.1. é mostrado o modelo de falhas adotado pela ferramenta. A arquitetura local de FIONA é mostrada na Seção 3.2. A Seção 3.3. descreve o mecanismo de construção de cenários de falhas.

3.1. Modelo de Falhas

O modelo de falhas para sistemas distribuídos assumido em FIONA é baseado no definido por Birman [Birman 1996], que descreve falhas de colapso (*halt*), parada segura (*fail-*

stop), omissão na transmissão e na recepção, rede, particionamento de rede, temporização e bizantinas. Como o foco da ferramenta é o teste de sistemas distribuídos de larga escala, foram consideradas as falhas mais comuns nesse ambiente. FIONA emula falhas de colapso, de omissão, de temporização, de duplicação e de particionamento de redes.

FIONA injeta falhas em aplicações que usam o protocolo UDP. Apesar desse protocolo ser não confiável, é comumente usado como base para implementação de protocolos de nível mais alto usados em aplicações com características de tolerância a falhas. Como exemplo pode-se citar o *middleware* de comunicação em grupo JGroups [Ban 1998], que por padrão usa UDP como base para sua pilha de protocolos.

3.2. Arquitetura Local

FIONA realiza a injeção de falhas através da instrumentação das classes de comunicação do sistema. Falhas em comunicação usando o protocolo UDP são ativadas através da instrumentação da classe `java.net.DatagramSocket`, que contém os métodos `send()` e `receive()` para envio e recebimento de datagramas. O injetor é composto de três partes: o agente JVMTI, as classes de sistema instrumentadas e as classes auxiliares. O agente JVMTI realiza a instrumentação das classes de sistema e mantém a comunicação com os demais injetores da rede; as classes instrumentadas realizam a injeção das falhas; as classes auxiliares armazenam a configuração do experimento e os logs de monitoramento. A Figura 1 ilustra a arquitetura local do injetor.

Assim, toda a vez em que os métodos `send()` e `receive()` são invocados, a classe instrumentada `java.net.DatagramSocket` acessa as funcionalidades das classes auxiliares de injeção de falhas para decidir quais falhas devem ser aplicadas à mensagem sendo enviada ou recebida.

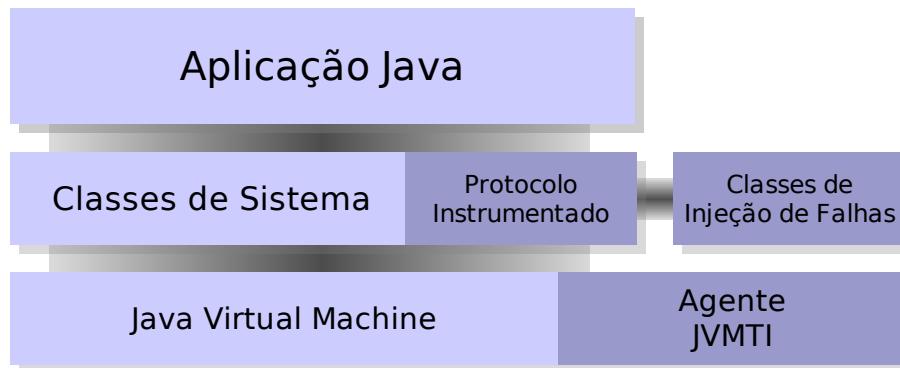


Figura 1. Arquitetura local do injetor FIONA

3.3. Cenário de Falhas

FIONA ainda permite que sejam definidos cenários de falhas. Através destes cenários é possível definir a carga de falhas que será injetada durante a condução dos experimentos. Ao final da execução de um experimento a ferramenta gera informações sobre as falhas injetadas, as quais podem ser visualizadas nos *logs* disponibilizados pelo programa. Estes cenários são descritos através de arquivos de configuração, os quais são analisados pelas classes de injeção de falhas responsáveis pela configuração da ferramenta.

Para cada falha do cenário de falhas pode ser especificada a maneira com que esta se manifestará. A ferramenta suporta três modos para a manifestação de falhas: permanentes (onde falhas ocorrem em todas as mensagens interceptadas), transientes (onde falhas ocorrem apenas uma vez) e intermitentes (falhas se manifestam de acordo com uma taxa de falhas). Neste último modo, FIONA foi implementada permitindo apenas uma distribuição uniforme de falhas no tempo.

Na Seção 4.2. é detalhado o fluxo de execução no processo de inicialização de cenários de falhas e as modificações efetuadas sobre os arquivos de configuração para a adição de distribuições de probabilidade na definição destes cenários.

4. Expansão da Ferramenta FIONA

Na Seção 2. foi mostrada a importância do mecanismo de injeção de falhas. A condução de experimentos baseada nesta abordagem visa reproduzir falhas reais em ambientes controlados. Para que os resultados destes experimentos sejam significativos, é proposto o uso de distribuições de probabilidade. Distribuições modelam comportamentos em determinadas situações. Por exemplo, Yoma [Yoma et al. 2005] propõe o uso de um modelo de Gilbert com restrições de duração de tempo baseado em distribuições gamma e geométrica para descrever a perda de pacotes UDP em redes utilizadas para a transmissão de dados de aplicações de tempo real. O uso de distribuições também permite que o comportamento das falhas injetadas seja variado de maneira controlada, pois, cada distribuição de probabilidade possui características bem definidas. Ainda é possível especificar o comportamento de uma distribuição através de informações como média e desvio padrão, fato que aumenta o controle do processo de condução de experimentos.

Nesta Seção são descritos os procedimentos adotados para adicionar a possibilidade de FIONA trabalhar com o uso de distribuições de probabilidade.

4.1. DESMO-J - *Discrete-Event Simulation and MOdelling in Java*

DESMO-J (*Discrete-Event Simulation and MOdelling in Java*) [Page 1999], é um *framework* de simulação de sistemas desenvolvido na Universidade de Hamburg. DESMO-J é totalmente desenvolvido na plataforma Java e é distribuído sobre a Licença GPL. A biblioteca suporta simulação discreta de sistemas baseada em eventos, e também possui funcionalidades de probabilidade e estatística. No escopo deste trabalho, as funcionalidades utilizadas do *framework* estão localizadas no pacote `desmoj.core.dist`. Este pacote contém todo o suporte da biblioteca às funções de probabilidade e, em especial, aos geradores de números randômicos.

Dentre os geradores de números randômicos disponíveis neste *framework* estão os geradores baseados em distribuições Bernoulli, Poisson, Normal, Exponencial etc. A utilização dos geradores citados, no ambiente FIONA, é mostrado na Seção 4.2.

4.2. Implementação

A primeira decisão a ser tomada é como adicionar as novas funcionalidades à ferramenta. Para garantir que não haja dependência de FIONA em relação ao *framework* DESMO-J, foi desenvolvida uma camada de abstração que possibilita o uso de um gerador de números randômicos genérico. Assim, as classes da biblioteca DESMO-J foram inseridas sobre esta camada de abstração, possibilitando aos demais módulos de FIONA usar os

recursos oferecidos sem estabelecer dependências. A Figura 2 é um diagrama de classes que ilustra de maneira simplificada a organização dos módulos de injeção de falhas da ferramenta.

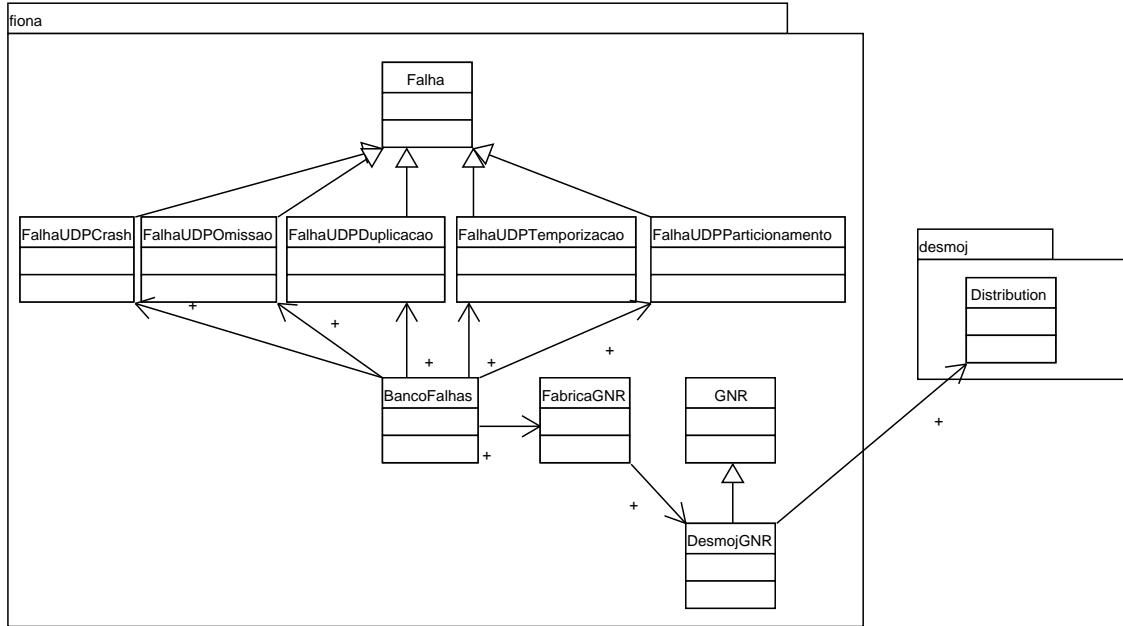


Figura 2. Diagrama simplificado das classes de injeção de falhas de FIONA

Como mencionado na Seção 3.3., FIONA permite a definição de cenários de falhas para a condução de experimentos via arquivo de configuração. Assim, as distribuições de probabilidade podem ser informadas neste arquivo. A Figura 3 mostra o formato de um arquivo de configuração para falhas de omissão de mensagens. O campo `padrão_de_repetição` indica como a falha irá se repetir durante o experimento, podendo ser permanente, transitória ou intermitente. Os campos `distribuição`, `semente` e `taxa_de_falhas` definem a frequência com que falhas serão manifestadas quando o modo de repetição for intermitente. O próximo parâmetro é referente ao tipo de ativação da falha (`tipo_ativação(tempo|mensagem)`). Uma falha pode se tornar ativa por um tempo definido em segundos ou pelo número da mensagem que está sendo trocada por um determinado *socket*. Os campos de `início` e `fim` de falha definem o intervalo da execução da aplicação que a falha estará ativa. A parte final da configuração indica em qual *socket* será injetada a falha.

```
UdpOmissionFault:<padrão_de_repetição>:[<distribuição>:<semente>:<taxa_de_falhas>:<tipo_ativação(tempo|mensagem)>:<início>:<fim>:<nodo_origem>:<porta_origem>:<nodo_destino>:<porta_destino>:<envio|recepção>:
```

Figura 3. Configuração para falhas de omissão

Na inicialização de FIONA uma das primeiras tarefas executadas é a leitura das informações mencionadas. A classe BancoFalhas é a responsável por esta tarefa. Durante o processamento do arquivo o módulo solicita um gerador de números randômicos à FabricaGNR, caso indicado no arquivo de configuração. A classe FabricaGNR

então inicializa o gerador especificado e retorna o objeto para o BancoFalhas, o qual configura a falha em questão com este gerador.

O processo de criação de um gerador é transparente para a classe BancoFalhas. A tarefa de criação é feita pela classe FabricaGNR que acessa as bibliotecas de geração de números aleatórios como DESMO-J. Esta abstração permite que bibliotecas adicionais sejam incorporadas ao programa sem que alterações sejam propagadas para os demais módulos. Cada gerador deve implementar a interface GNR, meio pelo qual geradores são manipulados pelos demais componentes do sistema. Portanto, quando mensagens são enviadas ou recebidas, as classes instrumentadas de FIONA acessam a classe BancoFalhas com o objetivo de verificar quais falhas devem ser acionadas para aquela mensagem. Caso a falha for do tipo intermitente, esta irá invocar a sua referência à GNR e assim obter valores randômicos da distribuição escolhida para determinar a ativação ou não desta falha.

5. Conclusão

Esforços visando a melhoria das técnicas de teste e validação de sistemas mostram-se válidos na medida em que a necessidade por sistemas confiáveis é exigida. Com o uso de distribuições de probabilidade para definir o comportamento do processo de injeção de falhas, é possível elaborar cenários de falhas capazes de representar a ocorrência de falhas observadas em ambientes reais.

A implementação da arquitetura para o uso de distribuições de probabilidade foi finalizada, portanto, a próxima etapa deste trabalho é a validação do mecanismo proposto. Testes serão realizados a fim de verificar o comportamento de aplicações na presença de falhas injetadas sobre diferentes distribuições de probabilidade. Também é importante analisar em quais situações uma certa distribuição de probabilidade mostra-se mais adequada para modelar a manifestação de falhas naquele ambiente.

Referências

- Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. E. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33.
- Ban, B. (1998). JavaGroups - group communication patterns in Java. Technical report, Department of Computer Science, Cornell University.
- Birman, K. P. (1996). *Building Secure and Reliable Network Applications*. Manning Pub., Co, Greenwich.
- Hsueh, M.-C., Tsai, T., and Iyer, R. (1997). Fault injection techniques and tools. *IEEE Computer*, 30(4):75–82.
- Jacques-Silva, G., Drebes, R. J., Gerchman, J., and Weber, T. S. (2004). FIONA: A fault injector for dependability evaluation of java-based network applications. *3rd IEEE Intl. Symposium on Network Computing and Applications*, pages 303–308.
- Page, B. (1999). DESMO-J a framework for discrete-event modelling and simulation. <http://www.desmoj.de>.
- Yoma, N. B., Busso, C., and Soto, I. (2005). Packet-loss modelling in ip networks with state-duration constraints. *IEE Proceedings Communications*, 152(1):1–5.

Geração de *logs* de Experimentos de Injeção de Falhas para Análise de Dependabilidade de Aplicações Distribuídas*

**Joana M. F. Trindade, Gabriela Jacques-Silva,
Taisy Silva Weber, Ingrid Jansch-Pôrto**

Instituto de Informática – Universidade Federal do Rio Grande do Sul
Caixa Postal 15064 – 90501-970 Porto Alegre, RS

{jmtrindade, gjsilva, taisy, ingrid}@inf.ufrgs.br

Resumo. *Sistemas distribuídos, por suportarem aplicações de alta disponibilidade, necessitam que sua dependabilidade seja validada. A injeção de falhas, nestes casos, torna-se indispensável, pois possibilita a inserção do tipo de falha desejada e no momento adequado aos testes de validação. Para facilitar uma análise do comportamento do sistema em resposta às falhas, é necessário monitoramento. O objetivo deste artigo é apresentar o sistema de monitoramento de FIONA quando em arquitetura local e distribuída, a geração de logs e sua relação com análise de dependabilidade de sistemas distribuídos.*

Abstract. *The support of high availability applications by distributed systems makes the validation of system dependability necessary. The use of fault injection mechanisms, in such cases, is essential, because it makes it possible to inject desired kinds of faults, at appropriate moments, for the validation tests. To provide a centralized analysis of the system's behaviour in response to the injected faults, monitoring is required. This paper presents FIONA's system to monitor its local and distributed architectures, the log generation and how it relates to distributed systems dependability analysis.*

1. Introdução

Em sistemas que suportam aplicações de alta disponibilidade, como *clusters*, sistemas distribuídos, servidores *WEB*, umas das características mais desejáveis é a confiabilidade desses sistemas. Objetivando o aumento na dependabilidade [Avizienis et al., 2004] de tais sistemas, são utilizadas mecanismos de tolerância a falhas. Uma técnica muito utilizada, com o fim de assegurar a eficácia de sistemas no fornecimento de seus serviços, é a validação através de testes de injeção de falhas.

A injeção de falhas consiste na inserção de falhas em um sistema ou aplicação, em um ambiente controlado, e na determinação do comportamento deste sistema em resposta às falhas nele injetadas [Hsueh et al., 1997]. Em sistemas de grande complexidade, em que soluções tradicionais de teste de software e confiabilidade não bastam para revelar importantes problemas de implementação, injeção de falhas é geralmente a única alternativa. E, por isso, na avaliação de dependabilidade do sistema, a injeção de falhas se torna essencial. Em ambientes de testes de validação, o tempo de espera por uma falha real é indeterminado, o que aumentaria o tempo de condução dos testes e prejudicaria uma

*Financiado pelos projetos ACERTE/CNPq (472084/2003-8) e DepGriFE/HP P&D Brasil

monitoração satisfatória de seu comportamento. Através da injeção de falhas, no entanto, é possível introduzir falhas no momento e do tipo necessários ao estudo do comportamento do sistema a ser analisado.

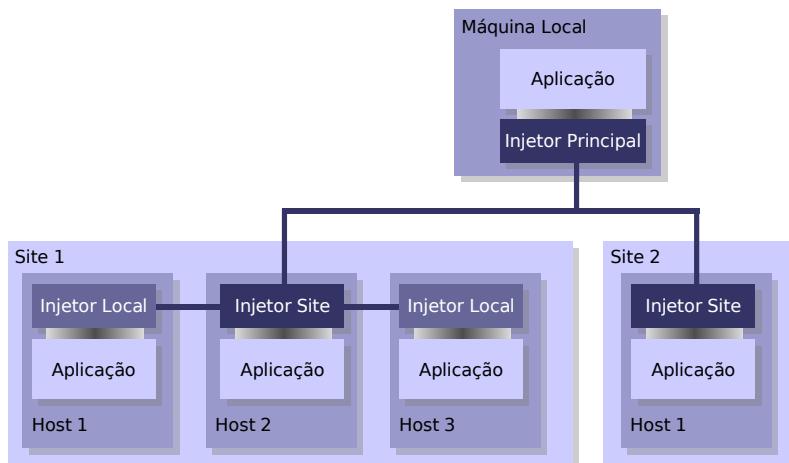
Entretanto, o comportamento do sistema alvo nem sempre pode ser facilmente observado por quem conduz os experimentos de injeção de falhas. É necessária, portanto, a implementação de mecanismos de monitoramento para observar o comportamento do sistema em resposta às falhas injetadas. Além disso, é preciso observar tanto as informações relativas ao experimento geradas pelo injetor de falhas, quanto os *logs* gerados pelo mecanismo de tolerância a falhas da aplicação alvo. Assim, com estas informações e a utilização de uma ferramenta de análise de dependabilidade apropriada ao experimento, é possível obter medidas confiáveis de dependabilidade.

Este artigo visa apresentar o sistema de monitoramento do ambiente distribuído para injeção de falhas FIONA (*Fault Injector Oriented to Network Applications*) [Jacques-Silva et al., 2004]. Serão explicados o funcionamento do sistema de coleta de dados acerca do experimento, quando injetadas falhas em arquiteturas locais e distribuídas. A geração de *logs* contendo os dados coletados na etapa anterior e, em seguida, a centralização dos vários *logs* gerados por FIONA no sistema distribuído, em um único arquivo ordenado.

A estrutura é como segue: A Seção 2 apresenta o sistema de monitoramento de FIONA. A Seção 3 descreve a coleta e a gravação dos dados de monitoramento em arquiteturas locais e distribuídas. E, por fim, a Seção 4 apresenta as considerações finais do artigo.

2. Injetor de Falhas FIONA

FIONA é um injetor de falhas de comunicação para aplicações em Java, que tem como objetivo a validação de sistemas distribuídos de pequena e larga escala. Apresentando uma arquitetura distribuída, FIONA visa facilitar a condução de experimentos de injeção de falhas, através da configuração centralizada de cenários de falhas que afetam o sistema sob teste. Na busca por escalabilidade, sua arquitetura distribuída foi baseada em uma proposta para arquitetura de monitoramento de larga escala. Tal estratégia foi utilizada no estabelecimento de uma hierarquia entre os injetores, classificando-os em três tipos de comportamento: *injetor principal*, *injetor de site* e *injetor local*.



Um *injetor local* executa em uma máquina e se comunica com o *injetor de site*. Cada injetor é responsável pela geração do seu próprio log. Os *injetores de site* se encarregam do recolhimento de tais *logs*, repassando-os ao *injetor principal*. O *injetor principal* é o responsável pelo gerenciamento do experimento. É ele que distribui a configuração do experimento e faz a centralização dos *logs* de todos os nodos participante do sistema, além de ser o responsável pela ativação de falhas que afetam mais de um nodo simultaneamente, como particionamento de rede.

FIONA baseia-se no modelo de falhas para sistemas distribuídos, proposto por Birman [Birman, 1996]. Por objetivar a validação de sistemas distribuídos de larga escala, **FIONA** possibilita a injeção das falhas mais freqüentes em ambientes deste tipo. São elas: falhas de *omissão*, *colapso*, *temporização* e *particionamento*. Uma comparação entre **FIONA** e outros injetores de falhas encontra-se em [Jacques-Silva, 2005].

FIONA possui implementado um sistema de monitoramento, o qual visa permitir, de forma centralizada, a análise do *log* do experimento. Porém, é importante ressaltar que para obtenção de medidas de dependabilidade após o experimento de injeção de falhas, é necessário observar tanto o *log* da ferramenta de injeção de falhas quanto o gerado pela aplicação alvo. Os *logs* da aplicação são dependentes da mesma, e não são tratados pelo sistema de monitoramento de **FIONA**. Estes devem ser passados como entrada para uma ferramenta apropriada de análise de dependabilidade, em conjunto com os *logs* do experimento de injeção de falhas.

O sistema de monitoramento de **FIONA** é dividido em duas partes: o monitoramento realizado localmente e o monitoramento de todos os nodos componentes do sistema (monitoramento distribuído).

2.1. Monitoramento Local

O monitoramento local é a geração de *logs* do experimento de injeção de falhas no próprio nó. Durante a execução, a cada falha injetada, é gerada uma entrada no arquivo de *log*. Cada entrada é composta por um conjunto de informações, tais como: qual o momento em que a falha foi injetada, o tipo de falha injetada, o número da mensagem no experimento, o endereço IP dos nodos de origem e destino e as portas utilizada na troca de mensagens entre os nodos.

2.2. Monitoramento Distribuído

Todos os nodos do sistema gravam um conjunto de informações sobre cada falha injetada no seu *log*, seja ele um injetor local, de *site* ou principal. Ao início de um experimento de injeção distribuída, cada nodo assume a responsabilidade por seu próprio *log*, e ao término da execução, este é enviado ao nodo de posição diretamente acima na hierarquia (Figura 1). Se o nodo em questão for um injetor de *site* ou principal, ele aguarda até todos os outros nodos a ele conectados um nível abaixo na hierarquia, enviarem-lhe seus *logs*. Um *injetor de site*, ao receber um *log*, repassa-o ao *injetor principal*, que também espera a recepção das informações de todos os nodos a ele conectados (*injetores de site*) para finalizar sua execução.

Uma vez que a análise dos resultados deve ser feita de forma centralizada, o salvamento final das informações é realizado apenas pelo *injetor principal*, podendo este ser tanto um injetor de falhas da aplicação, como somente um coordenador do experimento, não penalizando em desempenho neste segundo caso.

Cada *log* recebido de um *injetor de site* pelo *injetor principal* é gravado em disco. Com o *log*, são armazenados também o endereço da máquina a que pertenciam

as informações e o número do processo correspondente ao experimento. Nos nodos injetores locais e de *site*, as informações referentes às falhas injetadas são manipuladas por meio de *buffers*, salvos em memória, e enviados ao nodo a que estão conectados.

3. Ordenação de Logs

Para uma obtenção de medidas de dependabilidade confiáveis, a ferramenta injetora de falhas deve ser eficiente tanto na injeção de falhas propriamente dita, quanto na coleta dos dados referentes ao experimento e ao comportamento do sistema em reação às falhas injetadas. Em **FIONA**, seu sistema de monitoramento é responsável pela coleta e salvamento centralizado dos dados gerados durante a execução do experimento de injeção de falhas. Como há a possibilidade de mais de um nodo estar injetando falhas em uma aplicação, a busca pelas informações de todas as falhas injetadas por um nodo, por exemplo necessita de uma visão global de toda a execução do experimento. Para isto, os registros de cada falha injetada devem estar devidamente ordenados.

Entretanto, por se tratarem de várias máquinas com tempos locais diferentes entre si, uma ordenação direta dos tempos salvos em *log*, acabaria fornecendo uma ordem de acontecimentos inconsistente. Assim, é preciso que os dados referentes ao experimento sejam ordenados levando-se em consideração as diferenças de relógio entre cada uma das máquinas.

Neste sentido, foi desenvolvido o programa **LOrd** (*Log Orderer*). Implementado em Java, **LOrd** é um ordenador de *logs* gerados pelo sistema de monitoramento do ambiente distribuído de injeção de falhas **FIONA**. Esta seção é dividida em duas subseções: A Subseção 3.1 explica o método de ordenação utilizado e a Subseção 3.2 descreve sua implementação.

3.1. Método de Ordenação

Cada arquivo de *log* é gravado em disco com o horário local de seu nodo. Para fins de ordenação, então, é preciso que sejam padronizados os diversos horários em uma única referência temporal base. Isso é feito utilizando-se o algoritmo de sincronização de relógios, proposto por [Maillet e Tron, 1995]. Este método propõe a criação de um único relógio lógico através do cálculo das diferenças de hora local entre cada uma das máquinas do sistema e uma máquina de referência *r*.

Esta diferença é calculada, para cada máquina, uma vez no início e outra ao final do experimento. *r* envia uma mensagem a cada máquina. A máquina remota, então, obtém sua hora local e envia este valor de volta a *r*. O horário local de *r* é lido antes do envio e após o recebimento desta mensagem. Calcula-se o atraso na comunicação, que corresponde a uma média aritmética entre os dois valores acima obtidos. Este atraso é considerado equivalente ao valor lido na segunda máquina.

Segundo o algoritmo, a relação entre o horário local registrado pela máquina *i*, no tempo absoluto *t* e a hora local registrada pela máquina de referência *r* no mesmo tempo absoluto, *t* é regida pela equação:

$$lt_i(t) = \alpha_i + \beta_i lt_r(t) + \delta_i$$

Onde o tempo absoluto é representado por *t*, a constante α_i equivale ao valor do relógio local da máquina *i*, quando *r* possui *t* = 0. A diferença do horário local da máquina *i* em relação ao horário de *r* é igual a β_i . A granularidade e todas as outras

perturbações independentes de t , pequenas o suficiente para serem descartadas, são representadas por δ_i .

LOrd corrige o horário local de i lhe atribui um horário global $LC_i(t)$, sendo estimado por:

$$LC_i(t) = lt_r(t) \approx \frac{lt_i(t) - \alpha_i}{\beta_i}$$

3.2. Implementação

LOrd é executado duas vezes, uma antes do ínicio do experimento e outra logo após o seu término. Através de uma operação de *ping-pong* é feita a troca de mensagens entre a máquina referência r e cada uma das máquinas i . Por meio desta troca, são coletadas as amostras a serem utilizadas por **LOrd** no cálculo dos coeficientes de correção de horário para cada máquina. Em seguida, utilizando-se de uma classe de leitura, **LOrd** lê os *logs* gerados por **FIONA**.

Cada *log* é formado por uma ou mais linhas, em número igual ao de falhas injetadas. A primeira linha contém o nome do nodo a que pertence o arquivo. As demais linhas possuem informações sobre a falha correspondente. Cada registro de entrada no arquivo possui um número de campos dependente do tipo de falha injetada, podendo ser: *hora local*, *tipo de falha*, *número da mensagem*, *ip e porta de origem*, *ip e porta de destino*, *injeção realizada no envio ou na recepção*, *tempo de atraso da mensagem*.

Por ser o que contém informação referente ao horário de injeção da falha, *hora local* é o único campo além do nome da máquina (obtido na primeira linha do arquivo) a ser utilizado na geração dos coeficientes de correção. Com base nestes coeficientes, são ajustados os horários contidos nos *logs*, seguindo-se o algoritmo de sincronização apresentado. Como saída do programa, é produzido um único arquivo de *log*, em que todas as falhas injetadas estão ordenadas por tempo num horário global.

LOrd pode, adicionalmente, ser adaptado para quaisquer formatos de *log*, além daquele utilizado por **FIONA**. Para isso, basta a simples implementação de uma classe de leitura correspondente, assim permitindo a sincronização de outras aplicações que necessitem de sincronização *offline*.

4. Conclusão

Em injeção de falhas, mecanismos de monitoramento se fazem necessários para uma eficiente coleta de informações sobre as falhas injetadas na aplicação alvo. E a geração de *logs*, por estes mecanismos de monitoramento, é essencial na obtenção de métricas, na análise de dependabilidade e na determinação do comportamento das aplicações em resposta às falhas injetadas.

Este artigo apresentou **FIONA**, um ambiente para condução de experimentos de injeção de falhas, e seu sistema de monitoramento. **FIONA** possibilita a injeção de falhas localmente (em apenas um nodo) ou em arquiteturas distribuídas e, nesta segunda opção, gerando *logs* independentes. Seu próprio sistema de monitoramento recolhe estes *logs* ao final do experimento.

Para sincronização dos *logs*, foi desenvolvida a ferramenta (**LOrd**), a qual utiliza um método de sincronização *offline*. A escolha deste método não penaliza em desempenho a execução da aplicação, pois, por ser um algoritmo *offline*, é executado após o

término do experimento, permitindo a cada nodo a geração de seu *log* de maneira independente. A utilização de LOrd garante consistência de ordem no histórico dos eventos ocorridos, tornando mais simples o cruzamento do *log* do experimento com o da aplicação alvo, facilitando a análise de dependabilidade da aplicação.

Referências

- Avizienis, A., Laprie, J.-C., Randell, B., e Landwehr, C. E. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33.
- Birman, K. P. (1996). *Building Secure and Reliable Network Applications*. Manning Publications, Co, Greenwich.
- Hsueh, M.-C., Tsai, T. K., e Iyer, R. K. (1997). Fault injection techniques and tools. *IEEE Computer*, 30(4):75–82.
- Jacques-Silva, G. (2005). Injeção distribuída de falhas para validação de dependabilidade de sistemas distribuídos de larga escala. Mestrado em ciência da computação, Instituto de Informática, UFRGS, Porto Alegre.
- Jacques-Silva, G., Drebes, R. J., Gerchman, J., e Weber, T. S. (2004). FIONA: A fault injector for dependability evaluation of Java-based network applications. In *Proc. of the 3rd IEEE Intl. Symposium on Network Computing and Applications*, páginas 303–308, Cambridge, MA. IEEE Computer Society Press.
- Maillet, E. e Tron, C. (1995). On efficiently implementing global time for performance evaluation on multiprocessor systems. *Journal of Parallel and Distributed Computing*, 28(1):84–93.

Tolerância a falhas no modelo MultiCluster

Marcos Locateli Gontarski, Marcos Ennes Barreto

Centro Universitário La Salle (UNILASALLE)
Av. Victor Barreto 2288, CEP: 92.010-000 – Canoas – RS – Brazil
locatelei@bancomatone.com.br; barreto@unilasalle.edu.br

Resumo. Técnicas de tolerância a falhas constituem uma das principais abordagens para o provimento de alta disponibilidade para aplicações críticas que executam em agregados. Este artigo descreve o serviço de tolerância a falhas projetado para a biblioteca DECK, usada como ambiente de execução do modelo MultiCluster.

1. Introdução

Arquiteturas de agregados têm sido usadas já há algum tempo como uma alternativa de alto desempenho e relativo baixo custo para a execução de aplicações paralelas que exigem grande poder de processamento [Buyya 1999], e vêm sendo empregadas atualmente nos mais diversos cenários, desde centros de pesquisa aplicada até grandes empresas de processamento e armazenamento de dados.

No cenário comercial, arquiteturas de agregados têm sido usadas para diferentes tipos de servidores (aplicação, armazenamento, banco de dados, entre outros). O uso crescente deste tipo de arquitetura se deve principalmente ao relativo baixo custo que ela apresenta e ao emprego de redes de comunicação de alto desempenho; pontos que tornam estas arquiteturas bastante propícias ao desenvolvimento e à execução de aplicações críticas, tais como sistemas de comércio eletrônico e sistemas bancários. Nesse contexto, o uso de técnicas de tolerância a falhas é um dos principais mecanismos para a obtenção de alta disponibilidade e confiabilidade na execução dessas aplicações [Hwang 1998].

No lado acadêmico, as arquiteturas de agregados têm sido usadas para o desenvolvimento de técnicas de replicação de dados, escalonamento dinâmico e migração de processos, simulação de redes de sensores, entre outros [Youn et al 2000].

Este trabalho descreve o serviço de tolerância a falhas para a biblioteca de programação DECK, usada como ambiente de execução do modelo MultiCluster [Barreto et al 2000]; e apresenta também uma validação preliminar de algumas funcionalidades deste serviço.

2. Agregados de alta disponibilidade

Histórica e idealmente, agregados têm sido mais usados para alcançar alto desempenho do que alta disponibilidade. Em um agregado de alto desempenho, os processos executam paralelamente em vários nós (geralmente) homogêneos, trabalhando de forma cooperativa e compartilhando recursos de processamento e de entrada e saída. A homogeneidade e a baixa latência de comunicação, aliadas ao baixo custo

incremental, tornam os agregados uma excelente plataforma para a execução de aplicações que demandam por alto poder de processamento.

Nos agregados de alta disponibilidade, os nós compartilham os mesmos dispositivos de armazenamento, substituindo-se uns aos outros em caso de falha de hardware ou software. Em comparação ao caso anterior, num agregado de alta disponibilidade, nem todos os nós estão diretamente disponíveis ao usuário – determinados nós são mantidos como reservas, com funções de armazenamento e replicação de dados e programas; e estão aptos a assumir o processamento em caso de falha em um nó principal.

Agregados de alta disponibilidade normalmente não compartilham a carga de processamento como os agregados de alto desempenho, nem distribuem tráfego como os agregados que fazem平衡amento de carga. Nos agregados de alta disponibilidade, além da replicação dos dados, o serviço de tolerância a falhas pode migrar tarefas de um nó defeituoso para um nó operacional e reiniciar aplicações [Buyya 1999].

O software usado em agregados de alta disponibilidade geralmente agrupa os recursos que devem ser monitorados e, em caso de falha, reinicializados em outro local. Este “grupo de recursos” pode então ser gerenciado através de alguma técnica de tolerância a falhas [Youn et al 2000]:

- *Idle standby*: um nó reserva é capaz de assumir a gerência do grupo de recursos em caso de falha do nó principal. Pode haver interrupção de serviços caso o nó principal se recupere da falha e reassuma o grupo de recursos;
- *Rotating standby*: técnica semelhante a anterior, porém o nó principal, ao retornar do estado falho, se mantém como nó reserva do “novo” nó gerenciador. Neste caso, a interrupção de serviços pode ser minimizada ou eliminada;
- *Simple failover*: o nó principal executa uma aplicação crítica, enquanto o nó reserva executa uma aplicação não-crítica e mais os serviços de monitoramento do nó principal. No caso de falha do nó principal, o reserva pára a aplicação não-crítica e assume a aplicação crítica;
- *Mutual takeover*: esquema onde dois ou mais nós são igualmente capazes de assumir o grupo de recursos, um do outro. Pode ser usado em aplicações fracamente acopladas, onde cada grupo de nós executa um conjunto de tarefas distintas e, em caso de falhas, outro nó ou grupo de nós é capaz de assumir a execução das tarefas; e
- *Concurrent access*: esquema onde os nós acessam os mesmos recursos (um dispositivo de armazenamento externo, por exemplo) de forma concorrente, de modo que os dados produzidos em um nó possam ser recuperados por outro nó em caso de falhas.

3. Tolerância a falhas no modelo MultiCluster

O MultiCluster corresponde a um modelo para a concepção de grades locais a partir da integração de agregados de alto desempenho. O modelo utiliza a biblioteca DECK sobre a implementação em linguagem C dos protocolos de gerência e comunicação definidos no projeto JXTA [JXTA 2005].

DECK é uma biblioteca de programação desenvolvida no Instituto de Informática da UFRGS desde 1998; e que contém atualmente diferentes versões para diversas tecnologias (Ethernet, Myrinet, SCI e Infiniband) e protocolos (TCP, BIP, GM, SISCI, VIA) de comunicação. Todas as implementações DECK são organizadas em duas camadas: a camada inferior fornece recursos de multiprogramação, comunicação e conexão entre agregados; e a camada superior é responsável pelo fornecimento de serviços especializados (comunicação coletiva, tolerância a falhas, roteamento de mensagens, etc.).

No MultiCluster, diferentes cenários de utilização de grades são considerados (Figura 1), onde em cada cenário o usuário define parâmetros para a integração e comunicação entre os recursos integrados, através de arquivos de configuração.

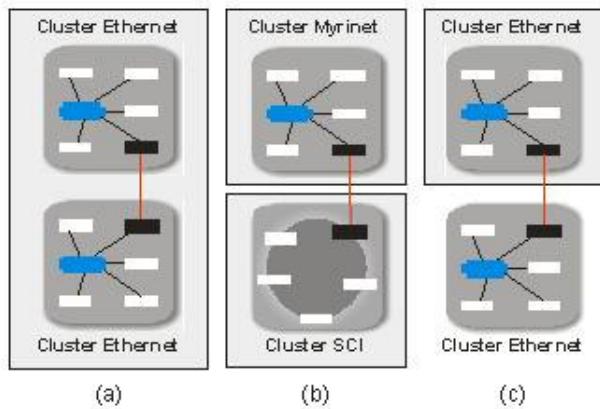


Figura 1. Cenários de integração do modelo MultiCluster.

Em cada agregado, um nó é configurado como máquina de acesso (marcada em preto na figura). Neste nó são executados alguns serviços especializados que permitem o controle dos demais nós do agregado, bem como o roteamento de mensagens entre nós pertencentes a diferentes agregados.

No cenário *a* o objetivo da integração é prover escalabilidade para aplicações fortemente acopladas, onde a aplicação considera a existência de nós com comunicação direta entre si, independente de suas localizações físicas; sendo distribuída de maneira uniforme entre os nós (segundo o modelo SPMD).

No cenário *b* o objetivo da integração é fazer uso seletivo dos agregados, de modo que a aplicação seja distribuída de acordo com as necessidades de cada tarefa e das características providas em cada recurso. Neste caso, os agregados são vistos como recursos distintos e as operações de comunicação são realizadas somente entre tarefas que pertencem a um mesmo agregado.

O cenário expresso na letra *c* corresponde a uma integração voltada para tolerância a falhas, que pode ser usada em combinação com qualquer um dos cenários anteriores. Neste cenário, a aplicação executa em um agregado, mas o modelo prevê outro conjunto de nós que é usado em caso de falhas. A máquina de acesso do agregado principal mantém uma conexão com a máquina de acesso reserva. No caso de falha de um dos nós, a tarefa atribuída ao mesmo pode ser repassada ao agregado reserva para que seja re-executada (Figura 2a e 2b). No caso de falha da máquina de acesso do agregado principal, os nós deste agregado podem “eleger” a máquina de acesso do

agregado reserva para o papel de controlador, passando a esta máquina o resultado das suas execuções (Figura 2c e 2d).

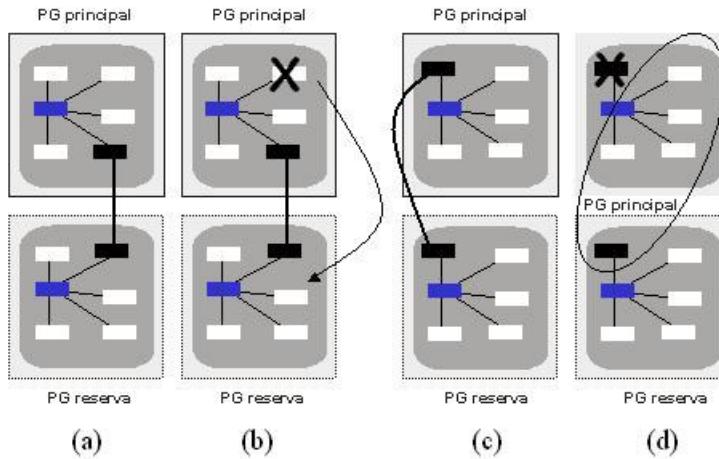


Figura 2. Migração de tarefa (2 a e 2b) e de *peergroup* (2c e 2d) no MultiCluster.

No serviço proposto, um agregado é definido como *peergroup* (PG) principal; e nele, a máquina de acesso executa os serviços de tolerância a falhas, gerência do agregado e roteamento de mensagens. Estes dois últimos serviços configuram esta máquina como um *rendezvous peer* e um *relay peer*, que na semântica do serviço, representam um nó que armazena informações sobre os demais pares e um nó usado para comunicação com pares pertencentes a outros *peergroups*, respectivamente. Adicionalmente, outro agregado é especificado como *peergroup* reserva, contendo um determinado número de nós e uma máquina de acesso; que executa os mesmos serviços, porém em caráter de monitoramento.

O serviço de tolerância a falhas proposto possui as seguintes características:

- implementa uma técnica mista (*idle standby + simple failover*): se o controlador do PG principal falha, o controlador reserva assume o controle do grupo até o final da execução da aplicação, mesmo que adiante o controlador principal retorne do estado falho. Além disso, o controlador do PG reserva executa funções de monitoramento do PG principal;
- a detecção de falhas dá-se mediante falhas de comunicação entre quaisquer pares da rede, e depende do tipo de aplicação que está sendo executada no MultiCluster. Ao escolher o cenário de tolerância a falhas, toda comunicação entre os pares obedece a um tempo máximo (*timeout*) de resposta. Se a aplicação for fortemente acoplada, a falha em um nó faz com que o serviço migre esta tarefa para um nó reserva, restaurando o contexto da tarefa a partir de informações gravadas em um arquivo de log a cada troca de mensagens. Se a aplicação for fracamente acoplada, então o serviço simplesmente aloca um nó reserva para que a tarefa executada novamente.
- A falha do controlador do PG principal é detectada quando algum par deixa de receber mensagens de monitoramento do controlador. Neste caso, este par estabelece uma conexão com o controlador reserva, passando a reportar a este coordenador as suas informações de monitoramento.

Tanto a migração de tarefas quanto a migração de *peergroup* somente é possível devido a possibilidade de configuração de *rendezvous peers* e *relay peers* dentro do modelo MultiCluster. Além disso, outro protocolo JXTA é essencial para o desenvolvimento do serviço: o *Peer Info Service*. Através deste serviço, um nó pode receber informações (carga de trabalho, estado de execução, etc.) dos demais nós. O serviço de tolerância a falhas usa este protocolo para monitorar o estado de cada nó (par) pertencente ao *peergroup*.

4. Validação do serviço

Parte do serviço de tolerância a falhas proposto foi implementado e está em fase de validação. O protótipo inicial foi desenvolvido em uma rede Linux, distribuição Fedora (2.6.5-1), com a implementação DECK/TCP e JXTA-C versão 2.1.

Neste protótipo, dois *peergroups* (principal e reserva) foram estabelecidos, cada qual com 4 pares (controlador + 3 nós). As aplicações usadas para a validação são:

- Replicação de arquivos entre pares de um *peergroup* e simulação de falha em um determinado par, o que ocasiona a transferência do arquivo replicado para um par pertencente ao *peergroup* reserva;
- Multiplicação de matrizes, onde um determinado par falha após processar uma parte da aplicação, fazendo com que a tarefa e o seu contexto tenham que ser migrados para um nó do *peergroup* reserva; e
- Ordenação distribuída de vetores, onde cada nó recebe uma parte do vetor a ser ordenado. Quando um par falha, a parte do vetor deve ser passada para um nó do *peergroup* reserva.

A figura 3 apresenta os resultados obtidos com a replicação de arquivos, executando em uma rede com máquinas Pentium III de 1.8 GHz e 256 MB de memória.

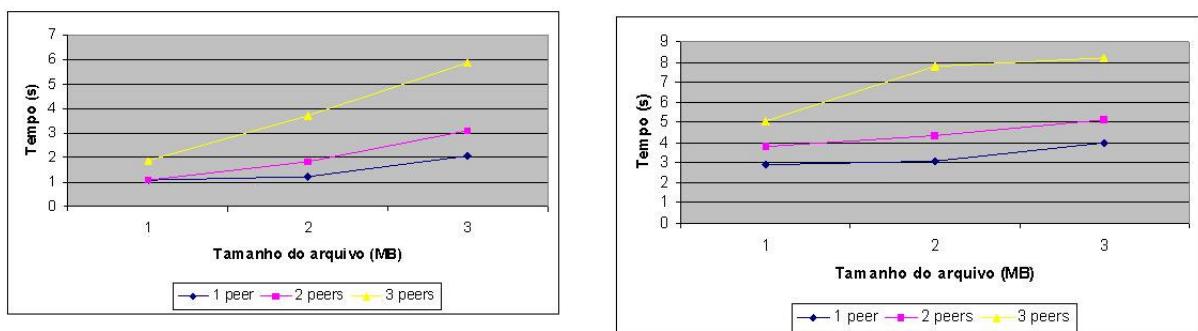


Figura 3. Tempo de replicação de arquivos no protótipo.

Nesta simulação, a quantidade de pares no *peergroup* principal e o tamanho do arquivo replicado foram variados. Na figura à esquerda são apresentados os tempos de replicação para um *peergroup* com 1, 2 e 3 nós processadores, além do nó controlador. Na figura à direita são apresentados os tempos de replicação para os mesmos casos, considerando a falha de um dos nós e consequente migração do arquivo replicado para um nó do *peergroup* reserva.

5. Trabalhos relacionados

Existem na literatura diversas soluções de alta disponibilidade e confiabilidade para agregados e servidores Web. Em [Youn et al 2000], um conjunto de propostas baseadas em sistemas Unix e Linux são comparadas em termos das técnicas de hardware e software utilizadas para o fornecimento de tolerância a falhas.

Em [Anwar et al 2004], um conjunto de protocolos de comunicação em grupo usados em redes P2P é analisado com vistas ao emprego deste tipo de protocolo em aplicações militares. A análise destes sistemas revela que redes P2P ainda provêem poucas funcionalidades relacionadas a segurança e tolerância a falhas.

Dentro do projeto JXTA existem algumas iniciativas relacionadas com tolerância a falhas: o projeto do (*distributed objects*) usa os protocolos JXTA para a replicação de objetos; e o projeto JXTA-RM implementa um protocolo de *multicast* confiável.

6. Conclusões

O fornecimento de técnicas de tolerância a falhas em redes P2P ainda é uma área nova de pesquisa, com poucas propostas já divulgadas. Neste trabalho, um serviço de tolerância a falhas proposto para o modelo MultiCluster é descrito. Este serviço está baseado na utilização de diferentes *peergroups*, que são usados para a execução da aplicação e de tarefas de monitoramento e tratamento de falhas, em diferentes cenários de utilização do modelo MultiCluster.

Referências bibliográficas

- Hee Yong Youn, Chansu Yu, Dong Soo Han, and Dongman Lee. (2000) The Approaches for High Available and Fault-Tolerant Cluster Systems." In.: Workshop on Fault Tolerant Control and Computing (FTCC-1), pp.107-116, May 2000.
- JXTA (2005). Projeto JXTA. <http://www.jxta.org>.
- Kai Hwang. (1998) Scalable parallel computing. New York: McGraw-Hill, USA.
- Marcos Barreto, Rafael Ávila e Philippe Navaux (2000) The MultiCluster model to the integrated use of multiple workstation clusters. In.: International Workshop on Personal Computers based Networks of Workstations (PC-NOW 2000). Quintana Roo, México.
- Pankaj Jalote (2002) Fault tolerance in distributed systems. New Jersey: Prentice Hall, USA.
- Rajkumar Buyya (ed) (1999) High performance cluster computing – architectures and systems. Vol. 1. New Jersey: Prentice Hall, USA.
- Zahid Anwar, William Yurcik, Roy Campbell. (2004) A survey and comparison of peer-to-peer group communication systems suitable for network-centric warfare. Disponível em <http://www.ncsa.uiuc.edu> (Novembro de 2004).

Modelagem de Requisitos de QoS Utilizando UML-RT:

Estudo de caso da especificação de requisitos de QoS para sistemas interativos baseados em redes com o uso de UML-RT

Edison Pignaton de Freitas, Elias Teodoro da Silva Jr, Fabiano Costa Carvalho

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{epfreitas, etsilvajr, fccarvalho}@inf.ufrgs.br

Resumo: A linguagem UML padrão não apresenta todo o ferramental necessário à modelagem de sistemas de tempo-real, mas existe uma extensão para projetos desta natureza que cobre esta deficiência. Para explicitar os requisitos de QoS presentes em sistemas interativos distribuídos em rede, tais como sistemas de vídeo-conferência, pode-se adotar como alternativa a modelagem UML com enfoque de tempo-real. Este trabalho busca apresentar o uso da linguagem UML-RT para modelar aplicações de tempo-real interativas baseadas em redes. Com a apresentação do estudo de caso, procura-se mostrar a utilização de algumas características da UML-RT de forma a conseguir um modelo que expresse com clareza os requisitos de tempo-real pertinentes.

1 – Introdução

Muito se fala sobre o uso de sistemas de tempo-real (TR) para aplicações de automação industrial e sistemas embarcados [EDWARDS 1997]. De fato, estes são os exemplos mais clássicos do uso de sistemas TR. Porém, um outro domínio de aplicação onde o enfoque de tempo-real pode encontrar solo bastante fértil é o dos sistemas interativos distribuídos em rede. Tais sistemas têm uma forte demanda por qualidade de serviço, e apresentam rígidos prazos a serem cumpridos, tanto quanto qualquer outro sistema TR.

O projeto de sistemas de tempo-real apresenta complicadores inevitáveis quanto à definição de seus requisitos mínimos de funcionamento adequado. Isto pode ser atribuído principalmente aos rígidos prazos de execução de tarefas. Num sistema de vídeo-conferência, por exemplo, é indispensável que suas tarefas cumpram os requisitos temporais, caso contrário a interatividade pode ficar comprometida.

A linguagem UML não apresentava, em suas primeiras versões, nenhum suporte às peculiaridades existentes no projeto de sistemas de tempo real. O tratamento destes requisitos somente foi introduzido à linguagem através do perfil UML-RT [OMG 2004], que consolidou uma série de propostas de extensão à linguagem. Em sua última versão, a 2.0, a UML apresenta bem amadurecidas as ferramentas de suporte ao projeto de sistemas de tempo-real.

Este trabalho propõe a utilização dos recursos presentes na UML-RT para a modelagem de sistemas interativos baseados em rede como os de vídeo-conferência. Apesar do uso de redes de Petri estocásticas temporizadas [TSAI, YANG, CHANG 1995] ser bastante adequado à modelagem de sistemas desta natureza, os diagramas UML-RT possuem uma expressividade que merece ser explorada neste domínio.

Na seqüência, a seção 2 apresenta alguns conceitos sobre sistemas de tempo-real, enquanto a seção 3 trata da extensão RT para a UML. Na seção 4 é apresentado um estudo de caso e finalmente na seção 5, as conclusões.

2 – Sistemas de Tempo Real

Sistemas de tempo-real são aqueles que possuem fortes requisitos no que se refere ao atendimento de compromissos de tempo de execução (*deadlines*), robustez e segurança, principalmente. Estas exigências são consequência do risco potencial de catástrofes em caso de falha, ou até mesmo no caso do seu funcionamento não ocorrer exatamente como o esperado [DOUGLASS 1999].

Pode-se classificar os sistemas de tempo-real da seguinte forma:

- *Hard Real-Time*: São aqueles nos quais os requisitos temporais devem ser atendidos de forma incondicional. A perda de um prazo (*deadline*) constitui um erro de computação. Neste tipo de sistema, um resultado atrasado é inaceitável.

- *Soft Real-Time*: Neste tipo de sistema os prazos podem ser eventualmente perdidos, ou pode haver um atraso tolerável, ou mesmo as duas hipóteses. Eles são regidos, em sua maioria, por uma média de exigência temporal.

Normalmente os sistemas podem ser classificados em posições intermediárias entre estes dois extremos [SHAW 2001], levando-se em conta o quanto rigorosos ou tolerantes à falhas eles são.

Pode-se dizer que um sistema de vídeo-conferência ocupa uma posição intermediária não muito distante do extremo rigoroso (*hard real-time*). Mesmo essa aplicação não sendo tão crítica quanto um controle de tráfego aéreo, por exemplo, a perda freqüente de limites temporais tornaria o sistema completamente inútil, pois não se alcançaria a interatividade desejada.

3 – Extensões da UML para Tempo-Real

Antes do surgimento das extensões UML para sistemas de tempo-real, a linguagem orientada a objetos mais usada neste domínio era a ROOM (*Real-Time Object Oriented Modeling*) [DOUGLASS 1999]. Estudos levaram à proposição do uso da UML na modelagem de sistemas de tempo-real. Porém, esta linguagem ainda não estava apta a representar, com a riqueza de detalhes necessária, as diferentes características estruturais e comportamentais de tais sistemas. Com o intuito de suprir esta deficiência na linguagem UML, foi proposta uma extensão com base em conceitos advindos da linguagem ROOM. Esta proposta resultou num conjunto de conceitos de propósito geral para a modelagem de sistemas de tempo-real. Mais tarde, apareceram outras propostas de extensões à UML para cobertura das necessidades de modelagem no domínio de tempo-real. Tais propostas foram finalmente padronizadas pela OMG através do perfil UML-RT.

O perfil UML-RT apresenta *frameworks* para construção de modelos: 1) de Recursos; 2) de Tempo; 3) da Concorrência; 4) de Análise de Escalonabilidade; e 5) do Desempenho. O foco deste trabalho é apresentar a utilização deste último *framework*, que se propõe a facilitar as seguintes tarefas do projetista: 1) Captura dos requisitos de desempenho no contexto do projeto; 2) Associação das características de QoS relacionadas ao desempenho com elementos específicos do modelo UML; 3) Especificação de parâmetros de execução, que podem ser usados na modelagem de ferramentas para computar características de desempenho previsto; e 4) Apresentação de resultados de desempenho, computados por ferramentas de modelagem ou obtidos por meio de testes.

3.1 – Expressando QoS através da UML-RT

Com a utilização do *framework* para análise de desempenho disponível no perfil UML-RT, os requisitos de QoS podem ser claramente expressos através de uma série de

estereótipos. Estes estereótipos permitem uma análise efetiva da influência de tais requisitos no projeto da aplicação [AAGEDAL, ECKLUND 2002].

Nos modelos do perfil UML-RT, um recurso é visto como um servidor que provê um ou mais serviços para um cliente. A limitação física do recurso é representada através dos seus atributos de QoS. Em geral, os atributos de QoS caracterizam *quão bem* um recurso realiza um serviço. O modelo das limitações de um recurso é feito através dos estereótipos *PAresource* e *PAhost*.

O conceito de QoS é bastante amplo, podendo ser usado para modelar diferentes propriedades. Este trabalho procurou seguir o conceito apresentado em [VOGEL 1995], segundo o qual as características modeladas são preferencialmente quantitativas (tamanho, tempo de serviço, capacidade, tempo de resposta, latência, largura de banda, *jitter*), mas também podem ser especificações qualitativas (compartilhamento, política de escalonamento ou modelo de falha), que no final vão provocar algum impacto quantitativo. Estas informações aparecem de forma clara nos modelos UML-RT com o uso dos estereótipos *PAresource* e *PAhost*. Já informações que tratam de aspectos quantitativos de execução de ações são representadas pelo estereótipo *PAstep*.

Naturalmente, não é suficiente conhecer a QoS que um recurso pode oferecer. Também é necessário saber o que é solicitado pelo cliente. Isto sugere a diferenciação entre QoS oferecida (do lado do recurso) e QoS requerida (do lado do cliente). Na análise quantitativa o que se busca determinar é a compatibilidade de QoS destes dois extremos. Com isto, o que se procura entender é como se dá a interação existente entre os dois extremos do sistema e que fatores a influenciam. O foco é, portanto, o contexto da interação. Os estereótipos *PAcontext*, *PAclosedLoad* e *PAopenLoad* fornecem a semântica desejada para estudo e análise deste aspecto do projeto.

4 – Estudo de Caso

O estudo de caso apresentado neste artigo é o de uma aplicação de vídeo-conferência. Um sistema como este apresenta uma série de características rigorosas no atendimento de limites temporais que merecem especial atenção durante o seu projeto. Caso estes requisitos não sejam devidamente expressos, sua implementação poderá não atender às necessidades para as quais foi projetado [BOJKOVIC, MILOVANOVIC 2002]. Vários requisitos de QoS poderiam ser tratados neste estudo de caso, porém, restringiu-se o escopo deste trabalho à análise de pelo menos dois, latência e *jitter*. A seguir serão apresentados o problema e os modelos utilizados nesta análise.

4.1 – Descrição do Sistema

Uma descrição básica da funcionalidade do sistema pode ser dada nos seguintes termos: Tem-se duas ou mais estações de trabalho que desejam se comunicar enviando imagem e som para a(s) outra(s) estação(ões) via uma aplicação *web*. Por razões de simplicidade, será considerado somente o sub-caso no qual existem apenas duas estações no sistema, porém a adição de outras estações em nada dificulta a análise ou o projeto. Dado que a comunicação e a troca de mensagens são sempre ponto-a-ponto, o que limitará o número de estações constituintes do sistema será a capacidade de cada uma em atender solicitações de outras e também de receber e tratar pacotes dos outros nós da rede. Portanto, o sistema consiste numa distribuição *peer-to-peer*, na qual cada nó é ao mesmo tempo servidor e cliente. Como servidor cada nó recebe uma requisição de um cliente e aceitando tal requisição, passa a transmitir pacotes de vídeo para esta estação. Como cliente, uma estação realiza uma requisição acessando uma determinada

URL e aguarda sua aceitação. Sendo aceita, a estação cliente passa a receber pacotes de vídeo que devem ser reproduzidos localmente.

Após a descrição básica do sistema, serão apresentados os requisitos de desempenho que devem ser atendidos. Os seguintes requisitos de desempenho serão analisados:

1 - o retardo máximo (latência) admitido entre o pedido de conexão e a sua aceitação e confirmação não deve exceder 500 milissegundos em 98% dos casos. Logo a probabilidade do retardo ser maior que 500 milissegundos deve ser menor que 2%.

2 - a apresentação dos *frames* ao usuário deve seguir um intervalo regular de 30 milissegundos, sendo garantido que este intervalo deve ser cumprido sem atraso em 99% dos casos. Isto significa que a probabilidade do intervalo entre *frames* ser superior a 30 milissegundos (*jitter*) deve ser menor que 1%.

Para que seja possível realizar a análise sobre os requisitos apresentados, faz-se necessário o conhecimento de características de execução dos componentes do sistema. Algumas destas características, estimadas com base em dados apresentados em [BOJKOVIC, MILOVANOVIC 2002] e [HALSALL 2000], são listadas a seguir:

1 – tempo médio de processamento de um *frame* no servidor (sem contar o tempo de captura da imagem e do som): 10 milissegundos;

2 – número de pacotes necessários para enviar um *frame* pela rede: 50

3 – latência média da rede: 3 milissegundos;

4 – tempo médio de processamento de um *frame* no cliente: 15 milissegundos;

5 – o número de *frames* por conferência é uma variável do sistema: \$N;

6 – o número de usuários por conferência é uma variável do sistema: \$NUsers.

4.2 – Modelos UML-RT

Com base na descrição do sistema e seus requisitos, é possível a criação de diagramas que facilitam a análise do mesmo, e com isto ponderar sobre as possíveis alternativas de projeto. Os diagramas apresentados neste trabalho foram gerados pela ferramenta Real-Time Studio da Artisan [ARTISAN 2004].

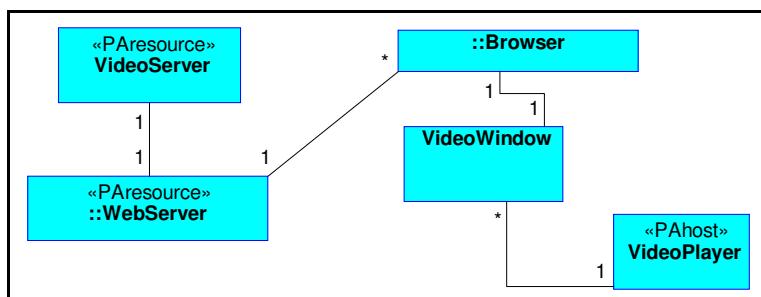


Figure 1. Diagrama de Classes

Na figura 1 observa-se uma visão parcial do diagrama de classes para a aplicação de vídeo-conferência. O diagrama de classes simplificado não apresenta os atributos nem as operações de cada classe. Optou-se por esta visualização para melhor destacar a utilização dos estereótipos no modelo. Pela observação do diagrama, a aplicação constitui-se, no lado cliente, basicamente de um navegador (**Browser**) pelo qual o usuário entra no sistema, um visualizador (**VideoWindow**) e um reproduutor de vídeo (**VideoPlayer**). Este último, por ser um recurso que pode ser compartilhado por mais de um visualizador, foi estereotipado como um **PAhost**. As classes que

representam a parte do servidor foram estereotipadas com *PAresource*, uma vez que apresentam características como capacidade de processamento, tempo de resposta e vazão da rede (*throughput*), todas relevantes para a análise do sistema.

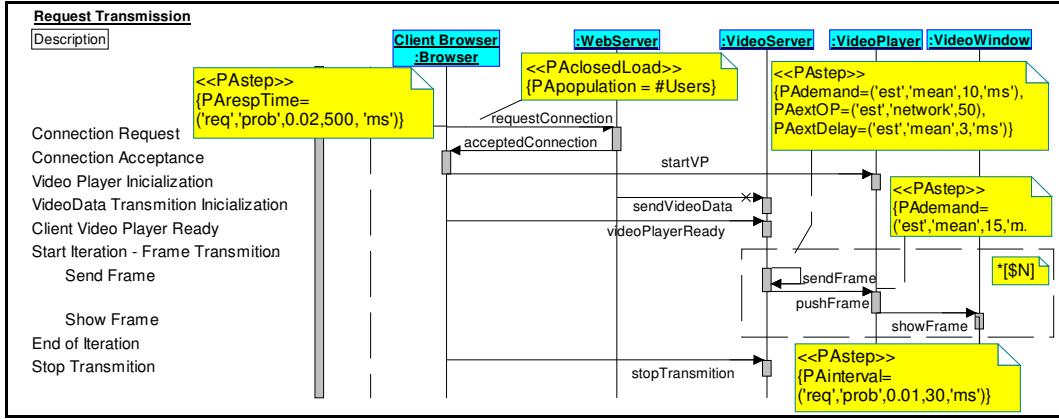


Figure 2. Diagrama de Seqüência

A partir do diagrama de classes e da descrição apresentada, escolheu-se o diagrama de seqüência para realização da análise dos requisitos temporais da aplicação. Esta escolha foi feita porque os objetivos principais da análise são a dinâmica de interação entre os componentes do sistema numa seqüência temporal, e o relacionamento entre os requisitos e características do sistema neste contexto. O diagrama de seqüência é a ferramenta mais adequada para esta tarefa.

Na figura 2, observa-se que os dois requisitos temporais descritos na sub-seção anterior encontram-se representados pelas *tags* (valores etiquetados) *PArespTime* (requisito do tempo de resposta para conexão), associada à chamada da função *requestConnection*, e *PAinterval* (requisito de intervalo entre *frames*), associada à execução da função *showFrame*. Na descrição destas *tags* encontram-se as abreviaturas “req” e “prob” que significam respectivamente “requisito” e “probabilidade”, além da unidade de tempo milissegundos, abreviada para “ms”.

As características dos componentes do sistema (tempo de processamento) encontram-se expressas através da *tag* *PAdemand*, enquanto as características de interação e contexto são expressas através das *tags* *PAextOP*, *PAdelay* e *PApopulation*. Na sua descrição encontram-se as abreviaturas “est” e “mean” que significam respectivamente “valor estimado” e “média”, além dos valores nominais característicos. Cada *tag* está associada a um momento da execução do cenário. Por exemplo, quando ocorre a chamada da função *sendFrame*, os fatores relevantes a serem analisados dizem respeito: 1) ao tempo de processamento daquele *frame* que será transmitido; 2) ao número de pacotes necessários para enviar um *frame*; e 3) ao retardo imposto pela rede. Estas informações estão descritas na *tag* ligada à seta que representa a chamada da função *sendFrame*.

Como se tem uma iteração das chamadas *sendFrame*, *pushFrame* e *showFrame* igual ao número de vezes quantos forem os *frames* a serem enviados (variável definida na descrição apresentada na sub-seção 4.1), destacou-se estas chamadas com um retângulo pontilhado. Isto destaca a representação da ocorrência das \$N execuções.

Deve ser ressaltada a expressividade do modelo apresentado na figura 2, uma vez que nele se encontram todas as informações necessárias para a análise dos dois requisitos de desempenho propostos.

5 – Conclusão

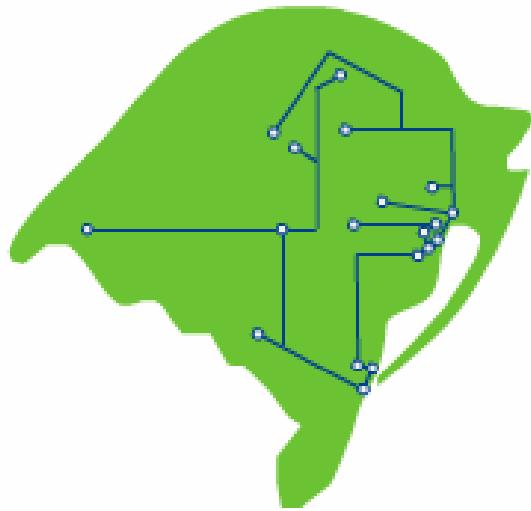
Este trabalho buscou chamar atenção para a possibilidade do uso da UML-RT na modelagem de sistemas interativos baseados em aplicações de rede. Como estes sistemas apresentam rigorosos requisitos de QoS, com prazos rígidos a serem cumpridos, tais como em sistemas de controle industrial, esta proposta mostra-se bastante pertinente.

Surge assim uma nova perspectiva para a utilização dos recursos providos pela UML-RT. Esta linguagem poderia ser utilizada para formalização de requisitos de QoS em contratos de serviços entre operadoras de telecomunicação e entre operadora e usuário. Desta forma, as partes contratantes ficariam obrigadas a cumprir exigências formalmente descritas segundo um modelo padronizado.

O estudo de caso apresentado permitiu visualizar o poder de expressão da extensão tempo-real da UML bem como sua adequação para modelagem de sistemas interativos baseados em rede. Deve-se considerar que atualmente a utilização de UML-RT neste tipo de sistema se mostra tão adequada quanto em sistemas que necessitam de concepção paralela de hardware e software. Segundo [SELIC 1999], o uso de UML no projeto destes sistemas também não era algo tão evidente na época em que foi proposto.

6 – Referências Bibliográficas

- [AAGEDAL, ECKLUND 2002] Aagedal, J. O., Ecklund, E. F “Modelling QoS: Towards a UML Profile”. In: <<UML>> 2002, Dresden, Germany, September 20 – Oct 4, 2002. Published in Spring LNCS 2460, ISBN 3-540-44254-5, pp. 275-289.
- [ARTISAN 2004] ARTISAN Software. Real-Time Studio. <http://www.artisansw.com/>.
- [BOJKOVIC, MILOVANOVIC 2002] Bojkovic, Z. S.; Milovanovic, D. A. Multimedia Comunication Systems: Techniques, Standards, and Networks, Prantice Hall, 2002.
- [DOUGLASS 1999] Douglass, Bruce P. Real-Time UML Second Edition – Developing Efficient Objects for Embedded Systems - Addison-Wesley, 1999.
- [EDWARDS 1997] Edwards, S., Lavagno, L., Lee, A., Sangiovanni-Vincentelli, A. “Design of Embedded Systems: Formal Models, Validation, and Synthesis”, In Proc. IEEE, vol. 85, pp. 366-390, March 1997.
- [HALSALL 2000] Halsall, F. Multimedia Communications: Applications, Networks, Protocols, and Standards, Addison-Wesley Publishing, 2000.
- [OMG 2004] OBJECT MANAGEMENT GROUP. UML Profile for Schedulability, Performance, and Time Specification, Feb 2004. <<http://www.omg.org/cgi-bin/doc?ptc/04-02-01>>.
- [SHAW 2001] Shaw, A. Real-Time Systems and Software, John Wiley& Sons, 2001.
- [SELIC 1999] Selic, B. “Turning Clockwise: Using UML in the Real Time Domain”, Communications of the ACM, v. 42, n.10: 46-54, Oct 1999.
- [TSAI, YANG, CHANG 1995] Tsai, J. P., Yang, S. J., Chang, Y. “Timing Constraint Petri Nets and Their Application to Schedulability Analysis of Real-Time System Specifications”, IEEE Transactions On Software Engineering. Vol.21, N. I, Jan 1995.
- [VOGEL 1995] Vogel, L. A. et al. “Distributed Multimedia and QoS: A Survey”, IEEE Multimedia, Summer 1995.



SESSÃO TÉCNICA 2

Segurança na troca de informações em uma rede sem fio no modo Ad Hoc

Floriano Ferreira Dos Reis Filho¹, Héctor Dave Orrillo Ascama¹ e Sergio Takeo Kofuji¹

¹Laboratório de Sistemas Integráveis – Universidade de São Paulo (USP)

CEP: 05508-900 - São Paulo - SP - Brazil

{floriano,hector,kofuji}@pad.lsi.usp.br

Resumo. Neste trabalho avalia-se um mecanismo de segurança sobre uma rede *Ad Hoc* com roteamento AODV (*On Demand Distance Vector*), este mecanismo é composto pelo protocolo WEP (*Wired Equivalent Privacy*) na camada de Enlace e o IPSec (*IP Security*) na camada de Rede. O IPSec é um protocolo das soluções não baseadas em padrões, utilizado na segurança de uma rede VPN (*Virtual Private Network*), realizou-se as devidas configurações e simulações em cenários reais para utilizá-lo em uma rede *Ad Hoc*. Pretende-se estimar como este mecanismo afeta o desempenho da rede *Ad Hoc* tendo-se como métrica a vazão de dados.

1. Introdução

Na atualidade, as pessoas utilizam algum tipo de dispositivo portátil com capacidade de se comunicar com a parte fixa de uma rede sem fio ou até com outros computadores móveis. Basicamente, existem dois tipos de Redes Móveis sem fio: as redes *Ad Hoc* e as redes de infra-estrutura.

Nas redes de infra-estrutura, as estações móveis devem estar em permanente contato com uma estação de suporte a mobilidade, conhecida como ponto de acesso. Em uma rede *Ad Hoc*, as estações móveis são capazes de se comunicar diretamente entre si, sem a necessidade de um ponto de acesso.

O objetivo principal deste trabalho é implementar segurança ao protocolo do tipo on-demand AODV (*Ad Hoc On-Demand Distance Vector Routing*), utilizando o WEP (*Wired Equivalent Privacy*) e o IPSec (*IP Security*).

Este artigo está organizado como segue. A seção 2 apresenta os conceitos necessários para o entendimento do trabalho. A seção 3 contém a descrição do cenário experimental. A seção 4 descreve as simulações e os resultados alcançados. Finalmente, a seção 5 contém as conclusões do trabalho desenvolvido.

2. Falhas de segurança nos protocolos de roteamento

Em uma rede *Ad Hoc*, o grau de comprometimento entre os nós é alto, o desempenho da rede depende de cada um dos nós, por estas características uma rede *Ad Hoc* se faz insegura.

Os aspectos de segurança nos protocolos de roteamento são críticos, pois muitos foram desenvolvidos sem considerar esta característica.

Neste tipo de rede cada nó deverá estar preparado para enfrentar um adversário, garantindo indiretamente maior grau de segurança para toda a rede. O principal ataque dos adversários é comprometer o processo de descoberta de rotas e aproveitar-se disto. Os pacotes de route request (*RREQ*) e route reply (*RREP*) podem ser alterados enquanto trafegam, ou podem ser forjados causando diversas anomalias no funcionamento da rede [Dahill, Levine, Royer e Shields 2002].

Ataques feitos nos protocolos de roteamento podem causar um comportamento anormal do tráfego de rede, podendo-se levar a negação de serviços DoS (*Deny of Service*) [Y. Zhang e W. Lee 2003].

A seguir apresenta-se alguns conceitos utilizados na proposta.

2.1. Protocolo de roteamento *AODV*

O protocolo *AODV* permite o roteamento em uma topologia que muda constantemente. É um protocolo reativo, constrói as rotas somente quando necessário. Ele utiliza um processo de inundação para descobrir as rotas, tenta aumentar a largura de banda disponível, minimizando o uso de mensagens periódicas para atualização de rotas.

O *AODV* fornece descoberta dinâmica de rotas e manutenção das mesmas entre os nós, utilizando links simétricos e três tipos de mensagens [Kullberg 2004]. Ele minimiza o número de broadcasts solicitados, pois cria rotas sob demanda, não precisa manter uma lista completa de rotas para todos os destinos possíveis.

Escolheu-se o protocolo *AODV* para este trabalho, devido aos seus dois objetivos principais:

- ✓ Descoberta de rota entre o nó de origem e o de destino.
- ✓ Armazenamento e gerenciamento de rotas ativas. [Royer e Toh 1999].

Para a descoberta de rota, utiliza-se dois processos: Pedido de Rota e Recebimento de Rota. O primeiro ocorre quando a estação de origem deseja enviar uma mensagem para outra estação e não tem uma rota válida para este destino. O segundo acontece quando se tem uma resposta, da rota que a estação de origem deseja, para estabelecer um canal de comunicação com a estação de destino.

2.2 Protocolo de segurança *WEP* e *IPSec*

A segurança é questão importante nas redes sem fio, por este motivo procurou-se utilizar protocolos de segurança, abaixo descritos, que operam na camada de enlace de dados e na camada de rede.

O *WEP* (*Wired Equivalent Privacy*) é o método de criptografia utilizado nas redes *wireless* 802.11. Opera na camada de enlace de dados e fornece criptografia, entre

as estações conectadas no modo *Ad Hoc*, com base no método *RC4 (Rivest Cipher)* da RSA, que usa um *IV (Initialization Vector)*, vetor de inicialização de 24 bits e uma chave secreta compartilhada de 40 ou 104 bits. O *IV* é concatenado com a chave secreta compartilhada para formar uma chave de 64 ou 128 bits que é usada para criptografar os dados. O resultado serve de entrada para um gerador de números pseudo-aleatórios, *PRNG (Pseudo Random Number Generator)*, que é baseado no algoritmo *RC4* [RSA Security Inc. 2003]. A saída do PRNG é uma seqüência pseudo-aleatória de bits baseada na chave composta e com o mesmo tamanho do texto a ser criptografado.

O *IPSec* agrupa proteção ao pacote IP, fornece protocolos para a segurança de tráfego de dados na rede *Ad Hoc*, autenticação de cabeçalho *AH (Authentication Header)*, segurança no encapsulamento do *payload*, conteúdo dos dados, *ESP (Encapsulating Security Payload)*, bem como protocolos para a gerência de chaves. Tendo-se o *IPSec* como opção para implementar *VPN (Virtual Private Network)*, a rede torna-se mais segura, pois os serviços podem ser utilizados por qualquer protocolo nas camadas superiores. Neste trabalho utilizou-se o *IPSec* para segurança das estações na rede *Ad Hoc* e não como ponte entre duas redes (*VPN*), como geralmente é utilizado, por possuir serviços para o controle de acesso, integridade, autenticação da origem dos dados e confidencialidade. No modo transporte, o cabeçalho *ESP* é inserido após o cabeçalho IP original, a criptografia do conteúdo dos dados é garantida. (fig.1).

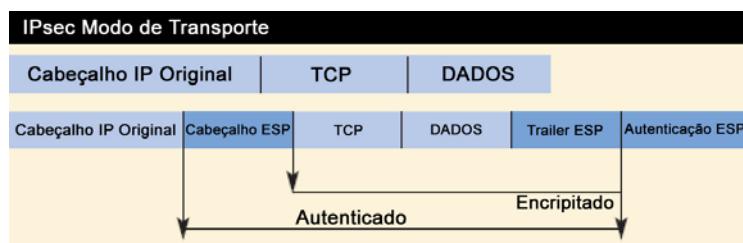


Figura 1. ESP após o cabeçalho IP original. [Meredith 2002]

3. Descrição do cenário para os testes

Criou-se um cenário prático de uma rede *Ad Hoc* de 2 nós. As simulações feitas com software não duplicam a realidade em um ambiente. Nele, fatores como taxa de sinal ruído varia e qualidade do *link* oscila devido à interferência de fatores climáticos, como a temperatura. Estas simulações ignoram as comunicações entre as camadas e não conseguem capturar os erros que ocorrem nas interações entre o sistema operacional, o hardware e o ambiente sem fio [Sanghari, Brown e Doshi 2003].

Embora o protocolo de roteamento *AODV* tenha sido desenvolvido [Royer 2003], não se teve um teste do mesmo na plataforma Windows, utilizando os protocolos da camada de enlace de dados e rede com um nível de segurança atribuído para suporte ao protocolo de roteamento.

3.1. Hardware e software utilizados

Para a realização do experimento, utilizou-se duas estações com a seguinte configuração: 1) Desktop com processador Intel Pentium 4 de 1.80 GHz e 768 MB de RAM. 2) Placas padrão IEEE 802.11b 11 Mbps *Wireless LAN* da 3Com, modelo 3CRDW696, pois fornecem conexão *Ad Hoc* segura e suporte 802.1x para *Windows XP*. 3) Sistema Operacional *Windows XP Professional* com *Service Pack 1*.

Para geração do tráfego em uma rede sem fio escolheu-se a ferramenta *Netperf* [Hewlett-Packard Company 2003], que permite gerar, receber e coletar dados, com o objetivo de fazer uma análise estatística da rede.

3.2. Configuração do cenário

Realizaram-se os testes em uma sala com 42 m², com um espaço de 1 metro entre as estações. Definiu-se um canal de comunicação 6, pois este canal trabalha na faixa de frequência compatível, entre 2426 a 2448 MHz, das interfaces wireless (2.4 GHz). A rede foi composta por duas estações, com endereços *IP* (*Internet Protocol*) privados, pois eles fazem parte de uma rede que será criada apenas para fins específicos em um cenário *Ad Hoc*.

O tamanho das mensagens transmitidas pela rede *Ad Hoc* varia entre 8 até 65.536 bytes. Para gerar o tráfego na rede utilizou-se a ferramenta Netperf. Cada nó atua primeiro como emissor e depois como receptor.

4. Resultados das experiências

Na fig. 2 mostra-se como os mecanismos de segurança *WEP* e *IPSec* afetam a vazão de dados. Os testes foram feitos da máquina A para a máquina B e vice-versa. Os resultados de cada um deles são apresentados nos gráficos.

Para estes testes, os parâmetros de vazão de dados e o tamanho das mensagens foram especificados para se evidenciar qual o ponto máximo onde pode-se ter a maior taxa de dados na transferência de uma mensagem.

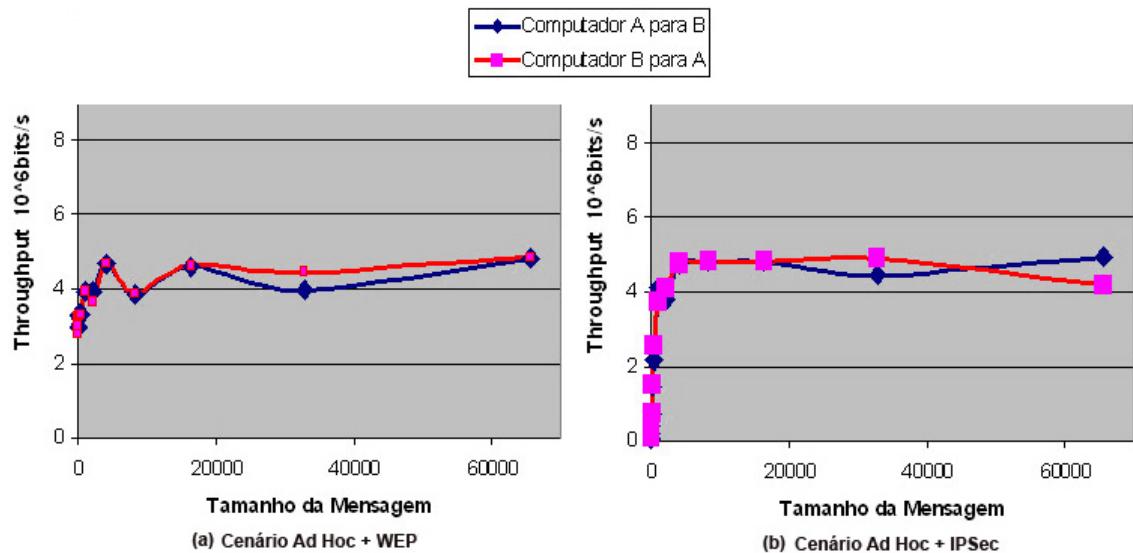


Figura 2. Vazão de dados no cenário sem fio.

A fig. 2a mostra que o nível maior para taxa de transferência foi 4,65 Mbps para tamanhos de mensagens de 16.384 bytes, para as duas estações. Quanto maior o tamanho da mensagem, maior a taxa de transferência de dados. Neste caso, com tamanho de mensagem de 65.536 bytes as duas estações alcançam a taxa máxima de transmissão no meio sem fio.

Na fig. 2b, o nível maior para taxa de transferência para as duas máquinas foi de 4,8 Mbps. Ambos os computadores depois de enviarem mensagens com o tamanho de 16.384 bytes tendem a diminuir a vazão de dados. Do computador B para o A, a taxa diminui para 4,19 Mbps com mensagem de 65.536 bytes. Do computador A para o B a taxa diminui para 4,48 Mbps. Com tamanho de mensagem até 16.384 bytes, a rede entra em um estado estável, aumentando-se o tamanho das mensagens a rede tem variações devido à sobrecarga para criptografia do conteúdo do pacote IP.

As linhas da fig. 3 correspondem à transmissão das mensagens entre o computador A e o computador B. Segue abaixo os cenários testados:

Cenário 1. *Ad Hoc Puro*

Cenário 4. *Ad Hoc com AODV*

Cenário 2. *Ad Hoc com WEP*

Cenário 5. *Ad Hoc com IPSec e AODV*

Cenário 3. *Ad Hoc com IPSec*

Cenário 6. *Ad Hoc com IPSec, AODV e WEP*

Consolidando-se os cenários, testados entre as duas máquinas, percebe-se o comportamento de uma rede *Ad Hoc* sem mecanismos de segurança e o efeito dos mecanismos que foram implementados.

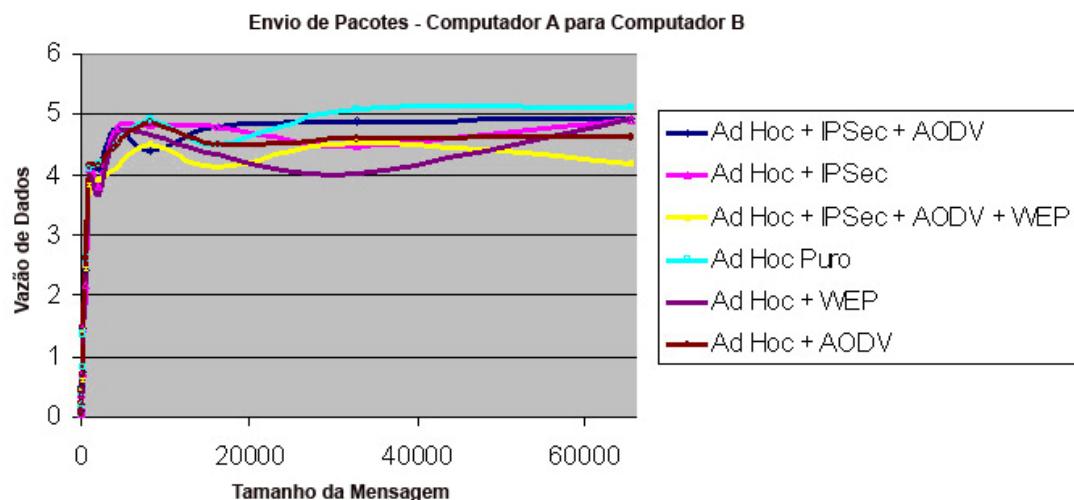


Figura 3. Vazão de dados nos seis cenários implementados.

Verifica-se que a sobrecarga do *IPSec* é maior que a do protocolo *WEP*. Este fato é devido aos mecanismos de segurança do *IPSec* (método de autenticação, confidencialidade do *ESP* e integridade do *SHA1*) serem mais sofisticados. Ambos, os protocolos, juntamente com o *AODV* degradam o desempenho da rede.

5. Conclusões

Neste trabalho apresentou-se um estudo sobre a influência da sobrecarga introduzida nas redes 802.11b pelos mecanismos de segurança *WEP* e *IPSec* sobre uma rede *Ad Hoc*. Os resultados obtidos mostraram que esses mecanismos introduzem informações adicionais de controle na rede que reduziram a vazão de dados se compararmos com o primeiro cenário, onde configurou-se apenas uma rede *Ad Hoc*, sem mecanismos de segurança.

Vê-se que os pontos máximos atingidos pelas transmissões são quase iguais, menos no caso da transmissão com o nível de segurança utilizando-se o protocolo *WEP*.

e o *IPSec*. Este comportamento é devido à sobrecarga introduzida na rede pelos mecanismos de segurança adotados. No *WEP*, além do tempo computacional requerido para as operações de criptografia e decriptografia, os quadros da camada de enlace possuem 8 bytes de informações adicionais, compostas pelo *IV* e pelo *ICV*. A sobrecarga se demonstra quando vemos a linha que corresponde ao cenário seis (*Ad Hoc* com *IPSec*, *AODV* e *WEP*) onde a vazão de dados diminui de forma significativa.

Para *WLANS* (*Wireless Local Area Network*) que não necessitam de um alto nível de segurança, recomenda-se utilizar o protocolo de criptografia *WEP* com 128 bits. Já em ambientes que a confidencialidade dos dados é uma prioridade, recomenda-se a utilização de *VPNs* (*Virtual Private Network*). Entretanto, se introduzirmos os protocolos *WEP* e *IPSec*, percebe-se na prática que a segurança não afeta de maneira eficiente a vazão de dados.

Observando-se as tecnologias de segurança, percebe-se, por outro lado, que há um legado de componentes, para rede sem fio com padrões antigos de tecnologias, utilizados no mercado. Em trabalhos futuros pode-se implementar novas simulações práticas e analisar o impacto dos novos mecanismos de segurança para redes sem fio (802.11i). Além disso, pode-se inserir mais dispositivos para possibilitar o roteamento de pacotes, com mais rotas disponíveis, na rede sem fio.

Referências

- B. Dahill, B. N. Levine, E. Royer e C. Shields. (2002) “A Secure Routing Protocol for Ad Hoc Networks”, Conference on Network Protocols (ICNP), pp. 2.
- Y. Zhang e W. Lee.(2003.) “Intrusion Detection in Wireless Ad Hoc Networks”, ACM Wireless Network Journal.
- Kullberg, Tuulia. (2004) “Performance of Ad-hoc On-Demand Distance Vector Routing”, Seminar on Internetworking, Helsinki University of Technology, pp. 1.
- RSA Security Inc. (2003), “RC4: Encrypty Algoritm of RSA Security”, <http://www.rsasecurity.com>
- E. M. Royer, Chai-Keong Toh, (1999) “A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks”, IEEE Personal Communications, Vol. 6, No. 2, pp. 46-55.
- M. P. Joseph, Corson. M. Scott. (1998) “Móbile Ad Hoc Networking and the IETF”. Mobile Computing and Communications Review, vol. 2, número 4.
- S. Sanghari, T. Brown, S. Bhandare. S. Doshi, (2003) “Ewant: The Emulated Wireless Ad Hoc Network Testbed”. Universidade do Colorado, pp 5, IEEE.
- E. M. Royer. (2003) “Ad Hoc On-Demand Distance Vector Routing”, Request for Comment RFC 3561.
- Hewlett-Packard Company, (2003) “Public Netperf Homepage”, <http://www.netperf.org/netperf/NetperfPage.html>
- Meredith, Gail. (2002) “Decoding IPSec, Understanding the Protocols of Virtual Private Networks”. Second Quarter. Cisco Systems.

Definindo e Usando Contexto Derivado de Multi-Sensores

Felipe Weber Fehlberg, Cláudio Fernando Resin Geyer, Iara Augustin, Adenauer Corrêa Yamin, Luciano Cavalheiro da Silva

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

`fwf@terra.com.br, geyer@inf.ufrgs.br, august@inf.ufrgs.br,`
`adenauer@inf.ufrgs.br, lsilva@inf.ufrgs.br`

Abstract. *The human interactions are enriched by several implicit factors, such as facial expressions and voice tune. The computer systems in general are not prepared to use that kind of data. Information about the execution environment of the equipments can be used to adapt its behavior to new situation. It can increase the quality of the service. This work presets a proposal of a Context Recognition System that will enable the use of several different sources of information to be used by a computations system. Using that high level information system will be able to adapt its behavior to the present Environment.*

Resumo. *A comunicação entre seres humanos é enriquecida por uma série de fatores implícitos, como expressão facial e entonação de voz. Os sistemas computacionais atuais geralmente não estão preparados para utilizar este importante tipo de dado. Informações sobre o contexto de execução dos equipamentos podem ser importantes para adaptar seu comportamento a diferentes situações, oferecendo desta forma, serviços de maior qualidade aos usuários. Este trabalho apresenta a proposta de um Sistema de Reconhecimento de Contexto que capacitará o uso de informações de diversos sensores para geração de dados de Contexto possibilitando o processo de adaptação das aplicações ao ambiente no qual estão inseridas.*

1. Área de Pesquisa

A computação ubíqua é uma visão sobre o futuro da tecnologia e foi inicialmente proposta por Mark Weiser [WEISER, 1991]. Esta visão, entretanto, ainda reside num futuro distante. Uma visão da proposta de Weiser, mais próxima da realidade tecnológica, foi proposta pela IBM e recebeu o nome de Computação Pervasiva (*Pervasive Computing*). Segundo está proposta, o ambiente computacional estará espalhado entre os dispositivos, utilizando, tanto a

infra-estrutura existente dos computadores ligados fisicamente à rede, quanto elementos móveis.

No cenário atual existe uma grande quantidade de dispositivos diferentes integrados através das redes de computadores. A popularização de dispositivos móveis já teve início com PDAs e celulares. Estes dispositivos são responsáveis por proporcionar ao usuário a mobilidade física, ou seja, a capacidade de movimento mantendo acesso aos recursos disponíveis. Além da mobilidade física dos dispositivos, os softwares deverão ser dotados de mobilidade lógica. A mobilidade lógica é capacidade de componentes de softwares movimentarem-se entre dispositivos para prover um serviço de melhor qualidade ao usuário, isto é, um componente de software poderá migrar de um PDA para um computador com capacidade de processamento maior e posteriormente retornar o resultado da computação ao usuário através do PDA, tudo isso de maneira transparente.

A consciência do contexto será responsável por detectar os recursos disponíveis no ambiente, possibilitando seu acesso pelos usuários e suas aplicações e adaptando as características das aplicações a esta disponibilidade.

Segundo a visão do grupo ISAM, um ambiente de Computação Pervasiva deverá ser construído através da integração dos conceitos da computação em grade, computação móvel e computação consciente do contexto [YAMIN, 2004].

2. Computação Consciente do Contexto

A comunicação entre seres humanos é enriquecida por uma série de fatores implícitos, como expressão facial e entonação de voz. A comunicação que envolve computadores seja ela feita com outros computadores, seja com seres humanos, é, na maioria das vezes, explícita e carente de complementos que agreguem significado às informações.

Esta comunicação poderia ser enriquecida com dados do ambiente no qual os computadores estão inseridos, habilitando-os a ter uma atuação mais eficaz. Este é um exemplo da área de atuação da Computação Consciente do Contexto (context-aware computing).

Ao contrário do método tradicional de entrada de dados em computação, as informações do contexto são utilizadas implicitamente. Uma aplicação é consciente do contexto se é capaz de adaptar-se às condições do ambiente (contexto considerado).

Um dos desafios da Computação Consciente do Contexto é interpretar a grande quantidade de informações disponíveis, interpretando-as e determinando ações a partir dessas interpretações. Pode-se, além disso, combinar informações provenientes de diferentes fontes (sensores) e fundi-las em uma nova informação com mais significado agregado [SCHMIDT, 1998].

Como os dispositivos da Computação Pervasiva têm, em geral, recursos limitados, é necessário otimizar seu uso através da utilização de recursos disponíveis no ambiente (fazendo uso de outros computadores com recursos de processamento disponíveis) [DEY, 2000].

Atualmente o desenvolvimento de aplicações conscientes do contexto é uma tarefa complexa e trabalhosa. Esta situação pode ser remediada com desenvolvimento de uma infra-estrutura

de suporte às tarefas comuns do desenvolvimento de aplicações conscientes do contexto modelagem do contexto e tratamento das informações [HENRICKSEN 2002].

3. Trabalhos Relacionados

Os principais projetos que fornecem uma estrutura para aquisição e tratamento de informações de contexto são: Context Toolkit [DEY, 2000], Solar [CHEN, 2004], Aura [JUDD, 2003]. Maiores informações sobre estes trabalhos podem ser obtidas no estudo realizado como trabalho individual [FEHLBERG, 2005].

O Context Toolkit é um framework para implementação de aplicações conscientes do contexto. A manipulação das informações do contexto se dá através componentes que precisam ser programados previamente. Estes componentes fazem parte da aplicação e sua reutilização em outras aplicações não é feita de maneira automática. A integração de informações do contexto não é uma tarefa simples segundo esta arquitetura.

Os autores do projeto Solar propõem a abstração de Operadores, objetos que processam informações do contexto e que podem ser combinados para a obtenção de um tratamento específico dos dados. Neste projeto, as informações de contexto são tratadas como fluxos de dados. Os Operadores devem ser compatíveis com os fluxos de dados para que eles sejam combinados. Isso implica um alto grau de acoplamento entre os sensores e os Operadores não sendo simples a tarefa de realizar a composição destes. Esta característica dificulta o tratamento e o uso de informações do contexto.

No projeto Aura é proposta uma camada de software que torna homogêneo o acesso aos dados do contexto. O acesso às informações é feito através de uma linguagem inspirada em SQL que é interpretada por esta camada de software. Este modelo, entretanto, não tem a característica de utilizar as informações de contexto de maneira implícita. As consultas ao contexto são realizadas explicitamente, de maneira semelhante a uma consulta a um banco de dados. É a aplicação que deve, portanto, requisitar informações sobre o contexto. Seria desejável que a aplicação pudesse ser notificada de modificações do contexto, podendo, desta forma, adaptar-se.

4. Projeto ISAM

O projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis) [YAMIN, 2004] tem como objetivo o desenvolvimento de uma infra-estrutura de suporte necessária para a implementação das aplicações móveis distribuídas com comportamento adaptativo em um ambiente da computação pervasiva. A consciência do contexto está presente tanto nas aplicações quanto nos serviços do middleware.

4.1. Subsistema de Reconhecimento de Contexto

O projeto ISAM propõe uma infra-estrutura para produção de informações do contexto de alto nível. Este serviço é parte integrante do middleware de execução e produz informações que podem ser utilizadas tanto pelas aplicações quanto pelo próprio middleware. Os dados são coletados em sua forma bruta e são processadas pelas camadas do sistema de reconhecimento de contexto até que sejam geradas informações de alto nível. O Sistema de

Reconhecimento de Contexto trabalha com informações já disponíveis na EXEHDAbase. Os elementos Sensore, Monitor e Coletor são responsáveis por disponibilizar a informação bruta sensoriada à base. Os demais elementos: agregador, tratador, preditor e notificador realizam a tarefa de gerar e tratar as informações de alto-nível.

- O Sensor é um abstrai a forma como as informações são obtidas do meio ambiente e as disponibiliza ao middleware. Desta forma é possível o isolamento da complexidade sobre o tratamento de diferentes tipos de dados dos sensores nestes componentes, deixando que o sistema de reconhecimento de contexto seja focado na tarefa de produzir dados de alto nível.
- O Monitor é responsável por reunir todas as informações dos Sensores em um determinado nodo e transferir estas informações à base.
- O Coletor recebe informações provenientes de diversos nodos através de seus Monitores e às disponibiliza ao Sistema de Reconhecimento de Contexto.
- Agregador trabalha sobre do mesmo tipo, aplicando funções como média e somatório sobre os dados.
- Tradutor transforma uma informação em outra. Pode ser usado para conversão de grandezas entre diferentes sistemas de medidas, por exemplo.
- O Preditor se baseia no histórico dos dados para gerar uma previsão sobre seu comportamento futuro.
- O Notificador informa às aplicações registradas, as alterações nos dados do Contexto, possibilitando sua adaptação.

5. Dados provenientes de multi-sensores

Em geral, os softwares de aquisição e tratamento de informações de contexto enfatizam o uso de uma única fonte de dados (mono-sensor). Laerhoven [LAERHOVEN, 2002] diz que o tratamento do contexto derivado de multi-sensores é um problema ainda a ser aprofundado.

A importância da utilização de diversas fontes de dados para produção de contexto de alto-nível é facilmente identificável. Os sensores existentes fornecem dados sobre uma determinada visão do ambiente, esta visão não é, na grande maioria das vezes, completa. Um detector de pressão sobre o assento de uma cadeira não é suficiente para geração do alto-nível sobre a localização de um determinado indivíduo, mas em conjunto com outros sensores pode fornecer ao sistema uma visão mais completa do mundo.

O relacionamento entre dados de diferentes sensores acrescenta uma complexidade na interpretação do contexto pelas aplicações, já que existe uma variabilidade muito grande entre as possíveis características dos dados dos sensores. Dados estáticos ou dinâmicos, ou ainda de diferentes domínios podem ser relacionados para a obtenção do contexto desejado.

Neste sentido, foi identificada a necessidade do sistema de reconhecimento de contexto do projeto ISAM ser capaz de gerar informações de alto-nível baseando-se em diferentes fontes de dados.

5.1 Integrando dados de sensores diferentes

Está sendo desenvolvida a modelagem de um Sistema de Reconhecimento de Contexto capaz de integrar informações oriundas de multi-sensores para serem utilizados no desenvolvimento de aplicações conscientes do contexto dentro do cenário da Computação Pervasiva disponibilizado pela arquitetura ISAM.

Neste sentido, planeja-se o desenvolvimento de uma linguagem de composição de elementos de Contexto que deverá ser utilizada pelas aplicações pervasivas para descrever o tratamento das informações brutas dos sensores até a obtenção de dados de contexto de alto-nível.

Levando-se em consideração que freqüentemente novos dados de sensores podem ser acrescentados à infra-estrutura de execução, a linguagem projetada deve ser capaz de descrever tratamentos específicos que podem ser necessários para esses novos dados. Isto pode acontecer com a carga dinâmica de código desenvolvido para este propósito. A carga dinâmica de código de execução poderá ser feita utilizando os próprios mecanismos existentes no middleware de execução EXEHDA.

Esta linguagem deverá ser interpretada em tempo de execução por um componente que será desenvolvido e disponibilizado no middleware, o Compositor de Contexto (ver Figura 1). Este novo elemento da arquitetura deverá ser capaz de atender requisições de diferentes aplicações interpretando dados dos sensores e gerando informações de Contexto de alto-nível.

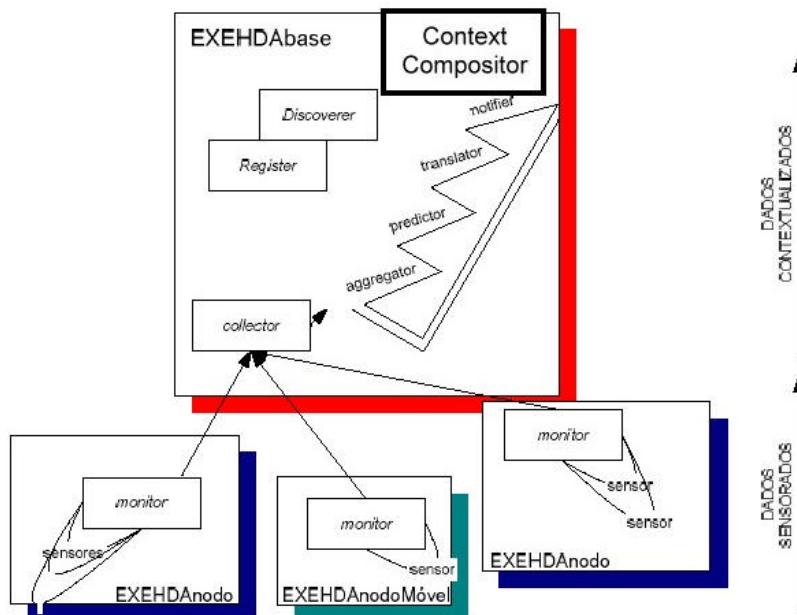


Figura 1. Sistema de Reconhecimento de Contexto.

Também é previsto o desenvolvimento de uma aplicação que faça uso das características oferecidas pelo sistema de reconhecimento do contexto de maneira que se possa validá-lo em situação real de utilização.

6. Conclusões

Informações sobre o Contexto de execução podem ser utilizadas pelas aplicações para disponibilizar um serviço com características apropriadas às diferentes situações que o ambiente pode apresentar. O uso destes dados na adaptação, no entanto, não é trivial. Ainda não surgiu um trabalho que resolva os principais problemas envolvidos neste processo.

Este trabalho apresenta uma proposta em desenvolvimento na Universidade Federal do Rio Grande do Sul que objetiva a criação de uma infra-estrutura capaz de fornecer dados de Contexto de alto-nível para aplicações Pervasivas. Os resultados obtidos serão tema de trabalhos futuros.

Referências

- AUGUSTIN, I. Abstrações para uma linguagem de Programação visando Aplicações Móveis Conscientes do Contexto em um Ambiente de Pervasive Computing. Porto Alegre; PPGC da UFRGS, 2003. 193 p. (tese de doutorado).
- CHEN, G. Solar: Building A Context Fusion Network for Pervasive Computing. 2004. - DARTMOUTH COLLEGE, Hanover, New Hampshire, USA. (tese de doutorado).
- DEY, A. K. Providing Architectural Support for Building Context-aware Applications. 2000. Georgia Institute of Technology, Atlanta, USA. (tese de doutorado)
- FEHLBERG, F. Middlewares para Reconhecimento do Contexto na Computação Pervasiva.
- HENRICKSEN K.; INDULSKA J.; RAKOTONIRAINY A. Modeling Context Information in Pervasive Computing Systems. Proceedings of the First International Conference on Pervasive Computing. Springer-Verlag p.167-180. 2002
- JUDD G.; STEENKISTE P. Providing Contextual Information to Pervasive Computing Applications. Proceedings of the First IEEE International Conference on Pervasive Computing and Communications. Computer Society, 133. 2003.
- LAERHOVEN, K. V.; SCHMIDT, A.; GELLERSEN H. Multi-Sensor Context Aware Clothing. In the Proceedings of the Sixth International Symposium on Wearable Computers, 2002, p. 49--56.", 2002
- YAMIN, A.. Arquitetura para um ambiente de Grade Computacional Direcionado à Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva. Porto Alegre; PPGC da UFRGS, 2004. (tese de doutorado)
- WEISER, M. The computer of the 21st Century. Scientific American, [S.l.], v.265, n.3, p.66–75, Setembro de 1991.

Uma Proposta de Mecanismo para Controle de Admissão para a Rede Backbone UMTS Empregando Serviços Diferenciados

Paulo Dias de Alecrim, Paulo Roberto Guardieiro

Faculdade de Engenharia Elétrica – Universidade Federal de Uberlândia (UFU)
Campus Santa Mônica - 38.400-902 – Uberlândia – MG – Brasil

alecrim@netsite.com.br, prguardieiro@ufu.br

Resumo. Neste trabalho propõe-se um mecanismo que efetue o controle de admissão nos pontos de ingresso à rede backbone UMTS empregando DiffServ. Tal mecanismo provê um tratamento diferenciado ao tráfego de diversos fluxos na rede. Os resultados obtidos a partir de modelagem e simulação demonstram que o mecanismo proposto trata o tráfego de forma diferenciada, oferecendo um certo nível de QoS mesmo em condições de mobilidade dos usuários.

1. Introdução

Sistemas UMTS são baseados em IP e abrem as portas para a comunicação e serviços de multimídia. A UMTS é uma tecnologia que suporta voz e dados em pacotes, oferecendo taxas máximas de transmissão em banda larga de até 2 Mbps e de 384 Kbps quando a unidade móvel está em movimento.

Nestes sistemas a interligação de unidades móveis tais como, celular ou um PDA (*Personal Digital Assistant*) à Internet, é um caminho sem volta. Para tanto, seus backbones, deverão prover mecanismo de QoS para garantir o funcionamento adequado de tais aplicações. Em vista disso, propõe-se neste artigo um mecanismo que efetue, controle de admissão, classificação de pacotes, policiamento e condicionamento de tráfego para a rede backbone UMTS. O mecanismo proposto é baseado na diferenciação de serviços denominado pelo IETF de *DiffServ* e usa a classificação de pacotes como mecanismo para obtenção de QoS, sendo uma extensão dos trabalhos [M.Puuskari 1999] e [Ericsson 2003]. Tal mecanismo é aplicado nos SGSN (*Serving GPRS Support Node*) e GGSN (*Gateway GPRS Support Node*) do backbone UMTS. Para tanto, foram avaliados parâmetros de QoS como atraso e vazão para as classes de tráfego UMTS.

Este trabalho está organizado conforme descrito a seguir. Na seção 2, apresenta-se a arquitetura UMTS. Na seção 3, descreve-se o problema abordado. Na seção 4, apresenta-se a proposta de mecanismo para provimento de QoS. Na seção 5, trabalhos relacionados são apresentados. Na seção 6, descreve-se o modelo de simulação. Na seção 7, analisam-se os resultados obtidos através das simulações. Finalmente, na seção 8, apresentam-se as conclusões finais.

2. Arquitetura da rede UMTS

As redes backbone UMTS serão totalmente baseadas no protocolo IP (*All IP*). Para a integração da rede backbone UMTS com a Internet é necessário um suporte eficiente do IP móvel, proposta da IETF para o suporte de mobilidade na camada de rede [Nichols 1998]. A abordagem do 3GPP (*Third Generation Partnership Project*) [3GPP 2002], [B3G/4G 2004] para a integração com o IP Móvel é baseada em nós da rede backbone

UMTS chamados de SGSN e GGSN. A Figura 1 ilustra uma arquitetura UMTS simplificada composta de uma rede backbone UMTS conectada à rede de acesso UTRAN (*UTRAN – UMTS Terrestrial Radio Access Network*) através da interface Iu. A rede backbone UMTS é conectada à rede IP externa através da interface Gi e a UTRAN ao equipamento do usuário UE (*UE-User Equipment*) com a interface Uu [M. Puuskari 1999].

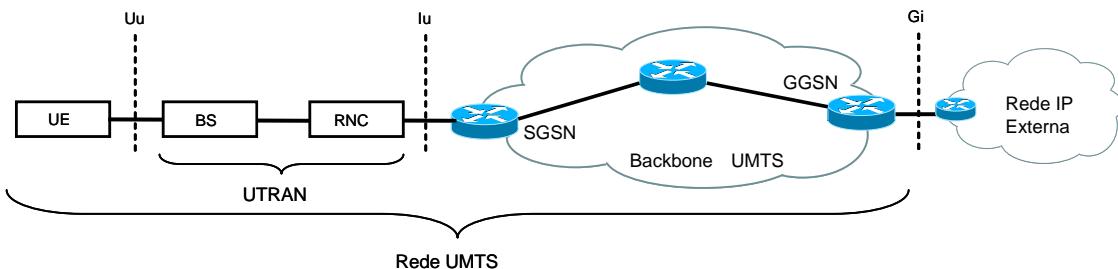


Figura 1. Arquitetura UMTS simplificada.

3. Definição do Problema

A demanda por aplicações de tempo real que possam ser acessadas a partir de unidades móveis suportadas pelas redes IP tem tido crescimento rápido nos últimos anos. Entretanto, ainda existem obstáculos para que tais aplicações sejam providas com QoS não só na rede de acesso sem fio, mas também na rede backbone UMTS [3GPP 2002]. Antes que uma unidade móvel possa efetivar a comunicação com a rede de dados externa, o contexto PDP deve ser ativado e nesta fase o perfil de QoS é negociado com a rede backbone UMTS. Quando uma unidade móvel solicita uma conexão com a rede, um túnel é estabelecido entre o terminal e o GGSN. Este túnel, conhecido na UMTS como contexto PDP, provê o UMTS *Bearer Service*. A rede backbone UMTS permite que sejam definidos perfis de tráfego para o qual o contexto pode se adequar (conversacional, *streaming*, interativa ou *background*), entre outros atributos que podem ser negociados [B3G/4G 2004]. Contudo, nenhum mecanismo de QoS é definido para dar suporte a esses perfis e garantir o seu cumprimento. QoS é um dos atributos mais importantes de um contexto PDP e é negociado durante ativação do PDP.

4. Proposta de Mecanismo para Provimento de QoS em Redes Backbone UMTS

Dentro do contexto apresentado na seção 3, propõe-se neste artigo um mecanismo de QoS baseado em *DiffServ* que efetue o controle de admissão nos pontos de ingresso à rede backbone UMTS. A avaliação do desempenho deste mecanismo será através de modelagem e simulação dos parâmetros de QoS como atraso e vazão para as classes de tráfego UMTS.

Em [3GPP 2002] e [M.Puuskari 1999] especifica-se a arquitetura de Serviços Diferenciados como a tecnologia recomendada para fornecer diferentes níveis de serviço. Os mecanismos de QoS especificados em [M. Puuskari 1999] são aplicados nos SGSN e GGSN como, controle de admissão, classificação de pacotes, policiamento e condicionamento de tráfego. Tal arquitetura é de simples implementação nos roteadores atuais e, portanto, tem sido apontada como um bom padrão para implementar QoS em redes backbone UMTS [GSM 03 60 1997].

5. Trabalhos Relacionados

A necessidade de provisãoamento de QoS na rede backbone UMTS, foi objeto de pesquisa em [M.Puuskari 1999], [3GPP 2002], [Dias 2001] e [GSM 09 60 1998]. Contudo, estes trabalhos não propuseram mecanismos para alocação de banda em função do controle de admissão para cada uma das classes de QoS UMTS de forma individual, conforme foram apresentados nas seções 2 e 3. Assim sendo, neste trabalho propõe-se realizar alocação de banda para cada classe de forma a oferecer um certo nível de QoS a cada uma delas em condições de congestionamento, além de distribuir a banda passante não utilizada entre as demais classes, caso uma delas não utilize toda a banda a ela alocada.

6. Modelo de Simulação

Por meio de modelagem e simulação, utilizando a ferramenta de simulação ns-2 [NS-2], foi realizado um estudo de desempenho do mecanismo de QoS proposto. A Figura 2 ilustra o cenário utilizado nas simulações, considerando-se o nó 1 como um SGSN e o nó 2 como um GGSN da rede backbone UMTS conforme apresentado na seção 2.

As fontes de tráfego escolhidas são uma combinação de todas as classes de tráfego UMTS, tais como: conversacional, *streaming*, interativa e *background*. Tomando este modelo de tráfego e a estrutura *DiffServ*, o PHB EF foi a opção escolhida para acomodar a classe conversacional, porque garante parâmetros mais rígidos. As outras três classes de tráfego UMTS podem ser acomodadas pelos PHBs AF, que são mais flexíveis. A classe *streaming* pode corresponder à classe AF4, a classe interativa à classe AF3 e a classe *background* corresponderia às classes AF2 ou AF1, ou mesmo à classe de melhor-esforço [Mohamed 2003], [Brian 2002].

O tráfego denominado de fluxo F1 é gerado a partir da agregação de fontes, que possuem a exigência de atrasos semelhante a da classe de tráfego conversacional. O fluxo F2 representa o tráfego *streaming*. O tráfego interativo F3 vem da agregação de tráfegos do tipo HTTP. Finalmente, o fluxo *background* F4, que modela o tráfego, tipicamente de e-mails e de fluxos de FTP, foi utilizado para congestionar a rede. F1 possui alta sensibilidade a atrasos e *jitter*. F2 tem uma alta sensibilidade a *jitter* e F3 tem uma alta sensibilidade a atrasos. F4 não é sensível a atrasos ou *jitter*.

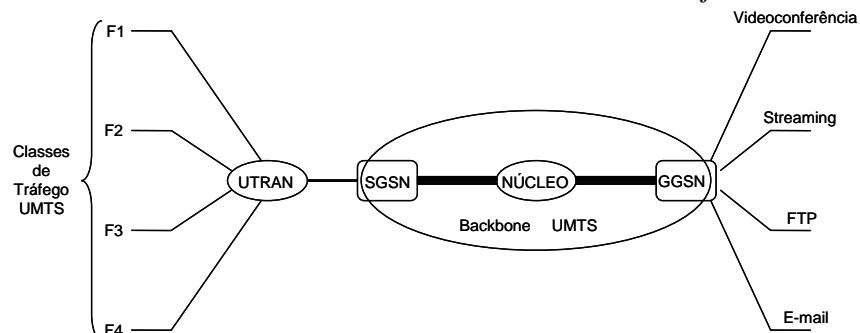


Figura 2. Modelo de Simulação.

Um classificador de pacote é responsável por marcar cada pacote, nas bordas da rede backbone UMTS (SGSN e GGSN), com o DSCP apropriado. A Figura 3 ilustra o algoritmo da disciplina de escalonamento para cada roteador no modelo que garante o encaminhamento de acordo com o DSCP de um pacote. O pacote entrante que chega a um nó de borda tem o seu QoS mapeado conforme o seu próprio PHB e então é

encapsulado com o DSCP adequado em seu cabeçalho. Em seguida, o pacote aceito é enfileirado. O escalonamento que usa o serviço de disciplina tipo PQ (*PQ-Priority Queueing*), encaminha os pacotes para o próximo passo. Este algoritmo foi escolhido em função do mesmo apresentar menor atraso para aplicações, tal como a classe conversacional F1 (mais sensível ao atraso).

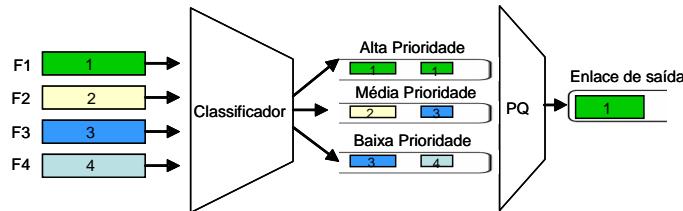


Fig. 3. Estrutura do algoritmo PQ aplicado aos roteadores.

7. Apresentação e Análise de Resultados

Nesta seção apresentam-se os resultados de simulação do mecanismo proposto na seção 4, bem como uma análise destes resultados. Para isso, foram avaliados parâmetros de QoS como atraso e vazão para as classes de tráfego UMTS.

Todas as classes de tráfego UMTS apresentaram uma taxa de ocupação inicial de 40%. Assim sendo, foi provocado um congestionamento na rede com o aumento do tráfego através de fluxos FTP até um limite de 100%. A avaliação do atraso fim a fim e da alocação de largura de banda foi caracterizada quando a intensidade de tráfego oferecida variou entre 40% a 100% de sua taxa inicial. Em relação aos requisitos apresentados pelos fluxos quanto aos itens de atraso e largura de banda, verifica-se que o atraso fim a fim sofrido pelo tráfego gerado pelo Fluxo 1 é menor que os valores encontrados pelas fontes do Fluxo 2. Esta característica está de acordo com o esperado, pois no escalonamento realizado por PQ a maior prioridade é dada ao tráfego de classe conversacional Fluxo 1, como faz para o EF PHB. Desta forma, assim que houverem pacotes dessa classe a serem transmitidos, sua fila será atendida em prioridade à fila do Fluxo 2, sem contudo interromper a transmissão de um pacote do Fluxo 2, caso os pacotes de Fluxo 1 tenham chegado durante a sua transmissão.

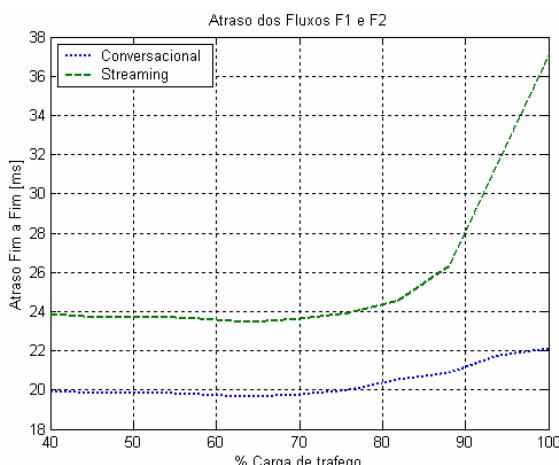


Figura 4. Atraso fim a fim (F1, F2).

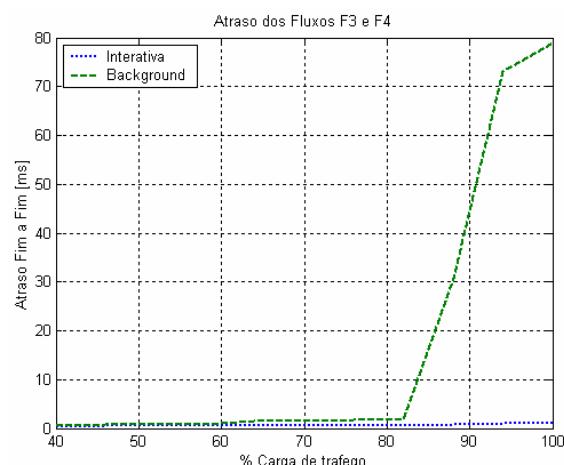


Figura 5. Atraso fim a fim (F3, F4).

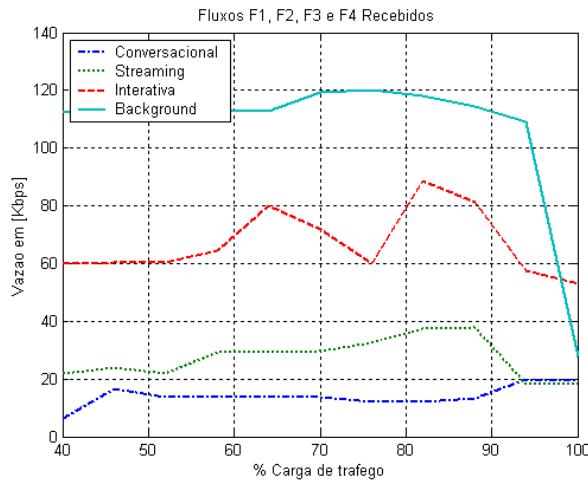


Figura 6. Vazão das classes F1, F2, F3 e F4 versus % Carga de tráfego.

As Figuras 4 e 5 mostram o atraso apresentado por cada aplicação e a Figura 6 mostra a vazão para cada fluxo quando a carga de tráfego varia de 40% a 100%. Pelo gráfico da Figura 6 observa-se que a vazão se manteve estabilizada para o fluxo F1 durante todo o tempo de observação. Este resultado já era esperado, pois em função da política do algoritmo PQ ocorreu descarte dos pacotes com menor prioridade, tendo o seu valor de pico nos pontos de maior congestionamento da rede, isto é, acima de 90%. Analisando a Figura 4, com o uso de *DiffServ* observa-se que o fluxo F1 obteve menores atrasos do que F2 de forma expressiva e estabilizada durante todo o período observado mesmo com o aumento da taxa de congestionamento na rede. Os atrasos dos fluxos F2 (Fig. 4), F3 e F4 (Fig. 5) se justificam em face da escala adotada, que pode ser explicado pelas longas filas escolhidas em face da ação do algoritmo PQ. Um tamanho menor de fila teria resultado em atrasos menores. O atraso dos fluxos F2, F3 e F4 foi superior durante todos os períodos observados, apresentando seu pico no ponto de maior congestionamento da rede, isto é, acima de 90%. Os pacotes dos fluxos F3 e F4 foram atrasados e sua entrega continua depois de finalizada a entrega dos pacotes dos outros fluxos. Durante este período final seu atraso foi reduzido, pois não havia tráfego na rede.

8. Conclusões

Neste trabalho foi proposto um mecanismo para controle de admissão para a rede backbone UMTS empregando Serviços Diferenciados para o provisionamento de QoS. Através de modelagem e simulação de inúmeros cenários com condições de tráfego e configurações diferentes, obtiveram-se resultados consistentes sobre o desempenho da rede em função da implementação do referido mecanismo. Por meio de controle de admissão e políticas adequadas, os resultados obtidos confirmam que a arquitetura *DiffServ* pôde diferenciar classes de tráfegos dentro da rede backbone UMTS, garantindo um certo nível de QoS a cada uma das quatro classes de tráfego definidas pelo 3GPP.

9. Referências

- 3GPP (2002), Technical Specification Group Services and System Aspects; QoS Concept and Architecture, 3G TS23.107 version 3.1.1.
- Andersson, Christoffer.(2001), GPRS and 3G Wireless Applications. Canadá: Editorial John Wiley & Sons, Inc.
- B3G/4G, (2004), Applications and Services. IEEE Wireless Communications, Outubro, Vol. 11 Nº 5.
- Brian Carpenter, Kathleen Nichol, (2002), “Differentiated Services in the Internet”, IEEE Proceedings, volume 90, pages 1479-94.
- Dias, K. L., Sadok, D. F. H.(2001), Internet Móvel:Tecnologias. Aplicações e, Livro Texto dos Minicursos – SBRC, Maio.
- Ericsson, (2003), Redes GSM / GPRS / UMTS, disponível em:
www.3gamericas.org/portuguese/Technology_Center/WhitePapers/.
- GSM 09 60 (1998), Technical Specification: General Packet Radio Service (GPRS); GPRS Tunneling Protocol (GTP) across the Gn and GP Interface.
- GSM 03 60 (1997), ETSI Technical Specification: General Packet Radio Service (GPRS); Service Description.
- GSM 04 60 (1999), “Digital Cellular Telecommunication System (Phase 2+), General Packet Radio Service (GPRS), Radio Link / Medium Access Control (RLCMAC),” v.6.3.0, March.
- M. Puuskari, (1999), “Quality of Service Framework in GPRS and Evolution towards UMTS,” 3rd European Personal Mobile Communication Conference, March.
- Mohamed A. EL - Gendy, Abhijit Bose, ang G. Shin: (2003), Evolution of the Internet and support for soft real- time applications. Proceedings of the IEEE 91 (7): 1086-1104.
- Nichols, K.; Blake, S.; Baker, F.; Black, D.(1998), *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers RFC 2474*, IETF.
- NS-2, Simulador de Redes Wireless: The VINT Project. The *Network Simulator ns-2: Documentation*. UC Berkeley.

Design and Simulation of the IEEE 802.16a physical layer

K. Jalandhar Reddy¹, S.Srikanth²
Student¹, Professor²

Dept. of Electronics, MIT, Anna University, India

k.jalandhar@gmail.com¹, srikanth@au-kbc.org²

Abstract

Fixed Broadband Wireless Access (BWA) is a technology that offers high-speed voice, video and data services for the “last mile,” which is presently dominated by the cable and digital subscriber line (DSL) technologies. The biggest advantage BWA has over its wired competitors is its increased capacity and ease of deployment, since networks can be created in just weeks by deploying a small number of base stations on buildings or poles to create high-capacity wireless access systems. The biggest obstacles for BWA deployment, on the other hand, are the cost of the customer premises equipment (CPE) and the overall performance of the system [1]. The recently approved IEEE 802.16a [2] is a new BWA standard operating in the 2-11 GHz range, and is touted as an important step towards widespread adoption of this technology.

The objective of this project is to model and simulate the IEEE 802.16a OFDM physical layer using C language. The simulations will compare the performance of different receiver models, while noting their complexity. The developed simulation will also provide a framework for developing new receiver models for 802.16a systems.

I. Introduction

Fixed Broadband Wireless Access (BWA) is a new “last mile” access technology that offers high-speed voice, video, and data services, which is presently dominated by the cable and digital subscriber line (DSL) technologies. BWA has a big advantage over its wired competitors in terms of capacity and ease of deployment. Wired broadband solutions require some existing infrastructure, like cable or telephone lines, which make deployment to underserved areas problematic and costly. Additionally, installation and maintenance of these networks typically require a technician to visit the customer premises, increasing the overall costs for the Internet Service Provider. BWA requires little infrastructure, and could theoretically be user-installed. Networks could be created in a short time by deploying a small number of base stations on buildings or poles to create high-capacity wireless access systems [1]. However, the wide-scale adoption of BWA will be strongly determined by its ability to overcome cost and performance barriers. If BWA can meet these challenges, it could easily be the next big revolution in wireless similar to Wireless LANs [3, 4].

The IEEE Wireless Metropolitan Area Networks (WirelessMAN) Standard 802.16a is a new standard for BWA [2]. Announced on January 30, 2003, this extension of the 802.16 standard covers fixed broadband wireless access in licensed and unlicensed spectrum from 2 to 11 GHz. Chair of the 802.16a working group Roger Marks proclaims, “The new IEEE 802.16a standard reshapes the broadband landscape. It closes the first-mile gap, giving users an easily installable, wire-free method to access core networks for multimedia applications [3].” License-exempt Wireless Internet Service Providers (WISPs) are thus given a boost through this standardization.

Some analysts believe that WISPs will become a big factor in shaping the broadband landscape in the years to come [4].

The goal of this project is to address one of the major hurdles in deployment of BWA, which is its performance within the hostile wireless channel. This project aims to evaluate the effects of different receiver models on the performance of the 802.16a physical layer specification. The design of the OFDM receiver is not part of the IEEE 802.16a standard, and is thus left up to manufacturers to come up with robust and cost effective implementations. The project will evaluate different receiver algorithms through simulation in C language. The IEEE 802.16a standard specifies a 256-point transform OFDM. This project will simulate the 256-point transform OFDM physical layer, since this air interface is mandatory for operation in license-exempt bands [5]. The project will model, simulate, and then, evaluate different receiver algorithm in terms of their performance, while noting the computational complexity for each of these algorithms.

The project will produce a written report, oral presentation, and an 802.16a physical layer simulator that will be released to the public. These deliverables could provide a good starting point for companies and universities interested in 802.16a. The report will evaluate some standard receiver models, while the simulator will provide a framework for developing and testing new models. Finally, the oral presentation will give a graduate-level class an overview of 802.16a, a look at some models for OFDM implementation, and an example of a communications system simulator.

II. Approach

The project will be composed of four major phases: research, modeling, simulation, and analysis.

A. Research

Initially, an intensive study of the 802.16a OFDM physical layer shall be performed, where the key issues in the design of an OFDM system that communicates through a fixed broadband wireless channel will be thoroughly analyzed. The issues facing OFDM and its performance in the indoor wireless channel have been studied extensively in the literature [6, 7, 8]. These results and analyses shall then be extended into the fixed broadband wireless case.

B. Modeling

The 802.16a OFDM physical layer will be modeled and simulated in C language. C language is a dataflow programming language that is suitable for communication system modeling and simulation [9]. C language provides a relevant interface that is useful for interactive control of key simulation parameters, which provides better intuitive understanding of the system.

C. Simulation

A C language system simulation will provide a suitable environment for the effective analysis of the performance of the 802.16a OFDM physical layer. A waveform level, discrete, baseband model will be developed, where semi-analytical Monte Carlo simulation runs will estimate the system performance. The semi-analytical simulation gives a good trade-off between computational complexity and simulation accuracy between a full Monte Carlo simulation and a purely analytical analysis, respectively [10]. Various simulation parameters can be automatically chosen by the simulator for predefined simulation runs, or interactively

changed by the user while a simulation is running. These parameters include a choice between the uplink and downlink scenario, the various rates supported by the standard (QPSK, 16-QAM, and 64-QAM), and the length of the cyclic prefix.

The block diagram of an 802.16a OFDM physical layer model is shown in Fig. 1.

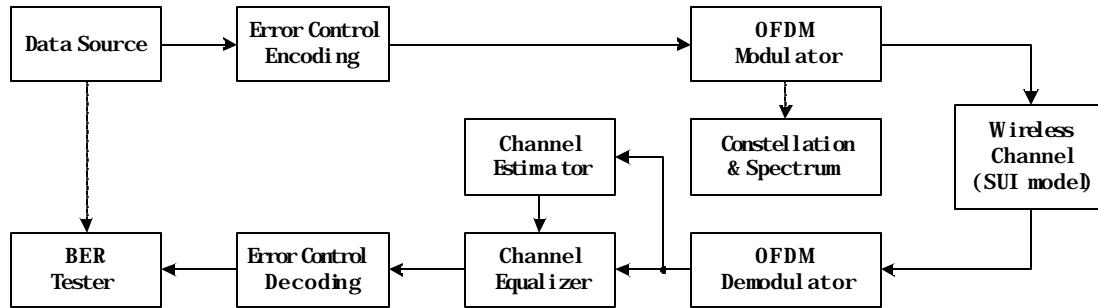


Figure 1. 802.16a Physical Layer Block Diagram

Each block shall be modeled as follows:

- a. **Channel:** A discrete channel model based on the Stanford University Interim (SUI) [11] model will be used. This model has a set of six typical channels which were selected for the three terrain types that are typical of the continental US [12]. These wireless channels are characterized by path loss, multipath delay spread, fading characteristics, Doppler spread, co-channel and adjacent channel interference, and antenna gain reduction factor. The user will also be given the option of modifying the channel characteristics while the simulation is running.
- b. **OFDM Modulator and Demodulator:** The OFDM modulator and demodulator blocks would be developed using standard IFFT and FFT blocks provided in C language. Perfect clock, symbol, and frame synchronization is assumed, and the effects of synchronization error will not be discussed in this project.
- c. **Channel Estimator:** The channel estimator will obtain a depiction of the channel state to combat the effects of the channel using an equalizer. Block estimation algorithm, the Least Squares (LS) [6] will be analyzed in terms of performance and complexity.
- d. **Channel Equalizer:** The channel equalizer will use the output from the channel estimator to ameliorate the effects of the channel and improve the performance of the system. A frequency domain equalizer will be thoroughly investigated in terms of its performance and complexity. Also, the performance of the system with and without the equalizer will be evaluated.
- e. **Error Control Coding:** Error control coding is essential for OFDM systems since it compensates for the bit errors that are inevitable in times of deep fade in the channel. The Reed Solomon encoder and decoder, convolutional encoder and Viterbi decoder, and the randomizer/de-randomizer and interleaver/de-interleaver will be implemented, and its effect on the overall bit error rate (BER) performance will be analyzed.

- f. **Performance Analysis Blocks:** To determine the qualitative and quantitative performance of the system and give a good intuitive understanding of the effects of certain parameters on the system, C language blocks that display performance curves will be developed for the simulation. These include bit error rate testers, spectrum analyzers, and constellation plotters.

D. Analysis

After generating the model of the OFDM system and performing simulation runs for various permutations of the system parameters (details of which are provided in the next section), extensive analyses on the results of the simulation will be performed. The effect of the channel parameters, the channel estimation method, the equalization structure, and the error control coding on the overall system BER performance will be deduced and compared to the theoretical results presented in the various papers [7, 8]. Key insights on efficient 802.16a receiver design and implementation will be discussed and clearly presented. The initial top-level C language transmitter and receiver block diagrams have already been developed and are shown in figure 2 and figure 3.

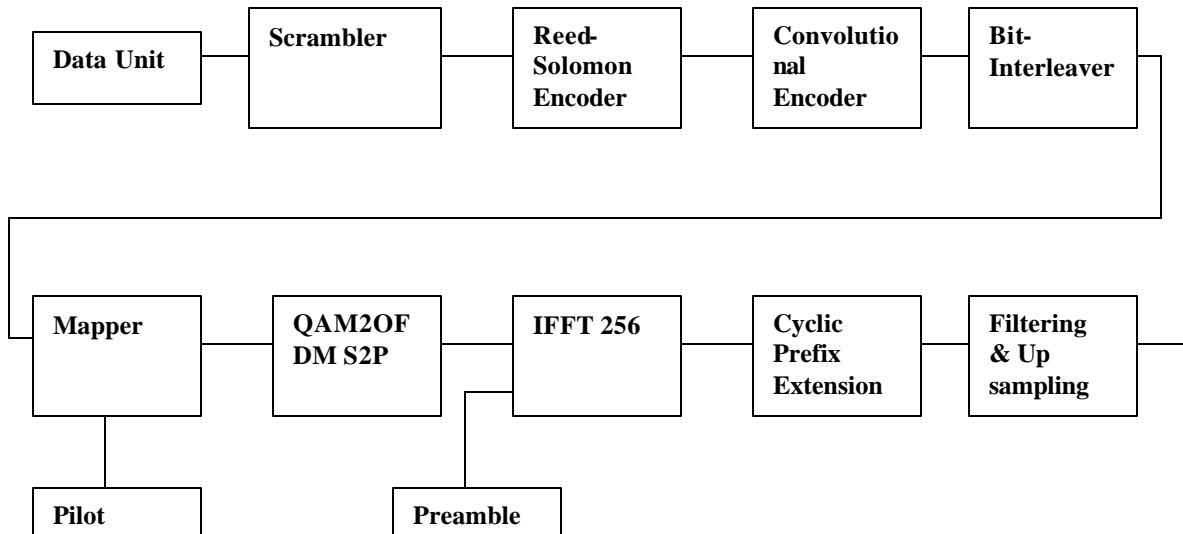


Figure 2: Transmitter block diagram

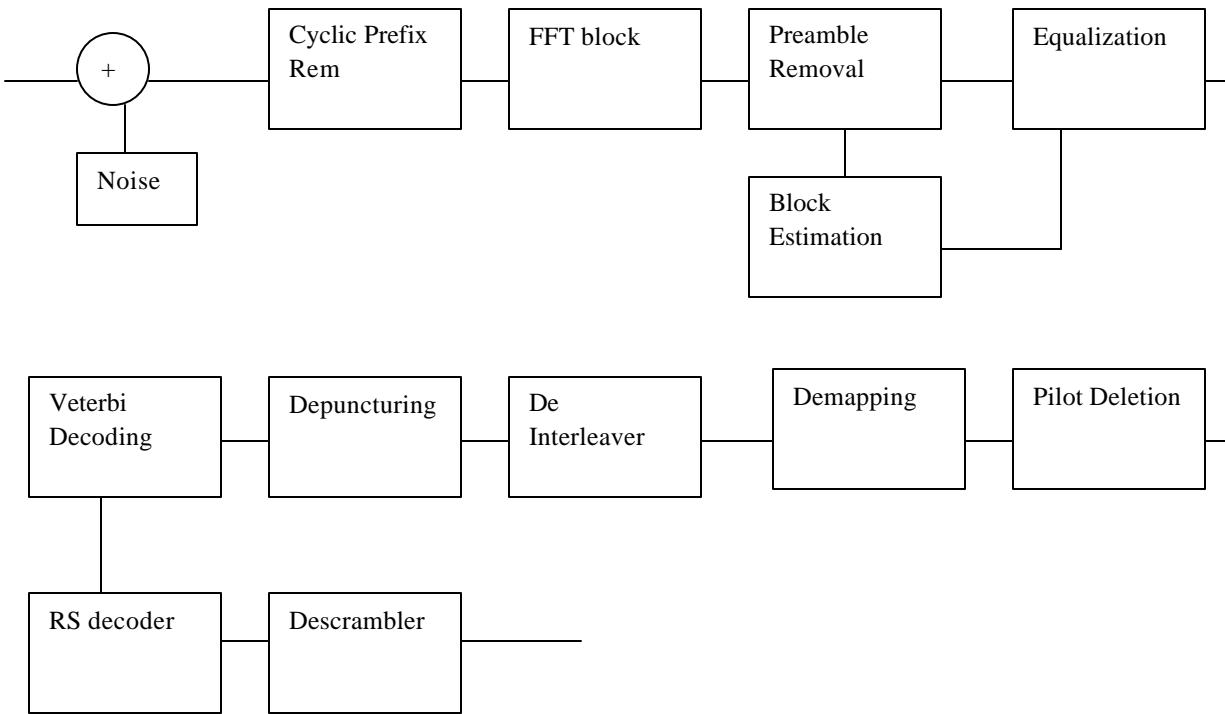


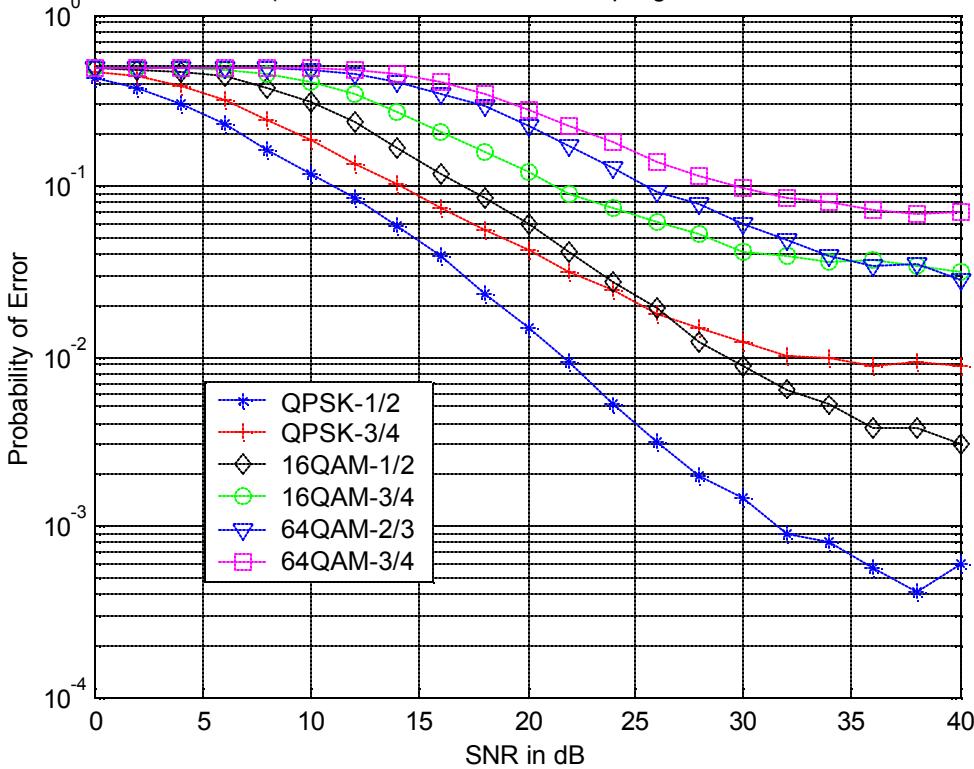
Figure 3: Receiver block diagram

III. Expected Outcomes

The first and most significant deliverable of this project will be a C language simulation of the OFDM physical layer of an 802.16a system. This includes modulation and decoding of OFDM and error control coding. An implementation of block estimation algorithm with a frequency domain equalizer will be performed using this simulation for six different bit rates specified in the standard. Six different channel models specified by Stanford University (Stanford University Interim models) will be used to simulate the broadband wireless channel.

A key performance measure of a wireless communication system is the BER. Curves of BER versus the signal to noise ratio (SNR) will be generated for the above modes of operation. These curves will also be plotted for a system that does not use channel coding, and thus the coding gain of the system will be determined. The BER curves will be used to compare the performance of the estimation algorithms used. Computational complexity of each model will be noted. Fading plots for the different channel models indicating the signal strength as a function of time will also be generated. These will provide a comprehensive evaluation of the performance of the OFDM physical layer for different states of the wireless channel.

BER Performance Vs SNR (for: 10 MHz BW, 8/7 oversampling rate, SUI-1 channel, Block Estimation)



This project will provide the researchers and the class an in-depth understanding of the advantages and challenges involved in the deployment of a fixed broadband wireless network. A keen understanding of OFDM as specified in the 802.16a standard and various receiver algorithms for OFDM will be gained. The simulation tool that will be developed as part of this project could also be distributed to chip and equipment manufacturers to test their designs for commercial 802.16a hardware. This tool could also be used by other researchers in the field of wireless broadband access to experiment with new algorithms for 802.16a systems.

Broadband wireless networks are key to promising applications like high-speed wireless Internet access and multimedia services such as video conferencing. BWA is expected to address the “last mile” issue that is currently dominated by DSL and cable technologies. Fixed wireless networks may prove to be extremely useful in regions where deployment of wired systems is not feasible. This project will provide valuable insight on the challenges faced and obstacles that have to be overcome before the successful commercial deployment of 802.16a networks could be realized.

References

- [1] I. Koffman and V. Roman, “Broadband Wireless Access Solutions Based on OFDM Access in 802.16”, *IEEE Communications Magazine*, Apr. 2002.
- [2] IEEE 802.16a, “Air Interface for Fixed Broadband Wireless Access Systems. Part A: Systems between 2–11 GHz”.

- [3] K. Mackie, "IEEE Approves 802.16a WirelessMAN Standard, Broadband Wireless Business Magazine", Feb. 3, 2003, Feb. 8, 2003,
<<http://www.shorecliffcommunications.com/magazine/news.asp?news=1211>>.
- [4] "Unlicensed Spectrum Drives Wireless Broadband Infrastructure Beyond Wi-Fi", *In-Stat MDR market report*, <<http://www.instat.com/press.asp?Sku=IN020617WN&ID=517>>.
- [5] C. Eklund, R. B. Marks, K. L. Stanwood and S. Wang, "IEEE Standard 802.16a: A Technical Overview of the WirelessMAN™ Air Interface for Broadband Wireless Access." *IEEE Communications Magazine*, vol. 40, no. 6, pp. 98-107, Jun. 2002.
- [6] M. Engels, *Wireless OFDM Systems: How to Make Them Work?*, Kluwer Academic, 2002.
- [7] S. Coleri, M. Ergen, A. Puri and A. Bahai, "Channel Estimation Techniques Based on Pilot Arrangement in OFDM Systems," *IEEE Transactions on Broadcasting*, vol. 48, no. 3, pp. 223-229, Sep. 2002.
- [8] X. Tang, M. Alouini and A. J. Goldsmith, "Effect of Channel Estimation Error on M-QAM BER Performance in Rayleigh Fading", *IEEE Transactions On Communications*, vol. 47, no. 12, pp. 1856, Dec. 1999.
- [9] P. Das and D. Koch, "On the use of Visual Programming Languages for Communication System Simulation," *IEEE Proceedings of Southeast Con*, Apr. 1991.
- [10] M. C. Jeruchim, P. Balaban and K. S. Shanmugan, *Simulation of Communication Systems: Modeling, Methodology, and Techniques*, Kluwer Academic, 2000.
- [11] V. Erceg, K. V. S. Hari, M.S. Smith, D. S. Baum, "Channel Models for Fixed Wireless Applications," *Contribution to IEEE 802.16.3*, Jul. 2001.
- [12] V. Erceg, L. J. Greenstein, S. Y. Tjandra, S. R. Parkoff, A. Gupta, B. Kulic, A. A. Julius and R. Bianchi "An Empirically Based Path Loss Model for Wireless Channels in Suburban Environments", *IEEE Journal on Selected Areas in Communications*, Jul. 1999.

Utilização de Agentes em um Modelo de Autorização para Acessos a Dados Médicos em Ambiente de Computação Móvel

Erico M. H. do Amaral¹, Gerson A. Soares², Raul C. Nunes^{1,2}, Roger C. Machado²

¹Curso de Ciência da Computação – Universidade Federal de Santa Maria
Av. Roraima – Bairro Camobi – Santa Maria - RS - Brasil

² Programa de Pós Graduação em Engenharia de Produção – Centro de Tecnologia
Universidade Federal de Santa Maria
Av. Roraima – Bairro Camobi – Santa Maria - RS - Brasil

{erico, gerson, ceretta}@inf.ufsm.br, cavilhas@gmail.com

Resumo. A crescente disponibilização eletrônica de dados médicos salienta a necessidade de mecanismos de controle de acesso efetivos. Este artigo descreve um modelo de autorização para acesso a dados médicos utilizando agentes em um ambiente sem fio. O objeto de estudo e consequente proteção configura-se em um sistema de monitoramento de curvas cardíacas em pós-operatório, denominado *CardioMonitor*, onde questões como segurança de informações em prontuário eletrônico são essenciais.

Abstract. The increasing of the electronic patient data claims by effective access control mechanisms. This paper describes an authorization model that uses agents on controlling health information on wireless environment. The goal is to protect the cardiac waves collected from a net of pacemakers. The monitoring system is called *CardioMonitor* and it is used on post-operative step. The collected waves are saved on an electronic way and its access must be protected.

1. Introdução

Marca-passos cardíacos são equipamentos que começam a ser utilizados em redes [HUTTEN, 1997] e [BRAECKLEIN, 2004]. Entretanto, a segurança do sistema é um desafio. Equipamentos que monitoram pacientes e coletam dados para diagnósticos necessitam dar garantias que o acesso aos equipamentos com informações do paciente (telemetria) e/ou aos seus dados somente sejam realizados por pessoal autorizado. Este artigo explora questões de autenticação no âmbito do projeto *CardioMonitor* [MAZZUTTI, 2003], um sistema de monitoramento de curvas cardíacas em pós-operatório.

De acordo com a Resolução 1.639 do CFM [CFM, 2002], o sistema de informações e a qualidade do serviço referente a dados de prontuário eletrônico em atividade médica deverá manter a integridade da informação para segurança dos processos de sistema. Conforme a norma ISO/IEC 15408, a integridade da informação é

obtida através do controle de vulnerabilidades, de métodos fortes de autenticação, de restrições de acesso e métodos de processamento dos sistemas operacionais.

Baseado nestas legislações, neste artigo determina-se uma política de segurança mínima capaz de abranger estes requisitos. Para tal, desenvolveu-se um modelo de autenticação que utiliza agentes e que deverá prover os níveis de segurança desejados.

O artigo está estruturado da seguinte forma: na seção 2 descreve-se o projeto CardioMonitor; na seção 3 avalia-se à tecnologia *wireless*, especificamente no que se refere às normas IEEE 802.11x e IEEE 802.15; a seção 4 avalia questões quanto à política de segurança, que servem de base para a definição do modelo de autenticação proposto; e, finalmente, na seção 5 estabelece-se as considerações finais.

2. Projeto CardioMonitor

Segundo [MAZZUTTI, 2003], o sistema de monitoramento cardíaco CardioMonitor destina-se ao monitoramento de pacientes internados em unidades de tratamento intensivo (UTI). O *software* do sistema controla os dados coletados nos equipamentos de monitoramento cardíaco e repassa esta informação à uma central, onde pode-se visualizar, processar e/ou armazenar os dados.

Os coletores de sinais são integrados a *palmtops* providos de dispositivos *bluetooth* capazes de transmitir e receber dados via ondas eletromagnéticas. Os dados coletados nos dispositivos são tratados e transformados em curvas graficamente contínuas, permitindo o acompanhamento do estado dos pacientes. Após a coleta os dados são transmitidos, via rede sem fio *bluetooth*, para uma central de monitoramento equipada com um microcomputador capaz de receber o sinal de vários emissores e mostrá-los em gráficos independentes, possibilitando assim, uma análise médica individual de cada paciente a partir desta central.

Descrição Técnica

O Sistema em análise, segundo [NUNES, 2002], consiste em um *handheld* atuando como eletrocardiógrafo e marca-passo de demanda, com transmissão *wireless* dos sinais por intermédio de uma conexão *bluetooth*. O *handheld* é ligado a um circuito para condicionamento dos sinais obtidos através de eletrodos ligados ao coração de um paciente em fase pós-operatória. O sinal amplificado é filtrado para a redução de ruídos e convertido do modo analógico para o modo digital. O sinal digitalizado é então transmitido em protocolo serial para o *handheld*. Um *software* executa no *handheld* para a exibição dos sinais cardíacos na tela do mesmo e para o controle das funções do marca-passo. Ao mesmo tempo em que isto é realizado, o sinal é transmitido pela conexão *bluetooth* para um computador central.

O Sistema funciona como uma rede de sensores monitorada local e remotamente, e é composto por nós responsáveis pelo sensoriamento e pelo envio das informações coletadas (PDA's) a um nó que agrupa informações. Este nó pode ser um nó comum da rede ou um nó de maior capacidade. Em todo o caso, a informação concentra-se na direção de um ponto centralizador. Observa-se que este ponto centralizador é de fundamental importância para a segurança da rede e que a comunicação *wireless* torna-se um ponto crucial para garantir a privacidade e confidencialidade dos dados do paciente e o sigilo profissional.

3. Tecnologias de Comunicação sem fio

É necessário que se faça uma breve descrição da norma IEEE 802.15, utilizada no escopo do projeto CardioMonitor, bem como da norma IEEE 802.11x, que trata sobre redes *wireless* em geral, uma vez que a proposta apresentada neste trabalho não restringe-se apenas à comunicação entre dispositivos *bluetooth*.

A especificação de WPAN, segundo [IEEE, 2002], determina que fazem parte de uma rede pessoal sem fio todos os dispositivos capazes de se conectar entre si, utilizando dispositivos *bluetooth*, conforme preconiza a norma IEEE 802.15, onde os mesmos podem formar redes utilizando conexões através de radio freqüência, através da licença ISM (*Industrial, Scientific, Medical*), de 2.4 Ghz de banda, com modulação FM, e utilizando um esquema TDD (*Time-Division Duplex*) para emular uma transmissão *full duplex*, sobre este canal, a informação é transmitida em pacotes que são agrupados em *slots* de tempo, e cada um deles pode ser enviado em saltos de freqüência.

O padrão IEEE 802.11x é definido em 6 métodos de modulação, e as técnicas de segurança estão implementadas no padrão 802.11i. O padrão 802.11b foi o primeiro modelo amplamente aceito e utilizado em conexões *wireless*, o padrão sugerido para a aplicação no projeto é o 802.11g, aprovado em junho de 2003, que trabalha sobre uma banda de 2.4 Ghz, com o diferencial de operar a uma taxa de transmissão de até 54 Mbps. Esta taxa de transferência pode ser influenciada pela interferência de vários produtos que utilizem ondas de radio freqüência, devido à banda utilizada. O método de detecção de colisões no meio de transmissão, adotado neste padrão é conhecido como CSMA/CA que diminui a capacidade de transmissão por este canal. O padrão 802.15 oferece interferência na comunicação do padrão 802.11g, ambos adotados no escopo deste trabalho, devido a esta característica, cuidados especiais na implementação do ambiente *wireless* sobre o padrão 802.11g devem estar bem claros e definidos na especificação do projeto.

4. Modelo de Autenticação

Nesta seção descreve-se a especificação da política de segurança implementada e, com base nesta, a arquitetura sistema de autenticação.

4.1. Política de Segurança

Todo modelo de autenticação necessita de uma política de segurança. No Brasil, a norma NBR/ISO 17799 define um “código de prática” para a gestão de segurança da informação e, na área médica, o CFM regula a utilização de sistemas informatizados. Especificamente no que diz respeito a modelos de autenticação, a resolução 1.639/2002 do CFM define as “Normas Técnicas para o Uso de Sistemas Informatizados para a Guarda e Manuseio do Prontuário Médico”. Quando armazenados, os sinais cardíacos coletados em equipamentos cardíacos tornam-se parte integrante do prontuário médico, logo, no âmbito do CardioMonitor, a política de segurança deve considerar tanto a norma NBR/ISO 17799 quanto a resolução 1.639/2002 do CFM.

Para atender a norma NBR/ISO 17799 e a resolução 1.639/2002 do CFM, o sistema CardioMonitor adota uma política de segurança que define um modelo de autenticação que: (1) possibilite a criação de diferentes perfis de usuários; (2) utilize mecanismos de acesso restrito e limitado a cada perfil; (3) identifique cada usuário

através de um método de autenticação seguro; e (4) possua um certificado digital que autentique a transmissão remota de dados do prontuário.

4.2. O Serviço

Para atender a política de segurança, o serviço de autenticação de acesso a dados médicos deve definir perfis de usuários e prover níveis de acesso diferenciados a cada um deles. Além disto deve definir um mecanismo simples e eficiente que habilite o acesso aos dados. A existência de um agente conectado a um servidor de autenticação é a opção adotada no CardioMonitor para prover dinamicidade ao serviço.

O agente estabelece comunicação com três repositórios mantidos pelo serviço a fim de garantir confidencialidade e integridade. Os repositórios armazenam informações específicas dos clientes que acessam a rede, tal como: a lista de clientes autorizados a conectar-se a rede, os perfis de usuário e o histórico de acesso dos clientes. Desta forma, o agente possibilita manter a integridade do perfil preestabelecido para os usuários.

O uso de agentes no modelo de serviço é um instrumento efetivo para o gerenciamento da segurança das informações e atende de forma rápida e eficiente as necessidades de um ambiente computacional sem fio, respeitando a política de segurança pré-estabelecida.

4.3. O Agente

A função do agente é monitorar requisições de acesso e detectar a conexão de dispositivos ao ambiente de rede, mantendo registros relativos às atividades realizadas por estes equipamentos.

A figura 1 ilustra o diagrama de caso de uso do serviço de autenticação por um dispositivo/usuário. Inicialmente o usuário conecta-se com o serviço de autenticação, e este tenta instalar o agente de autenticação no dispositivo (Mensagem Aut./Val.). O usuário então responde aceitando ou rejeitando a instalação do agente. Caso o usuário aceite o acesso é liberado, caso contrário bloqueado.

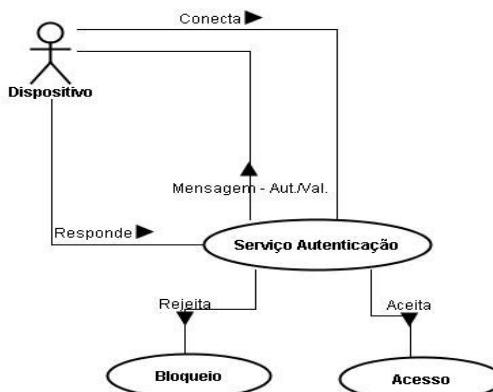


Figura 1. Diagrama Use-case

Salienta-se que para garantir integridade e confidencialidade o agente carrega a chave que possibilita o acesso. Neste processo o quesito segurança da informação é levado em consideração desde o inicio do projeto, o que garante uma facilidade maior no desenvolvimento prático do sistema proposto, permitindo que requisitos indispensáveis para a proteção das informações deste modelo não sejam omitidos.

O diagrama de seqüência descrito na figura 2 descreve as possíveis iterações de um usuário com o serviço de autenticação. Na primeira iteração o usuário aceita o agente e o acesso é liberado. Na segunda iteração o usuário rejeita a instalação do agente no dispositivo e consequentemente tem o acesso negado. Observa-se que sempre que um novo dispositivo efetua uma requisição de acesso, o serviço de autenticação envia um agente solicitando a habilitação do mesmo. A referência de cada dispositivo é determinada pelo seu ID, que é identificado pelo MAC address de sua placa de rede, ou conector *bluetooth*, em conjunto com o *login* e senha, informados pelo usuário e previamente armazenados em uma tabela. Caso o usuário do equipamento não responda ao agente, os recursos da rede não serão habilitados para este dispositivo.

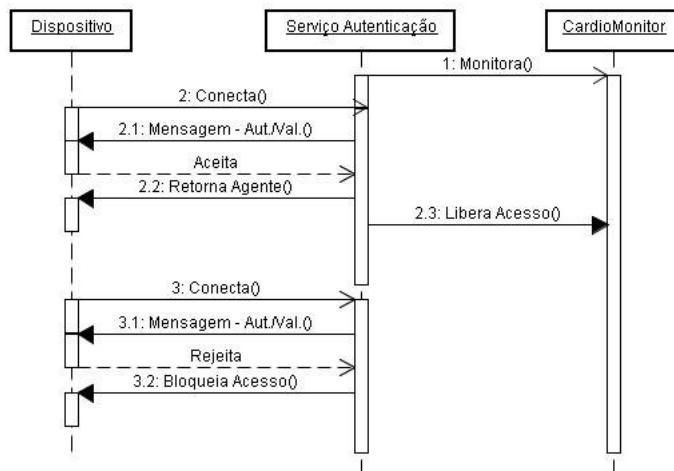


Figura 2. Diagrama de Seqüência do Sistema de Autenticação

O nível de acesso de cada usuário é determinado por um modelo de hierarquia de papéis predefinido, onde a confidencialidade aos recursos é garantida. O perfil mínimo permite ao usuário a utilização apenas de serviços como Internet, vinculados à porta 80. Isto garante a utilização do serviço por qualquer usuário que esteja no ambiente da rede e que possua um equipamento *wireless* habilitado. Desta forma, o controle de acesso à aplicação do CardioMonitor restringe-se aos usuários com nível de acesso adequado à este serviço. Todos os requisitos e regras destinados a este modelo são baseados na política de segurança previamente estabelecida.

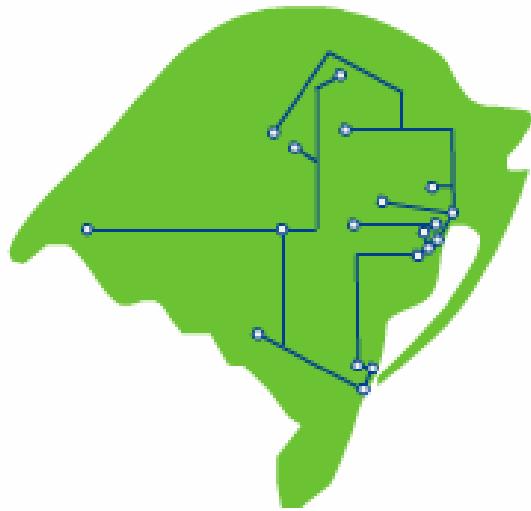
5. Considerações Finais

Fatores importantes são considerados críticos na implantação de sistemas de segurança de informação em uma organização. Itens como política de segurança, enfoque de implementação, comprometimento, gerenciamento e treinamento são de extrema necessidade. O objetivo deste trabalho foi desenvolver uma ferramenta capaz de complementar os requisitos estabelecidos para obtenção de um ambiente seguro.

O fato do modelo proposto estar de acordo com as normas de segurança vigentes, e obedecendo os requisitos definidos pela política de segurança, o caracteriza como uma boa solução para a gestão do problema apresentado.

Referências

- CFM. (2002), “Resolução 1.639/2002 do Conselho Federal de Medicina”. Disponível em <http://www.arnaut.eti.br/ResoCFM.htm>.
- IEEE 802.15. (2002), “*Standard for Information technology” - Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements.*”
- ISO/IEC JTC 1/SC 27. (1998), “*Glossary of IT Security Terminology. Information Technology – security techniques.*”.
- Mazzutti, Cristiano; Nunes, Raul Ceretta. (2003), “CardioMonitor: Ferramenta para Visualização de Curvas de Batimentos Cardíacos.” CRICTE, Univali, Itajaí.
- NBR/ISO/IEC 17799. (2002), “Tecnologia da informação: Código de prática para a gestão da segurança da informação”. Associação Brasileira de Normas Técnicas ABNT, 55pp.
- Nunes, Raul Ceretta; et all. (2002), “Sistema Integrado para Aquisição, Conversão e Visualização Remota de Sinais Cardíacos Utilizando PDA’s”. Universidade Federal de Santa Maria, Santa Maria.
- Pfleeger, Charles P.; Pfleeger, Shari L.(2003), “*Security in Computing.*” Prentice Hall: 3 ed. New Jersey.
- SUN Microsystems. (2005), “*How to develop a network security policy an overview of internet working site security.*” Disponível em <http://www.sun.com/software/white-papers/wp-security-dedvsecpolicy>.
- Hutten, H.; Schreier, G.; Kastner, P.; Schaldach, M. (1997), “Cardiac Telemonitoring by Integrating Pacemaker telemetry within Worldwide Data Communication Systems.” In Proc. of the IEEE International Conference EMBS, Chicago, Oct.30-Nov.2.
- Braecklein, M.; et all. (2004), “*New System Cardiological Home Monitoring With Integrated Alarm Function.*” OpenECG Worshop, Berlin.



SESSÃO TÉCNICA 3

Integração de Sistemas Embutidos utilizando *Web Services*

Guilherme Bertoni Machado , Frank Siqueira

Universidade Federal de Santa Catarina - Departamento de Informática e Estatística
Campus Universitário - Trindade - 88040-900 - Florianópolis - SC - Brasil

{bertoni, frank}@inf.ufsc.br

Resumo. *Sistemas Embutidos estão cada vez mais integrados à Internet através da interconexão destes dispositivos em redes TCP/IP. Web Services tem se mostrado uma arquitetura eficiente para a interconexão de sistemas através da rede; por outro lado, a integração de aplicações provenientes dos Sistemas Embutidos a outros sistemas vem se mostrando cada vez mais necessária. Este trabalho busca demonstrar a viabilidade de integração de Sistemas Embutidos a outros sistemas utilizando Web Services, através do estudo, da modelagem e da disponibilização de um web service projetado com o toolkit de desenvolvimento gSOAP tendo como ambiente o sistema embutido SHIP e da sua integração com outros sistemas.*

1. Introdução

A *World Wide Web* teve como proposta inicial somente a troca de documentos entre os computadores, utilizando como suporte para comunicação em ambiente distribuído a rede Internet. Porém, com o crescimento e popularização desta, surgem várias aplicações que, ao utilizar a arquitetura de comunicação da *Web* - que é baseada em TCP/IP - necessitam de um método eficiente para intercâmbio de dados (informações). A Arquitetura *Web Services*¹ (WS) busca a solução deste problema, pois ao utilizar padrões abertos de protocolos e linguagens, possibilita a integração das mais diversas aplicações distribuídas sem se preocupar com a heterogeneidade intrínseca dos ambientes distribuídos [1].

WS apresentam-se como uma evolução das tecnologias de comunicação baseadas em Objetos Distribuídos [2], pois ao invés de se referenciar a uma interface de um objeto, um *web service* busca uma mudança deste paradigma para uma Arquitetura Orientada a Serviços (SOA) [3].

Sabendo de toda a potencialidade que WS podem oferecer e analisando a tendência de uma maior integração entre Software e Hardware, nada mais natural em se propor a construção/implementação de *web services* também em sistemas embutidos, uma vez que estes representam a maior fatia do mercado de processadores e que suas aplicações estão cada vez mais sofisticadas, principalmente pelo fato de que estes estão cada vez mais integrados à Internet através da sua interconexão em redes TCP/IP.

O presente trabalho foi realizado com o apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq - Brasil e do projeto FunPesquisa-UFSC 2004.

¹Na literatura pesquisada também foram encontrados os termos *WebServices* e Serviços Web (termo em português). Optou-se por manter a grafia proposta pelo W3C, e *web service* como uma instância (aplicação) de um *Web Service*.

Como consequência deste processo de integração torna-se possível para as aplicações provenientes dos Sistemas Embutidos interagirem com aplicações disponíveis em outros sistemas. De modo a comprovar essa possibilidade, este trabalho se propõe a realizar a modelagem e a disponibilização de um *web service* num sistema embutido, sendo que a aplicabilidade dessa proposta é apresentada em um cenário (caso de uso) na área médica.

2. Principais Tecnologias Empregadas por *Web Services*

Sabendo que WS não é uma tecnologia específica, e sim um conjunto de protocolos de comunicação e interoperabilidade (consolidados e/ou emergentes) [4], revisamos as principais tecnologias empregadas pelos WS.

A linguagem **XML - Extensible Markup Language** [5] consiste em um padrão para especificar e processar as informações (dados). XML é empregada como base de comunicação dos WS [1]. Baseada em TAGs (delimitadores), a linguagem descreve os objetos, seus atributos, métodos e parâmetros de forma que os dados sejam interpretados pelas aplicações.

SOAP - Simple Object Access Protocol [5] é um protocolo leve para a troca de dados XML pela Web [6]. Serve como um envelope de um documento XML para que este possa ser transmitido pela Web. Seu envio pela rede pode ser feito usando diversas tecnologias, incluindo SMTP, HTTP e FTP dentre outros.

WSDL - Web Service Description Language [5] é uma linguagem, baseada em XML, utilizada para descrever WS. Esta descrição inclui detalhes como: definição dos tipos dos dados, operações suportadas pelo serviço, formatos das mensagens de entrada/saída, endereço de rede, mapeamento de protocolos, etc [6].

UDDI - Universal Description, Discovery, and Integration - consiste em um serviço de nomeação e localização de *web services* estruturado na forma de repositórios [7]. Após um *web service* ser colocado em operação, este pode ser publicado em um repositório UDDI. A partir desse momento, as informações necessárias para a localização e a utilização do serviço disponibilizado por este *web service* se tornam acessíveis para os clientes na forma de um arquivo WSDL.

3. Sistemas Embutidos e *Toolkits* para Desenvolvimento

Podemos definir um Sistema Embutido como um dispositivo microprocessado, portanto programável, que tem como proposta utilizar o seu poder computacional para uma finalidade específica. Ao contrário das plataformas "genéricas" como o PC, normalmente um sistema embutido é desenvolvido para uma aplicação exclusiva (como por exemplo, decodificadores de TV a cabo, controladores de fornos de microondas, aparelhos de DVD, chips de telefones celulares, dentre outros), logo, possui características bem peculiares tais como [8]: algoritmos complexos, interfaces com o usuário específicas, suporte a tarefas com requisitos temporais, suporte a múltiplas tarefas sendo executadas com prioridades diferentes, custo de fabricação reduzido, monitoração e controle do consumo de energia.

No mercado existem inúmeras soluções existentes que se encaixam, isto é, possuem as funcionalidades (pilha TCP/IP nativa e suporte a HTTP) necessárias para a in-

tegração dos dispositivos ligados a estes sistemas embutidos a ambientes distribuídos via WS.

Quanto ao desenvolvimento de *web services* para sistemas embutidos existe uma quantidade razoável de *toolkits*, sejam eles específicos para esta abordagem, ou então que possuem suporte para este tipo de desenvolvimento. Como principais restrições para a escolha de um toolkit apontamos as seguintes características: que funcione em hardware de porte limitado (processamento, memória, sistema operacional, etc), ambiente de desenvolvimento amigável e boa documentação para o usuário/desenvolvedor.

4. Características da Abordagem Proposta

Um vez que o sistema embutido usado como plataforma de estudo/utilização possui funcionalidades que possibilitam o armazenamento de dados e a troca destes via protocolos Internet, existe a possibilidade da construção de um *web service* neste dispositivo.

A proposta inicial deste trabalho consiste em disponibilizar uma descrição dos serviços disponíveis no *web service* via WSDL e que existam pelo menos dois tipos de serviço, para que numa próxima etapa possamos trabalhar aspectos de QoS e processamento em tempo-real com prioridades distintas. Uma vez definidos os serviços e a descrição destes, a elaboração da aplicação (cliente) pode ser feita de acordo com as necessidades e recursos do usuário, podendo ser uma página, outro sistema embutido, um programa, etc.

Basicamente o sistema proposto segue a estrutura representada na figura 1, na qual um cliente envia suas mensagens XML envelopadas com SOAP de acordo com o tipo de serviço requerido (previamente conhecido através do arquivo WSDL localizado no próprio Sistema Embutido e/ou em um repositório UDDI) através do protocolo HTTP.

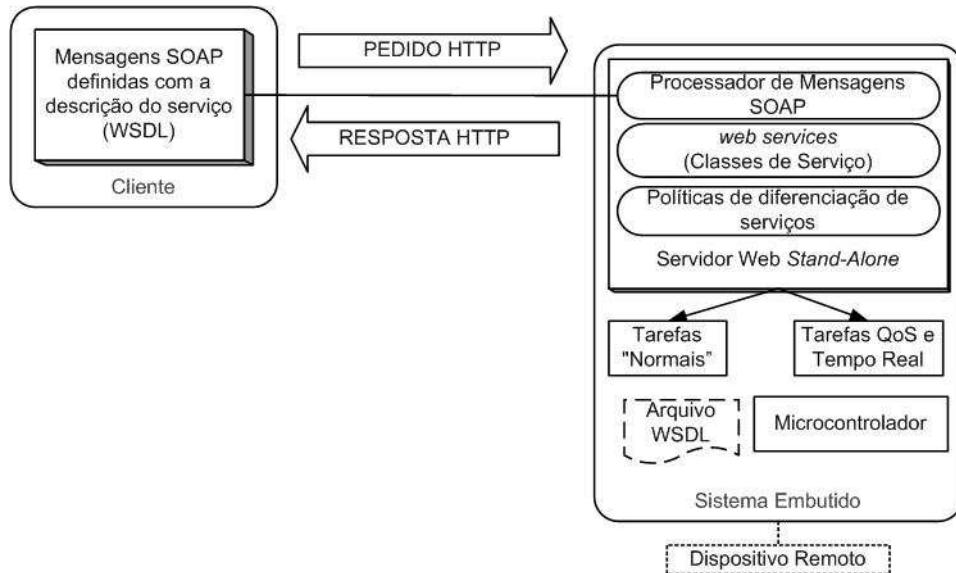


Figura 1. Sistema proposto.

No *web service* há um processamento (categorização, seguindo a metodologia proposta por Serra et al. [9], que consiste na distribuição eficiente dos serviços, ou seja, prioridades de escalonamento, baseado na diferenciação - *gold*, *silver*, *bronze* e *best effort* - destes) dessas mensagens que serão então processadas pelo microcontrolador. Feita a

execução destas mensagens, que pode ser uma resposta do próprio microcontrolador ou do dispositivo remoto ligado a ele, estas são retornadas ao cliente.

5. Cenário de Utilização da Proposta

Um exemplo de uso da nossa proposta é a utilização de sistemas embutidos no monitoramento e controle de pacientes, de acordo com a figura 2. Uma vez que os sinais vitais destes estejam sendo constantemente monitorados/controlados por equipamentos, que na nossa abordagem seriam os dispositivos remotos ligados às plataformas, estas fariam a aquisição dos sinais vitais (seguindo parâmetros pré-programados, isto é, aqueles que possuírem requisitos de tempo-real têm maior prioridade), e disponibilizariam via XML/SOAP as informações para o servidor de aplicação.

O papel do servidor de aplicação seria o de distribuir as diversas informações provenientes de n configurações como essa dentro do ambiente hospitalar para os respectivos clientes (a base de dados do hospital e/ou médicos em casa ou no próprio hospital, utilizando PCs, Palms, etc).

No nosso exemplo podemos ver claramente a diferenciação dos serviços: *gold* para o controle em tempo-real, *silver* para os alarmes enviados à equipe de plantão, *bronze* para a atualização da base de dados e *best effort* para os demais serviços.

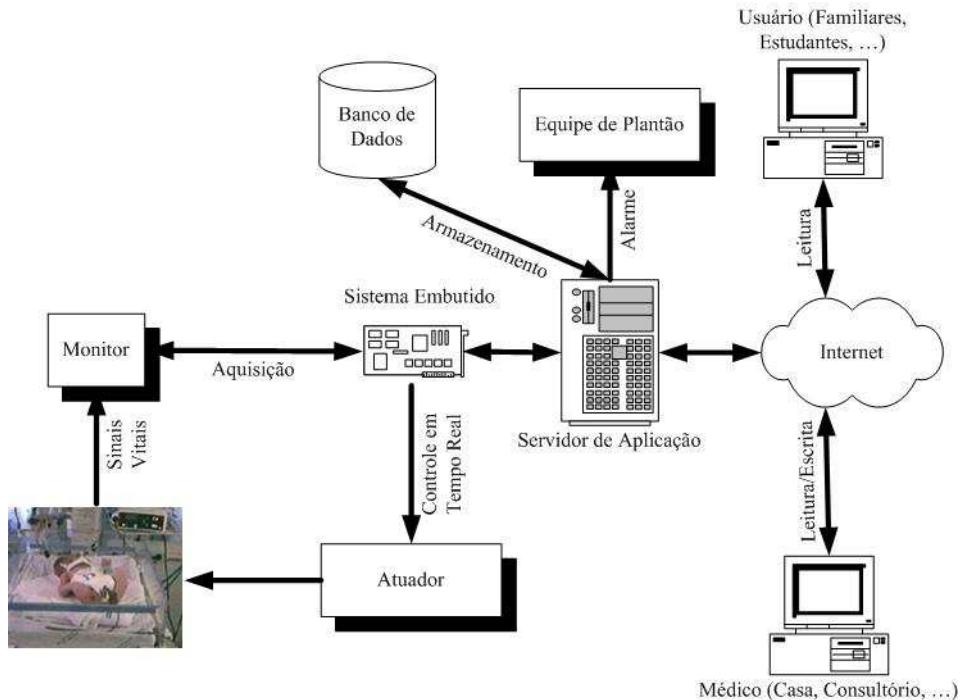


Figura 2. Cenário de Utilização

O grande mérito dessa proposta é que ao utilizar WS não precisamos nos importar com o resto do ambiente distribuído, isto é, quais são os sistemas legados utilizados no servidor de aplicação, como foi feita a modelagem do banco de dados, que tipo de software cliente os usuários estão utilizando, etc.

Todas essas vantagens partem do princípio de que WS utilizam XML sobre uma arquitetura aberta com padrões de *facto*, logo, a utilização dessa estrutura pode ser apli-

cada nos mais diversos cenários, tais como: monitoramento, Controle e Difusão de dados em linhas de transmissão (elétricas, de televisão, etc) e seus diversos sistemas/aplicações envolvidos e ambientes de chão de fábrica com seus diversos sistemas embutidos com interfaces de comunicação específicas.

6. Implementação

O primeiro passo para a verificação das idéias descritas no trabalho consistiu na seleção da plataforma alvo para implantação de WS em sistemas embutidos. Através de um convênio firmado com a empresa Boreste, obtivemos a plataforma SHIP (versão de desenvolvimento) [10], que possui as seguintes características:

- Microcontrolador ARM7TDMI, família AT91X40 com processador RISC;
- Memória flash de 512K Bytes, sendo 448K Bytes livres para as aplicações;
- Sistema Operacional desenvolvido pela Boreste estruturado para dar suporte a conexões *Ethernet* em comunicações TCP/IP. Tem como principais características o μ Boot para carga do sistema e um programa monitor - μ Monitor - para configuração e carga das aplicações na plataforma;
- Diversos dispositivos externos podem ser ligados à plataforma desde equipamentos de automação da manufatura, passando por sensores e atuadores, equipamentos médicos, etc.

Em seguida, foram detectadas as alterações necessárias na plataforma para possibilitar a disponibilização de WS. Foi verificada a necessidade de desenvolver aplicações com suporte a chamadas HTTP, ou seja o próprio *web service* é capaz de prover conteúdo *Web*, além de efetuar os seus serviços.

Em um segundo momento, mostrou-se necessário portar um *toolkit* de desenvolvimento para a plataforma alvo. Optamos pelo *toolkit* gSOAP [11], que se mostrou o mais compatível em relação ao suporte de *hardware/software* da plataforma. Este *toolkit* possui as seguintes características:

- Faz a composição (*binding*) de mensagens SOAP com C/C++ criando *Stubs* e *Skeletons*;
- Cria clientes C/C++ a partir da descrição WSDL;
- É independente de plataforma (possui exemplos de aplicações desenvolvidas em Windows, Linux, Unix, Mac OS X, Pocket PC, Palm OS, Symbian e *embedded Linux*);
- Aplicações podem ser criadas com menos de 100K Bytes, totalizando um consumo total de memória de 150K Bytes.

O desenvolvimento das aplicações para a plataforma SHIP é realizado no ambiente Cygwin [12] configurado com o *cross compiler* ARM GNU *toolchain* [13], com o kit de desenvolvimento (bibliotecas e exemplos) da plataforma e com o *toolkit* gSOAP (versão independente de plataforma).

No presente estágio do trabalho foram construídas aplicações cliente-servidor utilizando a plataforma para executar o serviço e PCs como clientes, demonstrando assim a possibilidade da execução de *web services* em sistemas embutidos. Em seguida estaremos nos dedicando ao desenvolvimento de cenários mais complexos, envolvendo um número maior de clientes e servidores e realizando a integração destes com servidores de bancos de dados e de aplicações, como os descritos na seção 5.

7. Conclusão

WS apresentam-se como uma forma de interconexão de aplicações, através da Internet, entre sistemas computacionais. Além disso, por possuírem uma arquitetura eminentemente aberta e padronizada os WS têm um grande potencial de uso para a computação distribuída.

Portanto, nada mais natural em se propor a construção de um *web service* em um sistema embutido podendo assim promover a integração deste em um ambiente distribuído. A partir desse estudo, esperamos possibilitar uma nova área de aplicação dos WS como meio de integração de Sistemas Embutidos a outros sistemas. Além disso, este trabalho se propõe a adicionar suporte a QoS em WS, visto que requisitos de QoS e sua política de utilização ainda não estão bem consolidados nesta tecnologia.

Referências

- [1] BONIATI, Bruno B.; PADOIN, Edson Luiz. Web services como middlewares para interoperabilidade em sistemas. *Revista do CCEI - Centro de Ciências da Economia e Informática*, v. 7, n. 12, p. 17 – 24, Agosto 2003.
- [2] VOGELS, Werner. Web services are not distributed objects. *IEEE Internet Computing*, v. 7, n. 6, p. 59 – 66, Nov. - Dec. 2003.
- [3] CHAPPELL, David; JEWELL, Tyler. *Java Web Services*. First. [S.l.]: O'Reilly, 2002.
- [4] KILGORE, Richard A. Simulation web services with .net technologies. *Proceedings of the Winter Simulation Conference*, v. 1, p. 841 – 846, 8 - 11 Dec. 2002.
- [5] W3C. *World Wide Web Consortium*. 2005. Disponível em: <<http://www.w3.org/>>. Acesso em: 29/04/2005.
- [6] ROY, Jaideep; RAMANUJAN, Anupama. Understanding web services. *IEEE Internet Computing*, v. 3, n. 6, p. 69 – 73, Nov. - Dec. 2001.
- [7] UDDI. *Universal Description, Discovery and Integration*. 2004. Disponível em: <<http://www.uddi.org/>>. Acesso em: Acessado em: 13/08/2004.
- [8] WOLF, Wayne. *Computer as Components: principles of embedded computing system design*. [S.l.]: Morgan Kaufmann, 2001.
- [9] SERRA, Antônio; GAÏTI, Dominique; BARROSO, Giovanni; RAMOS, Ronaldo; BOUDY, Jérôme. Uma plataforma distribuída com balanceamento de cargas para servidores web baseada na diferenciação de serviços. *XXXI SEMISH - Seminário Integrado de Software e Hardware*, Agosto 2004.
- [10] BORESTE. *SHIP - Software e Hardware Integrados em um Plataforma*. 2005. Disponível em: <<http://www.boreste.com>>. Acesso em: 28/02/2005.
- [11] GENIVIA INC. *gSOAP - C/C++ Web Services and Clients*. 2005. Disponível em: <<http://www.genivia.com>>. Acesso em: 28/02/2005.
- [12] CYGWIN. *Cygwin Information and Installation*. 2005. Disponível em: <<http://www.cygwin.com>>. Acesso em: 22/03/2005.
- [13] MACRAIGOR SYSTEMS. *Macraigor Systems GNU Tools*. 2005. Disponível em: <http://www.macraigor.com/full_gnu.htm>. Acesso em: 22/03/2005.

TCP Reno, TCP Vegas e TCP Westwood: Uma Comparação de Desempenho

Juliana de Santi¹, Michele Mara de Araújo Espíndula Lima¹

¹Colegiado de Informática – Universidade Estadual do Oeste do Paraná (UNIOESTE)
Cascavel – PR – Brazil

{jsanti,michele}@unioeste.br

Resumo. Apesar da sua relativa eficiência, o mecanismo de controle de congestionamento do TCP Reno, implementação padrão do TCP, apresenta uma série de problemas. Entre elas, tem-se que a estratégia utilizada para governar o comportamento da sua janela de transmissão é inherentemente oscilatória, o que faz com que o uso da banda passante disponível oscile significantemente. Tal problema torna-se ainda mais grave em redes com grande produto banda-atraso. Para melhorar a capacidade do TCP em estimar a banda disponível, aprimorar o seu desempenho em redes com grande produto banda-atraso TCP e garantir a estabilidade da janela, e consequentemente da taxa de transmissão outros mecanismos TCP tem sido propostos, dentre estes o TCP Vegas e TCP Westwood. Neste trabalho é feita uma comparação de desempenho entre TCP Reno, TCP Vegas e TCP Westwood baseada em simulações no NS.

1. Introdução

A implementação padrão do TCP, denominada TCP Reno, possui um mecanismo de controle de congestionamento composto de quatro algoritmos: *Slow Start*, *Congestion Avoidance*, *Fast Recovery* e *Fast Retransmit* [Allman et al. 1999]. Apesar de serem quatro algoritmos distintos, na prática os algoritmos são implementados como se fossem apenas dois: *Slow Start* e *Congestion Avoidance*, e *Fast Recovery* e *Fast Retransmit*. Os dois primeiros algoritmos são utilizados pelo TCP emissor para controlar a quantidade de dados que está sendo injetada na rede, evitando assim o congestionamento, ou seja, evitando injetar na rede uma carga maior do que esta pode absorver. Os dois últimos são utilizados pelo TCP para se recuperar de perdas de pacotes.

Se três ou mais reconhecimentos forem recebidos para o mesmo segmento, o segmento em questão é considerado perdido e o tamanho da janela de transmissão é reduzido à metade. Entretanto, quando uma perda é detectada pela expiração do intervalo de temporização (*timeout*), a janela de transmissão é reduzida drasticamente para um segmento e o TCP emissor volta para a fase de *Slow Start*.

Apesar da sua relativa eficiência, o mecanismo de controle de congestionamento do TCP apresenta uma série de problemas. Entre eles pode-se destacar que fluxos TCP, com grande RTT são prejudicados em detrimento dos que possuem RTT pequeno, não conseguindo a sua porção justa da banda passante. Além disto, o TCP Reno utiliza a estratégia AIMD (*additive-increase-multiplicative-decrease*), para governar o comportamento da sua janela de transmissão. Tal estratégia é inherentemente oscilatória, o que faz com que o uso da banda passante disponível também oscile significantemente. Desta

forma, tem-se períodos em que recursos são subutilizados e outros em que ocorre o congestionamento e suas consequências. Finalmente, o TCP tem que criar perdas para detectar o estado de congestionamento da rede. Os dois últimos problemas tornam-se ainda mais graves em redes com grande produto banda-atraso¹, ou em ambientes heterogêneos onde perdas não são decorrente apenas de congestionamento.

Para melhorar a capacidade do TCP em estimar a banda disponível, aprimorar o seu desempenho em redes com grande produto banda-atraso TCP e garantir a estabilidade da janela, e consequentemente da taxa de transmissão outros mecanismos TCP tem sido propostos, dentre estes o TCP Vegas e TCP Westwood.

No TCP Vegas a descoberta da banda disponível é dissociada de perdas. Ele faz isso através do monitoramento da diferença entre a uma estimativa de banda esperada e a vazão de dados que de fato foi transmitida atualmente. O TCP Westwood surge como um mecanismo para ambientes heterogêneos e de alta velocidade. Em [Santi 2004] foram investigadas as propostas de mecanismos TCP Vegas e TCP Westwood. Neste trabalho é feita uma avaliação da eficácia destas duas propostas e o TCP Reno, baseada em simulações no NS (*Network Simulator*) [VINTproject 2005] em cenários onde eles concorrem pelo recurso disponível.

Este artigo é organizado da seguinte forma: a Seção 2 apresenta o TCP Vegas. Na Seção 3 é apresentado o TCP Westwood. Na Seção 4, os resultados numéricos dos experimentos realizados são apresentados. Finalmente, na Seção 5, as conclusões são delineadas.

2. TCP Vegas

O TCP Vegas é um protocolo TCP modificado, introduzido em [Brakmo et al. 1994, Hengartner et al. 2000] como uma alternativa ao TCP Reno, para melhorar a utilização da banda disponível e diminuir o atraso.

Para melhorar a vazão e diminuir as perdas de pacotes, o TCP Vegas utiliza três mecanismos: mecanismo de retransmissão, mecanismo de prevenção de congestionamento e um mecanismo que modifica o *Slow-Start*.

O mecanismo de prevenção de congestionamento é responsável por ajustar as taxas de transmissão de maneira adequada de acordo com o estado de congestionamento da rede, e assim, corrigir o comportamento oscilatório do TCP Reno. Para calcular a banda disponível sem ter que recorrer à informações sobre perdas, o TCP Vegas estima a vazão esperada, V_e , e então faz a diferença ($diff$), entre este valor e a vazão obtida, V_o , e com base nesta diferença a taxa de envio é ajustada. A vazão esperada é dada por: $V_e = \frac{cwnd}{baseRTT}$, onde $baseRTT$ é o RTT mínimo medido, e $cwnd$ é a janela de congestionamento. Se $diff < \alpha$, a rede não está congestionada e $cwnd$ é incrementada linearmente durante o próximo RTT. Se $diff > \beta$, a vazão obtida é menor do que a esperada e $cwnd$ é aditivamente decrementada. Caso $\alpha < diff < \beta$, a janela permanece inalterada, dado que a janela $cwnd$, encontra-se no ponto de equilíbrio. Os valores α e β correspondem respectivamente a menor e a maior quantidade de dados extra na rede, e são determinados empiricamente.

¹O Produto banda X atraso é o resultado da banda do enlace pelo RTT, e serve para indicar o volume de dados que pode estar em transito em um determinado momento.

No TCP Vegas o crescimento de $cwnd$ na fase de *Slow-Start* é também condicionado ao valor obtido para df , visando, assim, detectar/evitar o congestionamento ainda na fase inicial quando se está tentando encontrar a banda disponível.

Ao contrário do TCP Reno que reduz a janela de congestionamento pelo menos a metade quando uma perda é detectada, no TCP Vegas a redução é de somente $\frac{1}{4}$. Isto contribui para a recuperação mais rápida e consequentemente para o aumento da vazão. Além disto, os ACKs são utilizados como base de informação para tomar a melhor decisão sobre o momento da retransmissão.

O TCP Vegas é eficiente na estabilização da rede, porém, por ser um mecanismo conservativo, ele perde espaço no enlace por interpretar a agressividade de outros mecanismos como sinal de congestionamento e decrementar sua capacidade de envio de dados.

3. TCP Westwood

O objetivo do TCP Westwood (TCPW) é utilizar de maneira mais eficiente o enlace, particularmente quando as perdas não são decorrentes somente de congestionamento [Gerla et al. 2004]. Assim como no TCP Reno, a janela de congestionamento, $cwnd$, é incrementada de forma exponencial no *Slow-Start* e linear no *Congestion Avoidance*. A diferença está na sua forma de decremento.

No TCPW é feita a estimativa da taxa (RE) de dados transmitida pela conexão num determinado período de tempo. Este valor é obtido a partir da quantidade de dados entregue com sucesso no último RTT, estimada em função dos ACKs recebidos. A RE é normalizada por um filtro passa-baixa, e então é utilizada para ajustar os valores de $cwnd$ e $ssthresh$ após a ocorrência de perdas de pacotes. A chegada de ACKs duplicados pode indicar que a capacidade do enlace foi atingida ou que estão ocorrendo perdas aleatórias. Daí os ajustes serem feitos em função de RE, permitindo que as filas sejam esvaziadas após o congestionamento, e o $ssthresh$ receba a banda disponível no momento do congestionamento. Com isso diminui-se a taxa de transmissão, sem, no entanto, deixar o enlace subutilizado.

O valor do limiar $ssthresh$ é dado por $ssthresh = (RE * RTTmim)/segSize$. Assim como no TCP Reno, após a ocorrência de *timeout*, $cwnd$ é ajustada para o valor inicial. Porém, quando a perda é detectada pelo recebimento de ACKs duplicados $cwnd$ é feito igual a $ssthresh$, que por sua vez é ajustada de acordo com RE, permitindo a recuperação mais rápida e prevenindo a subutilização dos recursos disponíveis.

O valor da janela $cwnd$ é proporcional ao produto banda-atraso, logo, em ambientes de produto elevado, $cwnd$ deve ser grande para ocupar todo o enlace disponível. Desta forma, uma divisão de $cwnd$ ao meio, deixa uma grande quantidade de recursos subutilizados por um longo período de tempo. Portanto, o ajuste adaptativo do TCPW traz benefícios significativos para estes ambientes.

Quando compartilhando o canal, o TCPW tenta utilizar os recursos recursos ociosos, ou seja, utiliza os recursos que não estão sendo utilizados pelas conexões das demais implementações TCPs, quando estas estão de recuperando de perdas ou na sua fase inicial. Isto, no entanto, não é entendido como falta de equidade², uma vez que o TCPW somente está utilizando recursos que de outra forma ficariam subutilizados.

²Eqüidade = recursos disponíveis são distribuídos igualmente entre as conexões que ocupam o enlace.

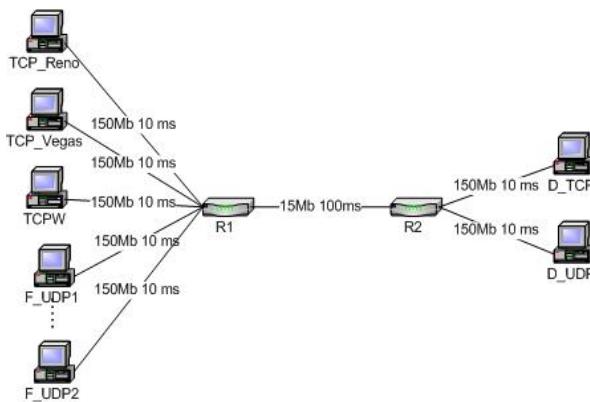


Figura 1. Cenário da simulação

4. Resultados Numéricos

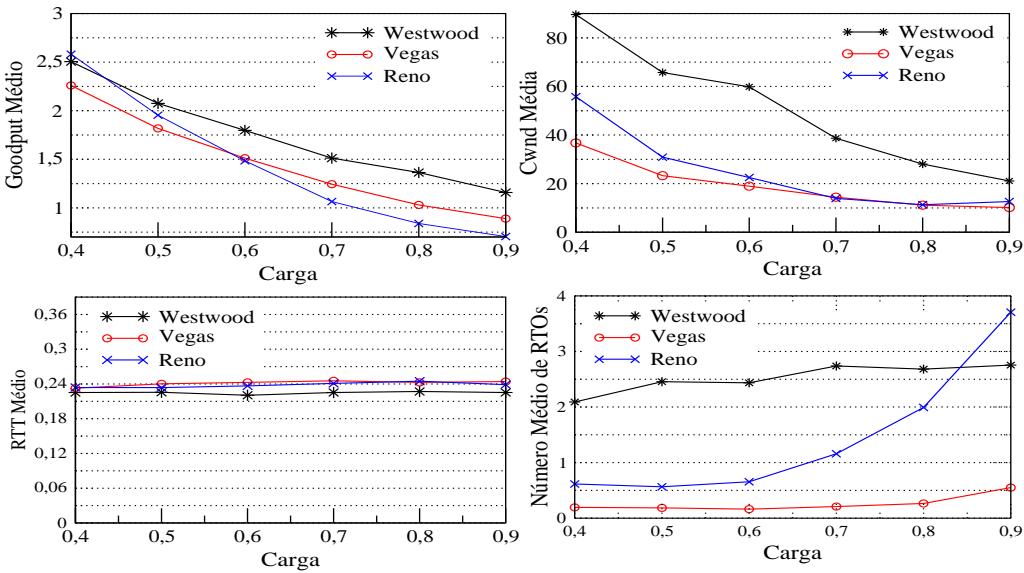
Para avaliar o comportamento de TCP Reno, TCP Vegas e TCP Westwood foram realizadas simulações no NS. A topologia, a capacidade dos enlaces, como também o tempo de propagação para cada enlace são apresentados na Figura 1. O tamanho do *buffer* utilizado pelos dois nós ligados ao enlace gargalo é de 800 segmentos e a política de AQM utilizada foi RED.

Um gerador de tráfego, denominado TrafficGen, foi utilizado para gerar cargas de tráfegos específicos. A carga foi variada de 0,4 a 0,9, de forma a permitir a análise do desempenho sob diferentes cargas. Uma distribuição híbrida Lognormal/Pareto foi utilizada para gerar o tráfego Web. O tráfego FTP foi gerado utilizando uma distribuição exponencial. Tanto o tráfego Web como o tráfego FTP foram gerados a partir dos nós TCP_{Reno} , TCP_{Vegas} , $TCPW$ com destino para o nó D_{TCP} . As três implementações TCP executam simultaneamente, cada uma sendo responsável por gerar $\frac{1}{3}$ da carga total, dado que a Internet é um ambiente em que várias versões de TCP coexistem. Com o intuito de gerar um cenário mais realista, foram incluídos fluxos não-adaptativos do tipo CBR/UDP que podem representar até 20% da capacidade do enlace. Estes fluxos são gerados e finalizados em diferentes intervalos a partir dos nós F_{udp_i} com destino ao nó D_{UDP} .

De modo a garantir que o crescimento da janela $cwnd$, fosse governado apenas pela rede e não pelo TCP receptor, ou seja, o controle de fluxo foi desativado através da utilização de um valor alto para a janela de recepção do receptor (1000). O tamanho dos segmentos gerados foi de 500 bytes.

Nas Figuras 2 e 3 são apresentados os valores médios obtidos por conexão para o *goodput*, o tamanho da janela, o RTT e o número de RTOs para cada uma das implementações TCP quando o tráfego principal é o tráfego FTP e WEB respectivamente. As comparações descritas a seguir são feitas sempre em relação ao TCP Reno.

Pode-se ver na Figura 2 que para o TCP Westwood o valor de $cwnd$ é bastante agressivo, sendo no mínimo 61% maior para cargas de 0,4 e no máximo 178% maior para cargas de 0,7. A agressividade da janela se reflete também no aumento de RTOs, para cargas abaixo de 0,9 sendo este aumento no mínimo 35% maior para carga de 0,8 e no máximo 335% maior com carga de 0,5. Apesar do considerável número de RTOs o seu

**Figura 2. TCPs executando simultaneamente com tráfego FTP**

goodput é maior, exceto apenas para cargas de 0,4. O aumento máximo no *goodput* é de 64% para cargas de 0,9 e mínimo de 6% para cargas de 0,5.

Para o TCP Vegas o valor de *cwnd* obtido é menor para cargas abaixo de 0,7 e é basicamente equivalente para as demais cargas. O número de RTOs é sempre menor, e a diferença aumenta a medida que o congestionamento se torna mais pesado; a redução no número de RTOs é de no mínimo 67% para cargas de 0,5 e de no máximo 87% para cargas de 0,8. O *goodput* é menor na fase inicial, sendo 12% e 7% menor a 0,4 e 0,5 de carga, mas torna-se maior para as demais cargas, sendo no mínimo 2% maior para cargas de 0,6 e no máximo 26% para cargas de 0,9.

Os valores de RTT são muito próximos, uma vez que a política de AQM utilizada foi a mesma para os três TCPs.

Pode-se ver na Figura 3, que para TCPW o valor de *cwnd* é no mínimo 7% maior e no máximo 58% maior para cargas de 0,4 e 0,8 respectivamente. O número de RTOs é 9% menor para cargas 0,9, e maior nos demais casos; o valor máximo apresentado é 116% maior para cargas de 0,4 e o valor mínimo é de 39% maior para cargas de 0,8. A diferença entre o *goodput* do TCPW e do TCP Reno não é tão grande como para tráfego FTP, mas ainda continua sendo maior na maioria dos casos.

Na presença de tráfego WEB, o desempenho do TCP Vegas quando concorre com as demais implementações TCP, é basicamente inferior ao desempenho do TCP Reno para todas as métricas estudadas. Isto acontece pelo fato do tráfego WEB ser de curta duração e pelo fato do TCP Vegas ser bastante conservativo. O valor médio de *cwnd* do TCP Vegas é menor que os demais TCPs, e se mantém basicamente constante para todas as cargas. O número de RTOs fica entre 116% maior a 0,4 de carga e 20% maior a 0,6 de carga; para as demais cargas o número de RTOs é menor, sendo no mínimo 2% menor e no máximo 32% menor com cargas de 0,4 e 0,9 respectivamente. O seu *goodput* acompanha a janela de congestionamento, sendo no mínimo 16% menor para cargas de 0,9 e no máximo 31% menor para cargas de 0,4.

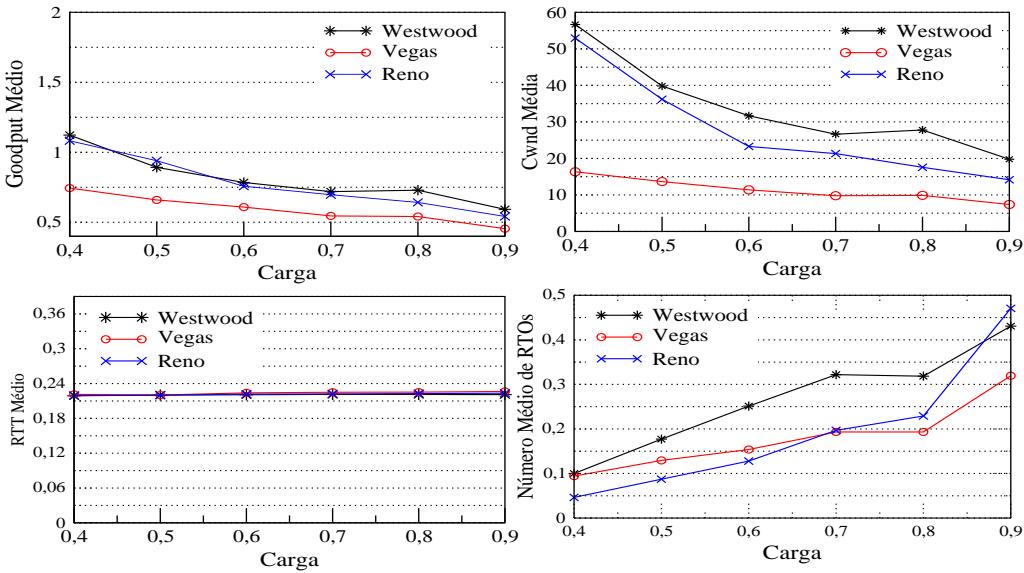


Figura 3. TCPs executando simultaneamente com tráfego WEB

5. Conclusões

Para os experimentos realizados o TCP Westwood apresenta o melhor *goodput* tanto para tráfego FTP quanto WEB. Mesmo com uma elevada ocorrência de *timeouts*, consegue o melhor *goodput*, o que pode ser atribuído ao seu elevado tamanho de janela de congestionamento. Isto ocorre porque o TCPW tenta utilizar os recursos ociosos, ou seja, utiliza os recursos que não estão sendo utilizados pelas conexões das demais implementações TCPs. Com relação ao TCP Vegas verifica-se que apresenta o pior *goodput* entre os TCPs comparados, mostrando-se assim um mecanismo conservativo e que por isso acaba sendo penalizado quando compete com outras versões TCPs. Conclui-se então, o TCPW é o mecanismo TCP mais eficiente entre os mecanismos avaliados para ser utilizado em ambientes heterogêneos, onde várias versões TCP coexistem.

Referências

- Allman, M., Paxson, V., and Stevens, W. (1999). TCP congestion control. RFC 2581.
- Brakmo, L. S., O'Malley, S. W., and Peterson, L. L. (1994). TCP vegas: New techniques for congestion detection and avoidance. In *SIGCOMM*, pages 24–35.
- Gerla, M., Ng, B. K. F., Sanadidi, M. Y., Valla, M., and Wang, R. (2004). TCP westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs. *Journal of Computer Communications*, 27(1).
- Hengartner, U., Bolliger, J., and Gross, T. (2000). TCP vegas revisited. In *Proceedings of IEEE INFOCOM*, pages 1546–1555, Tel Aviv, Israel.
- Santi, J. (2004). Desenvolvimento de uma política de AQM utilizando teoria de controle e Ótimo e mecanismos de controle de congestionamento TCP estáveis. Trabalho de Conclusão de Curso, UNIOESTE, 2004.
- VINTproject (2005). NS: Network simulator. <http://www.isi.edu/nsnam/ns>.

Análise do desempenho dos protocolos UDP e TCP entre os Kernels 2.4.27 e 2.6.9 do Linux em Redes Gigabit Ethernet

Carlos Brites¹, Gaspare Bruno¹

¹ Unilasalle – Centro Universitário La Salle - Ciência da Computação – Av. Victor Barreto - Canoas – RS – Brasil

carlos_brites@dell.com, gaspare@inf.ufrgs.br

Resumo. Tendo em vista que o desempenho das redes de computadores é um dos assuntos mais relevantes na atualidade, este artigo tem por objetivo apresentar os resultados de um estudo experimental de desempenho de redes Gigabit Ethernet no kernel 2.4.27 e o kernel 2.6.9 do sistema operacional Linux. Neste ambiente, utilizou-se os protocolos UDP e o TCP, porque serem os mais utilizados. Os resultados obtidos serão utilizados para futuras otimizações para aplicações específicas, como VOIP.

1. Introdução

Analizar ou medir o desempenho de qualquer que seja a tecnologia ou processo, é um fator importante para que se possa entender o seu real comportamento funcional, bem como planejar melhorias em função dos dados analisados para referência comparativa entre as mesmas ou diferentes tecnologias e processos. Através do experimento e análise pode se observar que o comportamento da largura da banda média é um fator importante para se determinar o tamanho de pacote. Estas informações são essenciais para o desenvolvimento de aplicações ou para o uso de tecnologia de rede. As redes de computadores acompanham a evolução tecnológica mundial, devendo ser robustas e suprirem as necessidades impostas pelas aplicações. Para tal, a rede deve fornecer uma conectividade de alto desempenho (Peterson e Daves, 2004).

O experimento teve como objetivo a análise e comparação da utilização da CPU com relação ao desempenho das redes Gigabit Ethernet nas versões dos kernel 2.4.27 e o kernel 2.6.9 do sistema operacional Linux, para cada conjunto de equipamentos (Hagen 2004).

O desempenho das redes mostra-se mais dependente de fatores como os sistemas operacionais, implementações de protocolos, drivers de rede, característica de hardware, capacidade de processamento e adaptadores de rede, do que os meios de transmissão física de dados propriamente dito. O projeto e implementações das pilhas de protocolo são um dos principais obstáculos que devem ser transpostos para que se tenha uma rede de computadores mais veloz e eficiente (Tanenbaum, 2003).

Baseado nas tecnologias existentes atualmente, este trabalho tem como objetivo identificar, através de experimentos práticos, parâmetros que interfiram nas funcionalidades de desempenho da rede de computadores em redes Gigabit Ethernet. O foco principal deste experimento foi à análise de desempenho da rede de computadores utilizando distribuições do sistema operacional Linux com kernel 2.4.27 e o kernel 2.6.9, bem com a utilização dos protocolos de transmissão TCP (Protocolo de Controle de Transmissão) e UDP (Protocolo Datagrama de Usuário). Os parâmetros em questão variam de acordo com a aplicação. Os resultados obtidos serão empregados em futuras

otimizações do código do *kernel* Linux para aplicações voltadas a transmissão de voz sobre IP.

Em um ambiente de rede de alto desempenho, como em redes *Gigabit Ethernet*, é também de comum interesse determinar o desempenho de equipamentos isolados, como por exemplo: determinar o nível de serviços da rede. A banda da rede e a utilização da CPU foram às métricas utilizadas para a análise de desempenho nos experimentos realizados em laboratório, de acordo com os dados analisados no capítulo que descreve os resultados.

O mau dimensionamento da rede, sistemas operacionais desatualizados, dimensionamento de memória, CPU, tamanho do *buffer* (banda), tamanho do pacote, aplicação, equipamentos inadequados ou não homologados e tantos outros são alguns dos fatores que podem interferir no desempenho da rede de computadores ou na utilização de recursos da CPU (Tanenbaum, 2003). Com a constante evolução tecnológica, a dificuldade de alinharmos os fatores mencionados, com isto, é possível ocasionar saturação da rede ou uma subutilização dos recursos da CPU, podendo gerando um mau desempenho do processo como um todo (Tanenbaum, 2003).

O artigo se subdivide nas seguintes seções: no capítulo 2, fala-se sobre a metodologia aplicada neste trabalho, no capítulo 3, mostra os resultados gerados através do experimento proposto, e no capítulo 4, é apresentado a conclusão dos dados gerados e os resultados.

2. Metodologia

Para este experimento foram utilizados dois conjuntos de computadores, com dois equipamentos em cada conjunto, com as seguintes configurações: Conjunto A utilizou os computadores Comp.A e Comp.A1 (**Servidor**, processador Intel 2.8GHz P4, memória 512MB, adaptadores de rede modelo A *Gigabit Ethernet*) e o Conjunto B utilizou os computadores Comp.B e o Comp.B1 (**Estação de trabalho**, processador Intel 2.8GHz P4, memória 512 MB, adaptadores de rede modelo B *Gigabit Ethernet*), e *switch Gigabit Ethernet*.

Para realizar a análise de desempenho em laboratório, de acordo com a proposta do artigo, é importante ressaltar que os experimentos ocorreram em um ambiente controlado e favorável, onde foi utilizada uma rede isolada composta por apenas um *switch* a fim de evitar tráfegos espúrios (Roesler, 2001). A ferramenta utilizada para realizar o processo de captura de dados foi o Netperf com a versão de 2.3 (Netperf 2005). O tempo despendido pela ferramenta Netperf para executar cada teste realizado nos laboratórios foi de 20 segundos, ou seja, tempo em que uma estação (cliente) permaneceu enviando uma série de pacotes a outra estação (*server*), para cada tamanho de pacote versus *buffer*.

Foram implementados *scripts* para automação dos testes, e os mesmos estavam configurados para utilizarem os protocolos UDP e TCP. Através desses *scripts*, cruzaram se diferentes tamanhos de buffers (8192, 16384, 32768 e 65535 bytes) com os diferentes tamanhos de pacotes (1, 32, 128, 256, 512, 768, 1024, 1400, 2048, 4096 e 16384 bytes). As variações de buffers e tamanhos de pacotes se devem a tentativa de testar vários ambientes encontrados em diferentes aplicações. Por exemplo, no caso de aplicações multimídia, mais precisamente voz sobre IP, é mais empregado pacotes com tamanho pequeno. Enquanto aplicações de transmissão de arquivos empregam pacotes

de tamanho grande. O processo de analise se repetiu até que todos os valores de buffers e pacotes fossem associados. A ferramenta Netperf gera dados, e através dos mesmos gerou-se os gráficos apresentados no capítulo de resultados. Para entendermos o comportamento da variação em relação a utilização dos recursos da CPU e da banda de rede, utilizou se diferentes tamanhos de pacotes e *buffers*.

3. Resultados

Para os experimentos aplicados no ambiente proposto, utilizaram quatro tamanhos de *buffers* distintos (8192, 16384, 32768 e 65535 bytes), como já descrito no capítulo anterior, onde se definiu a metodologia do experimento. Como os resultados gerados entre os diferentes tamanhos de *buffers*, se mostraram muito similares, se decidido representa-los utilizando o tamanho de *buffer* de 32768 bytes como valor padrão de *buffer* para o protocolo UDP e 65365 bytes para os resultados relacionados ao protocolo TCP, em redes *Gigabit Ethernet*.

3.1. Protocolo UDP em rede *Gigabit Ethernet* com *Kernel 2.4.27* e o *Kernel 2.6.9*

Através dos dados demonstrados no gráfico 1, o computador Comp.A apresentou resultados semelhantes entre o *kernel 2.4(K2.4)* e o *kernel 2.6(K2.6)*, conforme pode se observar nas linhas respectivas ao desempenho da banda de rede e na utilização de recursos da CPU.

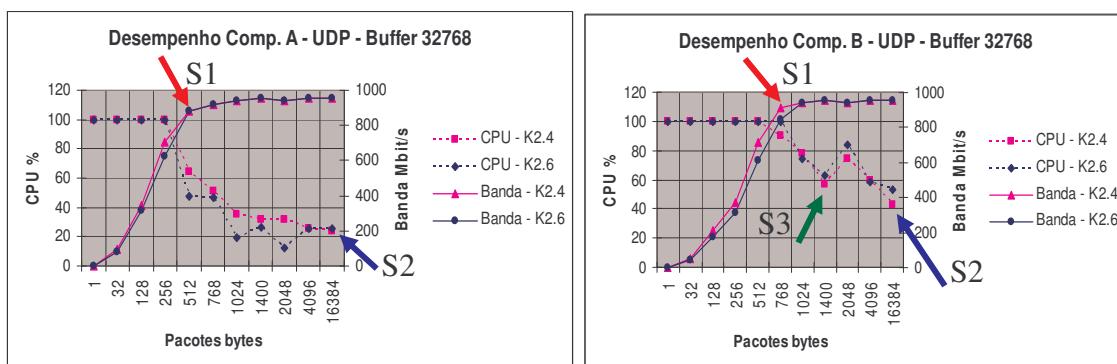


Gráfico 1. Comp.A - CPU versus B. Rede

Gráfico 2. Comp.B – CPU versus B. Rede

Os dados gerados pelo computador Comp.B (gráfico 2) utilizando o *kernel 2.4* e o *kernel 2.6*, apresentaram as mesmas semelhanças, onde observou se que os resultados apresentados para o desempenho da rede e a utilização da CPU seguiram as mesmas características para ambos os *kernels*.

Através do gráfico 1 e 2, pode-se observar a real diferença dos resultados apresentados entre o desempenho da banda da rede e da utilização de recursos da CPU, para os computadores Comp.A e Comp.B (gráficos 1 e 2) utilizando os *kernels* 2.4 e 2.6 do sistema operacional Linux. Conforme estas informações, verificou-se através da seta S1 que o desempenho da banda de rede no Comp.A apresenta melhores resultados quando utiliza pacotes a partir do tamanho de 512 bytes, desta forma, utiliza melhor a banda da rede, em torno de 850 Mbps trabalha próximo a saturação S1, ponto ideal para se ter uma ótima utilização da banda. Outro ponto importante a ser observado, é que aumentando o tamanho dos pacotes disponibilizados na rede, reduz-se drasticamente o consumo de utilização de recursos da CPU.

O melhor desempenho atingido pelo computador Comp.A se deu na utilização de pacotes de tamanho de 4096 e 16384 bytes, onde a banda da rede atingiu o ponto próximo à saturação, e a utilização dos recursos da CPU chegaram a 20%, conforme a seta S2. Já o Comp.B (gráfico 2) não segue o mesmo comportamento, apresentando um desempenho razoável da banda de rede apenas a partir de pacotes de tamanho de 1400 bytes, utilizando em torno de 60% dos recursos da CPU (Seta S3). A melhor utilização dos recursos da CPU ficou em torno de 40%, para pacotes de tamanho de 16384 bytes (seta S2).

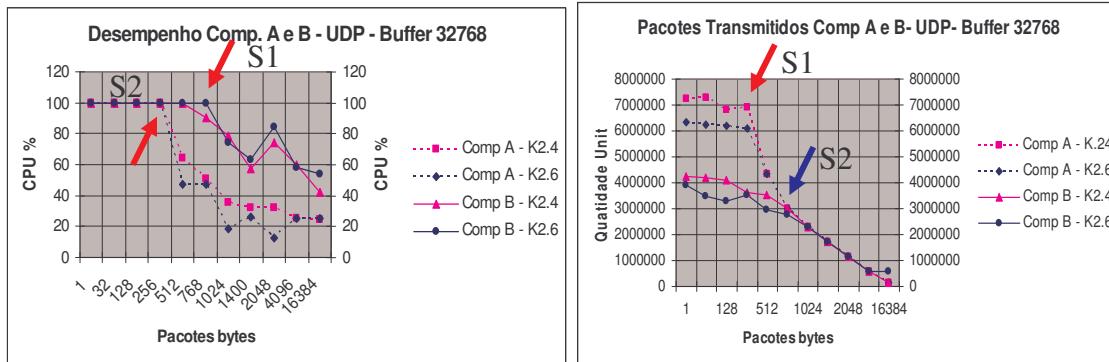


Gráfico 3. CPU - Comp.A versus Comp.B Gráfico 4. Pacote – Comp.A versus Com.B

Realizando uma análise comparativa entre os computadores Comp.A e o Comp.B, através do gráfico 3, pode-se identificar que a utilização da CPU entre os kernels 2.4 e 2.6 apresentaram pequenas diferenças, como já analisado. A utilização dos recursos da CPU é a diferença que mais se destaca entre os equipamentos, onde a taxa de utilização da CPU no Comp.A, para o melhor desempenho, ficou em torno de 20% utilizando pacotes de tamanho 16384 bytes. Já o computador Comp.B obteve uma taxa de 40% de desempenho da CPU utilizando o mesmo tamanho de pacote. Através das setas S1 e S2 no gráfico 3, percebe-se que o Comp.B permaneceu com a utilização da CPU saturada por um período maior que o Comp.A, para os mesmos tamanhos de pacotes.

Comparando a quantidade de pacotes transmitidos entre os computadores Comp.A e Comp.B (gráfico 4), nota-se que o kernel 2.4 transmite uma quantidade de pacotes superior ao kernel 2.6 (seta S1), para ambos os equipamentos, apresentando uma melhor eficiência para pacotes de tamanho menores que 512 bytes. Conforme demonstrado no gráfico 4, à medida que o tamanho dos pacotes aumenta, a quantidade de pacotes transmitidos na rede tende a diminuir e se equiparar entre ambos os kernels. A taxa de transmissão na rede, para pacotes de tamanho menores que 256 bytes, pelo Comp.A atingiu quase o dobro da quantidade de pacotes transmitidos pelo Comp.B, significando um ótimo desempenho na transmissão de pacotes.

Nota-se que o fato de o computador Comp.B utilizar mais CPU (Seta S1 gráfico 3) não significa que a taxa de transmissão de pacotes (seta S2 gráfico 4) seja a melhor, pois o computador Comp.A obteve uma baixa taxa na utilização de recursos da CPU trafegando uma quantidade superior de pacotes na rede, do computador Comp.B. É possível que as diferenças apresentadas através dos gráficos 3 e 4, entre os computadores Comp.A e Comp.B, sejam uma limitação do adaptador de rede ou uma característica do driver de rede, pois utilizam diferentes tipos de adaptadores de rede *Gigabit Ethernet*.

Como o protocolo UDP é um protocolo não orientado a conexão, ou seja, recomendado para transmissão de dados multimídia (tempo real) de voz e imagem, ideal para se trabalhar com pacotes menores, observou-se que o *kernel* 2.4 apresentou melhor desempenho para este tipo de pacotes em relação ao *kernel* 2.6 (Postel, 1980). Consta-se então que o *kernel* 2.4 é o mais indicado para este processo. Um outro fator que está relacionado entre as diferenças dos computadores, é que o computador Comp. A obteve melhor desempenho na utilização da CPU e na banda de rede. Pois a quantidade de pacotes pequenos (256 bytes) transmitidos pelo Comp.A foi superior, atingindo quase o dobro da quantidade de pacotes transmitidos na rede.

3.2. Protocolo TCP em rede Gigabit Ethernet com Kernel 2.4.27 e o Kernel 2.6.9

A utilização da CPU e o desempenho da banda da rede entre os *kernels* 2.4 e 2.6 para o Comp.A apresentam as mesmas características, conforme demonstrado na seta S1 no gráfico 5.

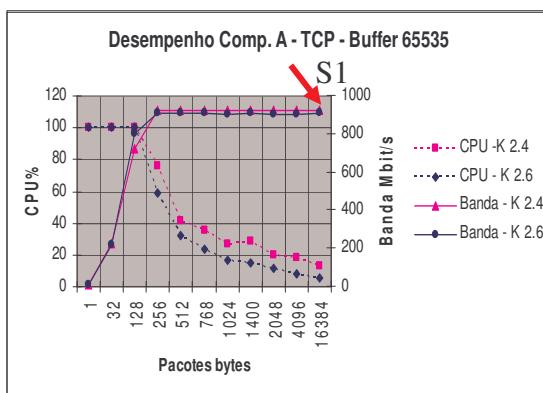


Gráfico 5. Comp.A - CPU versus B. Rede

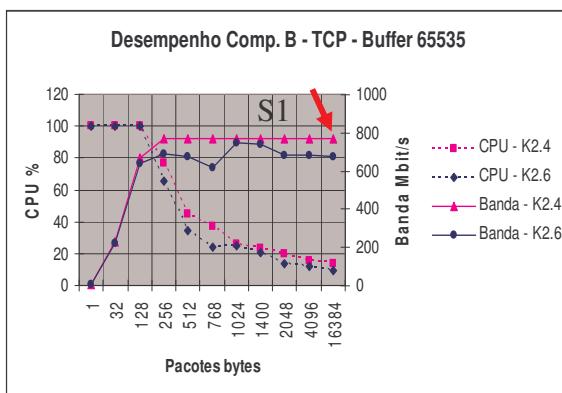


Gráfico 6. Comp.B - CPU versus B. Rede

Os dados representados pelo gráfico 6, demonstram que os *kernels* 2.4 e o 2.6 apresentaram o mesmo comportamento para o Comp.B. Comparando o desempenho da banda da rede entre os computadores Comp.A e Comp.B (gráficos 5 e 6), observou-se que o Comp.A utilizando o *kernel* 2.4, seta S1 gráfico 5, apresenta um melhor ganho na banda de rede ficando em torno de 900Mbps.

Realizando uma análise comparativa entre os computadores Comp.A e Comp.B sobre o desempenho da banda de rede mostrados nos gráficos 5 e 6, pode-se verificar que o protocolo TCP não é indicado para trabalhar com tamanho de pacotes menores que 128 bytes, pois não apresenta um bom desempenho da banda de rede.

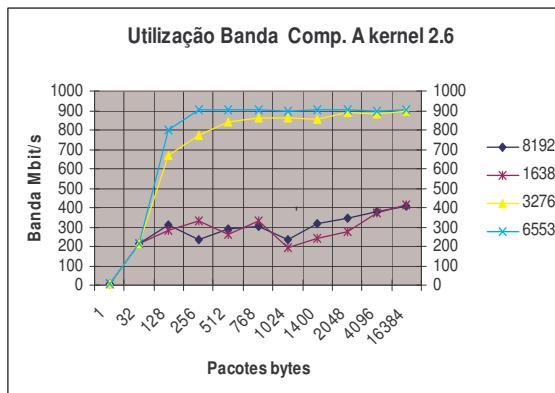


Gráfico 7. Comp.A – Utilização B. Rede

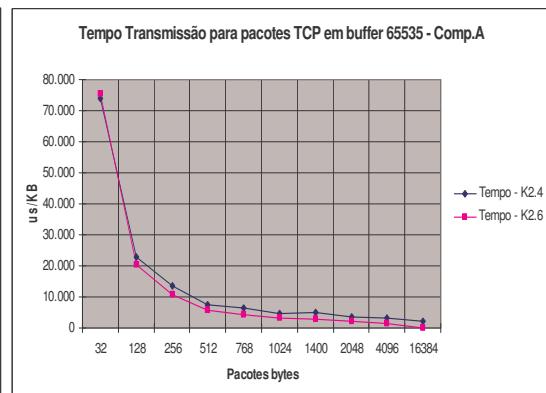


Gráfico 8. Tempo de transmissão pacote TCP

Comparando os dados apresentados através do gráfico 7, utilizando o protocolo TCP e *kernel* 2.6, pode-se verificar que existe uma melhor otimização apenas para buffers de tamanhos maiores que 32768 bytes. Nota-se que o *kernel* 2.6 não se preocupou em tratar *buffers* de tamanhos menores que 16384 bytes.

Os tempos de transmissão de pacotes utilizando o protocolo TCP para o computador Comp.A e o computador Comp.B não apresentaram qualquer diferença significativa, conforme demonstrado através do gráfico 8.

4. Conclusão

Ao longo dos experimentos identificou-se as variáveis que têm relação com o desempenho dos kernels 2.4 e 2.6, principalmente para o objetivo secundário da análise (otimização para VOIP) e também foi possível verificar na prática algumas características do comportamento dos protocolos UDP e TCP em rede *Gigabit Ethernet*. Percebe-se que a diferença mais significativa entre o computador Comp.A e o Comp.B está relacionada com as características dos adaptadores de rede ou de algum *driver* que faz acesso aos mesmos. Através dos dados contidos nos gráficos apresentados, pode-se perceber que os adaptadores de rede foram explorados até o máximo de sua capacidade. Com isso, em determinado momento utilizando tamanhos de pacotes específicos acabaram consumindo muito dos recursos de utilização da CPU, onde o consumo em alguns casos chegaram a 100% da utilização. Pode-se perceber que o protocolo UDP foi o que mais sofreu com este tipo de influência.

Através do experimento e dos dados contidos nos gráficos, o *kernel* 2.6 apresentou resultados semelhantes ao *kernel* 2.4, com isso, não foi notado nenhum ganho aparente que se tenha percebido nas análises realizadas, não havendo nenhum ganho interno perceptível na área de *Gigabit Ethernet* para o *kernel* 2.6, utilizando os protocolos UDP e TCP. Logo o TCP não sofreu ganho na pilha de transmissão de dados, e a taxa de utilização de recursos da CPU se manteve em um patamar alto para pacotes de tamanhos menores. Estes resultados apresentam que as alterações futuras realizadas podem trazer benefícios para ambos os kernels. Com isso em mente, o grupo decidiu continuar os trabalhos sobre o *kernel* 2.6, por ser o mais atual.

Um melhor desempenho para o protocolo UDP pode ser atingido através de otimização dos *drivers* de acesso ao dispositivo de rede. Quanto a este, um melhor desempenho pode ser atingido na otimização do hardware para trabalhar com pacotes de menor tamanho, de acordo com os dados demonstrados através dos gráficos e entre os equipamentos analisados no experimento.

Como identificado através dos dados analisados no capítulo de resultados, existe uma grande área de otimização para o *kernel* 2.6, em relação aos protocolos UDP, visto que não houve mudanças significativas entre as duas versões para a rede *Gigabit Ethernet*. Através desta, grandes quantidades de pacotes na rede de tamanhos menores não influenciaram na utilização da banda de rede, mas sim na utilização dos recursos da CPU. Devido a estas análises, o seguimento deste trabalho se dará através de estudos sobre VOIP (voz sobre IP), onde será realizado um trabalho voltado ao comportamento no ambiente proposto neste artigo de pesquisa e futuras otimizações.

Referências Bibliográficas

- Hagen, William (2004) Artigo Sobre “Migração do Linux 2.6”, URL <http://linuxdevices.com/articles/AT5793467888.html>, Fevereiro
- NetPerf, (2005), URL <http://www.netperf.org/netperf/NetperfPage.html>, Acessado em Setembro de 2005.
- Roesler, V. (2001), “Desempenho em Redes TCP/IP”, URL http://www.inf.unisinos.br/~roesler/disciplinas/0_redes/10_tcpip/4_Desempenho/desempenhotcp.pdf, Maio
- Postel J. (1980), RFC 768 “UDP (User Datagram Protocol)”, Agosto
- Peterson, L. e Dave, S. (2004), “Redes de Computadores 2th Edition”, LTC, Rio de Janeiro
- Tanenbaum, A. (2003), “Redes de Computadores 3th Edition”, tradução Vandenberg D. de Souza, Elsevier ISBN: 8535201572, Rio de Janeiro,

Uma Proposta de Escalonamento Descentralizado de Tarefas para Computação em Grade

Lucas Alberto Souza Santos¹, Patrícia Kayser Vargas^{2,3}, Cláudio F. R. Geyer¹

¹Instituto de Informática, UFRGS – Porto Alegre, RS, Brasil

²UniLaSalle – Canoas, RS, Brasil

³COPPE/Sistemas, UFRJ – Rio de Janeiro, RJ, Brasil

kayser@cos.ufrj.br, {lassantos, geyer}@inf.ufrgs.br

Resumo. Neste artigo é proposto um modelo de escalonamento de tarefas em ambiente de grade. Este modelo segue uma estrutura descentralizada baseada na tecnologia das redes Par-a-Par (P2P). As redes P2P possuem as vantajosas características de tolerância a falhas e escalabilidade. Com o progressivo aumento do compartilhamento de recursos geograficamente distribuídos, estas características se tornarão fundamentais para o sucesso da Computação em Grade.

O modelo proposto será integrado ao projeto ISAM na forma de um framework de escalonamento de aplicações. O sistema ISAM é um ambiente de Computação Pervasiva que engloba características da Computação em Grade. O modelo proposto será avaliado primeiramente por simulação e depois prototipado para avaliação experimental.

1. Introdução

A Computação em Grade é cada vez mais utilizada para a execução de aplicações que demandam elevados custos computacionais. Atualmente, os maiores usuários da tecnologia de grades computacionais são os pesquisadores da área de BioInformática e HEP(*High Energy Physics*). Há uma grande demanda nestas áreas por desempenho computacional e compartilhamento de dados. A tendência atual é o aumento da utilização das grades computacionais. Quando o número de domínios administrativos aumentar, exigirá soluções de escalonamento cada vez mais sofisticadas. Estas soluções devem promover uma maior escalabilidade do sistema, mantendo boa performance das aplicações que executam, além de utilizar mecanismos de tolerância a falhas, e suportar políticas locais de controle do uso dos recursos.

2. Sistemas de Gerenciamento de Recursos em Grade

Nota-se nos últimos anos um crescente desenvolvimento de grades computacionais e seus respectivos middlewares: Globus [Foster et al. 2001], Condor/Condor-G [THAIN et al. 2003], Legion [GRIMSHAW et al. 1999], Our-Grid [ANDRADE et al. 2003], ISAM [Yamin et al. 2003]. Muitas das propostas de gerenciamento de recursos e escalonamento de tarefas em grades formadas por domínios administrativos autônomos seguem uma organização hierárquica (forma de árvore) ou centralizada. Abaixo estão as características das principais propostas de gerenciamento de recursos em grade que utilizam um modelo de organização centralizado ou hierárquico:

O sistema **Globus** provê uma infraestrutura que permite que aplicações vejam recursos distribuídos heterogêneos como uma única máquina virtual. O projeto Globus é um esforço de pesquisa multi-institucional que busca permitir a construção de grades computacionais. O Globus oferece serviços de informação através de uma rede de diretórios hierárquica baseada em **LDAP**, esta rede é chamada **Metacomputing Directory Services (MDS)**. O MDS é formado por um conjunto de GRIS (Grid Resource Information Service) indexados por GIIS (Grid Index Information). Os recursos atualizam suas informações no GRIS periodicamente. Ferramentas de alto-nível como resources brokers e meta-escalonadores realizam buscas através de consultas ao MDS usando protocolos LDAP. O namespace do MDS é organizado hierarquicamente em forma de árvore. A versão atual do Globus Toolkit (GT4), utiliza uma nova versão do MDS. MDS4 possui funcionalidades similares mas utiliza protocolos baseados em XML ao invés do protocolo LDAP, e possui uma série de aprimoramentos.

O modelo de escalonamento do sistema Globus é descentralizado. O Globus é um toolkit para grid que não fornece suporte nativo à políticas de escalonamento, mas ele permite que resource brokers externos adicionem esta capacidade. Os serviços Globus são utilizados em uma variedade de sistemas de gerenciamento de recursos: Nimrod/G, NetSolve, GrADS, AppLeS e Condor-G.

O **Condor** é um sistema de gerenciamento de recursos, que tem o objetivo de prover uma plataforma computacional de alta-vazão, desenvolvido pela Universidade de Wisconsin em Madison nos EUA. O Condor realiza a descoberta de recursos ociosos em uma rede e a alocação destes recursos para execução de tarefas. A função principal do Condor é utilizar máquinas distribuídas que de outra forma estariam ociosas, promovendo a máxima utilização dos recursos disponíveis. O Condor é formado por um conjunto de diferentes daemons. Um cluster de estações de trabalho chamado de Condor pool é gerenciado pelo sistema. O sistema Condor possui uma arquitetura de escalonamento centralizada. O sistema Condor possui uma funcionalidade chamada flocking que permite que usuários possam utilizar recursos compartilhados de múltiplas Condor pools distribuídas.

3. Motivação

Um escalonador para grade depende diretamente da sistema de informação, módulo que possibilita a descoberta de recursos e monitoração. O Toolkit Globus e o sistema Condor, embora amplamente utilizado nas grades atuais, não apresentam as seguintes características fundamentais para suportar o crescente uso da computação em grade:

- Elevada escalabilidade;
- Tolerância a falhas e ataques;
- Auto-adaptação e dinamicidade;
- Completa autonomia dos centros regionais sobre seus recursos.

O modelo estruturado em árvore do sistema MDS do Globus é escalável e possui boa eficiência, todavia os nós mais altos da hierarquia são pontos críticos de falha, tornando o sistema suscetível a falhas e ataques. Qualquer falha no sistema de informação dos recursos afeta o escalonamento da grade.

A alternativa proposta neste trabalho é uma estrutura de escalonamento descentralizado colaborativo, onde domínios administrativos autônomos compartilham re-

cursos com outros domínios vizinhos, criando um sistema de compartilhamento, não-hierárquico, dinâmico, organizado em rede **Par-a-Par (Peer-to-Peer) P2P**. Este modelo de escalonamento será integrado ao ambiente em grade do sistema ISAM.

O sistema **ISAM (Infra-estrutura de Suporte às Aplicações Móveis Distribuídas)** [Yamin et al. 2002], em desenvolvimento no Instituto de Informática da UFRGS, é um middleware direcionado ao gerenciamento de recursos em redes heterogêneas, suportando a execução de aplicações distribuídas baseadas em componentes. A arquitetura do ambiente ISAM é organizada na forma de células autônomas cooperativas. Cada célula possui um escalonador local e políticas próprias de uso de seus recursos. Desta forma, um modelo de escalonamento de tarefas descentralizado cooperativo para a plataforma ISAM é fundamental para que o sistema alcance níveis elevados de escalabilidade, balançamento de carga e tolerância a falhas.

4. Trabalhos Relacionados

Existem algumas propostas de modelo para escalonamento descentralizado não-hierárquico em grades. Em [Shan et al. 2003], os autores afirmam que uma arquitetura descentralizada em P2P possibilitou melhor escalabilidade e tolerância a falhas, quando comparada com uma arquitetura centralizada. O projeto **OurGrid** [ANDRADE et al. 2003] utiliza uma rede de favores estruturada em P2P para compartilhamento de recursos de forma justa entre participantes de uma grade. O modelo de aplicações do sistema OurGrid é do tipo Bag-of-Task. O OurGrid possui um protótipo implementado com a tecnologia JXTA [Gong 2001]. O sistema **Triana Grid** [Taylor et al. 2003] utiliza um modelo de grade descentralizado em P2P. O Triana Grid utiliza as tecnologia JXTA e OGSA [Foster et al. 2002]. O sistema **LEAF (Learnig Agent based FIPA toolkit)** [Lynden and Rana 2002] é um sistema de grade descentralizado que utiliza explicitamente o modelo de agentes como pares de uma rede P2P.

5. Modelo Proposto

O modelo de grade proposto neste trabalho objetiva criar uma comunidade de centros regionais de computação como pares de uma rede lógica (*overlay network*) P2P. Os centros (domínios) que participam da rede P2P procuram compartilhar recursos computacionais, ou seja, utilizam recursos remotos e fornecem recursos computacionais para usuários de outros domínios. O sistema possui suporte para que os donos dos recursos possam estabelecer políticas de acesso a seus recursos, assim eles têm controle total sobre sua participação na grade computacional. Cada domínio administrativo da grade P2P é formado por uma célula ISAM, que é composta de uma máquina base e outros nodos computacionais. A máquina base é a entidade responsável pelo gerenciamento de todos os recursos da célula, onde os serviços principais executam, e onde executará o serviço que manterá a rede de escalonamento P2P funcionando.

Desta forma, a arquitetura da rede P2P formada pelas células ISAM se aproxima do modelo **Super-Peer** [Yang and Garcia-Molina 2002] de rede P2P (Figura 1). Este modelo de rede, é considerado um modelo híbrido de rede par-a-par, pois existem nodos da rede que se comportam como entidades centrais (base da célula) para alguns outros nodos (recursos da célula). Para facilitar a compreensão do modelo, podemos abstrair o modelo super-peer, encapsulando todos os recursos de uma célula ISAM, em uma única entidade nodo da rede P2P, sem perda informações.

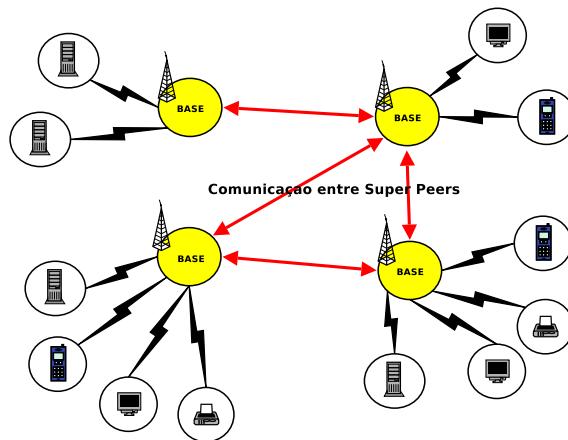


Figura 1. Modelo de rede super-peer formado pelas células ISAM

Os usuários deverão submeter aplicações organizadas em forma de DAG(Directed Acyclic Graph) a um dos pares da rede e o serviço de escalonamento P2P servirá de *Grid Broker* para comunicação com outros *grid brokers* de células vizinhas. Os recursos de um nodo da rede (célula ISAM) poderão ser acessados por usuários locais da célula, ou por usuários de outras células através da comunicação entre Grid Brokers. O serviço de Grid Broker é composto por um escalonador de grade com uma fila de tarefas globais, um escalonador local com uma fila de tarefas locais, um repositório de políticas e uma lista de nodos vizinhos (Figura 2).

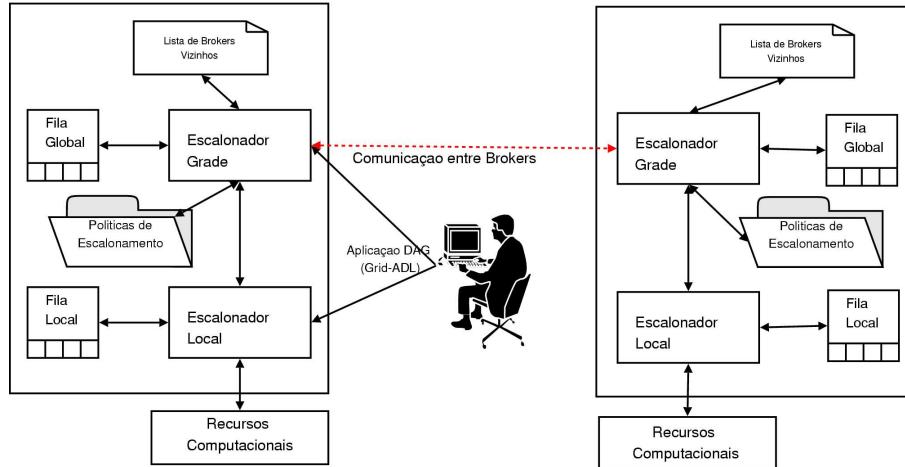


Figura 2. Arquitetura do componente Grid Broker

Os componentes do serviço de broker possuem as seguintes funções:

Escalonador Grade Sua função é escalar aplicações descritas na linguagem Grid-ADL nos recursos da grade. Este componente utiliza protocolos P2P para manter a rede em funcionamento e gerenciar a lista de células vizinhas na rede lógica par-a-par. É responsável por verificar o estado das tarefas das aplicações, e tomar as decisões adequadas para que as aplicações terminem com sucesso, mesmo em caso de falhas. Para escalarar, este componente compara o estado dos recursos da célula com as políticas de escalonamento. Através de um algoritmo de escalonamento, decide qual tarefa da fila de tarefas globais pode executar nos recursos

de sua célula local. As tarefas escalonadas são transferidas para a fila de tarefas locais, e não poderão mais migrar para outra célula.

Fila Global Nesta fila estão as tarefas globais da grade, pertencente a uma dada aplicação (DAG) disparada em algum ponto da grade. Este tipo de tarefa pode migrar de um centro administrativo a outro, de célula em célula, até que seja colocada em uma fila de escalonamento local.

Escalonador Local Sua função é escalar as tarefas da fila local nos recursos disponíveis da célula. Este componente realiza a monitoração do estado dos recursos e a contabiliza a utilização destes. Um algoritmo de escalonamento é responsável pela seleção da tarefa a ser executada e escolha do recurso onde será destinada a tarefa. As decisões do escalonamento são diretamente influenciadas pelas políticas de escalonamento definidas pelos gerentes dos recursos da célula. O usuário local (não-remoto) da célula tem a opção de disparar uma aplicação Grid-ADL diretamente neste escalonador. Desta forma, as tarefas desta aplicação passarão diretamente à fila de tarefas locais.

Fila Local Nesta fila estão as tarefas locais da célula, pertencente a uma dada aplicação (DAG) disparada em algum ponto da grade. Este tipo de tarefa não pode mais migrar a outro centro administrativo, será executada nos recursos da célula local.

Políticas de Escalonamento As políticas de escalonamento são propriedades que controlam quais recursos são escalonados dentre os recursos que atendem as requisições das tarefas. Estas propriedades definem permissões de **quem** pode acessar, **quando** acessar e **como** podem ser acessados os recursos conectados ao domínio administrativo. As políticas são divididas em dois grupos: As *Políticas Globais* são definidas pelo gerente do domínio administrativo, estas políticas são respeitativas ao conjunto formado por todos os recursos da célula. As *Políticas Locais* estão relacionadas a recursos individuais da célula, como um computador ou um banco de dados. Estas são definidas pelo dono do recurso.

6. Conclusão e Trabalhos Futuros

O modelo atende os requisitos de um Escalonador para Grades. O Grid Broker é a entidade responsável por manter a rede P2P, executar as políticas definidas pelos donos dos recursos e escalar as aplicações. A proposta será avaliada inicialmente através de simulação com o simulador MONARC 2 [monarc2 2005] implementado na linguagem Java. As principais classes do sistema MONARC serão estendidas para tornar possível o escalonamento em rede P2P. O modelo de aplicação utilizado no framework será baseado no framework GRAND [Vargas et al. 2004], que define aplicações descritas através de um DAG. As aplicações do modelo GRAND são constituídas por tarefas com ou sem dependências de arquivo entre si. O protocolo P2P utilizado no framework JXTA [Gong 2001] está sendo analisado e possivelmente será adotada uma extensão deste protocolo para atender os requisitos do modelo proposto.

As informações obtidas na simulação serão utilizadas para implementar um protótipo do modelo proposto. O protótipo será desenvolvido a partir dos componentes do middleware EXEHDA incluído no sistema ISAM utilizando a linguagem Java. A validação do protótipo será feita através da análise do desempenho de algumas aplicações escolhidas de forma a abranger os tipos de carga (leve, baixa, alta)(computacional, armazenamento) a que estão sujeitos os ambientes de computação em grade.

Referências

- ANDRADE, N., CIRNE, W., BRASILEIRO, F. V., and ROISENBERG, P. (2003). Our Grid: An approach to easily assemble grids with equitable resource sharing. In *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*.
- Foster, I., Kesselman, C., Nick, J. M., and Tuecke, S. (2002). The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. <http://www.globus.org/research/papers/ogsa.pdf>.
- Foster, I., Kesselman, C., and Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, 15(3).
- Gong, L. (2001). Jxta: A network programming environment. *IEEE Internet Computing*, 5(3):88–95.
- GRIMSHAW, A. S., FERRARI, A., KNABE, F., and HUMPHREY, M. (1999). Wide-area computing: Resource sharing on a large scale. *IEEE Computer*, 32(5):29–36.
- Lynden, S. and Rana, O. F. (2002). Coordinated learning to support resource management in computational grids. In *P2P '02: Proceedings of the Second International Conference on Peer-to-Peer Computing*, page 81, Washington, DC, USA. IEEE Computer Society.
- monarc2 (2005). Models of networked analysis at regional centres for lhc experiments.
- Shan, H., Oliker, L., and Biswas, R. (2003). Job superscheduler architecture and performance in computational grid environments. In *Proceedings of the Supercomputing 2003*.
- Taylor, I., Shields, M., Wang, I., and Philp, R. (2003). Distributed p2p computing within triana: A galaxy visualization test case. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 16.1, Washington, DC, USA. IEEE Computer Society.
- THAIN, D., TANNENBAUM, T., and LIVNY, M. (2003). Condor and the Grid. In BERMAN, F., FOX, G., and HEY, T., editors, *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley.
- Vargas, P. K., Dutra, I. d. C., and Geyer, C. F. (2004). Gerenciamento hierárquico de aplicações em ambientes de computação em grade. In *Escola Regional de Alto Desempenho (ERAD 2004)*, Pelotas, RS.
- Yamin, A., Augustin, I., Barbosa, J., and Geyer, C. F. (2002). ISAM: a pervasive view in distributed mobile computing. In *Proceedings of the IFIP TC6 / WG6.2 & WG6.7 Conference on Network Control and Engineering for QoS, Security and Mobility (NET-CON 2002)*, pages 431–436.
- Yamin, A., Augustin, I., Barbosa, J., Silva, L. C. d., Real, R. A., Cavalheiro, G., and Geyer, C. F. (2003). Towards merging context-aware, mobile and grid computing. *International Journal of High Performance Computing Applications*, 17(2):191–203.
- Yang, B. and Garcia-Molina, H. (2002). Designing a super-peer network. Technical Report 2002-13, Stanford University.

Projeto de Implementação dos Protocolos de Transporte no Modelo TCP/IP em Placas FPGAs

Alisson Roggia Ceolin¹, Cristiano Bonato Both¹

¹Departamento de Informática - Universidade de Santa Cruz do Sul (UNISC) – RS – Brasil

{alissonc, cboth}@unisc.br

Abstract: This article describes a hardware model project implementation TCP/IP (Transport Control Protocol/Internet Protocol) transport protocol, making possible the exchange messages between FPGAs (Field Programmable Gate Arrays). These plates make possible the hardware programming for specific application, normally aiming high performance. It is credited that with a set of FPGAs linked can be configured a new model of aggregates and to acquire an equal or superior performance the conventional aggregates.

Resumo: Este artigo descreve um projeto de implementação dos protocolos de transporte do Modelo TCP/IP (Transport Control Protocol/Internet Protocol) em hardware, possibilitando a troca de mensagens entre FPGAs Field Programmable Gate Arrays). Estas placas possibilitam a programação do hardware para uma aplicação específica, normalmente visando alto desempenho. Acredita-se que com um conjunto de FPGAs interligadas pode-se configurar um novo modelo de agregados e adquirir um desempenho igual ou superior a agregados convencionais.

1. Introdução

Agregados são conjuntos de computadores pessoais (PCs), dedicados ou não, e interligados por uma rede de alta velocidade utilizados para fins específicos. Esta arquitetura apresenta uma boa relação custo/benefício, embora existam sub-utilizações de alguns dispositivos de entrada e saída e algumas camadas de softwares ocasionando perda de desempenho.

Uma arquitetura de computadores dedicada para extrair o máximo desempenho, deve possuir a menor quantidade de camadas de softwares e a aplicação deve ser voltada para aquele hardware [Dehon 1999]. Um modelo que se enquadra nesta arquitetura são as placas FPGAs, que podem ser uma alternativa para problemas que necessitam um alto processamento.

Podem ser desenvolvidos agregados de FPGAs reconfiguráveis para adquirir desempenho em aplicações distribuídas, mas para que isso ocorra, estas FPGAs devem possuir meios de transmissão de dados, isto é, os protocolos de comunicação devem ser implementados em hardware.

Este artigo aborda a implementação dos protocolos de transporte em um agregado de FPGAs, sendo organizado com as seguintes seções: na seção 2 descreve-se o projeto ACR para agregados de computadores reconfiguráveis, na seção 3 ocorre uma breve descrição sobre os protocolos de transporte, na seção 4 é relatado o estágio atual

da implementação dos protocolos de transporte na placa FPGA e finalmente na seção 5 encontram-se as conclusões deste projeto.

2. O projeto ACR - Agregados de Computadores Reconfiguráveis

O projeto ACR visa utilizar a lógica encontrada em dispositivos reconfiguráveis para a obtenção de desempenho em aplicações distribuídas específicas. Sua motivação provém da possibilidade de construir agregados de processadores dedicados à resolução de cálculos que necessitam de alto poder de processamento. Enviando instruções diretamente para os processadores não há necessidade de tratamentos de interrupções adicionais de dispositivos de E/S, assim como, há redução de *overhead* em camadas de softwares, como por exemplo, pode-se citar o próprio sistema operacional, encontrados em agregados convencionais [Rose 2003].

Este modelo de agregados com FPGAs reduz o espaço físico a ser ocupado (salas com mecanismos de controle de refrigeração), simplifica o hardware a ser utilizado e com isso, espera-se adquirir um desempenho igual ou superior aos encontrados em arquiteturas convencionais. O problema encontrado no desenvolvimento de aplicações distribuídas em agregados de computadores tradicionais deverá ser encontrado também neste novo paradigma, pois estas aplicações são implementadas com um forte acoplamento ao hardware [Rose 2003].

Um exemplo de uma aplicação que necessita um grande poder de processamento foi desenvolvida no projeto ACR, para uma arquitetura específica. Este programa procura a melhor alternativa entre espectros de produtos farmacológicos utilizando algoritmos genéticos [Aguiar 2005].

Esta aplicação possui várias versões para diferentes arquiteturas de computadores, entre elas, está sendo desenvolvida uma versão seqüencial para placas reconfiguráveis. A implementação deste programa está sendo feita na linguagem Java e a sintetização para o microcontrolador (FentoJava) é realizado pela ferramenta Sashimi [Carro 2001] [Freitas 2004], que transpõe dinamicamente as instruções de uma linguagem de alto nível para as instruções interpretadas pelo microprocessador.

Para a utilização da aplicação distribuída em um conjunto de placas FPGAs deverá existir um mecanismo de troca de mensagens. Diversas são as arquiteturas para a comunicação de dados entre dispositivos, entretanto o padrão utilizado em redes de computadores é o modelo TCP/IP, porque são protocolos simples, funcionais e interoperáveis. Na Figura 1, pode-se visualizar um agregado de dispositivos reconfiguráveis contendo um nó servidor e “n” nós clientes.



Figura 1. Estrutura de comunicação entre as FPGAs.

Pode-se verificar na figura apresentada, um nó mestre interligado com diversos outros nós. A proposta é executar um programa em modo paralelo sendo iniciado no mestre auxiliado pelos seus nós. Este auxílio no processamento deverá ser realizado de acordo com a aplicação a paralelizada, normalmente são rotinas bem definidas que recebem e retornam parâmetros para o nó mestre, ou mesmo, um outro nó. Para esta troca de mensagens tornar-se possível neste modelo, toda a pilha TCP/IP, exceto a aplicação, deve ser implementada diretamente no hardware da placa.

Não encontrando-se uma versão completa do TCP/IP para ser utilizada, atualmente, estão sendo desenvolvidos novas funções a pilha encontrada, como o envio de pacotes do protocolo UDP (*User Datagram Protocol*) [Freitas 2004]. O objetivo deste projeto é implementar o restante dos protocolos de transporte da pilha TCP/IP, possibilitando o desenvolvimento de aplicações distribuídas em FPGAs, o que atualmente só é possível de forma seqüencial.

3. Protocolos de Transporte

A função dos protocolos de transporte é prover a comunicação entre aplicações instanciadas em computadores distintos mediante a interligação de uma rede de dados. Dois protocolos são definidos no modelo TCP/IP, UDP (*User Datagram Protocol*) e TCP (*Transmission Control Protocol*) [Comer 2001]

O UDP é um protocolo que possui sua implementação simplificada, na qual, pode-se adquirir alta performance. As suas principais desvantagens são a não garantia de entrega, ordenação e controle de duplicação de pacotes. Este protocolo é normalmente utilizado em meios de transmissão físicos que garantam a entrega de seus quadros.

O TCP é um protocolo de implementação mais complexa, é estabelecida uma conexão entre as máquinas origem e destino (conexão fim-a-fim), incluindo métodos que garantam a confiabilidade na entrega de pacotes, de forma ordenada, sem duplicação e com controle de fluxo.

Este protocolo originalmente não determina como a camada TCP deve ser implementada em cada equipamento (Darpa, 1981). Deste modo, é possível implementá-lo de maneiras distintas desde que o protocolo seja respeitado. As suas principais implementações são TCP Tahoe, TCP Reno e TCP Vegas.

De acordo com [Fall 1996], o TCP Tahoe opera com as técnicas *Slow Start*, *Congestion Avoidance* e *Fast Retransmit*. O principal problema desse algoritmo ocorre quando muitos segmentos são perdidos fazendo acionar a técnica *Slow Start* diversas vezes e provocando, dessa forma, uma baixa eficiência na rede.

O TCP Reno é um dos mais utilizados na Internet, atualmente, difere do TCP Tahoe, pois o ACK (número de confirmação) é realizado para cada segmento enviado e, além disso, a técnica *Fast Recovery* é incluída fazendo com que a janela de congestionamento seja aumentada pelo número de ACKs duplicados que o TCP transmissor recebe antes do recebimento de um novo ACK. A baixa eficiência do TCP Reno ocorre quando múltiplos segmentos são perdidos de uma *CWND* (janela de congestionamento), porque este algoritmo não retorna para a fase *Slow Start* e sim aciona a fase *Congestion Avoidance*, reduzindo a *CWND* para a metade do seu valor diversas vezes. Entretanto, se o timeout expirar, o TCP Reno começa a transmissão pela fase *Slow Start* [Fall 1996].

O TCP Vegas [Brakmo 1994] é uma implementação introduzida no TCP por meio de algumas modificações realizadas no mecanismo de controle de congestionamento no lado do transmissor. Nele o transmissor antecipa o princípio de congestionamento monitorando a diferença entre a taxa esperada e a taxa real. O Vegas usa a estratégia de ajuste da taxa de envio de dados da origem (janela de congestionamento) na tentativa de manter um pequeno número de pacotes armazenados nos switches durante uma transmissão. Ele é considerado eficiente referente a perda e atraso médio dos pacotes, porém deve ser atribuído tanto para o transmissor quanto o receptor para estas vantagens serem percebidas.

Os protocolos UDP e TCP são utilizados em diversas aplicações e deverão ser implementados em placas FPGAs, deste modo, o programador poderá optar por um protocolo otimizado, visando performance na entrega de pacotes sem conexão, ou um protocolo confiável (com conexão), entretanto com maior *overhead*. A versão do protocolo TCP a ser utilizada no projeto é a TCP Reno por ser utilizado em larga escala mundialmente e possuir um bom desempenho, porém não é estinta a idéia de implementar-se futuramente outro modelo TCP visando maior desempenho entre as FPGAs.

4. Protocolos de Transporte na placa FPGA

Para a programação de placas FPGAs devem ser utilizadas linguagens de descrição de hardware como o VHDL (*Very High Speed Integrated Circuit Hardware Description Language*), na qual todo o hardware é projetado de acordo com a necessidade da aplicação, permitindo a descrição do hardware em entidades interoperáveis.

Existem implementações da pilha TCP/IP desenvolvidas em VHDL, para sua utilização em hardware. Normalmente essas implementações possuem protocolos proprietários, não *open source* e/ou seu desenvolvimento é parcial, como por exemplo o projeto XSV Board [Sutton 2001], que possui implementados os protocolos: ARP (*Address Resolution Protocol*), IP e ICMP (*Internet Control Message Protocol*).

Para a utilização desta pilha TCP/IP com uma aplicação, por exemplo, a procura a melhor alternativa entre espectros de produtos farmacológicos [Aguiar 2005], devem existir componentes que interligam as interfaces do microcontrolador fentoJava com a pilha TCP/IP. Estes componentes de interligação foram desenvolvidos pelo [Freitas 2004] no projeto ACR.

Para o seu completo funcionamento deverão ser descritos em hardware os protocolos de transporte citados na seção 3.

Como descrito anteriormente, o mecanismo de recebimento de mensagens UDP já está completo e descrito na entidade “udp.vhd”. Esta entidade será expandida convertendo-se o código já existente formando o mecanismo de envio de pacotes UDP.

O protocolo TCP necessitará uma boa abstração de seu funcionamento e também uma adaptação para ser possível sua execução em FPGAs. Para a sua implementação, poderão ser utilizadas premissas encontradas em “icmp.vhd”, originalmente desenvolvido para enviar “echo” e responder “replay”, porém, possui um mecanismo semelhante ao necessário pelo protocolo TCP, no instante do envio e recebimento dos pacotes.

Todas as características como: conexão fim-a-fim, ordenamento, controle de fluxo e duplicação de pacotes deverão ser estudadas amplamente possibilitando uma implementação fiel deste protocolo em um modelo FPGA. Neste momento, o interesse é implementar o protocolo TCP com o máximo de suas características possíveis, porém em trabalhos futuros poderão surgir versões mais otimizadas vizando um desempenho ainda maior na execução de aplicativos distribuídos.

5. Conclusão

Com a realização deste projeto espera-se viabilizar a integração entre FPGAs possibilitando a utilização deste agregado para a execução distribuída de aplicações específicas. Acredita-se que, mesmo com *clocks* baixos (100MHz) comparados aos computadores convencionais (2GHz e 3GHz), pode-se executar aplicações distribuídas alcançando desempenhos satisfatórios, pois são reduzidas as complexidades dos hardwares e softwares.

No projeto ACR foi implementado uma aplicação que necessita poder computacional, para servir de caso de uso, também foi desenvolvido componentes que interligam as interfaces do microcontrolador fentoJava com uma pilha TCP/IP incompleta.

Para a implementação distribuída de uma aplicação em um agregado de placas FPGAs deve-se implementar os protocolos de transporte, possibilitando a troca de mensagens entre essas placas. Esta implementação está sendo realizada atualmente e após essa etapa ser concluída deverá ser realizada uma grande quantidade de testes para comparar o desempenho entre os diferentes tipos de arquiteturas.

References

- Aguiar, Alexandra de (2005) “Aplicação de Algoritmos Genéticos em Cluster Reconfigurável”. Universidade de Santa Cruz do Sul.
- Brakmo, L., O’Malley, S.W., Peteron L.L., (1994) “TCP Vegas: New Techniques for Congestion Detection and Avoidance”. In Proceedings of ACM SIGCOMM’94.
- Carro, Luigi (2001) “Sashimi: Manual do usuário” Universidade Federal do Rio Grande do Sul.
- Comer, Douglas (2000) “Internetworking with TCP/IP: principles, protocols, and architectures. 4^a ed. New Jersey: Prentice Hall.
- Darpa (1981) “Transmission Control protocol”, DARPA, RFC 793.
- Dehon, A. and Wawrynek, J. (1999) “reconfigurable Computing: What, Why, and Design Automation Requirements?” in proceedings of the 1999 Design Automation Conference, pp. 610-615.
- Fall, K., Floyd, S., (1996) “Simulation-based Comparisons of Tahoe, Reno and SACK TCP”, Computer Communication, vol 26.
- Freitas, Josué de (2004) “Integração em FPGA de um microcontrolador FentoJava com uma Pilha TCP/IP”. Universidade de Santa Cruz do Sul – Trabalho de Conclusão de Curso.

Rose, Cesar A. F. de and Navaux, Philippe O. A. (2003) "Arquiteturas Paralelas", Sagra Luzzatto, Brasil.

Sutton, Peter (2001) "VHDL XSV Board Interface Projects", <http://www.itee.uq.edu.au/~peters/xsvboard>, March.

Desenvolvimento de Agente SNMP¹ Embarcado para Microprocessador *Rabbit*

Elvis Lopes Monteiro^a, Westter José da Silva Santos^b, Dr. Cláudio Afonso Fleury^c

CEFET-GO – Centro Federal de Educação Tecnológica de Goiás

{elvislm^a, westter^b}@gmail.com, kaw^c@cefetgo.br

Resumo. Este artigo descreve a implementação de um agente SNMPv1 (*Simple Network Management Protocol Version 1*) para microprocessador *Rabbit Core Module 3200*, com aplicação no monitoramento e controle de dispositivos de acionamento elétrico em redes de Automação Industrial.

Abstract. This paper describes the implementation of an SNMPv1 agent (*Simple Management Protocol Version 1*) on a *Rabbit Core Module 3200 Microprocessor* to be applied on management and control of electrical activation devices in Industrial Automation Networks.

1. Introdução

O protocolo SNMP surgiu em 1988 e tornou-se de fato um padrão para gerenciamento de redes. É bastante difundido em dispositivos de redes de computadores (*hubs*, *switches*, *routers*). Através dele pode-se identificar e localizar diversos tipos de problemas, tanto na rede em si quanto nos elementos que a compõe, facilitando assim o trabalho dos gerenciadores de redes.

Os dispositivos de acionamento, como relés e contatores de potência, são largamente utilizados em automação industrial no acionamento de cargas elétricas como: motores, conjuntos de iluminação, sistemas de ar condicionado, e diversos outros equipamentos elétricos.

A motivação principal deste estudo é a utilização do protocolo SNMP em redes industriais de automação que possui dispositivos de acionamento elétrico, usando eletrônica embarcada, mais especificamente neste caso, o microprocessador *Rabbit* (agente SNMP embarcado).

2. Gerenciamento de Rede com SNMP

Numa rede gerenciada pelo SNMP pode-se distinguir os seguintes componentes:

2.1 *Gerente SNMP*: entidade que realiza consultas SNMP (*pollings*) nos objetos gerenciados contidos nos agentes, estes objetos possuem atributos definidos em uma base de informações gerenciais, a MIB. O gerenciamento é realizado através de operações SNMP de leitura (*GetRequest*, *GetNext*) ou de escrita (*SetRequest*) pela porta UDP 161, com o objetivo de monitorar os agentes SNMP, ou ainda receber informações (*traps*, relatórios gerados por eventos assíncronos) pela porta UDP 162.

2.2 *Agente SNMP*: entidade (programa executável) responsável por coletar os dados dos dispositivos gerenciados. Comunica-se com o gerente SNMP da rede informando os atributos dos objetos definidos na MIB, quando consultados, ou ainda gerando *traps* na

¹ SNMP – Simple Network Management Protocol – Protocolo Simples de Gerenciamento de Rede

hipótese de ocorrência de eventos pré-definidos, tais como anomalias, erros, falhas, etc, que exigem atenção imediata do gerente SNMP.

2.3. MIB (Management Information Base): base de dados conceitual onde são representados os elementos a serem gerenciados. Possui uma estrutura em árvore: o nó posicionado no início é denominado raiz (*root*). Todo nó que possuir filhos será denominado sub-árvore. E tudo que não tiver filhos será considerado nó de folha ou objeto. Um equívoco comum é a confusão entre MIB e arquivo de MIB. MIB, conforme orientação anterior é a definição conceitual dos recursos gerenciados. Arquivo de MIB refere-se às definições realizadas em um arquivo, formato texto, utilizando a linguagem de definição de sintaxe SMI (*Structure of Management Information*). A MIB deve existir tanto no gerente (arquivo de MIB compilado) como nos agentes (geralmente embutido na estrutura do código fonte).

2.4. Objetos SNMP: são os elementos que compõem a MIB. Estes objetos são na verdade variáveis, às quais estão associados: um tipo e um valor. Objetos se relacionam com as características das entidades físicas gerenciadas (hardware) ou com propriedades de determinado serviço ou aplicação (software).

As principais vantagens resultantes do uso do protocolo SNMP para a gerência de redes são:

- Monitoramento dos objetos gerenciados via *polling* (evento síncrono);
- Utilização de um protocolo padrão e aberto, regulamentado pelo IETF (RFCs);
- Protocolo bastante difundido e largamente utilizado;
- Interoperabilidade entre diversos dispositivos de vários fabricantes;
- Possibilidade de geração de eventos do tipo alarme (*traps*) que indicam quando algo de errado está acontecendo ou está para acontecer, e pode ser configurado para eventos específicos.

3. Metodologia

Adotou-se o estudo de caso como método de análise da viabilidade técnica da proposta, implementando-se um controle simples, que consiste no acionamento de um contator usado na ativação de uma carga elétrica de potência considerável. Após os testes, realizados em laboratório, dos circuitos e programas desenvolvidos, mostrou-se possível a implementação de um agente SNMP embarcado para microprocessador *Rabbit*, visando o gerenciamento de dispositivos interligados em uma rede de automação industrial.

4. Implementação do Estudo de Caso

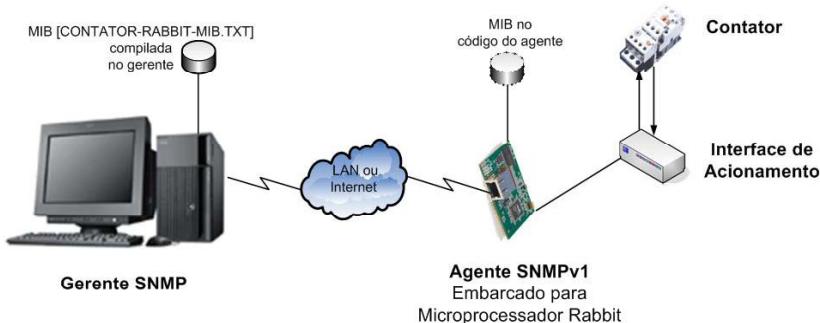


Figura 1 – Arquitetura do estudo de Caso - Interação entre Agente e Gerente SNMP

4.1 Características da implementação:

A seguir as características e funcionalidades dos recursos de *Hardware* e *Software*:

4.1.1 Recursos de *Software*:

a) *Aplicativo MG-Soft MIB Browser*: utilizado para visualização dos valores dos objetos da MIB, assim como para execução de operações de gerenciamento SNMP (*GetRequest*, *SetRequest*, etc).

b) *Dynamic C 8.1* [7]: compilador C para o desenvolvimento da aplicação Agente.

4.1.2 Recursos de *Hardware*:

a) *Kit de Desenvolvimento do Microprocessador Rabbit Core Module RCM3200* [7]: composto pelo microprocessador e sua respectiva placa de desenvolvimento. O microprocessador opera em frequência de 44.2 MHz, e possui memória Flash de 512 KB, memória SRAM de 256 KB, memória de código de 512 KB, 52 linhas de I/O², possibilidade de implementação de serviços TCP/IP (HTTP, POP3, SNMP, etc).

b) *Contator*: foi utilizado um dispositivo SIEMENS, modelo 3TF40 22-0A, para acionamento de carga elétrica.

c) *Circuito para acionamento e monitoramento do contator*: foi desenvolvido um circuito eletrônico para ligar o contator ao Rabbit. Com as seguintes funções: proteger o microprocessador através de optoacopladores³ e condicionar os sinais provenientes do microprocessador de forma que estes possam realizar o acionamento do contator (Figura 2), e ainda fazer o monitoramento de um dos contatos auxiliares do contator (Figura 3).

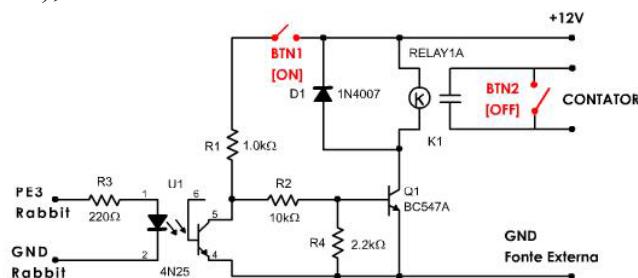


Figura 2 – Parte do circuito que efetua o acionamento do contator

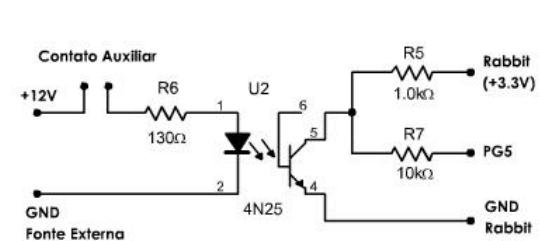


Figura 3 – Parte do circuito que efetua o monitoramento do contato auxiliar

4.2 Procedimentos para implementação:

a) *Criação da MIB de acordo com SMIVI*: Nesta fase foi necessário criar um modelo conceitual do gerenciamento proposto. Como a proposta era ser um modelo simples e didático, foram definidas duas ramificações, a saber: **controle** - responsável pelas características de controle; e **monitoramento** - responsável pelas características de monitoramento do contator. Dentro de cada uma destas entradas foram criados objetos, que gerenciam e controlam determinadas características. A Tabela 1 apresenta as funções associadas a cada objeto da MIB definida para o agente embarcado, assim

² I/O – Input/Output (Entrada/Saída)

³ Optoacopladores – Dispositivos que realizam acoplamento ótico isolando o circuito eletricamente.

como o OID (*Object Identification*) para cada um destes. O arquivo contendo a MIB pode ser visualizado em detalhes no site do projeto [8].

Tabela 1 – Objetos da MIB – Contator

Ramo	Objeto	OID	Descrição
Controle	contator0	1.3.6.1.4.1.12807.3.1.1.1.0	Liga ou desliga o contator0
Monitoramento	contFalha0	1.3.6.1.4.1.12807.3.1.2.1.0	Contagem de falhas do contator0
	contAciona0	1.3.6.1.4.1.12807.3.1.2.2.0	Contagem de acionamentos / desligamentos
	TempoFunc	1.3.6.1.4.1.12807.3.1.2.3.0	Tempo de funcionamento do agente SNMP

Na Figura 4 visualiza-se a estrutura da árvore de MIB contendo os objetos a serem gerenciados pelo agente SNMP embarcado.

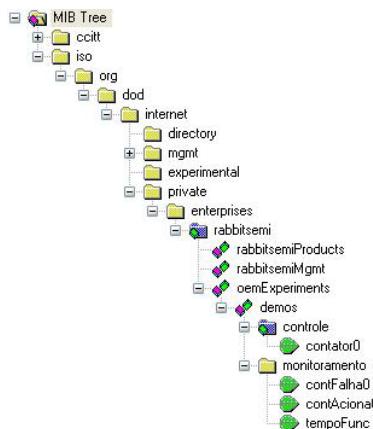


Figura 4 – Representação da Árvore de MIB do agente SNMP

- b) *Configuração do Ambiente de Desenvolvimento (Dynamic C 8.1)*: Mudanças nos arquivos de biblioteca (rcm3200.lib e tcp_config.lib) de maneira a configurar características específicas (configuração dos bits das portas paralelas, mascaramento e direcionamento de entrada ou saída, endereço IP, portas utilizadas, etc.).
- c) *Desenvolvimento do programa para o microprocessador Rabbit (código do agente SNMP embarcado)*: Escrita do código fonte em Linguagem C (para Dynamic C), onde foram especificadas as condições para o funcionamento do agente, com a utilização de funções do compilador para a implementação das operações SNMP e da MIB. O código fonte pode ser visualizado no site do desenvolvimento do projeto [8].
- d) *Desenvolvimento dos circuitos (interface) de acionamento/monitoramento*: O circuito de acionamento (Figura 2) permite o acionamento/desligamento da carga elétrica (ligada ao contator) tanto via operação SNMP (*SetRequest*), quanto via chave [sw1] na placa de desenvolvimento. Já o circuito de monitoramento (Figura 3) realiza o monitoramento do contato auxiliar do contator, verificando seu estado (aberto ou fechado). Quando aberto, deduz-se que, ou o contator está desligado, ou o contator está com problemas. Este estado é monitorado pelo código do agente, gerando-se uma *trap* sempre que este evento ocorrer.
- e) *Chaves de manutenção e de acionamento manual*: A chave de manutenção permite a manutenção do Rabbit e do circuito de acionamento/monitoramento do contator. Já a chave de acionamento manual (botoeira) efetua o acionamento ou desligamento da carga elétrica de forma manual, quando a chave de manutenção estiver ligada. A Figura 5 apresenta um diagrama de blocos da interligação do hardware que compõe este estudo.

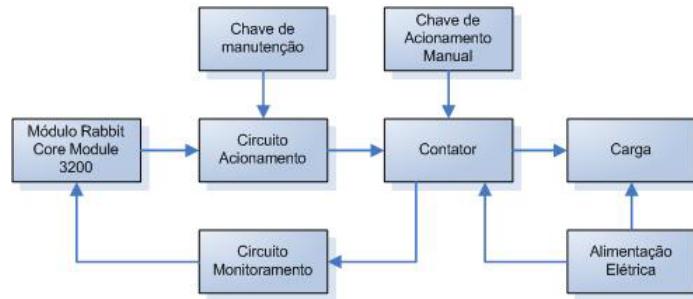


Figura 5 - Diagrama de Blocos da Interligação do Hardware

f) *Testes com agente SNMP:* visualização dos valores dos objetos com software MG-SOFT MIB Browser, e coleta de pacotes com o *Sniffer* (Analizador de Pacotes) Ethereal para visualização das mensagens SNMP e suas PDU's (*Protocol Data Unit*).

4.3. Funcionamento do Agente

Os objetos da MIB são definidos de maneira bastante intuitiva, de modo que ao se observar a árvore da MIB, tem-se uma clara percepção das funções dos objetos do agente SNMPv1 a serem monitorados. Este agente possui as seguintes funcionalidades:

- **Acionamento do contator** – acionamento através de comando SNMP [set] enviado pelo gerente SNMP da rede ou através de uma chave tipo *push-button* [sw1] localizado na placa do kit de desenvolvimento;
- **Monitoramento de algumas características do contator**
 - **Contador de Acionamento** – conta os acionamentos e desligamentos;
 - **Contador de Falhas** – conta as falhas detectadas pelo agente;
 - **Tempo de Funcionamento** – armazena o tempo de funcionamento do agente SNMPv1, desde o início das operações, ou desde o último *reset*;
 - **Checkagem de erros** – a cada falha ocorrida é gerado imediatamente um aviso assíncrono (*trap* tipo 20 – erro), e novos avisos continuarão a ser gerados a cada 30 segundos. No caso de restabelecimento do serviço, o envio destas *traps* será automaticamente interrompido, e partir disto será gerado um outro tipo de *trap* (*trap* tipo 30 – restabelecimento), informando o retorno do contator às suas operações.

5. Conclusões

A partir dos resultados obtidos nos experimentos descritos nesta investigação, observou-se uma solução para o monitoramento de dispositivos de acionamento elétrico via protocolo SNMP, que pode ser utilizada em automação industrial para acionamento de cargas como motores ou similares.

Os trabalhos relacionados com desenvolvimento de agentes SNMP são escassos, principalmente no que tange ao SNMP aplicado ao *Rabbit*. Mas dentre a literatura pesquisada destacam-se: as referências [4] e [6], que divulgam soluções de extensibilidade de agentes SNMP utilizando o NET-SNMP, a referência [5] aborda o desenvolvimento de agente SNMP embarcado, similar ao desenvolvido neste trabalho, mas com diferenças: a) No artigo em referência foi utilizada a plataforma TINI (*Tiny Internet Interface*); b) Não suporta SNMP; c) Não suporta o envio de notificações; d) execução de uma máquina virtual Java.

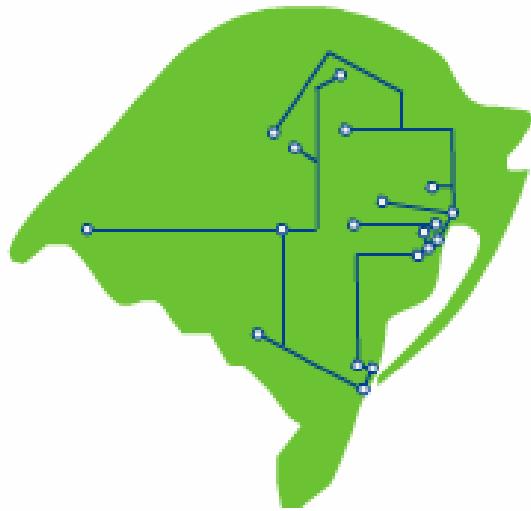
Dentre as vantagens da implementação de monitoramento através do SNMP, destacam-se: o acompanhamento on-line de algumas características pré-determinadas do sistema gerenciado, que reduz o tempo médio entre falhas, o caráter pró-ativo à manutenção do sistema, através de verificações síncronas (*polling*) e assíncronas (*traps*), que reduz prejuízos oriundos da demora na detecção de problemas.

O microprocessador *Rabbit* pode ser uma alternativa interessante quando se pretende realizar uma implementação de qualidade com custos reduzidos. O custo final é cerca de 10 vezes menor que o de uma estação de trabalho convencional (tipo PC), com a vantagem de redução da manutenção, visto que trata-se de um sistema muito mais simplificado, não tendo a necessidade de instalação e configuração de sistema operacional, que poderia ainda estar sujeito a ataques às fragilidades do sistema, como por exemplo vírus de computador ou falha de algum componente de software.

Embora este projeto tenha implementado o SNMPv1, ressalta-se que esta versão foi adotada devido à indisponibilidade de versões mais atualizadas para o módulo SNMP da biblioteca existente no *Dynamic C* v8.1. Recomenda-se que em desenvolvimentos de projetos para chão de fábricas, o protocolo SNMPv1 deva ser utilizado somente para o monitoramento, mas nunca para o controle de dispositivos. Isto deve-se à fragilidade do SNMPv1 em relação ao quesito segurança. Para situações de controle real indica-se o uso do SNMPv3, que incorpora conceitos de segurança, com protocolos de criptografia (DES, MD5 e SHA-1) e protocolos mais sofisticados de autenticação de mensagem como o HMAC.

Bibliografia

- [1] MAURO, Douglas; Devin J. Schimidt; **SNMP Essencial** – Tradução de Tereza Cristina Feliz de Souza – Rio de Janeiro: Campus, 2003.
- [2] STALLINGS, William. **SNMP, SNMPV2, SNMPV3, and 1 RMON and 1 and 2. 3.** ed. Massachusetts: Addison Wesley, 1999.
- [3] SUBRAMANIAN, Mani, - **NETWORK MANAGEMENT**, - Principles and Practice, Addison Wesley, 2000.
- [4] ZAMBENEDETTI, Lisandro. **Construção de Agente SNMP em Ambiente Linux**, www.inf.ufrrgs.br/granville/AgentesSNMP/FevMar-2002 (acessado em 04/2005).
- [5] NACAMURA, Luiz. GARCIA, Osvaldo. FERRASA, Adriano. AUGUSTO, Rômulo. **Desenvolvimento de um agente SNMP embarcado para o gerenciamento de carga em redes de distribuição elétrica**, XXX Seminário integrado de Hardware e Software.
- [6] KADIONIK, Patrice - **USING NET-SNMP UNDER LINUX AND µCLINUX - INTRODUCTION TO NETWORK MANAGEMENT**, www.enseirb.fr/~kadionik/embedded/snmp/english/NET- NMP_english.html (acessado em 04/2005).
- [7] Documentação referente ao *Rabbit Core Module RCM3200* e *Dynamic C RCM3200*, www.rabbitsemiconductor.com/products/rcm3200/index.shtml (acessado em 04/2005).
- [8] http://www.redes.cefetgo.br/gl_artigos/001/index.htm - site desenvolvido pelos integrantes do projeto.



MINICURSOS

MINICURSO 1

Título: Dotando serviços de segurança em Windows e Linux

Ministrantes: Prof. Dr. Vinicius G. Ribeiro e Fernando Patzlaff

Instituição: Centro Universitário La Salle - UNILASALLE

Resumo:

As redes de computadores possibilitaram grandes avanços, em termos de compartilhamento de recursos, facilidades de comunicação etc. Fabricantes diferentes inicialmente buscaram soluções diferentes. No mercado brasileiro, são comuns o emprego de duas redes locais: Windows e Linux. Cada um tem um histórico de evolução diferente, dados os seus diferentes objetivos. Considerando-se os serviços de segurança - privacidade, autenticação, acesso, integridade, disponibilidade e não-repúdio -, há alternativas que possibilitam dotar tais redes de alguns desses serviços, no intuito de minimizar a incerteza em termos de segurança. O presente mini-curso apresentará quais são os recursos e programas que podem ser empregados para tal objetivo.

MINICURSO 2

Título: VoIP - Transmissão de Voz em Redes IP

Ministrante: Prof. Dr. João Netto

Instituição: Universidade Federal do Rio Grande do Sul - UFRGS

Resumo:

O Mini-curso trata dos protocolos e estrutura da transmissão de voz em redes IP. Aborda os protocolos de sinalização H323 e SIP. Será feita uma demonstração com alguns equipamentos comerciais de VoIP.

MINICURSO 3

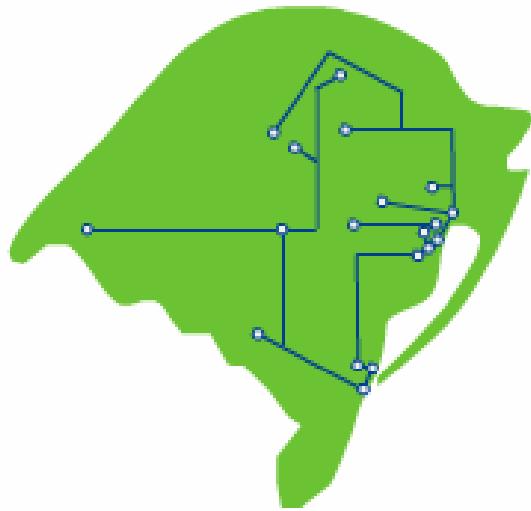
Título: Redes Sem-fio e Sistemas Celulares

Ministrante: Prof. Dr. Juergen Rochol

Instituição: Universidade Federal do Rio Grande do Sul - UFRGS

Resumo:

O Mini-curso está dividido em duas partes. Na primeira parte serão abordadas, de forma abrangente, as características das tecnologias de redes sem fio propostas pelas diferentes extensões da norma IEEE 802.11. Serão abordadas também as novas propostas da área de PAN (*Personal Area Networks*), como a norma IEEE 802.15 e as propostas de acesso de banda larga sem fio (BWA) sugerida pela norma IEEE 802.16. Também será analisada a proposta para redes MAN sem fio apresentada recentemente através da norma IEEE 802.20. Na segunda parte do Mini-Curso serão apresentadas as diferentes tecnologias que estão sendo discutidas em relação as últimas gerações de sistemas celulares. Serão mostradas algumas tendências em relação a sistemas de terceira e quarta geração referente, principalmente às suas características técnicas e aplicações. Também serão feitas algumas considerações sobre a evolução deste segmento das telecomunicações no Brasil.



OFICINAS

OFICINA 1

Título: Segurança em ambientes Linux

Ministrantes: Prof. Dr. Vinicius G. Ribeiro e Fernando Patzlaff

Instituição: Centro Universitário La Salle - UNILASALLE

Resumo:

O kernel 2.6 do Linux trouxe melhorias consideráveis, considerando-se os serviços básicos de segurança. O emprego de ferramentas como iptables não constitui novidade: há diversas aplicações que possibilitam interfaceamento com usuários que detém pouco conhecimento sobre Linux. Uma grande conquista é o ebtables: atuando sobre o nível 2 do modelo OSI, tal ferramenta possibilita melhor nível de segurança - no que tange a privacidade. A presente oficina permitirá que o acadêmico efetue a correta compilação e configuração, vindo a dotar mais esse serviço. Destaca-se que o emprego de ebtables não impede o emprego de iptables - como será ministrado. Sugere-se aos alunos, após a realização da oficina, que se verifique o impacto (ou não) do emprego do ebtables nas redes locais em que trabalham.

OFICINA 2

Título: Analise de segurança em uma rede wireless

Ministrantes: Prof. Evandro Franzen e Roberto Ely Scherer

Instituição: Universidade de Santa Cruz do Sul - UNISC

Resumo:

A mobilidade e a flexibilidade propiciada pelas redes wireless são características que estão tornando essa tecnologia muito difundida no mercado atualmente. Contudo, a busca pela segurança é fundamental para a utilização das vantagens que a rede wireless proporciona. A presente oficina analisará diferentes ferramentas de intrusão para detectar as deficiências dessa tecnologia e as principais soluções para melhorar a segurança nessas redes.

OFICINA 3

Título: Programando Web Services para Gerenciamento de Redes

Ministrante: Prof. Dr. Lisandro Zambenedetti Granville, Clarissa Cassales Marquezan, Ricardo Lemos Vianna, Weldson Q. de Lima

Instituição: Universidade Federal do Rio Grande do Sul - UFRGS

Resumo:

Os Web Services possuem aplicações diversas, e uma delas pode ser o gerenciamento de redes. Nesta oficina os participantes serão apresentados aos protocolos de gerenciamento de redes convencionais, ao protocolo SOAP (que dá suporte aos Web Services), e às soluções para gerenciamento utilizando Web Services. A seguir, um conjunto de Web Services será desenvolvido durante a oficina, assim como serão desenvolvidos os clientes (ou gerentes) que irão acessar tais Web Services em busca de informações de gerenciamento nos dispositivos de interesse.

Observação: Espera-se que os participantes tenham noções básicas de programação em computadores (não é necessário conhecimento em programação para redes).