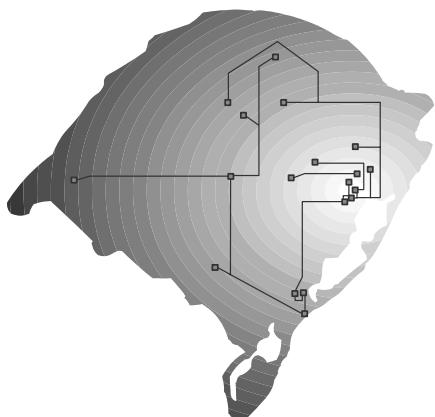


2^a Escola Regional de Redes de Computadores – ERRC 2004

**Canoas, RS - Brasil
12, 13 e 14 de julho de 2004**



Anais

Editores

*Cristina Moreira Nunes
Lisandro Zambenedetti Granville
Juergen Rochol*

Organização

Centro Universitário La Salle (UNILASALLE)
Universidade Federal do Rio Grande do Sul (UFRGS)

Colaboração

Universidade do Vale do Rio dos Sinos (UNISINOS)
Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS)
Universidade de Caxias do Sul (UCS)

Promoção

Sociedade Brasileira de Computação, Secretaria do Rio Grande do Sul (SBC-RS)
Laboratório Nacional de Redes de Computadores (LARC)

Patrocínio

Extreme Networks

Tiragem
100 exemplares

2^a Escola Regional de Redes de Computadores – ERRC 2004
(2. : 2003 12 a 14 de julho: Canoas, RS)

Anais / 2^a Escola Regional de Redes de Computadores – ERRC 2004; editores Cristina Moreira Nunes, Lisandro Zambenedetti Granville e Juergen Rochol, Porto Alegre, 12 a 14 de julho de 2004. Canoas : Sociedade Brasileira de Computação, Laboratório Nacional de Redes de Computadores, 2004.

x, 236p. ; 21 cm

1. Redes 2. Redes de Computadores 3. Programa I. Título II. Nunes, Cristina Moreira. III. Granville, Lisandro Zambenedetti. IV. Rochol, Juergen. IV.

Prefácio

Estes são os anais da segunda edição da **Escola Regional de Redes de Computadores** (ERRC) que é um evento resultante do esforço conjunto do Centro Universitário La Salle (UNILASALLE), da Universidade Federal do Rio Grande do Sul (UFRGS), da Universidade do Vale do Rio dos Sinos (UNISINOS), da Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS) e da Universidade de Caxias do Sul (UCS). Entre 12 a 14 de julho de 2004 diversas atividades serão desenvolvidas nas dependências do UNILASALLE, em Canoas. Esta 2^a edição da Escola é composta por mini-cursos, sessões técnicas, oficinas e palestras. O objetivo principal da Escola é permitir o debate e a discussão das atividades relacionadas à área de redes de computadores no Rio Grande do Sul. O público alvo da escola é formado por professores, profissionais e alunos de graduação e pós-graduação em redes.

Neste ano, 3 mini-cursos, 3 palestras e 3 oficinas serão apresentados aos participantes. Além disso, 12 sessões técnicas comportarão a apresentação de 36 artigos, um número expressivo em relação à primeira edição da Escola, que continha 24 artigos. Esta expansão no número de artigos é o resultado da grande submissão de trabalhos mas também do esforço da organização em propiciar a um número maior de estudantes a oportunidade de apresentar seus artigos à comunidade da Escola.

Acreditamos que esta segunda edição vem afirma a importância da ERRC para a comunidade de redes e áreas afins no sul do Brasil, e esperamos que os participantes possam aproveitá-la para discutir, trocar experiências e tecer parcerias. Esta segunda edição só foi possível graças à colaboração de várias pessoas. Gostaríamos de agradecer aos membros do comitê de programa e de mini-cursos que tiveram como tarefa a seleção de artigos das sessões técnicas e dos mini-cursos. Gostaríamos também de agradecer aos coordenadores das atividades da ERRC: Tasso Gomes de Faria (coordenador de mini-cursos), Ana Cristina Benso da Silva e Edgar Meneghetti (coordenadores de oficinas) e Luciano Paschoal Gaspary (coordenador de palestras). Não poderíamos de deixar de agradecer também aos alunos de graduação e pós-graduação que trabalharam incansavelmente para que a ERRC fosse possível: a todos nosso muito obrigado. Agradecemos também o importante apoio dado pela SBC, através de sua secretaria do Rio Grande do Sul, pelo Laboratório Nacional de Redes de Computadores (LARC), e pelo patrocínio da Extreme Networks. Por fim, desejamos a todos os participantes um ótimo e proveitoso evento.

Cristina Moreira Nunes, UNILASALLE/PUC-RS
Juergen Rochol, UFRGS
Coordenadores Gerais da ERRC

Lisandro Zambenedetti Granville, UFRGS
Coordenador do Comitê de Programa da ERRC

Canoas, julho de 2004

Coordenação

Coordenação Geral

Cristina Moreira Nunes, UNILASALLE/PUC-RS
Juergen Rochol, UFRGS

Coordenação do Comitê de Programa

Lisandro Zambenedetti Granville, UFRGS

Coordenação de Mini-cursos

Tasso Gomes de Faria, UNILASALLE/PUC-RS

Coordenação de Oficinas

Ana Cristina Benso da Silva, PUC-RS
Edgar Meneghetti, UCS

Coordenação de Palestras

Luciano Paschoal Gaspary, UNISINOS

Comitê de Programa

Alexandre da Silva Carissimi, UFRGS
Ana Cristina Benso da Silva, PUC-RS
Avelino Zorzo, PUC-RS
Benhur Stein, UFSM
Carlos Adriani Lara Schaeffer, UPF
Cláudio Geyer, UFRGS
Cristian Koliver, UCS
Cristiano Bonato Both, UNISC
Cristina Melchiors, UNISINOS
Cristina Moreira Nunes, UNILASALLE, PUC-RS
Edgar Meneghetti, UCS
Fábio Zanin, URI
Fernando Luis Dotti, PUC-RS
Gaspare Bruno, UNILASALLE
Gerson Battisti, UNIJUI
Gerson Cavalheiro, UNISINOS
Ingrid Jansch-Pôrto, UFRGS
João Cesar Netto, UFRGS
Jorge Barbosa, UNISINOS
Juergen Rochol, UFRGS
Katia Saikoski
Liane Margarida Rockenbach Tarouco, UFRGS
Lisandro Zambenedetti Granville, UFRGS
Luciano Paschoal Gaspary, UNISINOS
Marco Antônio Trentin, UPF
Marcos Barreto, UNILASALLE
Maria Janilce Bosquiroli Almeida, UFRGS
Marinho Pilla Barcelos, UNISINOS
Nelson Duarte Filho, FURG
Raul Weber, UFRGS
Ricardo Neisse, UFRGS

Taisy Weber, UFRGS
Tasso Gomes de Faria, UNILASALLE, PUC-RS
Valter Roesler, UNISINOS
Vinícius Ribeiro, UNILASALLE

Comitê de Avaliação de Mini-cursos

Gerson Battisti, UNIJUI
Marcelo Azambuja, FACCAT, ULBRA
Ricardo Balbinot, UNILASALLE, PUC-RS

Comitê de Organização

Gaspare Bruno, UNILASALLE
Laura Lopes, UFRGS
Liane Margarida Rockenbach Tarouco, UFRGS
Marcos Ennes Barreto, UNILASALLE
Ricardo Neisse, UFRGS
Tiago Fioreze, UFRGS
Valter Roesler, UNISINOS

Revisores

Alexandre da Silva Carissimi, UFRGS
Ana Cristina Benso da Silva, PUC-RS
André Detsch, UNISINOS
Avelino Zorzo, PUC-RS
Benhur Stein, UFSM
Carlos Schaeffer, UPF
Clarissa Marquezan, UFRGS
Claudio Geyer, UFRGS
Cristian Koliver, UCS
Cristiano Both, UNISC
Cristina Melchiors, UNISINOS
Cristina Moreira Nunes, UNILASALLE/PUC-RS
Diego Kreutz, UFSM
Edgar Meneghetti, UCS
Elgio Schlemer, UFRGS
Evandro Della Vecchia Pereira, UFRGS
Fabio Zanin, URI
Felipe Vilanova, UFRGS
Fernando Dotti, PUC-RS
Gabriela Jacques-Silva, UFRGS
Gaspare Bruno, UNILASALLE
Gerson Battisti, UNIJUI
Gerson Cavalheiro, UNISINOS
Ingrid Jansch-Pôrto, UFRGS
João Netto, UFRGS
Jorge Barbosa, UNISINOS
Juergen Rochol, UFRGS
Karina Kohl, UFRGS

Katia Saikoski
Liane Margarida Rockenbach Tarouco, UFRGS
Lisandro Zambenedetti Granville, UFRGS
Luciano Gaspary, UNISINOS
Marco Antônio Trentin, UPF
Marcos Barreto, UNILASALLE
Marinho Barcellos, UNISINOS/University of Manchester
Michelle Leonhardt, UFRGS
Nelson Duarte Filho, FURG
Raul Weber, UFRGS
Ricardo Neisse, UFRGS
Ricardo Vianna, UFRGS
Roberto Jung Drebes, UFRGS
Rodrigo Alves, UFRGS
Taisy Weber, UFRGS
Tasso Faria, PUC-RS
Tórgan Siqueira, UFRGS
Tiago Fioreze, UFRGS
Valter Roesler, UNISINOS
Vinicius Ribeiro, UNILASALLE

Sumário

Sessão Técnica 1: Gerência e Operação de Redes I

PTTs Brasileiros: Funcionamento e Panorama

Emerson Virti, Leandro Bertholdo, Mell Fogliatto, Liane Tarouco (POP-RS) 3

Administração de Sistemas de Autenticação de Usuários

Diego Kreutz (UFSM) 9

KDD-NetManager: uma Ferramenta de Descoberta de Conhecimento em Bases de Dados (DCBD) Aplicada à Gerência Proativa em Redes de Comunicação

Flávia Carvalho Marcelo Azambuja (FACCAT/PUC-RS),
Jorge Guedes (PUC-RS) 15

Sessão Técnica 2: Tolerância a Falhas

Implementação de um Injetor de Falhas de Comunicação para Aplicações Java Baseado em JVMTI

Júlio Gerchman, Gabriela Jacques-Silva, Taisy Weber (UFRGS) 23

Injeção de Falhas de Comunicação por Sondagem Dinâmica

Roberto Jung Drebes, Taisy Weber (UFRGS) 29

Uma Arquitetura de Injetor de Falhas Orientada a Aspectos para Validação de Sistemas de Comunicação de Grupo

Karina Kohl, Taisy Weber (UFRGS) 35

Sessão Técnica 3: Gerência e Operações de Redes II

Uma proposta de extensão de um chatterbot visando auxiliar na capacitação de profissionais de gerência de redes

Michelle Leonhardt (UFRGS), Leandro Bertholdo (POP-RS),
Liane Tarouco (UFRGS) 43

Uma Arquitetura para Correlação de Alarmes Baseada em Políticas e Web Services

Evandro Della Vecchia Pereira, Lisandro Zambenedetti Granville (UFRGS) 49

Um Sistema Integrado para a Administração de Computadores e Serviços de Redes Locais

Diego Kreutz, Benhur Stein (UFSM) 55

Sessão Técnica 4: Tolerância a Falhas II

Máquinas Virtuais como Suporte ao Desenvolvimento de Aplicações Tolerantes a Falhas

Tórgan Siqueira, Taisy Weber (UFRGS), Reynaldo Novaes (HP Brazil),
Ingrid Jansch-Pôrto (UFRGS) 63

Modelagem de Falhas para Simulação de Sistemas Distribuídos

Ruthiano Munaretti, Marinho Barcellos (UNISINOS) 69

fiji - Uma ferramenta para validação experimental do XFS

Leonardo Mello, Taisy Weber (UFRGS) 75

Sessão Técnica 5: Protocolos e Serviços

<i>Uma Visão Geral das Soluções para Uso de IPv6</i>	
Andrey Vedana Andreoli, Liane Tarouco (UFRGS)	83
<i>Análise Comparativa entre NIS e LDAP</i>	
Augusto Bueno, Alexandre Carissimi (UFRGS)	89
<i>Técnicas de combate a spam em servidores Linux que utilizam o Mail Transfer Agent Postfix</i>	
Fabio Dias, Cristina Nunes (UNILASALLE).....	95

Sessão Técnica 6: Segurança

<i>Segurança em Redes Ópticas: Tipos de Ataques e Métodos</i>	
André Panisson, Ricardo Vianna, Rodrigo Alves, Juergen Rochol (UFRGS)	103
<i>Aplicando a Técnica de Raciocínio Baseado em Casos na Identificação de Cenários de Intrusão em Logs de Firewalls</i>	
Samir Lohmann, Cristina Melchiori, Luciano Gaspary (UNISINOS)	109
<i>Avaliação de Geradores de Números Pseudo-Aleatórios</i>	
Vera Lúcia Silva, Juliano Monte-Mor, Nei Yoshihiro Soma (ITA)	115

Sessão Técnica 7: Comunicação sem Fio I

<i>Qualidade de Serviço em redes sem fios</i>	
Celso Pelliccioli (UNILASALLE), Cristina Nunes (UNILASALLE/PUC-RS)	123
<i>Servidores Móveis em Redes Ad hoc</i>	
Fabricia Faria, Vitório Mazzola, Mario Dantas (UFSC)	129

Sessão Técnica 8: Roteamento

<i>SIM Gateway: roteador para redes sem fios baseado em FreeBSD</i>	
Carlos Rolim (SIM Telecom), Carlos Santos (URI).....	137
<i>Roteamento IP em redes ópticas</i>	
Bruno Silva, Juergen Rochol, Lara Dalto de Souza, Vandersilvio da Silva (UFRGS)	143
<i>Analise Comparativa de Protocolos de Roteamento Redes Móveis Ad Hoc</i>	
Underlea Corrêa, Vitório Mazzola, Mario Dantas (UFSC)	149

Sessão Técnica 9: Estudos de Caso

<i>Relatório Técnico da Transmissão do SBRC</i>	
João Marcelo Ceron, Pablo Larina Rodrigues, Maiko de Andrade, Valter Roesler (UNISINOS)	159
<i>Implementação de um Filtro Adaptativo LMS Aplicado ao Cancelamento de Eco em Voz sobre IP</i>	
Guilherme Corsetti, Moisés Coster, Ricardo Balbinot, Fladhimyr Castello, Jorge Guedes (PUC-RS)	165
<i>Uma Ferramenta para Comparaçao dos Mecanismos de Controle de Congestionamento em Redes TCP/IP Utilizando o NS</i>	
Luiz Rodrigues, Michele Lima (UNIOESTE)	171

Sessão Técnica 10: Redes P2P e Grid I

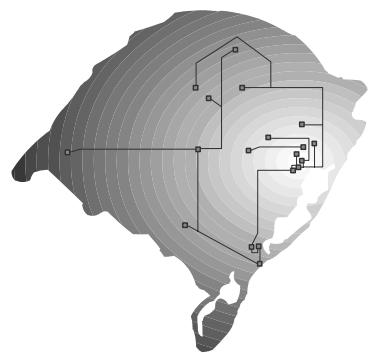
<i>Escambo: um Modelo de Comportamento e Reputação para Sistemas Peer-to-Peer</i>	
Rafael Righi, Felipe Rolim Pellissari, Carla Westphall (UFSC)	179
<i>Controlando P2P</i>	
Mell Fogliatto, Emerson Virti, Leandro Bertholdo (POP-RS), Liane Tarouco (UFRGS).....	185
<i>Redes Peer-to-Peer como Ferramenta para o Gerenciamento Cooperativo de Redes</i>	
Diego Rosa (UFRGS)	191

Sessão Técnica 11: Comunicação sem Fio II

<i>Propostas de Pesquisa de Segurança em Redes Sem Fio</i>	
André Peres (ULBRA), Raul Weber (UFRGS).....	199
<i>Os novos padrões de segurança em redes wireless</i>	
Afonso Araujo Neto, Bruno Silva (UFRGS)	205
<i>Secure Roaming Wireless: Uma Abordagem de Segurança para Redes Locais 802.11 com Criptografia Forte</i>	
Marcos Jose Sarres (Aker Security Solutions), Mario Dantas (UFSC)	211

Sessão Técnica 12: Redes P2P e Grid II

<i>Obtenção de Tolerância a Falhas na Ferramenta de computação em grid MyGrid</i>	
Hélio Silva, Carolina Ming Chiao (UFRGS)	219
<i>Estudo da Tecnologia P2P Visando sua Aplicação em Sistemas Baseados em Agentes</i>	
Elder Santos, Jose Fernando Machado, Jr. (UFRGS)	225
<i>Uma Ferramenta para Execução de Simulações em Larga Escala</i>	
André Detsch, Gerson Cavalheiro, Luciano Gaspari (UNISINOS).....	231



Sessão Técnica 1

Gerência e Operações de Redes I

PTTs Brasileiros: Funcionamento e Panorama

Emerson Virti, Leandro Bertholdo, Mell Fogliatto, Liane Tarouco
CERT-RS / POP-RS
{emerson, leandro, mell, liane}@penta.ufrgs.br

Resumo. Este trabalho apresenta um panorama da situação de cada um dos cinco pontos de troca de tráfego (PTT) brasileiros. No intuito de verificar os princípios de funcionamento de um PTT, realizou-se um pequeno esboço do roteamento na Internet bem como a função do protocolo BGP (Border Gateway Protocol) no roteamento de pacotes entre sistemas autônomos (AS).

1. Introdução

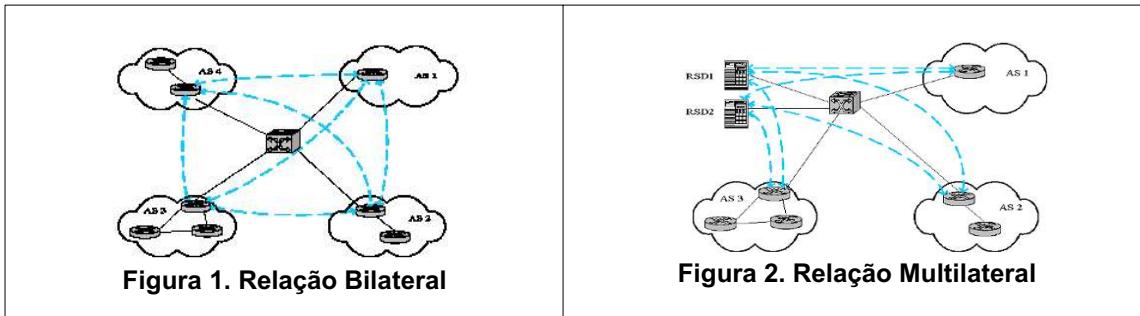
A topologia da Internet em seus primórdios poderia ser descrita como um conjunto de roteadores básicos, conhecidos também por roteadores de borda, onde a esses se conectavam outros roteadores, chamados de secundários. Pela grande difusão da Internet, esse modelo denominado hierárquico, foi substituído por um modelo distribuído [Gestor 2004]. Nesse contexto surge o conceito de sistemas autônomos (AS), onde sob um mesmo AS abrigava-se um grupo de roteadores que se comunicavam usando o mesmo protocolo e trabalhavam sob uma mesma gerência de redes [Bassam 1997]. A Internet global passou a ser vista como um conjunto de sistemas autônomos [Tanenbaum 1997]. Necessitava-se, agora, criar um protocolo de roteamento inter-ASes que, além de proporcionar a troca de pacotes entre esses sistemas autônomos, permitisse às suas gerências aplicarem “políticas de roteamento”, onde pudessem definir, por exemplo, que não aceitariam que pacotes vindos de um determinado sistema autônomo trafegassem em suas redes. Com tal finalidade surge o Border Gateway Protocol (BGP) [RFC 1771].

O aparecimento de grandes redes públicas e comerciais suscitou a criação dos chamados Pontos de Troca de Tráfego (PTT). Com o aumento no volume do tráfego de dados, a necessidade de implantação dos PTTs era engrandecida devido aos custos relativos ao transporte desses dados por conexões de longa distância bem como devido à necessidade de minimizar o tempo de acesso nas comunicações.

No Brasil, a medida em que o volume do tráfego de dados crescia, diversos PTTs foram surgindo. Atualmente, o país dispõe de cinco pontos de troca de tráfego, dos quais três são gerenciados por instituições públicas. Neste trabalho, através da exposição de dados coletados junto às gerências dos PTTs brasileiros no mês de março de 2004, abordar-se-á um comparativo entre esses PTTs nacionais.

2. Pontos de Troca de Tráfego (PTT)

Os pontos de troca de tráfego são formados basicamente por um switch que atua como comutador, interligando roteadores de diferentes sistemas autônomos com o intuito de trocar tráfego. Quando as sessões BGP são estabelecidas diretamente entre os roteadores dos ASes tem-se a chamada relação bilateral (figura 1). Por outro lado, quando esses roteadores estabelecem seções BGP com um servidor de rotas e este se encarrega de divulgar os prefixos aprendidos a todos os participantes (figura 2), tem-se um Acordo de Troca de Tráfego Multilateral (ATM).



3. Características dos PTTs Brasileiros

Através da coleta de dados junto aos pontos de troca de tráfego do país, procurou-se tecer um esboço da situação de cada um relativo aos seus números de participantes, localizações, volumes de tráfego, número de prefixos exportados e crescimento anual. A identificação e as principais características de cada um dos cinco PTTs brasileiros são abaixo apresentadas.

FIX: Localizado em Brasília (DF), é o Ponto Federal de Interconexão de Redes. Este PTT propicia a troca de tráfego a quatro ASes. Tem seu volume de tráfego com média de 8Mbps e máximo de 15Mbps.

PTT-SP: Contando com 27 participantes, é o maior PTT do país. Está localizado em São Paulo – capital e possui um volume médio de fluxo de 600Mbps, chegando a atingir 1100Mbps. Atualmente é gerenciado por uma empresa privada.

PRIX: Localizado em Curitiba (PR), conta hoje com 7 participantes. Tem 15Mbps de volume de tráfego médio, alcançando o máximo de 30Mbps.

RSIX: Com um tráfego médio de 100Mbps, chegando a 160Mbps de pico, o PTT-RS conta hoje com 15 participantes e está localizado em Porto Alegre (RS).

OPTIX: Também situado em São Paulo, o OptiGlobe Internet Exchange opera com tráfego médio de 70Mbps, chegando a operar com 150Mbps. É gerenciado por uma empresa privada. Por razões comerciais, o número de participantes não foi fornecido.

Como forma de verificar o contexto de cada ponto de troca de tráfego do país, é importante avaliar o volume de troca de tráfego em cada um. Os gráficos demonstrados na figura 3, foram confeccionados pelas gerências de cada PTT e apresentam o histórico dos tráfegos nos PTTs brasileiros.

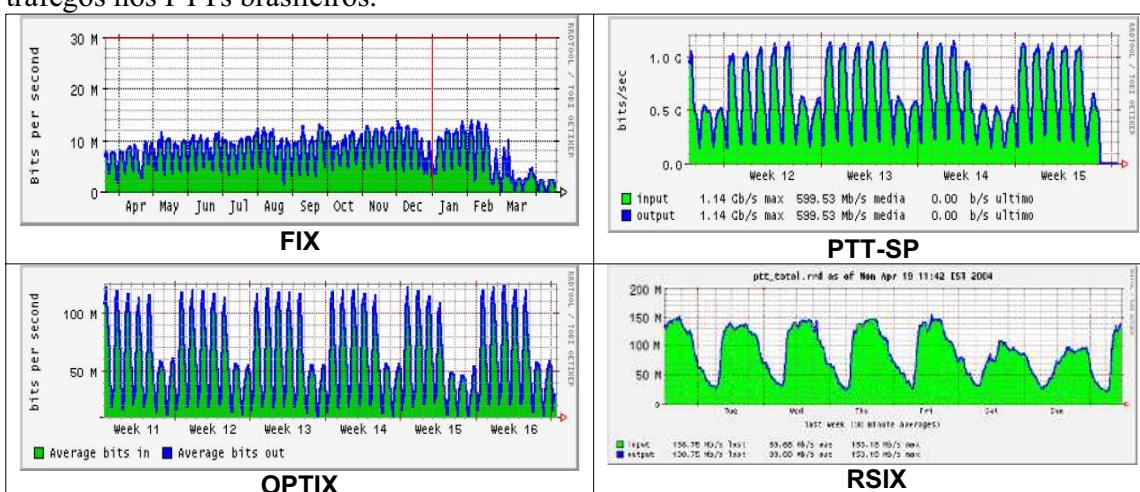


Figura 3. Demonstrativo dos volumes de tráfego nos PTTs – Dados de Março de 2004. O PRIX não forneceu esses dados.

Com a mensuração do volume total de tráfego médio nos PTTs foi possível tecer um demonstrativo (figura 4) que possibilita comparar seus tráfegos totais.

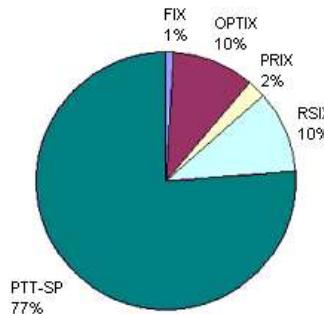


Figura 4. Comparativo entre os volumes médios de tráfego no mês de Março de 2004 nos PTTs brasileiros.

Observando o histórico do volume de tráfego nos PTTs nacionais verificou-se um crescimento superior a 30% em todos eles. Há casos como no PTT-RS, por exemplo, em que em um ano houve um crescimento de quase 400% em seu volume de tráfego. A figura 5 apresenta a evolução no volume de tráfego do RSIX.

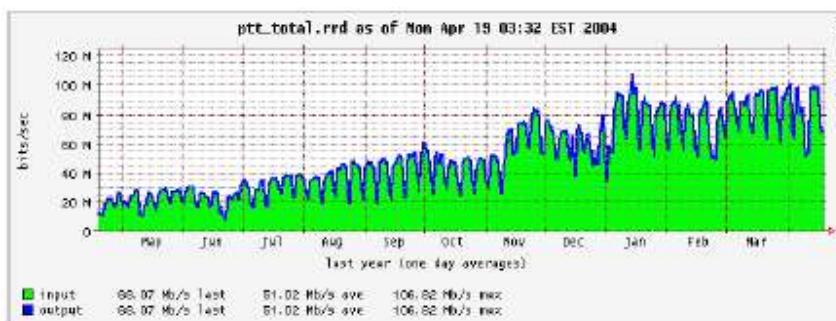


Figura 5. Evolução no Volume de Tráfego no RSIX – Dados do final de Março de 2004.

Pela coleta de dados com os responsáveis pelos PTTs, obteve-se um demonstrativo que apresenta o número de prefixos exportados por cada ponto de troca de tráfego (figura 6). Esses prefixos são os endereços dos blocos IP anunciados em cada PTT.

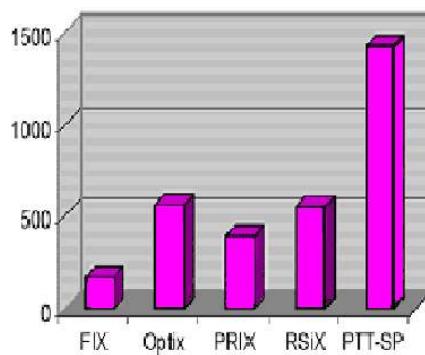


Figura 6. Número de Prefixos Exportados.

A observação da conectividade através dos PTTs brasileiros permite termos um esboço da situação da malha BGP no país. Através da relação de ASes conectados aos PTTs nacionais, pode-se construir o gráfico que abaixo é apresentado.

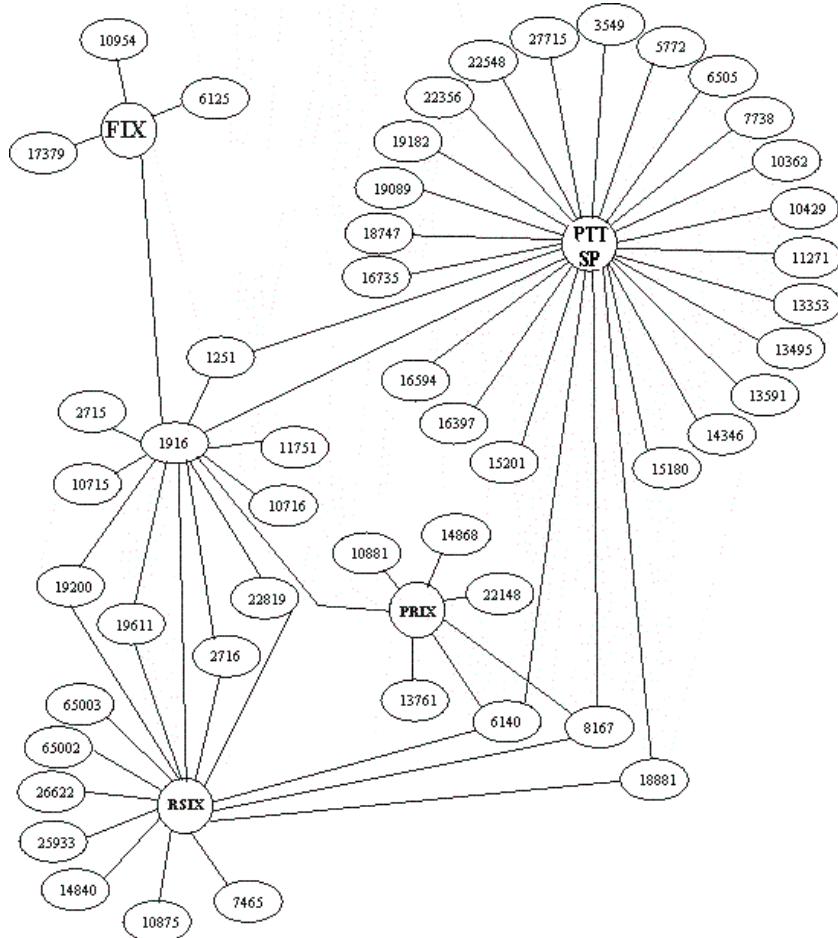


Figura 7. Conectividade entre os PTTs do país

OBS: o OPTIX não pode integrar a figura 7 por não ter fornecido os ASes a ele conectados.

Por motivos estratégicos, o tráfego entregue a um PTT por uma instituição participante é encarado como sendo pagamento ao mesmo obtido nesta troca. É desejável, portanto, que a relação entre o volume de dados fornecido e volume recebido seja próxima de 1 (um). Tomando a Embratel como exemplo, em 2002, para essa participar de um PTT, era necessário, entre outros pré-requisitos, que a referida taxa fosse de, no máximo, 1,5. Nesse contexto, os números apresentados pelo RSIX são satisfatórios. Nele, as taxas de troca de tráfego variam entre 0,8 e 1,2. Abaixo segue um demonstrativo das participações dos integrantes do RSIX quanto ao tipo de troca de tráfego que realizam.

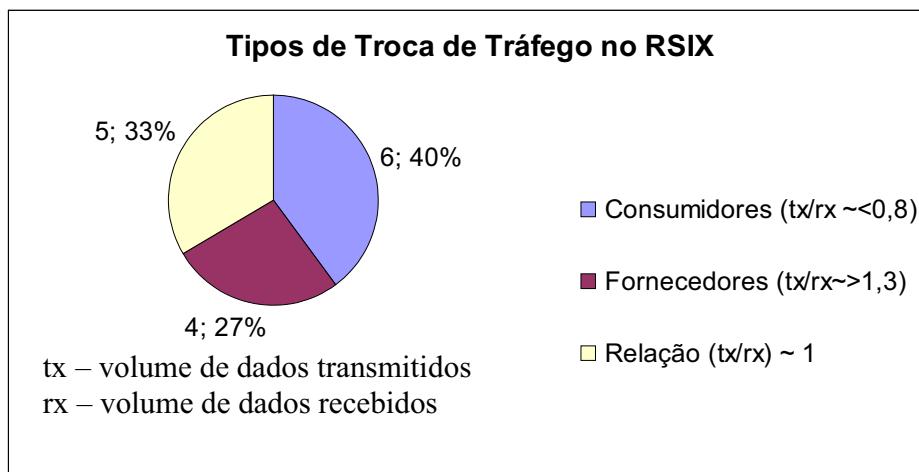


Figura 8. Tipos de Troca de Dados no RSIX – Dados de Março de 2004

4. Conclusões

Com este trabalho analisou-se o funcionamento básico do roteamento de pacotes em um ponto de troca de tráfego, bem como o princípio do protocolo BGP nesses PTTs.

Através da visualização da situação de cada um dos cinco ponto de troca de tráfego brasileiros, pôde-se observar suas semelhanças e diferenças. Verificou-se um aumento no tráfego de dados em todos os PTTs pesquisados. Pela observação dos resultados obtidos, conclui-se que o PTT-SP destaca-se em relação aos demais em todos os itens apresentados.

Referências

- Halabi, Bassam (1997) “Internet Routing Architectures”, Cisco Systems.
- Tanenbaum, Andrew S (1997) “Redes de Computadores”. editora Campus.
- Comitê Gestor Internet/BR (2004) “Roteamento”,
<http://www.gtrh.tche.br/ovni/roteamento3/introducao.htm>, Maio.
- Andreolli, Andrey (2004) “Controle do Protocolo BGP em PTT’s – 15ª Reunião do Grupo de Trabalho em Engenharia de Redes – GTER 15”,
<http://eng регистрация.br/gter15/videos/gerenciamentobgp/>, Maio.
- RFC 1771, (2004) “A Border Gateway Protocol 4 (BGP-4)”,
<http://www.ietf.org/rfc/rfc1771.txt>, Maio.
- RSIX, (2004) “Ponto de Troca de Tráfego Internet”, <http://www.rsix.tche.br>, Maio.
- ANSP, (2004) “Ponto de Troca de Tráfego da ANSP”, <http://www.ansp.br>, Maio.
- PRIX, (2004) “Ponto de Troca de Tráfego do Paraná”, <http://prix.pop-pr.rnp.br>, Maio.
- FIX, (2004) “Federal Interconnection of Brasilia”, <http://www.rnp.br>, Maio.
- OPTIX, (2004) “OptiGlobe Internet Exchange”, <http://www.optiglobe.com.br>

Administração de Sistemas de Autenticação de Usuários*

Diego Luís Kreutz

¹Laboratório de Sistemas de Computação — LSC

Núcleo de Ciência da Computação — NCC

Universidade Federal de Santa Maria — UFSM

kreutz@inf.ufsm.br

Resumo. *Ferramentas para o gerenciamento integrado de diferentes diretórios online são cada vez mais necessárias. Isso por que é comum a administração de uma rede local envolver o gerenciamento de bases de dados como o OpenLDAP, NIS e Samba. Nesse sentido, este trabalho apresenta um sistema simples para o gerenciamento integrado de diferentes diretórios online. Além disso, seu principal foco são ambiente acadêmicos, incluindo funcionalidades e recursos extras que podem agilizar e simplificar o controle das contas dos usuários de uma rede local.*

1. Introdução

O controle de acesso a máquinas e sistemas são partes críticas e comprometedoras em redes locais que prezam por segurança e disponibilidade. Esse controle de acesso é normalmente realizado através do uso de diretórios *online* como o LDAP¹, NIS² e Samba. Os dois primeiros são comumente utilizados para autenticar usuários em sistemas Unix, enquanto que o segundo serve de autenticador para ambientes Windows.

Atualmente existem vários utilitários e ferramentas de gerenciamento de diretórios *online*. Alguns desses utilitários são baseados em linha de comando enquanto outros fazem uso de interfaces gráficas. Cada um desses sistemas possui suas características e aplicações específicas. Uma característica mais geral das ferramentas disponíveis é a pouca simplicidade e facilidade de realizar um gerenciamento integrado de diferentes autenticadores de usuários em uma rede local.

A proposta deste trabalho é apresentar um sistema simples e adaptável para o gerenciamento integrado de diretórios como o LDAP, NIS e Samba. O projeto, desenvolvimento e implementação desse sistema foi baseado em necessidades básicas da rede do Núcleo de Ciência da Computação (NCC) da UFSM. O sistema torna possível a administração simultânea de diferentes sistemas de autenticação de usuários, como o OpenLDAP e Samba. Além disso, ele disponibiliza funcionalidades como cadastro de usuários em massa, configuração de leitores de e-mail como o *pine*, criação de exemplos de páginas *html* e *php* nas contas dos usuários, envio de e-mail de boas vindas e controle de expiração de senhas.

Na próxima seção é brevemente apresentada a arquitetura do sistema. Na seqüência são abordadas algumas funcionalidades e particularidades do sistema. Na seção seguinte são apresentados alguns trabalhos relacionados. Por fim, aparecem a conclusão e as perspectivas de continuidade.

2. Arquitetura

A proposta de um sistema de gerenciamento integrado de diretórios *online* surgiu a partir de necessidades administrativas do NCC. Em meados de 2000 os servidores NT foram subs-

*Fomento CNPq: processo 380049/03-1

¹Lightweight Directory Access Protocol (protocolo leve de acesso a diretórios)

²Network Information Service (serviço de informação de rede)

tituídos pelo Samba, rodando em GNU/Linux. Os servidores Solaris e AIX também foram migrados para GNU/Linux. A base de dados de usuários e máquinas (NIS) foi migrada para o OpenLDAP. No entanto, essa nova configuração da rede ainda requeria o gerenciamento de duas bases de informação, o OpenLDAP e o Samba. Como os dois sistemas estavam rodando em servidores GNU/Linux, muitas tarefas poderiam facilmente ser automatizadas e sincronizadas. Além disso, o cadastro de vários usuários a cada início de ano letivo, a configuração de leitores de e-mail por parte dos usuários e o controle mais rigoroso sobre as senhas dos usuários continuavam sendo tarefas pouco práticas. Por isso, decidiu-se pelo desenvolvimento de um sistema que fosse capaz de suprir as principais necessidades de gerenciamento da rede e ao mesmo tempo fosse legível, simples e modular a ponto de ser facilmente ajustável à diferentes ambientes e contextos.

A figura 1 ilustra a arquitetura do sistema. Ele é basicamente constituído por três partes. A primeira é formada pelos dados de entrada, que podem ser estáticos ou interativos. Estáticos significa que existe um arquivo de entrada para ser processado. Esse arquivo de entrada contém conjuntos de registros que podem ser utilizados para proceder cadastros ou remoções em massa. Os dados interativos são utilizados quando o administrador deseja realizar de forma interativa apenas um cadastro, acompanhando os passos de geração e inclusão do novo registro e criação da *home* do usuário com dados e configurações padrão.

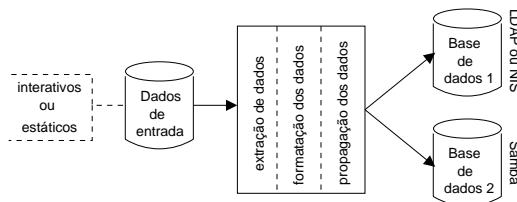


Figura 1: Ilustração da arquitetura do sistema

A segunda parte do sistema é constituída pelos módulos de gerenciamento e atualização. A primeira etapa, do funcionamento desses módulos, passa pela extração e classificação dos dados de entrada. Uma segunda fase consiste na formatação e preparação dos registros a serem incluídos ou manipulados. A etapa final é a propagação dos dados para as respectivas bases de informação.

O sistema é capaz de gerenciar simultaneamente uma ou mais bases de dados. Essa característica torna a ferramenta uma boa opção para o gerenciamento integrado de diferentes diretórios *online*, como o LDAP, NIS e Samba.

3. Características, Implementação e Manutenção do Sistema

O sistema foi planejado com o intuito de ser simples e adaptável. Um dos objetivos é permitir que administradores de redes ajustem facilmente o sistema para as suas necessidades ou situações particulares do dia-a-dia.

A primeira versão do sistema foi implementada utilizando *shell-scripts*. Uma característica marcante é a legibilidade, que torna a manutenção e extensão do sistema bastante simples. Outro fator que levou a essa opção esta relacionado aos usuários alvos, administradores de rede. Estes normalmente possuem ao menos algum conhecimento em linguagens *shell-script*. A maior parte dos gerenciadores de rede não são familiarizados com linguagens de programação como Perl, C e Java. Além disso, códigos nessas linguagens costumam ser menos legíveis e de difícil entendimento, pois as preferências de programação variam bastante de desenvolvedor para desenvolvedor. Não obstante a isso, a extensão ou adaptação do sistema demandaria mais trabalho caso o sistema fosse implementado em linguagens como Java ou C.

Apesar da simplicidade do sistema, os comandos de atualização e manipulação dos registros das bases de dados podem ser configurados para rodar sobre canais criptografados, utilizando SSL. O tráfego das senhas, por exemplo, é realizado com as mesmas criptografias e padrões de segurança utilizados pelos sistemas de autenticação para consultar e autenticar usuários na rede.

A praticidade e legibilidade do sistema permitem a fácil inclusão e manutenção de módulos. Uma outra característica é quanto a instalação do sistema. Ela é prática e fácil, bastando copiar o sistema para o local desejado, configurar algumas variáveis e o sistema estará funcional. Isso porque os comandos e módulos foram implementados utilizando caminhos relativos, ou seja, não há a necessidade de instalar o sistema (copiar os comandos) em local específico, como ocorre com sistemas como o *smbldap-tools* [Tournier, 2004].

As próximas seções apresentam algumas opções de configuração e funcionalidades do sistema.

3.1. Opções de configuração

Uma das características presentes no sistema é a possibilidade de configurá-lo para diferentes ambientes. Ele pode ser utilizado para gerenciar simultaneamente diferentes bases de informação.

Na figura 2 são apresentadas algumas opções de configuração do sistema. No caso, o objetivo principal é trabalhar com OpenLDAP, NIS e Samba.

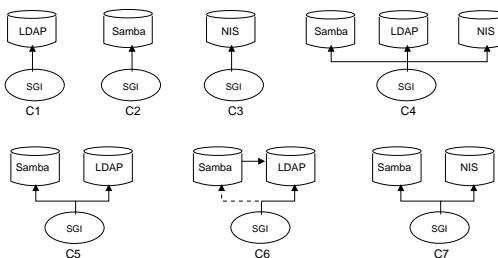


Figura 2: Ilustração de algumas possibilidades de configuração e utilização.

O sistema pode ser utilizado para gerenciar bases independentes, como é o caso das configurações C1, C2 e C3. As opções de configuração C4, C5, C6 e C7 buscam gerenciar simultaneamente mais de uma base de registros. Nestes casos, a manipulação dos dados de usuários é propagada para os diferentes diretórios *online*.

As configurações C4 e C6 representam dois casos especiais. A configuração C6 apresenta um leve diferencial em relação a arquitetura C5. Nela os dados do Samba são também armazenados no OpenLDAP, ou seja, existe apenas uma base de informação para ser gerenciada. Mas, por motivos de padronização, pode-se optar por continuar utilizando os comandos padrão do Samba. Neste caso, os comandos Samba terão seus efeitos propagados para o OpenLDAP pelo servidor Samba, o que torna o sistema mais comprehensível e legível.

A configuração C4 é interessante para ambientes de transição, onde está sendo realizado o porte de sistemas. Neste contexto, manter temporariamente o NIS, Samba e LDAP pode ser uma boa opção, não necessitando portar e configurar de uma única vez todos os computadores da rede. Em uma situação desse gênero os três sistemas permanecerão ativos até que todas as máquinas e sistemas tenham sido devidamente configurados.

Permitir que o sistema facilmente seja configurado para alguma dessas configurações é relativamente fácil. A implementação pode seguir basicamente dois caminhos: utilização de vetores de comando ou módulos separados. No caso de vetores de comandos é executado o comando correspondente a cada configuração. Esta opção de configuração é indicada em um arquivo de configuração ou variável de ambiente do sistema. Uma alternativa é

a implementação de módulos independentes para as diversas configurações. Neste caso, quando o sistema for configurado, basta indicar o conjunto de módulos que serão utilizados. Essa segunda abordagem foi a escolhida na implementação da primeira versão do sistema.

3.2. Funcionalidades comuns

As funcionalidades padrão do sistema incluem: inclusão e remoção interativa de um usuário, alteração de dados cadastrais de um usuário, troca de senha, inclusão e remoção de grupos, listagem de usuários e grupos, listagem de dados específicos de um determinado grupo ou usuário, listagem dos usuários que pertencem a um determinado grupo (o que pode gerar mais de uma consulta a uma base OpenLDAP), bloqueio e liberação de usuários, troca de *login*, troca de *shell*, listagem de usuários bloqueados, listagem de usuários com senhas expiradas e inclusão e remoção de usuários em grupos. Muitas dessas funcionalidades estão presentes em praticamente qualquer sistema de gerenciamento de diretórios *online* como o OpenLDAP, NIS e Samba.

3.3. Funcionalidades complementares

Para facilitar o gerenciamento e cadastro de usuários o sistema possui algumas funcionalidades complementares. Entre elas podem ser citadas a inclusão de conjuntos de usuários, a geração de arquivos de configuração de leitores de e-mail, a inclusão de dados como arquivos de configuração e modelos de páginas pessoais (configuração e uso).

As *homes* dos usuários podem estar na máquina em que o sistema está rodando ou em uma máquina remota. Existem duas possibilidades de gerar os diretórios de usuários: 1) o servidor NFS³ exporta o diretório raiz das *homes* para a máquina que roda o sistema; 2) utilizar comandos remotos para a geração dos diretórios dos usuários.

Uma característica interessante para laboratórios acadêmicos, por exemplo, é a possibilidade de gerar o cadastro simultâneo de um número qualquer de usuários a partir de um único arquivo de entrada. Na seqüência são apresentadas duas opções de configuração de um arquivo de entrada para o cadastro de vários usuários em massa.

Opção 1: o administrador do sistema entra apenas com o nome completo dos usuários. Nessa opção o administrador tem a possibilidade de solicitar ao sistema a geração de *logins* para os usuários e cadastro dos mesmos. O processo pode ser feito em dois passos: a) geração dos registros de cadastro; b) inclusão dos registros nas bases de dados. A primeira fase não é necessária. Ela é interessante somente quando o administrador deseja visualizar os registros antes que eles sejam efetivamente cadastrados. Abaixo segue um exemplo de como seria o arquivo de entrada no caso da opção 1.

```
# nome completo do usuario
Galinha Caipira
Porco Espinho
```

Opção 2: o administrador entra com os detalhes do usuário, como *login*, grupo e diretório raiz da *home*. Abaixo segue um exemplo de um arquivo de configuração para essa opção. Esse arquivo é semelhante ao que seria o arquivo intermediário da opção 1, comentada anteriormente.

```
# usuario | nome | grupo | diretório base
gcaipira|Galinha Caipira|administrador|/home/admin
pespinho|Porco Espinho|grp2004|/home/usuarios
```

No processo de geração do cadastro será avaliado a validade de grupos e *logins*. Caso algum grupo ainda não esteja cadastrado, será indicado a necessidade de cadastro do mesmo. Se algum *login* estiver duplicado no sistema será alertado a necessidade de modificar o(s) *login(s)* problemático(s). Neste caso, o administrador pode optar por uma geração automática de *logins* (opção 1), indicando ao sistema para tentar formar um *login* a

³Network File System (sistema de arquivo de rede)

partir do sobrenome e das iniciais dos demais nomes. Caso ainda existam *logins* duplicados, podem ser escolhidas opções como gerar um *login* a partir das iniciais do nome ou apenas com o primeiro nome ou o sobrenome.

Ambas as opções (1 e 2) irão gerar dados de saída semelhantes aos que seguem.

# identificador		senha randômica		identificador	login	nome do usuário
01		YuI8JkH		01	gcaipira	NOME: Galinha Caipira
02		MTb32YA		02	pespinho	NOME: Porco Espinho

O resultado final do cadastro será um arquivo *ps*⁴, pronto para ser impresso. As senhas e *logins* devem ser separados. Para manter a relação entre senhas e usuários existem os identificadores. Cada usuário receberá a sua senha provisória. Para tanto, basta destacar a senha do identificador e entrega-lá ao usuário.

O sistema, além de gerar a conta de um usuário e uma senha randômica, gera também o diretório *home* com algumas configurações e dados padrão. São criados os arquivos de configuração básica do *shell*, com alguns exemplos de uso, o diretório *public_html* com exemplos simples e documentados de páginas em *html* e *php*, é enviado um e-mail de boas vindas e contendo também informações sobre o uso da rede e é ainda criada uma configuração automática para leitores de e-mail como o *pine*. Desta forma o usuário recebe instruções básicas e essenciais de utilização e normas de conduta da rede.

O sistema pode ainda ser programado para controlar a expiração de senhas. Isso pode ser feito através do agendamento do módulo de verificação de expiração no *crontab* do sistema GNU/Linux. Caso a senha esteja próxima de expirar (número x de dias) é enviado um e-mail ao usuário informando o fato.

Os usuários podem trocar suas senhas via uma página segura. A página principal, para a troca de senhas, propaga a nova senha para todas as bases de dados. No entanto, caso o usuário preferir ele poderá manter senhas distintas para os diferentes sistemas (exemplo: uma senha para o OpenLDAP e outra para o Samba). Isso pode ser feito pelas páginas CGI auxiliares.

4. Trabalhos relacionados

O gerenciamento de diretórios *online* é uma necessidade antiga entre administradores de redes. Ferramentas como o *Swat*⁵ [Team, 2004], *Webmin* [Cameron and et. al., 2004], gerenciador de cadastros *LDAP*⁶ [Duergner et al., 2003], o pacote *yp-tools* [Kukuk, 2003], ferramenta gráfica de configuração do *Samba*⁷ [Sam, 2003], *gsmb* [Foucher@gch.iut-tlse3.fr, 1999], *JXplorer* [Betts, 2002] e *smbldap-tools* [Tournier, 2004] auxiliam os administradores de rede a realizar tarefas de manutenção em serviços como o *Samba*, *NIS* e *LDAP*. A maior parte destas ferramentas são gerenciáveis por interface gráfica. No entanto, elas carecem no que diz respeito a integração e ao gerenciamento de ambientes como os que comumente figuram em redes locais de laboratórios de computação.

Cada uma dessas ferramentas possibilita o gerenciamento de alguns serviços. Elas possuem características distintas e particulares. Utilitários como o *JXplorer* e o *LDAP Account Manager* possibilitam o gerenciamento de hierarquias e contas de usuários em diretórios *LDAP*. Ferramentas como o *Swat* e o *gsmb* facilitam a manipulação de contas em servidores *Samba*. O pacote *yp-tools* inclui ferramentas para gerenciar o *NIS*. Com excessão

⁴PostScript (linguagem de descrição em alto nível de páginas/telas/imagens impressas)

⁵Samba Web Administration Tool (ferramenta Web de administração do Samba)

⁶LDAP Account Manager (gerenciador de registros LDAP)

⁷Samba GUI Configuration Tool (ferramenta gráfica de configuração do Samba)

do *yp-tools* e do *smbldap-tools*, as demais ferramentas possuem interfaces gráficas simples e amistosa. Porém, dependendo do tipo de operação desejada, essas interfaces tornam-se pouco práticas e flexíveis.

O conjunto de aplicativos do *smbldap-tools* demonstra ser o mais flexível e útil. Eles possibilitam o gerenciamento de bases de informação LDAP e Samba simultaneamente. Além de poder migrar dados de bases NT para LDAP. No entanto, para redes acadêmicas essa ferramenta carece de recursos auxiliares. Além disso, seu código é relativamente fixo, o que pode tornar a manutenção ou instalação pouco flexível.

É nesse sentido que surgiu a idéia, aqui apresentada, de um sistema de gerenciamento integrado de contas de usuários do LDAP, NIS e Samba. Simplicidade e organização estão entre suas características. Devido a isso, esse sistema também é facilmente extensível e adaptável a variados tipos de ambientes e especificidades de domínios administrativos.

5. Conclusão e Trabalhos Futuros

Muitas vezes o gerenciamento de sistemas de informação e autenticação de redes locais é uma tarefa pouco prática, pois é comum existir mais de uma base de dados de usuários para ser gerenciada. Isso por que normalmente são utilizados sistemas como o NIS, ou o OpenLDAP, e o Samba para a autenticação de sistemas Unix e Windows, respectivamente. Logo, um sistema que possibilite um gerenciamento síncrono e integrado é interessante em um ambiente desse gênero.

Este artigo apresentou um sistema simples e prático para realizar o gerenciamento integrado de diferentes bases de informação, como o OpenLDAP, NIS e Samba. Além disso, foram mostradas algumas características que podem ser interessantes e facilitar a vida de administradores de laboratórios de computação. Entre elas podem ser citadas o cadastro de usuários em massa, a configuração automática de leitores de e-mail como o pine, a geração de modelos de páginas *html* e *php* nas contas dos usuários e o controle de expiração das senhas.

Trabalhos futuros. Entre os trabalhos futuros estão a integração das diferentes configurações do sistema (seção 3.3), produção de uma documentação e disponibilização do sistema. Além disso, implementar e disponibilizar opções de personalização de comandos, ou seja, o usuário poderá definir o nome e o modo que melhor lhe convier para utilizar as funcionalidades do sistema.

Referências

- (2003). Samba GUI Configuration Tool 1.0 Beta. <http://www.eatonweb.com/samba/>.
- Betts, C. (2002). JXplorer - Java LDAP Browser. <http://jxplorer.org>.
- Cameron, J. and et. al. (2004). Webmin. <http://www.webmin.com/>.
- Duergner, M., Gruber, R., Lutz, T., and Walchshäusl, L. (2003). LDAP Account Manager. <http://lam.sourceforge.net/>.
- Foucher@gch.iut-tlse3.fr (1999). Gsmb - simplified management of smbpasswd. <http://www.culte.org/projets/developpement/gsmb/>.
- Kukuk, T. (2003). Linux NIS Tools: yp-tools. <http://www.linux-nis.org/nis/yp-tools/>.
- Team, S. (2004). Swat - Samba Web Administration Tool. <http://us2.samba.org/samba/samba.html>.
- Tournier, J. (2004). Smbldap-tools User Manual (Release: 0.8.4). <http://docs.biostat.wustl.edu/smbldap-tools/html/index.html>.

KDD-NetManager: uma Ferramenta de Descoberta de Conhecimento em Bases de Dados (DCBD) Aplicada à Gerência Proativa em Redes de Comunicação

Flávia Pereira de Carvalho^{1,2}, Marcelo Azambuja^{1,2}, Jorge Guedes Silveira²

¹ Faculdade de Informática de Taquara
Faculdades de Taquara (FACCAT)
Av. Oscar Martins Rangel, 4500 - Taquara, RS - Brasil.

² Laboratório GPARC – Faculdade de Engenharia Elétrica
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Avenida Ipiranga, 6681 - Porto Alegre, RS – Brasil.

{fpereira, azambuja}@faccat.br, guedes@ee.pucrs.br

Resumo. Este trabalho apresenta estudos e resultados relacionados ao uso de Descoberta de Conhecimento em Base de Dados (DCBD) sobre a base de uma ferramenta de Gerenciamento de Redes. Esses resultados foram obtidos através do uso da ferramenta de Mineração de Dados KDD-NetManager, desenvolvida no transcorrer desta pesquisa, aplicada à base do Sistema Gerente Free Network Management System (FreeNMS), sistema este também desenvolvido pelo grupo de pesquisa envolvido neste trabalho. Através dos testes realizados com a ferramenta KDD-NetManager sobre a base de dados da empresa testbed utilizada nos testes, foi possível comprovar o grande benefício que as técnicas de DCBD podem fornecer para a área de gerenciamento de redes, auxiliando e sugerindo tomada de decisões.

1. Introdução

Técnicas de Descoberta de Conhecimento em Bases de Dados (DCBD) são alvos de estudo já há algum tempo pela comunidade científica. O volume de dados disponível nos SGBDs (Sistemas Gerenciadores de Bancos de Dados) das corporações é cada dia maior, tornando os sistemas de DCBD um meio de encontrar informações úteis “escondidas” nestes enormes volumes de dados [Feldens 1997].

A motivação para o uso de técnicas de DCBD neste trabalho ocorreu devido a esse crescente aumento do volume de dados que também ocorre em Sistemas de Gerenciamento de Redes (NMS – Network Management System). Com o auxílio da ferramenta de DCBD implementada durante as pesquisas, foi possível gerar ações de pró-atividade nas atividades de operação e manutenção da rede de um grande provedor de serviços de rede. Especificamente, os testes envolveram as informações do Sistema de Registro de Problemas (TTS – Trouble Ticket System) da referida empresa utilizada como plataforma de testes (*testbed*). As funcionalidades implementadas através do uso de técnicas de DCBD permitiram, por exemplo, que imediatamente à abertura de um “chamado” no sistema TTS (registro de um problema sobre um equipamento ou serviço), o sistema KDD-NetManager indique (apontando inclusive a “precisão” desta dica) o motivo para a falha registrada, bem como a solução para esta falha.

O referido sistema TTS é parte integrante da Plataforma de Gerenciamento de Redes FreeNMS [FreeNMS 2004], também desenvolvido pelo grupo que compõe este

trabalho. Técnicas e conceitos de RNA (Redes Neurais Artificiais), Gerência de Redes Baseada na Web (Web-based Management) e desenvolvimento baseado em Softwares Livres foram utilizados neste trabalho. Em [Fayyad 1995] pode-se encontrar os principais conceitos e técnicas da área de DCBD.

2. Ferramenta de Gerência de Redes - FreeNMS

O Free Network Management System (FreeNMS) é uma plataforma de Gerência de Redes que possui entre suas funcionalidades, características de Gerência de Falhas, Performance, Contabilização, Configuração e Segurança [FreeNMS 2004]. As RFCs (Request for Comments) 1156 e 1213, bem como [Soares 1995] e [Perkins 1997] relatam com detalhes o modelo de gerenciamento SNMP de redes.

Como adicional, o FreeNMS possui um Sistema de Registro de Problemas (TTS – Trouble Ticket System) que permite o registro e acompanhamento de todos os problemas que ocorrem na rede gerenciada ([Melchiors 1999] discute a necessidade e conceitos de sistemas TTS). Como exemplo de algumas características do TTS-FreeNMS pode-se citar o registro completo de todos os problemas em elementos de rede, encaminhamento de problemas para o corpo técnico responsável de forma automática e a manutenção do histórico de todas as ações tomadas, referentes a um problema, desde a sua abertura até o seu fechamento, além da integração absoluta com o sistema FreeNMS [Lunardelli 2002].

3. Especificação e Projeto do Sistema KDD-NetManager

Nesta seção são descritos todos os procedimentos técnicos envolvidos no estudo de caso prático realizado para DCBD no banco de dados do Sistema FreeNMS, utilizando o sistema de DCBD KDD-NetManager.

De acordo com [Feldens 1997] e [Fayyad 1995], os sistemas de DCBD costumam ser “especialistas”. Isto significa que estes sistemas são desenvolvidos tendo em vista uma série de características “esperadas” na base de dados que será utilizada nas descobertas de conhecimento. Alguns destes sistemas chegam a ser implementados para uma base de dados específica “única”, com nomes de campos e tipos de dados invariáveis (somente os “dados” realmente variando).

Desta forma, o protótipo aqui implementado não foge a esta regra (trabalhos estado-da-arte em DCBD visam melhorar estes algoritmos e sistemas disponíveis, tornando-os menos especialistas ou dependentes do tipo e fonte de dados disponível [Uludag 2003]).

3.1. Ambiente Operacional e de Desenvolvimento

Como Sistema Operacional e servidor Web (HTTP) foram utilizados os Softwares Livres GNU/Linux e Apache, respectivamente. PostgreSQL foi o SGBD acessado e utilizado como base de dados (este SGBD era o mesmo utilizado no ambiente *testbed* – uma empresa parceira do projeto FreeNMS). Para o desenvolvimento foram utilizadas as linguagens de desenvolvimento C e PHP. Naturalmente, SQL (Structured Query Language) foi utilizada intensivamente para o desenvolvimento dos módulos, haja visto a necessidade de acessos constantes ao Banco de Dados (BD). O ambiente e interface para os usuários é totalmente baseado na Web.

3.2. Tipos de Descobertas Implementadas

Genericamente, os tipos de descobertas implementadas foram: Dependência (Regra Curta), Descrição de Conceitos (Aprendizado Supervisionado), Identificação de Classes (Clusterização) e Detecção de Desvio. Houve casos onde junções destas diversas técnicas permitiram criar algumas das funcionalidades pretendidas para o sistema. Maiores detalhes sobre “tipos de DCBD” podem ser vistos em [Fayyad 1995], [Feldens 1997] e [Carvalho 2003].

3.3. Resultados do Protótipo KDD-NetManager

A seguir são listados alguns dos “tipos de descobertas” que foram implementados no protótipo do Módulo DCBD.

3.3.1. Tipo de Descoberta: Dependência (Regra Curta)

Este tipo de descoberta é baseado na dependência entre dois atributos existentes nas tabelas do BD. Quando um valor pode influenciar outro, há uma dependência (ou relação) entre estes valores, e muitas regras podem ser descobertas a partir desta constatação.

Basicamente, o algoritmo procura campos nos diversos registros cujos conteúdos sejam idênticos (entre os registros). Encontrando semelhanças, “contadores” vão sendo incrementados. Aquelas semelhanças, entre todos os registros pesquisados, cujos contadores alcancem um número mínimo de “suporte”, serão considerados “regras”.

Um exemplo de regra que poderia ser encontrada por esta descoberta na base de dados deste protótipo:

SE < modelo_equipamento = Cisco 2500 > **ENTÃO** < falha = placa queimada > (com 45% de “suporte”¹)



Figura 1. Tela do Protótipo, Utilizando “Regra Curta”.

¹ “Suporte” (estatístico): dentre os casos disponíveis para a análise realizada, esta indução é baseada dentro de um percentual de, por exemplo, um mínimo de 45% dos casos.

3.3.2. Tipo de Descoberta: Identificação de Classes (Clusterização)

Enquanto no tipo de descoberta “Descrição de Conceitos (aprendizado supervisionado)” (seção 3.3.3, a seguir) há uma orientação clara e definida sobre como a base de conhecimento deve ser utilizada (os atributos e as classes são previamente definidas e informadas para o processo de DCBD), no tipo de descoberta “Identificação de Classes” (como o nome já deixa claro) as classes não são induzidas pelo software, mas sim necessitam ser detectadas e determinadas pelo sistema de DCBD. Um exemplo de resultado obtido:

- equipamentos tipo “switch”, com falhas do tipo “desligado incorretamente”, voltaram a repetir o mesmo problema em um prazo menor do que 10 dias.

O algoritmo deverá iniciar lendo o primeiro registro existente na base de dados e compará-lo com todos os demais registros da base (e assim sucessivamente, para cada registro). Os algoritmos que possuírem semelhanças entre si poderão vir a ser considerados clusters (se alcançarem um valor mínimo de “suporte”). No algoritmo implementado, é realizado um ranking das “relações de registros” (dois ou mais registros semelhantes) que mais características idênticas foram localizadas durante a pesquisa envolvendo todos os registros. As semelhanças entre os grupos de registros é que permitirão a identificação de clusters.

3.3.3. Tipo de Descoberta: Descrição de Conceitos (Aprendizado Supervisionado)

Este tipo de descoberta consegue determinar a classe que cada tipo de dado pertence, baseada em características previamente conhecidas (ou descobertas no mesmo momento, iterativamente) e comuns entre os membros de cada classe. Este tipo de descoberta pode colaborar na “aprendizagem baseada em exemplos”.

No protótipo implementado neste trabalho, a Descrição de Conceitos usa como base de conhecimento os clusters identificados anteriormente pela descoberta Clusterização. Dada uma determinada Falha, em um determinado Equipamento (isto sendo “analisado” no momento da abertura de um novo “chamado” no sistema), esta descoberta aponta qual a possível solução, conforme demonstram as Figuras 2 e 3.

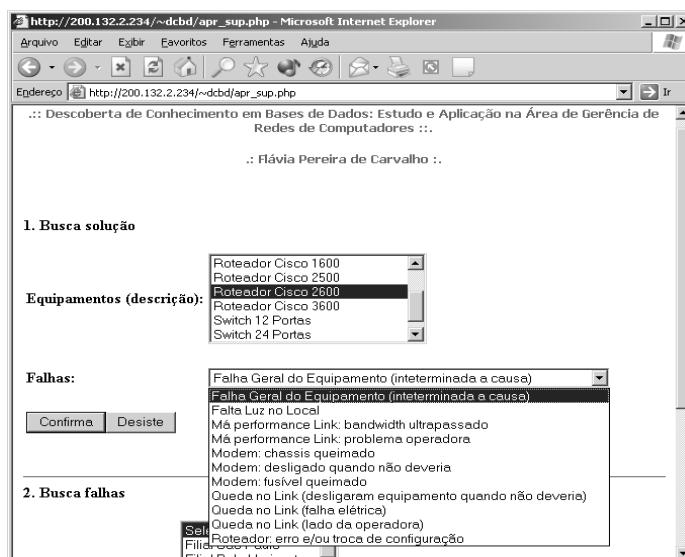


Figura 2. Rotina de DCBD do protótipo “Aprendizado Supervisionado”.

3.3.3.1. Exemplo de Descoberta Realizada com Descrição de Conceitos

Neste exemplo, dado o Modelo de Equipamento, uma lista “montada” dinamicamente pelo sistema aponta todos os tipos de falhas que já ocorreram com aquele modelo de equipamento, como pode ser observado na Figura 2.

O usuário então pode selecionar o tipo de falha que o equipamento está apresentando e, com isto, o sistema consegue sugerir a solução para a falha. Um exemplo de solução indicada no momento da “abertura de chamado” simulada com esta DCBD está demonstrado na Figura 3:

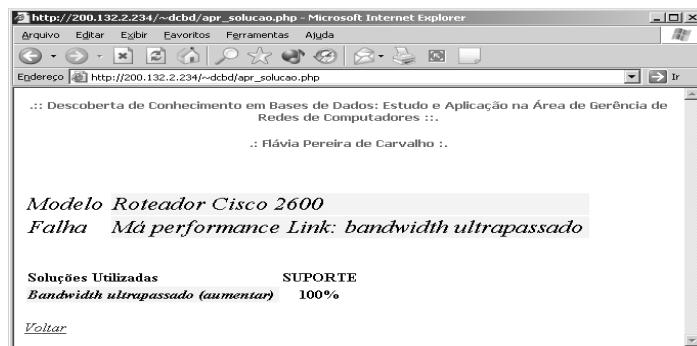


Figura 3. “Aprendizado Supervisionado” - a partir do Modelo do Equipamento e Falha, o sistema sugere a solução provável.

3.3.4. Tipo de Descoberta: Previsão

Este tipo de DCBD implementada visa “prever” possíveis problemas na rede e indicar aos administradores as prováveis próximas falhas. Estas rotinas são muito interessantes e relativamente fáceis de serem desenvolvidas. Apesar da aparente complexidade envolvida, na prática isto não acontece.

O algoritmo implementado no protótipo utiliza os seguintes artifícios:

- Registros que possuam os mesmos modelos de equipamentos, com as mesmas falhas, são analisadas e treinam a base de conhecimento que será necessária (criação de regras e de conhecimento para a rotina de verificação propriamente dita). O que interessa para a rotina de treinamento, além dos campos serem iguais, é “aprender” a média de tempo entre cada ocorrência de um determinado problema para um determinado equipamento.
- Tendo esta base de conhecimento sendo constantemente treinada, a rotina de DCBD necessita percorrer a base real de dados e comparar se há equipamentos entrando na “faixa de risco” que a base de conhecimento aponta (é feita uma comparação entre as duas bases, a cada execução do processo de Previsão).

O ideal em um sistema deste tipo é que, por exemplo, a cada madrugada o processo seja executado, e as previsões encontradas encaminhadas por e-mail para o gerente da rede e para uma tela de “alarmes” específica do sistema.

4. Conclusão

Através dos testes realizados com a aplicação da ferramenta de DCBD KDD-NetManager sobre a base de dados da plataforma de Gerência de Redes FreeNMS,

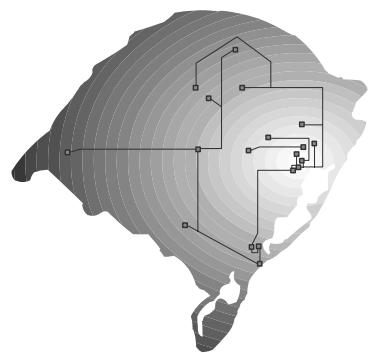
constatou-se a adequação deste enfoque no desenvolvimento de um sistema especialista de apoio à Gerência de Redes.

Os resultados obtidos não deixam dúvidas sobre a utilidade desta tecnologia quando utilizada em conjunto com um Banco de Dados de um Sistema Gerenciador de Redes típico. Nas pesquisas realizadas com os softwares líderes desta área, tais como HP OpenView e IBM Tivoli/NetView, não foram encontradas características que gerem resultados como os que foram aqui demonstrados. Desta forma, para um sistema em desenvolvimento, como é o caso do FreeNMS, que busca seu lugar no mercado, a utilização em conjunto e otimizada de um software de DCBD como o protótipo iniciado neste trabalho, poderia ser um diferencial de qualidade muito importante.

Como continuação desta pesquisa, o grupo pretende principalmente ampliar os estudos sobre algoritmos e técnicas utilizando RNA, que têm se mostrado muito indicados para trabalhos envolvendo Descoberta de Conhecimento [Mendes 2003].

Referências Bibliográficas

- Carvalho, Flávia. (2003) Técnicas de Descoberta de Conhecimento em Bases de Dados Aplicadas à Gerência Proativa em Redes de Comunicação. Dissertação de Mestrado – PUCRS.
- Fayyad, Usama; Simoudis, Evangelos. (1995) Tutorial: Knowledge Discovery and Data Mining. Fourteenth International Joint Conference on Artificial Intelligence. Montreal, Quebec, Canadá, IJCAI. Disponível em: <<http://www-aig.jpl.nasa.gov/public/kdd95/tutorials/IJCAI95-tutorial.html#Content>>. Acesso em: Jan. 2004.
- Feldens, Miguel. (1997) Engenharia da Descoberta de Conhecimento em Bases de Dados: Estudo e Aplicação na Área de Saúde. Dissertação de Mestrado – UFRGS.
- FreeNMS - Free Network Management System. Documentação e Descrição do Projeto. Disponível em: <<http://www.freenms.org>>. Acesso em: Mar. 2004.
- Lunardelli, Fernando; Azambuja, Marcelo. (2002) Sistemas TTS: Uma abordagem voltada para Sistemas de Gerenciamento de Redes. Projeto Parks/PUCRS Management System. Publicações do Projeto.
- Melchiors, Cristina. Tarouco, Liane. (1999) DUMBO: Uma abordagem para Gerenciamento de Falhas Utilizando Raciocínio Baseado em Casos. In: Simpósio Brasileiro de Redes de Computadores, 17., Salvador, Anais... Salvador: SBC, 1999.
- Mendes, Daniele; Oliveira, Marcio. (2003) Tutorial de Redes Neurais: Aplicações em Bioinformática. Disponível em: <<http://www.lncc.br/~labinfo/tutorialRN/>>. Acesso em: Out. 2003.
- Perkins, David. McGINNIS, Evan. Understanding SNMP MIBs. Chapter 5. Prentice Hall PTR. Upper Saddle River, New Jersey, 1997.
- Soares, L.; Lemos, G.; Colcher, S. Redes de Computadores: Das LANs, MANs e WANs às Redes ATM. Editora Campus, 1995.
- Uludag, Mahmut. (2003) Multi-Relational Rule Discovery. Research Progress Report, Eastern Mediterranean University, North Cyprus. Disponível em: <<http://cmpe.edu.tr/rila/>>. Acesso em: Nov. 2003.



Sessão Técnica 2

Tolerância a Falhas I

Implementação de um Injetor de Falhas de Comunicação para Aplicações Java Baseado em JVMTI*

Júlio Gerchman¹, Gabriela Jacques-Silva^{1†}, Taisy Silva Weber¹

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul
Caixa Postal 15064 – 90501-970 Porto Alegre, RS

{juliog,gjsilva,taisy}@inf.ufrgs.br

Resumo. O uso de aplicações distribuídas em Java em sistemas de missão crítica exige a validação de seus mecanismos de tolerância a falhas. Para realizar essa validação experimentalmente foi desenvolvido FIONA, um injetor de falhas de comunicação para aplicações Java. FIONA utiliza a nova interface de controle e depuração de máquinas virtuais JVMTI disponibilizada com a versão 1.5 da J2SE. O uso dessa interface proporciona a realização de experimentos de injeção de falhas de modo transparente à aplicação em teste, sem necessidade de alteração do código-fonte.

1. Introdução

Aplicações distribuídas representam um desafio à área de tolerância a falhas, pois além de ser necessário considerar problemas internos a cada nó participante, erros na comunicação também podem ocorrer. Para que este tipo de aplicação seja utilizada em sistemas de alta disponibilidade, é necessária a implementação de técnicas que permitam o funcionamento correto do sistema mesmo na ocorrência falhas. Uma etapa fundamental no desenvolvimento de sistemas tolerantes a falhas é a fase de validação, na qual é verificado o comportamento do sistema em vários cenários de falha. Não verificar os mecanismos de detecção e correção de erros pode permitir a ocorrência de comportamentos inesperados e errôneos já em fase operacional do *software*.

A injeção de falhas é uma técnica de validação experimental de sistemas, na qual falhas são introduzidas controlada e artificialmente no sistema, tendo sua resposta monitorada [Hsueh *et al.* 1997]. Desta maneira, é possível testar a eficiência dos mecanismos de tolerância a falhas implementados e também determinar medidas de dependabilidade, como a cobertura da detecção de erros e o impacto no desempenho dos mecanismos desenvolvidos.

Este trabalho apresenta o injetor de falhas de comunicação FIONA (*Fault Injector Oriented to Network Applications*) destinado a validação de aplicações distribuídas desenvolvidas usando a plataforma Java, que é largamente usada no desenvolvimento deste tipo de sistema devido à sua portabilidade. Esta ferramenta injeta as falhas na própria

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

†Bolsista do Conselho Nacional de Desenvolvimento Científico e Tecnológico

máquina virtual Java (JVM), acima do nível da pilha de protocolos de comunicação do sistema operacional, sendo assim facilmente portável.

Para permitir a injeção de falhas é usada a interface de programação nativa *Java Virtual Machine Tool Interface* (JVMTI) [Sun Microsystems 2004]. A JVMTI é uma interface que possibilita tanto monitoramento como controle de execução de aplicações em execução em uma JVM. Um dos recursos disponibilizados pela JVMTI é a instrumentação de *bytecodes*, que neste trabalho é usado para a inserção de código de injeção de falhas na aplicação.

2. Injeção de Falhas

A injeção de falhas pode ser realizada de várias formas, sendo as mais comuns por simulação, por *hardware* ou por *software*. A ferramenta apresentada neste trabalho se enquadra na última categoria. Uma ferramenta de injeção de falhas por *software* geralmente é um trecho de código que usa todos os ganchos (*hooks*) possíveis do processador e do sistema para criar um comportamento incorreto de maneira controlada [Carreira e Silva 1998]. A estratégia de implementação por *software* é mais flexível e tem menor custo que os outros métodos, além de poder simular tanto problemas de *software* como de *hardware*. Em FIONA são simuladas falhas de comunicação, sendo que estas podem tanto ser consideradas falhas de *hardware* como também resultado de alguma falha de *software*. Exemplos deste tipo de falhas incluem mensagens que chegam ao destinatário fora de ordem ou que são perdidas.

Outros exemplos de ferramentas de injeção de falhas de comunicação são OR-CHESTRA [Dawson *et al.* 1996] e ComFIRM [Barcelos *et al.* 2000]. A primeira ferramenta foi desenvolvida especificamente para teste de dependabilidade de protocolos distribuídos. A injeção de falhas é através da inserção de uma camada na pilha de protocolos, chamada de PFI (*Protocol Fault Injection*). A ferramenta ComFIRM (*Communication Fault Injection through OS Resources Modification*) injeta falhas diretamente no núcleo do sistema operacional Linux, no nível mais baixo do tratamento de mensagens pelo subsistema de rede. O código da ferramenta é inserido diretamente no núcleo do sistema operacional, o que diminui consideravelmente a intrusão temporal da ferramenta no sistema sob teste. Uma ferramenta para injeção de falhas em Java é Jaca, que usa reflexão computacional para alterar o comportamento da aplicação. Jaca injeta falhas de interface e recentemente foi estendida para injeção de falhas de comunicação [Jacques-Silva *et al.* 2004]. Tal extensão exige uma fase de pré-processamento para alteração das referências das classes de comunicação para as classes instrumentadas. O pré-processamento pode ser tanto para a alteração do código da aplicação quanto para a alteração de *bytecodes*.

FIONA usa como abordagem de injeção de falhas uma interface de programação nativa padrão para desenvolvimento de ferramentas de monitoramento e depuração em Java, chamada JVMTI (*Java Virtual Machine Tool Interface*). FIONA baseia sua implementação em um agente JVMTI, desenvolvido em C, juntamente com classes de injeção de falhas desenvolvidas em Java. Através do uso da linguagem nativa no menor número possível de casos, FIONA mantém sua portabilidade para todas as máquinas virtuais que implementam JVMTI.

3. JVMTI – Java Virtual Machine Tool Interface

A *Java Virtual Machine Tool Interface* (JVMTI) é uma interface de programação para monitoração e controle de execução de aplicações Java, que permite a interceptação de eventos da máquina virtual, instrumentação das aplicações em tempo de execução e é desenvolvida diretamente pela Sun, a criadora da plataforma. A JVMTI proporciona uma interface com a máquina virtual Java, permitindo o seu controle, depuração, perfilamento e monitoramento, entre outras atividades. Esta interface é uma parte da *Java Platform Debugger Architecture* (JPDA), que provê interfaces de programação para a depuração de aplicações. JVMTI é oferecida a partir da versão 1.5 da API e vem substituir as interfaces *Java Virtual Machine Profiling Interface* (JVMPI) e *Java Virtual Machine Debugger Interface* (JVMDI), que serão descontinuadas nas próximas versões.

O componente cliente das funções da JVMTI é chamado *agente*. O agente é uma biblioteca dinâmica que é notificada de eventos de interesse através de *eventos* como, por exemplo, o carregamento de uma classe ou o início da execução de uma *thread*. Após a notificação, o agente pode chamar funções que podem alterar ou inspecionar em maior detalhe o estado da máquina virtual. O agente executa juntamente com o código da máquina virtual Java, proporcionando controle com uma intrusão mínima na execução da aplicação. Como é possível ainda ele ser controlado por um processo em separado, pode ser mantido compacto, interferindo ainda menos na execução [Sun Microsystems 2004].

O uso da JVMTI descarta a necessidade de alteração e recompilação das aplicações alvo para execução de experimentos de injeção de falhas, pois toda a instrumentação pode ser realizada durante a execução. O agente JVMTI é carregado opcionalmente no disparo da máquina virtual. Essas características facilitam e flexibilizam o uso de um injetor programado com essa interface.

4. FIONA – Fault Injector Oriented to Network Applications

FIONA – *Fault Injector Oriented to Network Applications* – é uma ferramenta desenvolvida para condução de experimentos de injeção de falhas em aplicações Java baseada em JVMTI com o objetivo específico de implementação de falhas de comunicação. FIONA pode ser utilizado para a verificação do funcionamento correto de mecanismos de tolerância a falhas em aplicações Java distribuídas de uma maneira fácil e flexível.

O modelo de falhas implementado trata de falhas que podem ocorrer no protocolo UDP (*User Datagram Protocol*). Esse protocolo é não confiável e não orientado a conexão; no entanto, é comumente utilizado como base para implementação de protocolos que implementam a confiabilidade necessária através de camadas superiores adicionais. Um desses exemplos é o *middleware* de comunicação de grupo JGroups [Ban 1998], que usa UDP por *default* na base de sua pilha de protocolos. Este é um caso onde um injetor de falhas na comunicação no protocolo UDP é de grande utilidade.

4.1. Arquitetura e Funcionamento

FIONA implementa injeção de falhas na comunicação pelo protocolo UDP através da instrumentação da classe `java.net.DatagramSocket`, que contém os métodos `send()` e `receive()` usados para enviar e receber datagramas desse protocolo. A

instrumentação é realizada pelo agente JVMTI durante o carregamento desta classe. Na fase de inicialização também são carregadas outras classes auxiliares, necessárias para a configuração do injetor e controle do experimento.

O agente JVMTI é notificado do carregamento da classe `java.net DatagramSocket` através do evento *Class File Load Hook*, capturado na fase de inicialização da máquina virtual (*start phase*), antes da fase de execução da aplicação. No momento da notificação, uma imagem do arquivo de classe já foi carregado em memória, porém ainda não foi processado pela máquina virtual. O agente então substitui essa imagem por uma versão instrumentada, localizada junto aos outros arquivos do injetor. É essa versão instrumentada que será interpretada e executada de modo transparente à aplicação. Assim, ela não necessita de modificações em seu código-fonte para execução com injeção de falhas.

Quando a máquina virtual entra na fase de execução (*live phase*), a classe `BancoFalhas` é instanciada. Essa classe, ao ser instanciada, lê o arquivo de configuração do injetor e armazena quais falhas devem ser ativadas durante a execução da aplicação. `BancoFalhas` também provê métodos estáticos para acesso às falhas carregadas, que serão usados pela classe `java.net DatagramSocket` instrumentada.

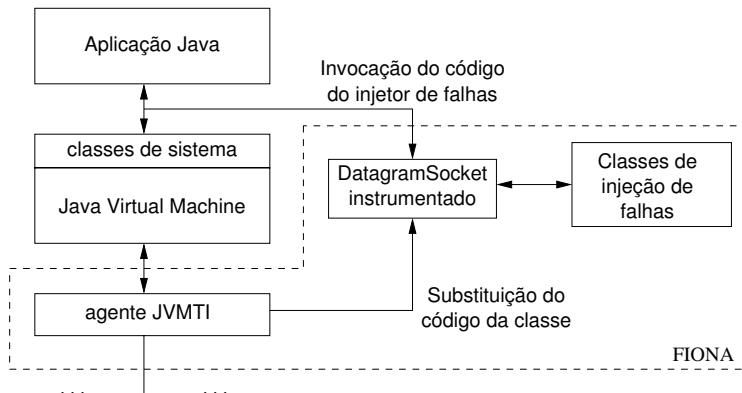


Figura 1: Arquitetura de FIONA

Como dito anteriormente, o envio e recebimento de um datagrama UDP pela aplicação se dá através dos métodos `send()` e `receive()` da classe `java.net DatagramSocket`. A chamada a esses métodos executará o código instrumentado, carregado anteriormente pelo agente JVMTI. A classe chama o método `BancoFalhas.retornaFalhas()`, passando como argumento o pacote a ser enviado. Este método retorna um vetor com falhas (objetos de especializações da superclasse `Falha`) que têm a possibilidade de serem injetadas no momento, dado o endereço e porta do remetente e do destinatário do datagrama.

4.2. Modelo de Falhas e Configuração do Injetor

As falhas consideradas por FIONA são baseadas no modelo de falhas para sistemas distribuídos definido por Cristian [Cristian 1991], que descreve falhas de omissão, de temporização, de resposta e de colapso. As falhas de omissão ocorrem quando um servidor não responde a uma requisição; as de temporização, quando a resposta ocorre fora

do intervalo especificado, situação geralmente associada a problemas de desempenho; as de resposta, quando a mensagem de retorno é incorreta por apresentar um valor ou uma transição de estado errôneos; as de colapso, quando o servidor pára totalmente de responder. Esse modelo foi escolhido porque, além dessas falhas serem muito comuns em um ambiente de aplicações distribuídas, o protocolo UDP não as trata.

Foram implementadas em FIONA falhas de omissão no envio e recebimento de mensagens, temporização e colapso de *host*. Durante a execução dos métodos `send()` ou `receive()`, as falhas definidas na configuração do injetor são testadas para verificar se devem ser ativadas no momento através de seu método `injeta()`, que retorna um valor booleano. Esse método é implementado de forma específica para cada classe de falha.

Caso pelo menos uma das falhas de omissão deve ser injetada, ou o pacote não é enviado (no método `send()`) ou é descartado e torna-se a esperar por um novo (no método `receive()`). No envio de pacotes, caso nenhuma falha de omissão deva ser injetada mas uma de temporização sim, uma *thread* é criada para enviar o pacote após um atraso estabelecido. Essa *thread* é criada para que o fluxo de execução retorne para a aplicação logo após a chamada a `send()` e se encerra após o envio efetivo. Se nenhuma falha deve ser injetada, o pacote é enviado normalmente.

Através de FIONA também é possível validar protocolos que utilizam multicast UDP. Como a classe Java que implementa essa funcionalidade, `java.net.MulticastSocket`, é subclasse de `java.net.DatagramSocket` e os métodos `send()` e `receive()` não são reimplementados, a injeção de falhas acontece transparentemente, usando o mesmo mecanismo. Esta característica pode ser explorada para fazer teste em aplicações que usem primitivas de multicast como base para camadas que implementam a confiabilidade em níveis superiores da pilha de protocolos.

As falhas que devem ser ativadas durante a execução da aplicação são especificadas em um arquivo texto contendo sua configuração. Esse arquivo é lido e interpretado por FIONA quando a classe `BancoFalhas` é instanciada, no início da execução da aplicação. Em cada linha é especificada uma falha e seus parâmetros. A seguir vemos um exemplo de configuração de falha de temporização:

```
FalhaUdpTemporizacao:<repeticao>:<inicio>:<taxa_falhas>:  
<atraso_minimo>:<atraso_maximo>:<host_origem>:<porta_origem>:  
<host_destino>:<porta_destino>:
```

O primeiro campo especifica qual a falha está sendo configurada, neste caso, uma falha de temporização. O campo `repeticao` especifica o padrão de repetição da falha: transitória (`t`, apenas uma ocorrência, na mensagem cujo número é especificado no campo `inicio`), permanente (`p`, a partir da mensagem de número especificado em `inicio` a falha sempre é ativada) ou intermitente (`i`, ocorre com uma probabilidade especificada no campo `taxa_falhas`). Os campos de atraso mínimo e máximo são particulares da falha de temporização. O *socket* no qual a falha será injetada é definido pelos campos de `hosts` e portas origem e destino.

5. Conclusões e Trabalhos Futuros

O artigo apresentou a implementação do injetor de falhas de comunicação FIONA. FIONA injeta falhas de comunicação em aplicações Java que utilizam o protocolo UDP de maneira completamente transparente para a aplicação, permitindo validar os mecanismos de tolerância a falhas desta.

A implementação de FIONA se deu através do uso da nova interface de controle, monitoramento e depuração da plataforma Java, a JVMTI. Essa interface permite a criação de programas (os agentes) capazes de inspecionar e alterar o estado de uma máquina virtual sem a interferência da aplicação. FIONA foi utilizado para o teste do mecanismo de detecção e recuperação de mensagens em uma aplicação tipo *shared whiteboard*, no qual demonstrou sua eficácia e facilidade de uso.

Trabalhos futuros seguem duas linhas complementares. Uma delas é a extensão da ferramenta para incorporação de modelos de falhas para o protocolo TCP e para o mecanismo de RMI (*Remote Method Invocation*). A outra é a extensão da ferramenta de modo a realizar a injeção distribuída de falhas, visando facilitar a condução de experimentos de validação em aplicações distribuídas, como ferramentas para *grid computing* e sistemas que utilizam comunicação em grupo.

Referências

- Barcelos, P. P. A.; Leite, F. O.; Weber, T. S. *Building a Fault Injector to Validate Fault Tolerant Communication Protocols*. In Proceedings of International Conference on Parallel Computing Systems. Ensenada, México. Agosto 1999.
- Ban, B. *JavaGroups - Group Communication Patterns in Java*. Department of Computer Science, Cornell University. Julho 1998.
- Carreira, J.; Silva, J. G. *Why do Some (weird) People Inject Faults?* ACM SIGSOFT, Software Engineering Notes, Volume 23, Número 1, pp. 42-43. Janeiro 1998.
- Cristian, F. *Understanding Fault-Tolerant Distributed Systems*. Communications of the ACM, Volume 34, Número 2, pp. 56-78. Fevereiro 1991.
- Dawson, S.; Jahanian, F.; Mitton, T. *ORCHESTRA: A Probing and Fault Injection Environment for Testing Protocol Implementations*. In Proceedings of IPDS'96. Urbana-Champaign, Estados Unidos. Setembro 1996.
- Hsueh, M.-C.; Tsai, T.; Iyer, R. *Fault Injection Techniques and Tools*. IEEE Computer, Volume 30, Número 4, pp. 75-82. Abril 1997.
- Jacques-Silva, G.; Moraes, R. L. O.; Weber, T. S.; Martins, E. *Validando Sistemas Distribuídos Desenvolvidos em Java Utilizando Injeção de Falhas de Comunicação por Software*. Anais do V Workshop de Testes e Tolerância a Falhas. Gramado, Brasil. Maio 2004.
- Sun Microsystems, *Java Virtual Machine Tool Interface*. <http://java.sun.com/j2se/1.5.0/docs/guide/jvmti/>. 2004.

Injeção de Falhas de Comunicação por Sondagem Dinâmica*

Roberto Jung Drebels, Taisy Silva Weber

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

{drebels, taisy}@inf.ufrgs.br

Abstract. This paper presents fault injection as a technique for application and protocol validation. Associating fault injection to network emulation, it shows the techniques used for communication fault injector implementation in the Linux environment. Finally, it suggests the use of dynamic probing (DProbes) for new fault injector building.

Resumo. Este trabalho apresenta a injeção de falhas como técnica de validação de protocolos e aplicações. Associando-a à emulação de redes, mostra as técnicas usuais de implementação de injetores de falhas de comunicação em ambiente Linux. Por fim, sugere a utilização de sondagem dinâmica (DProbes) para a construção de novos injetores de falhas.

1. Injeção de Falhas

A adoção dos sistemas computacionais nas mais diversas tarefas exige a dependabilidade destes sistemas. De forma simplificada, *dependabilidade* abrange duas características principais: *disponibilidade*, a capacidade de um sistema estar pronto para uso; e *confiabilidade*, a capacidade desse sistema continuar em serviço, mesmo na presença de falhas.

É necessário, então, que existam técnicas para a avaliação da dependabilidade. Essas técnicas funcionam através do estudo do comportamento do sistema ou de um modelo dele na presença de falhas. Nas *técnicas experimentais* a idéia é observar o sistema real sobre a presença de falhas e daí levantar conclusões sobre sua dependabilidade. Técnicas relacionadas ao uso, isto é, simplesmente utilizar o sistema aguardando a ocorrência natural de falhas observando como ele responde às mesmas, possuem um inconveniente: falhas naturais podem levar longos períodos para ocorrerem. Estatisticamente, é possível acelerar sua ocorrência ao observarmos um grande número de sistemas idênticos simultaneamente, mas nesse caso troca-se a necessidade de tempo pela de quantidade.

Uma idéia para acelerar a ocorrência de falhas é mudarmos para uma atitude ativa perante o sistema, criando falhas através de um *injetor de falhas*. Trabalhando sobre o sistema real temos uma avaliação *efetiva* da sua dependabilidade, sem estarmos sujeitos à variabilidade temporal da ocorrência de falhas naturais, e, mais importante, podendo observar o sistema sob a ocorrência de falhas específicas, em momentos específicos.

*Desenvolvido em parceria com HP Brasil P&D e Projeto CNPq ACERTE (#472084/2003-8)

Injetores de falhas podem ser implementados em hardware ou software. Injetores por hardware atuam no nível elétrico, sendo efetivamente injetores de falhas; enquanto injetores por software atuam no nível lógico, sendo portanto na realidade injetores de erros. Entretanto, ambas as abordagens costumam ser chamadas de injeção de falhas. Neste artigo a ênfase é nos injetores de falhas por software.

2. Falhas de Comunicação

A comunicação de equipamentos eletrônicos é feita através de dispositivos que atuam sobre o ambiente físico, recebendo e transmitindo sinais eletromagnéticos. A ocorrência de interferência sobre esses sinais, bastante freqüente, se não tratada, pode acarretar na perda de informações. Além disso, a utilização de *buffers* finitos nos equipamentos implica na possível perda de mensagens. Por estas razões é comum o emprego de mecanismos de tolerância a falhas, como cálculos de verificação e correção ou retransmissão na comunicação.

Aplicações adaptativas permitem que mesmo com flutuações no nível de qualidade dos parâmetros de comunicação – taxas de transmissão e perda, latência e variância (*jitter*) – ela se adapte e permaneça fornecendo um serviço aceitável. No seu desenvolvimento, idealmente, deve-se testar essas aplicações sob cada cenário de comunicação. Uma possibilidade é executá-las sobre as diversas tecnologias de acesso, mas isto necessitaria de uma vasta gama de equipamentos, das diversas tecnologias. Outra alternativa é a emulação, através da variação desses parâmetros de comunicação, de várias tecnologias utilizando-se de apenas algumas delas, de forma que para a aplicação seja indistingüível uma rede com parâmetros reais de emulados. Aqui, novamente, a injeção de falhas de comunicação pode ser utilizada, não como mecanismo de validação das técnicas de tolerância a falhas de protocolos e algoritmos distribuídos, mas para esta conformação de tráfego.

3. Falhas de Comunicação em Ambiente Linux

Os experimentos de injeção de falhas só podem ocorrer na plataforma que suporta as aplicações a serem testadas ou validadas. O sistema operacional Linux atingiu maturidade. Além disso, pela sua origem, é imediatamente associado às aplicações de rede e comunicação. E a mesma disponibilidade do código fonte que torna atraente sua escolha como ambiente de experimentação com injeção de falhas também lhe dá portabilidade entre desde equipamentos embarcados de aplicação específica de comunicação, passando por servidores, chegando até *clusters*, *grids*, etc. Projetos como *Carrier Grade Linux* [Open Source Development Labs, 2003], e o *Data Center Linux* [Open Source Development Labs, 2004] tem como objetivo empregar este sistema em ambientes onde a alta disponibilidade é essencial, como justamente serviços de telecomunicações e provimento de dados.

Definida a plataforma alvo ainda é necessário determinar a técnica e localização utilizadas na injeção de falhas. A injeção de falhas por software, além de mais barata, é mais apropriada quando se deseja trabalhar com estruturas de dados de mais alto nível do sistema operacional, como pacotes de comunicação. Quanto à localização, ela pode

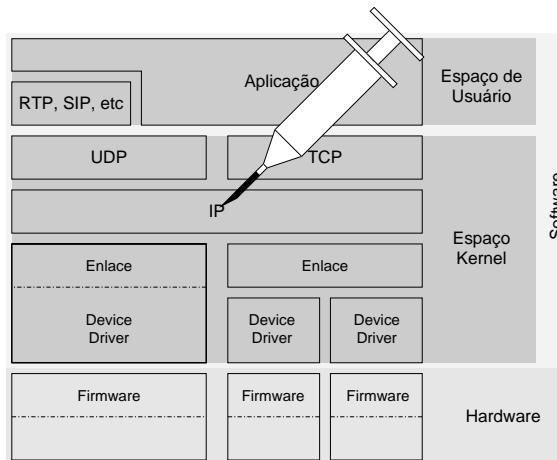


Figura 1: Localização da Injeção de Falha de Comunicações

ocorrer em qualquer ponto do trajeto percorrido pelas mensagens desde a aplicação de origem até a de destino, nas diversas camadas de comunicação.

A localização de um injetor de falhas será determinada pelo local no sistema operacional da implementação dos protocolos associados a cada uma das camadas de comunicação. Aplicações são implementadas como processos de usuário. Assim, para injetarmos falhas associadas ao nível de aplicação, o mais apropriado é através da modificação da aplicação. Há casos onde existem protocolos intermediários entre os níveis de aplicação e transporte, implementando ou funções dos níveis de sessão e apresentação, ou *middleware* com novas funções. Exemplos dessas camadas intermediárias são os protocolos RTP, XDR, SIP, ou ainda *middlewares* de comunicação de grupo ou de suporte a computação em *grid*. No Linux esses protocolos costumam ser implementados como bibliotecas de ligação dinâmica. Os protocolos de nível de transporte (UDP e TCP) são implementados internamente ao *kernel* e utilizados através de chamadas de sistema. Estes, por sua vez, utilizam o protocolo de rede IP, também implementado dentro do *kernel*, que utiliza os *drivers* de dispositivo para ter acesso aos diversos tipos de enlace. Finalmente, o nível físico é implementado através do *firmware* gravado diretamente no controlador de rede.

Se o objetivo de um injetor de falhas é emular as características de desempenho de rede para diversas tecnologias de acesso, sua implementação mais apropriada torna-se no nível IP, pois por suas características (serviço de datagrama, sem retransmissão) este protocolo simplesmente espelha as limitações dos níveis inferiores. A perda ou atraso de um quadro no nível de enlace implica no mesmo comportamento de um pacote IP, com a vantagem de termos a independência de tecnologia de acesso neste protocolo. Portanto, a modificação das dinâmicas de desempenho dos pacotes IP pode emular características dos protocolos inferiores, sendo o que mais se aproxima das falhas de comunicação dos níveis de acesso à rede. A relação entre as camadas e sua implementação no sistema operacional Linux pode ser observada na Figura 1.

Por ser a implementação do protocolo IP do Linux interna ao *kernel*, qualquer modificação que vise alterar o processamento de pacotes deve ter acesso às suas estruturas internas. A arquitetura de sistemas operacionais monolíticos isola processos de usuário

das estruturas do *kernel*: todo o acesso às suas funções é feito através de chamadas de sistema, e a memória a ele associada fica protegida de acessos dos processos. Para atuarmos sobre os pacotes dentro do *kernel*, portanto, devemos estar executando código interno ao mesmo. Uma possibilidade é a modificação do *kernel*, através da sua rescrita e nova compilação. Essa abordagem, utilizada em injetores como ComFIRM [Leite, 2000], apesar de obter excelentes resultados, possui o inconveniente da necessidade de adaptação dessas modificações a cada nova versão do *kernel*. É possível, ainda, utilizar as chamadas de sistema `ptrace()` que param a execução de processos a cada entrada e saída de outras chamadas de sistema. Ao parar o processo em chamadas de comunicação (recebimento e envio de dados) é possível omitir ou atrasar estas funções. Entretanto, essa técnica tem efeito considerável na carga do sistema, pois efetua trocas de contexto a cada chamada de sistema. Mesmo assim, esta técnica, utilizada dentre outras na ferramenta INFIMO [Barcelos et al., 2004], não necessita a modificação de código de sistema. Apresentamos a seguir um novo recurso para depuração do ambiente Linux, as sondas dinâmicas, que possibilita mais uma abordagem para a injeção de falhas.

4. Sondagem Dinâmica como Forma de Injeção de Falhas de Comunicação

Um mecanismo de sondas dinâmicas (DProbes, *dynamic probes*) foi recentemente incorporado ao Linux, estando disponível através de *patches* para a versão estável do *kernel* (2.6), com perspectivas de inclusão à versão oficial. As especificações *CARRIER GRADE LINUX* e *DATA CENTER LINUX* requisitam mecanismos de depuração dinâmica e para isto já utilizam este mecanismo. As sondas dinâmicas do Linux descendem diretamente de mecanismo semelhante existente no sistema operacional OS/2 da IBM (*Dynamic Trace*) [Moore, 2000], mas mecanismos de *trap* similares existem também em outros sistemas como z/OS, OS400, AIX e z/VM, embora não de forma tão extensiva e generalizada.

As sondas por software são inspiradas nas ponteiras de sondagem, dispositivos de hardware tipicamente usados para a monitoração e injeção de falhas em circuitos em funcionamento. Apesar de operarem por software, também apresentam baixa intrusão, pois permitem que o fluxo de um programa (tanto em espaço de usuário quanto de sistema) seja rápida e facilmente desviado para uma função de teste ou injeção de falhas. Essa baixa intrusão é interessante ao trabalharmos com protocolos, pois estes obedecem a regras temporais de operação entre os parceiros da comunicação. Além disso, a arquitetura do mecanismo implica em uma baixa dependência das características do sistema operacional hospedeiro, o que lhe garante portabilidade entre variantes de sistemas operacionais UNIX.

DProbes é um habilitador de outras tecnologias de depuração [Moore, 2001]. Seu principal componente é um mecanismo para a interceptação em qualquer ponto de execução (*probepoint*, ponto de sondagem). A cada ponto de sondagem é associada um tratador de sondagem (*probe handler*), que corresponde a uma função com acesso à memória do *kernel*, de usuário e aos registradores do processador.

O mecanismo usado é semelhante a um *watchpoint*, mas utiliza-se o termo *probe* porque a execução não é parada, apenas executa-se uma função pré-determinada. Por não exigir interatividade, sua implementação como um tratador de interrupção torna-se

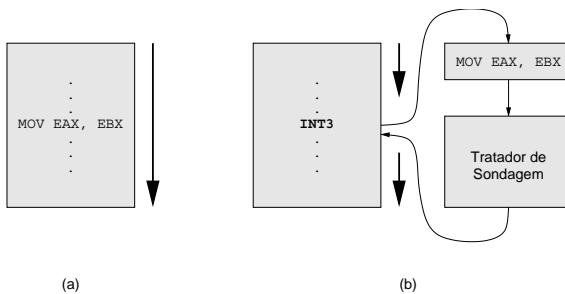


Figura 2: Código não modificado (a) e Implementação de pontos de sondagem na arquitetura IA32, através da substituição de instruções (b)

mais apropriada. Como interrupções chaveiam o processador para modo privilegiado, é possível ainda atuar em uma tarefa, interrupção ou mesmo troca de contexto. Um *probe* permite a observação e modificação do estado de um sistema em qualquer ponto do código sem intrusão significativa.

Em cada ponto de sondagem, DProbes substitui a instrução de máquina original por uma instrução de interrupção, como mostra a Figura 2 (b). Para tratar esta interrupção, ele executa isoladamente a instrução. Se esta executar sem a ocorrência de exceção, o controle é passado ao tratador de sondagem e em seguida retorna ao código original.

Ao colocar pontos de sondagem nas funções de envio de pacotes e no tratamento de interrupção do controlador de rede, um injetor de falhas de comunicação pode ser construído como uma aplicação do DProbes. Esta abordagem está sendo utilizada no desenvolvimento do novo ambiente de experimentação de falhas NEEDLE (*Network Experimentation Environment using DProbes for Link Errors*). Um exemplo de *probe* que intercepta o fluxo de execução sempre que uma mensagem é transmitida é mostrado na Figura 3.

5. Conclusões

As técnicas de injeção de falhas, já comuns para a validação dos mecanismos de detecção e recuperação de falhas, também podem ser aplicadas a emulação de cenários de redes de longa distância ou tecnologias sem fio. Isso facilita o teste de aplicações adaptativas, pois dispensa a necessidade de uma variedade de equipamentos de tecnologias distintas. Entretanto, a construção de um injetor de falhas por software pode ser localizada em diversos pontos de um sistema alvo de testes, e essa localização está relacionada aos níveis de rede onde se deseja injetar falhas.

Este artigo apresenta o mecanismo de depuração por sondas dinâmicas como nova técnica para a injeção de falhas de comunicação em ambiente Linux. A ferramenta NEEDLE implementada utilizando essa proposta pretende facilitar o trabalho de projetistas de protocolos e aplicações de rede, fornecendo uma plataforma útil e de fácil instalação para seus experimentos de validação.

Por utilizar o mecanismo de DProbes, NEEDLE implica em baixa intrusão, tanto em respeito ao desempenho das aplicações ou protocolos a serem validados, quanto à interferência no código do *kernel* utilizado nos testes, pois o injetor de falhas é construído

```
#include <linux/module.h>
#include <linux/kprobes.h>

struct kprobe kp;

void handler_post(struct kprobe *p, struct pt_regs *regs,
                  unsigned long flags) {
    printk("tratador de sondagem, dev_queue_xmit executando.");
}

int init_module(void)
{
    kp.pre_handler=NULL;
    kp.post_handler=handler_post;
    kp.fault_handler=NULL;
    kp.addr = (kprobe_opcode_t *) &dev_queue_xmit;
    register_kprobe(&kp);
    printk("probe instalado\n");
    return 0;
}

void cleanup_module(void)
{
    unregister_kprobe(&kp);
    printk("probe removido\n");
}
```

Figura 3: Exemplo de probe. Uma mensagem é exibida a cada execução da função `dev_queue_xmit()`

como um *plug-in* de um *kernel* não modificado.

Referências

- [Barcelos et al., 2004] Barcelos, P. P. A., Drebes, R. J., Jacques-Silva, G., e Weber, T. S. (2004). A toolkit to test the intrusion of fault injection methods. In *LATW 2004 Digest of Papers*, páginas 152–157, Cartagena de Indias, Colômbia. IEEE.
- [Leite, 2000] Leite, F. O. (2000). ComFIRM – injeção de falhas de comunicação através da alteração de recursos do sistema operacional. Dissertação de Mestrado, Instituto de Informática/UFRGS, Porto Alegre, Brasil.
- [Moore, 2000] Moore, R. J. (2000). Dynamic probes and generalised kernel hooks interface. In *Proceedings of the 4th Annual Linux Showcase*.
- [Moore, 2001] Moore, R. J. (2001). A universal dynamic trace for Linux and other operating systems. In *Proceedings of the 2001 USENIX Annual Technical Conference*. USENIX.
- [Open Source Development Labs, 2003] Open Source Development Labs (2003). Carrier grade Linux requirements definition version 2.0. Disponível em: http://www.osdl.org/docs/carrier_grade_linux_requirements_definition_version_20.pdf. Acesso em: Jun 2004.
- [Open Source Development Labs, 2004] Open Source Development Labs (2004). Data center Linux technical capabilities: Executive summary version 1.0. Disponível em: http://www.osdl.org/lab_activities/data_center_linux/DCL_ExecSumm_TechCapabilities_1_0.pdf. Acesso em: Jun 2004.

Uma Arquitetura de Injetor de Falhas Orientada a Aspectos para Validação de Sistemas de Comunicação de Grupo

Karina Kohl Silveira, Taisy Weber

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

{kohl, taisy}@inf.ufrgs.br

Resumo: Sistemas de comunicação de grupo são blocos de construção eficientes para o desenvolvimento de sistemas distribuídos com características de dependabilidade. A injeção de falhas é uma abordagem que permite acelerar a ocorrência de erros e defeitos em um sistema para que seja possível a validação das suas propriedades de dependabilidade, assim como a avaliação do impacto dos mecanismos de detecção e remoção de erros no desempenho do sistema. Esse artigo apresenta uma arquitetura de injetor de falhas baseada em orientação a aspectos para a validação de sistemas de comunicação de grupo. É mostrado como se pode evitar a intrusão espacial do injetor de falhas, utilizando-se o paradigma de orientação a aspectos.

1. Introdução

Um sistema tolerante a falhas caracteriza-se pela capacidade de fornecer o serviço especificado, mesmo na ocorrência de falhas. No desenvolvimento destes sistemas, surgem problemas relacionados à validação dos mesmos e além das suas funcionalidades normais, os mecanismos de tolerância a falhas devem ser testados. A injeção de falhas é uma técnica de validação e pode ser definida como a introdução intencional de falhas em um sistema para observar seu comportamento [ARL90].

Neste trabalho pretende-se apresentar uma alternativa para injeção de falhas utilizando-se recursos do paradigma de orientação a aspectos [KIC97]. Uma tecnologia relativamente nova, baseada em reflexão computacional e que surgiu para aumentar a expressividade do paradigma de orientação a objetos. A opção por aspectos surgiu ao se observar à possibilidade de evitar a intrusão espacial do injetor de falhas. Aspectos possibilitam a interceptação e até mesmo modificação de atributos, construtores e métodos do programa a ser validado sem alteração do código fonte.

O principal objetivo deste artigo é propor a injeção de falhas baseada no paradigma orientado a aspectos para a validação de mecanismos de *membership* de um protocolo de comunicação de grupo. Esse tipo de protocolo trabalha com visões de um grupo e quando é identificado *crash* ou alguma falha de comunicação, o membro é isolado e é criada uma nova visão sem a presença do membro com problemas. O injetor se baseia na possibilidade de interceptação das mensagens enviadas entre os membros do grupo, simulando falhas de comunicação e *crash*, sem a necessidade de alterar o código fonte do protocolo de *membership*. As falhas injetadas permitem verificar a criação de novas visões refletindo a exclusão de membros.

Entre ferramentas de injeção de falhas de comunicação podem ser citadas CSFI, ORCHESTRA, ComFIRM e a extensão da ferramenta Jaca. A ferramenta CSFI

(*Communication Software Fault Injection*) [CAR95] foi uma das primeiras desenvolvidas para injeção de falhas de comunicação e o objetivo principal era o de avaliar o impacto de falhas em sistemas paralelos. A versão existente de CSFI foi implementada para um sistema *transputer* T805. ORCHESTRA [DAW96] foi desenvolvido para teste de dependabilidade de protocolos distribuídos. A ferramenta ComFIRM (*Communication Fault Injection through OS Resources Modification*) [BAR2000] propõe-se a injetar apenas falhas de comunicação e fica situada no núcleo do sistema operacional Linux. A ferramenta Jaca, foi estendida recentemente para permitir falhas de comunicação [JAC2004] e é utilizada para a validação de aplicações orientadas a objetos escritas em Java. O maior objetivo de Jaca é de injetar falhas utilizando características de programação de alto nível durante a execução do programa, corrompendo valores de atributos, parâmetros de métodos ou valores de retorno. Jaca está baseada em reflexão computacional. A proposta deste artigo diferencia-se de Jaca por usar orientação a aspectos.

Este artigo está organizado da seguinte forma: a seção 2 apresenta alguns conceitos relacionados ao artigo. Na seção 3 é apresentada a injeção de falhas baseada em *software*, uma arquitetura genérica para injetores de falhas e a arquitetura baseada em aspectos com seu modelo de falhas, as regras de injeção de falhas e uma discussão sobre intrusividade.

2. Conceitos Relacionados

A injeção de falhas pode ser definida como a introdução intencional de falhas em um sistema para observar seu comportamento [ARL90]. Acelerar a ocorrência de erros e defeitos é uma abordagem eficaz para a validação das propriedades de dependabilidade de um sistema assim como para avaliar o impacto dos mecanismos de detecção e remoção de erros no desempenho do sistema.

O tipo de injeção de falhas apresentada nesse trabalho é conhecido como SWIFI (*Software-implemented fault injection*). Nessa abordagem as falhas são injetadas no *software* através de *software*, corrompendo o código ou os dados das aplicações. Falhas são injetadas durante a execução da aplicação alvo e um monitor associado ao injetor fornece dados sobre o comportamento desse sistema em presença das mesmas. A análise desses dados indica se o sistema atende à especificação, ou seja, se realmente mascara ou recupera-se das falhas a que se propôs na fase de projeto [HSU97].

A interferência da injeção de falhas na aplicação alvo, também chamada de perturbação ou intrusão, pode ser temporal e espacial. Na intrusão temporal o tempo de execução é aumentado devido às atividades do injetor de falhas junto à aplicação alvo. Já a intrusão espacial, refere-se à modificação do código da aplicação alvo [BAR2001].

Sistemas de comunicação de grupo [GAR99] são blocos de construção poderosos para o desenvolvimento de sistemas distribuídos tolerantes a falhas. A comunicação de grupo é um meio de oferecer comunicação multiponto a multiponto, pela organização dos processos em grupos. Um sistema de comunicação de grupo orientado a visões, provê serviços de *membership* e também de *multicast* confiáveis. Um serviço de *membership* mantém uma lista dos processos de um grupo que estão correntemente ativos e conectados. A saída de um serviço de *membership* é chamada de visão. Os serviços de *multicast* confiáveis entregam as mensagens aos membros da visão atual.

A programação orientada a aspectos é um paradigma de programação relativamente novo, introduzido por Kickzales [KIC97], e é baseado em decomposição aspectual. A decomposição aspectual complementa a decomposição funcional e tenta superar a limitação de decomposição funcional de capturar e representar funcionalidades que cruzam o sistema. Após a separação do sistema em construções funcionais, a decomposição aspectual é

aplicada ao projeto para se capturar interesses cruzados [PAP2004]. Um aspecto é uma unidade modular de implementação cruzada. O aspecto encapsula comportamentos que afetam várias classes em módulos reusáveis.

Linguagens orientadas a aspectos utilizam cinco elementos principais para modularizar *crosscutting concerns*: *join points*, *pointcuts*, *introductions*, *advices* e *aspects*. Um *join point* é um ponto bem definido no fluxo do programa (por exemplo, chamadas de métodos e construtores). Um *point cut* seleciona um *join point* em particular filtrando um subconjunto de todos *join points* baseados em critérios definidos. Os critérios podem ser nomes explícitos de funções ou nomes de funções especificados por coringas. As *introductions* permitem que novos métodos ou campos sejam inseridos a uma classe existente. Um *advice* é usado para definir o código adicional que deve ser executado nos *join points*.

3. Injeção de Falhas baseada em Aspectos

O injetor proposto é implementado como sendo um aspecto do sistema de comunicação de grupo e onde o principal *concern* é efetivamente a comunicação, pois o injetor irá trabalhar com a interceptação das chamadas de envio e recepção de mensagens.

O objetivo é a validação do comportamento do sistema de comunicação de grupo na presença de falhas de comunicação e *crash*. O resultado esperado é que o serviço de *membership* instale uma nova visão quando perceber que um determinado membro do grupo não está se comportando da forma esperada, isto é, reconheça a falha injetada como sendo uma situação errônea e “cumpra” sua especificação.

Como o paradigma de orientação a aspectos trabalha no meta nível, o injetor proposto permite que seja definido um roteiro de injeção de falhas sem que exista a necessidade de análise ou alteração do código fonte da aplicação.

3.1. Arquitetura

A arquitetura proposta é baseada na arquitetura genérica sugerida por Hsueh [HSU97]. O ambiente é basicamente constituído de um sistema alvo, um injetor de falhas, uma biblioteca de falhas, um gerador de carga, uma “biblioteca de carga”, um controlador, um monitor, um coletor de dados e um analisador de dados.

O injetor de falhas injeta falhas no sistema alvo assim como executa comandos enviados pelo gerador de carga. O monitor rastreia a execução dos comandos e inicia a coleta de dados quando necessária. O coletor de dados realiza a coleta *on line* de dados e o analisador de dados, que pode estar *off line*, realiza o processamento de dados e a análise. O controlador controla o experimento. Em função desse modelo, será apresentada a arquitetura para o injetor proposto.

Usando o paradigma de orientação a aspectos e seus elementos básicos, foi possível o agrupamento de alguns elementos da arquitetura proposta por Hsueh. Mais de uma funcionalidade de cada elemento de Hsueh se encontra em um único elemento da orientação a aspectos. A Figura 1 apresenta a arquitetura para o injetor de falhas baseado em programação orientada a aspectos.

Como o objetivo do injetor é a validação de um sistema de comunicação de grupo, então o gerador de carga para o sistema alvo, nada mais é que qualquer aplicação distribuída construída sobre o sistema de comunicação de grupo a ser validado. Por esse motivo, o gerador não está representado na arquitetura, mas sim externamente a ela.

O controlador e o injetor de falhas de Hsueh se tornaram um único elemento. Esse será implementado como um elemento *advice* da orientação a aspectos. Como já foi citado, um *advice* é responsável pelo disparo da lógica adicional que será executada. Além disso, é o *advice* que reconhece o momento que um determinado método deve ser interceptado. Portanto essas duas funcionalidades cumprem os papéis delegados ao controlador e ao injetor, pois é o controlador que é responsável por identificar o momento da injeção e o injetor deve realizar a injeção.

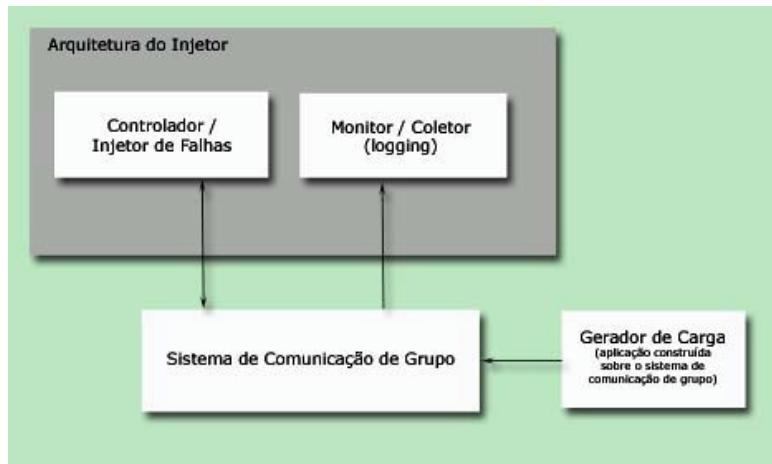


Figura 1 – Arquitetura do injetor baseado em aspectos

O monitor e o coletor também foram agrupados em um único elemento. O monitor é responsável por identificar que uma falha será injetada e dispara o coletor, que irá realizar o *log* das injeções realizadas. O monitor/coletor será implementado como um único *advice*, pelo mesmo motivo que o controlador e o injetor foram. A atividade de *log* é um dos principais exemplos citados na literatura para o uso de orientação a aspectos.

3.2. Modelo de Falhas

Aplicações que utilizam técnicas de tolerância a falhas são construídas para suportar um dado conjunto de falhas, dentro do modelo de falhas especificado. Portanto, considerando a técnica de injeção de falhas, é importante a identificação do modelo de falhas tolerado para que o procedimento de validação tenha sentido. No contexto desse trabalho, o escopo de falhas a ser utilizado compreende as falhas de comunicação, descritas abaixo e definidas por Cristian [CRI91]:

- Falha de *crash* de processo: ocorre uma parada prematura do processo no sistema. A partir da falha, o processo não realiza qualquer função. Antes da ocorrência da falha o processo apresentava comportamento correto.
- Falha por omissão de envio: um processo falha por intermitentemente omitir o envio de mensagens;
- Falha de temporização: um processo falha pela violação do limite de tempo exigido para a execução de uma determinada função (por exemplo, não enviar o *ack* de recebimento de uma mensagem).

4. Intrusividade

A intrusividade é uma característica indesejada, pois pode influenciar nas medições obtidas e gerar resultados que não condizem com a realidade. Um injetor de falhas com uma

intrusão muito grande no sistema pode mascarar erros e defeitos. Quanto menor a intrusão gerada pelo injetor, melhores e mais corretos os resultados obtidos.

O injetor proposto não apresenta intrusão espacial no código do sistema alvo. O paradigma de orientação a aspectos permite que os métodos do sistema sejam interceptados e então seja adicionada a lógica adicional para alteração de métodos e campos. Porém essa lógica não está no código fonte da aplicação, que não precisa sequer ser conhecido, e sim no código do injetor de falhas e será ativado em tempo de execução. Porém a intrusão temporal não é possível de ser evitada, pois código adicional tem que ser executado.

5. Metodologia

O injetor será acionado toda a vez que alguma rotina de comunicação for chamada. Por exemplo, se um membro do grupo enviar uma mensagem, a chamada de envio poderá ser interceptada e atrasada, simulando uma falha de temporização, ou simplesmente não ser enviada, simulando uma falha de omissão ou de *crash*. Nesse caso as chamadas aos métodos de envio e recepção correspondem ao conceito de *joint point* na orientação a aspectos.

A figura do *advice* no paradigma orientado a aspectos é o que irá permitir a execução da lógica de injeção de falhas. É o comportamento que será inserido nos *join points*. Além disso, as *introductions* permitirão que novos métodos ou campos sejam inseridos a uma classe existente, sendo útil no momento de interceptação de mensagens, podendo corrompê-las.

Um dos principais serviços de um sistema de comunicação de grupo é o *membership* que comprehende o controle das visões, pois são elas que mantêm a integridade do sistema distribuído, evitando que membros falhos interfiram na computação. As visões contêm os identificadores de todos os membros ativos e conectados em um determinado momento e quando algum membro falha o serviço de *membership* identifica e isola o membro falho, instalando uma nova visão. Para validar se o sistema de comunicação de grupo realmente atua da forma especificada, o injetor irá interceptar chamadas de envio ou recepção de mensagens e fazer com que não seja entregues ou recebidas, o que será entendido pelo serviço de *membership* como uma falha no membro que enviou a mensagem ou que não recebeu a mensagem. Porém não é aconselhável que todas as mensagens sejam alvo do injetor, o que impossibilitaria a execução normal do programa afetando até mesmo o experimento de injeção de falhas. O modelo de disparo das falhas será tratado no próximo tópico.

5.1. Disparo das Falhas

O disparo ou a ativação das falhas deve estar relacionado a alguma condição que possa ser definida em termos espaciais ou temporais. Uma falha disparada espacialmente torna-se ativa quando o sistema alcança determinado estado. Por exemplo, uma falha pode ser ativada após a quinta mensagem ser enviada. Uma falha disparada temporalmente deve ser ativada após um determinado período de tempo [BAR2001].

A partir disso, optou-se pelo disparo espacial de falhas, realizado em três momentos:

- Quando um determinado número (gerado aleatoriamente) de membros estiver presente na visão, permitindo a simulação de *crash* ao atingir o número de membros definido;
- Quando um contador de mensagens enviadas ou recebidas por um determinado membro atingir um número gerado aleatoriamente, pode-se omitir a próxima mensagem a ser

enviada simulando uma falha de omissão ou atrasar a confirmação da próxima mensagem recebida, simulando uma falha de temporização;

- Quando um contador de mensagens enviadas por um determinado membro atingir um valor fixo, permitindo a inserção de falhas de forma determinística, de crash, omissão ou temporização, caso nenhum dos casos anteriores seja atingido.

6. Considerações Finais

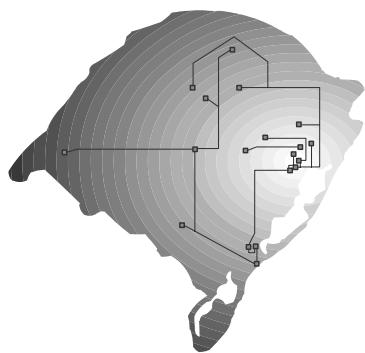
O artigo apresenta uma arquitetura para um injetor de falhas baseada no paradigma de orientação a aspectos. Esse injetor tem como objetivo a validação de sistemas de comunicação de grupo, que são ferramentas valiosas para a construção de sistemas distribuídos que necessitam de características de dependabilidade. O injetor segue a filosofia da abordagem SWIFI, onde um software injeta falhas no sistema alvo, corrompendo código ou dados. O modelo de falhas que a arquitetura leva em consideração são as falhas de *crash*, omissão e temporização.

A principal vantagem oferecida pela orientação a aspectos na construção de um injetor de falhas é a possibilidade de interceptação e alteração de métodos de um sistema, em tempo de execução, sem a necessidade de alteração do código original da aplicação. Dessa forma, é possível eliminar a intrusão espacial do injetor de falhas no código do sistema alvo. Essa propriedade se torna ainda mais importante quando o código da aplicação não está disponível, impossibilitando alterações e recompilação.

Como última consideração, é proposto como trabalho futuro a extensão da arquitetura baseada em aspectos apresentada nesse artigo, para que seja utilizada na validação de outros sistemas, distribuídos ou não, além dos sistemas de comunicação de grupo.

Referências

- [ARL90] ARLAT, J. Et al. *Fault Injection for dependability Validation: a methodology and some applications*. IEEE Transactions on Software Engineering, v. SE-16, n.2, Fevereiro 1990.
- [BAR2001] BARCELOS, P.P.; WEBER, T. *INFIMO – Um Toolkit para Experimentos de Intrusão de Injetores de Falhas*. Tese de doutorado. UFRGS. 2001.
- [CAR95] CARREIRA, J., SILVA, J.G *Assessing the Effects of Communication Faults on Parallel Application*. Proceedings of IPDS'95. Abril 1995.
- [CRI91] CRISTIAN, F. *Understanding Fault-Tolerant Distributes Systems*. Communication of the ACM, v.34, n.2, Fevereiro 1991.
- [DAW96] DAWSON, S. JAHANIAN, F., MITTON, T. *ORCHESTRA: A Probing and Fault Injection Environment for Testing Protocol Implementation*. Proceedings of IPDS'96. Setembro 1996.
- [HSU97] HSUEH, M., TSAI, T., IYER, R. *Fault Injection Techniques and Tools*. IEEE Computer, v. 30, issue 4, Abril1997.
- [JAC2004] JACQUES, G., MORAES, R., WEBER, T., MARTINS, E. *Validando sistemas distribuídos desenvolvidos em Java utilizando injeção de falhas de comunicação por software*. V Workshop de Testes e Tolerância a Falhas. Maio 2004
- [KIC97] KICKZALES, G. *Aspect Oriented Programming*. Proceedings of the European Conference on Object-Oriented Programming. Junho 1997
- [PAP2004] PAPAPETROU, O., PAPADOPOULOS, G. *Aspect Oriented Programming for a component-based real life application: A case study*. Proceedings of the 2004 ACM Symposium on Applied Computing. Março 2004.



Sessão Técnica 3

Gerência e Operações de Redes II

Uma proposta de extensão de um *chatterbot* visando auxiliar na capacitação de profissionais de gerência de redes

Michelle Denise Leonhardt¹, Leandro Marcio Bertholdo², Liane Margarida Rockembach Tarouco¹

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

²Ponto de Presença da RNP no Rio Grande do Sul (POP-RS)
Rua Ramiro Barcelos, 2574 – Porto Alegre – RS – Brazil

{mdleonhardt, liane}@inf.ufrgs.br, berthold@penta.ufrgs.br

Resumo. Com o crescimento do número e da heterogeneidade dos equipamentos presentes nas atuais redes de computadores, o gerenciamento eficaz destes recursos torna-se crítico. O que se verifica, porém, é que nem sempre profissionais treinados e com vasta experiência atuam nas redes de pequenas corporações. Muitas vezes estagiários são designados para monitorar equipamentos e redes, principalmente em turnos onde os profissionais mais qualificados não desejam permanecer trabalhando. Observa-se, então, a necessidade de uma alternativa inovadora que seja capaz de suprir as necessidades de treinamento e economia de tempo de um profissional menos capacitado que atue na área de gerenciamento de rede. Este trabalho tem como objetivo propor uma solução, através do uso de chatterbots, para o problema da falta de capacitação e treinamento de alguns profissionais que atuam na área.

1. Introdução

As redes de computadores nasceram e evoluíram a fim de facilitar a troca de dados, informações e serviços entre usuários e entidades separadas. O gerenciamento de redes pode ser definido como a prática de monitorar e controlar uma rede, de modo que ela corresponda às expectativas de seus usuários; o planejamento de ampliações ou modificações na rede, a fim de suprir a demanda nas operações da rede e a incorporação de novos elementos na rede sem interferir nas operações já existentes [Lewis 1995].

Em geral, as atividades básicas do gerenciamento de redes envolvem o controle e administração de forma racional dos recursos de hardware e software em um ambiente distribuído buscando melhor desempenho, eficiência e segurança do sistema. Assim, o gerenciamento tem por objetivo maximizar o controle organizacional das redes de computadores, de maneira mais eficiente e confiável, ou seja, planejar, supervisionar, monitorar e controlar qualquer atividade da rede.

Em sua pesquisa, Biesczad [Biesczad et. Al. 1998] salienta que um dos principais pontos da atividade de gerenciamento é que possíveis falhas ou comportamentos incoerentes devem ser diagnosticados rapidamente e solucionados automaticamente ou através de um operador preparado para tomar as providências necessárias. O que se observa, porém, é que nem sempre profissionais treinados e com vasta experiência atuam nas redes de pequenas corporações. Muitas vezes estagiários são designados para monitorar equipamentos e redes, principalmente em turnos onde os profissionais mais qualificados não desejam permanecer trabalhando.

Observa-se, então, a necessidade de uma alternativa inovadora que seja capaz de suprir as necessidades de treinamento e economia de tempo de um profissional menos capacitado que atue na área de gerenciamento de rede. Tais profissionais necessitam obter informações de forma simples, até que possam se tornar suficientemente seguros de si e capacitados para tomarem suas próprias decisões. Nesse contexto, a solução proposta neste artigo busca suprir tal necessidade, uma vez que ajudaria o usuário a encontrar problemas em uma rede de forma natural, bem como esclareceria dúvidas sobre o comportamento de uma rede sem a necessidade de um conhecimento detalhado da mesma.

O que se propõe como trabalho, enfim, é a utilização de uma interface em linguagem natural, ou seja, um *chatterbot*, capaz de auxiliar no gerenciamento de redes. Gerentes e operadores de rede com pouca experiência podem não utilizar adequadamente os dados coletados de uma rede. Assim, um *chatterbot* pode servir como um interpretador de tais dados e fonte de consulta para solução e localização de problemas em uma rede, tentando reaplicar o papel de um gerente de rede mais treinado e capacitado.

O *chatterbot* proposto pode proporcionar ao usuário um mecanismo para a aprendizagem significativa. Na aprendizagem significativa o aprendiz constrói seu próprio conhecimento, ou seja, não atua como um receptor passivo. Ele deve fazer uso dos significados que já internalizou para poder captar os significados dos materiais educativos. Assim, ao mesmo tempo que está progressivamente diferenciando sua estrutura cognitiva, está também identificando semelhanças e diferenças e, consequentemente, reorganizando seu conhecimento.

O artigo está estruturado da seguinte forma: na seção 2 apresenta a arquitetura proposta para extensão de um *chatterbot* que tem por objetivo auxiliar no treinamento de profissionais que atuam na área de gerência de redes. A seção 3 mostra o trabalho desenvolvido com Elektra, um *chatterbot* também baseado no ALICE que tem sido utilizado para análise do diálogo entre usuário e máquina e para auxiliar na educação a distância. Finalmente, a seção 4 encerra o trabalho com as conclusões levantadas e os trabalhos a serem desenvolvidos no futuro.

2. Solução Proposta

A arquitetura da solução proposta é apresentada na figura 1 e será detalhada no decorrer do artigo.

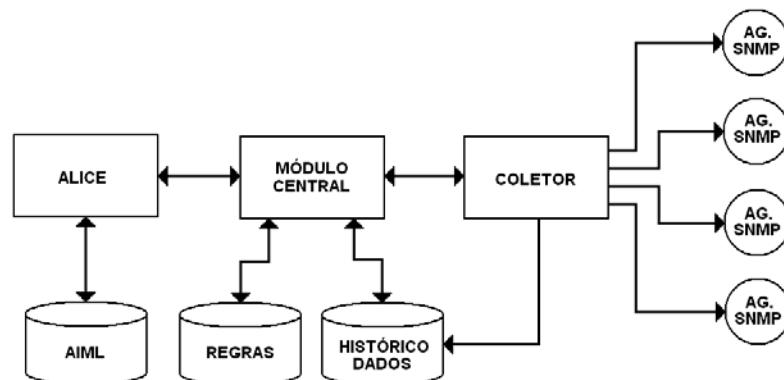


Figura 1. Arquitetura Proposta

2.1. Chatterbots, ALICE e a linguagem de marcação AIML

Chatterbots são, por definição, programas que procuram simular uma conversação, em linguagem natural, com o objetivo de levar o interlocutor a pensar que está falando com outro ser humano. Embora não seja este o objetivo do sistema a ser desenvolvido, ele se enquadra em tal definição, já que é capaz de manter um diálogo tal qual um ser humano o faz. O estudo dos *chatterbots* engloba diversas áreas de pesquisa como lingüística computacional, interação homem-computador e inteligência artificial. Sua utilização na educação tem sido bastante explorada e tem se mostrado extremamente eficaz [Medina and Tarouco 2003]. Por esta razão, pode se tornar um recurso bastante interessante para o treinamento de profissionais para atuarem no gerenciamento de redes de computadores. Mais adiante será descrito resumidamente o trabalho realizado com um *chatterbot* desenvolvido na UFRGS que é utilizado para o ensino de conceitos básicos de redes de computadores [Leonhardt et al. 2003].

O *chatterbot* a ser utilizado como base neste trabalho é o ALICE (*Artificial Linguistic Internet Computer Entity*) [Wallace 1995]. Sua escolha pode ser justificada pelo fato de que é um dos *bots* mais utilizados na atualidade, é gratuito e parte integrante do projeto GNU e é ponto de partida para diversos projetos que procuram agregar mais funcionalidades ao sistema [Galvão et al. 2003].

A base do conhecimento e comportamento de ALICE é construída através da linguagem de marcação AIML (*Artificial Intelligence Markup Language*) [Taylor 2003], uma das filhas da linguagem XML (*eXtensible Markup Language*). O AIML é uma linguagem de fácil aprendizagem e utilização. Ela apresenta um conjunto de *tags* e comandos simples para implementação da base de conhecimento de um *chatterbot* e serve para analisar as mensagens enviadas pelo usuário e decidir a forma como estas mensagens devem ser respondidas.

2.2. SNMP

Utilizando SNMP e MIB (RFC 1442 - *Structure of Management Information for version 2 of the Simple Network Management Protocol - SNMPv2*), pode-se capturar os objetos e, a partir daí, fazer verificações para auxiliar no gerenciamento da rede. A fim de entender melhor o funcionamento da rede e da própria MIB, a tabela 1 sintetiza alguns problemas que podem ocorrer e os objetos que podem ser usados para verificar tais problemas, tendo como referência a MIBII (RFC 1213 - *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*).

Table 1. Exemplos de Objetos gerenciáveis da MIBII (RFC1213)

Objeto	Categoría FCAPS	Descrição	Condição Alerta
ifOperStatus	Fault Performance	Representa o estado corrente/atual da interface. O valor 3 indica que nenhum pacote pode ser enviado.	Se o estado operacional da interface divergir do estado desejado, pode-se disparar uma condição de alerta, pois existem interfaces em desacordo com o estado desejado da mesma.
ifInDiscards	Fault Performance	Quantidade de pacotes sem erros que foram descartadas (entrada)	Quando muitos pacotes são descartados, pode haver problemas de congestionamento na rede
ifOutDiscards	Fault Performance	Quantidade de pacotes sem erros que foram descartadas (saída)	Quando muitos pacotes são descartados, pode haver problemas de congestionamento na rede

2.3. Módulo Coletor e Histórico de Dados

O módulo coletor é o responsável pela coleta de informações da rede e armazenamento no banco de dados do histórico de informações. Ele é programado para disparar consultas

(através de SNMP) sobre o estado da rede em determinados intervalos de tempo, podendo ser utilizado também fora de tais intervalos, quando solicitado pelo módulo central.

A constante coleta de informações sobre a rede e o armazenamento das mesmas em um banco de dados pode ser muito útil para que o *chatterbot* seja capaz de responder algumas perguntas relevantes para o gerenciamento de redes, como por exemplo, o caso de ataques DoS ou de possíveis falhas em algum equipamento. Isso ocorre porque o comportamento normal da rede é medido através do histórico de suas informações e não apenas de observação momentânea.

2.4. Módulo Central e Banco de Regras

O módulo central é o responsável por relacionar informações recebidas do banco de dados ou de consultas momentâneas para enriquecer as respostas do *chatterbot* em termos mais práticos. Assim, ele é o responsável por receber a informação solicitada pelo usuário e gerar as respostas dinâmicas do sistema.

Para que tais respostas sejam geradas, é necessária a criação de um banco de regras que deve auxiliar na escolha da melhor resposta pelo módulo central. As regras, por suas vez, são extraídas de um conjunto de casos previamente observados em redes em funcionamento ou de documentação de trabalhos na área [Melchiors 1999] [Gaspari and Fagundes 2003].

Quando um usuário faz uma pergunta ao *chatterbot*, o módulo central é disparado e é responsável por tomar a atitude necessária. Ele pode solicitar ao módulo coletor uma consulta momentânea a um objeto ou ainda buscar informações no banco de dados para poder fornecer a resposta que tende a se aproximar mais do que se deseja saber. As regras são implementadas através da atribuição de pesos aos casos observados e cálculo de probabilidade de resposta. O módulo central também pode utilizar ferramentas como *ping* e as demais descritas em [Medina 1994] para elaborar sua resposta.

A integração do módulo central com o *chatterbot* ALICE pode ser feita de duas formas. A primeira delas envolve a utilização da tag <system>. Esta tag é responsável por chamadas a programas executáveis externos e sua utilização pode ser ilustrada através de outros sistemas que utilizam tal recurso [Maghsoudi and Arthanari 2004]. A outra forma é através da extensão do próprio código fonte do *chatterbot* ALICE, através da criação de novas tags para chamadas externas que atendam a necessidades específicas de parâmetros ou demais informações pertinentes.

Cabe salientar que o objetivo do sistema não é o de resolver e nem diagnosticar problemas. O *chatterbot* proposto apenas serve como ponto de partida para ensinar ao usuário os possíveis problemas que podem ocorrer na rede e como o mesmo deve proceder em caso de ocorrência de tais problemas. Ele deve informar, além do comportamento da rede em relação ao que foi perguntado e de uma explicação teórica sobre o assunto, os objetos que consultou para verificar as condições e o que se pode fazer para resolver algum eventual problema, quando for o caso.

3. Elektra: Ensinando redes de computadores através de linguagem natural

A Profª Elektra (disponível para visitação em <http://penta3.ufrgs.br:2002/>) foi criada na Universidade Federal do Rio Grande do Sul baseada em estudos sobre Inteligência artificial na educação e no já existente *chatterbot* ALICE. Ela visou inicialmente responder perguntas sobre física para alunos do ensino secundário que estivessem se preparando para o vestibular mas depois sua utilização foi estendida para os alunos do Curso de Especialização a Distância

em Informática na Educação, acrescentando em sua base de conhecimento dados e conceitos sobre Redes de Computadores e Internet.

Para este trabalho foram exploradas diversas funcionalidades do AIML. A primeira funcionalidade explorada foi a possibilidade do *chatterbot* escolher uma entre diversas respostas a serem apresentadas para uma mesma pergunta. Isto se dá através da adição de mais de uma resposta para uma categoria de conhecimento, que é escolhida aleatoriamente no momento em que o robô é solicitado, oferecendo assim, a oportunidade de que, ao ser questionado mais de uma vez sobre o mesmo assunto, o *chatterbot* possa apresentar uma resposta diferente da primeira. Este recurso parte do pressuposto que se o usuário do ambiente tornou a questionar o robô sobre um mesmo tópico é porque provavelmente a resposta apresentada não satisfez sua necessidade. É importante ressaltar que o robô não necessariamente apresenta uma resposta diferenciada se o assunto for persistido, visto que as respostas são escolhidas de forma aleatória, podendo assim a mesma resposta ser apresentada novamente.

O uso de respostas modeladas combinadas com as várias funcionalidades do AIML pareceram, em primeiro momento, suficientes pois se poderia prevenir uma diversidade de respostas, trazer ilustrações e direcionar o usuário a *websites* dedicados ao assunto trabalhado. Entretanto, para avaliar a qualidade das interações, em um segundo momento, resolveu-se analisar as respostas dos alunos verificando os arquivos de *log* gerados pelo servidor nas interações dos alunos.

Analizando estes arquivos, um dos problemas observados foi que as respostas do *chatterbot* apresentavam-se muito diretas e objetivas, de forma que o mesmo não estimulava a continuação do diálogo. Observou-se como problema adjunto as diferentes formas de comunicação expressas pelos alunos nas suas interações com o *chatterbot*, visto que estes eram de regiões diferentes do Brasil. Assim o robô muitas vezes era portador do conhecimento questionado pelo aluno mas não apresentava a resposta correspondente, uma vez que não reconhecia a pergunta.

Para solucionar tal problema, iniciou-se a tarefa de correções e aperfeiçoamentos no *chatterbot*. A primeira etapa foi melhorar as respostas, deixando Elektra aparentemente mais amigável e interessada no assunto ao qual estivesse tratando com o aluno, de forma a estimular a continuidade do diálogo. Nesta etapa, foram revisadas as respostas que o robô fornecia, acrescentando assim no final de algumas das respostas perguntas e outras informações ao usuário, ou pedidos para que questionasse mais o *bot*.

A segunda etapa demandou maior atenção pois dizia respeito as diferentes formas de se perguntar a mesma coisa ao robô. Através de uma análise de diálogos extraiu-se cada pergunta que não estava registrada na base de conhecimento do *bot*. Como na maioria das vezes o robô já tinha uma resposta pronta, adicionaram-se então estas perguntas de maneira que ele apresentasse as respostas já existentes sobre aquele assunto.

4. Conclusões e Trabalhos Futuros

Este artigo veio apresentar uma proposta de uma solução utilizando chatterbots para suprir principalmente a necessidade de treinamento e capacitação de alguns profissionais que atuam no gerenciamento de redes de computadores.

A continuidade do trabalho envolve a finalização da implementação propriamente dita e a validação, testes e avaliação da solução através de seu uso na rede do POP-RS. Para fins de avaliação de uso da tecnologia do *chatterbot*, estão sendo feitas pesquisas através da utilização do *chatterbot* Elektra [Leonhardt et al. 2003]. Este *chatterbot* possui respostas

estáticas para perguntas sobre redes de computadores e se mostrou uma boa opção de integração em um ambiente virtual de ensino com o mesmo fim. [Medina and Tarouco 2003].

Ao final do trabalho, espera-se obter um *chatterbot* capaz de suprir as dificuldades e inseguranças que novos profissionais costumam encontrar quando atuam no gerenciamento de redes. Espera-se ainda validar a utilização de uma interface em linguagem natural como meio de comunicação com a máquina em um ambiente de redes de computadores, onde costuma ocorrer a integração de diversas ferramentas e possibilidades de reação a problemas.

Desta forma, um usuário pode observar uma rede e seu comportamento sem a necessidade do aprendizado de ferramentas específicas e linguagens complicadas. Quando o mesmo estiver mais ciente do comportamento de uma rede, pode, com tempo, habituar-se na utilização da ferramenta que melhor se adapte as suas necessidades.

5. Referências

- ALICE (1995) - Artificial Linguistic Internet Computer Entity The A.L.I.C.E A.I. Foundation Disponível em: <<http://alicebot.org>>, Acesso em: 12.08.2003.
- Biesczad, A.; Pagurek, B.; and White, T. (1998) Mobile Agents for Network Management, IEEE Communications Surveys, Vol. 1, No. 1, Fourth Quarter.
- Galvão, A. M., Neves, A., Barros, F. A. (2003) Persona-AIML: Uma Arquitetura para Desenvolver Chatterbots com Personalidade In: Encontro Nacional de Inteligência Artificial - ENIA 2003/SBC Campinas. Anais do Congresso da SBC'2003. p.1 – 10, Campinas: SBC.
- Gaspary, Luciano P., Fagundes, Leonardo (2003) L. Avanços Rumo à Integração de Tecnologias de Gerenciamento de Redes e Segurança. In: I Escola Regional de Redes de Computadores, Anais. pp. 153. Porto Alegre. Minicurso.
- Leonhardt, Michelle D.; Castro, Daiane, D.; Tarouco, Liane M. R. (2003) Elektra: inteligência Artificial na Educação a Distância de Jovens e Adultos In: Congresso de Educação a Distância Mercosul., pp. 165-170. Florianópolis, Brasil.
- Lewis, Lundy. (1995) Managing Computer Networks: A Case-Based Reasoning Approach. Norwood: Artech House, 205p.
- Maghsoudi, Shahin; Arthanari, Tiru (2004): Learning Interface for Virtual Education. In: 8th World Multiconference on Systemics, Cybernetics and Informatics July 18 - 21, Orlando, Florida, USA. Disponível em: <http://www.alicebot.org/articles/learning-Interface.pdf>
- Medina, Roseclea D. (1994) SAFO Sistema Agregador de Ferramentas de Operação de Rede, Universidade Federal do Rio Grande do Sul, Brasil. Trabalho individual.
- Medina, Roseclea D; Tarouco, Liane. M. R. (2003) Tecnologias Aplicadas no Ensino de Redes de Computadores: um Protótipo de Laboratório Virtual para Facilitar a Aprendizagem Significativa. Anais CACIC'2003 - La Prata, Argentina.
- Melchiors, Cristina. (1999) Raciocínio Baseado em Casos Aplicado ao Gerenciamento de Falhas em Redes de Computadores. Porto Alegre: PGCC/UFRGS., Dissertação de Mestrado.
- Taylor, A. (2003) The AIML Mini Reference and Primer. Disponível em: <<http://hippie.alicebot.com/~ataylor/index.html>>. Último acesso em 08.06.2003.

Uma Arquitetura para Correlação de Alarmes Baseada em Políticas e Web Services

Evandro Della Vecchia Pereira, Lisandro Zambenedetti Granville

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

edvpereira@inf.ufrgs.br, granville@inf.ufrgs.br

Resumo. No gerenciamento de redes de computadores, alarmes são fundamentais para a percepção de erros ou falhas. Com a utilização de SNMP, agentes são configurados para enviarem traps aos gerentes, alertando sobre a constatação de alguma anormalidade em dispositivos ou na própria rede. Porém, vários agentes em uma mesma rede podem constatar a mesma falha e enviarem o mesmo alerta ao gerente. Ou ainda, se a mesma falha não é resolvida em determinado tempo, inúmeros alertas são gerados. Quando há apenas um gerente ou apenas um nível de gerentes, não há como eliminar alertas repetidos ou considerados inúteis para o administrador, mas quando há mais de um nível de gerentes, uma correlação de alarmes pode ser feita. Esta correlação faz com que o número de alarmes seja reduzido, o que ajuda o administrador na visualização das falhas em seu software de gerenciamento. Este trabalho propõe uma arquitetura onde é feita uma correlação dos alarmes e a comunicação entre gerentes é feita através de Web Services.

1. Introdução

Alarmes são muito úteis no gerenciamento de redes de computadores, pois a iniciativa é do dispositivo, avisando ao gerente que há algum problema. Dessa forma o gerente não precisa consultar todos os dispositivos da rede em um intervalo de tempo muito pequeno. Além disso, o alarme é disparado seguindo limiares definidos pelo administrador, como por exemplo, em determinado momento o administrador define que um determinado percentual de colisões de pacotes na rede é muito alto e deve ser gerado um alarme, e, em outro momento o administrador decide baixar esse percentual.

Os alarmes mais utilizados são os *traps*, transportados pelo SNMP. Cada alarme é transportado em uma mensagem, não havendo correlação entre alarmes. Esta possível correlação poderia reduzir o número de mensagens. Se um problema persiste por muito tempo, os alarmes são gerados continuamente, relatando o mesmo problema, o que pode "inundar" a rede com tráfego desnecessário. Outro problema é que determinadas falhas ocasionam outras, se não forem corrigidos os problemas. Como não há correlação levando-se em consideração determinado intervalo de tempo, todas falhas geram alarmes, o que "inunda" mais ainda a rede.

A partir do SNMPv2, a comunicação entre gerentes se tornou possível, porém os alarmes são repassados sem nenhuma correlação, e o tráfego continua sem ser reduzido.

Para tal correlação, é necessário que algum sistema de gerência analise os alarmes vindos dos dispositivos, verifique regras de correlação definidas pelo administrador em uma base de dados e repasse ao gerente do nível superior menos mensagens de alarme, reduzindo o tráfego e melhorando a visualização de alarmes pelo administrador, sem nenhum prejuízo à possível resolução das falhas acusadas pelos alarmes.

Muitas vezes, a comunicação entre gerentes é realizada em uma rede de longa distância. Como mensagens SNMP são transportadas sobre UDP, corre-se o risco da perda de alarmes. Uma solução para este problema é a utilização da tecnologia Web Service. Além de serem orientados à conexão, os Web Services são componentes independentes de software capazes de oferecer mecanismos de chamadas de procedimento remoto. Web Services utilizam como protocolo de transporte o SOAP, que pode ser encapsulado em HTTP, FTP, SMTP entre outros. O mais aconselhado, para não ter problemas em filtros de *firewalls*, é o HTTP.

Na seção 2 será mostrado o conceito de Web Services, na seção 3 o conceito de correlação de alarmes. A arquitetura proposta será apresentada na seção 4 e por fim, na seção 5 serão mostradas as conclusões.

2. Web Services

Web Services são componentes independentes de aplicações que são disponibilizados na Web de tal maneira que outras aplicações Web possam achá-los e utilizá-los. Variam de simples a complexos e trazem a promessa de flexibilidade e de computação distribuída na Internet [Roy and Ramanujan, 2001].

Os Web Services, considerados sucessores de CORBA e DCOM, são um dos principais passos na evolução da Web. Eles permitem que objetos ativos sejam colocados em Web sites para fornecer serviços distribuídos a clientes. Os Web Services têm sido utilizados em *e-commerce*, no gerenciamento de informações distribuídas, já que sistemas de bancos de dados distribuídos têm dificuldades com compatibilidade de plataformas e softwares [Abiteboul et al., 2000]. Também podem ser descritos como aplicações capazes de executar transações na Internet. Eles podem ser acessados tanto pelos clientes (ex.: browsers) como por outros Web Services [Sahai et al., 2001].

Os Web Services podem ser vistos como integradores de computadores, dispositivos, bancos de dados e redes em um único sistema virtual, onde os usuários podem trabalhar utilizando browsers [Vaughan-Nichols, 2002].

Uma das principais utilidades dos Web Services são as chamadas de procedimento remoto: usuários ativam determinadas tarefas em um dispositivo/máquina remota [Preece and Decker, 2002].

Uma definição mais completa para Web Service seria: um serviço disponível na Internet que utiliza um sistema de mensagens XML, e não depende de sistema operacional nem linguagem específica. Na figura 1 é mostrado como é feita a comunicação de um Web Service básico em alto nível [Cerami, 2002].

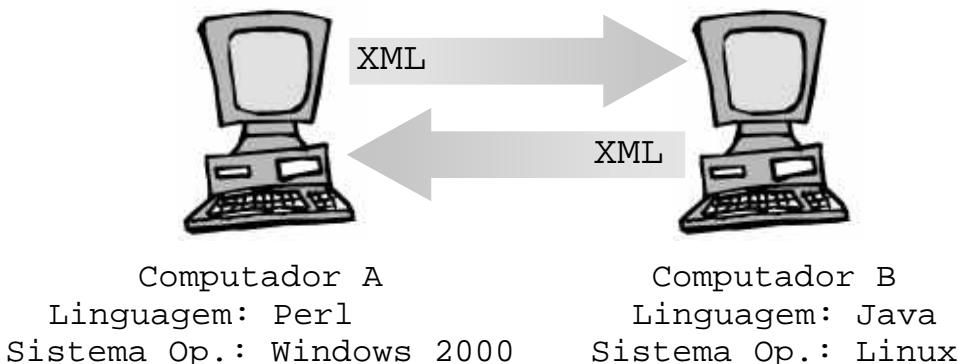


Figura 1: Web Service básico

3. Correlação de Alarms

Alarms gerados por dispositivos de redes de computadores e enviados ao(s) gerente(s) são essenciais para ajudar o administrador à verificar e, se possível, solucionar problemas. Porém, muitas vezes, alarms relatando o mesmo problema são gerados porque o intervalo de tempo definido para geração de alarms é pequeno. Isso faz com que a visualização do administrador que utiliza um software gerente seja prejudicada. Outra situação que ocasiona uma quantidade maior de alarms é que algumas falhas fazem com que vários dispositivos gerem o mesmo alarme, sendo que na verdade há apenas uma falha.

Um alarme é a notificação de um evento considerado anormal, que pode ou não representar um erro (ou falha). Como mencionado anteriormente, um único incidente pode causar a geração de vários alarms. Abaixo são listados alguns fatores relacionados a estes fatos [de Castro and Nogueira, 1998]:

- um agente pode gerar vários alarms em consequência de uma única falha;
- uma falha pode ser intermitente, o que resulta em um novo alarme para cada ocorrência da falha;
- uma falha em um item pode ocasionar a geração de alarms cada vez que o dispositivo for solicitado (uma consulta por exemplo);
- uma única falha pode ser detectada por diferentes itens na rede;
- uma falha em um item pode afetar outros itens na rede.

Como pode ser visto, uma redução das mensagens de alarme pode ser necessária, para agilizar o processo de análise e solução de problemas, pelo administrador. Essa redução de mensagens pode ser realizada através da correlação de alarms. A correlação deve levar em conta os fatores mencionados anteriormente. Ou seja, além de serem correlacionados os alarms originados por agentes diferentes relatando a mesma falha, alarms devem ser correlacionados também em relação ao tempo. Mensagens contendo o mesmo alarme em instantes de tempo diferentes devem ser analisadas para que seja feita a verificação se eles estão relatando a mesma falha (quando a falha não foi solucionada). Essas mensagens também podem significar uma falha atuando de forma intermitente ou ainda, a falha pode ter sido solucionada e o item voltou a apresentar problemas.

Algumas técnicas de correlação de alarms são comumente utilizadas em *softwares* comerciais. São elas [Zupan and Medhi, 2003]:

- raciocínio baseado em regras;

- raciocínio baseado em modelos;
- raciocínio baseado em casos;
- codebook;
- modelo de grafo de transição de estados;
- modelo de máquina de estados finita.

Além de utilizar técnicas de correlação de alarmes, é necessário conhecer o comportamento de eventos presentes em uma rede. Segundo [Melo et al., 2000], os seguintes tipos de comportamento de eventos estão presentes em redes IP gerenciadas através de SNMP:

- *start/stop*: eventos que se apresentam aos pares. Um dos eventos indica quando certa condição se torna válida e outro indica quando esta não é mais válida. Como exemplo tem-se *traps* RMON [Waldbusser et al., 2003], *traps* SNMP de mudança de estados e eventos de monitoração do próprio gerente SNMP;
- *storm*: seqüência de vários eventos do mesmo tipo, em um determinado período;
- fluxo de eventos correlatos: quando a ocorrência de um evento está associada com a ocorrência de outro;
- fluxo de eventos independente: não tem relação significativa com outros eventos.

4. Arquitetura Proposta

Nesta seção será mostrada uma arquitetura que tem como objetivo correlacionar alarmes de forma hierárquica (figura 2). Os alarmes são gerados nos dispositivos de rede através de *traps* SNMP ou RMON, estes são correlacionados em um gerente de primeiro nível e repassados ao gerente de nível superior. Ou seja, a arquitetura prevê pelo menos dois níveis de gerente e teoricamente não há limite nos níveis gerenciais.

O envio das mensagens de alarmes já correlacionados é feito com o uso de Web Services. O processo de correlação de alarmes e envio ao nível superior se repete até que os alarmes cheguem ao gerente do último nível. Cada gerente deverá ter uma base de dados contendo as regras necessárias para que a correlação seja feita. Estas regras devem permitir tanto a correlação de alarmes originados em dispositivos diferentes como a correlação de alarmes originados no mesmo dispositivo, porém em instantes diferentes. Isso porque em determinadas falhas, o mesmo alarme pode ser originado de tempo em tempo, conforme mencionado anteriormente.

A correlação será feita com o auxílio de políticas (regras) armazenadas em bases de dados que os gerentes tenham acesso. O administrador alimentará as bases de dados com as regras de correlação, de acordo com as técnicas de correlação e tipos de comportamento de eventos apresentados na seção 3. O uso de gerenciamento baseado em políticas [Westerinen et al., 2001] permite que a árvore de correlação de alarmes seja dinâmica. Um exemplo do dinamismo da árvore de correlação seria o caso de um gerente, seguindo as políticas definidas pelo administrador, enviar suas correlações a um gerente X no período das 7h às 8h, e ao gerente Y, no período das 8h às 9h. Dessa forma o gerente de nível superior muda conforme o horário em questão.

Com a utilização de Web Services na comunicação entre gerentes, há a possibilidade destes se encontrarem em domínios administrativos diferentes, visto que os Web

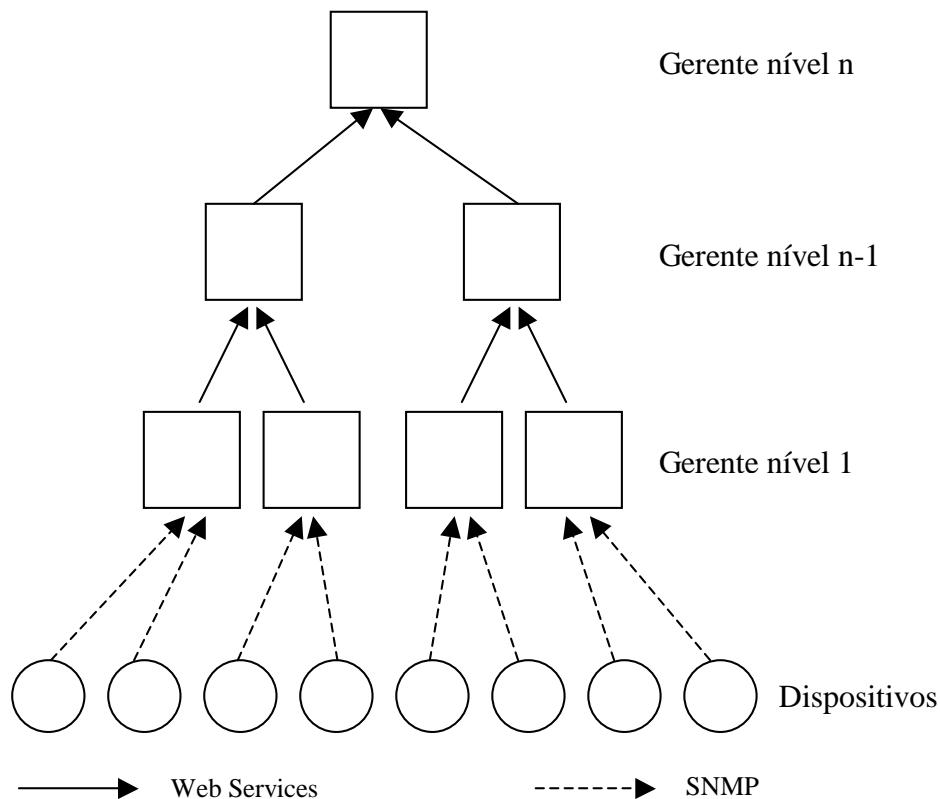


Figura 2: Arquitetura Proposta

Services podem ser encapsulados em HTTP, SMTP, entre outros. Protocolos como o HTTP e SMTP geralmente tem acesso livre em *firewalls*, evitando ter que solicitar ao administrador de determinado domínio administrativo uma liberação de tráfego em certas portas.

Outra vantagem na utilização dos Web Services na comunicação entre gerentes, utilizando HTTP ou SMTP é que estes são orientados à conexão. Desta forma há garantia na entrega das mensagens. Para evitar problemas como servidor indisponível por alguns instantes, por exemplo, optou-se pelo uso de Web Services baseados em SMTP. Assim, se o servidor estiver indisponível por alguns instantes, não haverá problema no envio das mensagens, ao contrário do HTTP que não permite uma comunicação *off-line*. A utilização de Web Services sobre SMTP é totalmente inexplorado, o que requer pesquisa para que seja comparado com soluções que utilizam Web Services sobre HTTP ou outro protocolo.

5. Conclusões

A partir dos estudos realizados, foi constatado que correlação de alarmes é um assunto não muito explorado e que a utilização da tecnologia Web Service para realizar a correlação aparentemente não é utilizada. Um aspecto importante na utilização de Web Services é a facilidade de implementação e manutenção, sem contar a fácil interoperabilidade que esta tecnologia propicia. Outro aspecto importante é a garantia de entrega das mensagens de alarme, tornando o sistema confiável (no caso da utilização de SMTP para entrega das

mensagens, deve-se levar em consideração o *timeout* definido para servidores SMTP e se uma rede ou servidor ficar inoperante por mais tempo que o *timeout*, a garantia de entrega deixa de existir).

A alimentação das bases de dados pelo administrador, com regras de correlação, torna o sistema flexível. Com isso, novos agentes podem ser implementados, com novos *traps* e mesmo assim o gerente poderá realizar as correlações.

Um estudo sobre ferramentas comerciais deve ser feita para que se possa comparar resultados. A idéia deste trabalho é implementar a arquitetura proposta e verificar se esta solução é melhor ou não do que soluções que não utilizam Web Services. Para isto, testes de desempenho deverão ser realizados após a implementação completa da arquitetura.

Referências

- Abiteboul, S., Benjelloun, O., and Milo, T. (2000). Web services and data integration. In *Proceedings INTERNATIONAL CONFERENCE ON WEB INFORMATION SYSTEMS ENGINEERING (WISE)*.
- Cerami, E. (2002). *Web Services Essentials*. O'Reilly.
- de Castro, T. and Nogueira, J. (1998). An alarm correlation system for sdh networks. In *Proceedings Telecommunications Symposium - SBT/IEEE International ITS '98*. Pages 492 - 497, Vol. 2.
- Melo, E., Vieira, E., and Sari, S. (2000). Tratamento de eventos. *Boletim bimestral sobre tecnologia de redes*, 4(4). Disponível em <<http://www.rnp.br/news/gen/0007/art10.html>>. Acesso em 08 Jun.
- Preece, A. and Decker, S. (2002). Intelligent web services. *IEEE Intelligent Systems*, pages 15–17.
- Roy, J. and Ramanujan, A. (2001). Understanding web services. *IEEE Computer Society - ITPro*, pages 69–73.
- Sahai, A., Machiraju, V., Ouyang, J., and Wurster, K. (2001). Message tracking in soap-based web services. *Technical Reports Hewlett-Packard Laboratories*.
- Vaughan-Nichols, S. J. (2002). Web services: Beyond the hype. *Industry Trends*, pages 18–21.
- Waldbusser, S., Kalbfleisch, C., and Romascanu, D. (2003). *Introduction to the Remote Monitoring (RMON) Family of MIB Modules: RFC 3577*. IETF.
- Westerinen, A. et al. (2001). *Terminology for Policy-Based Management: RFC 3198*. IETF.
- Zupan, J. and Medhi, D. (2003). An alarm management approach in the management of multi-layered networks. In *Proceedings 3rd IEEE Workshop on IP Operations and Management (IPOM)*. Pages 77 - 84.

Um Sistema Integrado para a Administração de Computadores e Serviços de Redes Locais*

Diego Luís Kreutz¹, Benhur Stein¹

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

{kreutz,benhur}@inf.ufsm.br

Resumo. Este trabalho apresenta um sistema para o gerenciamento integrado e simplificado de serviços e máquinas de uma rede local, que está sendo desenvolvido e utilizado no Núcleo de Ciência da Computação (NCC) da UFSM. No decorrer do texto serão apresentadas a versão inicial do sistema, em funcionamento há aproximadamente um ano na rede do NCC, e a nova versão, que está em fase de projeto e desenvolvimento.

1. Introdução

O gerenciamento integrado de sistemas e serviços de rede é um objeto de pesquisa e desenvolvimento atual [Yoon et al., 2003, White et al., 2002, Schwartz et al., 2000, van Hemmen, 2000, Drake, 1997]. No entanto, poucos sistemas simples e práticos de gerenciamento integrado de máquinas e serviços de rede existem, e em geral, não são muito difundidos e utilizados.

Atualmente existem várias ferramentas e protocolos de gerenciamento de serviços e dispositivos. As ferramentas são normalmente para configuração e manutenção de uma única máquina ou serviço. Alguns protocolos são proprietários e restritivos, outros são mais recentes e estão em fase de projeto e desenvolvimento. Existem ainda os protocolos mais antigos e relativamente bastante utilizados. No entanto, estes comumente centram suas atenções apenas em aspectos como gerência de dispositivos.

O objetivo deste trabalho é o desenvolvimento de um sistema integrado de gerenciamento de computadores e serviços de uma rede local. A proposta é construir um sistema capaz de gerar arquivos de configuração para serviços e computadores a partir de uma única base de dados. Além disso, automatizar a propagação das atualizações de informação. Em uma primeira instância, resolver questões mais imediatas de administração de serviços de rede.

A primeira fase foi atingida pelo desenvolvimento, implantação e teste de um sistema centralizado de gerenciamento de alguns serviços da rede do NCC [Kreutz et al., 2003a]. Em uma segunda etapa, a proposta é dar continuidade ao desenvolvimento e incrementação de funcionalidades do sistema.

Este texto apresenta a proposta geral de uma ferramenta integrada de gerenciamento de serviços e computadores de rede, a versão inicial do sistema e o estado atual, alguns breves comentários sobre o estado da arte em sistemas de gerenciamento de redes locais, as perspectivas de continuidade e a conclusão.

*Agência de fomento CNPq: processo 380049/03-1

2. Trabalhos relacionados

Alguns exemplos de ferramentas e protocolos de gerenciamento e configuração de sistemas e redes são o Linuxconf [Gélinas, 2003], SMIT¹ [Segura, 2000], SNMP [Case et al., 1990], NetInfo [Apple Computer, Inc., 1998] e NetConf [Enns, 2004, Wasserman, 2003].

O Linuxconf e o SMIT são ferramentas que basicamente visam a configuração local de uma única máquina. Ambos são práticos e fáceis de manipular.

Protocolos como o SNMP e NetConf são soluções mais amplas, visando a configuração dos diversos dispositivos de uma rede. No entanto, não objetivam oferecer soluções para o gerenciamento de serviços de rede.

O NetInfo é uma solução mais completa para o gerenciamento de máquinas e serviços de uma rede. No entanto, um inconveniente é a necessidade de criar uma interface de comunicação (entre o NetInfo e aplicativos) para praticamente todos os programas e sistemas.

Uma outra forma de realizar a configuração de dispositivos de rede é através de interface de linha de comando (CLI²). Essa é a maneira mais difundida e utilizada pelos administradores de redes [Lee and Choi, 2002].

Protocolos e ferramentas como os apresentados nos parágrafos anteriores são limitados e/ou normalmente destinados apenas a configuração e monitoramento de dispositivos de rede. É comum não englobarem a geração de arquivos de configuração, atualização e manutenção de serviços de rede. Além disso, protocolos como o SNMP possuem limitações quanto a escalabilidade, principalmente devido a sua forte característica de simplicidade. E soluções como o NetInfo são proprietárias e pouco flexíveis.

A proposta central deste trabalho não é desenvolver um concorrente a protocolos recentes como o NetConf. O objetivo principal é desenvolver um sistema simples e eficiente para o gerenciamento automatizado e integrado das tarefas mais comuns dos administradores de redes. Entre essas tarefas podem ser citadas a configuração de serviços como o DNS, DHCP, NFS e IPTables. Estes serviços são normalmente gerenciados, de forma individual, manualmente ou por ferramentas específicas.

3. Arquitetura e Funcionamento do Sistema Proposto

O objetivo é criar um ambiente simples e prático para o gerenciamento integrado de computadores e serviços de rede. Com isso em mente foi projetada a arquitetura do sistema.

A figura 1 apresenta a arquitetura do sistema. A base de dados fornece as informações de configuração de máquinas e serviços. Esses dados são utilizados pelos módulos geradores para criar os arquivos de configuração de máquinas e serviços. Os módulos de atualização são responsáveis por propagar os arquivos de configuração aos respectivos serviços e/ou computadores. O processo todo é controlado pelos agentes inteligentes.

Pode ser observado a existência de duas bases de informação. Na verdade, existe apenas uma base de informação dividida em duas sub-árvores de dados. A primeira contém as descrições de computadores e serviços. A segunda armazena os arquivos de configuração de máquinas e serviços, criados pelos módulos de geração.

Na base de dados a descrição dos computadores contém as principais informações necessárias para o funcionamento de cada máquina e de serviços como o DNS³ e DHCP⁴.

¹System Management Interface Tool

²Command Line Interfaces

³Domain Name Service

⁴Dynamic Host Configuration Protocol

Além disso, os serviços prestados por máquinas servidoras também são indicados e descritos nessa base de dados. As descrições de formato e configurações estáticas de serviços constituem um outro tipo de informação armazenada nessa base de dados. Esse conjunto de dados permite a geração completa dos arquivos de configuração dos serviços de rede gerenciados pelo sistema.

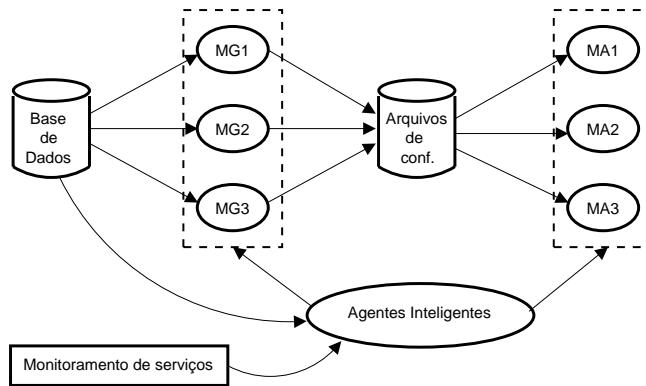


Figura 1: Ilustração da arquitetura do sistema (MG - Módulo de Geração; MA - Módulo de Atualização)

Os módulos de geração são componentes especializados que utilizam as descrições de formato, as informações estáticas e demais dados necessários para manter atualizados os respectivos serviços ou computadores. A geração das tabelas de DNS pode ser um bom exemplo. Para gerar estas tabelas são necessárias as informações estáticas (como partes do cabeçalho), as descrições de formato dos arquivos de configuração do DNS e os nomes e IPs das máquinas da rede.

A manutenção de serviços e máquinas é feita pelos módulos de atualização. Estes são divididos em dois grupos especialistas: atualização de serviços e atualização de computadores. Os componentes de manutenção de serviços são responsáveis por matar os processos, copiar os novos arquivos de configuração e iniciá-los novamente. As máquinas são configuradas ou atualizadas pelo segundo grupo de componentes. Sua função é copiar os novos arquivos de configuração para a máquina e re-iniciar serviços locais ou o computador, buscando efetivar as atualizações.

O sistema como um todo é comandado pelos agentes inteligentes. São eles quem detectam atualizações de dados, ativando os módulos de geração e atualização. Os módulos de geração e atualização são componentes especialistas, independentes e não tem noção da existência das demais partes do sistema, o que facilita a extensão, manutenção do sistema ou inclusão de novos módulos. Além disso, os módulos de atualização podem ser locais ou remotos. O acréscimo, para manter os módulos remotos em relação aos módulos locais, é basicamente comunicações a distância entre eles e os agentes inteligentes. Logo, a presença dos agentes inteligentes, de certo modo, simplifica tarefas básicas de manutenção e atualização do sistema propriamente dito.

Os elementos de monitoramento de serviços tem como função verificar a disponibilidade de cada serviço e alertar sobre uma possível necessidade de atualização. Eles avisam os agentes quando algum serviço está com sua disponibilidade debilitada ou não está acessível. Os agentes, a partir de regras de manutenção de serviços, verificam os procedimentos a serem seguidos. Estes podem incluir ações como a atualização do serviço ou a transferência dele para uma nova máquina. No último caso, na base de dados deve estar definida qual será a máquina que substituirá a prestação de um determinado serviço no caso de falhas. O serviço será ativado na nova máquina e os demais serviços e configurações relacionados a

essa nova alteração serão atualizados. A troca do servidor SMTP⁵ poderia ser um exemplo. Neste caso, as tabelas do servidor DNS e as regras do IPTables, do *firewall* da rede, provavelmente deveriam ser atualizadas. Os agentes inteligentes são responsáveis por verificar essas dependências na base de dados e tomar as devidas providências para assegurar o correto funcionamento da rede como um todo.

4. Desenvolvimento do Sistema

O sistema está em fase de desenvolvimento. Para testes e validação foi desenvolvida uma versão inicial que gerencia de forma integrada e automática diferentes serviços de uma rede local. Em uma segunda etapa, essa versão inicial está sendo expandida e melhorada para atingir os objetivos do trabalho descritos na seção anterior.

4.1. Um sistema para o gerenciamento de serviços

A primeira versão do sistema surgiu de necessidades básicas de gerenciamento da rede do NCC. Entre as tarefas administrativas estavam o gerenciamento dos servidores DNS, DHCP, NFS⁶ e do filtro de pacotes IPTables. Essas tarefas eram feitas de forma individual e independente. A inclusão ou alteração de uma máquina demandava em grande parte dos casos a alteração dos quatro serviços. Por isso foi criada uma versão inicial que gerencia de forma automática e centralizada esses serviços da rede.

Na concepção da primeira versão do sistema integrado de gerenciamento de máquinas e serviços de rede [Kreutz et al., 2003a] optou-se pela simplicidade, praticidade e funcionalidade. Por isso, foi desenvolvido um sistema modular baseado em uma entrada de dados simples. Um arquivo de texto puro, contendo as informações acerca dos computadores e roteadores da rede. Informações essas necessárias para manter os serviços de DNS, DHCP, NFS e IPTables.

A figura 2 apresenta uma visão geral do sistema. Ele é basicamente constituído por três camadas. A primeira representa os dados de entrada, no caso um arquivo de texto puro, a segunda camada é constituída de diversos módulos de geração de arquivos de configuração para os diferentes serviços. Todos esses módulos utilizam a mesma entrada de dados. A terceira camada é a parte de atualização dos respectivos servidores e serviços.

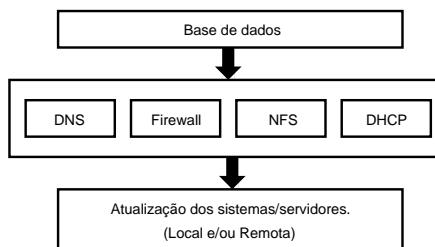


Figura 2: Ilustração da arquitetura da versão inicial do sistema

Após alterar o arquivo de texto contendo a base de dados o administrador pode solicitar a atualização de um ou de todos os serviços. A atualização e reinicialização de um serviço pode ser feita tanto na máquina local (que contém o sistema) como em uma máquina remota, tornando possível o gerenciamento centralizado de uma rede com diversos servidores.

Com esse sistema torna-se prático e fácil a manutenção de diferentes serviços, mesmo quando esses serviços estão localizados em diferentes servidores. Além disso, a

⁵Simple Mail Transfer Protocol

⁶Network File System

modularidade do sistema permite a fácil incorporação e automatização do gerenciamento de novos serviços, bastando apenas incluir um novo módulo, independente dos demais.

4.2. Continuando o desenvolvimento do sistema

Um dos fatores que levou a evolução do sistema foi a limitação da versão inicial. Um exemplo é a base de dados, formada por um único arquivo de texto. Ela pode tornar-se ineficiente para manipular grandes volumes de informação, adaptar novos campos nos registros e formar hierarquias de relação e dependência. Outro aspecto é a dificuldade em controlar e manter a consistência dos registros em um arquivo de texto puro.

Um passo inicial foi a troca da base de dados. O protocolo LDAP⁷ [Laird and Soraiz, 1999, Hodges and Morgan, 2002, Kreutz et al., 2003b] mostrou-se como uma boa opção. Escalabilidade, legibilidade, eficiência, segurança, replicação nativa, consistência dos dados e flexibilidade estão entre suas características.

Entre as próximas etapas estão o desenvolvimento dos agentes inteligentes e dos componentes de tolerância a falhas. Os agentes inteligentes compõem a parte responsável por tomadas de decisão a partir dos dados dos componentes de monitoramento, manutenção, atualização de serviços e máquinas e gerenciamento do sistema de uma forma mais geral. Os componentes de tolerância a falha (agentes de monitoramento) são responsáveis pela disponibilidade dos serviços da rede. Com isso cria-se um sistema integrado, automatizado e ativo para o gerenciamento de serviços e máquinas de uma rede local.

5. Conclusão e trabalhos futuros

A administração de redes locais ainda é, geralmente, uma tarefa que envolve a manipulação e configuração de um grande número de serviços de forma independente. Um dos fatores que contribui para isso é a falta de sistemas integrados de gerenciamento de serviços e máquinas de rede.

O objetivo principal deste trabalho é a construção de um sistema prático e adaptável de gerenciamento integrado de serviços e dispositivos de rede. Com isso em mente, foi criado e implementado um protótipo inicial do sistema que é capaz de centralizar e facilitar a administração de diferentes serviços de rede, sendo que estes serviços podem estar em uma única máquina ou em máquinas distintas.

Dando continuidade ao desenvolvimento da proposta desse trabalho (seção 3) está sendo desenvolvida uma versão mais flexível e escalável da ferramenta. Nessa fase começam a ser utilizadas tecnologias como o LDAP, que permitem a construção de bases de dados íntegras, hierárquicas e maleáveis, o que torna possível incluir características que permitirão um desenvolvimento mais completo do sistema.

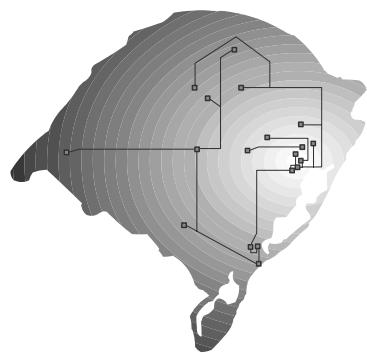
O sistema proposto, e sua versão funcional, apresentam facilidades que agilizam e centralizam o gerenciamento de vários serviços de uma rede local. Basta alterar a base de dados e solicitar a atualização dos diferentes serviços que são afetados pelas modificações realizadas, ou deixar que os agentes inteligentes procedam automaticamente as atualizações.

Trabalhos futuros. O objetivo é dar continuidade a elaboração e desenvolvimento do sistema. Dentro desse contexto, um passo inicial é evoluir as interfaces de gerenciamento para englobar as diferentes camadas e hierarquias do diretório LDAP. Uma segunda etapa é acrescentar a parte funcional de atualização de máquinas clientes da rede. Na primeira versão existem apenas os módulos para gerenciamento e atualização de serviços. Por fim, desenvolver e integrar os agentes inteligentes e os componentes de monitoramento.

⁷Lightweight Directory Access Protocol

Referências

- Apple Computer, Inc. (1998). Mac OS X Server: Basic Introduction To NetInfo Domains. <http://docs.info.apple.com/article.html?artnum=30832>.
- Case, J., Fedor, M., Schoffstall, M., and Davin, J. (1990). RFC 1157 - simple network management protocol (SNMP). <http://www.faqs.org/rfcs/rfc1157.html>.
- Drake, D. W. (1997). Using linux to solve the problem of mixed operating system lan services. In *Proceedings of the 25th annual ACM SIGUCCS conference on User services*, pages 67–72. ACM Press.
- Enns, R. (2004). NETCONF Configuration Protocol (draft-ietf-netconf-prot-02). <http://www.ietf.org/internet-drafts/draft-ietf-netconf-prot-02.txt>.
- Gélinas, J. (2003). Linuxconf - Linux Administration Made Easy! <http://www.solucorp.qc.ca/linuxconf/>.
- Hodges, J. and Morgan, R. (2002). Lightweight Directory Access Protocol (v3). Technical report, IETF - Network Working Group. <http://www.ietf.org/rfc/rfc3377.txt>.
- Kreutz, D. L., Fioreze, T., and Stein, B. (2003a). Integrando e Centralizando a Administração de Serviços de uma LAN. In *II Simpósio de Informática da Região Centro do RS*. <http://www.sirc.unifra.br/artigos2003/Artigo14.pdf>.
- Kreutz, D. L., Mazzutti, C., and Stein, B. (2003b). Utilizando a Tecnologia LDAP para Centralizar, Controlar e Representar Dados, Garantindo Integridade e Consistência. In *XVIII Congresso de Iniciação Científica e Tecnológica em Engenharia*.
- Laird, C. and Soraiz, K. (1999). LDAP Comes Age. *WebServer OnLine Magazine*, 6(12).
- Lee, B. and Choi, T. (2002). Cli-based mediation mechanism using xml for configuration management (draft-lee-xmlconf-xcli-00.txt). <http://www.ietf.org/internet-drafts/draft-lee-xmlconf-xcli-00.txt>.
- Schwartz, B., Jackson, A. W., Strayer, W. T., Zhou, W., Rockwell, R. D., and Partridge, C. (2000). Smart packets: applying active networks to network management. *ACM Trans. Comput. Syst.*, 18(1):67–88.
- Segura, S. (2000). System Management Interface Tool (SMIT). Technical report. <http://www-1.ibm.com/servers/aix/products/aixos/whitepapers/smit.pdf>.
- van Hemmen, L. J. G. T. (2000). Models supporting the network management organization. *Int. J. Netw. Manag.*, 10(6):299–314.
- Wasserman, M. (2003). Using the NETCONF Configuration Protocol over Secure Shell (ssh) (draft-ietf-netconf-ssh-00.txt). <http://www.ietf.org/internet-drafts/draft-ietf-netconf-ssh-00.txt>.
- White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., and Joglekar, A. (2002). An integrated experimental environment for distributed systems and networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):255–270.
- Yoon, J.-H., Ju, H.-T., and Hong, J. W. (2003). Development of SNMP-XML translator and gateway for xml-based integrated network management. *Int. J. Netw. Manag.*, 13(4):259–276.



Sessão Técnica 4

Tolerância a Falhas II

Máquinas Virtuais como Suporte ao Desenvolvimento de Aplicações Tolerantes a Falhas

Tórgan F. de Siqueira¹, Taisy S. Weber¹, Reynaldo Novaes², Ingrid Jansch-Porto^{1*}

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul – Porto Alegre, RS

²Hewlett-Packard do Brasil – Porto Alegre, RS

{torgan,taisy,ingrid}@inf.ufrgs.br, reynaldo.novaes@hp.com

Resumo. *Máquinas virtuais têm sido empregadas no compartilhamento de recursos de hardware em servidores e estações de desenvolvimento. Suas características incluem alto desempenho, suporte a múltiplos sistemas operacionais virtualizados, estabilidade, segurança e possibilidade de injeção de falhas. Visto para acelerar o ciclo de desenvolvimento de aplicações tolerantes a falhas, analisa-se as características das máquinas, UML, FAUmachine e XEN, e sua adequação quanto a desempenho, transparência, portabilidade, intrusividade, instrumentabilidade, monitorabilidade e transportabilidade.*

1. Introdução

O conceito e as primeiras implementações de máquinas virtuais (MV) são conhecidos desde meados da década de 60, primeiro como cenário de desenvolvimento, e posteriormente levados à indústria através dos conhecidos IBM System 370 (S/370) e IBM System 390 (S/390) [Kohlbrenner et al., 2004]. Atualmente, são utilizadas como servidores e estações de desenvolvimento, tendo impulso tanto na indústria [VMware Inc., 2004] quanto na academia ([Dike, 2004], [FAUmachine, 2004], [XEN, 2004], [Plex86, 2004], [Liang et al., 2003] e [Thain and Livny, 2003], por exemplo). Como servidores, as MVs destinam-se a compartilhar recursos de *hardware* entre diversos sistemas operacionais (SO), ou cópias destes, e cada instância executa um serviço específico. Entretanto, muitas máquinas podem e são usadas como ferramentas de suporte ao desenvolvimento e depuração, tanto de SOs quanto de aplicações.

O objetivo deste artigo é identificar as características de algumas máquinas virtuais e sua adequação ao desenvolvimento de aplicações tolerantes a falhas para Grids. No âmbito do projeto DependableGrid (UFRGS/ HP Brasil P&D), uma das metas é salvar e restaurar o estado de aplicações, seja no próprio computador após *crash* seguido de recuperação, ou em outro computador do Grid. Neste contexto, as propriedades avaliadas foram escolhidas considerando todo o ciclo de desenvolvimento das aplicações e de seus mecanismos de tolerância a falhas e validação, o que abrange níveis distintos, indo do kernel do SO a aplicações Java distribuídas.

Na seção 2, são descritas as propriedades mencionadas, em função do ambiente de desenvolvimento. Na seção 3, são apresentadas as características gerais e de três máquinas virtuais: UML, FAUmachine e XEN. A seção 4 apresenta uma comparação entre as MVs de acordo com as propriedades, e as conclusões são apresentadas na seção 5.

2. Ambientes de Desenvolvimento de Aplicações Tolerantes a Falhas

O objetivo do uso de MVs para desenvolver de aplicações tolerantes a falhas é acelerar o ciclo de desenvolvimento. Isto se alcança, por exemplo, permitindo o estudo de vários

*Este trabalho foi desenvolvido conjuntamente com a HP Brasil P&D

cenários sem precisar reconfigurá-los a cada vez; utilizando uma infraestrutura de *hardware* comum e variando o SO; reaproveitando instalações de distribuições; confinando testes de rede e assim eliminando a interferência entre sistemas não-relacionados.

Dentre as características que um ambiente de desenvolvimento de aplicações tolerantes a falhas apresenta, algumas recebem mais atenção quando da avaliação de MVs: desempenho, transparência, portabilidade, intrusividade, instrumentabilidade, monitorabilidade e transportabilidade. Estas características foram sendo anotadas ao longo de experiências de desenvolvimento, acrescidas recentemente pela necessidade de distribuição de aplicações na plataforma *Grid*.

O **desempenho** é necessário para não penalizar a máquina hospedeira. As medidas principais são o tempo de carga da MV e o de execução das aplicações dentro desta, além do impacto sobre as demais instâncias em execução.

A **transparência** diz respeito à percepção da MV pela aplicação em desenvolvimento. A inserção da MV não deve alterar o desenvolvimento e/ou características da aplicação. Neste caso, por exemplo, injeta-se falhas na MV e estuda-se o comportamento/resposta da aplicação, evitando a injeção de falhas no sistema hospedeiro. Injetar falhas na MV também permite que execuções concorrentes não só continuem, mas também sejam avaliadas em caso de falha daquela que foi abortada, particularmente útil em sistemas distribuídos sob teste em ambiente confinado.

A **portabilidade** de uma MV é deseável, pois permite o desenvolvimento de aplicações portáveis desde os estágios iniciais, quando houver esta necessidade. A portabilidade diz respeito à plataforma hospedeira em termos de *hardware/SO* nativo.

A **intrusividade** refere-se ao custo de inserir a MV no desenvolvimento da aplicação. Idealmente, a MV não deve ser intrusiva, isto é, a aplicação deve ter seu desenvolvimento independente do nível inferior ser um sistema nativo ou uma MV.

A **instrumentabilidade** e **monitorabilidade** da MV referem-se ao grau em que se pode injetar falhas e monitorar o estado do sistema. Estão entre as mais importantes, pois determinam em que extensão uma MV é utilizável em um ciclo de desenvolvimento.

Por fim, a **transportabilidade** é um item a ser considerado em caso de desenvolvimento de sistemas distribuídos, *Grids* e *middleware*. Espera-se que, se a MV não for transportável entre dois hospedeiros, que pelo menos no nível de aplicações (ou processos) isto seja possível.

3. Máquinas Virtuais

Máquinas virtuais podem ser classificadas em dois grupos: as que virtualizam o conjunto completo de instruções da arquitetura, denominadas ISA-VM (*Instruction Set Architecture Virtual Machine*), e as máquinas que suportam uma interface binária para aplicações, denominadas ABI-VM (*Application Binary Interface Virtual Machine*), que virtualizam chamadas de sistema [Figueiredo et al., 2003].

Máquinas que usam o mesmo ISA têm melhor desempenho do que as que têm diferentes ISA, pois não precisam traduzir código em tempo de execução ou modificar o código binário da aplicação. As máquinas virtuais clássicas são ISA-VMs que suportam a execução de sistemas operacionais completos, como IBM S/390, VMware e Plex86. Estas máquinas suportam a execução concorrente de vários sistemas SOs sobre o mesmo *hardware*. Assim, três máquinas foram escolhidas: a XEN, que embora faça paravirtualização, se aproxima de uma ISA, e a FAUmachine e UML, do segundo grupo. Máquinas comerciais foram descartadas, pois procura-se uma solução em *software* livre.

3.1. UML - User-Mode Linux

O UML é uma MV que executa sobre o SO hospedeiro permitindo que um usuário não privilegiado a instancie e que se torne privilegiado dentro desta. A virtualização ocorre através da interceptação das chamadas de sistema, e a máquina não tem acesso ao *hardware* nativo, exceto se permitido. Do ponto de vista do usuário, o UML é um sistema Linux completo, e do ponto de vista do hospedeiro, é uma aplicação de usuário. Correntemente, o UML executa apenas sobre Linux, mas há planos de portá-lo para outras plataformas [Buchacker and Sieh, 2001], [Höxer et al., 2002], e [Dike, 2004].

O UML executa os binários do hospedeiro sem necessidade de recompilação ou religação. Tudo que a máquina cria fica confinado a ela, não se propagando para outras instâncias ou hospedeiro. Seus parâmetros de memória e disco podem ser modificados dinamicamente, e possui escalonador e sistema de memória virtual próprios, usando o sistema hospedeiro somente para acessos ao *hardware*. O escalonador nativo somente implementa a decisão feita no correspondente do UML. Os dispositivos UML são mapeados para os correspondentes reais no sistema hospedeiro. Com UML, é possível virtualizar um rede inteira dentro de um hospedeiro, inclusive isolando-a da rede física, se desejado. A gerência da máquina pode ser feita localmente, remotamente ou via *daemons*.

Cada instância UML utiliza um espaço de endereçamento próprio, na forma de arquivos temporários no hospedeiro. Estes arquivos podem ser mapeados para memória via *tmpfs*. A utilização de memória apresenta alguns problemas, já que cada instância UML se comporta exatamente como um *kernel*, isto é, não libera memória a não ser que seja solicitado a fazê-lo, mantendo dados em *cache*. Já existe uma alternativa que usa *mmap* e um dispositivo virtual, */dev/anon*, reduzindo a utilização de memória física do hospedeiro, mas ainda não é totalmente estável.

Uma máquina UML é vista como um arquivo no sistema nativo, e corresponde a uma imagem de um disco, contendo partições e arquivos da MV. Se um hospedeiro precisa instanciar um número grande de máquinas, este esquema é oneroso em termos de armazenamento, já que as imagens de discos são da ordem de gigabytes (3 a 4, para um sistema razoavelmente completo). A solução é utilizar *copy-on-write* (COW), com um sistema-base comum a todas as instâncias e mantendo as escritas individuais de cada uma em arquivos separados. Se necessário, pode-se incorporar estas cópias separadas ao sistema-base posteriormente. Como desvantagens, a instanciação de várias máquinas UML pode provocar instabilidade no sistema nativo, e a queda do hospedeiro provoca a queda de todas as instâncias. Processos dentro da UML executam com desempenho levemente inferior se comparado à execução no hospedeiro. Devido à construção da UML, uma instância pode se apropriar dos ciclos de processador, degradando o desempenho das demais.

O UML foi construído originalmente para trabalhar com *tracing threads* (TT), sendo cada processo da MV mapeado para um processo no hospedeiro. A TT rastreia chamadas de sistema nos processos da UML, intercepta-as e desvia o fluxo de execução para o *kernel* UML. O problema com este esquema é a possibilidade dos processos poderem acessar o hospedeiro, uma vez que estão no seu espaço de endereçamento. Além disso, este mecanismo é lento, pois há quatro chaveamentos de contexto, uma entrega e um retorno de sinal. Para resolver este problema, um novo mecanismo foi implementado, o *Separate Kernel Address Space* (SKAS). Este mecanismo requer modificação no *ptrace* do *kernel* hospedeiro, mas é pouco intrusivo, segundo a documentação. O SKAS utiliza dois chaveamentos de contexto e é mais rápido do que as TTs, e a abertura de */proc/mm* inicia um espaço de endereçamento novo e vazio para os processos UML [Dike, 2004]. Do ponto de vista da UML, o SKAS melhora o desempenho, mas não modifica sua funcio-

nalidade, enquanto que do ponto de vista do hospedeiro, os processos UML desaparecem, restando apenas os quatro essenciais, e há redução na carga do processador.

3.2. FAUmachine

O projeto FAUmachine derivou do UML, e é semelhante ao VMware ou VirtualPC. Esta máquina também executa como processo de usuário, e é restrita à arquitetura i386. Atualmente executa sobre o Linux, mas portes para OpenBSD e Windows já estão em progresso. Esta máquina é utilizada como plataforma experimental no projeto europeu DBench, que pesquisa *benchmark* de dependabilidade [FAUmachine, 2004], [Sieh and Buchacker, 2002].

Esta MV é iniciada por um carregador adaptado e executa um *kernel* Linux ligeiramente modificado. As características de memória, discos, cdrom e rede são configuráveis no momento da criação da máquina. Assim como no UML, é possível instalar uma distribuição Linux sobre a MV. Existe suporte à rede e a conexão pode ser feita transparentemente. A compatibilidade binária com respeito ao hospedeiro é mantida. A FAUmachine é acompanhada do *Experiment Controller Expect* (ECE), que pode executar experimentos completos a partir de um *script* e injetar falhas no *hardware* virtual.

A injeção de falhas pode ocorrer no núcleo de processamento, periféricos ou rede. Para efeitos da MV, o nível de *hardware* é o *kernel* hospedeiro, e a injeção de falhas é feita via interface *ptrace*. As possíveis falhas ocorrem: no núcleo de processamento, nos periféricos, nos componentes de rede externos, na configuração do sistema, no ambiente, na interação e na operação. Este conjunto permite avaliar a resposta do sistema sob carga pesada de processamento, e evita que se tenha que injetar falhas no SO nativo. O código do injetor e o da MV são separados.

O acesso ao processador é direto, isto é, não simula uma CPU. A memória RAM é simulada via arquivo mapeado em memória, e a unidade de gerência de memória é simulada via *mmap* e *munmap*. Os periféricos, incluso os de rede, também são simulados, e os *drivers* de dispositivos são construídos sobre chamadas de sistema ao SO nativo. O teclado e a console são implementados por um *xterm*, enquanto que dispositivos de rede são implementados por chamadas *socket*. A rede e o roteamento são configurados à parte, permitindo interação transparente com a rede real. *Traps* (interrupções e exceções) são implementados por sinais, e o relógio de tempo real usa *gettimeofday()*.

A injeção de falhas é disparada temporalmente, de forma pré-programada ou aleatória, e as falhas são configuradas em um arquivo descritivo. Utilizando-se o ECE, não é necessário interação humana com o injetor, e uma instância do ECE é suficiente para todas as máquinas virtuais de um hospedeiro. A FAUmachine não implementa proteção de memória do *kernel*, e o processo *Tracer* é também o injetor.

3.3. XEN

O XEN é um monitor para máquinas virtuais (*Virtual Machine Monitor* - VMM), que suporta a arquitetura x86. Ele foi concebido para suportar múltiplos SOs virtuais, e atualmente suporta Linux, com portes para Windows XP e NetBSD parcialmente concluídos [XEN, 2004], [Barham et al., 2003] e [Fraser et al., 2003].

Diferente das duas MVs anteriores, este VMM foi construído visando alto desempenho e forte isolamento entre SOs virtuais. Também neste caso um usuário instancia um VMM, que exporta uma interface para binários. O XEN não utiliza virtualização total da máquina hospedeira, mas um mecanismo denominado paravirtualização, no qual a abstração de *hardware* oferecida aos SOs virtualizados é similar, mas não igual ao *hardware* real. A virtualização total imporia penalidades maiores ao desempenho. Em vista disto, os SOs virtualizados precisam ter seu código-fonte portado para o XEN.

O mecanismo de funcionamento do XEN retira o *kernel* virtual do nível 0 (onde os SOs normalmente executam) e o transfere para o nível 1, passando ele próprio a executar neste nível. As aplicações continuam a executar no nível 3 (menos privilegiado). Cada SO virtualizado está contido e executa em um domínio. O XEN, propriamente, é denominado *hypervisor* e executa num domínio especial, Domínio0, criado na inicialização da máquina hospedeira. A virtualização oferecida pelo XEN divide-se em três áreas: gerência de memória, processador e periféricos. A gerência de memória é a parte mais complexa de tratar, tanto no *hypervisor* quanto no SO portado, em função de idiossincrasias da arquitetura x86. A virtualização do processador requer que o SO portado execute num nível menos privilegiado, neste caso, o nível 1. Instruções privilegiadas que o SO tenta executar são validadas pelo XEN, e falham caso o SO tente executá-las diretamente. Exceções, incluso falha de memória e *traps* de *software*, são virtualizadas por tabelas de manipuladores. A maior penalidade imposta ao desempenho ocorre nas chamadas de sistema e nas falhas de páginas de memória. Os dispositivos de entrada e saída (E/S) não são emulados, mas abstraídos. Esta interface melhora o isolamento entre os SOs virtualizados, e é construída usando *buffers* assíncronos em memória compartilhada.

Assim como interrupções de *hardware*, o XEN suporta um mecanismo de entrega de interrupções de *software*, ou seja, notificações aos domínios, baseado em mapas de bits. O *hypervisor* permite apenas o controle básico de operações, e o controle e gerência do *hypervisor* e domínios são feitos por um *software* de controle que executa no Domínio0. Este *software* controla os outros domínios, inclusive sua criação e terminação, bem como interfaces virtuais de rede e dispositivos.

4. Comparativo

Quanto ao **desempenho**, dentre as MVs discutidas, o XEN é a que apresenta melhor resultado, haja visto sua construção. O UML e a FAUmachine apresentam desempenhos semelhantes, com leve vantagem do primeiro. Possivelmente, o que torna a FAUmachine mais lenta que o UML são suas extensões para injeção de falhas e monitoramento.

A **transparência** é dependente da aplicação em desenvolvimento: se estiver no nível do usuário, todas as MV são elegíveis. Porém, se a aplicação precisar interagir com o SO, dependendo do nível a que se deve chegar, deve-se considerar características individuais das MVs. De modo geral, o UML é menos intrusivo, a não ser pelo SKAS. A FAUmachine se posiciona num meio-termo e o XEN é o mais intrusivo de todos. A **portabilidade** depende da continuidade do desenvolvimento das próprias máquinas.

O UML é uma espécie de meta-arquitetura suportada no Linux, e deve ser possível executá-lo nas arquiteturas de *hardware* suportadas pelo Linux. O XEN foi desenvolvido para a arquitetura x86. Quanto ao SO virtualizado, UML é restrito ao Linux, o mesmo acontecendo com a FAUmachine, enquanto que o XEN já está portando NetBSD e Windows XP. Com relação ao SO hospedeiro, o UML prevê um futuro porte para outros SOs. Quanto ao XEN, ele é derivado de um *kernel* Linux, mas por ser um VMM esta característica não se aplica estritamente.

Referente à **intrusividade** em termos de aplicações, nenhuma MV é intrusiva. Com respeito ao *kernel*, o XEN é o mais intrusivo de todos.

Todas as MVs têm código-fonte aberto, e portanto a **instrumentabilidade** e **monitorabilidade** podem ser acrescidas às máquinas, se já não existirem. Em particular, a FAUmachine, por seus propósitos, já oferece um grau de instrumentabilidade maior.

Finalmente, na **transportabilidade** deve-se avaliar em que nível a operação deve ocorrer. Se for apenas a imagem da MV, a FAUmachine é transportável, assim como o

UML, ainda que o novo hospedeiro tenha que oferecer suporte a elas. O XEN não é facilmente transportável, pois envolveria o transporte do carregador e a partição hospedeira.

5. Conclusão

Dadas às características das máquinas, pode-se afirmar que são utilizáveis como ferramentas de suporte ao desenvolvimento de aplicações tolerantes a falhas. A escolha entre as máquinas apresentadas depende em parte da natureza da aplicação. Testes estão sendo conduzidos para aprofundar o conhecimento sobre as capacidades e limitações, bem como a adequação de cada máquina em função dos tipos de aplicações. A viabilidade de outras máquinas virtuais também está sendo considerada, e será direcionada, caso necessário, pelas limitações encontradas nas MVs aqui apresentadas.

Referências

- Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. In *Proceedings of the 19th ACM symposium on Operating systems principles*, pages 164–177.
- Buchacker, K. and Sieh, V. (2001). Framework for testing the fault-tolerance of systems including os and network aspects. In *The 6th IEEE International Symposium on High-Assurance Systems Engineering*, pages 95–105.
- Dike, J. (2004). The user-mode linux kernel home page. URL: <http://user-mode-linux.sourceforge.net/>.
- FAUmachine (2004). Faumachine project. URL: <http://www3.informatik.uni-erlangen.de/Research/UMLinux>.
- Figueiredo, R. J., Dinda, P. A., and Fortes, J. A. B. (2003). In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, pages 550–559.
- Fraser, K. A., Hand, S. M., Harris, T. L., Leslie, I. M., and Pratt, I. A. (2003). The xenoserver computing infrastructure. Technical Report 552, Computer Laboratory, University of Cambridge, Cambridge, United Kingdom.
- Höxer, H.-J., Buchacker, K., and Sieh, V. (2002). Implementing a user-mode linux with minimal changes from original kernel. In *9th International Linux System Technology Conference*, pages 72–82, Köln, Germany.
- Kohlbrenner, E., Morris, D., and Morris, B. (2004). The history of virtual machines. URL: <http://cne.gmu.edu/itcore/virtualmachine/history.htm>.
- Liang, Z., Venkatakrishnan, V. N., and Sekar, R. (2003). Isolated program execution: An application transparent approach for executing untrusted programs. In *Proceedings of the 19th Annual Computer Security Applications Conference*, pages 182–191.
- Plex86 (2004). The plex86 x86 virtual machine project. URL: <http://www.plex86.org>.
- Sieh, V. and Buchacker, K. (2002). Umlinux - a versatile swifi tool. In *Proceedings of the 4th European Dependable Computing Conference on Dependable Computing*, pages 159–171.
- Thain, D. and Livny, M. (2003). Parrot: Transparent user-level middleware for data-intensive computing. In *Workshop on Adaptive Grid Middleware – AGridM 2003*.
- VMware Inc. (2004). Vmware. URL: <http://www.vmware.com>.
- XEN (2004). Xen: The xen virtual machine monitor. URL: <http://www.cl.cam.ac.uk/Research/SRG/netos/xen/>.

Modelagem de Falhas para Simulação de Sistemas Distribuídos*

Ruthiano S. Munaretti Marinho P. Barcellos

¹PIPCA - Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Unisinos - Universidade do Vale do Rio dos Sinos
Av. Unisinos, 950 - São Leopoldo, RS - 93022-000

{ruthiano,marinho}@exatas.unisinos.br

Resumo. É reconhecida a dificuldade de se testar experimentalmente sistemas distribuídos tolerantes a falhas. Simulação fornece um ambiente controlado para investigação e testes perante situações atípicas nestes sistemas. O presente artigo aborda uma extensão do framework Simmcast de forma a permitir a simulação de sistemas distribuídos tolerantes a falhas. Para tal, define um modelo de falhas suportado, e como as mesmas são mapeadas nos componentes fundamentais do Simmcast, nodo e caminho. Por fim, introduz o conceito de cenários de falhas: combinações pré-concebidas de comportamentos de falhas e objetos que facilitam a especificação de falhas a serem simuladas durante um experimento.

1. Introdução

É reconhecida a dificuldade de se testar experimentalmente sistemas distribuídos tolerantes a falhas, particularmente em sistemas de maior porte na Internet. Simulação, nesse caso, pode ajudar a remediar o problema, pois representa um passo intermediário antes da implementação completa e utilização do sistema. Simulação fornece um ambiente controlado e facilita a análise da interação entre os componentes, permitindo testar cenários difíceis de serem reproduzidos.

O Simmcast ([1]) é um framework para simulação discreta de protocolos e sistemas distribuídos. Sua arquitetura é adequada à simulação de sistemas distribuídos, no entanto presentemente não existe suporte algum a experimentos com falhas. Almeja-se estender o Simmcast de forma a permitir a execução de experimentos com sistemas distribuídos tolerantes a falhas, tal como modelagem de protocolos de consenso e detectores de defeitos. A versão estendida do Simmcast, denominada Simmcast-FT, permitirá criar ambientes simulados onde falhas ocorram nos componentes fundamentais de uma topologia em momentos específicos e/ou segundo condições especificadas no arquivo que descreve a simulação. Este artigo representa o primeiro passo nesse sentido, descrevendo o projeto da arquitetura do sistema, sua API e as justificativas para as principais decisões de projeto.

O artigo está organizado da seguinte forma: a Seção 2 indica quesitos necessários para a simulação de sistemas distribuídos tolerantes a falhas e revisa trabalhos relacionados. A Seção 3 mostra a arquitetura do framework de simulação Simmcast através de uma ótica hierárquica, em camadas. A Seção 4 discute quais são os componentes do sistema subjacente sujeitos a falhas, e como as mesmas podem ser modeladas. A Seção 5 encerra o trabalho e faz considerações sobre trabalhos futuros.

*Este trabalho foi desenvolvido com apoio do CNPq.

2. Simulação de Falhas em Sistemas Distribuídos

Simulação discreta é uma prática popular no dimensionamento de redes de computadores, e tem sido amplamente usada no projeto de novos protocolos de comunicação ou avaliação de desempenho de protocolos existentes em novos cenários de rede. Exemplos de simuladores de redes são OPNET ([2]), VINT NS-2 ([3]), SSF ([4]) e Simmcast ([1]). Simuladores oferecem um ambiente controlado para validar o comportamento de protocolos existentes, fornecem infra-estrutura para desenvolvimento de novos protocolos e oportunizam o estudo de suas interações. É natural que esta técnica seja empregada também no desenvolvimento de protocolos e sistemas distribuídos tolerantes a falhas; exemplos incluem [5], [6] e [7]. Este último oferece um ambiente orientado a objetos para teste de implementações de protocolos distribuídos tolerantes a falhas. Seu foco é em sistemas de tempo-real, cujo processo de testes em ambientes reais é especialmente complexo.

O NS-2 é o simulador mais popular no ambiente acadêmico nos dias de hoje, sendo bastante usado para investigação de novos protocolos de comunicação e seus mecanismos de controle. NS-2 e Simmcast possuem diversos aspectos em comum. Particularmente, ambos envolvem a construção de novos componentes e descrição de um cenário de simulação, conforme a seguir. Primeiro, um simulador pode ser estendido criando-se novos componentes, modificando-se ou aumentando-se os existentes, resultando em novos protocolos, geradores de carga e tecnologias de transmissão. Isto é feito de forma a refletir as características e funcionamento desejados do experimento realizado. Segundo, a descrição de um cenário de simulação consiste na indicação dos diversos componentes que serão utilizados no experimento, bem como da ligação e interação entre eles. Isto é realizado através de uma linguagem de configuração específica, usualmente do tipo *script*.

Na simulação de sistemas distribuídos tolerantes a falhas, são especificadas pelo usuário quais tipos de falhas ocorrerão no ambiente e sob que circunstâncias. Portanto, além de uma “categoria de falha”, deve ser indicada uma condição que define o momento em que o componente falhará, quando uma falha é ativada, e o momento que a falha cessará. Vislumbra-se casos em que falhas são ativadas e/ou desativadas de forma *determinística*, segundo uma situação específica, e *probabilísticas*, quando determinados componentes do sistema subjacente têm associadas propriedades de falha. Exemplos são, respectivamente, quando um nodo falha estando os demais nodos em um determinado estado, e quando um determinado nodo é configurado com tempo médio entre falhas e tempo médio de recuperação. Objetiva-se um modelo de especificação de falhas conciso, mas que seja flexível e poderoso.

3. Arquitetura do Simmcast

Esta seção descreve a arquitetura do Simmcast, mostrando suas camadas, e enfatizando os conceitos que servem de base para a extensão do *framework* com suporte à simulação de falhas; não é objetivo da mesma descrever o Simmcast como um todo (vide [1, 8, 9]). O Simmcast é um *framework* de simulação cuja construção pode ser descrita através de um conjunto de camadas, onde cada camada é desenvolvida de forma independente das superiores, fazendo o uso de serviços oferecidos pelas inferiores. A arquitetura, um histórico e análise das principais decisões de projeto podem ser encontrados em [9].

Na arquitetura, a **camada 0 (Java)** oferece o suporte a *threads*, unidade básica de execução do simulador, enquanto a **camada 1 (Simmcast Engine)** é o motor de execução do simulador, uma máquina de eventos discretos baseada em processos. Estes processos são uma especialização das *threads* Java. A **camada 2 (kernel do simulador)** é a primeira camada visível ao usuário do simulador, sendo na mesma definidos os diversos componentes que formam o *framework*. Nodos (Node) e caminhos (Path) são definidos

nesta camada, sendo que a execução de um nodo é definida através de um conjunto de processos definidos na camada 1. A **camada 3 (roteamento)** define componentes como roteadores (RouterNodes) e estações (HostNodes), bem como algoritmos de roteamento necessários em simulações de redes com modelagem da topologia física. A **camada 4 (protocolo)** contém a especialização dos diversos componentes definidos em camadas inferiores (como Node e HostNode, por exemplo), a fim de se adicionar a lógica do protocolo desejado. A **camada 5 (aplicação)** é onde se situa a aplicação de usuário para execução no simulador. A **camada 6 (configuração da simulação)** é utilizada para modelagem de componentes e comportamento da rede, o que se dá através de um arquivo de configuração, definindo assim um cenário de simulação.

Para suporte à simulação de falhas, os itens mais importantes da arquitetura do simulador estão localizados na camada 2. Esta contém os dois componentes fundamentais do Simmcast, utilizados na composição de topologias: nodos e caminhos, ou **Node** e **Path**, respectivamente. Um nodo é uma entidade abstrata no Simmcast que pode representar elementos diferentes de um sistema real, contendo uma ou mais *threads* (classe **NodeThread**, especialização de um processo definido na camada 1), que compartilham memória e “rodam sobre o mesmo hardware”. Um caminho (classe **Path**), por sua vez, representa a conexão lógica entre dois nodos quaisquer, criando um caminho através do qual pacotes (classe **Packet**, também na camada 2) podem ser transmitidos.

4. Suporte à Simulação de Falhas no Simmcast

Esta seção discute a extensão do framework para suporte à simulação de falhas. A interface utilizada para especificar falhas no sistema é ponto chave para o sucesso do simulador, e ela se manifesta de duas formas: através da estrutura de classes do *framework* (como instâncias de seus componentes são criados e acessados), e do arquivo que descreve um experimento de simulação. Isto reflete no projeto do Simmcast-FT, conforme apresentado nas subseções a seguir.

4.1. Extensão de Componentes com um Modelo de Falhas

Um sistema distribuído tolerante a falhas admite um determinado conjunto de falhas, conforme sua especificação, em termos de funcionalidade, de premissas sobre o ambiente e do modelo de falhas adotado. O modelo de falhas, tal como [10, 11, 12, 13], define um conjunto de categorias de falhas; a especificação de um sistema distribuído tolerante a falhas pode então se basear na nomenclatura apresentada pelo modelo, visando definir precisamente quais condições são tratadas, o que assume-se não ocorrer, e o que não é tratado. O modelo de falhas adotado no Simmcast-FT, largamente baseado em [13], é apresentado na Tabela 1.

Tabela 1: Tipos de falha.

Tipo de Falha	Descrição
<i>Colapso</i>	Componente pára silenciosamente de funcionar
<i>Colapso com Recuperação</i>	Componente pára silenciosamente de funcionar, mas pode retornar mais tarde (com amnésia de estado)
<i>Omissão</i>	Componente omite resultados, de forma completa ou parcial
<i>Temporização</i>	Funcionamento com tempo arbitrário
<i>Sintática</i>	Comportamento incorreto, detectável
<i>Semântica</i>	Comportamento correto com sentido incorreto

Conforme indicado anteriormente, o suporte à simulação de falhas no *framework* é realizado através da extensão dos componentes básicos do Simmcast, nodo e caminho. No

Simmcast-FT, a interface destes componentes passa a oferecer métodos que são usados para estabelecer um comportamento de falhas para a instância do objeto correspondente. Existe um método para cada tipo de falha, de acordo com a Tabela 1.

Para especificação da condição de ativação da falha, métodos tomam como argumento de entrada um *predicado de ativação*: o mesmo é periodicamente avaliado, e quando resulta verdadeiro, a falha é ativada no componente. Quando a falha é ativada, o componente muda de estado e duas ações ocorrem instantaneamente: atuação da falha sobre o componente (por exemplo, causando a perda de mensagens), e avaliação de um *predicado de desativação*. Caso este seja verdadeiro, a falha será imediatamente desativada, sendo percebida como uma *falha efêmera*. Caso contrário, a falha continua ativa e o predicado de desativação continua a ser avaliado de forma periódica, até que se torne verdadeiro ou a simulação acabe. A peridiocidade para avaliação de predicados é definida pelo usuário, sendo o valor padrão configurado em 1 (unidade de tempo de simulação). No momento em que uma falha é ativada em um componente, o tempo é registrado como atributo do componente; enquanto permanecer ativa, o predicado de ativação não é avaliado.

Na expressão booleana que representa um predicado podem aparecer as constantes *True* e *False*, o valor atual do relógio, o tempo de ativação da última falha do componente, um valor extraído de uma distribuição aleatória definida pelo usuário, e um método a ser executado sobre o componente. Portanto, o usuário aciona a falha indicando no arquivo .sim o seguinte método:

```
<node>.<FaultType>(<on>, <off> [, <args>])
```

onde *<node>* é a identificação do objeto nodo, *<FaultType>* é o tipo de falha, *<on>* é o predicado de ativação e *<off>* de desativação. Dependendo do tipo de falha, argumentos opcionais (indicados acima por *[, <args>]*) são utilizados. A seguir, descreve-se o mapeamento das diferentes categorias de falha nos componentes fundamentais do Simmcast, nodo e caminho.

4.2. Falhas de Nodo

O tipo mais simples de falha de um nodo é a de **colapso**, quando o mesmo silenciosamente pára de funcionar. Todas as mensagens que se encontravam em propagação em conexões diretas com o nodo, ou nas filas de envio na camada subjacente do mesmo, são silenciosamente perdidas; o estado do nodo é apagado (amnésia). Em caso de colapso sem recuperação, o predicado off será *False*. De outra forma, será especificado em função do relógio atual, o tempo da última falha e o tempo (médio) de recuperação. Com falha de **temporização** em nodos, é possível configurar um valor arbitrário para um relógio de um nodo, bem como uma dada velocidade de avanço, de forma que divirja do relógio da simulação.

O restante das falhas de nodos refletem na troca de mensagens. Quando um nodo falha por **omissão**, ele deixa de ser visto pelos demais, o que corresponde ao descarte de todas as mensagens enviadas pelo nodo. Quando uma falha **sintática** em um nodo é ativada, mensagens enviadas pelo nodo assumem uma sintaxe incorreta; predicados probabilísticos podem ser usados para que parte das mensagens seja afetada. A falha **semântica** de um nodo, quando ativada, faz com que o mesmo passe a enviar mensagens que são sintaticamente corretas (e portanto não geram exceção), mas contendo valores incorretos (semântica incorreta).

4.3. Falhas de Caminhos

A arquitetura de falhas do Simmcast-FT estende, de forma elegante, o modelo de falhas definido para nodos, para os componentes que representam caminhos. Assim, caminhos

estão sujeitos às mesmas categorias de falhas do que nodos, e podem também ser ativados ou desativados via predicados. Falhas em caminho, como na vida real, impactam na transmissão de mensagens.

Quando uma falha de **colapso** é ativada em um caminho, o transporte de mensagens pelo mesmo é encerrado, com descarte silencioso de quaisquer mensagens futuras ou em propagação, em ambas as direções. Assim como em nodos, colapsos podem ser modelados com e sem recuperação, dependendo do predicado de desativação. O tipo de falha mais comumente associado a caminhos é o de **omissão**, com descarte de parte das mensagens. Falhas de **temporização** aplicadas a caminhos fazem com que mensagens transportadas sejam atrasadas segundo um valor constante ou obtido de uma distribuição.

Falhas **sintáticas** são aplicadas a caminhos de forma a embaralhar, ou seja, corromper, mensagens: tal como para nodos, assume-se que mensagens afetadas sejam detectáveis por receptores. Por fim, caminhos podem ser alvo de falhas **semânticas**: quando tal falha é ativada, todas as mensagens que são transportadas pelo caminho, incluindo aquelas que se encontram em propagação, podem ter seus valores internos alterados.

4.4. Cenário de Falhas

Nas seções anteriores, foram apresentadas as diversas categorias de falhas e como as mesmas são mapeadas nos componentes do Simmcast. Embora a interface resultante seja concisa (extensão de apenas duas classes), elegante (simétrica, aplica os mesmos métodos às classes) e flexível (devido a forma com que falhas são ativadas e desativadas), vislumbra-se a combinação de um certo conjunto de falhas em componentes chamados “cenários de falhas”. Os cenários podem ser usados para representar casos típicos, que expressem um modelo comum, ou que sejam próximos da realidade. Estas configurações prontas ou personalizadas, fornecidas junto ao *framework*, tem o intuito de facilitar o uso do modelo de falhas no Simmcast-FT.

Em diversas situações, é desejável que uma determinada falha seja aplicada a um conjunto de nodos. Para tal, a classe **Network** é estendida de forma a oferecer métodos para seleção de um conjunto de nodos ou um conjunto de caminhos. Os critérios para seleção passíveis de aplicação são os seguintes: **aleatório**, segundo uma dada probabilidade; **aleatório**, porém aplicado a um **subconjunto** de nodos/caminhos; **regional**, selecionando um conjunto de nodos segundo o identificador de um nodo central e um valor de escopo. Esta facilidade pode ser usada então, via arquivo “.sim”, para selecionar uma região da topologia, ou um conjunto aleatório de nodos, e então sobre os mesmos aplicar um determinado comportamento de falhas.

5. Conclusões

Simulação fornece um ambiente controlado e facilita o teste das condições específicas que podem ocorrer em sistemas distribuídos tolerantes a falhas e que são difíceis de se reproduzir. Este artigo apresentou o projeto da API do Simmcast-FT, uma extensão do Simmcast para execução de experimentos com sistemas distribuídos tolerantes a falhas. A interface do Simmcast-FT permite descrever ambientes onde falhas ocorrem em componentes individuais ou em conjuntos de componentes em momentos específicos e/ou segundo condições especificadas no arquivo que descreve a simulação.

Este é um trabalho em andamento, e limita-se a apresentar a extensão da arquitetura do Simmcast, definindo um modelo de falhas suportado e um mapeamento em componentes do Simmcast-FT. Por fim, introduz-se o conceito de cenários de falhas, combinações de comportamentos de falhas e objetos que facilitam a especificação de falhas a

serem simuladas durante um experimento. A arquitetura se encontra em fase de prototipação, porém os resultados iniciais são animadores e serviram como “prova de conceito” e incentivo para a realização da mesma.

Referências

- [1] M. P. Barcellos, H. H. Muhammad, and A. Detsch. Simmcast: a simulation tool for multicast protocol evaluation. In *XIX Simpósio Brasileiro de Redes de Computadores, SBRC 2001*, volume 1, pages 418 – 433, Florianópolis, Brasil, 2001.
- [2] Opnet technologies. <http://www.opnet.com/>.
- [3] The Network Simulator VINT ns-2. <http://www.isi.edu/nsnam/ns>.
- [4] Scalable simulation framework - ssf. <http://www.ssfnet.org/>.
- [5] C. Ciarfella, L. Moser, P. Melliar-Smith, and D. Agarwal. The Totem Protocol Development Environment. In *International Conference on Network Protocols, ICNP’94*, 1994.
- [6] B. Weiss, G. Gridling, U. Schmid, and K. Schossmaier. The SimUTC Fault-Tolerant Distributed Systems Simulation Toolkit. In *7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, March 1999.
- [7] G. A. Alvarez and F. Cristian. Cesium: Testing hard real-time and dependability properties of distributed protocols. In IEEE, editor, *3rd Workshop on Object-Oriented Real-Time Dependable Systems - (WORDS ’97)*, Newport Beach, 1997.
- [8] H. H. Muhammad, G. B. Bedin, G. Facchini, and M. P. Barcellos. Quebrando a barreira entre simulação e experimentação prática em redes de computadores. In SBC, editor, *XXII Simpósio Brasileiro de Redes de Computadores, SBRC 2004*, volume 1, Gramado, Brasil, maio 2004. SBC. a ser publicado.
- [9] M. P. Barcellos, G. Facchini, L. F. Cintra, and H. H. Muhammad. Projeto do framework de simulação simmcast: uma arquitetura em camadas com ênfase na extensibilidade. In SBC, editor, *XXII Simpósio Brasileiro de Redes de Computadores, SBRC 2004*, volume 1, Gramado, Brasil, maio 2004. SBC. a ser publicado.
- [10] K. Birman. *Building Secure and Reliable Network Applications*. Prentice Hall, 1996. 500 p.
- [11] Pankaj Jalote. *Fault Tolerance in Distributed Systems*. Prentice-Hall, 1998.
- [12] V. Hadzilacos and S. Toueg. *Distributed Systems*, chapter 5, Fault-Tolerant Broadcasts and Related Problems, pages 97–146. Addison-Wesley, 2nd. edition, 1998.
- [13] P. Veríssimo and L. Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, 2001.

FIJI – Uma Ferramenta para Validação Experimental do XFS

Leonardo Garcia de Mello, Taisy Silva Weber

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – CEP 91.501-970 – Porto Alegre – RS – Brazil
lmello@inf.ufrgs.br

Resumo. Sistemas de arquivos são componentes essenciais para servidores de alta disponibilidade, sendo preferível o uso daqueles que sejam seguros e relativamente independentes de ações por agentes humanos para a recuperação. Uma das abordagens para alcançar alta disponibilidade em sistemas de arquivos é a do tipo *journaling*, ou *log* de metadados. ReiserFS, ext3, JFS, e XFS são exemplos de implementações de *journaling* para Linux. Este artigo apresenta um injetor de falhas baseado em recursos de depuração para validação experimental do mecanismo de *journaling*. São mostradas as técnicas para construção do injetor e um teste aplicado sobre o XFS como sistema alvo.

1. Introdução

O termo comercialmente conhecido como “alta disponibilidade” (*HA* de *High Availability*) representa uma característica de sistemas computacionais projetados para evitar ao máximo as interrupções, planejadas ou não, na prestação de serviços. Em alta disponibilidade, o ideal é haver poucas falhas e, mesmo quando estas acontecerem, que o seu tempo médio de reparo (ou *MTTR*, de *Mean Time To Repair*) seja tão pequeno quanto possível.

Atualmente Linux é muito utilizado em servidores de redes [BAR 2000], em um cenário onde demandas quanto à disponibilidade geralmente são bastante elevados. Praticamente todas as plataformas atuais dispõem de sistemas de arquivos com alta disponibilidade [SEL 2000]. Para uso em aplicações de missão crítica são usados sistemas de arquivos baseados em *journaling* tais como ext3, JFS, ReiserFS e XFS para o Linux.

Este artigo propõe uma estratégia de validação experimental para avaliar a eficiência do mecanismo de *journaling* para sistemas de arquivos através de injeção de falhas por *software* [HSU 97]. Esta estratégia visa como sistema alvo uma implementação do XFS, um sistema de arquivos baseados em *journaling* para o Linux e é apoiada por uma ferramenta para injeção de falhas para sistemas de arquivos baseados em *journaling* – o FIJI.

2. Integridade para dados e metadados

Informações são organizadas em disco na forma de dados e metadados. Durante as operações envolvendo dados e metadados, é preciso que a representação do sistema de arquivos em disco seja mantida consistente – isso mesmo após a ocorrência de falhas. Em sistemas de arquivos como o ext2, construídos com alocação baseada em blocos, o sistema de arquivos pode ficar inconsistente após a ocorrência de falhas porque dados e metadados são gravados em disco de forma assíncrona. Sempre que inconsistências acontecem, é preciso executar algum utilitário de verificação - como o fsck [BAR 2000].

O fsck realiza uma série de verificações ao longo de todo o sistema de arquivos para validar as suas entradas e assegurar-se de que blocos alocados em disco estão todos sendo referenciados corretamente. Mas, para discos de grande volume, a execução do fsck pode consumir um tempo demasiadamente elevado comprometendo a disponibilidade do servidor. Em sistemas com muitos gigabytes em arquivos, por exemplo, a execução do fsck pode consumir até 10 horas ou mais [BAR 2000]. Conforme a severidade do caso, pode ser necessária a presença física do administrador do sistema para informar a senha de *root* e executar manualmente o programa de verificação.

Sistemas de arquivos para alta disponibilidade surgiram para contornar essas dificuldades. Por meio deles, é possível diminuir a chance de serem introduzidas inconsistências entre dados e metadados e reduzir o tempo médio de reparo (*MTTR*) na ocorrência de falhas. Atualmente as principais técnicas empregadas para auxiliar a obter alta disponibilidade em sistemas de arquivos Unix são o *journaling* e o *Soft Updates* [SEL 2000].

Enquanto a técnica de *Soft Updates* apenas é utilizada em sistemas BSD, a técnica de *journaling* baseia-se na redundância para aumentar a confiabilidade dos dados e metadados - mas sem aumentar significativamente os custos de hardware. Ela já é adotada por sistemas de arquivos em sistemas operacionais para plataformas diversas, tais como Solaris, AIX, Digital UNIX, HP-UX, Irix e Windows NT [SEL 2000]. Mas apesar de os sistemas de arquivos baseados em *journaling* terem sido adotados quase como um padrão pela indústria de *software*, atualmente não se encontram publicações sobre avaliação de medidas da sua disponibilidade.

3. Sistemas de arquivos baseados em journaling

Sistemas de arquivos baseados em *journaling* fazem um controle sobre as mudanças realizadas nos metadados (JFS, ReiserFS e XFS), ou nos dados e metadados (ext3) associados a um sistema de arquivos. A idéia consiste em tratar diferentemente dados e metadados, usando uma área dedicada em disco (o *log*, ou *journal*) para manter um histórico das mudanças. Tais sistemas implementam uma política de *write-ahead logging*, fazendo com que registros sejam armazenados no *log* antes de as operações serem efetuadas [SEL 2000].

Um sistema de arquivos deste tipo pode ser empregado para aplicações que lidem principalmente com um grande número de arquivos pequenos e façam uso freqüente da chamada de sistema *sync*. Isso acontece porque as várias alterações nos metadados são transferidas para o disco através de uma única operação, que acrescenta várias transações em uma mesma área [SEL 2000].

Uma grande vantagem da abordagem baseada em *journaling* é de que ela facilita obter alta disponibilidade em sistemas de arquivos já existentes. Exemplo disso é o resultado que pôde ser obtido com o sistema de arquivos ext2, que pôde ser reimplementado como sendo o ext3 [TWE 2000].

Quanto aos tempos de recuperação, para sistemas de arquivos baseados em *journaling* eles são bastante reduzidos. A tarefa de verificação consiste apenas em inspecionar as transações pendentes do *log* - ao invés de percorrer todos os blocos buscando inconsistências. Com isso, na ocorrência de defeitos, o sistema de arquivos pode ser levado a um estado consistente pela aplicação das transações pendentes no *log* - ao invés de ser necessário inspecionar toda uma unidade de disco com o *fsck*.

3.1. O sistema de arquivos XFS

O XFS foi criado ainda em 1994 pela SGI para substituir o EFS [XFS 2004]. Ele é o mais antigo entre os sistemas de arquivos conhecidos, e a ênfase do seu projeto foi na capacidade de trabalhar com arquivos bastante grandes - da ordem de "terabytes". O XFS pode trabalhar usando um tamanho de bloco variando entre 512 bytes e 64 kbytes, provendo o suporte para sistemas de arquivos distribuídos (incluindo NFS versão 3), ACLs no padrão POSIX 1003.e, e quotas para usuários e grupos.

O estado de um sistema de arquivos XFS é o resultado da combinação de informações localizadas em três locais diferentes: disco, memória e *log* do *journaling*. O sistema de arquivos XFS somente estará em um estado consistente após ele ter sofrido *shutdown*, onde todos os dados residentes em memória (*buffers* e *cache*) são gravados em disco e as entradas do *log* de transações são aplicadas no sistema de arquivos.

4. Injeção de falhas no XFS

Qualquer sistema computacional é construído baseado em um modelo de falhas, e para essas falhas é que são definidos e implementados mecanismos de detecção de erros ou recuperação. Para a realização de um experimento de injeção de falhas, faz-se necessária a definição de um modelo de falhas que seja tão próximo quanto possível das falhas reais a que estará sujeito o sistema alvo durante operação em ambiente de produção. Com base no modelo de falhas são definidos e armazenados os cenários de falhas para um experimento. Esses cenários permitem repetir um experimento tantas vezes quantas forem necessárias, emulando operação durante um período de tempo compatível com o necessário para obter a medida de dependabilidade definida pelo usuário. Para este artigo, o objetivo dos experimentos de injeção de falhas é determinar a cobertura da detecção das falhas que possam ocorrer durante a utilização de um sistema de arquivos baseado em *journaling* e o tempo médio de recuperação do sistema após falha.

A maioria dos mecanismos de tolerância a falhas disponíveis nos sistemas de arquivos baseados em *journaling* consideram principalmente as falhas de hardware transientes (mais especificamente de

falta de energia), onde assume-se um modelo de *crash* para o sistema. Isto representa uma situação em que o sistema de arquivos estaria sendo utilizado e ocorreria a falha, fazendo com que a máquina fosse reiniciada sem que ocorresse um *flush* das informações em memória (tais como *buffers* e *cache*) ou das transações no *log* (elementos que, conforme a Seção 3.1, definem o estado do sistema de arquivos). Não estão previstos no modelo outros tipos de falhas comuns - como alteração de conteúdo de memória ou corrupção de disco.

Este modelo de falhas define o cenário de falhas que é gerado pela ferramenta FIJI. FIJI injeta falhas de *crash* de processos para determinar a cobertura dos mecanismos de recuperação e medir o tempo médio de recuperação. Entretanto, o modelo vai além disso pois propõe-se a avaliar o comportamento do XFS sob falhas do meio de armazenamento - frequentes em disco - que possam provocar a corrupção dos *logs*.

5. Arquitetura para injeção de falhas

O injetor de falhas FIJI caracteriza um ambiente de falhas como definido na Figura 1 derivado de arquitetura proposta por Hsueh [HSU 97]. O sistema alvo, ou seja o sistema sob teste por injeção de falhas, é uma máquina com uma partição Linux formatada com o utilitário mkfs.xfs [XFS 2004].

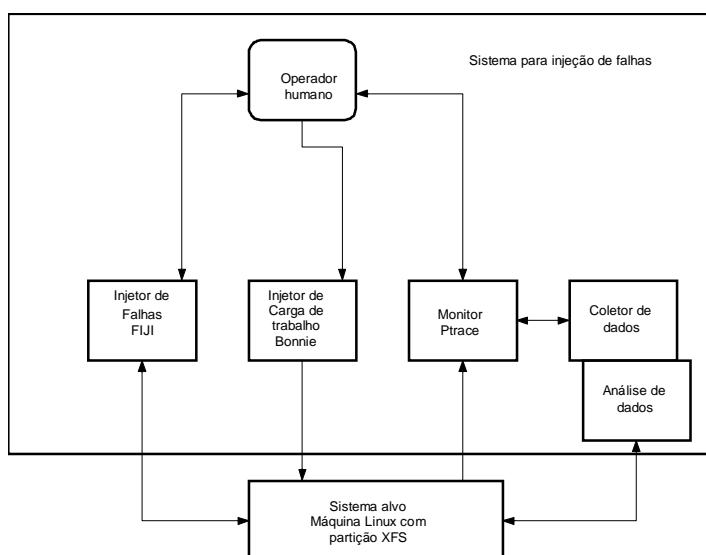


Figura 1 – Ambiente para injeção de falhas

O controle é exercido por um usuário, desenvolvedor ou avaliador de sistemas, que coordena a realização dos experimentos. A injeção de falhas é executada pela ferramenta FIJI, simulando a ocorrência de falhas do tipo *crash* e corrupção de meio magnético.

O utilitário *bonnie* atua como gerador de carga de trabalho, que deve gerar um *workload* correspondente às aplicações reais - tais como arquivos muito grandes (mais de 130 Mb), arquivos pequenos (menos de 600 Kb), servidor de arquivos, servidor de email, etc.

6. FIJI

A ferramenta FIJI (*Fault Injector for Journaling filesystems*), desenvolvida no PPGC da UFRGS, é baseada nos recursos de depuração do sistema operacional Linux (os quais permitem a interceptação e manipulação de chamadas de sistema). Estes recursos foram utilizados para a implementação de um injetor de falhas específico para sistemas de arquivos baseados em *journaling*. Manipular os parâmetros de chamadas de sistema (ou *system calls*) equivale a alterar as requisições feitas ao sistema operacional.

Conceitualmente, o FIJI localiza-se entre a aplicação alvo e o sistema operacional. As falhas injetadas pelo FIJI estão de acordo com o modelo de falhas descrito anteriormente, e encontram-se especificadas no código-fonte da própria ferramenta. O FIJI é implementado como um processo

concorrente à aplicação alvo, usando os recursos oferecidos pelo sistema operacional Linux através da chamada de sistema *ptrace* [BAR 2000]. A chamada de sistema *ptrace* permite executar um processo de três maneiras: passo-a-passo, usando *breakpoints* ou executá-lo até a próxima chamada de sistema.

Para que seja possível controlar a execução de um processo com *ptrace*, é preciso que o FIJI esteja conectado ao processo. A função *ptrace* permite que isso seja conduzido de duas formas [LAD 98]:

- Conectando-se a um processo em execução, por meio da requisição PTRACE_ATTACH; ou
- Ativando-se o *flag* de depuração do processo antes de executá-lo, da seguinte forma: quando duplica-se um processo via *fork*, ativa-se o *flag* de depuração via uma requisição PTRACE_TRACEME.

A partir do momento em que o FIJI é executado, ele cria uma cópia de si mesmo através da função *fork* [TAN 97]. Neste novo processo gerado (processo filho) ativa-se o *flag* de depuração chamando-se a função *ptrace* com a requisição PTRACE_TRACEME descrita anteriormente. A seguir, o gerador de carga de trabalho, no caso o *bonnie*, é executado por meio da função *execve*. Em função de ter o *flag* de depuração ativado, na primeira chamada de sistema encontrada o processo filho pára e sinaliza o FIJI. Enquanto isso, o FIJI espera que o processo filho (já executando comandos) indique estar pronto para ser depurado. A espera é realizada com o auxílio da função *wait4*.

O FIJI executa uma requisição ao sistema operacional via função *ptrace* e espera, via função *wait4*, que o gerador de carga de trabalho sinalize o fim da execução. O sinal enviado, neste caso, é o SIGTRAP.

O FIJI recebe diretamente do processo filho a indicação de que está parado. Após receber o sinal de que o gerador de carga de trabalho encontrou uma chamada de sistema e parou, o FIJI pode requisitar ao sistema operacional que leia ou escreva na memória e nos registradores.

Agora, o FIJI pode inspecionar e manipular a memória do processo, o valor de seus registradores e a sua estrutura *user*. Isto é feito executando-se a chamada da função *ptrace* com o parâmetro PTRACE_GETREGS.

Por meio da chamada da função *ptrace* com o parâmetro PTRACE_GETREGS, é possível obter o valor de todos os elementos descritos na Figura 2. O trecho de código que segue foi obtido a partir do código-fonte para o *kernel* versão 2.6.6 para o sistema operacional Linux, no arquivo /usr/src/linux/include/asm/user.h.

```
struct user_regs_struct {
    long ebx, ecx, edx, esi, edi, ebp, eax;
    unsigned short ds, __ds, es, __es;
    unsigned short fs, __fs, gs, __gs;
    long orig_eax, eip;
    unsigned short cs, __cs;
    long eflags, esp;
    unsigned short ss, __ss;
};
```

Figura 2 – Elementos acessíveis à chamada de sistema *ptrace*

Assim que o FIJI lê o valor dos registradores, ele faz um teste para verificar se o conteúdo do registrador EAX é 03 (o que corresponde a uma chamada de sistema *read*) ou 04 (o que corresponde a uma chamada de sistema *write*) [BAR 2000]. Caso seja uma dessas, ele modifica o conteúdo do registrador EDX – o qual contém o tamanho do *buffer* utilizado. Depois disso, o FIJI executa novamente a função *ptrace* com o parâmetro PTRACE_SETREGS para zerar o registrador EDX e continuar executando a *syscall* com este parâmetro modificado. Zerando o registrador EDX, a operação de leitura ou escrita torna-se nula – mantendo o meio de armazenamento inalterado, segundo o modelo de falhas.

Apesar de haver amplas possibilidades para injeção de falhas, visto que podem ser alterados o conteúdo de registradores e posições de memória através da função *ptrace*; este recurso deve ser usado com moderação. Isso é necessário porque pedir a um processo que execute até um ponto de parada exige um custo computacional elevado em chaveamentos de contexto.

Existe um parâmetro, que deve ser informado para o FIJI pela linha de comando, informando dentro de quantos segundos o log será descartado e as chamadas de sistemas começarão a ser suprimidas. O descarte de log do journaling permite simular uma situação em que houve uma corrupção no meio de armazenamento. Para o modelo de falhas definido, basta realizar estas ações para comprometer a integridade do sistema de arquivos XFS.

7. Teste da ferramenta

Nesta seção são apresentados os testes iniciais da ferramenta, e a comprovação da injeção de falhas realizada segundo o modelo de falhas proposto. Com o uso de um programa exemplo (cujo código-fonte está na Figura 3), é feita a injeção de falhas e os resultados são comprovados inspecionando um arquivo de saída.

Visando o teste da ferramenta FIJI, foi criada uma aplicação que armazena seqüências de caracteres em um arquivo. A seqüência e o resultado final de uma execução desta aplicação já são conhecidos. Alterações no tamanho do arquivo gerado ou em seu conteúdo significam que o injetor está agindo sobre a aplicação.

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
main()
{
    int fd;
    int i,c;
    char * msg1="AAAAAAA\n";
    char * msg2="BBBBBBB\n";

    if ((fd=open("arq_test.txt", O_SYNC | O_RDWR | O_CREAT, S_IRWXU |
S_IRWXG | S_IROTH)) < 0) {
        printf("Erro ao abrir o arquivo");
        exit(1);
    }
    for (i=0; i<5000; i++) {
        c = write(fd, msg1, 8);
        c = write(fd, msg2, 8);
    }

    close(fd);
    return(0);
}
```

Figura 3 – Listagem do programa para teste do FIJI

A metodologia de testes empregada é como segue: tomando-se este programa, com o resultado de execução já conhecido, pode-se injetar falhas e observar o resultado da execução. Como o roteiro de falhas é previamente definido, o resultado da execução – desde que o injetor funcione corretamente – também é conhecido.

Ao fim de uma execução correta do programa exemplo, existe um arquivo chamado arq_test.txt com 80.000 bytes. Ele contém seqüências alternadas de frases “AAAAAAA” e “BBBBBBB” – sempre seguidas por um avanço de linha (caractere “\n”) – vide Figura 4.

Depois de compilado este programa, a execução do mesmo apresenta o resultado da Figura 4:

```
[root@mail root]# gcc prog_test.c -o prog_test
[root@mail root]# ./prog_test
[root@mail root]# ls -l arq_test.txt
-rwxr-xr-- 1 root root 80000 Abr 30 08:53 arq_test.txt
```

Figura 4 – Resultado da execução do programa de teste do FIJI

No teste para injeção de falhas, especificou-se que 3 segundos após o início do processo, o *log* deveria ser destruído e as chamadas de sistema *read/write* descartadas. Uma nova execução do programa, agora com injeção de falhas, fornece como resultado a saída que está na Figura 5:

```
[root@mail root]# ./fiji -C 3 ./prog_test
...
[root@mail root]# ls -l arq_test.txt
-rwxr-xr-- 1 root root 1616 Abr 30 09:06 arq_test.txt
```

Figura 5 - Resultado da execução do programa de teste com o FIJI

Os defeitos provenientes dos erros injetados no experimento, observados no resultado da execução do programa exemplo, são condizentes com as regras de injeção de falhas consideradas. Desta forma, o funcionamento da ferramenta FIJI foi comprovado.

8. Conclusões

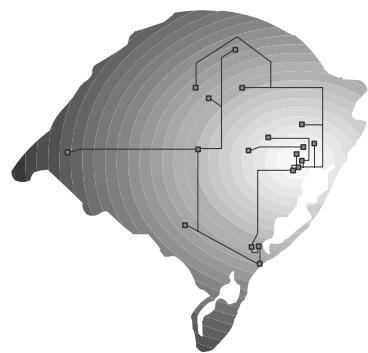
Atualmente estão sendo conduzidos experimentos para validar experimentalmente sistemas de arquivos baseados em *journaling* como o XFS e consolidar o ambiente proposto de acordo com uma metodologia desenvolvida pelo grupo de tolerância a falhas da UFRGS e aplicada anteriormente com sucesso em validação de mecanismos de recuperação em banco de dados.

Os experimentos estão direcionados para XFS em Linux, mas espera-se que a metodologia seja suficientemente genérica e portável para permitir a validação de outros sistemas de arquivos baseados em *journaling*. Testes preliminares comprovam a adequação da ferramenta ao modelo de falhas.

Os experimentos já realizados no XFS usando FIJI comprovam a eficiência do mecanismo de *journaling* para *crash* de processo. No momento estão sendo conduzidos os experimentos para avaliação do comportamento do XFS sob falhas do meio de armazenamento.

9. Referências bibliográficas

- [BAR 2000] BAR, Moshe. **Linux Kernel Internals**. New York: McGraw-Hill, 2000, 351 p.
- [HSU 97] HSUEH, Mei-Chen; TSAI, Timothy K.; IYER, Ravishankar K. Fault-Injection Techniques and Tools. **Computer**, v. 30, n. 4, Apr 1997, p.75-82.
- [LAD 98] JOHNSON, Michael K. **Linux application development**. [S.l.]:Addison-Wesley, 1998. 538 p.
- [SEL 2000] SELTZER, M. I.; et. al. *Journaling* versus soft-updates: Asynchronous meta-data protection in file systems. In: USENIX ANNUAL TECHNICAL CONFERENCE, 2000. **Proceedings...** San Diego, California, USA, 2000. Disponível em: <<http://www.lcs.ece.cmu.edu/~soule/papers/seltzer.pdf>>. Acesso em: ago. 2003.
- [TWE 2000] TWEEDIE, Stephen C. Ext3, *Journaling* Filesystem for Linux. In: OTTAWA LINUX SYMPOSIUM, 2000. **Proceedings...** Disponível em: <<http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html>>. Acesso em: out. 2003
- [XFS 2004] Sistema de arquivos XFS para Linux. Disponível em: <<http://xfs.sourceforge.net>>. Acesso em: abr. 2004.



Sessão Técnica 5

Protocolos e Serviços

Uma Visão Geral das Soluções para Uso de IPv6

Andrey Vedana Andreoli, Liane Tarouco

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

{andrey, liane}@penta.ufrgs.br

Resumo. Este artigo tem por objetivo apresentar um breve relato sobre algumas das principais soluções para uso de IPv6 e interoperabilidade com IPv4, discutidas pelo IETF e em redes experimentais ao redor do Mundo. Trata-se do início de um estudo inicialmente teórico que visa esclarecer os pontos fortes, as limitações e contextos onde cada solução pode ser melhor aplicada. Tais resultados têm fornecido subsídios para testes de implementações em uma rede de testes, que na medida do possível têm sido transportados para ambientes de produção.

1. Introdução

Como consequência do vasto crescimento da Internet, algumas limitações têm se apresentado na versão atual do Internet Protocol, conhecido como IPv4. A principal das limitações envolve o espaço de endereçamento, atualmente com 32 bits. Com a explosão de dispositivos conectados à Internet, o total de endereços únicos que podem ser utilizados no IPv4 tem se tornado limitado e isso clamou por uma nova tecnologia [COM01]. Os objetivos desta nova tecnologia seriam de aumentar o espaço de endereçamento, oferecer novos recursos para as aplicações que tem surgido e para melhorias na infra-estrutura de rede.

Em 1994, iniciou-se o estudo pelo IETF de uma nova versão do protocolo IP, que surgiu a partir de 1995 através da RFC 1883 [DEE95]. A nova versão do protocolo IP foi intitulada IPng – IP Next Generation.

Em diversos aspectos o protocolo IPv6 [DEE98] assume as mesmas funcionalidades que fizeram o sucesso do IPv4 [ISI81]. Por trata-se de uma inovação que estará incorporada nos diversos níveis da Internet, sua entrada deve ser gradual e capaz de manter a interoperabilidade com o atual IPv4.

A área de roteamento - como sendo o nível mais baixo onde o IPv6 atua - exerce um papel importante e essencial, já que através deste nível operam os demais níveis da Internet, a citar: gerência e segurança de redes, além do desenvolvimento de aplicações.

A área de roteamento representa um conjunto vasto de recursos, formando então a chamada infra-estrutura de rede responsável pelo funcionamento do IPv6. Inicialmente, devido à situação que grande parte dos equipamentos de rede não tinha suporte nativo a IPv6, tornou-se comum a utilização de túneis, fazendo com que o tráfego IPv6 fosse roteado sobre redes IPv4 de forma transparente. No entanto, tal flexibilidade implicava em overhead e ainda na ineficácia do cálculo dos melhores caminhos para determinadas redes IPv6.

A primeira utilização de IPv6 em túneis de maior abrangência surgiu em 1996, chamada de 6Bone [BON04], sendo definida pelo IETF como uma rede de testes, para prover conectividade IPv6 a redes interessadas e contribuir para sua expansão. A adesão a este projeto foi muito grande pelo motivo de ser uma forma simples de uma rede possuir conectividade IPv6 sem adquirir novos equipamentos, permitindo testes com novas aplicações e a experimentação do IPv6 na prática.

A partir daí, diversos recursos tem surgido, além do enfoque de uso de IPv6 não ser mais apenas para validação da tecnologia, mas de uso em produção e sendo objeto de pesquisa para muitas redes experimentais como: Europa com a 6NET [NET04b], Estados Unidos com a Internet2 [INT04], além da iniciativa da Rede Nacional de Pesquisa no Brasil [RNP04].

1. Recursos para IPv6 atuais

Tendo ciência do tamanho da mudança que o IPv6 inclui sobre o IPv4, já que se trata do protocolo/tecnologia básico para a operação da Internet, é esperado que à médio/longo prazo o uso de IPv6 seja fortemente atrelado ao funcionamento em harmonia com o IPv4. Isto tem se revelado como uma das principais preocupações dos grupos de trabalho do IETF, como o V6 Operations [6OP04], e também nas discussões em redes experimentais.

Em resumo, três fluentes têm se apresentado no que tange a esta forma de convivência. Cada uma delas possui características básicas que a diferenciam completamente quanto à infra-estrutura e a forma de acesso IPv4/IPv6. Cada uma será apresentada a seguir:

2.1 Dual Stack

A primeira delas é chamada de “Dual Stack” [GIL00], que se trata de uma integração da pilha IPv6 em sistemas operacionais e equipamentos de rede com a pilha IPv4. Neste caso, o mesmo nodo passa a ter a possibilidade de acesso tanto via IPv4 como IPv6. Inclusive, neste mesmo nodo, algumas aplicações podem usar a pilha que desejarem. Em contrapartida, é necessário que a infra-estrutura nesse caso também seja dual stack, ou seja, que forneça tanto conectividade IPv4, quanto IPv6. Grande parte dos sistemas operacionais de hoje oferecem suporte a este recurso, a citar: Windows XP, FreeBSD, Linux, entre outros. Em equipamentos de rede como roteadores e switch/routers, a realidade também é muito semelhante, visto que já existem diversas implementações de protocolos de roteamento IPv6, a citar: RIPng [MAL97], OSPFv3 [COL99] e MP-BGP [BAT00] além dos tradicionais protocolos de roteamento IPv4.

2.2 Tunelamento

O recurso de tunelamento oferece a possibilidade de acesso IPv6 sobre redes IPv4, ou seja, mesmo não tendo infra-estrutura com suporte ao IPv6, os pacotes são encapsulados sobre IPv4. Trata-se de uma forma amplamente empregada no passado, que visa superar diversos problemas com equipamentos de rede antigos, sem suporte ao IPv6.

A seguir são apresentados alguns recursos/soluções de tunelamento:

2.2.1 IPv6-over-IPv4

Trata-se da forma mais elementar de tunelamento que encapsula os pacotes IPv6 em pacotes IPv4. Definido em [GIL00], trata-se de uma forma de tunelamento ponto a ponto, onde o nodo que encapsula deve explicitamente ser configurado com o IP do nodo destino do túnel. Caso entre o nodo inicial ou final do túnel haja uma firewall, deve ser configurado para permitir pacotes IPv4 entre os nodos cujos códigos de protocolos sejam 41 (IPv6) e 58 (ICMPv6). Para os nodos intermediários ao túnel, o tráfego IPv6 será visto simplesmente como tráfego IPv4, obviamente.

2.2.2 Tunnel Broker

Como forma mais automática ao modelo anterior, surge a abordagem de Tunnel Broker, definida em [DUR01]. Tal abordagem prevê uma forma automática onde um host dual-stack situado em uma rede IPv4-only tem a necessidade de conectividade IPv6. Neste caso, o host dual stack acessa um servidor que fornece algum nível de autenticação, retornando um script que estabeleça uma conexão entre o host e o Tunnel Broker Tunnel Server, fornecendo assim uma forma de um tunelamento IPv6 automática. Tal prática pode envolver hosts ou gateways de redes dentro de uma instituição, já que esta é de certa forma automática e depende apenas de planejamento.

Nada impede a disponibilização de Tunnel Broker Servers remotos, mas no ponto de vista de roteamento esta opção pode ser inviável pelo delay que fornece em seu acesso. Neste ponto existe uma iniciativa chamada de Freenet6 [FRE04], que fornece uma espécie de Tunnel Broker Server público no Canadá, com a facilidade de estabelecimento de túneis a partir de qualquer rede, em qualquer país do Mundo. Tal recurso foi testado e de fato pode ser visto como a forma mais eficaz de acesso experimental a rede IPv6 mundial.

2.2.3 6to4

Definida em [CAR01], esta forma de tunelamento é considerada uma das mais triviais, visto que sua configuração é mínima e o bloco de endereçamento é automático, baseado no endereço IPv4 dos nodos envolvidos. O prefixo base, alocado pelo IANA é o 2002::/16, que é complementado pelo endereço IPv4 do nodo, na forma de 2002:IPVADDRESS::/48, resultando em um bloco /48 da mesma forma que os prefixos de produção 2001::/16. Neste caso, na comunicação entre nodos 6to4, o destino é roteado de acordo com o endereço IPv4 envolvido no bloco 6to4 a ser alcançado. Dessa forma, nenhum protocolo de roteamento IPv6 necessita ser utilizado, visto que o pacote será roteado via IPv4.

Na comunicação com hosts que não sejam 6to4, surge a necessidade então de uma entidade “relay”, que possa fazer a ligação entre as redes 6to4 e as redes IPv6 nativas, tendo uma interface em cada uma destas redes. Neste caso a entidade relay deve fazer uso de protocolos de roteamento IPv6 para exportar o prefixo da rede 6to4 para a rede de produção (2001::/16) e vice-versa. A localização destes relays públicos é feita através do recurso de anycast, conforme [HUI01].

2.2.4 DSTM

O recurso de Dual Stack Transition Mechanism (DSTM) [BUN02] tem como foco redes IPv6-only, ou seja, que possuem infra-estrutura com suporte apenas a IPv6. Neste

caso, a solução atua como forma de manter o acesso a partir de hosts IPv6 a aplicações sobre IPv4 ou ainda pela necessidade de acesso a algum host/serviço IPv4 pela Internet.

Na comunicação entre nodos IPv6, a solução nada precisa fazer já que a infraestrutura é IPv6 nativa. No caso de acesso IPv4, duas entidades DSTM entram em ação:

- DSTM Server – Faz o controle de endereços IPv4, recebendo e gerenciando os pedidos de conexões IPv4 ao DSTM client, que é o host que IPv6 que deseja fazer o acesso com destino a IPv4.
- DSTM Gateway – Fornece conectividade IPv4, tendo pelo menos uma de suas interfaces sobre IPv4, além de uma interface na rede IPv6.

Em termos de implementação, atualmente ambas as entidades podem ser utilizadas no mesmo host.

Assim sendo, o DSTM client que deseja fazer a conexão sobre IPv4 envia seu pedido ao DSTM Server, que processa e retorna as seguintes informações ao requisitante:

- O IP da família v4 alocado para a conexão;
- O tempo em que o endereço será alocado;
- Os IPs que serão usados para o túnel até o DSTM gateway;

Com posse destas informações, o DSTM client tem plenas condições de obter conectividade IPv4, em posse de um endereço v4 global.

2.3 Tradução

O recurso de tradução envolve cenários onde hosts IPv6-only consigam se comunicar com hosts IPv4-only e vice-versa. Serão apresentadas duas formas conhecidas envolvendo este recurso:

2.3.1 NAT-PT

Da mesma forma que o recurso de NAT é utilizado hoje no IPv4, o Network Address Translation with Protocol Translation (NAT-PT)[TSI00] oferece uma forma de tradução de um pacote IPv4 em um pacote IPv6 semanticamente equivalente. Uma analogia a esta forma pode ser feita lembrando novamente do NAT IPv4 que converte um pacote IPv4 com endereço privado para um pacote equivalente com endereço global.

Através deste recurso surge a possibilidade de manter uma rede com serviços em IPv4, tendo conectividade através da tradução dos pacotes a qualquer rede da Internet sobre IPv6.

2.3.2 ALG

O recurso de Application Layer Gateway (ALG) oferece recursos muito semelhantes ao NAT-PT, mas possui uma diferença muito sutil. Ao invés de fazer a conversão entre pacotes para torná-los semanticamente equivalentes, o ALG recebe as conexões de seus clientes que são feitas diretamente a ele e em caso necessário, abre uma nova conexão ao destino. Uma analogia a esta solução pode ser feita a proxies web, que usam essa mesma filosofia.

Um exemplo de aplicação é que um servidor ALG receba conexões IPv4 de uma rede IPv4-only e ao observar que a conexão é destinada a uma rede IPv6, abrir uma nova conexão a partir de sua interface IPv6, fazendo o mesmo processo no retorno deste acesso.

3 Conclusões

Tratando-se dos principais recursos disponíveis para uso de IPv6 em conjunto com IPv4, cada um destes apresenta pontos fortes, deficiências e contextos que possam ser melhor explorados. A primeira abordagem, envolvendo o uso de “dual stack” continua sendo a alternativa mais aconselhável nos casos onde há possibilidade de uso de infra-estrutura IPv6, ou seja, que haja disponibilidade de equipamentos de redes e hosts com suporte a IPv6. Dessa forma poderão ser agregados serviços sobre IPv6 sem prejudicar o funcionamento de serviços IPv4. Ao mesmo tempo, surge a necessidade de proteção a partir de redes IPv6, visto que surge a possibilidade de hosts dual stack serem atacados pela pilha IPv6. Essa característica deve ser muito bem analisada, pois pode comprometer o plano de segurança sobre IPv4.

Na existência de equipamentos de rede sem suporte ao IPv6 e por consequência, sem infra-estrutura com suporte nativo ao IPv6, a opção de túneis pode ser muito bem vista. É muito comum encontrarmos equipamentos de redes antigos ou sem possibilidade de upgrade, forçando então ao uso apenas em IPv4. Nestes casos, a opção de túneis pode mapear redes inteiras ou pequenos segmentos, fornecendo acesso IPv6 sobre a infra-estrutura IPv4. Quanto as opções de túneis, dependendo da realidade de cada rede, uma opção diferente de tunelamento pode ser melhor empregada, necessitando de um estudo específico. Como limitações desta opção, pode-se citar o risco do roteamento não percorrer os melhores caminhos, pois em um túnel que percorre três hops de distância, para o protocolo de roteamento que usa este túnel a distância é de apenas um hop. Outro fator importante nessa abordagem é o overhead introduzido pelo processamento no encapsulamento. A opção de DSTM, que foca em uma rede IPv6 nativa como base, mas para a atual realidade ainda é uma opção distante, já que é baseada no uso em massa de IPv6, tendo o intuito de garantir operabilidade com entidades IPv4.

A opção de tradução pode ser vista em contextos onde existe de alguma forma conectividade IPv6 na infra-estrutura, mas que alguns segmentos ou alguns serviços legados apenas permitem o uso sobre IPv4. Tal realidade pode ser aplicada a estações de trabalho antigas ou mesmo sistemas legados. Como limitações podemos incluir o overhead que acaba sendo semelhante ao processamento no encapsulamento nos túneis. No caso da solução com NAT-PT, todos os problemas enfrentados com o NAT também ser transportados para a realidade em IPv6, merecendo um estudo mais aprofundado.

4 Trabalhos Futuros

Como trabalhos futuros espera-se analisar mais profundamente cada abordagem, além de elaborar testes de suas implementações em uma rede experimental com sistemas BSD e Linux, além de roteadores Cisco, com o intuito de avaliar a operação de cada das soluções. A partir desse patamar de testes, será possível obter resultados mais concretos sobre cada solução, além de possibilitar que as melhores soluções sejam implantadas de

forma segura e gradativa em instituições da Rede Tche e na Rede Metropolitana de Porto Alegre (Metropoa).

Referências Bibliográficas

- [6OP04] **V6 Operations Working Group on IETF** - <http://www.ietf.org/html.charters/v6ops-charter.html> - Acesso em Junho de 2004
- [BAT00] T. Bates, Y. Rekhter, R. Chandra, D. Katz - **Multiprotocol Extensions for BGP-4** – RFC 2858 – IETF – Junho de 2000
- [BON04] **Site do Projeto 6Bone** – www.6bone.net - Acesso em Janeiro de 2004
- [BUN02] Bund, Toutain, Medina et al - **Dual Stack Transition Mechanism (DSTM)**
Internet Draft ; draft-etf-ngtrans-dstm-08.txt July 2002
- [CAR01] B. Carpenter - **Connection of IPv6 Domains via IPv4 Clouds** - RFC 3056 - IETF - Fevereiro de 2001
- [COL99] R. Coltun, D. Ferguson, J. Moy - **OSPF for IPv6** – RFC 2740 – IETF – Dezembro de 1999
- [COM01] Douglas Comer – **Redes de Computadores e Internet** – Pág.265-275 – Porto Alegre – Bookman – 2001.
- [DEE95] S. Deering, R. Hinden - **Internet Protocol, Version 6 (IPv6) Specification** - RFC 1883 – IETF – Dezembro de 1995
- [DEE98] S. Deering, R. Hinden - **Internet Protocol, Version 6 (IPv6) Specification** – RFC 2460 – IETF – Dezembro de 1998
- [DUR01] A. Durand, P. Fasano, D. Lento - **IPv6 Tunnel Broker** - RFC 3053 - IETF - Janeiro de 2001
- [FRE04] **Freenet6 Project** – www.freenet6.org – On Line – Acesso em Junho de 2004
- [GIL04] R. Gilligan, E. Nordmark - **Transition Mechanisms for IPv6 Hosts and Routers** - RFC 2893 - IETF - Agosto de 2000
- [HUI01] C. Huitema - **An Anycast Prefix for 6to4 Relay Routers** - RFC 3068 - IETF - Junho de 2001
- [INT04] **Projeto Internet2** – www.internet2.edu - Acesso em Janeiro de 2004
- [ISI81] Information Sciences Institute/ University of Southern California - **Internet Protocol** - RFC 791 – IETF - Setembro de 1981
- [MAL97] G. Malkin, R. Minnear - **RIPng for IPv6** – RFC 2080 – IETF – Janeiro de 1997
- [NET04b] **Projeto 6NET da Rede Géant** – www.6net.org - Acesso em Janeiro de 2004
- [RNP04] **Projeto IPV6 da RNP** – www.rnp.br/ipv6 - Acesso em Janeiro de 2004
- [TSI00] G. Tsirtsis, P. Srisuresh - **Network Address Translation - Protocol Translation (NAT-PT)** - RFC 2766 - IETF - Fevereiro de 2000

Análise comparativa entre NIS e LDAP

Augusto Peixoto Bueno, Alexandre Carissimi

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – CEP: 91501-970 – Porto Alegre – RS – Brasil

apbueno@inf.ufrgs.br, asc@inf.ufrgs.br

Resumo. O Lightweight Directory Access Protocol (LDAP) vem sendo usado em um número cada vez maior de aplicações distribuídas porém pouco se sabe sobre seu desempenho em ambientes de produção. Este trabalho apresenta uma análise do LDAP e suas vantagens em substituição ao Network Information Services (NIS) para a autenticação de usuários do serviço de e-mail (POP/SMTP AUTH) na rede do Instituto de Informática da UFRGS.

1. Introdução

Um dos desafios no desenvolvimento de sistemas distribuídos é localizar e identificar recursos de forma não ambígua. Entende-se por recursos, computadores, serviços e até mesmo usuários vinculados a uma certa aplicação. Para que isso seja possível é necessário armazenar e, posteriormente, consultar uma série de informações sobre esses recursos. Essas informações devem, portanto, serem organizadas e armazenadas para permitir de maneira rápida e confiável a sua recuperação. Nesse contexto dois serviços são largamente utilizados: serviço de nomes e serviço de diretório.

O *Lightweight Directory Access Protocol* (LDAP) é um protocolo de acesso a diretórios que define várias operações para recuperação, atualização e remoção de informações em um diretório. Apesar do crescimento no seu uso, pouco se encontra sobre desempenho do LDAP em ambientes de produção. Alguns estudos foram feitos, mas, em sua maioria, são com o objetivo de marketing de um produto[5] ou foram feitos em ambientes controlados [1,6,8], poucos são situações reais de uso [2,4]. Este trabalho possui como objetivo avaliar o uso e o desempenho do LDAP no ambiente de rede do Instituto de Informática da UFRGS, quando comparado ao NIS (*Network Information Service*).

Este trabalho está organizado da seguinte forma. Na seção 2 são abordados alguns conceitos básicos sobre serviços de nomes e de diretórios. Na seção 3 é feito uma breve descrição do que é o LDAP e na seção 4 sobre o NIS. Na seção 5 é analisado comparativa entre o NIS e o LDAP mostrando os resultados obtidos neste estudo. Finalmente, na seção 6, discute-se as principais conclusões obtidas neste trabalho e os trabalhos futuros.

2. Serviços de Nomes e Diretórios

Muitas vezes um serviço de diretório é confundido com um serviço de nomes, mas esses são, essencialmente, diferentes. Um serviço de nomes é responsável pela tradução de um nome em seu atributo. Já um serviço de diretório é muito mais flexível, pois permite a obtenção de quaisquer informações sobre um recurso a partir de um atributo (característica) ou de um conjunto de atributos. O DNS (*Domain Name System*), o NIS (*Network Information System*) são, respectivamente, exemplos de serviços de nomes e de diretórios comumente empregados em ambientes de rede de muitas organizações.

Um conceito importante a ambos é o de espaço de nomes. Um espaço de nomes é a coleção de todos os nomes válidos em um determinado escopo e dividem-se em duas categorias: hierárquicos e *flat* (planos). Um espaço de nomes hierárquico é, por definição, infinito. Isso ocorre através da criação de diversos níveis, como por exemplo, em uma estrutura de arquivos com diretórios e subdiretórios. Já, um espaço *flat* é limitado pelo tamanho máximo de um nome. Entretanto, basta não limitar esse tamanho para tornar o espaço de nomes *flat* também infinito. Um espaço hierárquico facilita o armazenamento e a organização de informações o que possibilita, em relação ao *flat*, delegar a administração de diferentes contextos de nomes a diversas pessoas ou organizações.

É comum confundir um serviço de diretório com um banco de dados, já que ambos guardam informações. De fato, um diretório é um tipo de banco de dados, mas com uma finalidade específica, focada para sua aplicação. Assim, para cada uso que um diretório pode ter, é necessário criar uma estrutura única para esse uso. Isso difere dos bancos de dados convencionais, onde a estrutura é padronizada, cabendo a quem desenvolve a aplicação relacionar adequadamente as entidades. Além disso, um banco de dados é otimizado para tratar tanto leituras quanto escritas, enquanto um diretório é voltado para leitura de informações, considerando que atualizações são menos freqüentes. Outra diferença entre diretórios e bancos de dados é que estes últimos normalmente suportam o conceito de transações, ao passo que os diretórios não o fazem obrigatoriamente.

3. O *Lightweight Directory Access Protocol* (LDAP)

O LDAP [7] é um protocolo assíncrono baseado em troca de mensagens. Apesar de ser referenciado como serviço de diretório, o LDAP é, de fato, um protocolo de comunicação da camada de aplicação. O LDAP segue o modelo cliente-servidor e, de uma maneira geral, a comunicação entre clientes e servidores LDAP ocorre da seguinte forma. Inicialmente, o cliente estabelece uma sessão com o servidor. Esse processo é chamado de *binding* (mapeamento). Nesse momento, o cliente pode ser ou não autenticado pelo servidor, assim como negociar um método de criptografia para tornar a comunicação segura. Uma vez autenticado o cliente executa operações de leitura e escrita sobre os dados do diretório, o que possibilita a recuperação e atualização das informações. Ao concluir suas operações, o cliente encerra a sessão com o servidor através da operação *unbinding*. O LDAP é implementado sobre o protocolo de transporte TCP.

Modelos do LDAP: Os serviços fornecidos por diretórios LDAP são organizados em modelos. A RFC2251 define dois: modelo de protocolo e modelo de dados. Entretanto, é comum se empregar a definição de quatro modelos proposta por Howes [3], a saber:

1. Modelo de informação: fornece a estrutura das informações armazenadas em um diretório LDAP. A unidade de informação é a entrada (objeto). A estrutura que define quais objetos existem para que as entradas possam ser instanciadas e os atributos de cada objeto é chamada *Schema*. É no *Schema* que também está associada a sintaxe de cada atributo, se é multi ou monovalorado, se o valor é obrigatório ou não.
2. Modelo de nomes: define a organização das entradas no diretório de forma hierárquica em uma estrutura denominada DIT (*Directory Information Tree*). As entradas são dispostas de acordo com seu *distinguished name* (DN) que serve como um identificador único. Um DN é uma seqüência de *relative distinguished names* (RDN) que juntos apontam para uma única entrada. Cada RDN é um ponto único na DIT.

3. Modelo funcional: define três categorias de operações para a manipulação de entradas do diretório independentemente de linguagens de programação: pesquisa, atualização e autenticação. A partir do LDAPv3 são incluídas operações estendidas.
4. Modelo de segurança: provê mecanismos que permitem aos usuários provar sua identidade (autenticação) e ao servidor controlar o acesso dos usuários autenticados (autorização). Inclui acesso anônimo, uso de par usuário e senha (senha "aberta" na rede) e a partir do LDAPv2 a utilização de Kerberos. O LDAPv3 introduz o uso de SASL além da definição de uma operação estendida do LDAP relativa à segurança, a *Extension for Transport Layer Security for LDAPv3*, baseada em SSL.

Replicação: A replicação em serviços de diretórios é importante por aumentar sua capacidade de expansão bem como seu desempenho e tolerância a falhas. Na especificação original do LDAP a replicação não era prevista., porém, atualmente, existe um grupo de trabalho do IETF padronizando o sistema de replicação do LDAP.

4. Network Information Service (NIS)

O NIS é um repositório de arquivos que normalmente é usado como ferramenta de administração de ambientes de rede cujo principal objetivo é manter a consistência das informações administrativas e gerenciais de uma rede. O NIS mantém suas informações em uma estrutura denominada *map*, organizada na forma de um par *{chave, valor}*. Ao ser utilizado no gerenciamento de uma rede, os mapas contêm informações sobre diversos aspectos dessa, tais como: usuários, senhas, grupos, usuários, entre outros. Os mapas são derivados de arquivos de configuração. Por exemplo, a informação contida em */etc/hosts* é usada para criar um mapa que possui o nome do *host* como chave e o endereço IP como valor. Os pares chave-valor, também conhecidos como registros, extraídos de */etc/hosts* criam o mapa *hostsbyname*. O NIS usa um espaço de nomes *flat*.

O NIS segue um modelo cliente-servidor onde os servidores são responsáveis pelo armazenamento dos mapas e por atender às requisições dos clientes. Existem dois tipos de servidores: mestre e escravo. O servidor mestre possui autoridade sobre os mapas e o(s) servidor(es) escravo(s) mantém réplicas. Além de aumentarem a disponibilidade, os servidores escravos ajudam no aumento do desempenho, pois são capazes de responder às mesmas requisições que o servidor mestre. Um NIS cliente faz requisições de consulta a mapas para os servidores e espera as respostas. Estas requisições são feitas via RPC (*Remote Procedure Call*) e não fazem distinção entre servidor mestre e servidor escravo.

5. Análise Comparativa NIS e LDAP

O objetivo é comparar o NIS e o LDAP para autenticação de usuários do serviço de e-mail (POP/SMTP AUTH) usando como métricas carga na rede e tempo médio de resposta. Essas métricas são usadas por serem dois parâmetros importantes de desempenho em qualquer protocolo de rede. Considera-se ainda o espaço ocupado em disco.

A rede do Instituto de Informática historicamente autentica seus usuários via NIS. Para empregar o LDAP converteu-se a base NIS existente para um diretório LDAP usando ferramentas de migração disponibilizadas no OpenLDAP. Nessa conversão foi utilizado o *schema* padrão *PosixAccount* [7] que oferece suporte as informações NIS. A base de usuários possui cerca de 1700 entradas (objetos), cada uma composta por 400 bytes.

Metodologia: nesse estudo considera-se apenas a carga gerada pelo servidor de e-mail quando os usuários do Instituto de Informática acessam seus e-mails via POP ou quando enviam mensagens usando SMTP AUTH (SMTP Autenticado). Outras autenticações (*login*, impressão, etc) continuam sendo feitas no servidor NIS original (servidor a parte). Inicialmente, com o servidor de e-mail do Instituto de Informática autenticando os usuários via NIS, foi feita a captura dos pacotes de dados e controle relacionados com o NIS. Após isso, o servidor de e-mail teve o método de autenticação alterado para usar LDAP e os dados foram novamente coletados. Nesse caso, duas coletas distintas foram feitas: com e sem a utilização do índice para os atributos *uid*, *uidNumber* e *gidNumber* (a indexação é descrita mais adiante). Os dados foram coletados em horários considerados de pico de utilização, gerando, em média, quatro requisições de autenticação por segundo, tanto para o NIS quanto para o LDAP. Os resultados obtidos nessa experiência são validados estatisticamente com um intervalo de confiança de 95%.

Plataforma experimental: O servidor LDAP/NIS executa em um DELL PowerEdge 600sc, Pentium IV 2.4GHz, 1 GB RAM, 40 GB de disco rígido (SCSI) e barramento de dados de 533 MHz. O cliente é o servidor de e-mail do Instituto de Informática: DELL PowerEdge6400, biprocesso Xeon 700 MHz, 512 MB RAM, 27 GB de disco rígido (SCSI) em RAID 5 e um barramento de dados de 133MHz. O servidor LDAP utilizado foi o OpenLDAP 2.1.23 sobre uma máquina FreeBSD 4.9-stable. A base de dados usada foi a BDB (Berkeley DB v4.1). A mesma máquina executa o servidor NIS v1.31.2.1. O servidor de e-mail usa RedHat Linux 8 (kernel 2.4.23) e o daemon de mail é o *sendmail* 8.12.8. Todas as máquinas utilizadas estão ligadas a um switch de 100 Mbit/s.

Funcionamento básico do NIS e do LDAP para autenticação de usuários: O NIS utiliza RPC para troca de mensagens. Inicialmente uma conexão TCP é feita para requisitar ao serviço *portmap* a informação da porta a ser usada nas requisições RPC. Em seguida, o cliente faz uma ou mais requisições do tipo *match*, usando UDP, para recuperar as informações necessárias a autenticação, entre elas o mapa *groupbyname*. Já, no LDAP, toda comunicação é feita em uma conexão. O cliente inicia uma sessão executando a operação *bind*. Em seguida, o cliente requisita ao servidor as informações necessárias para autenticação do usuário. Em resposta, o servidor envia apenas os atributos requisitados pelo cliente e não o conteúdo inteiro da entrada associada a solicitação (como no NIS). Pode ser necessário, para atender todo o processo de autenticação, que o cliente faça mais de uma requisição mudando apenas o atributo desejado. O cliente encerra a sessão executando a primitiva *unbind*. Esses procedimentos são ilustrados na figura 1. No caso específico do LDAP existe a possibilidade de utilizar índices para otimizar o procedimento de busca de informações no diretório (análogo ao uso de chaves primárias e secundárias em banco de dados). A definição se um atributo é indexado ou não é feita no momento da definição da estrutura de diretório.

Análise de resultados: A tabela I fornece, com base nos dados coletados, a carga média na rede (em Mbits/s) gerada pelo NIS e pelo LDAP. No caso do LDAP foram considerados os acessos ao diretório com e sem a utilização dos índices para os atributos *uid*, *uidNumber* e *gidNumber*. Esses atributos foram escolhidos porque seus valores são os consultados no procedimento de autenticação de usuários.

Ao analisar a tabela I, percebe-se que o protocolo LDAP, tanto não-indexado como indexado, é mais eficiente na utilização da rede, apresentando uma carga média na rede

cerca de 75% menor que a do NIS. Isso se deve ao fato do tamanho médio dos pacotes LDAP ser cerca 71% menor que os pacotes NIS. Essa diferença, em parte, é devido ao NIS executar a cada autenticação de usuário uma *procedure* chamada *ALL*, a qual retorna todo o mapa *groupbyname* (dados que representam todos os grupos de usuário existentes na rede). Já o servidor LDAP retorna apenas os atributos requisitados pelo cliente.

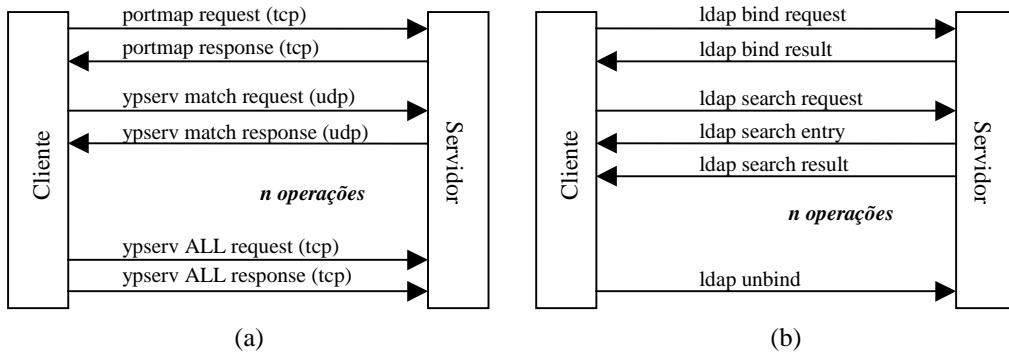


Figura 1 – Comunicação entre cliente e servidor: (a) NIS e (b) LDAP

Tabela I – LDAP versus NIS: carga média na rede em Mbits/s

LDAP (indexado)	LDAP (não indexado)	NIS
0.069	0.076	0.32

A figura 2 analisa o mesmo conjunto de dados em termos de tempo médio de resposta. Novamente para o LDAP são considerados dois casos: com (LDAP indexado) e sem (LDAP não indexado) a utilização de índices para os atributos *uid*, *uidNumber* e *gidNumber*. Percebe-se que o LDAP indexado teve um desempenho em torno de dez vezes maior que o NIS. Porém, o LDAP não-indexado apresenta um desempenho aproximadamente 26 vezes menor que o caso indexado e 2,6 vezes menor que o NIS. Fica evidente o interesse em identificar quais atributos serão os mais frequentemente acessados em um diretório LDAP para torná-los atributos do tipo indexado.

Em relação ao espaço em disco, a base LDAP é composta por um conjunto de arquivos que representam a base de dados propriamente dita (entradas) e os arquivos de índices dos atributos indexados. Neste estudo, o tamanho total ocupado pelo LDAP foi de cerca de 12 Mbytes contra 7.5 Mbytes ocupados pelas informações mantidas pelo NIS.

Considerações gerais: O LDAP é um padrão, o que facilita sobremaneira seu emprego em ambientes heterogêneos. Por utilizar um espaço de nomes hierárquico, o LDAP possui uma escalonabilidade maior que o NIS (que emprega um espaço de nomes *flat*), além de permitir um particionamento da base de dados (subárvores) em diversas máquinas o que contribui para uma descentralização de serviços e consequente平衡amento de carga entre servidores. Aproveitando características similares de diretórios a banco de dados, a centralização de informações permite um melhor controle sobre usuários e recursos da rede. Um usuário cadastrado em uma base LDAP pode herdar, por ser um objeto, uma série de configurações básicas que, se bem aproveitadas, auxiliam em aspectos de segurança da rede, com por exemplo, permissões, controle de acesso a recursos, etc. Finalmente, o LDAP suporta facilmente criptografia através do emprego de TSL/SSL, ao passo que o NIS trafega todas suas informações em texto claro

na rede. É verdade que o NIS+, agrupa proteção a comunicação entre clientes e servidores porém, além de não ser facilmente configurável, apresenta alguns problemas de compatibilidade em ambientes UNIXes de diferentes fabricantes. A principal desvantagem do LDAP é a necessidade de planejar a estrutura do diretório (DIT), suas entradas e índices. Uma estrutura muito específica pode dificultar modificações e gerar, com diversos níveis, um custo de processamento mais elevado para atender uma requisição.

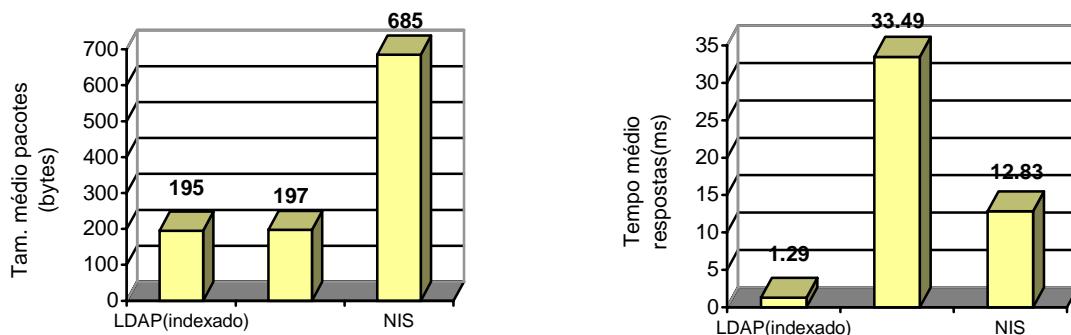


Figura 2 – LDAP versus NIS: tamanho médio dos pacotes

6. Conclusões

Este trabalho faz uma análise do LDAP como opção ao NIS na autenticação de usuários do serviço de e-mail. O LDAP apresentou melhor tempo de resposta, menor tráfego na rede porém, um consumo de espaço em disco pouco maior para armazenar as informações de usuários da rede. Foi mostrado que a indexação do diretório deve ser planejada com cuidado, pois o desempenho do serviço de diretório LDAP é afetado pela escolha dos índices. Como trabalhos futuros pretende-se investigar a carga sobre o servidor e o impacto do custo de processamento no uso de criptografia .

7. Referências Bibliográficas

- [1] Choi, J.; et alli. Performance Evaluation of Two OpenLDAP Directory Server Back-end Designs. <ftp://ftp.openldap.org/incoming/perfeval.zip> (Acesso em: 04/01/04)
- [2] Dixon, W.; et alli. An Analysis of LDAP Performance Characteristics. <http://www.crd.ge.com/cooltechnologies/pdf/2002grc154.pdf> (Acesso em: 29/12/03)
- [3] HOWES, T.; SMITH, M.; GOOD, G. Understanding and Deploying LDAP Directory Services. 1st ed. Macmillan Technical Publishing, 1999
- [4] Klasen, N. Directory Services for Linux in comparison with Novell NDS and Microsoft AD. <http://www.daasi.de/staff/norbert/thesis/html/thesis.html>. (Acesso em: 03/01/04).
- [5] Mindcraft Inc. LDAP Directory Server Comparison: Netscape and Novell. <http://www.mindcraft.com/perfreports/ldap/netscape/dirserver10.html> (Acesso: 4/01/04).
- [6] Snyder, J. Sizing up LDAP servers. Network World, 2000. Disponível em: <http://www.nwfusion.com/reviews/2000/0515rev2.html>. (Acesso em: 28/12/03).
- [7] WAHL, M.; HOWES, T.; KILLE, S. Lightweight Directory Access Protocol (v3): RFC 2251. [S.L]: Network Working Group, 1997.
- [8] WANG, X.; Schulzrinne, H; Kndlur, D.; Verma, D. Measurement and Analysis of LDAP Performance. International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'2000), Santa Clara, CA, 2000, p. 156-165.

Técnicas de combate a *spam* em servidores Linux que utilizam o *Mail Transfer Agent Postfix*.

Fábio de Oliveira Dias¹, Cristina Moreira Nunes¹

¹Centro Universitário La Salle – Av. Victor Barreto, 2288 – Canoas – RS – Brasil

fabio@cefetrs.edu.br, nunes@lasalle.tche.br

Resumo. O *spam* tornou-se um grande problema para a comunidade internet, em virtude do aumento da demanda de recursos computacionais exigidos para suportar o volume, cada vez maior, de mensagens indesejadas. Este artigo aborda técnicas de combate ao *spam* em servidores Linux que fazem uso do *Mail Transfer Agent Postfix*. Procura, também, demonstrar que a combinação destas diversas técnicas (consulta em servidores de DNS reverso e em Realtime Blackhole Lists, utilização de listas de acesso locais e bloqueio de anexos nocivos) pode ser capaz de minimizar os efeitos decorrentes desse problema. Espera-se que, com a aplicação da metodologia desenvolvida, mais de 98% do total de mensagens indesejadas possa ser corretamente descartado.

1. Introdução

Segundo a *United Nations Statistics Division* (Divisão de Estatísticas da Organização das Nações Unidas) [United Nations Statistics Division 2004], a internet mundial já possui mais de 600 milhões de usuários ativos, sendo que o Brasil responde por mais de quatorze milhões destes. Um dos recursos mais utilizados e que são providos por esta tecnologia é o sistema de correio eletrônico. Bilhões de mensagens trafegam diariamente pela internet levando dados de uma parte a outra do planeta em questão de segundos.

Atualmente, conforme o *Spamhaus Block List* [Spamhaus 2004], entidade de pesquisa que efetua estudos sobre o assunto, a maior parte do tráfego de mensagens de correio eletrônico pode ser classificada como UCE (*Unsolicited Commercial E-mail*, Mensagem Comercial Não-solicitada) ou, em sua designação popular, *spam*. Segundo a mesma entidade, o Brasil já é o quarto país na produção mundial de *spams*.

A prática do *spam* tem gerado preocupação e provocado muitas reações por parte dos administradores de rede e dos provedores de acesso. O *spam* gera excessiva ocupação da largura de banda, provocando congestionamentos na rede e consumo de recursos computacionais valiosos. Os usuários perdem tempo e dinheiro para abrir, ler e apagar as mensagens inúteis, podendo, ainda, ser atacados por vírus, *scams* e outras pragas enviadas em massa. Assim, a única parte que costuma se beneficiar com esse tipo de prática é o *spammer*, pois o seu meio de divulgação tem custo baixo e é extremamente eficaz, considerando as taxas de retorno de métodos tradicionais como mala-direta e *telemarketing*.

Além disso, segundo a 15^a Pesquisa Anual de Informática da Escola de Administração de Empresas de São Paulo da Fundação Getúlio Vargas [Meirelles 2004], o número de servidores que utilizam o sistema operacional Linux no Brasil vem crescendo consideravelmente a cada ano, em virtude de sua confiabilidade, segurança e melhor aproveitamento de recursos. Nestes servidores, a utilização do MTA (*Mail Transfer Agent*, Agente de Transferência de Mensagens) Postfix também está em expansão, em virtude de

sua robustez e pelo histórico de vulnerabilidades de seu concorrente direto em sistemas Linux, o Sendmail.

Neste artigo, serão abordadas algumas técnicas de combate ao *spam* em sistemas Linux que utilizam o agente de transferência de mensagens Postfix. A motivação principal deste artigo é a de abordar um tema de relevância, partindo do estudo das técnicas de envio de *spam* até verificar os procedimentos que devem ser adotados para minimizar este problema. Como consequência, serão buscadas formas de reduzir a ocorrência de *spam* em servidores Linux que fazem uso do MTA Postfix a níveis considerados míimos.

Na próxima seção será exposto o referencial teórico que serve de embasamento a este artigo. A seção 3 abordará a descrição metodológica desenvolvida. Já a última seção conterá as conclusões do estudo efetuado.

2. Referencial Teórico

Conforme a RFC 2505 [Lindberg 1999] e Hoepers [Hoepers 2003], os *spammers* costumam utilizar, basicamente, três técnicas para enviar mensagens não-solicitadas em massa: entrega direta (*direct delivery*), *open relay* e *open proxy*. Além disso, no caso de pragas digitais como vírus, utilizam os computadores infectados como nova fonte de envio de mensagens indesejadas para terceiros.

A entrega direta consiste na ativação (mesmo que apenas momentânea) de um servidor SMTP (*Simple Mail Transfer Protocol*, Protocolo Simples de Transferência de Mensagens) em uma determinada máquina, que servirá para enviar milhões de *spams* para qualquer parte do mundo em questão de minutos. A desvantagem desta técnica é que o endereço IP (*Internet Protocol*, Protocolo Internet) do *spammer* pode ser facilmente identificado e bloqueado.

Outras formas comuns de envio de *spam* são através dos mecanismos de *open relay* e *open proxy*. Servidores que possuem *relay* ou *proxy* abertos permitem o encaminhamento de mensagens a terceiros através de seus serviços. Assim, os *e-mails* que chegam aos destinatários contêm o endereço IP do servidor vulnerável. A desvantagem desta técnica, além da grande possibilidade de registro do endereço IP invasor nos *logs* do sistema, consiste no maior grau de conhecimento exigido para a sua implementação. Andreolly [Andreolly et al 2001] considera que estes fatos são considerados incidentes de segurança e devem ser reportados aos responsáveis pelos domínios envolvidos.

A RFC 821 [Postal 1982] define o protocolo SMTP. Na época de sua concepção, não foram dimensionados os problemas que alguns aspectos referentes à segurança e não contemplados poderiam ocasionar. Em 2001, a RFC 2821 [Klensin 2001] abordou novamente o protocolo SMTP, tornando obsoleta a RFC 821. Entretanto, nenhuma funcionalidade relacionada à segurança do protocolo foi alterada ou adicionada.

A fim de abrandar as deficiências explícitas da RFC 821, foi lançada a RFC 2505. Esta RFC provê algumas orientações que visam prevenir a aceitação indevida de *spam* pelos MTAs. Conforme esta RFC, quando ocorre uma conexão SMTP, três situações distintas podem ocorrer, todas relacionadas ao resultado da consulta do endereço IP de origem junto a um serviço de DNS (*Domain Name Server*, Servidor de Nomes de Domínio) reverso. São elas:

- este endereço IP não está associado a um nome de domínio FQDN (*Fully Qualified Domain Name*, Nome de Domínio Completamente Qualificado);

- o nome de domínio contido no *header* (cabeçalho) da mensagem foi forjado e não corresponde ao real domínio associado a este endereço IP;
- o endereço IP corresponde ao domínio informado no *header* da mensagem.

Na maioria dos casos, o nome de domínio do endereço de origem da mensagem é falsificado. Assim, se o resultado da comparação do endereço IP com o domínio informado pelo remetente corresponder a uma das duas primeiras situações descritas anteriormente, é muito provável que a mensagem possa ser considerada *spam*.

Mecanismos providos pelo protocolo e descritos na RFC 2821, como SMTP VRFY, SMTP EXPN e SMTP ETRN, devem ser desativados. Tais mecanismos permitem aos *spammers* confirmar, através de ataques de força bruta, por exemplo, a lista de usuários de um determinado servidor e, assim, obter novos destinatários de mensagens indesejadas.

Como forma de reforçar a segurança e evitar ao máximo a aceitação de *spams*, além das providências citadas e recomendadas pela RFC 2505, a metodologia descrita a seguir deverá também adotar um bloqueador de anexos.

Atualmente, os vírus de computador e outras pragas digitais equivalentes (*scams*, *spywares*, *trojans*, *worms*, etc) têm se propagado, basicamente, pelo repasse de *e-mails* de máquinas infectadas aos endereços constantes em seus catálogos. Na maioria dos casos, estas mensagens contêm anexos que podem ser executados com a simples visualização da mensagem. Tais anexos, que usam extensões diferentes das já tradicionais *.bat*, *.com*, *.exe*, *.dll* e *.pif*, tais como *.cpl*, *.scr*, *.reg*, entre outras, também devem ser filtrados.

3. Descrição Metodológica

A adoção de medidas que buscam minimizar o *spam* deve agir de duas maneiras, procurando se certificar que o servidor não está sendo utilizado como *relay* não-autorizado e evitando a entrada de mensagens indesejadas.

Em um primeiro momento, devem ser tomadas atitudes que impeçam a utilização do MTA como propagador de *spam*, através de *open proxy* ou *open relay*.

Como os serviços *proxy* fogem ao escopo deste artigo, será assumido que eles estão corretamente configurados ou simplesmente desativados. Serviços *proxy* abertos e mal-configurados permitem que o *spammer* envie uma requisição HTTP com alvo na porta 25 (SMTP) do *host* de destino [Hambridge and Lunde 1999]. Criado este caminho, é por aí que poderão ser enviadas as mensagens com os mais variados destinos. O endereço IP que constará nas mensagens será o do *host* possuidor do serviço *proxy* que está sendo abusado.

Como parte da correta configuração do MTA Postfix, deve-se certificar que o *relay* só poderá ser efetuado por *hosts* situados nas faixas de rede que possuem permissão para tanto. Isso é feito com a adição das seguintes linhas no arquivo */etc/postfix/main.cf* [Venema 1999]:

```
mynetworks_style = subnet
mynetworks=192.168.0/24, 192.168.1/24, IP/máscara...
```

A seguir, as seguintes linhas também devem ser adicionadas ao arquivo */etc/postfix/main.cf*, de forma a atender as recomendações da RFC 2505 e solicitar que sejam efetuadas as consultas ao serviço de DNS reverso:

```
smtpd_client_restrictions =
permit_mynetworks,reject_non_fqdn_sender,reject_non_fqdn_recipient,
```

```

reject_unknown_sender_domain,reject_unknown_recipient_domain,
hash:/etc/postfix/access,reject_unknown_client
smtpd_sender_restrictions =
permit_mynetworks,check_sender_access hash:/etc/postfix/access,
reject_unknown_sender_domain,check_relay_domains
smtpd_recipient_restrictions =
permit_mynetworks,reject_invalid_hostname,reject_non_fqdn_sender,
reject_non_fqdn_recipient,reject_unknown_sender_domain,
reject_unknown_recipient_domain,reject_unauth_pipelining,
reject_unauth_destination,permit

```

Cabe salientar que após cada alteração no arquivo */etc/postfix/main.cf*, o comando '*postfix reload*' deve ser executado para que as alterações tenham efeito.

O arquivo */etc/postfix/access*, citado em algumas das linhas acima, refere-se ao mecanismo que gerencia listas brancas e listas negras, respectivamente, e que permite um ajuste fino à técnica adotada. Este ajuste busca evitar que ocorra bloqueio ou aceitação de mensagens de forma indevida. Em alguns poucos casos, há servidores de DNS que não efetuam a correta correspondência de seu DNS reverso. Entretanto, isto é apenas um sinal de que o servidor está mal-configurado. Não significa que e-mails provenientes destes lugares possam ser classificados como *spam* e devam ser rejeitados. Por outro lado, há também servidores utilizados para o envio de *spam* que possuem o DNS reverso corretamente configurado.

O conteúdo deste arquivo deve ser inserido ou editado, de maneira que ele siga o seguinte formato:

<i>aaa.bbb.ccc.ddd</i>	<i>OK</i>
<i>zzz.bbb.ccc.ddd</i>	550 <i>Spam permanentemente negado</i>

A primeira linha informa ao MTA que este deve aceitar mensagens provenientes do endereço *aaa.bbb.ccc.ddd*. A segunda linha diz que o código de erro permanente 550, seguido da mensagem '*Spam permanentemente negado*', deve ser retornado, na eventualidade da ocorrência de conexões por parte do IP *zzz.bbb.ccc.ddd*.

Cada alteração no arquivo */etc/postfix/access* requer a execução do comando *postmap /etc/postfix/access*, para que as mesmas façam efeito.

Para que o MTA Postfix faça consultas em RBLs mantidas por entidades que combatem o *spam*, algumas configurações devem ser efetuadas ao arquivo */etc/postfix/main.cf*. O exemplo a seguir demonstra como seria solicitada a consulta em três RBLs distintas:

```
maps_rbl_domains = sbl.spamhaus.org, relays.orbd.org
```

Além disso, a seguinte linha deve ser acrescentada às seções *smtpd_client_restrictions* e *smtpd_recipient_restrictions*:

```
reject_maps_rbl
```

Para que ocorra o bloqueio de anexos potencialmente nocivos, o Postfix deve ser configurado para que faça a análise de conteúdo dos *headers* e do corpo das mensagens. Essa checagem é efetuada através da avaliação de expressões regulares e necessita que o módulo PCRE (*Perl Compatible Regular Expression*, Expressões Regulares Compatíveis Perl) esteja instalado. Para que a verificação seja efetuada, as seguintes linhas devem ser adicionadas no arquivo */etc/postfix/main.cf*.

```

header_checks = pcre:/etc/postfix/header_checks
mime_header_checks = $header_checks
nested_header_checks = $header_checks
body_checks = pcre:/etc/postfix/body_checks
body_checks_size_limit = 51200

```

Os arquivos */etc/postfix/header_checks* e */etc/postfix/body_checks* (referenciados nas linhas acima) devem conter, linha por linha, as expressões regulares que devem ser avaliadas e as ações que devem ser tomadas em casos positivos. Diversas características das mensagens podem ser validadas, tais como assunto, conteúdo, remetente, destinatário e anexos. Assim, cada mensagem que entra no servidor é verificada e pode ser rejeitada, caso não se enquadre nas regras pré-estabelecidas. A linha a seguir demonstra o conteúdo que estes arquivos devem apresentar para que ocorra o bloqueio de anexos com extensões executáveis potencialmente nocivas (como, por exemplo, *.bat*, *.com*, *.cpl*, *.dll*, *.exe*, *.pif*, *.reg* e *.scr*) e o retorno que deve ser fornecido ao remetente.

```
/^Content-Type|Disposition):.*(file)?name=.*.(bat|com|cpl|dll|exe|pif|reg|scr) / REJECT Mensagem rejeitada devido à presença de um arquivo ${3} anexado.
```

As técnicas abordadas até aqui se referem às relacionadas diretamente ao MTA Postfix. Entretanto, a sua aplicação isolada não é suficiente, tendo em vista a variação das técnicas utilizadas por *spammers*. Deve ser fortemente cogitada a possibilidade de combinar à metodologia um filtro estatístico bayesiano. Há diversas ferramentas GPL disponíveis, dentre as quais se destaca o SpamAssassin, que se encontra disponível nas versões mais atuais do sistema operacional Linux. É bastante recomendável, ainda, a combinação de um sistema antivírus, como o Amavis.

4. Conclusões

De acordo com a RFC 2505, o *spam* é considerado extremamente prejudicial à estrutura da internet e a seus usuários pelos seguintes motivos:

- ocorre em grande volume, isto é, as pessoas recebem grande quantidade de *e-mails* inúteis em suas caixas postais;
- não há relação direta entre eventuais áreas de interesse do destinatário e o *spam* enviado;
- há a geração de um grande custo financeiro para os provedores de acesso e corporações, que têm de manter estruturas robustas (como alta largura de banda e servidores e discos de alta capacidade e desempenho) para comportar o tráfego intenso; e,
- ocorre a utilização de *relays* não-autorizados como forma de propagar o *spam*. Ao evitar que seja exposta a verdadeira origem, o *spammer* também demonstra que possui consciência de que seus atos são prejudiciais.

Portanto, a metodologia descrita neste artigo busca, através da união de diversas técnicas, minimizar a ocorrência de *spam* em MTAs Postfix. As técnicas abordadas (verificação de DNS reverso, uso de RBLs, bloqueio de anexos, etc) possuem desvantagens que se tornam menos evidentes quando combinadas.

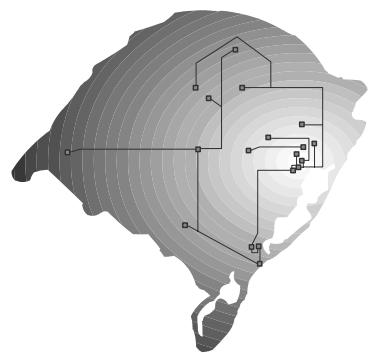
Com a finalidade de obter alguns resultados preliminares, foi efetuada uma pequena experimentação. Foram analisadas, através dos *logs* do sistema, 458 mensagens recebidas no período de 48 horas, entre os dias 07 e 08 de junho de 2004, pelo servidor *mail.cefetrs.edu.br* (onde foi implantada a metodologia descrita). Dentre as mensagens aceitas, 218 eram desejadas (95,19%) e 11 consideradas *spam* (4,81%). Entre as bloqueadas, 225 eram consideradas *spam* (98,25%) e 4 desejadas (1,75%). Apesar do caráter meramente ilustrativo da experimentação, seus resultados podem ser considerados satisfatórios. Cabe salientar que testes mais apurados deverão ser efetuados.

A rejeição de 100% de mensagens indesejadas é uma meta que surge como inalcançável, em virtude das variações de técnicas utilizadas pelos *spammers* e da vasta quantidade de servidores mal-configurados e que podem sofrer explorações de *relay*. No entanto, a adoção de algumas medidas que combatam o *spam* pode amenizar os inúmeros danos ocasionados por este problema.

O correio eletrônico já é um meio de comunicação consagrado e extremamente difundido em virtude de sua simplicidade, agilidade e eficiência. Entretanto, o *spam* é uma grande ameaça ao futuro desta ferramenta. Assim, cada membro da comunidade internet deve auxiliar, de alguma forma, no combate a essa enorme quantidade de lixo eletrônico que é propagada diariamente. O *spam* só ocorre porque há um bom retorno financeiro decorrente desta prática. A primeira forma de combate deve ser, portanto, o boicote às suas inúmeras ofertas. Caso não ocorra algum viés na tendência do aumento da remessa de mensagens indesejadas, dentro de pouco tempo não haverá mais condições para a execução de nenhuma tarefa útil – apenas a de apagar *spams*.

5. Referências Bibliográficas

- Androutsopoulos, Ion *et al* (2000) “An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages”, National Center for Scientific Research ‘Demokritos’, Athens, Greece, disponível em <http://portal.acm.org/citation.cfm?id=345569&dl=ACM&coll=GUIDE> e em http://adsabs.harvard.edu/cgi-bin/nph-bib_query?2000cs.....8019A, março.
- Andreoly, A. *et al* (2001) “Controle de SPAM baseado em pré-detecção da vulnerabilidade de Mail Relay”, Computer Emergency Response Team (CERT-RS/UFRGS), disponível em http://www.rnp.Br/news/gen/0207/mail_relay.htm, abril.
- Hambridge S. and Lunde A. (1999) “RFC 2635: Don’t spew – A Set of Guidelines for Mass Unsolicited Mailings and Postings (spam)”, disponível em <http://www.ietf.org/rfc/rfc2635.txt>, março.
- Klensin J. (2001) “RFC 2821: Simple Mail Transfer Protocol”, disponível em <http://www.ietf.org/rfc/rfc2821.txt>, março.
- Lindberg G. (1999) “RFC 2505: Anti-Spam Recommendations for SMTP MTAs”, disponível em <http://www.ietf.org/rfc/rfc2505.txt>, março.
- Meirelles, F. (2004) “Pesquisa Anual de Administração de Recursos de Informática – Sumário de Resultados da Pesquisa”, 15ª edição, Escola de Administração de Empresas de São Paulo da Fundação Getulio Vargas, disponível em <http://www.fgvsp.br/cia/pesquisa/Pesq04GV.pdf>, abril.
- Postal, J. (1982) “RFC 821: Simple Mail Transfer Protocol”, disponível em <http://www.ietf.org/rfc/rfc821.txt>, março.
- Spamhaus Block List (2004) “Top 10 Spam Countries – April 2004”, Disponível em <http://www.spamhaus.org>, abril.
- United Nations Statistics Division (2004) “Internet Users (ITU estimates 2003)”, Disponível em http://unstats.un.org/unsd/mi/mi_series_results.asp?rowID=608&fID=r15&cgID=, março.
- Venema, W. (1999) "Postfix Documentation", disponível em <http://www.postfix.org/documentation.html>, março.



Sessão Técnica 6

Segurança

Segurança em Redes Ópticas: Tipos de Ataques e Métodos de Detecção

André Panisson, Ricardo Lemos Vianna, Rodrigo Sanger Alves, Juergen Rochol

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

{panisson, rvianna, sanger, rochol}@inf.ufrgs.br

Resumo. Redes ópticas estão sob intenso desenvolvimento e seu uso tem crescido rapidamente. Portanto, avaliar questões de segurança dessa tecnologia torna-se bastante importante. Sendo assim, este trabalho objetiva apresentar métodos de ataque e de que maneira os mesmos podem ser detectados. Por fim, será apresentada, como uma solução de segurança, a técnica de criptografia quântica.

1. Introdução

Instituições, e.g. universidades e empresas, têm seu funcionamento cada vez mais dependente das infraestruturas de rede. Nesse contexto, tópicos como segurança em sistemas computacionais tornam-se cada vez mais importantes. Assim, alguns estudos foram iniciados no intuito de investigar fatores de segurança em um tipo de rede que se tem destacado pelo rápido crescimento em função da demanda cada vez maior por largura de banda: as redes ópticas [Kartalopoulos 2003]. Novos métodos de ataque e de detecção de ataque surgem, decorrentes das particularidades inerentes às redes ópticas. Além disso, alguns tradicionais métodos devem ser re-estudados.

Este artigo está organizado como segue. A Seção 2 resume os principais tipos de ataque comuns às redes ópticas. Na Seção 3, importantes métodos de detecção são detalhados. Por fim, na Seção 4, são apresentadas as conclusões deste estudo.

2. Tipos de Ataques em Redes Ópticas

De modo geral, os ataques sobre uma rede qualquer podem ser agrupados em seis categorias principais [Nevaste 1999]: análise de tráfego, *eavesdropping* (escuta), *data delay*, negação de serviço, degradação de QoS e *spoofing*. Entretanto, considerando-se o escopo deste trabalho, algumas simplificações podem ser feitas. Análise de tráfego e *eavesdropping* têm características similares e podem ser tratados juntos. Ataques de atraso (*delay attacks*) serão ignorados devido à imunidade oferecida pela tecnologia óptica a este tipo de ataque [Bergman et. al. 1998]. *Spoofing* pode ser prevenido com a adoção de criptografia de dados, que será abordada no final deste trabalho. Negação de serviço pode ser considerado como uma degradação de QoS levada ao extremo, e serão tratados juntos sob o nome “rompimento de serviço”. Assim, as seis categorias iniciais de ataque ficam reduzidas a duas: *eavesdropping* e rompimento de serviço.

Para realizar um dos dois tipos de ataques, o atacante necessita de um método. São abordados aqui três métodos em especial [Nevaste 1999, Elsenpeter e Velte 2002]: *In-Band Jamming*, *Out-of-Band Jamming* e Observação Não-Autorizada. Os dois

primeiros são usados para realizar rompimento de serviço, enquanto o último é empregado para realizar *eavesdropping*. No método *In-Band Jamming*, o atacante injeta um sinal para reduzir a capacidade do receptor de interpretar os dados transmitidos. No *Out-of-Band Jamming*, o atacante explora *crosstalk* em componentes ópticos, injetando um sinal de comprimento de onda diferente do usado na banda de comunicação, porém dentro da banda passante do componente. Já na Observação Não-Autorizada, o atacante escuta componentes *crosstalk* de um sinal adjacente, através de um recurso compartilhado, para obter informações deste sinal adjacente.

3. Métodos para Detecção de Ataques

Nesta seção são apresentados métodos para detecção de ataques, e de que forma eles relacionam-se com os tipos de ataques apresentados anteriormente. Algoritmos distribuídos para localização de ataques estão fora do escopo deste trabalho, mas podem ser encontrados, com mais detalhes, em [Bergman et. al. 1998].

3.1. Métodos de Detecção de Energia de Banda

Detecção de energia consiste na medida da energia óptica recebida em uma banda. Pode ser usada para registrar uma mudança de energia em relação ao valor esperado. Pelo fato de um valor medido ser comparado com um valor esperado, uma leve diminuição na energia pode levar um longo tempo para ser detectada. Se a análise estatística for feita sobre grandes números ou sobre um longo período de tempo, então um tempo médio muito longo pode ser necessário para estabelecer com uma certeza razoável que uma mudança na amostra é estatisticamente significante. Mudanças pequenas, mas detectáveis, na energia recebida podem não ser atribuídas a ataques (por exemplo, envelhecimento de componentes ou reparos na fibra), sem assim afetar os sinais de comunicação. Portanto, muitos métodos usam técnicas de detecção de energia sobre um limiar, com os devidos limiares relacionados aos níveis nos quais os serviços de comunicação serão degradados.

No caso de interferências *in-band*, a energia no receptor não será reduzida, mas será aumentada. Um detector de limiar pode verificar claramente um ataque de interferência desse tipo. Uma interferência esporádica poderia aumentar a taxa de erros de forma inaceitável, sem causar um aumento na energia que justifique o disparo de alarme, particularmente se a análise estatística do sinal recebido for claramente determinada. Mesmo que esta análise seja claramente determinada, a natureza esporádica do ataque pode não causar anomalias estatísticas por um longo período de tempo, embora afete a baixa taxa de erros que é necessária para o funcionamento do canal de transmissão.

No caso de interferências *out-of-band*, haverá concorrência pelo ganho do sinal nos amplificadores, e a energia do canal recebido poderá ser reduzida. No entanto, certos ataques desse tipo podem levar a uma degradação no sinal sem uma redução significativa na sua energia. Supondo que um sinal precise atravessar um amplificador EDFA e que haja competição por ganho neste amplificador, o sinal pode não ser adequadamente amplificado.

Técnicas de detecção de energia também têm sido usadas para detectar escutas por perdas de energia. A base para esses tipos de sistemas de segurança é que

apenas uma escuta que drene uma quantidade suficiente de energia do sinal pode apresentar uma detecção satisfatória de tal escuta. Há muitas desvantagens nesse sistema, entre elas o fato de que não consegue detectar um ataque em que um sinal de interferência seja adicionado após a escuta, pois, em contrapartida à perda de energia, haverá um ganho devido à interferência.

3.2. Métodos de Análise do Espectro Óptico

Analisadores espectrais (OSAs) medem o espectro de um sinal óptico, e há muitas implementações desses analisadores. Podem oferecer um diagnóstico mais detalhado do que uma simples detecção de energia. Eles são capazes de detectar mudanças na forma do espectro, mesmo que essa mudança não implique em uma mudança na energia sobre todo o canal. Por exemplo, dois sinais podem ter a mesma energia total, mas espectros diferentes. Um analisador espectral é capaz de distinguir entre dois sinais, enquanto um detector de energia não consegue. Embora analisadores de espectro possam oferecer mais informações que detectores de energia, eles ainda dependem de comparações estatísticas entre amostras. Dessa forma degradações não freqüentes do sinal não serão detectadas ou serão detectadas apenas após um longo período de tempo. Embora ofereçam mais informações que detectores de energia, analisadores espectrais geralmente assumem a existência de alguns efeitos em médio prazo que os fazem mais lentos que outros métodos de detecção.

No caso de interferência *in-band*, um OSA detecta aqueles que afetam de forma significativa o espectro recebido. Fornece mais informação do que detectores de energia, mas para o caso de interferência por *crosstalk*, não fornece mais informações do que um conjunto de detectores de energia específicos para cada extensão de onda.

Já no caso de ataques de interferência *out-of-band*, um OSA será capaz de determinar a fonte do ataque, se a banda analisada pelo OSA for suficientemente larga de forma a abranger a freqüência do ataque.

3.3. Métodos de Sinais Piloto

Sinais piloto são sinais enviados nas mesmas conexões e nodos que os dados de comunicação, porém distinguíveis desses dados. Tem como propósito a detecção de interrupções na transmissão. Sinais piloto são freqüentemente enviados em portadoras diferentes do sinal de transmissão, mas podem também ser distinguidos do *payload* de dados por determinados “*time slots*” (em um sistema TDMA) ou por determinados códigos (em sistemas CDMA). Sinais piloto estão geralmente localizados em portadoras com freqüências entre os canais WDM, assim como fora da banda de transmissão. Se os sinais piloto estão numa freqüência próxima dos canais de transmissão, são geralmente referenciados como sinais “*subcarrier multiplexed*” (SCM). Tais sinais permitem tanto a transmissão de sinalização de rede quanto os sinais pilotos na mesma portadora, pois o sinal não precisa necessariamente ser estático.

Em interferências *out-of-band*, a concorrência por ganho nos amplificadores afeta todos os canais da banda, embora não sejam todos igualmente afetados. Se os sinais piloto usam os mesmos amplificadores que os sinais de dados, então serão afetados da mesma forma. Caso contrário, não é possível detectar ataques desse tipo.

3.4. Métodos que Utilizam OTDRs (Reflectômetros Ópticos no Domínio do Tempo)

OTDRs (*Optical Time Domain Reflectometry*) [Abbate e Caputo 2002] podem ser consideradas aplicações especiais de sinais piloto. Ao invés de analisar um sinal piloto no ponto em que os sinais de comunicação são recebidos, a análise é feita sobre o seu eco. Por causa do freqüente uso de OTDRs e pelo fato destes analisarem o eco do sinal piloto no lugar do sinal em si, OTDRs são considerados em separado, embora compartilhem de muitas características dos sinais piloto. OTDRs são geralmente usados para diagnosticar falhas, curvas (cotovelos) e desperdícios na fibra. Dessa forma, normalmente são mais bem adaptados para detectar ataques que envolvem escutas na fibra. Pelo fato de operarem sobre a reflexão de sinais de volta da fibra, podem dar informação sobre os ataques que estejam acontecendo. Note-se que o sinal usado pelos OTDRs podem ser usados como sinais de supervisão e gerenciamento, podendo também ser submetido à interferência da mesma forma que os sinais piloto. O uso de isoladores ópticos, em conjunção com amplificadores ópticos, é comum, e pode exigir a existência de OTDRs em cada amplificador.

No caso de ataques de interferência *in-band*, parte do sinal de interferência será retornado nas reflexões e se tornará observável. Em redes ramificadas, tais como redes nas quais diferentes canais são demultiplexados em diferentes fibras, secções diferentes podem ser testadas através do envio de diferentes sinais de teste, cada um em um canal diferenciado.

3.5. Novos Métodos para Detecção de Ataques em Redes Ópticas

Alguns métodos novos propostos por Médard, Marquis e Chinn [Médard et. al. 1998] são baseados na noção de que os sinais de entrada e saída de um dispositivo devem ter uma relação matemática que é conhecida pelo sistema de gerenciamento da rede que oferece o serviço. Assim sendo, uma comparação destes sinais será capaz de detectar um ataque se alguma função não produzir uma saída conforme um conjunto conhecido de parâmetros.

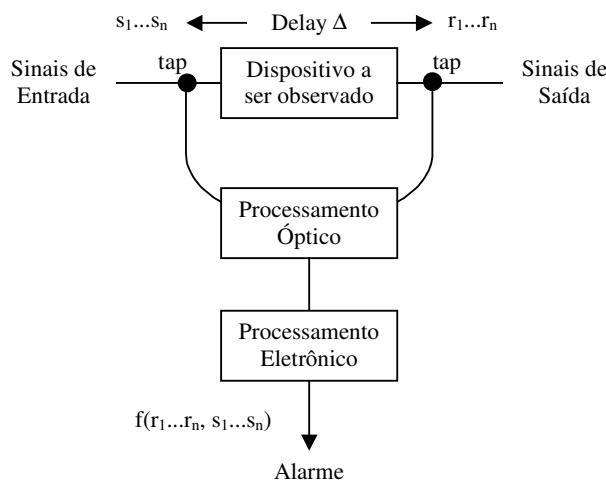


Figura 1. Sistema de detecção de ataques em redes ópticas

O caso geral é mostrado na Figura 1, em que múltiplos sinais de entrada ($s_1 \dots s_n$) são enviados ao dispositivo na esquerda, e os sinais de saída ($r_1 \dots r_n$) aparecem na direita. O dispositivo analisado poderia ser um amplificador óptico em um link ou nodo da fibra, um multiplexador/switch/demultiplexador ou um combinador. O método de detecção insere uma escuta em ambos os pontos de entrada e saída, e consome uma insignificante, porém conhecida porção do sinal para fins de teste. Ambas escutas de entrada e saída são conectadas a uma unidade de processamento óptico opcional. São então transformadas em um sinal elétrico que é processado por uma unidade de processamento eletrônica, cuja saída é função dos sinais de entrada e saída do dispositivo, função essa que faz a medição da operação do dispositivo com respeito a alguns parâmetros. O valor de saída da função determina se será ou não gerado um sinal de alarme. Esta técnica não requer a modificação do dispositivo, pois apenas adiciona um invólucro em torno deste. O alarme poderia estar conectado a um sistema de gerenciamento que poderia processar os alarmes de todos os dispositivos observados.

4. Conclusões

Neste artigo, a questão dos ataques a redes ópticas foi abordada. Inicialmente, apresentou-se uma classificação dos tipos de ataques a redes. Alguns desses foram, então, agrupados, de forma que puderam ser tratados juntos. Outros, como ataques de atraso, os quais não fazem sentido para redes ópticas, não foram tratados. Ataques tipo *spoofing*, que podem ser prevenidos através da adoção de esquemas de criptografia, também estão fora do escopo deste trabalho. Além disso, foram vistos três métodos em especial para realização de ataques dos tipos *eavesdropping* e rompimento de serviço: *In-Band* e *Out-Band Jamming* e Observação Não-Autorizada. Em seguida, diversos métodos para detecção de ataques foram apresentados.

Um campo de pesquisa que poderá colaborar muito na questão da segurança em redes ópticas é a questão da criptografia quântica. Embora não possa impedir ataques do tipo rompimento de serviço, a criptografia quântica pode evitar ataques da categoria *eavesdropping*. Em [Gisin et. al. 2002, Bienfang et. al. 2004], pode-se encontrar uma técnica de gerar chaves criptográficas simétricas de maneira que um atacante escutando a transmissão não poderá descobrir a chave.

Nessa técnica utilizam-se lasers para gerar pulsos individuais de luz, os fôtons. A certeza de que os dados não poderão ser lidos é assegurada pelo fato de que a simples tentativa de leitura do fóton, conforme a teoria quântica, causa a destruição deste. Cada fóton pode ser enviado em dois modos de polarização diferentes. Em cada modo, a polarização do fóton em uma das duas orientações possíveis representa o bit 0, e a outra representa o bit 1. O transmissor escolhe randomicamente um modo e uma orientação para cada fóton e o envia em um canal chamado “canal quântico”. O receptor escolhe randomicamente um modo de leitura e tenta detectar o fóton. Nesse momento, o transmissor usa um segundo canal para informar ao receptor qual o modo de envio usado para transmitir os fôtons. O receptor, então, descarta as medidas feitas no modo incorreto, e informa ao transmissor quais as medidas que foram feitas corretamente (mas não os seus respectivos valores, que formarão a chave). Com isso, o transmissor sabe qual foi a chave gerada no receptor. Nesse ponto, tem-se ambos, transmissor e receptor, com a mesma chave, podendo os mesmos realizar uma comunicação criptografada em um canal qualquer. Se uma tentativa de escuta é feita no canal quântico, uma escolha

randômica de modo de leitura deverá ser feita. A leitura do fóton implica na sua conversão para energia elétrica e, por isso, sua destruição. O atacante deverá gerar um novo fóton para ser enviado ao receptor, mas é impossível saber se o seu modo de leitura estava correto, e, portanto, se o valor lido também estava. Alguns valores serão, então, enviados errados ao receptor, causando diferenças entre as chaves do transmissor e do receptor. Através da comparação de pequenas porções da chave gerada, o transmissor e o receptor percebem que houve escuta no canal quântico, e a chave será descartada.

Por fim, convém ressaltar que a pesquisa a respeito de segurança em redes ópticas deve acompanhar as freqüentes inovações feitas pelos atacantes, além da sofisticação de suas técnicas. Os métodos de detecção de ataque apresentados neste estudo constituem um bom panorama da área, porém as inovações na área de segurança são freqüentes, tanto do ponto de vista de ataque quanto - consequentemente - de defesa, reforçando a necessidade de constante atualização.

Referências

- Abbate, A. L. R. e Caputo, M. R. C. (2002) "Aplicação do OTDR na Análise de Problemas de Atenuação em Fibras Ópticas: Estudo de Casos", In: Inatel Revista Telecomunicações, Vol. 5, Número 2.
- Bergman, R., Médard, M. e Chan, S. (1998) "Distributed Algorithms for Attack Localization in All-Optical Networks", In: The Internet Society's Symposium on Network and Distributed System Security.
- Bienfang, J. C., Gross, A. J., Mink, A., Hershman, B. J., Nakassis, A., Tang, X., Lu, R., Su, D. H., Clark, C. W., Williams, C. J., Hagley, E. W. e Wen, J. (2004) "Quantum key distribution with 1.25 Gbps clock synchronization" In: Optics Express, Vol. 12, Issue 9.
- Elsenpeter, R. e Velte, T. J., Optical Networking: A Beginner's Guide, McGraw-Hill, 2002.
- Gisin, N., Ribordy, G., Tittel, W. e Zbinden, H. (2002) "Quantum cryptography" In: Reviews of Modern Physics, Vol. 74, Issue 4.
- Kartalopoulos, S. V., DWDM: Networks, Devices and Technology, IEEE Press, 2003.
- Médard, M., Marquis, D. e Chinn, S. R. (1998) "Attack Detection Methods for All-Optical Networks", In: Internet Society's Symposium on Network and Distributed System Security.
- Nevaste, K. (1999) "Optical Network Security", Disponível em: http://www.tml.hut.fi/Opinnot/Tik-110.501/1999/papers/optical_netsec/onetsec.html, May.

Aplicando a Técnica de Raciocínio Baseado em Casos na Identificação de Cenários de Intrusão em *Logs de Firewalls*

Samir Lohmann, Cristina Melchior, Luciano Paschoal Gaspary

Programa Interdisciplinar de Pós-Graduação em Computação Aplicada

Universidade do Vale do Rio dos Sinos (UNISINOS)

Av. Unisinos, 950 – 93022-000 – São Leopoldo – RS – Brasil

samir.lohmann@ska.com.br, {cmelch, paschoal}@exatas.unisinos.br

Resumo. As ferramentas de análise de logs de firewalls existentes atualmente são muito úteis na obtenção de certos diagnósticos de problemas de redes corporativas. Entretanto, falta a elas pro-atividade para que encontrem cenários de intrusão automaticamente. Este artigo apresenta os progressos feitos até o momento no desenvolvimento de um módulo para uma destas ferramentas (SEFLA) que, através da técnica chamada Raciocínio Baseado em Casos, analisa os eventos gerados por um firewall e identifica cenários de intrusão de forma automática. Espera-se também que tal módulo auxilie na correta configuração de regras de filtragem de firewalls.

1. Introdução

Do ponto de vista da gerência de segurança, os *logs* de firewalls são ricos em informações, pois através deles pode-se identificar (a) suspeitas de ataques, (b) os serviços mais e os menos requisitados, (c) os *hosts* que ocupam mais e menos banda, (d) os principais usuários e (e) eventuais anomalias [Chapman e Zwicky, 1995; Symantec, 2001; Taylor, 2002].

Existem diversas ferramentas de análise de *logs* que permitem classificar, caracterizar, armazenar históricos e visualizar de forma amigável os eventos gerados por firewalls [Locatelli, 2003] Dentre elas, pode-se citar o Reptor [Wankwood, 2003] e o SEFLA [Locatelli, 2003]. Tais ferramentas possuem diferentes funcionalidades, mas as estatísticas e os eventos apresentados são meros espelhos do *log*. Embora sejam de grande valia para obter alguns diagnósticos, falta a elas pro-atividade, já que a identificação de cenários de intrusão é feita pelo usuário, através da análise manual dos eventos. Esta abordagem é pouco eficiente na detecção de comportamentos suspeitos, pois exige monitoramento constante por parte do gerente de segurança, sem o qual cenários de intrusão são simplesmente ignorados. Isto acontece devido ao grande volume de eventos gerados diariamente pelo *firewall*, e também porque se depende exclusivamente do usuário para detectar cenários de intrusão.

O objetivo deste trabalho é desenvolver um módulo para a ferramenta SEFLA que, através da técnica de Inteligência Artificial denominada Raciocínio Baseado em Casos (CBR – *Case-based Reasoning*) [Kolodner, 1993], analise os eventos gerados pelo *firewall* Symantec Enterprise e identifique cenários de intrusão de forma automática. Apesar de não ser um objetivo primário, espera-se também que a ferramenta seja capaz de auxiliar na correta configuração de regras de filtragem de firewalls.

Os estudos realizados até o momento apontam CBR como uma técnica adequada, pois tem a capacidade de identificar similaridade entre situações novas e situações já conhecidas armazenadas em sua base de conhecimento. Tal base armazena as situações antigas como casos. Durante o processo de raciocínio para a resolução de uma nova situação, esta é comparada aos casos armazenados na base de conhecimento e os casos mais similares são utilizados para propor soluções ao problema corrente [Locatelli et al, 2004].

O artigo está organizado da seguinte forma: a seção 2 explica os fundamentos de CBR e descreve uma proposta de modelagem para casos de intrusão. A arquitetura da ferramenta SEFLA com o novo módulo proposto é introduzida na seção 3. Por fim, na seção 4, são feitas considerações sobre problemas encontrados até agora e previstos para o futuro.

2. Modelagem de Cenários de Intrusão através de Casos

Um caso é uma representação em baixo nível de um problema da vida real. As fases utilizadas na solução de problemas em um CBR típico são o casamento e a adaptação. O casamento é a fase na qual as novas situações que chegam ao sistema (doravante denominadas casos correntes) são comparadas aos casos armazenados no sistema. Se nesta fase for encontrado algum caso armazenado que possua uma similaridade mínima com um caso corrente, parte-se para a adaptação, onde a solução do caso armazenado é adaptada ao caso corrente e, em seguida, avaliada. Atualmente, o protótipo não conta com as fases de adaptação e avaliação.

Parte Administrativa				
Descrição	Descrição do Caso de Intrusão			
Gravidade	[1 , 2 , 3]			
Parte Classificatória				
Classificação	[Mesmo_IP_Origem , Mesmo_IP_Destino , Mesma_Porta_Destino , Diferentes_Portas_Destino , Intervalo_de_x_horas ...]			
Parte Descritiva				
Sintoma n				
Relevância	[1 , 2 , 3]			
Similaridade_Mínima	[1 , 0.5 , 0]			
Número_Min_Eventos				
Atributos do Evento				
Nome	Operador	valor		
Nome_do_Atributo	[Igual , Diferente , Maior_que , Menor_que , ...]	valor_do_Atributo		
Outros Atributos				

Figura 1 – Estrutura de um caso de intrusão

A figura 1 ilustra a estrutura de um caso. Cada caso é composto por uma parte administrativa, que possui a descrição do caso e a gravidade do mesmo, que pode variar de um até três (do menos para o mais grave). A parte classificatória especifica quais são os atributos utilizados no agrupamento de eventos para formar os casos correntes. O agrupamento é explicado na seção 3. Por fim, a parte descritiva contém os sintomas do caso. Cada sintoma tem, obrigatoriamente, uma relevância (de um a três, do menos para o mais relevante), a similaridade mínima necessária (total, parcial ou nenhuma) e o número mínimo de eventos necessários que um sintoma de um caso corrente deve ter para se dizer que equivale ao sintoma do caso armazenado. Além desses atributos obrigatórios, há uma série de atributos opcionais, preenchidos de acordo com cada sintoma. Para cada atributo, pode-se especificar um operador (Igual, Diferente, Maior_que, Menor_que, Intervalo) e um valor, que pode ser fixo ou então uma máscara. No caso do atributo Hora, por exemplo, pode-se especificar, além de um valor fixo (como 12:00), uma máscara como “Madrugada: 00:00 até 6:00”. Certos operadores só poderão ser utilizados em atributos cujo tipo de dados permita tal operador (ex: Maior_que só pode ser usado em campos numéricos).

Parte Administrativa		
Descrição		Acesso bem-sucedido após varredura
Gravidade		3
Parte Classificatória		
Classificação		Mesmo_IP_Origem
Parte Descritiva		
Sintoma 1		
Relevância		1
Similaridade_Mínima		1
Número_Min_Eventos		1
Atributos do Evento		
Nome	Operador	Valor
Tipo	Igual	PORT_SCANNING
Sintoma 2		
Relevância		1
Similaridade_Mínima		1
Número_Min_Eventos		1
Atributos do Evento		
Nome	Operador	Valor
Tipo	Igual	STATISTIC
Protocolo		Igual
Bytes Enviados		Maior_que Bytes Recebidos

**Figura 2 – Acesso bem-sucedido
após varredura**

Parte Administrativa		
Descrição		Upload suspeito
Gravidade		2
Parte Classificatória		
Classificação		Mesmo_IP_Origem
Parte Descritiva		
Sintoma 1		
Relevância		1
Similaridade_Mínima		0.5
Número_Min_Eventos		3
Atributos do Evento		
Nome	Operador	Valor
Tipo	Igual	ACCESS_DENIED
Sintoma 2		
Relevância		1
Similaridade_Mínima		1
Número_Min_Eventos		1
Atributos do Evento		
Nome	Operador	Valor
Tipo	Igual	STATISTIC
Protocolo		Igual
Bytes Enviados		Maior_que Bytes Recebidos

Figura 3 – Upload Suspeito

Nas figuras 2 e 3, dois exemplos de casos são apresentados. No caso ilustrado na figura 2, um usuário executou uma varredura de portas e logo depois conseguiu acesso a algum computador da rede, já que foi gravado um evento estatístico (uma conexão foi realizada com sucesso). Já no caso da figura 3, o usuário obteve alguns acessos negados (o que pode indicar que está tentando explorar senhas fracas) e depois fez um *upload* (já que teve uma conexão para protocolo FTP e enviou mais dados do que recebeu). Isto indica que este usuário pode estar transferindo um arquivo com código malicioso para depois executá-lo.

3. Arquitetura da Ferramenta

SEFLA é uma ferramenta que possibilita estruturar arquivos de *log* do *firewall* Symantec Enterprise (SEF) alimentando, através de um *parser*, um banco de dados MySQL. Como este banco de dados é um arquivo estruturado (ao contrário do *log* original), é possível classificar, caracterizar, armazenar históricos e visualizar de forma amigável os eventos gerados pelo SEF. A interface com o usuário é feita através de *scripts PHP* hospedados em um servidor *web*, que executa os *scripts* e envia para o usuário páginas HTML. A figura 4 ilustra a arquitetura atual de SEFLA.

O módulo apresentado neste trabalho utiliza a mesma base de dados, porém possui um repositório de casos de intrusão conhecidos, além de um mecanismo de agrupamento, recuperação e seleção, que identifica cenários de intrusão e gera alarmes para o gerente de segurança, quando encontrar casos suspeitos. Os novos componentes são mostrados em destaque na figura 4. Os casos armazenados são criados e mantidos pelo gerente da rede, de acordo com o perfil de uso da rede.

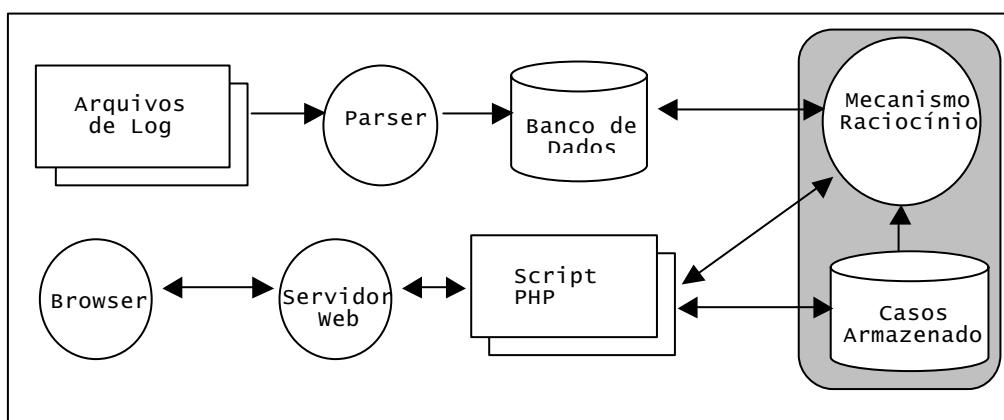


Figura 4 – Arquitetura proposta

Há uma diferença fundamental entre os sistemas CBR citados na literatura e o proposto neste trabalho. Em um CBR tradicional, os problemas chegam prontos de um ambiente externo. Como exemplo, pode-se citar solicitações de suporte a *helpdesk*, e IDSs baseados em CBR, onde cada pacote que chega à rede é analisado. O protótipo proposto não recebe problemas prontos. É necessário, primeiramente, agrupar os eventos do *log* do *firewall*. O critério de agrupamento é obtido a partir do classificador dos próprios casos armazenados, o que faz com que cada caso corrente seja comparado com todos os casos armazenados. O produto desse agrupamento é um conjunto de casos correntes. As fases do CBR aqui proposto são:

- a. Agrupamento: a partir dos atributos do classificador de cada caso armazenado, todos os eventos do *log* são agrupados, e cada grupo corresponde a um caso corrente. Por exemplo, se o classificador for *Mesmo_IP_Origem*, todos os eventos cujo *Endereço_IP_Origem* for 200.244.90.5 formam um caso corrente.
- b. Recuperação: os casos correntes obtidos no passo anterior serão comparados com todos os casos cujo classificador for o mesmo pelo qual foram agrupados. Em seguida, a similaridade com cada um destes casos é calculada. A comparação e o cálculo são realizados através do algoritmo proposto em [Locatelli et al, 2004].
- c. Seleção: Se no passo anterior foram encontrados um ou mais casos correntes que alcançaram ou ultrapassaram a similaridade mínima necessária de algum caso armazenado, o caso corrente com maior similaridade será selecionado e um alerta será gerado ao administrador da rede.

4. Considerações

Com base em estudos preliminares e não conclusivos realizados até o momento, foram percebidos alguns problemas que precisam ser tratados, seja neste trabalho, seja por trabalhos futuros. Em primeiro lugar, o algoritmo necessário para atingir os objetivos aqui propostos é caro computacionalmente, o que é agravado devido ao tamanho dos *logs* analisados, que pode chegar a vários *gigabytes*. Apesar do protótipo não ter compromisso com desempenho (pois não é um IDS e sim, um sistema de apoio ao gerente de segurança), espera-se que o processamento dos *logs* aconteça no menor tempo possível para que os dados analisados sejam tão recentes quanto possível.

Em segundo lugar, os estudos realizados até o momento indicam que determinadas ações de atacantes não são registradas pelo *firewall*. Como exemplo, pode-se citar alguns tipos de varreduras camufladas, as quais não chegam a estabelecer uma conexão TCP. Como um trabalho futuro, sugere-se uma abordagem mista, na qual não só os *logs* do *firewall* são analisados, mas também dados de outras fontes, como *logs* de Sistemas de Detecção de Intrusão (IDS).

Referências

- Chapman, D. B., Zwicky, E. D. (1995) “Building Internet Firewalls”, O'Reilly & Associates.
- Kolodner, J. (1993) “Case-Based Reasoning”, Morgan Kaufmann Publishers.
- Locatelli, F. E. (2003) “Uma Ferramenta baseada na Análise de Logs para Classificação, Caracterização e Correlação de Eventos Gerados pelo Symantec Enterprise Firewall”, Trabalho de Conclusão de Graduação, Centro de Ciências Exatas e Tecnológicas, Universidade do Vale do Rio dos Sinos (UNISINOS).
- Locatelli, F. E., Dillenburg, F., Melchior, C., Gaspary, L. P. (2004) “Identificação de Cenários de Intrusão pela Classificação, Caracterização e Análise de Eventos gerados por Firewalls”, In: XXII Simpósio Brasileiro de Redes de Computadores, Gramado, p. 851-864.
- Symantec Enterprise Firewall (2001) “Symantec Enterprise VPN, and VelociRaptor Firewall Appliance Reference Guide”, Symantec.

Taylor, T. (2002) "Security Complete", Sybex.

Wankwood (2003) "Reptor", <http://www.wankwood.com/reptor>, February.

Avaliação de Geradores de Números Pseudo-Aleatórios

Vera Lúcia da Silva¹, Juliano de Almeida Monte-Mor¹, Nei Yoshihiro Soma¹

¹ Divisão de Ciência da Computação – Instituto Tecnológico de Aeronáutica (ITA)
12228-901 – São José dos Campos – SP – Brazil
`{verals, montemor, nysoma}@ita.br`

Resumo. A geração de números aleatórios é de fundamental importância para o processo da segurança de dados, desta forma, a escolha de um gerador de números pseudo-aleatórios não pode ser realizada ao acaso. Faz-se necessário o uso de conjuntos de testes estatísticos para auxiliar nessa atividade. Este trabalho descreve os testes estatísticos do Sistema DIEHARD, um rigoroso conjunto de testes para geradores de números pseudo-aleatórios. Foram implementados e analisados alguns geradores das linguagens de programação mais utilizadas e outros tradicionalmente conhecidos, visando avaliar seu grau de aleatoriedade.

1. Introdução

Números aleatórios, dentre outras utilidades, são usados para inserir dados imprevisíveis e não-determinísticos em algoritmos para comunicação segura entre computadores, principalmente em protocolos de autenticação, na geração de assinaturas digitais e de chaves usadas na criptografia.

Entretanto, os geradores de números pseudo-aleatórios produzem números por meio da execução repetitiva de um algoritmo. Como não são capazes de gerar números verdadeiramente aleatórios, sua saída pode ser previsível.

É importante que o grau de aleatoriedade e a independência dos valores gerados sejam garantidos para que os resultados sejam satisfatórios, assegurando a autenticação, a confidencialidade e a integridade das informações compartilhadas.

Desta forma, como já ressaltado por Donald Knuth, “um gerador de números pseudo-aleatórios não pode ser escolhido aleatoriamente” [Knuth 1998], faz-se necessário identificar suas características, avaliando sua performance, aleatoriedade e o grau de independência dos valores gerados.

A existência de conjuntos de testes estatísticos objetiva auxiliar nessa atividade. A teoria dos testes estatísticos fornece algumas medidas quantitativas para aleatoriedade. Entretanto, não há um número exato de testes que devem ser realizados para provar que uma seqüência é verdadeiramente aleatória. Quanto mais testes forem executados, maior será a certeza de aleatoriedade da seqüência.

Dentre os conjuntos de testes estatísticos mais difundidos é possível citar: Diehard, Crypt-XS, NIST Statistical Test Suíte. Neste trabalho foram pesquisados os testes estatísticos do Sistema Diehard, por ser esses considerados mais rigorosos, cf. e.g. [Knuth 1998]. Eles foram utilizados para avaliar alguns dos geradores de números pseudo-aleatórios mais utilizados e conhecidos, na tentativa de verificar se são

adequados para a geração de números imprevisíveis e não-determinísticos necessários para aplicações seguras.

Os assuntos abordados neste artigo estão divididos em quatro seções. Na seção 2 são descritos os testes do sistema Diehard. A seção 3 demonstra a análise e os resultados obtidos com a aplicação do conjunto de testes Diehard aos principais geradores de números pseudo-aleatórios. E finalmente, a seção 4 tece as considerações finais deste trabalho.

2. O Sistema Diehard

O Sistema Diehard foi desenvolvido por George Marsaglia, professor e pesquisador do Departamento de Estatísticas e Pesquisas em Computação da Universidade Flórida. Ele consiste num conjunto de 15 testes estatísticos que procuram identificar padrões e distribuições não-uniformes em seqüências aleatórias, e.g. [Marsaglia 1984, 2004].

A idéia principal de cada teste é sumarizada a seguir e na seqüência é apresentado o resumo dos testes estatísticos utilizados (Tabela 1):

(1) *Espaçamento entre Aniversários*: o nome vem do bem conhecido problema de se determinar quantas pessoas são necessárias para que pelo menos duas delas tenham a mesma data de aniversário, com probabilidade não inferior a 50%, que para o caso é de vinte e três pessoas. A idéia do teste é o de se considerar como data de nascimentos, números de 8 bits e testar o espaçamento entre duas ocorrências.

(2) *Permutação com 5 Elementos e Sobreposição*: Utiliza-se uma seqüência de 10^6 inteiros de 32 bits. Cada conjunto de 5 inteiros consecutivos desta seqüência é classificado de acordo com as $5!$ inversões de paridade entre as permutações. Procede-se a uma inserção de um dígito mais à direita e eliminação do mais à esquerda (sobreposição). Devido a essa sobreposição, mostra-se que em termos estatísticos as classes de inversões passam de $5!$ para $5!-21$. O teste é executado duas vezes, tem-se então a geração de duas seqüências de 10^6 inteiros.

(3) *Posto de Matrizes 31 x 31 e 32 x 32*: Consideram-se 40.000 matrizes de 31 x 31 e 32 x 32 que são divididas de acordo com os postos das mesmas, cada um dos elementos das matrizes é formado a partir de números de 31 (32) bits, sendo portanto matrizes com entradas 0 ou 1. Pode-se mostrar que para tais matrizes e com entradas aleatórias, os postos das mesmas são, via de regra, maiores ou iguais que 29. O teste considera a freqüência de ocorrência dos postos de 29 à 31 (ou 32).

(4) *Posto de Matrizes 6 x 8*: A exemplo do caso anterior as matrizes (100.000 delas) são obtidas a partir de seis números inteiros aleatórios de 32 bits. De cada um destes números um dado byte específico é escolhido. Assim, uma linha da matriz é formada pelos 8 bits retirados destes 6 números. Procede-se então ao cálculo do posto da matriz resultante.

O *Teste dos Macacos* é utilizado nos testes (5) e (6), sendo que a idéia é a de simular a digitação de um macaco que ao utilizar um dado teclado, o faz de maneira aleatória. A “produção literária” resultante da digitação do macaco é avaliada em termos de aleatoriedade, e.g. [Marsaglia 1993].

(5) *Fluxo de Bits*: o arquivo de entrada é visto como um fluxo de bits, da forma b1, b2, etc. Considera-se um alfabeto binário onde cada palavra é composta por 20

letras. Assim, a primeira palavra é b1b2...b20, a segunda é b2b3...b21, e assim por diante. O teste conta o número de palavras de 20 letras ausentes numa seqüência de 2^{21} palavras de 20 letras sobrepostas (isso equivale a $2^{21} + 19$ bits, que corresponde ao tamanho da cadeia analisada).

(6) *Sobreposição de Pares, Quádruplos e DNA:* No teste *Sobreposição de Pares* são contadas as palavras com 2 letras de um alfabeto de 2^{10} letras, que não aparecem numa seqüência de 2^{21} palavras sobrepostas. O teste *Sobreposição de Quádruplos* é similar ao anterior, entretanto, nele são consideradas palavras de 4 letras de um alfabeto com 2^5 letras. No *DNA* o alfabeto tem 4 letras e as palavras possuem 10 letras. Pode-se mostrar que para todos os casos, a quantidade média de palavras “perdidas” tem comportamento igual para os três casos, isto é, distribuição normal com igual média e igual desvio padrão.

(7) *Quantidade de 1's em Fluxo de Bytes:* Dado um arquivo de inteiros de 32 bits examina-o como uma seqüência de bytes. Cada byte deve ocorrer por sua vez, com probabilidades iguais respectivamente a 1, 8, 28, 56, 70, 56, 28, 8, 1 de um total de 256 possibilidades. São contadas as ocorrências de cada palavra numa seqüência de 256.000 palavras contendo 5 letras sobrepostas.

(8) *Quantidade de 1's para um Byte Específico:* Como no teste anterior, considera-se um arquivo de entrada como uma seqüência de números inteiros de 32 bits. Para cada inteiro, um byte específico é escolhido, e.g., o mais à esquerda do inteiro com 32 bits. Cada byte pode conter de 0 a 8 números 1 (8 bits), com probabilidades iguais respectivamente a 1, 8, 28, 56, 70, 56, 28, 8, 1 sobre um total de 256 possibilidades. Esses bytes específicos produzem palavras de 5 letras, sendo que as palavras, a exemplo do teste anterior, também se sobrepõem, de modo que cada nova palavra é formada pelas últimas 4 letras da palavra anterior mais à próxima letra lida. As ocorrências são testadas numa seqüência de 256.000 palavras.

(9) *Conjunto de Estacionamentos:* Simula-se um estacionamento onde seja possível estacionar aleatoriamente um carro, representado por um círculo de raio 1. O objetivo do teste é estacionar sucessivamente os carros no estacionamento um a um. Cada tentativa de se estacionar um carro conduz a uma colisão (outro carro na posição escolhida) ou a um sucesso, onde cada sucesso acarreta um incremento à lista dos carros já estacionados. Pode-se mostrar que a distribuição da quantidade de sucessos é uma dada distribuição normal.

(10) *Distância Mínima em Quadrado:* Nesse teste escolhem-se 8000 pontos aleatórios em um quadrado de lado 10.000. São determinadas as distâncias mínimas entre todos os $(n^2 - n) / 2$ pares de pontos. Pode-se mostrar que se a distribuição dos pontos for realmente aleatória, então d^2 , o quadrado da distância mínima obedece à uma dada distribuição exponencial.

(11) *Distância em Esferas:* São escolhidos 4000 pontos aleatórios num cubo de aresta 1000. Cada ponto é considerado como sendo o centro de uma esfera cujo raio é o segmento de reta desse centro com seu vizinho mais próximo. Pode-se mostrar que o volume da menor destas esferas obedece a uma dada distribuição exponencial.

(12) *Compressão:* São utilizados valores de ponto flutuante no intervalo [0, 1), onde cada um desses é obtido a partir de um número inteiro do arquivo de testes. O método começa com um inteiro $k = 2^{31}$ e são determinadas quantas iterações (J) são

necessárias para reduzir k a 1 usando a “compressão” $k = \lceil kU \rceil$, onde U é um número do arquivo testado. Os valores J ’s são agrupados em 43 classes e as freqüências de ocorrência calculadas.

(13) *Somas Sobrepostas*: A exemplo do teste anterior são escolhidos inteiros com representação em ponto flutuante para que se obtenha uma seqüência U_1, U_2, \dots de variáveis uniformes no intervalo $[0, 1]$. Calcula-se então as somas sobrepostas, $S_1 = U_1 + \dots + U_{100}$; $S_2 = U_2 + \dots + U_{101}$. É possível mostrar que os valores S_i ’s podem ser transformados em uma distribuição uniforme.

(14) *Corridas*: Determina-se quantas seqüências (corridas) de uma dada sucessão ininterrupta de dígitos está em ordem crescente (ou decrescente). Uma corrida de comprimento k consiste em exatamente k dígitos em ordem crescente (ou decrescente). Por exemplo, as corridas |129|8530|789| têm duas corridas crescentes 129 e 789; e uma corrida decrescente 8530.

(15) *Craps*: Vem de um jogo em que cada lance corresponde ao arremesso de 3 dados. São jogadas 200.000 partidas e contabiliza-se tanto a quantidade de vitórias quanto o número de lançamentos de dados necessários para terminar cada jogo. Pode-se mostrar que a quantidade de vitórias é uma distribuição normal caso os dados não sejam viciados.

Tabela 1 – Diehard e seus respectivos Testes Estatísticos.

Teste Diehard	Teste Estatístico
1. Espaçamento entre Aniversários	χ^2 com 1000 graus de liberdade
2. Permutação com 5 Elementos e Sobreposição	χ^2 com 99 graus de liberdade
3. Posto de Matrizes 31×31 e 32×32	χ^2 com 3 graus de liberdade
4. Posto de Matrizes 6×8	χ^2 com 2 graus de liberdade
5. Fluxo de Bits	Distribuição Normal $\mu=141.909$ e $\sigma=428$
6. Sobreposição de Pares, Quádruplos e DNA	Distribuição Normal $\mu=141.909$ e $\sigma=290$
7. Quantidade de 1’s em Fluxo de Bytes	χ^2 com 2500 graus de liberdade
8. Quantidade de 1’s para um Byte Específico	χ^2 com 2500 graus de liberdade
9. Conjunto de Estacionamentos	Distribuição Normal $\mu=3.523$ e $\sigma=21.9$
10. Distância Mínima em Quadrado	Kolgomorov-Smirnov $\mu=0.995$
11. Distância em Esferas	Kolgomorov-Smirnov $\mu=40\pi$
12. Compressão	χ^2 com 42 graus de liberdade
13. Somas Sobrepostas	Kolgomorov-Smirnov $\mu=0.5$
14. Corridas	Kolgomorov-Smirnov
15. Craps	χ^2 com 21 graus de liberdade

Os testes do Diehard apresentam seus resultados baseados em análise estatística. Cada teste retorna um valor p (p -value), sendo este obtido por meio da função $p=F(X)$, onde F é a distribuição normal de amostras aleatórias da variável X .

O valor de p varia entre 0 e 1, onde valores próximos a estes limites indicam que a seqüência não apresenta um grau aceitável de aleatoriedade. Considera-se que se o valor de $p < 0.025$ ou $p > 0.975$, o gerador fracassou para o teste, e.g. [Marsaglia 2004]. No caso de um teste ter como resultado mais de um valor p , a falha foi considerada, neste trabalho, quando a quantidade de valores falhos foi maior que a quantidade de valores aceitáveis, e em caso de empate considerou-se que o gerador passou no teste.

3. Avaliação de Geradores

Neste trabalho foram testados e avaliados os seguintes geradores: *Rand* - gerador de 16 bits da biblioteca padrão da linguagem C/C++; *Random* - gerador da biblioteca padrão da linguagem Java; *Random* - gerador da biblioteca padrão da linguagem Delphi; *LRand* - gerador de 32 bits da linguagem C/C++, e.g. [Wichmann and Hill 1987]; *Ran3* - *Numerical Recipes*, e.g. [Press et al. 1992]; e *Blum-Blum-Shub* - muito utilizado na criptografia, e.g. [Rukhin et al. 2001].

Para cada gerador selecionado foram produzidos 10 arquivos de testes, os quais foram submetidos ao Sistema Diehard. Os resultados dos testes de cada arquivo foram contabilizados, identificando os testes para os quais os geradores falharam ou passaram. A Figura 1 apresenta o número de falhas nos 15 testes relacionados na Tabela 1 para os 10 arquivos de entrada.

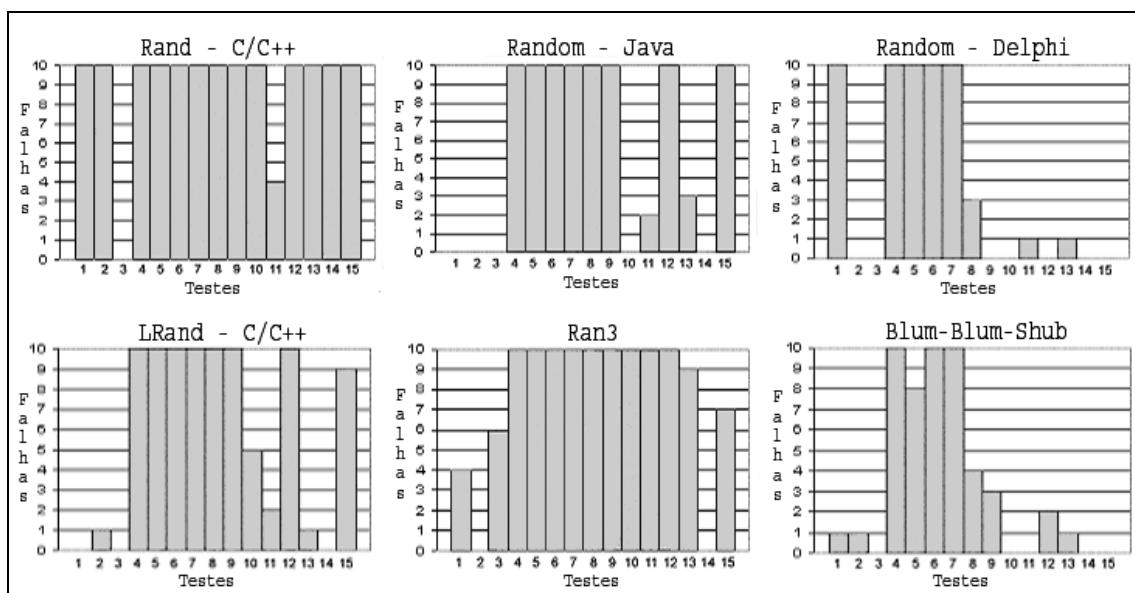


Figura 1. Avaliação dos Geradores.

Nos 10 arquivos de entrada todos os geradores falharam nos testes: Posto de Matrizes 6x8; Sobreposição de Pares, Quádruplos e DNA; e Quantidade de 1's em Fluxo de Bytes. Constatou-se que geralmente as falhas de um gerador sempre ocorrem nos mesmos testes.

Os resultados mostraram que para arquivos de testes diferentes, criados a partir de um mesmo gerador, os testes podem resultar em valores diferentes para p . Entretanto, a diferença dos valores de p é mínima e, em poucos casos, as sequências de um mesmo gerador falham em testes diferentes.

O gerador *Blum-Blum-Shub* teve o melhor resultado nos testes, falhando em média em apenas 5 testes. O gerador *Rand* teve o pior resultado, falhando em média em 13.4 testes. Os geradores *Random* – Linguagem Delphi, *Random* – Linguagem Java, *LRand* e *Ran3*, falharam em média em 5.5, 8.5, 8.8 e 11.6 testes respectivamente.

4. Conclusões

Para a segurança de sistemas computacionais é importante assegurar a autenticação, a confidencialidade e a integridade dos dados compartilhados. A geração de números aleatórios é um fator chave para garantir estes elementos no processo de segurança.

Neste trabalho foram implementados e analisados 06 geradores de números pseudo-aleatórios. Foram criados para cada gerador 10 arquivos com seqüência de números aleatórios diferentes. Os resultados dos testes do Diehard foram analisados, contabilizando a quantidade de testes para os quais os geradores falharam.

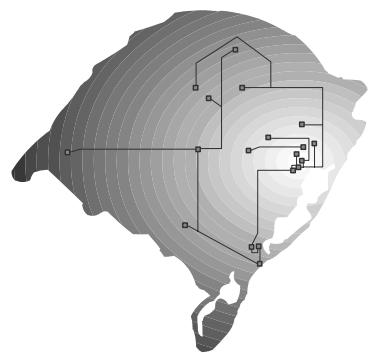
Não é aconselhável a utilização dos geradores *Rand* – Linguagem C/C++ e *Ran3 - Numerical Recipes* em algoritmos para segurança de informações, pois os mesmos falharam em média em 89% e 77% dos testes estatísticos do Diehard respectivamente. Note-se que observação dessa natureza já havia sido apresentada em [Knuth 98] em relação a geradores congruenciais.

Dentre os geradores testados é possível destacar o gerador *Blum-Blum-Shub*, pois ele obteve sucesso em média em 66% dos testes estatísticos do Diehard, e também o gerador *Random* da biblioteca padrão da Linguagem Delphi, pois foi o melhor entre os geradores das linguagens, com média de 63% de sucesso nos testes.

Foi desenvolvido um ambiente gráfico integrado e amigável para o Sistema Diehard, visando tornar menos árdua a tarefa de avaliação de geradores de números pseudo-aleatórios. Este ambiente gráfico, bem como os relatórios técnicos produzidos, encontram-se disponíveis com os autores.

Referências

- Knuth, D.E. “The Art of Computer Programming”, Third Edition. Volume 2: Seminumerical Algorithms, Capítulo 3: Números Aleatórios (Pseudo), Addison-Wesley Publishing Company, Massachusetts, 1998.
- Marsaglia, G. (2004) “The Marsaglia Random Number Cdrom including the Diehard Battery of Tests of Randomness”, <http://stat.fsu.edu/~geo/>, pesquisado em 10/06/2004.
- Marsaglia, G. (1993) “Monkey Tests for Random Number Generators”, In: Computers & Mathematics with Applications, 9, 1–10.
- Marsaglia, G. (1984) “A Current View of Random Number Generators”, Keynote Address, Computer Science and Statistics: 16th Symposium on the Interface, Atlanta. Published by Elsevier Press.
- Press, William H., et al. “Numerical Recipes in C: The Art of Scientific Computing”, Cambridge University Press, 1988-1992.
- Rukhin, A., et al. (2001) “A Statistical Test Suite for Random and Pseudorandom Number Generator for Cryptographic Applications”. In: Nist Special Publication 800-22.
- Wichmann, B., Hill, D. (1987) “Three-Generator Random Number Algorithm”, In: Byte Magazine, March, pp. 127-8.



Sessão Técnica 7

Comunicação sem Fio I

Qualidade de Serviço em redes sem fios

Celso A Pelliccioli¹, Cristina Moreira Nunes¹

¹Centro Universitário La Salle – Av. Victor Barreto, 2288 – Canoas – RS – Brazil

pelliccioli@cdl-poa.com.br, nunes@lasalle.tche.br

Resumo. Este artigo descreve como é feito o controle de acesso ao meio pela camada MAC (Media Access Control) no padrão de rede sem fio IEEE 802.11. Além disso, apresenta um dos principais problemas desse padrão que é a prestação de Qualidade de Serviço. Redes 802.11 utilizam o conceito de melhor esforço no tratamento do tráfego. A importância com Qualidade de Serviço torna-se cada vez mais prioritária, na medida que aplicações exigem uma grande parte de recursos da rede como banda, atrasos, latência e disponibilidade. O artigo aborda técnicas de controle do MAC para obter a diferenciação de serviço um dos requisitos básicos de Qualidade de Serviço.

1. Introdução

Hoje redes sem fio tornaram-se cada vez mais presentes no nosso dia a dia, principalmente em ambientes de trabalhos ou até mesmo em redes domésticas. A simplicidade e o desaparecimento do cabeamento aumentaram consideravelmente a criação de WLANs (*Wireless Local Área Networks*).

Foi pensando nisso que o IEEE (*Institute of Electrical and Electronics Engineers*) desenvolveu o padrão 802.11. O padrão 802.11 é responsável em definir regras para controlar o acesso ao meio da camada MAC (*Médium Access Control*) e aceitar diversos padrões do meio físico na transmissão sem fio.

Apesar do crescimento de redes WLANs a especificação 802.11 é muito limitada em prestar qualidade de serviço. (QoS – *Quality of Service*). A dificuldade em fornecer QoS em redes sem fio se deve aos seguintes fatores: mobilidade das estações, baixa vazão, atraso na troca de mensagens, taxa de transmissão sujeita a agentes externos, entre outros.

2. Camada MAC

Na camada MAC do padrão 802.11 são oferecidos dois tipos de mecanismos para o controle de acesso ao meio, como mencionado anteriormente, o mecanismo de acesso básico, chamado de DCF, e o mecanismo de acesso centralizado, chamado de PCF.

A função do modo PCF é controlar as transmissões de uma rede através de um ponto central, chamado PC (*Point Coordinator*), utilizando o serviço síncrono, já o modo DCF utiliza o serviço assíncrono, fornecendo o acesso ao meio utilizando o protocolo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) [Hayes 1999].

2.1 PCF

A função PCF é opcional e permite que as estações tenham prioridade no acesso ao meio. Essa prioridade é coordenada por uma estação chamada de PC (*Point Coordinator*) e,

geralmente, esta função é designada ao *Access Point* de uma rede BSS (*Basic Service Set*). No modo PCF o ponto de coordenação controla o acesso ao meio determinando qual estação deve transmitir através de consultas a estações relacionadas. Esse processo é chamado de *pooling*, sendo que o ponto de coordenação divide o tempo em períodos chamados de superquadros. Um superquadro é formado por um período livre de contenção (CFP - *Contention Free Period*) que é controlado pelo PCF, onde o acesso ao meio é ordenado e livre de colisões. O PC segue o modelo de consulta utilizando o algoritmo *round-robin*, já o segundo período é controlado pelo DCF e é chamado de CP (*Contention Period*), onde ocorre a concorrência pelo meio de acesso.

O controle de acesso funciona da seguinte forma: o ponto de coordenação envia um sinal chamado de *beacon*. *Beacon* é um sinal transmitido para as estações informando o inicio do período livre de contenção (CFP). O PCF tem uma prioridade mais alta do que o DCF, isto porque ele tem um período de tempo menor em relação ao DIFS, chamado de PIFS (*Point Interframe Space*). DIFS e PIFS são períodos de tempos utilizados para transmissões dos quadros, por isso nenhuma estação pode enviar antes do ponto de coordenação. As estações, ao receberem o *beacon*, atualizam seu vetor, chamado de NAV (*Network Allocation Vector*), com a informação de duração máxima do período livre de contenção (*CFP_Max_Duration*). Durante o período livre de contenção, o *Access Point* interroga as estações enviando um quadro de consulta para as estações cadastradas em sua tabela, se alguma estação deseja transmitir é enviado um quadro de ACK para o *Access Point*, mas antes de transmitir o ACK a estação deve esperar o tempo de período SIFS. Para qualquer transmissão, a estação deve aguardar um período de tempo SIFS (*Short Interframe Space*). Logo após o término da consulta na tabela, o *Access Point* reinicia a pesquisa, após esperar um tempo PIFS.

No modo PCF as transmissões são encerradas com um envio de um quadro *CF-End* pelo *Access Point*, passando do modo PCF para o DCF e voltando a utilizar o modo de contenção, sendo assim, o PCF volta a operar no próximo período de contenção livre.

2.2 DCF

O mecanismo DCF emprega o protocolo CSMA/CA este protocolo tem a implementação obrigatória, existindo um opcional que utiliza quadros de controle chamados RTS/CTS (*Request to Send/Clear to Send*), seu uso é opcional, mas deve ser implementada a sua função.

O protocolo CSMA/CA funciona da seguinte forma: antes de uma estação começar a transmitir, ela verifica se existe uma outra estação transmitindo no meio de comunicação. Se a estação detectar que o meio está inativo por um determinado período de tempo superior a DIFS, espaço distribuído entre os quadros, o quadro é transmitido. A estação receptadora, logo após o recebimento, envia um quadro de confirmação (ACK) num período de tempo SIFS, bem menor que o DIFS [MELO 2003]. Somente após o recebimento do ACK, a estação de origem assume que quadro chegou corretamente. Caso o ACK não chegue ao seu destino no intervalo de tempo esperado, significa que ocorreu uma colisão ou o algoritmo de CRC (*Cyclic Redundancy Checks*) detectou um erro no quadro.

Quando a estação deseja transmitir e o meio está ocupado ou a estação não recebeu o ACK de confirmação, a estação aciona um temporizador chamado de *backoff*, o qual funciona da seguinte forma: a estação calcula um valor aleatório no intervalo de 0 a

CW (*Contention Window* – CW_{min} e CW_{max}). Esse intervalo é calculado usando a seguinte expressão:

$$T_{\text{backoff}} = \text{Rand}(0, CW) * T_{\text{slot}}$$

Toda vez que o meio estiver inativo o *backoff* é decrementado, se a portadora detectar uma transmissão o *backoff* é parado naquele instante conservando o seu valor atual, o *backoff* é reiniciado no momento que o meio estiver inativo novamente, após o término do cálculo do *backoff* a estação pode transmitir. Estações que estão esperando mais de uma vez, ciclo de *backoff*, tem a vantagem em relação a estações que venham querer transmitir. Para reduzir a probabilidade de uma transmissão não sucedida o valor da janela de contenção é duplicado até atingir o valor máximo de CW_{max} . Caso o número máximo de tentativas (sete) de transmissões seja alcançado, o pacote é descartado e a estação inicia um novo *backoff* [Mello 2003].

Utilizando os quadros de controle. Uma estação deseja transmitir um quadro de dados, a estação tem a opção de transmitir um RTS, O quadro RTS tem a finalidade de reservar o meio, Os quadros RTS e CTS incluem informações da duração da transmissão e o tempo necessário da resposta correspondente do ACK [Rubstein 2003]. As demais estações utilizam estas informações contidas para atualizar o seu NAV, usando o NAV a estação sabe quando uma transmissão termina. Quando uma estação deseja transmitir um quadro de dados ela espera um tempo DIFS e envia um RTS este não possui nenhuma prioridade em relação as demais transmissões, a estação de destino recebe o RTS e envia o quadro CTS, após o meio estiver livre num tempo SIFS. As estações vizinhas recebem o CTS e ou RTS e atualizam o seu NAV, as estações terão que aguardar para acessar o meio conforme as informações contidas no seu vetor (NAV). A estação transmissora recebe o CTS e aguarda o meio estar livre e só então envia o seu quadro de dados aguardando um ACK de confirmação conforme o modelo básico do DCF. A utilização de quadros de controles RTS/CTS é usada para evitar terminais escondidos. Quando a estação A está transmitindo para B, porém C deseja transmitir para B mesmo que A esteja transmitindo, C acha que o meio está livre, o que resulta em colisão.

3. Qualidade de Serviço em redes WLANs IEEE802.11

A Qualidade de Serviço é definida pela ISO (*International Standard Organization*) como o efeito coletivo do desempenho de um serviço, o qual determina o grau de satisfação de um usuário. A Qualidade de Serviço em uma rede propõe a garantia do desempenho, conforme a necessidade das aplicações exigidas na transmissão. Essa necessidade pode ser baseada em reservas de recursos ou por classes de serviços diferenciados.

O fornecimento de QoS em redes sem fio difere de uma rede com fio. Existem alguns problemas nessas redes que exigem uma maior reserva de recursos da rede. Como problemas, pode-se citar: mobilidade, taxa de transmissão mais baixa, condições atmosférica, injustiça no compartilhamento de banda, instabilidade provocada pelo protocolo TCP, entre outros.

Em redes sem fio, o conceito de QoS é mais amplo. O tratamento é feito na diferenciação do acesso ao meio, ou seja, a estação tem uma maior probabilidade de acessar o meio utilizando classes de prioridades. Algumas propostas para implementar a diferenciação de serviço entre estações e classes de tráfego descrevem como podem ser efetuadas as modificações necessárias para prover a diferenciação através dos métodos DCF e PCF.

3.1 Diferenciação de Serviço

A diferenciação de serviço no modo DCF e PCF torna-se possível através de alterações de alguns parâmetros do funcionamento do protocolo MAC.

A diferenciação no modo PCF aborda como deve ser feito o escalonamento das consultas as estações. De acordo com [Caeiro 2002], o *pooling*, em vez de utilizar a técnica de escalonamento justo como *Round-Robin*, deve utilizar o WFQ (*Weighted Fair Queueing*), onde a criação de filas é baseada em classes de serviços por prioridades, e cada fila utiliza a técnica *Round-Robin*. Já em [Rubstein 2002] e [Rezende 2002], os autores propõem uma classificação na consulta por prioridade ou por alocação de *slots* de tempo. Como no modo livre de contenção (PCF) exige a utilização de um ponto de acesso (PC) a diferenciação de serviço é aplicado neste ponto. Na consulta por prioridade os autores propõem utilizar um mecanismo de consulta que leva em conta a prioridade da estação. Na técnica de alocação de *slots*, o ponto de acesso distribui os *slots* de tempo conforme a prioridade da estação.

Algumas técnicas de diferenciação de serviços propostas no modo DCF aplicam-se na variação de parâmetros no DIFS, *Backoff* e variação do tamanho de quadros.

Conforme [Melo 2003], a técnica de variação do tamanho no DIFS é definida na variação do parâmetro de tempo. O modo DCF libera o acesso ao meio conforme o tempo do DIFS. Uma estação com um DIFS maior terá uma baixa prioridade na hora de transmissão, em contrapartida, uma estação com um DIFS menor terá maior probabilidade de encontrar o meio livre na hora de sua transmissão. O DIFS é definido da seguinte forma:

$$\text{DIFS} = \text{SIFS} + N * \text{SlotTime}$$

Na fórmula acima, a variável N é o parâmetro que deve ser alterado para alcançar a diferenciação. De acordo com o padrão 802.11, o valor N será igual a 2, sendo que os valores de N não podem ser 0 ou 1, pois o DIFS teria um mesmo valor de um SIFS ou PIFS, causando falha no suporte ao MAC.

A diminuição do período aleatório do *Backoff* também influencia no tempo em que uma estação deve esperar antes de transmitir. A diferenciação é obtida através da mudança dos parâmetros do CW. Essa diferenciação está na escolha de valores num intervalo distinto, para o tamanho da janela CW, [MELO 2003]. Como exemplo, pode-se ter uma transmissão com prioridade alta, $CW_{min} = 21$, e outra estação que com prioridade mais baixa, $CW_{min} = 54$. A estação com baixa prioridade irá esperar mais tempo para acessar o meio, enquanto a estação com prioridade alta irá escolher um intervalo entre 0 e 21, reduzindo assim o tempo gasto com o *backoff* e um número mínimo de *slots*. Além disso, pode ocorrer que a estação com o intervalo entre 0 e 21 escolha aleatoriamente o valor 7 e a estação com intervalo 0 e 54 escolha 3. Desta forma, a estação de baixa prioridade terá a probabilidade de acessar o meio mais rápido.

Neste trabalho, foram estudadas várias técnicas utilizadas na diferenciação de serviços. Conforme [Vaidya 2000], o intervalo do *backoff* é calculado proporcionalmente ao tamanho do quadro a ser enviado e inversamente proporcional ao peso atribuído ao fluxo ao qual pertence o quadro. Com isso, o cálculo empregado utiliza um *backoff* menor já que o quadro a ser transmitido tem um tamanho maior. Já [Aad 2001] e [Cast 2001], aborda uma proposta diferente para o cálculo do *backoff*. Neste caso, o *backoff* é definido conforme a prioridade da transmissão, sendo escolhidos os intervalos da janela de contenção.

Uma outra técnica é a de diferenciação através do tamanho do quadro, que tem como característica o aumento do tamanho do quadro de dados na hora da transmissão. Uma estação, após ganhar o acesso ao meio, transmite seus dados com quadros de maior tamanho ganhando, assim, mais tempo de acesso ao meio. Esta técnica tem alguns problemas. Um deles é quando o meio produz taxas elevadas de erro. Outro problema está na utilização de aplicações em tempo real, já que este recurso utiliza, justamente, quadros de tamanhos menores para reduzir o retardo. Além disso, quanto maior o quadro enviado, maior será a probabilidade de fragmentação do mesmo, tornando a transmissão mais lenta.

4. Conclusões e Trabalhos Futuros

Em redes sem fios que utilizam o padrão 802.11 é adotada a filosofia de transmissão de melhor esforço, ou seja, cada estação recebe o mesmo tratamento na transmissão de seus dados e compartilha a largura de banda com todos os fluxos de dados das demais estações da rede. Este é o grande problema encontrado nas redes 802.11, não há garantia de que o serviço realizado com sucesso, nem mesmo de desempenho, mas algumas aplicações exigem este comprometimento e tais garantias na transmissão. Por isso é fundamental prover QoS em rede sem fios.

Foram apresentadas algumas propostas que podem ser utilizadas para prover qualidade de serviço em uma rede 802.11. Foi concluído que controlar o acesso ao meio é uma forma de se obter a diferenciação de serviço e com isso implantar prioridades para estabelecer QoS.

Como trabalhos futuros pretende-se explorar as técnicas estudadas no artigo para atingir a diferenciação de serviço, através de simulações. Com base na utilização do *Network Simulator* pretende-se criar um modelo de simulação entre duas redes sem fios. Este modelo deve propor um ambiente o mais próximo possível da realidade. Para atingir a validação das técnicas empregadas na simulação será necessário criar dois ambientes paralelos. Um propondo atingir a diferenciação, aplicando as técnicas de controle ao acesso ao meio e outro com um ambiente sem nenhuma alteração. Deve ser levar em conta os números de estações da rede, bem como o período de tempo, velocidade do link entre as estações, o tipo de tráfego gerado, se ocorrer mobilidade entre as estações de cada rede, ambas as redes devem conter as mesmas coordenadas. O resultado final fornecido pelo simulador entre as simulações deve ser analisado e avaliado.

5. Referências Bibliográficas

- Aad. I., Castelluccia. C. (2001) “Differentiation mechanisms for IEEE 802.11”, disponível em <http://www.inrialpes.fr/planete/people/aad/infocom2001.pdf>.
- Anastasi. G., De Stefano. E., Lenzini. L. (2000) “QoS Provided by the IEEE 802.11 Wireless LAN to Advanced Data Applications: a Simulation Analysis”, disponível em <http://portal.acm.org/citation.cfm?id=340910&dl=ACM&coll=portal>.
- Battiti, R., Li B. (2003) “Supporting Service Differentiation with Enhancements of The IEEE 802.11 MAC Protocol: Models and Analysis”, disponível em <http://citeseer.ist.psu.edu/battiti03supporting.html>.
- Caeiro, L., Rocha, R. (2002) “QoS em redes sem fios: estado da arte e novas soluções”, disponível em www.fccn.pt/crc2002/v2/ppts/dia27/sessao5/luisa_caeiro.ppt.

- Chang, M.C., Lee Kang-Won. (2002) "Wireless QoS Analysis for a Rayleigh Fading Channel", disponível em <http://www-tkn.ee.tu-berlin.de/~aaguiar/wjc/icc98.3.ps>.
- Gaertner, G. Cahill, V. (2004) "Understating Link Quality in 802.11 Mobile Ad Hoc Networks", disponível em <http://dsonline.computer.org/0401/f/w1stop.htm>.
- Garg, P. et al . (2003) "Using IEEE 802.11e MAC for QoS over Wireless", disponível em <http://mosquitonet.stanford.edu/software/802.11e/ipc84.pdf>.
- Gast, M. (2002) "802.11 Wireless Networks:The Definitive Guide", Editado por Mike Loukides.
- Hayes, Vic., et al (1999) " Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: Higher-speed physical layer extension in the 2.4 GHz band", disponível em <http://standards.ieee.org/wireless/>.
- ISI (2004) "The Network Simulator –ns-2", disponível em <http://www.isi.edu/nsman/ns/>.
- Leon-Garcia, A, Widjaja I.(1999) "Communication Networks", Editado por Sue Culbertson.
- Melo Filho, J.C. (2003) "Mecanismo de Controle de Qualidade de Serviço em Redes IEEE 802.11", disponível em <http://www.gta.ufrj.br/ftp/gta/TechReports/Coelho03/Coelho03.pdf>.
- Melo Filho, J.C., Rezende, J.F., Pirmez, L. (2003) "Seleção Dinâmica de Parâmetros de Controle de Qualidade de Serviço em Redes IEEE 802.11 Infra-Estruturadas", disponível em <http://www.gta.ufrj.br/ftp/gta/TechReports/FRPS03.pdf>.
- Nedeltchev, P. (2001) "Wireless Local Área Networks and the 802.11 Standard", disponível em <http://www.cisco.com/warp/public/784/packet/jul01/pdfs/whitepaper.pdf>.
- Rubstein, M., Rezende, J.F. (2002) "Qualidade de Serviço no Controle de Acesso ao Meio de Redes 802.11", Workshop em Qualidade de Serviço e Mobilidade 2002.
- Silva, A., et al. (2003) "Redes Wireless Bluetooth", Anais da Escola Regional Redes Computadores, 2003.
- Srikant, S. (2003) "Analysis of 802.11b MAC: A QoS, Fairness, and Performance Perspective", disponível em <http://www.ecsl.cs.sunysb.edu/~srikant/>.
- Tanenbaum, A. (1997) "Redes de Computadores", Tradução da 3ª edição, Rio de Janeiro, Campus.
- Truong, H.L., Vannuccini G. (2003) "The IEEE 802.11e MAC for Quality of Service in Wireless LANs", disponível em <http://citeseer.ist.psu.edu/554451.html>.
- Vaidya, N., Bahl, P., Gupta, S. (2000) "Distributed Fair Scheduling in a Wireless LAN", disponível em <http://portal.acm.org/citation.cfm?id=345939&dl=ACM&coll=portal&CFID=11111111&CFTOKEN=2222222>.

SERVIDORES MÓVEIS EM REDES *AD HOC*

Fabricia Faria, Vitório Mazzola, M. A. R. Dantas

Departamento de Informática e Estatística – INE
Universidade Federal de Santa Catarina
Trindade – Florianópolis – SC – Brasil
{flemos, mazzola, mario}@inf.ufsc.br

Resumo. *Redes móveis Ad hoc são caracterizadas por sua formação temporária e dinâmica, onde serviços e dispositivos podem estar disponíveis ou não. Este trabalho apresenta um modelo que possibilita a execução de web services em dispositivos móveis wireless. Nossa objetivo é tornar possível a qualquer dispositivo em uma rede móvel Ad hoc oferecer um conjunto de serviços a qualquer outro dispositivo na mesma rede. Nossa contribuição busca permitir que usuários de redes móveis Ad hoc tenham acesso a alguns serviços que são, atualmente, só disponíveis a usuários de redes infra-estruturadas.*

1. INTRODUÇÃO

A tecnologia de *web services* vêm sendo muito pesquisada e várias propostas e padrões surgiram com o intuito de fornecer serviços que fossem executados sem a intervenção do usuário, ou seja, de aplicação para aplicação, [Zhu 2003], [Tsai 2003], [Benatallah 2004].

O objetivo principal desta pesquisa é disponibilizar a tecnologia de *web services* aos usuários de dispositivos móveis, que tiveram nos últimos tempos, suas capacidades computacionais aumentadas, possibilitando com isso a execução de um número cada vez maior de serviços.

Para atingir este objetivo, é apresentada neste trabalho uma proposta de modelo para a execução de *web services* em dispositivos móveis sem fio, de modo que o próprio dispositivo seja o fornecedor do serviço para a rede móvel *Ad hoc*. Neste cenário, foram identificados diversos problemas, dentre eles a questão da mobilidade, as limitações de energia e de largura de banda quando comparada às redes fixas. Embora, não sejam estes os únicos problemas, o trabalho objetiva tratar prioritariamente os anteriormente mencionados.

Web services utilizados com a computação móvel *wireless* resultará em ganhos significativos para consumidores e fornecedores de aplicações, pois permitirá aos usuários dessas redes, consumirem uma gama de aplicações disponíveis hoje, somente aos usuários das redes infra-estruturadas.

Este artigo está organizado conforme descrito a seguir. A seção 2 apresenta os conceitos relacionados à tecnologia de *web services*. A seção 3 introduz as redes móveis *Ad hoc*. A seção 4 define o modelo proposto. Finalmente, a seção 5 conclui o trabalho e sugere alguns tópicos a serem explorados na continuidade desta pesquisa.

2. WEB SERVICES

Web Services [W3C A] surgiram com o intuito de tornar possível a interação aplicação – aplicação sobre a rede, utilizando para isso padrões abertos como SOAP (*Simple Object*

Access Protocol) [W3C C], UDDI (*Universal Description Discovery & Integration*) [OASIS], WSDL (*Web Service Definition Language*) [W3C B], todos estes com base na linguagem XML (*Extensible Markup Language*) [W3C D].

Um *web service* será acessado através da *web*, utilizando um documento WSDL contendo todas as informações necessárias para a utilização do serviço, como por exemplo os protocolos de transporte, o formato das mensagens, entre outros dados. Este documento fica publicado em um diretório central, conhecido como UDDI, que fornece um mecanismo padrão capaz de classificar os *web services* disponíveis de modo que este serviço possa ser localizado e utilizado. Na abordagem dos autores [Milenkovic 2003], cada dispositivo em um ambiente *wireless* possui um diretório UDDI que contém os serviços disponíveis.

XML é a chave principal da tecnologia de *web services*, permitindo a troca de dados em ambientes heterogêneos. Mensagens XML são empacotadas e trocadas através do protocolo SOAP. Mensagens SOAP podem ser transmitidas através de diversos protocolos de rede.

Considerando que esta pesquisa trata de *wireless web services*, levamos em conta as pesquisas realizadas por [Yuan 2002], que sugere a adoção de uma das seguintes arquiteturas:

- § **Wireless portal network:** nesta arquitetura a comunicação se dá via um *gateway WAP*. O portal age como um *proxy* para os clientes *wireless* não havendo a necessidade de se armazenar código no dispositivo móvel.
- § **Wireless extended Internet:** esta arquitetura é descentralizada, ou seja, não necessita de uma entidade de controle, mas *hubs web services* centralizados são requeridos. Como foi descrito na seção 4, [Steele 2003] adotou esta arquitetura em seu trabalho.
- § **Wireless Ad hoc network:** Esta arquitetura permite que os dispositivos *wireless* tornem-se servidores para seus *peers*. *Wireless peers* podem fornecer conteúdo, roteamento do tráfego na rede, e muitos outros serviços.

Para o desenvolvimento de *web services*, duas plataformas de desenvolvimento, J2EE e .NET estão competindo para se tornarem padrão para o desenvolvimento destes serviços, [Chung 2003].

Existem várias pesquisas na área de *web services*, com o intuito de resolver os problemas ainda existentes na tecnologia como composição de serviços, agregação de serviços, segurança, qualidade de serviço (QoS), entre outras.

3. REDES MÓVEIS AD HOC

As *MANETs* (*Mobile Ad hoc Network*) operam em um ambiente móvel *wireless*, incorporando funcionalidades de roteamento nos próprios dispositivos móveis conhecidos como *mobile nodes*, [Corson and Macker 1999]. Devido à falta de infra-estrutura, cada nodo age como um roteador, transmitindo pacotes de dados para outros nodos, [Cao 2004].

A topologia é dinâmica pois os nodos em uma *MANET* estão em constante movimentação, embora sejam tipicamente *multihop*, [Corson and Macker 1999].

O problema quanto à limitação de energia, uma das principais preocupações neste ambiente, está sendo pesquisado nos algoritmos de roteamento que estão sendo desenvolvidos, [Mohapatra 2004], nas aplicações desenvolvidas para estas redes e nos próprios dispositivos que operam na rede, todos estes, buscando um único objetivo, consumirem cada vez menos energia.

O ambiente móvel possui alguns desafios e desvantagens que devem ser cuidadosamente observados e na medida do possível, tratados. Alguns exemplos são performance, custo, indisponibilidade temporária, baixa capacidade de processamento (CPU) e memória, pequena duração da bateria [Pilioura 2003], [Mohapatra 2004].

A figura 3.1 mostra um simples exemplo de uma *MANET*.

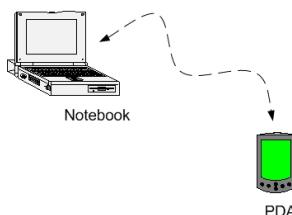


Figura 3.1 - Rede *Ad hoc*

Segundo [Corson and Macker 1999], dentre as várias características das *MANETs* podemos citar: topologia dinâmica, limitada largura de banda, variável capacidade dos links, limitação de energia e segurança física limitada.

Tratando-se dos métodos de comunicação, o modo de envio de pacotes *broadcast* é muito utilizado nas redes *Ad-hoc* e têm sido bastante pesquisado no intuito de minimizar alguns problemas como colisão, contenção e re-broadcast redundante, que aparecem quando utilizamos comunicação *broadcast*, [Zhu 2004].

4. TRABALHOS RELACIONADOS

Em seu trabalho [Friedman 2002] definiu o armazenamento de *web services* em redes móveis *Ad hoc* como uma maneira de tornar serviços mais acessíveis a dispositivos móveis. Neste trabalho, os métodos são executados localmente, reduzindo com isso a comunicação. Cada serviço deverá ter um único *proxy* dentro de cada *MANET* no qual o serviço está sendo usado. Outra consideração importante assumida pelo autor é que o primeiro nodo que acessar o serviço tornar-se-á o *proxy* para tal serviço.

A pesquisa do autor [Steele 2003] propõe um sistema baseado em *web services* que permite a integração dinâmica e temporária de módulos de software ou aplicações sobre dispositivos em *MANETS*. A principal diferença deste trabalho com o nosso é que ele assume a arquitetura *Wireless Extended Internet*, onde existe um servidor com quem os dispositivos móveis se comunicam.

Os autores [Yang 2003] descrevem uma estrutura para organizar e acessar *web services* eficientemente em ambientes *broadcast*. No modo *broadcast*, um *service broker* periodicamente transmite os serviços disponíveis sobre o canal *wireless*. Clientes escutam o canal, identificando o serviço de interesse e baixam o serviço para execução local.

A recente pesquisa dos autores [Cao 2004] define um framework para o acesso aos dados, permitindo que os nodos armazenem dados ou apenas o caminho para estes dados, com o objetivo de economizar espaço. Os nodos devem armazenar dados diferentes daqueles armazenados por seus vizinhos, aumentando assim, a disponibilidade a estes dados, embora isso, cause um maior atraso, pois os nodos devem acessar os dados de seus vizinhos no lugar de acessar esses dados localmente.

5. MODELO PROPOSTO

Utilizaremos a arquitetura *Wireless Ad hoc network*, ideal para o ambiente *Ad hoc* pois não utiliza nenhuma estrutura física para disponibilizar os serviços aos clientes da rede.

Tendo em vista que a capacidade dos dispositivos móveis vem aumentando consideravelmente e considerando que a perda da conexão nos ambientes móveis *wireless* é constante, o modelo proposto executará o *web service* localmente. A figura 5.1 ilustra o modelo de forma geral.

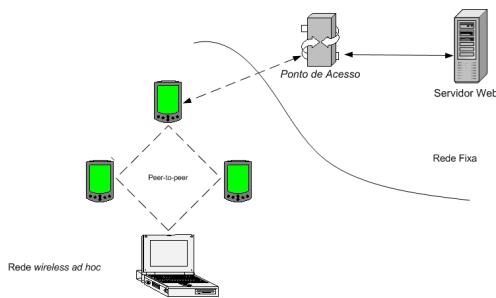


Figura 5.1 – Arquitetura geral do modelo

Uma vez implementado, o funcionamento do modelo proposto ocorrerá da seguinte maneira: um usuário portando um dispositivo móvel, se conectará ao servidor que estará na rede fixa para fazer *download* dos módulos móvel e fixo.

Quando o usuário do dispositivo móvel entrar na *MANET*, através de uma mensagem de *broadcast*, ele informará aos seus vizinhos, o serviço disponível. Chamaremos aqui este dispositivo de DMF (Dispositivo Móvel Fornecedor).

O modelo proposto terá um diretório UDDI que ficará no servidor. Quando um dispositivo cliente estiver interessado em consumir o serviço oferecido pelo DMF, ele estabelecerá uma conexão com o DMF para acessar o UDDI que terá as informações necessárias para consumir o serviço. Cada DMF deverá manter uma tabela com o endereço conhecido dos clientes, para que os clientes sejam notificados do novo endereço do DMF, caso haja perda de conexão.

O dispositivo cliente executará o serviço localmente conectando-se ao DMF para fazer o *download* do serviço, quando necessitar atualizar o serviço ou quando concluir a operação e desejar atualizar a base de dados do servidor.

A figura 5.2 apresenta a arquitetura do sistema um pouco mais detalhada.

Na rede fixa, encontra-se o servidor contendo o sistema como um todo, seus módulos móvel e fixo que serão utilizados pelos dispositivos na *MANET*.

Na rede móvel, como descrito anteriormente, primeiramente o DMF fará *download* dos módulos móvel e fixo e entrará na *MANET* tornando o serviço disponível aos dispositivos interessados em consumirem o serviço. Uma conexão será estabelecida e os módulos móvel e fixo serão baixados, por exemplo, para o dispositivo móvel A para execução local.

O sistema encontra-se em desenvolvimento, sendo que a plataforma adotada neste trabalho é a .Net. Partimos do princípio que os dispositivos móveis, ao entrarem na *MANET*, estarão equipados com a tecnologia necessária ao consumo deste serviço que são o .Net Compact Framework e o SQL Server CE.

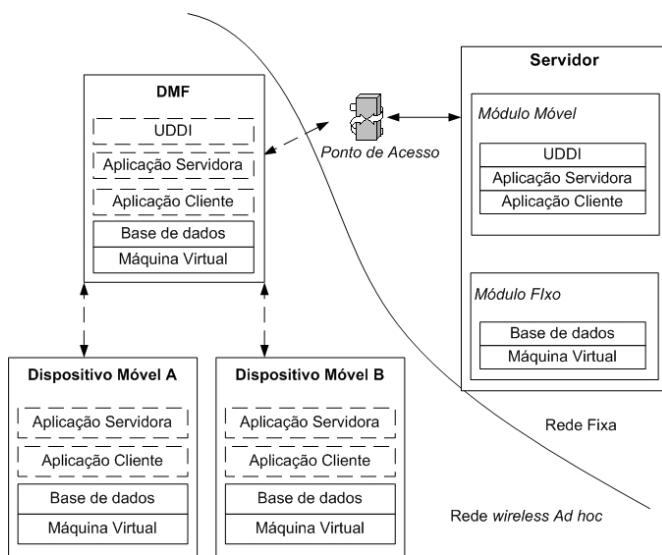


Figura 5.2– Arquitetura específica do sistema

É importante ressaltar aqui, que nossa arquitetura não estará presa à plataforma de desenvolvimento. Apesar desta implementação estar baseada na plataforma .NET da Microsoft, uma segunda implementação deverá ser desenvolvida com base na plataforma J2EE, da SUN. Isto permitirá obter resultados sobre a influência da plataforma de desenvolvimento no sistema como um todo.

6. CONCLUSÕES E TRABALHOS FUTUROS

Disponibilizar *web services* em *MANETs* é um importante desafio devido aos vários problemas encontrados nestes ambientes, dentre os principais, a questão da mobilidade, e as limitações de energia e de largura de banda. Embora estas redes apresentem ainda várias limitações, o que caracteriza um grande leque para a pesquisa, elas já estão sendo usadas principalmente em desastres, lugares remotos ou quando existe a necessidade de montar rapidamente um ambiente como no caso de um evento.

A principal contribuição do trabalho está na possibilidade de trocar dados sem a necessidade de infra-estrutura, contribuindo no sentido de disponibilizar aos usuários destas redes, serviços acessíveis a qualquer hora e em qualquer lugar, tornando-se um diferencial, uma vez que não se dispõe de serviços nestes ambientes.

O modelo proposto está em fase de implementação. Uma aplicação exemplo está sendo desenvolvida para utilizar um *web service* de uma livraria. Os testes iniciais serão realizados utilizando três dispositivos móveis que formarão a *MANET* e atuarão tanto como clientes como fornecedores do serviço. Nesta fase, as desvantagens do modelo proposto serão identificadas e alterações necessárias serão realizadas.

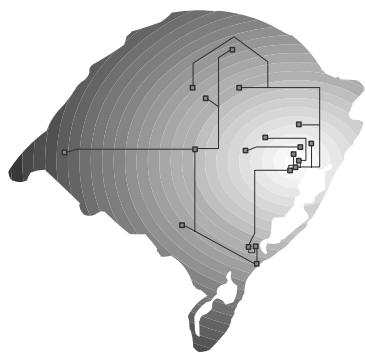
Algumas limitações do nosso trabalho que devem motivar trabalhos futuros incluem a questão da segurança e a garantia da qualidade de serviço (*QoS*). Os dois aspectos deverão ser contemplados futuramente no modelo.

AGRADECIMENTOS

Gostaríamos de agradecer à Flavia Delicato pela sua valiosa ajuda, sempre trocando idéias, tirando dúvidas e respondendo rapidamente aos diversos e-mails.

BIBLIOGRAFIA

- Benatallah, B., Casati, F., Toumani, F. (2004) "Web Service Conversation Modeling: a cornerstone for e-business automation", Internet Computing, IEEE, p. 46-54.
- Cao, G., Yin, L., Das, C. R. (2004) "Cooperative Cache-Based Data Access in Ad Hoc Networks" IEEE Computer, p. 32-39.
- Chung, J. Y., Lin, K. J., Mathieu, R. G. (2003) "Web Services Computing: Advancing Software Interoperability". IEEE Computer, p. 35-37.
- Corson, S. and Macker, J. (1999) "Mobile Ad hoc Networking (MANET)", Disponível on-line, acesso em 02/05/2004.
- Friedman, R. (2002) "Caching Web Services in Mobile Ad-Hoc Networks: Opportunities and Challenges", ACM Workshop On Principles Of Mobile Computing, p. 90-96.
- Milenkovic, M., Robinson, S.H., Knauerhase, R.C., Barkai, D., Garg, S., Tewari, A., Anderson, T.A., Bowman, M. (2003) "Toward Internet Distributed Computing", IEEE Computer, p. 38-46.
- Mohapatra, P., Gui, C., Li, J. (2004) "Group Communications in Mobile Ad Hoc Networks" IEEE Computer, p. 52-59.
- OASIS, UDDI, <http://www.uddi.org/>. Disponível on-line, acesso em 15/01/2004.
- Pilioura, T., Tsalgatidou, A., Hadjiefthymiades, S. (2003) "Scenarios of using Web Services in M-Commerce", ACM SIGecom Exchanges, p. 28-36.
- Steele, R. (2003) "A Web Services-based System for Ad-hoc Mobile Application Integration", Coding and Computing ITCC International Conference. IEEE, p. 248-252.
- Tsai, T-M., Yu, H-K., Liao, P-Y., Shih, H-T. (2003) "Semantic Modeling among Web Services Interfaces for Services Integration – SOTA (Smart Office Task Automation) platform", Proceedings of the 14th International Workshop on Database and Expert Systems Applications IEEE, p. 579-583.
- W3C (2004 A) Web Services Technical Report, <http://www.w3.org/2002/ws>. Disponível on-line, acesso em 15/01/2004.
- W3C (2004 B) WSDL 2.0 Technical Report, <http://www.w3.org/TR/wsdl20/>. Disponível on-line, acesso em 15/02/2004.
- W3C (2004 C) SOAP 1.2 Technical Report, <http://www.w3.org/TR/soap12-part1/>. Disponível on-line, acesso em 15/01/2004.
- W3C (2004 D), XML Technical Report, <http://www.w3.org/TR/2004/REC-xml-20040204/>. Disponível on-line, acesso em 20/04/2004.
- Yang, X., Bouguettaya, A., Medjahed, B., Long H., He W. (2003) "Organizing and Accessing Web Services on Air", Systems, Man and Cybernetics, IEEE Transactions, p. 742-757.
- Yuan, M. J. and Long, J. (2002) "Java Readies Itself for Wireless Web Services", <http://www.javaworld.com/javaworld/jw-06-2002/jw-0621-wireless.html>, Disponível on-line, Acesso em 18/05/2003.
- Zhu, C., Lee, M.J., Saadawi, T. (2004) "A Border-aware Broadcast Scheme for Wireless Ad Hoc Network" Consumer Communications and Networking Conference First IEEE, p. 134-139.
- Zhu, J. (2003) "Web Services Provide the Power to Integrate" Power and Energy Magazine, IEEE, p. 40-49.



Sessão Técnica 8

Roteamento

SIM Gateway: roteador para redes sem fios baseado em FreeBSD

Carlos Oberdan Rolim¹, Carlos A. M. dos Santos²

¹ SIM Telecom – Rua Imigrantes, 500 — 98800-000 Santo Ângelo, RS

²Universidade Regional Integrada do Alto Uruguai e das Missões — URI
Av. Universidade das Missões, 464 — 98802-470 Santo Ângelo, RS

ober@sol.psi.br, casantos@urisan.tche.br

Resumo. Este artigo descreve a implementação de um roteador de baixo custo destinado a prover serviços de interconexão entre diversas empresas através de uma rede sem fios. O sistema é baseado em uma versão reduzida do sistema operacional FreeBSD, carregada a partir de um flash disk, e oferece uma interface para fácil configuração via WWW.

1. Introdução

Redes baseadas em cabeamento têm se mostrado inadequadas para atender locais de difícil acesso, devido aos altos custos de instalação e manutenção. Em função disto, há uma demanda crescente por conexões sem fio, que independem de cabos ou da infra-estrutura provida pelas operadoras de telefonia. Para atender a essa demanda os provedores de acesso necessitam de equipamentos adequados. Este artigo descreve um roteador para redes sem fio que foi desenvolvido para ser utilizado comercialmente por uma empresa de telecomunicações.

O texto a seguir está organizado da seguinte forma: a seção 2 discute alguns conceitos relevantes de redes wireless; a seção 3 descreve a proposta inicial do projeto, os serviços fornecidos pelo roteador, os principais problemas encontrados e as soluções adotadas, e a administração do sistema; a seção 4 descreve as modificações e melhorias feitas a partir da versão inicial; na seção 5, por fim, são apresentadas as conclusões e as perspectivas para trabalhos futuros.

2. Conceitos básicos

As redes sem fio mais populares atualmente seguem o padrão IEEE 802.11b e usam portadora com frequência de 2,4GHz. Devido à saturação da faixa espectral, algumas redes estão migrando para a faixa dos 5,8GHz, mas o custo dos equipamentos que operam nessa faixa ainda é muito elevado (tipicamente o dobro do preço).

Há muitos aspectos comuns entre as redes 802.11b e 802.3; a diferença fundamental é o modo como se controla a comunicação entre as estações. O modo mais simples é o Independent Basic Service Set (IBSS), também chamado de *Ad-Hoc*, no qual cada máquina de um grupo pode se comunicar com outra máquina do mesmo grupo de forma independente. IBSS é mais utilizado em conexões ponto-a-ponto ou em ambientes livres de obstáculos que obstruam a comunicação.

Já no modo Basic Service Set (BSS) um equipamento chamado Access Point (AP) atua como intermediário, fazendo o papel de ponte (*bridge*) entre as máquinas conectadas

Tabela 1: Sensibilidade dos cartões

Velocidade	Sensibilidade	
	PRISM II	Lucent
1MBit/s	-91dBm	-94dBm
2MBit/s	-88dBm	-91dBm
5.5MBit/s	-87dBm	-87dBm
11MBit/s	-82dBm	-84dBm

Fonte: [Intersil, 2003, ORiNOCO, 2003]

via rádio e as conectadas via cabo [Flickenger, 2002]. Quando uma máquina precisa se comunicar com outra ela manda os pacotes para o Access Point e este os encaminha para a estação destino apropriada. Para usar os serviços de uma rede 802.11b uma estação deve fazer a *associação* a um Access Point.

3. Solução desenvolvida

3.1. Proposta inicial

As interfaces de rádio 802.11b em geral vêm embutidas em cartões PCMCIA. Todos eles podem ser configurados para operar tanto em modo IBSS quanto BSS, mas a maioria dos “drivers” não permite configurá-los para trabalhar como AP. Com o driver *wi* do sistema operacional FreeBSD, entretanto, cartões com *chipset* PRISM II (Intersil) podem ser usados com esta finalidade.

A idéia inicial do trabalho foi, utilizando-se um PC e um cartão de rádio, construir um AP com as mesmas funcionalidades de um equipamento comercial, mas a um custo inferior. Ao comparar cartões com chipsets Hermes (Lucent) e PRISM II, entretanto, descobriu-se que este possui características que o tornam inadequado para operar como AP em ambientes abertos. Equipamentos com esse tipo de cartão possuem uma potência exagerada, que polui com ruído desnecessário a faixa espectral à sua volta, e baixa sensibilidade, fator extremamente importante para que um AP possa “ouvir” os clientes que desejam associarem-se à rede.

A tabela 1 demonstra a sensibilidade dos diferentes cartões. Um AP com um cartão PRISM II se comportaria como uma pessoa que gritasse o tempo todo e fosse surda ao mesmo tempo. Decidiu-se então direcionar o projeto para a construção de um roteador, o que atenderia a uma outra necessidade do mercado, já que os clientes BSS, em sua maioria, possuem capacidade muito limitada para operar como roteadores.

3.2. Requisitos e serviços básicos a serem prestados

DHCP (Dynamic Host Configuration Protocol) Em redes sem fio o DHCP é extremamente útil devido à facilidade com que o usuário passa a fazer parte da rede. O roteador é capaz de trabalhar tanto como servidor DHCP quanto como cliente.

DNS (Domain Name Service) Inicialmente considerou-se desnecessário prover um servidor de nomes. O sistema somente seria cliente DNS dos servidores já existentes na rede.

NAT (Network Address Translation) O sistema possui suporte a NAT 1:N e NAT 1:1.

Roteamento O ideal seria prover roteamento dinâmico, mas o *daemon* de roteamento do FreeBSD (routed) suporta apenas o protocolo RIP, cujo uso é impraticável em redes sem fio, devido à quantidade de mensagens de atualização de rotas enviadas aos pontos da rede. No momento está sendo usado apenas roteamento estático.

Filtro de Pacotes É um mecanismo essencial para a segurança da rede. O sistema provê filtragem como base em tipo de protocolo, tipo de pacote, endereço de origem e endereço de destino.

Modelagem de Tráfego (Traffic Shaping) Esse serviço é cada vez mais necessário nas redes atuais, pois possibilita limitar a parcela da banda consumida por cada um dos pontos de presença da rede.

SNMP Um agente SNMP permite coletar dados estatísticos da rede. Os administradores de rede necessitam desses dados, em especial o tráfego de pacotes, para otimizarem o funcionamento da rede. Foi instalada uma versão reduzida do net-snmp¹ somente para consulta de informações.

VPN (Virtual Private Network) permite trafegar informações de forma segura passando por redes não confiáveis. Esse serviço era necessário para que alguns usuários conectassem suas redes locais a redes remotas para compartilhamento de recursos.

3.3. Sistema operacional

Desde o inicio foi procurado um sistema operacional que permitisse o gerenciamento e monitoramento remoto e que, acima de tudo, suportasse os principais cartões wireless do mercado. Deveria também ser robusto ao ponto de não corromper o seu sistema de arquivos em caso de queda de energia e poder ser recuperado rapidamente em caso de um desastre maior.

Optou-se pelo FreeBSD², por sua flexibilidade, estabilidade e segurança. FreeBSD é derivado do BSD4.4 (da universidade da Califórnia, em Berkley) e é desenvolvido desde 1993 em um trabalho coletivo feito através da Internet. Por ser um software de código aberto, e de grande qualidade, revelou-se vantajoso em relação aos sistemas operacionais comerciais. Ao contrário do Linux, outro software muito popular, o FreeBSD é um *sistema operacional* e não um kernel com outras coisas por cima que o fazem, então, uma “distribuição”.

3.4. Servidor sem disco rígido, monitor ou teclado

Em caso de uma queda de energia o sistema operacional testa o sistema de arquivos no próximo *boot*, identificando e corrigindo os erros. Em casos críticos, porém, é necessária a execução manual de comandos para o reparo. Isto não é desejável em um produto sem teclado nem monitor, e que muitas vezes ficará em locais de difícil acesso. A solução foi montar o sistema de arquivos em modo *read-only*. Dessa forma ele permaneceria intacto mesmo ocorrendo uma queda de energia ou outro tipo de problema no sistema.

O ideal seria não depender de meios mecânicos para armazenamento como disquete, disco rígido ou CD-ROM. Adotou-se então um *flash disk*, reconhecido pelo BIOS da máquina como um disco padrão IDE. Há flash disks de 16MB, 32MB, 64MB e maiores, com preços igualmente crescentes. Uma unidade de 32MB ficou num patamar aceitável de preço e tamanho para o projeto, mas não comportava sequer a instalação mínima do FreeBSD.

A solução foi criar um FreeBSD personalizado, que ocupa apenas 12MB. Para isto foi gerado um *kernel* enxuto, que ocupa apenas 1,1MB. Os dispositivos do diretório /dev foram compactados também. No momento do boot do sistema o kernel e o diretório /dev são descompactados e colocados em um *ram disk*. O diretório /var também é montado em memória (os arquivos de registro do sistema são perdidos na hora de reinicialização). Obteve-se assim um sistema enxuto, que utiliza um meio de armazenamento quase imune a falhas a um custo baixo se comparado aos similares comerciais.

¹<http://net-snmp.sourceforge.net/>

²<http://www.FreeBSD.org>

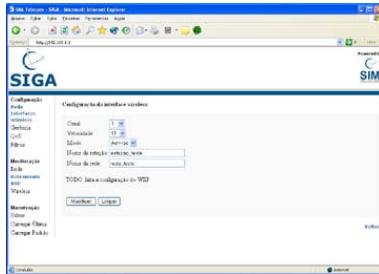


Figura 1: Interface de configuração via WWW

O problema do acesso ao sistema foi resolvido usando a porta serial como console. Quando necessário, basta conectar um terminal (ou um PC rodando um emulador de terminal).

3.5. Gerenciamento

A configuração do roteador pode ser feita de forma manual. Basta usar um terminal, ou iniciar uma sessão remota via SSH, e então editar os arquivos necessários. Para um usuário experiente do FreeBSD essa tarefa é trivial, mas para os novatos é um tanto complexa. Como um das metas do projeto é desenvolver um produto de fácil configuração e gerenciamento, a alternativa foi desenvolver uma interface de gerenciamento amigável, para que mesmo pessoas não especializadas pudessem administrá-lo.

A primeira opção considerada foi desenvolver um agente SNMP, que ficaria no roteador, e uma ferramenta de administração, que ficaria da estação usada para gerenciar o sistema. Esta solução se revelou pouco flexível, pois exigiria que o programa de administração fosse instalado em cada estação. Observou-se que seria muito mais simples desenvolver uma aplicação que rodasse no roteador, desempenhando as tarefas de gerenciamento, e fosse acessível via WWW. O agente SNMP continuou a ser usado, mas apenas para coletar estatísticas de operação do roteador.

Usa-se um servidor HTTP capaz de autenticar usuários por sessão. Quando um usuário quer acessar a interface de administração, ele precisa fornecer um nome e uma senha válidos. A figura 1 mostra a tela de configuração de uma interface wireless. As configurações e informações são manipuladas através de scripts CGI escritos na linguagem de programação Perl (uma versão estática do interpretador que fornece todas as funções básicas em apenas 1.3MB). Apesar de a linguagem ser interpretada, não se observou qualquer problema de desempenho.

4. Evolução do sistema

O projeto foi colocado em operação em 25 provedores de acesso no Rio Grande do Sul em meados de 2003. Durante o período de aproximadamente um ano diversas correções de erros foram feitas e novas necessidades observadas. Melhorias acabaram sendo desenvolvidas para suprir essas necessidades. A seguir são descritas algumas dessas melhorias no sistema:

Suporte a PPPoE (PPP over Ethernet) Com a inclusão de suporte a PPPoE no kernel o sistema passou a funcionar como servidor PPPoE para clientes da LAN local, ao mesmo tempo em que pode trabalhar como cliente PPPoE autenticando-se em um servidor remoto. Os dados que trafegam na rede são transportados usando MPPE (Microsoft Point-To-Point Encryption Protocol) [Pall and Zorn, 2001] usando criptografia de 40 a 128 bits.

Suporte a PPTP O suporte a VPN inicialmente foi feito usando o aplicativo vtun³, que tem como vantagens a alta taxa de compactação obtida, o que resulta em grande desempenho no tráfego dos dados, e a facilidade de configuração. Entretanto o vtun usa seu próprio protocolo de comunicação sobre TCP e UDP e não suporta PPTP, L2TP ou IPsec. É necessário que tanto o lado cliente quanto o servidor rodem em plataforma Unix. Como era necessário fazer o sistema autenticar em servidores Windows, que suportam apenas PPTP e L2TP, substituiu-se o vtun pelo PPTP Client.

Suporte a IPsec no kernel para quando houver a necessidade de conexões com criptografia forte de 128 bits. A troca de chaves dinâmicas foi possibilitada devido ao uso do racoon⁴ para gerenciamento das mesmas.

Troca de pacotes de firewall e nat Os aplicativos IPFW e NATD existem desde a versão 2.0 do FreeBSD. Eles são eficientes, mas não oferecem alguns recursos necessários ao sistema, como por exemplo NAT para mais de um endereço IP e construção de regras de firewall mais flexíveis, robustas e enxutas. Dessa forma passou-se a usar o IPFilter em conjunto com o IPNat, disponíveis no FreeBSD versão 5, para fazer firewall e NAT. Isto tornou necessário atualizar a versão do sistema operacional utilizada.

Troca de *traffic shaper* Durante a transmissão de dados os pacotes que chegam ao sistema são colocados em uma fila de espera até serem tratados pelo sistema e então despachados pela interface adequada segundo critérios de roteamento. O algoritmo normalmente usado para despachar os pacotes nessa baseia-se em um FIFO (First In First Out). Como o sistema passou a ser utilizado para roteamento de VoIP (Voice over IP) surgiram os seguintes problemas:

latência O atraso geralmente é ocasionado pela concorrência entre os vários serviços pela banda de rede existente.

jitter É o resultado de pacotes que chegam ao destino em intervalos irregulares. Este problema afeta drasticamente a qualidade da transmissão de voz.

perda de pacotes Devido aos pacotes de voz serem transmitidos usando UDP, não há garantia de entrega. Pacotes que forem perdidos acabam tornando uma conversa ininteligível.

Esses problemas eram causados pelo dummynet, utilizado como *traffic shaper*. Ele utiliza o algoritmo WFQ para gerenciar a fila de pacotes e por isto não consegue prover uma real qualidade do serviço (QoS).

O pacote ALTQ (Alternate Queueing) [Cho, 2004], portado do OpenBSD para o FreeBSD, provê um conjunto de políticas de gerenciamento de filas e componentes para QoS. Para habilitar suporte ao mesmo foi necessário aplicar um *patch* no kernel e então recompilar alguns módulos e binários.

Com a inclusão do ALTQ o sistema passou a ter suporte múltiplas políticas de gerenciamento de banda. Atualmente estão sendo usadas somente CBQ (Class-Based Queueing) [Floys and Jacobson, 1995] e PRIQ (Priority Queue), os quais garantem além de gerenciamento de quantidade de banda disponível ao cliente a possibilidade de dar maior prioridade aos pacotes de voz, fazendo com que sejam despachados antes que os demais.

5. Conclusões e planos futuros

O trabalho apresentado neste artigo foi desenvolvido inicialmente para solucionar alguns problemas encontrados no dia-a-dia em redes sem fios. Como resultado disto, foi criado

³<http://vtun.sourceforge.net/>

⁴<http://www.kame.net/racoon/>

um produto que integra diversos conceitos e tecnologias, é versátil, seguro e de baixo custo e que apresenta várias vantagens em relação a um AP comercial.

Como exemplo, podemos citar o equipamento AP2000, da Proxim. Seu custo é de aproximadamente R\$ 2300,00 e ele não é capaz de prestar alguns dos serviços mais simples, como roteamento, QoS, NAT, autenticação PPTP, tunelamento. Além disso seu suporte a *firewall* se resume a um filtro de pacotes baseado em identificação de protocolo e número de porta.

O SIM Gateway passou rapidamente do meio acadêmico para o uso comercial em empresas provedoras de serviços de conexão sem fios via rádio no Estado do RS.

O uso de software de código aberto foi um fator decisivo para o sucesso do empreendimento. Apesar do caráter comercial do produto, não foram feitas alterações proprietárias no software utilizado, mantendo a total compatibilidade com o software como ele é distribuído. Isto garante a possibilidade de usar as novas versões que venham a ser desenvolvidas no futuro.

Mesmo sendo o produto considerado atualmente completo, em ambiente de produção, identificamos três necessidades que devem ser atendidas no futuro. A primeira delas é uma atualização da interface de configuração via WWW, para adaptá-la aos novos serviços que foram incluídos depois de estar pronta a versão inicial.

A segunda necessidade é a inclusão de um servidor de nomes, pois atualmente o sistema atua apenas como cliente. O principal empecilho para isto é a limitação de espaço disponível no *flash disk*.

A maior dificuldade ainda está em suportar roteamento dinâmico. Usar OSPF seria a melhor alternativa, mas até o momento não foi encontrado um servidor OSPF com um tamanho aceitável para ser armazenado no *flash disk*.

Referências

- Cho, K. (2004). Altq: Alternate queueing for BSD UNIX. <http://www.cs1.sony.co.jp/person/kjc/kjc/software.html#ALTO>.
- Flickenger, R. (2002). *Building Wireless Community Networks*. O'Reilly & Associates, Sebastopol.
- Floys, S. and Jacobson, V. (1995). Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386.
- Intersil (2003). PRISM II: the complete “antenna-to-computer” solution. <http://www.intersil.com/data/fn/fn4904.pdf>.
- ORiNOCO (2003). ORiNOCO world pc card. <http://www.orinocowireless.com/products/all/orinoco/docs/ds/PC-card.pdf>.
- Pall, G. and Zorn, G. (2001). Microsoft Point-To-Point Encryption (MPPE) protocol. <http://www.ietf.org/rfc/rfc3078.txt>.

Roteamento IP em redes ópticas

Bruno Castro da Silva, Juergen Rochol, Lara Dalto de Souza, Vandersilvio da Silva

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul
Caixa Postal 15064 – 90501-970 Porto Alegre, RS

{bcs, juergen, ldsouza, vsilva}@inf.ufrgs.br

Resumo. As tecnologias ópticas têm desempenhado um papel de crescente importância nos grandes backbones da internet. Entretanto, a estruturação da rede ainda reflete tecnologias desenvolvidas para a transmissão de voz, o que explica a atual tendência de integração do nível IP diretamente sobre a plataforma óptica, removendo camadas como o ATM e SDH/SONET. Essa integração traz consigo inúmeros desafios e dificuldades. O presente artigo irá apresentar e criticar propostas de soluções para um destes problemas, mais especificamente o de roteamento IP em redes ópticas.

1. Introdução

É bastante plausível assumir que a estruturação futura da rede mundial de computadores será baseada na integração de redes ópticas com tecnologias baseadas no IP. Entretanto, até agora, essas tecnologias ainda oferecem problemas significativos de integração. Enquanto que as tecnologias IP foram desenvolvidas tendo em vista a transmissão de dados, as tecnologias ópticas, como o SDH/SONET (*Synchronous Digital Hierarchy/Synchronous Optical NETwork*), foram pensadas para facilitar a transmissão de voz. O que se percebe atualmente é a necessidade de uma unificação que permita a diversificação e diferenciação dos serviços oferecidos via rede, de preferência de modo a utilizar a imensa largura de banda disponibilizada pelas redes ópticas. Para que a utilização de IP diretamente sobre a plataforma óptica seja possível, será preciso resolver alguns problemas que emergem da remoção das camadas intermediárias comumente utilizadas, como as camadas ATM e SDH/SONET. Este trabalho irá focar um desses problemas, o do roteamento de tráfego IP em plataformas ópticas.

2. Caracterização do problema

Em redes ópticas, é aceitável assumir que o roteamento baseado na comutação de datagramas se mostre impraticável [Ghani, 2000]. Isso se deve ao fato de que os datagramas (ex: pacotes IP) devem ser roteados *hop by hop* em cada nó por onde passam, seguindo um modelo *store-and-forward*. Tendo em vista as dificuldades de se implementar *buffers* ópticos¹, esse modelo é de difícil adaptação para tecnologias não-elétricas.

Atualmente, a solução utilizada em redes ópticas é trafegar IP sobre uma pilha de camadas formada por ATM, SDH/SONET e WDM ou, mais recentemente, DWDM

¹Geralmente são utilizados trechos longos de fibra que causam um delay em determinado sinal, ao invés de se realizar uma conversão óptico-elétrico-óptico (OEO).

(*Dense Wavelength Division Multiplexing*). No futuro, provavelmente as camadas ATM e SDH, que atuam como intermediárias entre o IP e o DWDM, serão removidas e suas funções serão transferidas diretamente para o nível óptico ou para o IP [Ghani, 2000]. Essa transformação irá modernizar a estrutura das redes ópticas, as quais ainda se sustentam em tecnologias *voice-centered*. A consequência direta disso será a obtenção de menores tempos de processamento e menor latência. Infelizmente, surgem alguns problemas quando se tenta encapsular IP diretamente na plataforma óptica:

- sinais digitais ópticos necessitam de regeneração 3R (re-amplificação, reformatação e re-sincronização), e no entanto o IP não prevê essas funcionalidades;
- atualmente não existem roteadores completamente ópticos, e backbones ópticos que trafegassem diretamente IP exigiriam roteadores que comutassem milhões de pacotes por segundo, o que atualmente é impossível;
- a implementação de redes baseadas na pilha de camadas IP → ATM → SONET/SDH → DWDM oferece um problema de escalabilidade devido à necessidade de se garantir que cada camada seja capaz de disponibilizar as necessidades de velocidade exigidas pelas demais;
- redes ópticas atualmente precisam implementar conversores óptico-elétrico-óptico (OEO) em diversos dispositivos, o que acarreta o surgimento de gargalos na rede;
- a criação de *buffers* para armazenar dados ópticos não é simples, o que implica que a comutação precisa ser do tipo *cut-through*, e não *store-and-forward*.

Tendo em vista esses problemas e a impossibilidade de se utilizar comutação baseada em pacotes, estão sendo propostos esquemas alternativos de roteamento em redes ópticas, como o roteamento de rajadas e o roteamento por comprimentos de onda².

3. Estado da arte e soluções propostas

As novas abordagens para estruturação de redes ópticas propõem que a arquitetura de 4 camadas (IP, ATM, SDH/SONET, DWDM) seja colapsada a fim de evitar os múltiplos *headers* e os problemas já comentados, como o da escalabilidade. As tarefas do ATM e do SDH seriam movidas para o IP e DWDM, respectivamente. Assim, o QoS atualmente provido pelo ATM poderia vir a ser responsabilidade do IP, por exemplo, enquanto que a adaptação em caso de falhas, provida pelo anel auto-reconfigurante do SDH/SONET, se tornaria responsabilidade da camada DWDM.

Sabe-se que, atualmente, os roteadores mais rápidos disponíveis são os chamados *Terabit IP Routers*, ou *Carrier Class Routers*. Esses roteadores são capazes de suportar, segundo alguns fabricantes, de 0.5 a 90 milhões de pacotes por segundo. Entretanto, a maior parte desses roteadores ainda é desenvolvida para o mercado SONET/SDH. Roteadores totalmente ópticos, ou que sejam capazes de rotear IP diretamente sobre a camada óptica, ainda não são uma realidade.

Uma técnica de comutação mais interessante do que a baseada em comutação de datagramas é o MPLS. O MPLS (MultiProtocol Label Switching), ou MPλS, (*Multiprotocol Lambda Switching*, o equivalente óptico do MPLS), é baseado em um conceito similar

²Existe ainda uma terceira alternativa, a comutação óptica de pacotes (OPS – *Optical Packet Switching*). O OPS não será comentado neste trabalho porque funciona exatamente como o roteamento normal de pacotes, sendo bastante ineficiente para redes ópticas devido a necessidade de conversões OEO.

ao de circuitos virtuais. O MPLS associa um *label* (rótulo) aos pacotes IP de determinado fluxo de dados, o qual é roteado ponto-a-ponto pelo backbone [Mateus et al., 2003]. O *label* é uma abstração genérica que pode ser desde um VC de rede ATM até a especificação de comprimentos de onda em uma fibra óptica. A grande vantagem deste método de comutação é que o encaminhamento do fluxo se dá através do chamado LSP (*Label Switch Path*), que funciona como um circuito virtual [Ramaswami and Sivarajan, 1995].

O estabelecimento de um LSP entre os roteadores internos da rede MPLS, chamados de LSRs (*Label Switching Routers*), segue etapas semelhantes às necessárias para o estabelecimento de um circuito. O estabelecimento de um LSP se dá através do uso de um protocolo de distribuição de *labels* chamado *Label Distribution Protocol* (LDP). Esse protocolo é o responsável por distribuir informações de LSPs entre os LSRs de modo que eles possam criar as associações entre *labels* e fluxos [Qiao, 2000]. O primeiro *label* de um pacote é designado por um roteador de borda chamado de *Label Edge Router* (LER) no momento em que o pacote ingressa em uma rede MPLS.

3.1. GMPLS

Uma generalização do MPLS, e que constitui um forte candidato a método de roteamento para redes ópticas comutadas, é chamada de GMPLS (Generalized MPLS). O GMPLS é baseado na utilização de identificadores de canais ópticos como rótulos. A técnica GMPLS pode ser utilizada com o DWDM, e neste caso o *label* possui associação direta com o comprimento de onda de determinado canal óptico.

O GMPLS estende o MPLS a fim de prover um plano de controle (sinalização e roteamento) para dispositivos que sejam capazes de comutar dados tanto no domínio de espaço (ex: portas de entrada/saída), tempo (ex: ADMs SONET) e comprimento de onda (lambda ópticos). O plano de controle do GMPLS promete simplificar o gerenciamento da rede, automatizando tarefas relacionadas ao estabelecimento fim a fim das conexões e garantindo o nível de qualidade de serviço que é exigido pelas novas e sofisticadas aplicações³.

O GMPLS também é responsável pela extensão de protocolos de sinalização e de roteamento do MPLS de modo a acomodar as características de TDM/SONET às redes ópticas. Os protocolos de sinalização mais populares são o RSVP-TE (*RSVP Traffic Engineering*) e o CR-LDP (*Constraint-based Label Distribution Protocol*)⁴. Além disso, um novo protocolo, chamado de LMP (*Link Management Protocol*), foi introduzido a fim de gerenciar e manter a conectividade e gerenciar falhas no link.

Quanto ao plano de controle do GMPLS, são previstas várias funções, dentre as quais as mais importantes para este trabalho são a de gerência e disseminação do status do link, por serem fundamentais para o roteamento em GMPLS. O roteamento em GMPLS é baseado em extensões dos protocolos OSPF e IS-IS. Note que a utilização de um algoritmo baseado no OSPF é interessante devido à necessidade de se ter acesso, em todos os nodos (OXCs e roteadores), a informações topológicas e a respeito do estado dos enlaces.

³Para isso são utilizadas tabelas de equivalência para encaminhamento (FECs) que agrupam de fluxos com mesmos requisitos de engenharia de tráfego.

⁴A sinalização no GMPLS tem a vantagem de resolver o problema da necessidade de configuração manual de dispositivos ópticos como *Optical Cross Connect* (OXCs) e *Optical Add/Drop Multiplexer* (OADM).

Para finalizar a contextualização do GMPLS enquanto alternativa para roteamento óptico, é apresentado um quadro contendo uma breve comparação entre vários domínios de chaveamento e os correspondentes modos de encaminhamento.

Tipo de chaveamento	Tipo de tráfego	Encaminhamento	Ex. de dispositivo
Pacotes/células	IP, ATM	<i>hop-by-hop</i> , VC	Roteador IP/switch ATM
Tempo	TDM/SONET	Slot de tempo em ciclos repetitivos	<i>digital OXC, add/drop multiplexing</i>
Comprimento de onda	(independe)	baseado em lambdas	DWDM
Domínio de espaço	(independe)	Linha, fibra	OXC

Quadro 1: Comparação entre os modos de chaveamento em redes ópticas.

3.2. Burst switching

O Optical Burst Switching (OBS) é uma tecnologia de rede proposta no final dos anos 90 com a finalidade de aumentar a eficiência do transporte de IP em redes ópticas. O OBS tenta obter uma boa relação custo/eficiência de maneira automática, usando para isso a combinação de tecnologias ópticas e elétricas/eletroônicas [Gauger and Dolzer, 2003].

O grande diferencial do OBS é que os canais de transmissão podem ser alocados dinamicamente e os dados podem ser transmitidos em rajadas (conjuntos de pacotes). As rajadas de dados passam pelos nodos sem sair do domínio óptico, o que evita os *delays* causados por conversões OEO. As informações de controle, por sua vez, são sinalizadas e processadas separadamente. Em contraste com o *switching* óptico de pacotes, o OBS agrupa eletronicamente uma série de pacotes em uma rajada de tamanho variável, de acordo com o nodo de destino e com a classe de QoS envolvida. Os dados de controle são utilizados para configurar as matrizes de comutação antes que os dados cheguem nos respectivos nodos intermediários (controle *out-of-band*). Além disso, as rajadas são comutadas de modo assíncrono nos nodos, e só deixam o domínio óptico quando alcançam o destino final. É fácil perceber que a granularidade envolvida na transmissão é intermediária, situando-se entre o circuito e o pacote, uma vez que uma rajada é constituída por uma seqüência de vários pacotes [Qiao, 2000].

É válido salientar que, ao contrário do que acontece durante o estabelecimento de um circuito virtual comum, não há espera pela confirmação de recebimento das informações de sinalização. Se ocorrer algum problema durante o tratamento eletrônico das mensagens de controle, a rajada será prontamente descartada nos nodos intermediários. Esse tipo de reserva de recursos sem confirmação, conhecido como *one-way*, faz com que os nodos intermediários passem a trabalhar em modo de envio imediato (*cut-through*), evitando atrasos desnecessários [Mateus et al., 2003].

Note que para o bom funcionamento do OBS é preciso haver uma definição satisfatória do tamanho ótimo para as rajadas, em função da rede e das características de serviço desejadas. Embora existam várias heurísticas, essa ainda é uma questão em aberto [Buchta et al., 2003].

3.3. OBS e a comutação de lambdas

O OBS oferece também uma solução ao problema do desperdício de banda que é alocada de maneira exclusiva e não é utilizada plenamente, típico de redes baseadas na comutação

de lambdas. Esse problema é resolvido exatamente pelos métodos de sinalização já comentados, e faz com que o OBS, quando comparado com mecanismos de roteamento baseados puramente na comutação por lambdas, apresente melhor utilização da largura de banda, menor latência e menor desperdício. Entretanto, a comutação OBS também oferece alguns problemas, como o da *resolução de contenção*, que diz respeito a situações onde duas rajadas tentam utilizar o mesmo lambda. Nesse caso, uma das rajadas deve ser descartada ou contida através da inserção de atrasos, conversão de comprimentos de onda ou deflexão para outra fibra. Nenhuma dessas alternativas é de fácil implementação, o que dificulta a escalabilidade em redes baseadas em OBS.

3.4. Métodos de comutação por rajadas

Existem algumas variações para o método de comutação por rajadas, as quais diferem basicamente quanto ao modo como os recursos são liberados após a transmissão:

1. **TAG (Tell-and-Go)**: O bloco de controle é transmitido por um canal separado pouco antes da transmissão da rajada. Ao final, um novo sinal é enviado a fim de liberar os recursos. Não há confirmação de recebimento dos sinais de controle;
2. **IBT (In-band Terminator)**: Semelhante ao modo como ocorre a comutação de pacotes tradicional, no sentido em que as informações de controle e de terminação são enviadas como adendos aos dados, no mesmo canal;
3. **RFD (Reserve a Fixed Duration)**: Semelhante ao TAG, mas informa juntamente com o sinal de controle o tempo máximo esperado para a duração da rajada. É a proposta que está sendo melhor vista no momento, já que geralmente é capaz de operar em modo *cut-through*. Os nodos intermediários trabalham em domínio eletrônico apenas durante a leitura dos blocos de sinalização, em função dos quais ajustam os comutadores DWDM e fazem a reserva de banda.

4. Considerações finais

Entre as tecnologias que vêm sendo usadas na integração IP/nível óptico, os paradigmas GMPLS e OBS se destacam por proporem soluções inteligentes e flexíveis na alocação de recursos. A comutação de IP em redes ópticas pode ser solucionada pelos dois métodos, embora ambas as tecnologias se encontrem em estágios iniciais de desenvolvimento e ainda não possam ser consideradas soluções definitivas e plenamente confiáveis.

O GMPLS vem sendo visto como a melhor estrutura de integração entre IP e DWDM, pois proporciona bons controles de engenharia de tráfego, além de se adequar à tecnologia WDM quando comprimentos de onda são usados como rótulos.

O OBS, por sua vez, se comparado com a comutação de lambdas ou de pacotes ópticos, proporciona uma boa relação custo/benefício devido à melhor utilização dos recursos disponíveis, menor latência e a maior adaptabilidade. A complexidade de implementação também é relativamente menor, embora possa levar a problemas de escalabilidade.

Para finalizar, segue um breve quadro comparativo acerca das características relativas dos vários paradigmas de comutação óptica. O **Quadro 4** deixa claro, por exemplo, que a comutação de lambdas apresenta baixa utilização da banda, causada pelo desperdício provocado por canais inativos, e que o método de rajadas possui uma baixa

latência, graças ao processo de sinalização sem confirmação. É válido ressaltar que as características apresentadas não são absolutas, e só fazem sentido quando interpretadas no contexto de comparação com os demais paradigmas.

Paradigma de comutação óptica	Largura de banda (utilização)	Latência (incluindo setup)	Adaptabilidade (tráfego e falhas)
Lambda	Baixa	Alta	Baixa
Rajada	Alta	Baixa	Alta
Pacotes	Alta	Baixa	Alta

Quadro 2: Comparação entre os vários paradigmas de comutação óptica.

É válido salientar que nenhuma destas tecnologias está amadurecida. Só o tempo será capaz de mostrar as deficiências e vantagens de cada uma. Entretanto, o *burst-switching* aparenta ter uma leve vantagem devido ao fato de aproveitar melhor a banda, e também por praticamente evitar a reserva de um lambda exclusivo por conexão, devido à velocidade de *release* após a rajada. Além disso, o OBS evita a necessidade de *bufferização* nos nodos intermediários, o que faz com que a comutação na rede óptica seja muito mais eficiente. O GMPLS, por outro lado, aparenta ser uma melhor solução na busca de QoS e reserva fixa de banda, e também possui características de implementação que possibilitam um grau de escalabilidade superior.

Como comentário final, é interessante lembrar que as tecnologias comentadas não precisam ser entendidas como soluções mutuamente exclusivas, e sim como os primeiros passos dados na busca de uma arquitetura capaz de unir o IP às plataformas ópticas.

Referências

- Buchta, H., Patzak, E., and Saniter, J. (2003). Maximal and effective throughput of optical switching nodes for optical burst switching. In *Proceedings of the Optical Fiber Communication Conference (OFC2003)*, Atlanta, EUA.
- Gauger, C. and Dolzer, K. (2003). Trends in optical burst switching. In *Proceedings of the SPIE ITCom 2003 Conference*, Orlando, EUA.
- Ghani, N. (2000). Ip over optical. In *Proceedings of Optical Networking and Communications Conference 2000 (ICOMM2000)*, Dallas, EUA.
- Mateus, G. R., Loureiro, A. A., and Nogueira, J. M. (2003). Alocação de caminhos comutados por rótulo em redes multiplexadas por comprimento de onda. In *SPG'2003 - Anais da VII Semana de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais (SPG'2003 UFMG)*, Minas Gerais, Brasil.
- Qiao, C. (2000). Labeled optical burst switching and ip/wdm integration. Technical report, IEEE Communications Magazine.
- Ramaswami, R. and Sivarajan, K. N. (1995). Routing and wavelength assignment in all-optical networks. *IEEE/ACM Transactions on Networking*, 3(5):489–500.

ANALISE COMPARATIVA DE PROTOCOLOS DE ROTEAMENTO EM REDES MOVEIS AD HOC

Underléa Cobreira Corrêa, Vitório Bruno Mazzola, M.A.R. Dantas

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina
(UFSC)

Campus Universitário – Trindade 88040-900 – Florianópolis – SC – Brasil

{underlea, mazzola, mario}@inf.ufsc.br

Resumo. Neste artigo apresentamos uma análise comparativa, para verificar as principais características dos protocolos com uma abordagem híbrida, que implementa o paradigma de agentes móveis, e aqueles que não fazem uso de tal mecanismo. Nossa objetivo é o estabelecimento de parâmetros que apontem as características mais apropriadas de um protocolo de roteamento para execução dentro de um ambiente de rede móvel ad hoc. Os resultados de nossa pesquisa indicam que os protocolos de roteamento considerados apresentam aspectos inerentes apropriados para o desenvolvimento de um framework de sub-sistema que atenda as exigências estabelecidas para uma performance diferenciada de roteamento em redes móveis.

1. Introdução

As redes móveis *ad hoc* (MANETs) são ambientes que se destacam dentro da comunicação wireless [WU 04]. Temos verificado uma tendência na miniaturização, aumento de capacidade de processamento e armazenamento dos dispositivos móveis que reunidos a aplicações capazes de oferecer boa interatividade entre hardware e software, têm contribuído com o aumento da popularidade dessas redes [PERKINS 01].

As MANETs são relativamente novas, todavia o conceito de redes de pacote de rádio móvel existe desde os anos 70, não muito antes de iniciar o desenvolvimento da tecnologia de comutação de pacotes, que cresceu dentro do que conhecemos agora como *internet*, o Departamento de Defesa (DoD) dos EUA pesquisa como habilitar essa tecnologia para operar sem as restrições da infra-estrutura fixa [PERKINS 01].

Uma das motivações originais das MANETs foi encontrada, segundo [PERKINS 01], na necessidade de sobrevivência dos militares em campos de batalha. Sendo que em regiões tais como desertos e florestas vírgens não há infra-estrutura de comunicação territorial, bem como em situações que permitam a existência de uma, corre-se o risco de destruição da comunicação local. A arquitetura *ad hoc* com seus dispositivos móveis auto-organizáveis é também usada na assistência em desastres, conferências, redes de sensores, área de redes pessoais (PAN – *Personal Area Network*) e aplicações de computação embutida [WU 04, MIGAS 03].

Entre os diversos problemas que as MANETs apresentam, temos o problema da localização dos *hosts*. Devido ao fator mobilidade, diversos trabalhos de pesquisa, como [JOA-NG 99, NIKAЕIN 01, NIKAЕIN 00, DOV 02, MARWAHA 02, MINAR 99] têm sido desenvolvidos com o intuito de oferecer entre outros, um algoritmo de roteamento que defina a topologia da rede e esteja constantemente informando o estado e a

localização de cada *host*. Alcançar a eficácia de um protocolo de roteamento para uma rede móvel *ad hoc* é uma tarefa instigante, já que esse deve executar eficazmente sobre uma larga escala de configurações das MANETs.

Com o intuito de realizar uma análise de domínio dos protocolos de roteamento híbrido, buscamos nesse artigo comparar as características dos algoritmos que utilizam a abordagem híbrida, através do uso de agentes móveis, [MARWAHA 00, ROYCLHOUDHURY 00, ONISHI 01] e aqueles que não fazem uso de tal mecanismo [JOA-NG 99, NIKAEIN 01, NIKAEIN 00].

O artigo está estruturado da seguinte forma. Características das MANETs na seção 2. Na seção 3 os protocolos de roteamento abordados nesta pesquisa. Comparações na seção 4. Conclusões e trabalhos futuros na seção 5.

2. Características das MANETs

As MANETs [PERKINS 01] são caracterizadas por um sistema autônomo de nodos móveis independentes, podendo operar de modo isolado Figura 1 ou por intermédio de um *gateway* de interface com a rede fixa Figura 2, onde seus nodos são equipados com transmissores e receptores usando antenas onidirecionais, que captam sinais de todas as direções (*broadcast*), altamente direcional (*peer-to-peer*), ou uma combinação de cada. Possuem tecnologia de comunicação *wireless*, onde os dispositivos computacionais móveis são capazes de trocar informações diretamente entre si ou através de *multi-hops* sem a necessidade de infra-estrutura de comunicação [PERKINS 2001, CORSON 98] e devem estar fisicamente habilitados para se comunicarem mutuamente mesmo quando roteadores, estações base ou provedores de serviços de *internet* (ISPs), não podem ser encontrados [CAMPBELL 03]. As MANETs empregam a estrutura tradicional TCP/IP para fornecer uma comunicação fim-a-fim entre os nodos [PERKINS 01] contudo, devido a sua mobilidade e aos recursos limitados em redes *wireless*, cada camada do modelo TCP/IP requer definição e modificações para funcionar eficientemente.

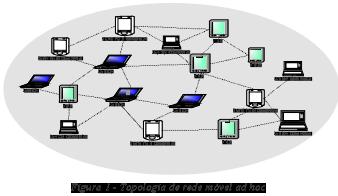


Figura 1 - Topologia de rede móvel ad-hoc

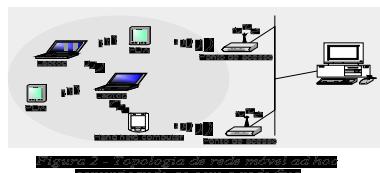


Figura 2 - Topologia de rede móvel ad-hoc com roteador de acesso à rede fixa

CORSON 99 define que as características mais salientes das MANETs são: a topologia dinâmica e móvel onde os nodos são livres para moverem-se arbitrariamente, causando dificuldade na determinação de uma rota válida e na localização dos *hosts* para estabelecimento da comunicação; segurança limitada já que essas redes são geralmente mais propensas ao aumento das possibilidades de escuta e invasões; rápidas instalações sem a necessidade de infra-estrutura fixa; conectividade através de um canal de comunicação entre dois ou mais *hosts* dentro de uma mesma área geográfica de alcance de ondas de rádio; largura de banda forçada, pois os *links wireless* continuam tendo

capacidade significantemente mais baixa do que as redes fixas e operação de energia limitada onde os nodos contam com baterias para funcionar.

3. Roteamento em Redes Móveis Ad Hoc

Por mais que o ambiente MANET seja produtivo, a natureza dinâmica de sua topologia dificulta a realização do roteamento *multi-hop* [ONISHI 01].

Os protocolos de roteamento para MANETs, segundo [ABOLHASAN 03] podem ser classificados em três grupos diferentes: *global/pro-ativo*, *on-demand/reativo* e híbrido. No protocolo de roteamento pro-ativo, é definido por [ABOLHASAN 03] que as rotas para todo destino são determinadas no início e mantidas usando um processo periódico de atualização da rota. Já no protocolo reativo um nodo inicia a descoberta da rota somente quando ele deseja se comunicar com seu destino[NIKAEIN 01]. O algoritmo de roteamento híbrido combina propriedades básicas das duas primeiras classes de protocolos em uma e é considerado por [ABOLHASAN 03] a nova geração dos protocolos de roteamento para MANETs projetados para aumentar a escalabilidade permitindo que os nodos com proximidade trabalhem juntos para formar uma espécie de *backbone* a fim de reduzir o *overhead* de descoberta da rota. Todavia, os protocolos [MARWAHA 02, ONISHI 01] que fazem uso de agentes móveis buscam oferecer um algoritmo de roteamento que defina a topologia da rede e esteja constantemente informando o estado e localização de cada host, buscando atender a lista de propriedades qualitativas e quantitativas desejáveis a um protocolo de roteamento, sugerido em [CORSON 99].

3.1. Algoritmo de Roteamento para MANET

Nessa seção abordaremos as características dos protocolos de roteamento para o ambiente de redes móveis *ad hoc* que serão utilizados neste trabalho de pesquisa. Primeiramente relataremos três trabalhos bem conhecidos e categorizados como protocolos de roteamento híbrido, em conseguinte trataremos de outros dois trabalhos de pesquisa que utilizam agentes móveis onde [MARWAHA 02] pode ser caracterizado como híbrido e [ONISHI 01] caracteriza-se pela utilização da abordagem pro-ativa. Dentre os estudados temos:

A. ZHLS – Zone-based hierarchical link state

O ZHLS [JOA-NG 99] é um protocolo híbrido de roteamento *peer-to-peer*, característica essa que evita gargalo de tráfego, previne único ponto de falha e simplifica o gerenciamento da mobilidade. O ZHLS incorpora informação da posição dentro de uma abordagem de roteamento hierárquico que utiliza sistema de posição global (GPS – *Global Position System*). A rede é dividida dentro de zonas não sobrepostas e o roteamento é feito pela definição do *zone_ID* e do *node_ID* do destino.

B. DDR – Distributed dynamic routing algorithm for mobile ad hoc networks

O DDR [NIKAEIN 00] é uma abordagem híbrida que se baseia no conceito de zonas não sobrepostas, onde cada nodo precisa conhecer somente o próximo salto para todos os nodos dentro de sua zona, reduzindo dessa forma o roteamento de informação e utilização da largura de banda. O DDR evita *broadcasting* enviando somente a informação necessária embutida em *beacons* (sinais de alarme) para os vizinhos, e o tamanho de suas zonas aumentam e diminuem dinamicamente de acordo com algumas

características como densidade do nodo, taxa de conexão e desconexão da rede, força de transmissão e mobilidade do nodo. A idéia principal do DDR é construir uma floresta de uma topologia da rede de modo distribuído usando somente a troca periódica de mensagens entre os nodos e seus vizinhos.

C. HARP – Hybrid ad hoc routing protocol

O HARP [NIKAEIN 01] é um protocolo de roteamento de nível de zona hierárquico, onde o roteamento é executado em nível de *intra-zone* e *inter-zone*. O roteamento *intra-zone* confia num mecanismo pro-ativo (herdado do DDR), já a *inter-zone* confia num mecanismo reativo como no ZHLS. No HARP cada nodo mantém somente a informação de roteamento daqueles nodos que estão dentro de sua zona, e de suas zonas vizinhas; é responsável pelo descobrimento e mantimento dos trajetos para satisfazer os requerimentos da aplicação; evita atraso extra causado por falha no trajeto durante a transmissão de dados, restaura o trajeto antes do período de instabilidade, além de gerar e selecionar trajetos conforme o nível de estabilidade da zona.

D. Mobile agents based routing protocol for mobile ad hoc networks (*Ant-AODV*)

Esse protocolo [MARWAHA 02] tenta superar desvantagens como redução da capacidade de aproveitamento da rede devido à atualização contínua das tabelas de roteamento dos nodos móveis causada pelo AODV [DAS 03] e a dependência dos nodos móveis que aguardam a informação das rotas para os vários destinos fornecidos pelos agentes formigas. Devido a isso esse protocolo busca combinar o método de roteamento AODV [DAS 03] com *Ant-based routing* [MINAR 99] para dar origem a o método de roteamento híbrido *Ant-AODV* hábil a reduzir o atraso fim-a-fim e a latência da descoberta da rota fornecendo alta conectividade quando comparado aos métodos de roteamento AODV [DAS 03] e *Ant-based* [MINAR 99]. O protocolo de roteamento *Ant-AODV* usa troca de mensagens de erro de rota (RERR) para informar os nodos da falha do link como no AODV, e mensagens *Hello* são espalhadas através de *broadcast* para manter a tabela dos vizinhos atualizada para uma escolha aleatória do próximo salto.

E. The multi-agent system for dynamic network routing

Esse protocolo de roteamento [ONISHI 01] é caracterizado como pro-ativo, onde suas rotas são feitas em direção oposta da busca pretendida; cada nodo conhece qual o próximo nodo para transmitir pacotes de mensagem, cada nodo na rede disputa o mesmo papel no roteamento; os pacotes de mensagem podem ser transmitidos através de várias rotas. Tem como vantagem o menor atraso em fazer uma comunicação e uma estabilidade por causa da constante quantidade de pacotes de controle; economiza o tamanho dos pacotes de mensagem, possui alta robustez e flexibilidade. Esse modelo está interessado apenas em uma rota provável e não em todo o mapa da rede como a maioria dos protocolos de roteamento pro-ativo.

F. Mobile agents for routing topology discovery, and automatic network reconfiguration in ad hoc networks

A pesquisa [MIGAS 03] propõe acessar diferentes modelos de uso dos agentes móveis e estáticos para determinar a melhor rota através da rede móvel *ad hoc* fornecendo benefícios como melhor desempenho, escalabilidade, comunicação fim a fim confiável, reduzir possíveis atrasos e minimizar perdas. A idéia é baseada no fato de que cada nodo terá um agente estático que executará no fundo monitorando a disponibilidade

dos recursos da rede e os agentes móveis coletarão informações dos agentes estáticos usando-as para determinar, por exemplo, o melhor trajeto para roteamento de tráfego na rede.

4. Comparação

Nessa seção demonstramos através da Tabela 1 a comparação entre os protocolos de roteamento estudados na seção 3.1, considerando algumas diferenciações e as principais características dos algoritmos em estudo, tendo como pontos considerados:

1. Característica de abordagem pro-ativa e reativa
2. Suporte a múltiplos trajetos
3. Faz *flood* limitado para transmitir informações de roteamento ou solicitação de rotas
4. Possui mecanismos para aumentar as entradas na tabela de roteamento
5. Aplica estratégia de consumo de energia e questões de segurança
6. Usa *beacons* para enviar informações
7. Executa roteamento em dois níveis
8. Utiliza agentes móveis e estáticos para efetuar comunicação
9. Aponta sistema de roteamento orientado a curtos trajetos
10. Os nodos utilizam mensagens de requisição de rotas ao destino
11. Mensagens de erro de rota (RERR) são utilizadas para informar os nodos sobre uma falha no enlace local
12. Mensagens *Hello* são utilizadas para manter a tabela dos vizinhos atualizada
13. Utiliza sistema de posição de posição global (GPS –Global Position System)
14. Baseia-se no conceito de zonas
15. Baseia-se na metáfora de formiga

Tabela 1 – Comparação das Características de roteamento

Protocolo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DDR	xx	x	-	-	x-	x	x	-	-	-	-	-	-	x	-
HARP	xx	x	x	-	x-	x	x	-	-	x	-	-	-	-	-
ZHLS	xx	x	-	-	-	x	-	x	-	-	-	x	x	-	
Ant-AODV	xx	-	-	-	-	-	-	x	x	x	x	x	-	-	x
The multi agent system for dynamic network routing	x-	x	x	x	x	-	-	x	x	-	-	-	-	-	-
Mobile agents for routing, topology discovery, and automatic network reconfiguration	-	x	x	-	x	-	-	xx	x	-	-	-	-	x	-

Legenda: X = possui a característica

- = ausência da característica

Por meio de nossa comparação percebemos que cada algoritmo de roteamento busca superar através da abordagem híbrida e/ou da aplicação do paradigma agente móvel, desvantagens como desperdício dos recursos da rede wireless, falta de escalabilidade em relação a freqüente mudança topológica da rede e a conexão fim-a-fim apresentadas nos protocolos de roteamento, e.g. [GUANGYU 02 e MINAR 99] que aplicam abordagem puramente pro-ativa ou reativa. Além disso, podemos também estabelecer o domínio de aplicações dos algoritmos em estudo por meio das

características coletivas que os unificam e pelos aspectos distintos peculiares a cada protocolo.

Dentre os protocolos de roteamento comparados utilizaremos em nossos experimentos futuros os algoritmos de roteamento HARP [NIKAEIN 01], Ant – AODV [MARWAHA 02] e o *Multi-agent system for dynamic network routing* [ONISHI 01]. Justificamos a escolha do HARP dentre os abordados por esse aplicar o nível de estabilidade da zona como uma métrica de determinação de rota e manutenção adiantada do trajeto simplificado do protocolo de roteamento o que não é o caso do DDR e ZHLS, além disso, o HARP utiliza o DDR em seu algoritmo para gerar a estrutura lógica com relação às propriedades da rede (número de nodos na rede, número de comunicações, freqüência de mudança topológica). Outro algoritmo a ser analisado será o Ant-AODV por apresentar característica híbrida como o HARP e ainda empregar o uso de agentes móveis. Além do HARP e o Ant-AODV analisaremos o algoritmo *The multi-agent system for dynamic network routing* por esse ser um algoritmo de abordagem pro-ativa que aplica o paradigma agente móvel.

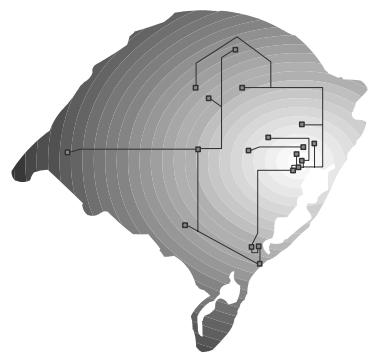
5. Conclusão

Neste artigo apresentamos uma análise comparativa que identifica características coletivas dos protocolos de abordagem híbrida, que implementam o paradigma de agentes móveis, e aqueles que não fazem uso de tal mecanismo. Nosso objetivo foi o estabelecimento de parâmetros que apontem os aspectos fundamentais de um protocolo de roteamento híbrido dentro de um ambiente de rede móvel *ad hoc*. Os resultados de nossa pesquisa indicam que os protocolos de roteamento considerados apresentam aspectos inerentes apropriados para o desenvolvimento de um *framework* de sub-sistema que atenda as exigências estabelecidas para uma performance diferenciada de roteamento dentro do contexto dinâmico das redes móveis *ad hoc*.

6. Referências

- ABOLHASAN, M., WYSOCKI, T., DUTKIEWICZ, Eryk. (2003) “A review of routing protocols for mobile ad hoc networks”, Elsevier Computer Science, www.elsevier.com/locate/adhoc, Aug.
- CORSON, S., MACKER, J. (1999) “Mobile Ad Hoc Networking (MANET)”. IETF RFC 2501, <http://www.ietf.org/rfc/rfc2501.txt>, Jan.
- CORSON, S., MACKER, J. (1998) “Mobile Ad Hoc Networking and IETF”, ACM Mobile Computing and Communication Review, Vol. 2 and 3, Numbers 1,2,3,4, http://protean.itd.nrl.navy.mil/manet/manet_home.html, Oct.
- CAMPBELL, A. T., CONTI M., GIORDANO S.. (2003) “Mobile Ad Hoc Networks”, Kluwer Academic Publishers. Manufactured in The Netherlands. Mobile Networks and Applications,p. 483–484.
- DAS, S., PERKINS, C. and ROYER, E. (2003) “Ad hoc on demand distance vector (AODV) routing” RFC 3561, <http://www.ietf.org/rfc/rfc3561.txt>.
- GUANGYU, Pei; GERLA, M., Tsu-Wei Chen (2000) “Fisheye state routing: a routing scheme for ad hoc wireless networks”, Communications ICC. IEEE International Conference, Vol.1, Jun, p. 70-74.

- JOA-NG, M. and I-TAI, L. (1999) "A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks", IEEE Journal on selected areas in communications, vol 17, nº 8, p. 1415-1425.
- MARWAHA, S., THAM, C. K. and SRINIVASAN, D. (2002) "Mobile Agents based Routing Protocol for Mobile Ad Hoc Networks", Global Telecommunications Conference. GLOBECOM '02. IEEE, Vol. 1, p. 17-21.
- MARWAHA, S., THAM, C. K. and SRINIVASAN, D. (2002) "A Novel Routing Protocol using Mobile Agents and Reactive Route Discovery for Ad Hoc Wireless Networks", Networks ICON, 10th IEEE International Conference, p. 311 – 316.
- MINAR, N., KRAMER, K.H., and MAES, P. (1999) "Cooperating mobile agents for dynamic Networking", Software Agents for Future Communications Systems, Chapter 12.
- MIGAS, N., BUCHANAn, W., J., WILLIAM, McARTNEY, K. A. (2003) "Mobile Agents for Routing", Topology Discovery, and Automatic Network Reconfiguration in AdHoc Networks, Enginnering of Computer-Based Systems, 10th IEEE International Conference and Workshop, p. 200-206.
- NIKAEIN, N. and BONNET, C. (2001), "HARP- Hybrid Ad Hoc Routing Protocol", In: IST2001 – International Symposium on Telecommunications <http://www.eurecom.fr/~nikaeinn/harp.ps>.
- NIKAEIN, N., LABIOD, H. and BONNET, C. (2000) "DDR – Distributed dynamic routing algorithm for mobile ad hoc networks" Mobile and Ad Hoc IEEE Networking and Computing. MobiHOC 2000 First Annual Workshop, p19- 27.
- ONISHI, R., YAMAGUCHI, S., MORINO, H. and SAITO, T. (2001) "The Multi-agent System for Dinamic Network Routing", Autonomous Decentralized Systems, 5th IEEE International Symposium, p. 375-382.
- PERKINS, C. E. "Ad Hoc Networking", 1st ed. United States of America, Addison-Wesley, p. 11- 27.
- ROYCLHOUDHURY, R., BANDYOPADHYAY, S., K., P.; (2000) "A distributed mechanism for topology discovery in ad hoc wireless networks using mobile agents", Mobile and Ad Hoc Networking and Computing, MobiHOC IEEE 2000 First Annual Workshop, p. 145-146.
- WU, J. and STOJMENOVIC I. (2004) "Ad Hoc Networks", Computer, Volume: 37, Issue: 2, Feb.2004, p. 29-31.



Sessão Técnica 9

Estudos de Caso

Relatório Técnico da Transmissão do SBRC

João Marcelo Ceron, Pablo Lerina Rodrigues, Maiko de Andrade, Valter Roesler

Pesquisa de Redes de Alta Velocidade – PRAV - Unisinos

{jmarcelo, pabloalr, maiko, roesler}@exatas.unisinos.br

Abstract: This paper describes the methodology used by PRAV team's, from UNISINOS, for the transmission of the lectures during 22º Simpósio Brasileiro de Redes de Computadores, carried through from May 10th to 14th, 2004 in Gramado. The transmission was performed in two auditoriums. The signal was codified in each auditorium and retransmitted to a server located in POP-RS (Point of Presence of Internet in RS), where it was redirected for the viewers. An interactivity mechanism was inserted, allowing remote questions in real time to the speaker. The videos were recorded simultaneously, and made available for asynchronous access in the web page of PRAV. The transmitted signal was also used to test the cache system of RNP's GT-Vídeo.

Resumo. Este artigo relata a metodologia utilizada pela equipe do PRAV, da UNISINOS, para a transmissão das palestras durante o 22º Simpósio Brasileiro de Redes de Computadores, realizado nos dias 10 a 14 de maio de 2004 em Gramado. Efetuou-se a transmissão das palestras que estavam sendo efetuadas em dois auditórios. O sinal foi codificado no local e transmitido para um servidor no POP-RS (Ponto de Presença da Internet no RS), onde era redirecionado para os telespectadores. Foi inserido um mecanismo de interatividade, permitindo perguntas remotas em tempo real ao palestrante. Os vídeos foram gravados simultaneamente, e hoje se encontram disponíveis na página do PRAV para acesso assíncrono. O sinal transmitido também foi utilizado para testar o sistema de cache do GT-Vídeo, da RNP.

1. Introdução

Com o crescente número de usuários com acesso à Internet de banda larga [ARGEZ 2003], as transmissões multimídia em tempo real tornaram-se uma realidade, possibilitando que usuários visualizem uma melhor qualidade de vídeo e áudio. Neste contexto, as transmissões multimídia obtiveram um grande avanço, possibilitando a criação de trabalhos nesta área.

Pode-se citar como exemplo os trabalhos realizados no projeto RMAV¹ da RNP, onde observa-se que a maioria dos consórcios formados trabalharam em projetos ligados a aplicações multimídia de tempo real, como os trabalhos de telemedicina [FLORE 2001] e sobre teleconferência e videoconferência [ROES 2000].

Este artigo relata a metodologia utilizada pela equipe do PRAV (Pesquisa em redes de alta velocidade) da UNISINOS para a transmissão ao vivo com interatividade durante o 22º Simpósio Brasileiro de Redes de Computadores (SBRC 2004).

¹ <http://www.rnp.br/remav/>

Em linhas gerais, cada codificador fazia simultaneamente a codificação e o armazenamento dos vídeos em disco, transmitindo um fluxo codificado em modo *unicast* para um servidor localizado no POP-RS² (Ponto de Presença da Internet no Rio Grande do Sul). Este, por sua vez, fazia o interfaceamento como os usuários pelo *site* do evento. Incorporou-se neste modelo um ambiente de interatividade, possibilitando ao usuário fazer questionamentos ao palestrante, através de uma interface *Web*.

O sistema também teve como objetivo a disponibilização dos vídeos para acesso assíncrono após o evento, através de um servidor de vídeo sob-demanda. Para isso, toda a transmissão foi armazenada simultaneamente à codificação.

O modelo de transmissão colaborou para que fossem feitos testes no Dynavideo Vod [LIMA 2001], que é um sistema que implementa *multicast* em nível de aplicação, com uma estrutura distribuída em vários POPs pelo Brasil.

Este artigo está dividido da seguinte forma: a seção 2 descreve a arquitetura utilizada para a transmissão, bem como os recursos e o modelo de interatividade utilizado. Na sessão 3, é apresentado o sistema D-Vod. Na sessão 4, são analisadas algumas estatísticas coletadas durante a transmissão, e por fim, na sessão 5, serão descritos os principais problemas e conclusões que a equipe se deparou durante as transmissões, visando alertar outras pessoas interessadas no mesmo assunto para tomarem medidas preventivas, evitando assim que tais problemas se repitam em outros eventos.

2. Descrição da Arquitetura e recursos utilizados

O objetivo da transmissão foi possibilitar que o público em geral obtivesse acesso a algumas das atividades em andamento no SBRC. Limitou-se o número de transmissões em duas, pois era o que permitia o número de recursos alocados. Decidiu-se, então, pela filmagem de todas as sessões técnicas, todos os tutoriais e o 5º workshop da RNP. A seguir, será descrita a arquitetura utilizada com maiores detalhes.

A primeira decisão a ser tomada foi em relação à localização do servidor. Havia a possibilidade de que o mesmo fosse colocado no laboratório de pesquisa do PRAV, na Unisinos, que faz parte da rede da RNP. Isso já tinha sido feito em outras ocasiões com sucesso, entretanto, devido à característica deste evento, onde a maioria dos acessos seriam de fora do Rio Grande do Sul, o local que mais otimizaria o tráfego seria a sua colocação junto ao POP-RS. Isso fica claro ao observar a topologia de rede da figura 1a. Os pacotes eram transmitidos de cada codificador (localizado nos auditórios) até o servidor. Quando um usuário solicitava assistir à transmissão, ele estabelecia uma conexão *unicast* com o servidor. Caso o servidor ficasse na Unisinos, existiria um tráfego adicional entre o POP-RS e a Unisinos. Logo, optou-se por colocar o servidor junto ao POP-RS.

Pode-se observar também através da figura 1a, que a largura de banda entre Gramado e o POP-RS era de dois enlaces E1 (2Mbit/s), perfazendo um total de 4Mbit/s. Esta banda era compartilhada por todas as 45 estações do evento, mais os *notebooks* conectados através de *wireless*, via tecnologia 802.11b, conforme detalhado na figura 1b.

² <http://www.pop-rs.rnp.br>

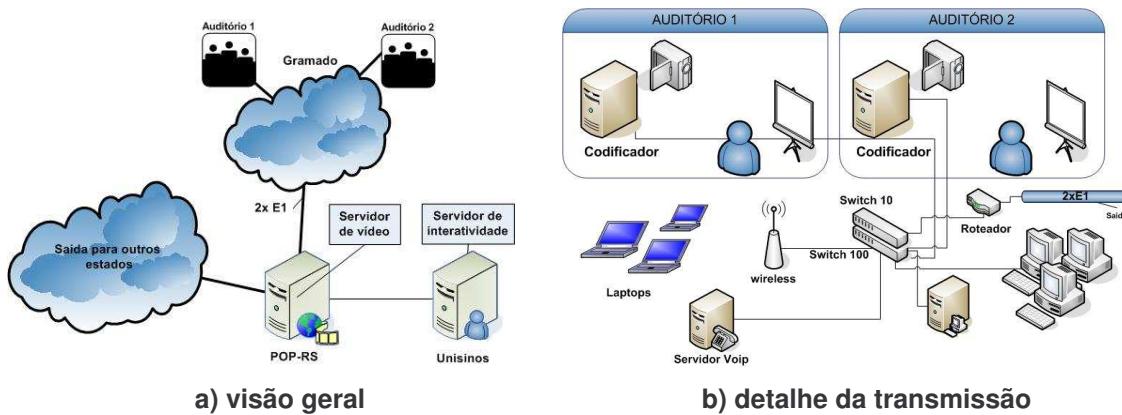


Figura 1. Arquitetura utilizada

A fim de garantir uma banda aceitável para a transmissão, solicitou-se à equipe do POP-RS que fosse feita uma priorização no tráfego para as máquinas transmissoras de vídeo (*encoders*). Definiu-se que cada *encoder* teria reserva de banda até o limite de 256 kbit/s, acima disto os pacotes entrariam no modo de melhor esforço (*Best Effort*). Como a codificação era feita em 3 taxas (250 kbit/s + 100kbit/s + 34 kbit/s), percebe-se que a reserva ficou abaixo do ideal, que deveria ser de, no mínimo, a soma das três taxas, ou 384 kbit/s.

Os *Codecs* (*Coders-Decoders*), são algoritmos de codificação usados para codificar e decodificar um fluxo de dados, os utilizados neste ambiente de transmissão foram da série Windows Media 9, para áudio e vídeo. Foi possível captar o áudio diretamente da mesa de som do auditório, proporcionando uma melhor qualidade.

Buscando obter uma solução escalável, ou seja, adaptável a usuários localizados em redes com diferentes larguras de banda, foi feita a codificação do áudio e vídeo em três níveis de compressão, listados a seguir: a) taxa de 34 kbit/s, priorizando o áudio, visando usuários com pouca banda disponível, conectados via modem ou redes congestionadas; b) 100 kbit/s, para usuários de banda larga, como ADSL e *cable modem*; c) 250 kbit/s, para usuários locais.

Para armazenamento do vídeo, utilizou-se a qualidade mais alta transmitida, que no caso foi 250 kbit/s. Os vídeos estão disponibilizados em <http://vod.prav.unisinos.br>.

Para a captura do vídeo, foram utilizadas duas câmeras Sony DCR PC-110, acopladas a um tripé e localizadas estratégicamente a fim de proporcionar um melhor ângulo de filmagem.

Em termos de hardware, foram utilizadas quatro máquinas. Três delas foram utilizadas como codificadores, sendo que dois para uso nos auditórios e uma de backup (codificador 3). A quarta máquina foi utilizada como servidor, e ficou localizada no POP-RS. As configurações das máquinas podem ser vista na tabela a seguir.

Codificador 1 e 2	Codificador 3	Servidor
Pentium III 700MHz Memória: 256 Mbytes Disco Rígido: 8GBs Placa de captura: BT878 Windows XP	Pentium IV 1.8GHz Memória: 256 Mbytes Disco Rígido: 40GBs Placa de captura: Broadway Windows 2000	Pentium IV 1.8GHZ Memória: 256 Mbytes Disco Rígido: 40GBs Windows 2003 Server

Pode-se notar que os “codificadores 1 e 2” são máquinas de baixo processamento, o que trouxe uma série de problemas, posteriormente descritos neste artigo. Alguns critérios foram levados em consideração para a escolha do software, tais como a possibilidade de codificação em diferentes taxas de transmissão, popularidade de seus decodificados (clientes), e custos. A solução escolhida foi o *Windows Media Encoder* para a codificação, e como servidor, o *Windows Media Server*.

Este sistema também permitiu a escalabilidade, uma característica muito importante [CHAD 1995], pois efetua automaticamente a gerência da melhor qualidade a transmitir para o usuário final, ou seja, das taxas de compressão previamente definidas no codificador, conforme descrito anteriormente. Outras ferramentas similares foram testadas, como pode ser visto em [ROES 2003], entretanto possuíam alto custo ou recursos limitados, como limite de usuários conectados.

A equipe de filmagem contava com quatro pessoas, duas em cada auditório. Achou-se importante que uma pessoa de cada equipe ficasse em estado de alerta, para que eventuais problemas durante a transmissão fossem brevemente solucionados, assim como possibilitar um revezamento entre os integrantes. Foi decidido pela equipe, que seria importante um ambiente interativo, onde fossem maximizados a interação dos usuários com o evento. Assim, através do *site* do evento, foi disponibilizado simultaneamente dois “auditórios virtuais”, onde cada usuário escolhia a sessão que mais lhe interessava.

Ao entrar no auditório virtual, o usuário encontrava o formulário ilustrado na figura 2, possibilitando ao mesmo fazer questionamentos ou comentários. Quando isso ocorria, o questionamento era enviado para o servidor de interatividade, localizado no PRAV, onde era armazenado em uma base de dados, e apresentado em uma página HTML.

Auditório Virtual 1

Clique aqui para assistir no Windows Media Player

Caso tenha algum problema para acessar tente esse link: <mms://200.132.0.69/av1> diretamente no Windows media Player

Use o formulário abaixo para fazer perguntas ao palestrante e também para fazer comentários sobre a transmissão.

A transmissão do evento automaticamente se adapta ao ambiente de rede no qual você se encontra.
Qualquer dúvida maior use o [ICQ 117606068](#).
Para assistir você precisa ter o Windows Media Player 9 instalado. Se você não tem o Media Player, [clique aqui](#) para fazer o download.

Nome: _____

Instituição: _____

Pergunta: _____

Enviar

Volta

Figura 2. Tela de interatividade.

Esta página HTML possuía um acesso restrito, e poderia ser acessada pelo *chair*, presidente da sessão, ou por um membro auxiliar da equipe da filmagem, ao final das apresentações, para que o questionamento fosse feito ao ministrante. No caso os questionamentos foram controlados pela equipe de filmagem. Na prática, a quase totalidade dos questionamentos que ocorreram foram para avisar de algum problema ou pergunta técnica sobre a transmissão, levando à conclusão que é necessário criar um mecanismo de interação técnica com o usuário final, desvinculado das perguntas ao palestrante.

3. Dynavideo Vod

Uma cooperação foi estabelecida com o GT-Video, grupo de trabalho de vídeo da RNP (Rede Nacional de Pesquisa), para que o fluxo de transmissão oriundo de Gramado fosse retransmitido no sistema Dynavideo-vod [LIMA 2001].

O servidor de vídeo instalado no POP-RS recebia uma conexão de um servidor do sistema D-VOD, que também se encarregava de distribuir o fluxo.

O D-vod é um sistema de transmissão de vídeo sob demanda que simula *multicast no nível de aplicação*, permitindo a um servidor redirecionar o vídeo requerido para determinado cliente. O que diferencia o D-vod dos demais servidores de vídeo sob demanda é a possibilidade de localização dos vídeos dinamicamente, conforme a sua localização, e armazená-los em *cache*. Isso é feito por uma série de servidores instalados nos diversos POPs da RNP. A vantagem da *cache* é que o vídeo é transmitido uma única vez, e o próximo cliente que tentar acessar o mesmo não precisará ir até a origem dos dados, podendo buscar o vídeo no servidor de *cache* mais próximo, evitando redundância de tráfego.

Quando uma requisição é feita no D-Vod, o servidor mais próximo do cliente irá repassá-la para o próximo servidor na sua hierarquia. Caso este servidor não tenha o vídeo, ele fará a solicitação ao próximo, e assim sucessivamente até que seja encontrado o vídeo. Encontrado o vídeo, o servidor irá transmitir para quem fez a solicitação, até encontrar o cliente final.

4. Estatísticas de Acesso

Arquivos de *log* foram gerados durante toda a transmissão, para que fosse possível obter uma idéia mais precisa de todo o processo, e analisar possíveis falhas. Dois fatores importantes para a avaliação são o número de acessos e tempo médio de visualização dos vídeos, e são mostrados na figura 3. Vale salientar que os gráficos referem-se somente ao sistema de transmissão do PRAV, portanto, além dos usuários remotos mostrados, existiam ainda os que estavam assistindo via o sistema Dynavideo, descrito na sessão anterior.

Para geração dos gráficos, no parâmetro tempo médio de acesso, foram desconsiderados acessos menores que 18 segundos, pois este foi o tempo médio obtido para fazer a bufferização somado ao atraso do vídeo.

Observa-se, através do primeiro gráfico da figura 3, que o auditório 1 contou com 200 a 400 acessos por dia, e o auditório 2 entre 50 e 200. Nos dias 13 e 14, não houve filmagem no auditório 2, visto que o único evento transmitido foi o workshop da RNP2. Através do segundo gráfico percebe-se que o tempo médio de permanência é baixo, da ordem de poucos minutos. Analisando os *logs*, percebeu-se que algumas máquinas entravam e saiam da transmissão diversas vezes, resultando num maior número de acessos e menor tempo médio de permanência. Na prática, existiam aproximadamente 10 pessoas em cada auditório assistindo constantemente as palestras. Os valores poderiam ter sido maiores caso houvesse uma maior divulgação da transmissão via Internet, porém, isso não foi feito pelo temor de diminuir muito o número de inscritos.

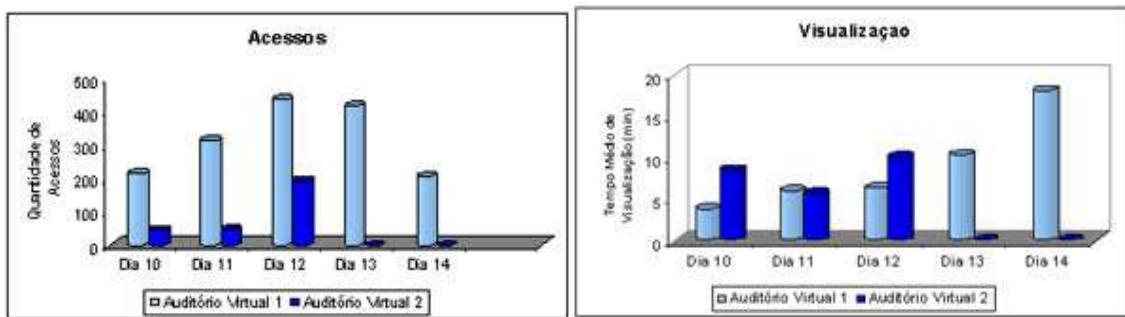


Figura 3. Resultados de número de acessos e tempo médio de visualização.

5. Problemas Encontrados e conclusões

O sistema funcionou adequadamente, entretanto, as máquinas possuíam baixo poder de processamento para suportar as três taxas estipuladas, conforme detalhado na sessão 2. Recomenda-se, no mínimo, o equivalente a Pentium IV 1GHz, 256MB de RAM e disco de 20Gbytes. A interatividade ficou prejudicada pela falta de informação da transmissão, entretanto, é imprescindível. Outra conclusão obtida é a necessidade de existir um mecanismo de perguntas aos palestrantes e outro diferente para avisos de falha.

Bibliografia

- [ARGA 2003] ARGAEZ, Enrique De. *Broadband Usage Keeps Growing.* <http://www.internetworldstats.com/articles/art030.htm>. (09/08/2004).
- [FLOR 2001] FLORENTINO, Pablo Vieira; DANTAS, Geysa Vinhaes; “*Sessão Médica HiperClínica – Aplicação em TeleMedicina: Uma experiência de Implementação*”. Anais do III Workshop RNP2, UFSC, Florianópolis, SC, Maio, 2001.
- [ROES 2000] ROESLER, Valter; “*Transmissão multimídia em redes de computadores: um relato para redes locais e Internet2*”. Anais do II Workshop RNP2, UFMG, Belo Horizonte, MG, Brasil, Maio, 2000.
- [ROES 2003] ROESLER, Valter; ANDRADE, Maiko de; CERON, João Marcelo; “*Aulas remotas on-line utilizando transmissão de vídeo: estudo de*”. XIV Simpósio Brasileiro de Informática na Educação - SBIE 2003.. Rio de Janeiro. RJ. Brasil. Novembro, 2003.
- [SCHU 1993] SCHULZRINNE, A.; CASNER, S.; FREDERICK, Ron; “*RTP: A Transport Protocol for REAL-Time Applications*”, Internet Engineering Task Force, Internet Draft, Oct. 20, 1993. <http://citeseer.ist.psu.edu/schulzrinne01rtp.html>
- [LIMA 2001] LIMA, Pedro; e LEITE, TAVARES, Tatiana Aires; CARNEIRO, Virginia de Paula e SOUZA FILHO, Guido Lemos. “*Arquitetura e Implementação do DynaVideo.*” In: XXVII CONFERÊNCIA LATINO-AMERICANA DE INFORMÁTICA, 2001, Venezuela. Anais do XXVII CONFERÊNCIA LATINO-AMERICANA DE INFORMÁTICA. 2001.
- [CHAD 1995] CHADDHA, N., WALL, G. A., AND SCHMIDT, B. “*An end to end software only scalable video delivery system.*” In Proceedings of the Fifth International Workshop on Network and OS Support for Digital Audio and Video (Durham, NH, Apr. 1995), ACM.

Implementação de um Filtro Adaptativo LMS Aplicado ao Cancelamento de Eco em Voz Sobre IP

Guilherme Rauter Corsetti, Moisés Coster, Ricardo Balbinot, Fladhimyr Câmara Castello, Jorge Guedes Silveira

Departamento de Engenharia Elétrica - Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), Porto Alegre - Rio Grande do Sul - Brasil

{gcorsetti, mcoster, rbalbinot, fcastello, jguedes}@gparc.org

Resumo. O objetivo desse artigo é mostrar uma implementação de um cancelador de eco utilizando o algoritmo LMS (Least Mean Square). O algoritmo LMS é um dos pioneiros para o desenvolvimento de processos adaptativos. Sua característica simples de implementação é responsável por sua grande popularidade entre os desenvolvedores. O eco, na maioria das vezes, pode ser considerado um problema de grande relevância na telefonia tanto convencional quanto em transmissão por redes IP, sendo responsável pela má qualidade e um grande desconforto para os usuários durante as ligações. Para que seja testada a eficiência do cancelador que será implementado, serão simulados de diferentes atrasos do eco na rede. A partir dos resultados desses testes da simulação, poderá ser analisado o tempo que o cancelador necessitará para aprender as características do sinal de eco e eliminá-lo.

1. Introdução

Um filtro adaptativo pode se ajustar ao ambiente onde está sendo utilizado, transformando um sinal de entrada em um sinal de saída desejado. Para que ocorra essa transformação pode ser usado o algoritmo LMS, que será tratado com mais profundidade na Seção 2. Através de um filtro adaptativo, é possível, por exemplo, cancelar o eco em um sistema de telefonia.

O eco é um problema comum, que ocorre quando há uma reflexão do sinal de voz, por exemplo, nas redes telefônicas. Esse problema pode ter diversas origens. Existem dois tipos de eco na rede de telefonia convencional: o eco acústico e o eco híbrido. O primeiro é causado por ruídos existentes no ambiente, como reflexões do som nas paredes onde a pessoa está falando e pela realimentação do alto-falante para o microfone. O segundo é gerado em transformadores híbridos, usado na rede pública de telefonia comutada (RPTC) para a transmissão de dados, onde um link de quatro fios

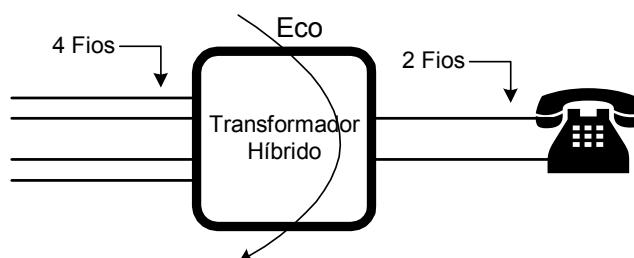


Figura 1 - Geração de eco no transformador híbrido

tem que ser transformado em um link de dois fios (para ligação à residência do usuário) (ver Figura 1).

O trabalho sobre processamento de sinais adaptativos, referenciado em [Widrow and Stearns 1985], foi um grande impulso na pesquisa de algoritmos para o processamento adaptativo. O algoritmo LMS desenvolvido nessa obra permitiu uma evolução significativa nessa área. Existem diversos outros algoritmos adaptativos além do LMS que podem ser implementados para o desenvolvimento de um cancelador de eco, como por exemplo, as redes neurais que estão sendo cada vez mais utilizadas em todos os tipos de processos adaptativos.

Neste artigo será implementado um filtro adaptativo LMS, cuja função será cancelar o eco de uma rede IP ligada a uma rede de telefonia convencional. O cancelador será testado com diferentes atrasos do eco, assim poderemos verificar se o mesmo será eficiente para o seu propósito. Para que possa ocorrer a simulação dos atrasos do eco, será implementado um algoritmo que possua tal capacidade.

2. Filtro Adaptativo LMS

O filtro adaptativo LMS tem como função, no cancelador de eco, alterar o sinal r' (ver Figura 2), de tal modo que ele gere um sinal y , na saída do filtro, que possa ser subtraído do sinal $(s + r)$ (ver Figura 2), cancelando o sinal r e assim minimizando o erro (ϵ). A velocidade com que o filtro realiza o processo de aproximação de y e r é o que garante o sucesso do cancelamento do eco. Na Figura 2 é mostrado um modelo de funcionamento de um filtro adaptativo no modo de cancelamento de ruído.

Inicialmente, considera-se um sinal (s) acrescido de um ruído (r) qualquer (neste caso será o eco) que vem da rede. A soma desses dois sinais $(s + r)$ é chamada de sinal entrada (d). Precisa-se obter uma amostra de r de forma a termos o sinal de referência r' , que pode ser considerado uma versão distorcida, atrasada, mas correlacionada do sinal

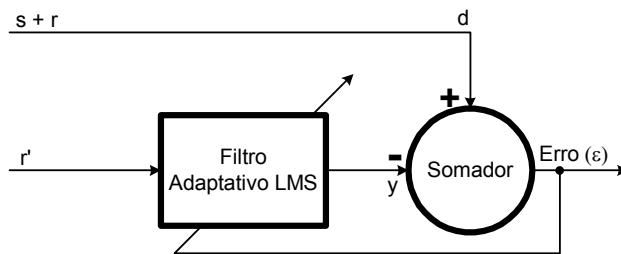


Figura 2 - Modelo de Filtro Adaptativo

r. O sinal r' passa pelo filtro que ajusta seus coeficientes para que o sinal de saída y seja igual a r , de forma que o erro (ϵ) seja apenas formado pelo sinal desejado s [Widrow and Stearns 1985].

Conclui-se que o erro é igual a: $\epsilon = s + r - y$

Chega-se ao erro ideal quando r e y são iguais, matematicamente ficará: $r = y$

Então: $\epsilon = s$

O erro ideal será igual ao sinal que estamos querendo que não seja anulado, ou seja, o sinal desejado s . As amostras dos sinais de entrada d são armazenadas em vetores que chamamos de x_k , onde k é um índice de tempo. Como existe apenas uma entrada desse sinal, ela é chamada de entrada simples. A matriz de vetores de entrada ficará assim:

$$\mathbf{X}_k = [x_k \ x_{k-1} \ \cdots \ x_{k-L}]^T$$

O combinador linear adaptativo (ALC) é utilizado no algoritmo LMS, para gerar a saída y. O ALC consiste na utilização de pesos no processo adaptativo. Os pesos ficam localizados dentro do filtro adaptativo LMS. Esses pesos funcionam de acordo com o ϵ , de forma que eles vão se ajustando até que o sinal y fique igual ao sinal r.

A definição do número de pesos utilizados no processo é muito importante, pois se forem adicionados muitos pesos o processo necessitará de muita memória e acabará, com isso, prejudicando a velocidade de execução do processo adaptativo. Caso sejam adicionados poucos pesos o processo de aprendizagem do filtro pode ficar lento demais. Os pesos são armazenados no que chamamos de vetor-de-pesos que corresponde a:

$$\mathbf{W}_k = [w_{0k} \ w_{1k} \ \cdots \ w_{Lk}]^T$$

onde W é o vetor de peso, w_{Lk} são os pesos, sendo k o índice de tempo e L o índice correspondente do último vetor de peso (o mais recente). Os pesos vão convergindo até chegar no vetor-de-pesos considerado ideal, W^* . Esse ajuste “percorre” o que chamamos de função mean-square-error (MSE). Para que o vetor de pesos chegue ao W^* , ele deve alcançar o valor mínimo do MSE, pois lá está localizado o W^* . A função do MSE é definida por: $MSE = \xi = E[\epsilon^2] = E[d^2] + W^T RW - 2P^T W$

onde ϵ é o erro, d é o sinal de entrada, , W o vetor-de-pesos, R é uma matriz quadrada definida por: $R = E[X_k X_k^T]$

E o vetor P é definido como: $P = E[d_k X_k]$

O algoritmo LMS é muito importante porque é um algoritmo simples e de fácil computação. Ele é representado por: $\mathbf{W}_{k+1} = \mathbf{W}_k - \mu \nabla_k = \mathbf{W}_k + 2\mu \epsilon_k \mathbf{X}_k$

onde, o ∇ é o gradiente do erro, W é o vetor de pesos, k o número de iterações, μ é a constante de ganho que regula a velocidade e estabilidade do processo de adaptação, ϵ é o erro estimado e X é o vetor de amostras de entrada. O parâmetro μ é introduzido na equação com a finalidade de dar estabilidade ao processo de aprendizagem do filtro, e seu valor está compreendido entre: $0 < \mu < 2/\text{tr}\{R\}$, onde, $\text{tr}\{R\}$ é o somatório da diagonal da matriz de autocorrelação.

3. Cancelador de Eco IP

Um cancelador de eco tem como função eliminar o eco que está sendo gerado em uma chamada. Normalmente o eco é gerado quando há uma conexão com a RTPC, na parte de hardware do sistema chamado híbrida (que converte a chamada de quatro fios para dois fios) e este eco pode ser eliminado já no gateway de entrada da Rede IP. O eco acústico também existe na rede IP, ele é gerado da mesma forma que na rede de telefonia convencional (originado de reflexões da fala no ambiente do locutor) e é agravado quando se utiliza um computador como meio de comunicação devido à realimentação do alto-falante para o microfone. O eco é variável de chamada para chamada e até mesmo em uma mesma chamada, ou seja, é necessário que se tenha um mecanismo que se ajuste de acordo com a variação do eco e os filtros adaptativos possuem essa capacidade.

Os gateways e roteadores têm seus próprios canceladores de eco (cancelam o eco híbrido). O cancelador de eco implementado neste artigo objetiva cancelar apenas o eco acústico.

Na Telefonia IP, são necessários, então, dois canceladores de eco. Um cancelador de eco localizado no telefone IP (para eliminar o eco acústico) e o outro no gateway de interface com a RPTC (para eliminar o eco híbrido). A seguir é mostrado o modelo geral de um cancelador de eco em uma rede IP ligada à rede de telefonia, utilizando um filtro adaptativo LMS (ver Figura 3):

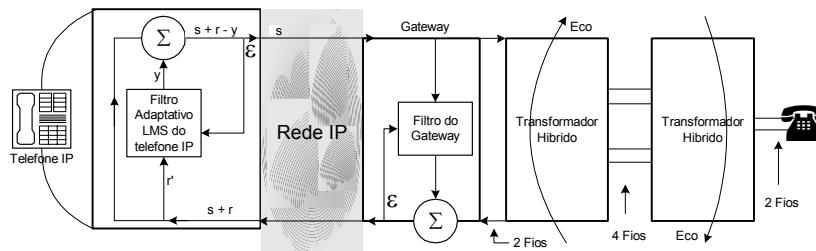


Figura 3 – Modelo de cancelador de eco de um telefone IP ligado a um telefone convencional

Quando há comunicação de telefone IP para telefone IP, existe apenas o eco acústico, então é necessário apenas o cancelador de eco que fica localizado no telefone IP. A seguir é mostrado geral de um cancelador de eco de uma ligação de um telefone IP (ver Figura 4) para o outro:

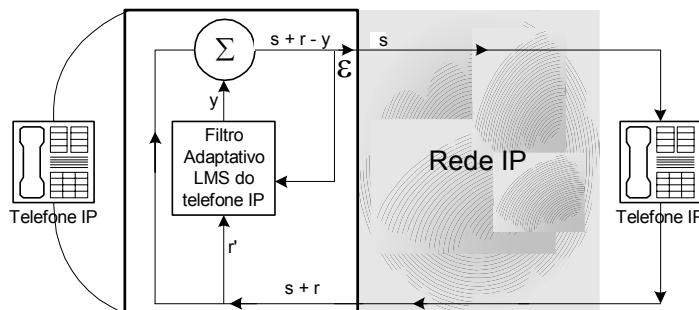


Figura 4 – Modelo de cancelador de eco de telefone IP para telefone IP

O eco nem sempre é considerado um problema, pois se o tempo de retorno do mesmo estiver abaixo de 16-20ms, este adiciona um bom reforço na voz que dá a impressão de que a ligação continua estabelecida. Acima de 20ms ele vira um problema que precisa ser controlado [Douskalis 1999]. Quanto maior o atraso e a sua variação mais difícil fica de cancelá-lo.

No caso de “fala dupla”, que ocorre quando ambos os lados da chamada estão falando, o processo de aprendizagem do filtro precisa ser paralisado (o que também deve ocorrer quando o locutor próximo ao cancelador está falando), mas o cancelamento de eco deve continuar sendo executado normalmente, com a mesma qualidade. Quando o cancelador detectar que não há mais energia no sinal de voz do locutor mais próximo o processo adaptativo pode voltar a funcionar.

O tempo necessário para que o cancelador de eco aprenda as características do sinal de eco é chamado de tempo de convergência. Para ser considerado aceitável, o tempo de

convergência tem um limite de 50 no máximo 60ms para ser executado [Douskalis 1999]. Esse limite precisa ser respeitado, pois um dos principais problemas na comunicação IP é o atraso do sinal da voz.

4. Implementação

Para a implementação do algoritmo de cancelamento de eco, foi escolhido o ambiente Delphi. Primeiramente gravamos diversas amostras de voz com codificação PCM, a 8 bits e 8kHz. Essas amostras foram utilizadas para que o cancelador pudesse ser testado.

Sobre a amostra de voz utilizada foi acrescentado eco, primeiramente com um atraso fixo e depois fizemos o atraso do eco variar. Como o eco na realidade sofre uma atenuação em relação ao sinal original, essa característica também foi simulada. O objetivo disso é simular uma fala com um sinal mais atrasado e atenuado com relação ao original (eco acústico). Sobre o arquivo gerado foi aplicado o algoritmo de cancelamento de eco que foi desenvolvido. A seguir é mostrado um fluxograma (Figura 5), que mostra como o algoritmo foi feito.

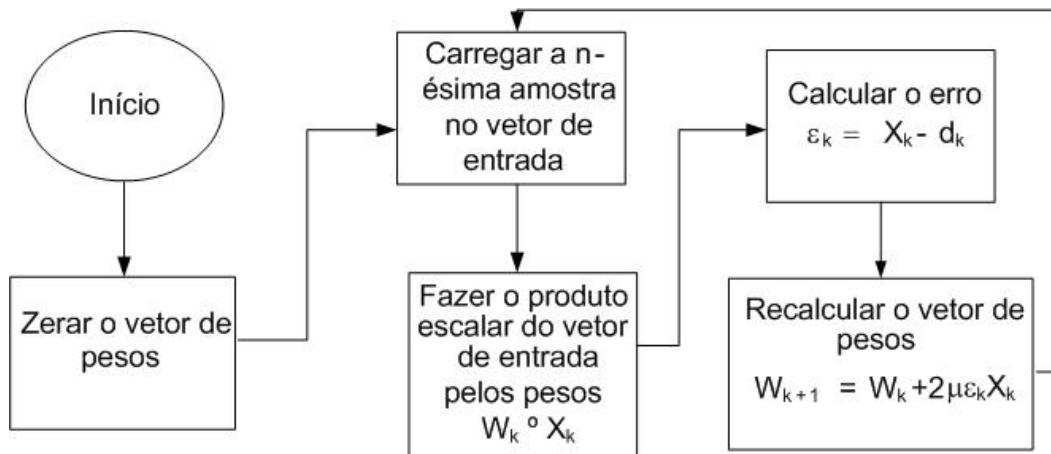


Figura 5 – Fluxograma da implementação do algoritmo de cancelamento de eco

O algoritmo ilustrado pelo fluxograma foi aplicado e após isso foi gerado um novo arquivo. Utilizamos um vetor com 128 pesos e o μ utilizado foi de 0,00007.

5. Testes e Resultados

Após a implementação foi verificado que o processo de cancelamento de eco teve êxito. Com os três arquivos (o original, o com o eco adicionado e com o eco cancelado), pudemos fazer comparações entre os mesmos. Apenas “ouvindo” o som de cada arquivo já foi possível verificar o cancelamento de eco.

Para que pudesse ser uma melhor avaliação dos resultados foi implementado um algoritmo, onde cada amostra de áudio pudesse ser analisada visualmente. Na Figura 6 pode-se notar claramente que o eco foi cancelado.

O primeiro gráfico mostra o arquivo com a amostra original, o segundo mostra a amostra com eco adicionado e o terceiro mostra a amostra após a aplicação do cancelador de eco. O eco adicionado nesse caso tinha 1 db de atenuação com relação a amostra original e 100ms de atraso constante.

Como se pode notar o gráfico com o cancelamento de eco está mais semelhante ao original do que ao do eco. Os gráficos não ficaram iguais, isso se deve ao tempo necessário para que o filtro se adapte.

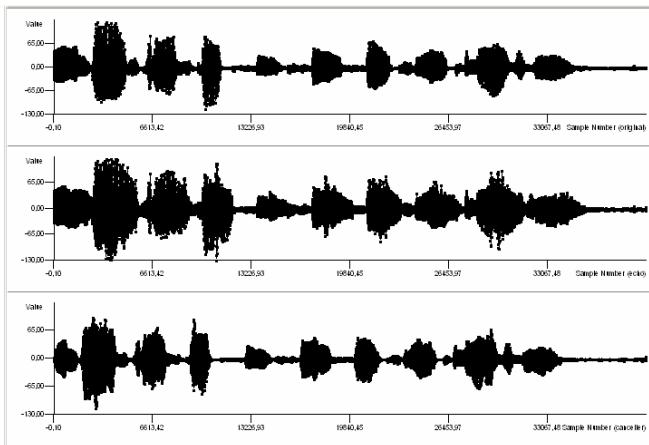


Figura 6 – Gráficos de comparação

O cancelador de eco também foi testado em um ambiente onde o atraso do eco varia. O seu desempenho obteve resultados semelhantes ao de quando o eco era constante. O filtro convergiu corretamente e dessa forma teve a capacidade de cancelar o eco.

6. Conclusões

Após a implementação e os testes que foram feitos, pudemos concluir que o algoritmo LMS é um algoritmo de baixa complexidade computacional. Sua implementação é simples, comparada a de outros algoritmos adaptativos.

Notamos também que o LMS, início do processo de adaptação é um algoritmo lento. Isso porque ele inicia sua adaptação apenas quando o número de amostras analisados pelo filtro é igual ao tamanho do vetor de pesos. O valor do μ foi extremamente importante, pois dependendo do μ utilizado o filtro não consegue convergir corretamente e não funciona.

7. Bibliografia

- B. Widrow, and S. D. Stearns, “Adaptive signal processing”, Prentice-Hall, Englewood Cliffs, 1985, pp. 1-364.
- B. Douskalis, “IP Telephony the Integration of Robust VoIP Services”, Prentice-Hall, 1999, pp.253-258.
- The International Engineering Consortium, “Echo Cancellation”, Web Forum Tutorials, http://www.iec.org/online/tutorials/echo_cancel/glossary.html, acesso 05/07/2004.
- “Echo Analysis for Voice over IP,” Version 1, 2001, pp. 1-24.
- P.R.G. Franco, M.C.F. De Castro and F.C.C. De Castro, “Introdução ao Processamento Adaptativo de Sinais Digitais”, PUCRS – FENG – DEE – Mestrado em Engenharia Elétrica, Porto Alegre, 2001, pp. 1-19.
- M. Rudson, “Acoustic Echo Cancellation Using Digital Signal Processing”, University of Queensland, 2003, pp. 69 – 72.

Uma Ferramenta para Comparação dos Mecanismos de Controle de Congestionamento em Redes TCP/IP Utilizando o NS

Luiz Antonio Rodrigues¹, Michele Mara de A. E. Lima¹

¹Colegiado de Informática – Universidade Estadual do Oeste do Paraná (UNIOESTE)
Caixa Postal 711 – 85.810-110 – Cascavel – PR – Brasil

{larodrig,michele}@unioeste.br

Resumo. Neste trabalho é apresentada uma ferramenta que através de simulações permite ao usuário a avaliação conjunta de implementações TCP, com e sem suporte a ECN e políticas de AQM, em diferentes cenários de rede, nos quais, podem ser feitas variações de tipo de tráfego e escolha das métricas de eficiência a serem utilizadas nas avaliações, tais como: throughput, goodput, taxa de perda, atraso, jitter, etc. A ferramenta foi desenvolvida de forma a permitir que a execução das simulações e a recuperação dos resultados gerados pudessem ser feitas via Web. Para tanto, foram utilizadas as seguintes tecnologias: JSP (Java Server Pages), o servidor Tomcat 4.1.10 e o simulador de redes NS (Network Simulator).

1 Introdução

O congestionamento em redes de computadores ocorre quando há insuficiência de recursos para acomodar a carga presente na rede ou quando ocorre um desbalanceamento do tráfego nos nós da rede. Quando ocorre o congestionamento e pacotes são descartados tem-se a diminuição da vazão e o aumento do atraso fim-a-fim dos tráfegos. Além disto, os recursos utilizados pelos pacotes que foram descartados são desperdiçados e quando estes pacotes são retransmitidos, novos recursos precisam ser alocados.

A implementação atual mais popular do TCP, denominada *TCP Reno*, possui um mecanismo de controle de congestionamento composto de quatro algoritmos: *Slow Start*, *Congestion Avoidance*, *Fast Retransmit* e *Fast Recovery* [Allman, 1999]. Os dois primeiros são utilizados pelo TCP emissor para controlar a quantidade de dados injetada na rede, enquanto os outros dois são utilizados para a recuperação de perdas sem a necessidade de se esperar que um *timeout* aconteça. Desta forma, o TCP ajusta a sua taxa de transmissão de acordo com a estimativa de banda passante disponível. Tal adequação é governada pelo recebimento de ACKs enviados pelo receptor. 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. Quando uma perda é detectada pela ocorrência de um *timeout*, a janela de transmissão é reduzida drasticamente para um segmento e o TCP emissor entra na fase de *Slow Start*.

A principal deficiência do TCP Reno é que seu desempenho é degradado quando múltiplos pacotes são perdidos em uma mesma janela. Quando múltiplas perdas ocorrem, o tamanho da janela é reduzido à metade ou a um segmento diversas vezes e o retorno ao tamanho original da janela ocorre após um considerável intervalo de tempo. Isto acontece porque apenas o primeiro pacote perdido é recuperado pelo *Fast Retransmit* e os demais somente após a ocorrência do *timeout*. Visando suprir esta

deficiência três novas soluções foram propostas: o TCP SACK ameniza os problemas de desempenho do TCP Reno provendo informações de quais e quantos pacotes foram recebidos corretamente pelo TCP receptor [Mathis, 1996]; o TCP *NewReno* aprimora o mecanismo de recuperação de perdas do TCP Reno através da avaliação e diferenciação dos tipos de reconhecimentos recebidos [Floyd, 1999]; finalmente o TCP *Limited Transmit* foi desenvolvido para melhorar o desempenho do mecanismo de recuperação de perdas do TCP Reno para conexões com tamanho pequeno de janela [Allman, 2001].

A notificação de congestionamento é feita no TCP através do descarte de segmentos nos roteadores congestionados. O Gerenciamento Ativo de Filas (*Active Queue Management* - AQM) foi introduzido a fim de se obter o gerenciamento mais eficiente nas filas dos roteadores. A idéia central de AQM é a notificação antecipada da existência de congestionamento incipiente através da marcação/descarte de pacotes, de forma a permitir que os emissores TCP possam reduzir sua taxa de transmissão antes que as filas fiquem cheias, evitando assim, a degradação do desempenho do TCP.

RED (*Random Early Detection*), o algoritmo de AQM recomendado pelo IETF para a Internet, faz a marcação/descarte dos pacotes utilizando propriedades estatísticas sobre o tamanho médio da fila. Primeiramente ele estima o valor do tamanho médio da fila, e o compara com dois limiares, que representam os limiares mínimo (*minth*) e máximo (*maxth*). O valor da probabilidade de marcação/descarte depende de qual das três regiões, definidas pelos dois limiares, o valor do tamanho estimado da fila encontra-se. Se o valor calculado for menor que *minth*, nenhum pacote será marcado/descartado. Se o valor calculado estiver entre *minth* e *maxth*, o algoritmo está na zona de prevenção e os pacotes são marcados/descartados com uma probabilidade *p* que cresce linearmente até *maxp*. Caso o tamanho médio da fila esteja acima de *maxth*, o algoritmo inicia o controle de congestionamento e descarta todos os pacotes que chegam [Floyd, 1993].

Determinar corretamente os parâmetros de RED é um desafio. Quando os valores destes limiares não são corretamente definidos, RED pode vir a se comportar até pior que a política *tail drop*, que descarta pacotes de forma reativa e não preventiva como em RED. Devido aos problemas apresentados por RED outras políticas de AQM foram propostas para superar suas deficiências, entre elas pode-se citar: ARED Floyd, 2001], FRED [Lin, 1997] e Blue [Lima, 2003].

No mecanismo de AQM, a notificação de congestionamento é feita geralmente aos nós finais através do descarte de pacotes. Entretanto, esta não é a única forma de notificação de congestionamento possível de ser utilizada por AQM. O mecanismo de ECN substitui o descarte de pacotes utilizando um bit de cabeçalho do pacote para sinalizar o congestionamento, caso o protocolo de transporte entenda tal notificação. [Ramakrishnan, 2001].

Evitar a ocorrência de congestionamento e controlá-lo são de capital importância para a operação da rede. Em redes *best-effort* o congestionamento pode degradar ainda mais os baixos níveis de qualidade de serviço. Em redes com suporte a QoS (*Quality of Service*), o congestionamento pode inviabilizar o compromisso de oferecimento de tal suporte. Assim sendo, a prevenção e o controle de congestionamento são fatores essenciais para aprimorar o desempenho das aplicações e a otimização de recursos da rede. Desta forma, é importante estudar os mecanismos de controle de congestionamento, visando verificar quais deles é o mais eficaz em prevenir o congestionamento, ou ainda fazer o controle eficiente quando ele ocorre, minimizando assim, os seus danos.

Um estudo sobre todos estes mecanismos pode ser mais bem desenvolvido utilizando simuladores. A vantagem de se usar simuladores é que eles permitem obter resultados bastante satisfatórios sem causar impactos indesejados no mundo real, evitando desperdícios de recursos. Além disto, podem ser realizados testes em larga escala que podem ser controlados e reproduzidos. O uso de simuladores é uma maneira bastante apropriada de se determinar, com resultados muito próximos da realidade, quais são os mecanismos de controle de congestionamento mais adequados para determinadas situações e a viabilidade de implantação dos mesmos.

O NS (*Network Simulator*) é hoje o simulador de redes padrão utilizado no meio acadêmico. Entretanto, para utilizá-lo de forma eficiente exige-se do usuário não apenas o conhecimento do simulador, bem como da linguagem TCL para gerar os scripts de simulação. Além, disto, faz-se necessário o uso de ferramentas específicas para pós-processar a enorme quantidade de dados gerados durante as simulações.

Neste trabalho é apresentada uma ferramenta que através de simulações permite ao usuário a avaliação conjunta de implementações TCP, com e sem suporte a ECN e políticas de AQM, em diferentes cenários de rede, nos quais, podem ser feitas variações de tipo de tráfego e escolha das métricas de eficiência a serem utilizadas nas avaliações, tais como: *throughput*, *goodput*, taxa de perda, atraso, *jitter*, etc. A ferramenta é bastante simples, e não se exige do usuário conhecimentos aprofundados no que diz respeito ao simulador NS, nem tampouco da linguagem TCL. Além disto, ela foi desenvolvida de forma a permitir que a execução das simulações e a recuperação dos resultados gerados pudessem ser feitas via Web. Desta forma, pode-se instalar a ferramenta e o simulador em uma máquina com grande capacidade de processamento e possibilitar que a simulação e o processamento dos dados gerados possam ser feitos a partir de qualquer máquina que possua um *browser* Web.

Este artigo está organizado da seguinte forma. A seção 2 apresenta um breve resumo das diferentes tecnologias utilizadas no desenvolvimento desta ferramenta. A seção 3 descreve a estrutura e o funcionamento da ferramenta. Finalmente, na seção 4, as conclusões são delineadas.

2 Tecnologias Utilizadas

As tecnologias empregadas no desenvolvimento do simulador incluem JSP (*Java Server Pages*), que é uma tecnologia para desenvolvimento de aplicações Web, o servidor Tomcat em sua versão 4.1.10, para suporte a tecnologia JSP e o simulador de redes NS (*Network Simulator*), utilizado para execução das simulações.

Servlets são uma tecnologia Java para o desenvolvimento de aplicações do tipo cliente-servidor. São programas executados no servidor Web, funcionando como uma camada intermediária entre as requisições recebidas do Web *browser* ou outro cliente HTTP e bancos de dados ou aplicações disponíveis no servidor HTTP. *Java Server Pages* (JSP) é a tecnologia Java para desenvolvimento de aplicações Web que permite combinar código HTML estático com conteúdo gerado dinamicamente através de *servlets*. A plataforma JSP necessita de um servidor que converta automaticamente qualquer página JSP em um *servlet* equivalente, isto é, que gere código fonte Java a partir de um documento HTML. Existem diversos servidores atualmente trabalhando com JSP. Um deles é o servidor *Tomcat* da Apache. O *Tomcat*, mais especificamente a sua versão 4.1.10, é uma referência oficial na implementação de *servlet* 2.6 e JSP 1.2. A principal vantagem da utilização do *Tomcat* é o fato dele ser gratuito.

O NS trata-se se um simulador de redes dirigido por eventos discretos que simula vários tipos de redes IP. Ele pode trabalhar simulando protocolos de transporte como o TCP e o UDP; tráfegos de aplicações como FTP, Telnet, Web, CBR (Constant Bit Rate) e VBR (*Variable Bit Rate*); políticas de gerenciamento de filas e políticas de escalonamento como *Tail Drop*, RED e suas variações; etc. O NS pode ainda gerar diversos arquivos de *log* de todos os eventos ocorridos durante o processo de simulação, permitindo que os dados desejados sejam obtidos com grande facilidade. Uma das grandes vantagens do NS reside no fato de ele ser totalmente gratuito e com código fonte aberto.

3 Estrutura da Ferramenta

A ferramenta proposta, nomeada como “*TeCePe*” possui três etapas principais. A primeira etapa é a configuração dos parâmetros da simulação, que deve ser feita pelo usuário através de um formulário em HTML. Em uma segunda etapa é realizada a simulação através do NS, gerando um conjunto de arquivos com os *traces* da simulação. Na terceira etapa estes arquivos são submetidos a um pós-processamento e os resultados gerados são coletadas pelo sistema e disponibilizados via navegador *Web*.

A estrutura de dados do *TeCePe* é composta basicamente por quatro classes principais:

- A classe *BeanUsuario*, que contém os atributos e métodos para cadastro, alteração e exclusão de usuários do sistema;
- A classe *BeanSimulacao*, que implementa os atributos e métodos da simulação requisitada, tais como solicitar uma simulação e recuperar os dados gerados por uma simulação. Os parâmetros são configurados por meio de um objeto da classe *BeanParam* e os resultados são acessados através da classe *BeanResultado*;
- A classe *BeanParam*, que contém os atributos e métodos referentes aos parâmetros da simulação requisitada por um usuário devidamente cadastrado;
- A classe *BeanResultado*, que possui atributos e métodos para recuperar os resultados de uma simulação previamente requisitada e completada, de acordo com solicitação da classe *BeanSimulacao*.

Estas classes são escritas em Java e permitem a execução das principais tarefas realizadas pelo simulador, como cadastro de usuários, pedido de simulação e visualização dos resultados gerados. A interface que interage com estas classes é construída através de arquivos JSP. Estes arquivos possuem os formulários e as chamadas aos métodos das classes descritas acima. Assim, quando o usuário deseja realizar uma simulação, ele acessa a página de configuração da simulação, que por sua vez instancia um objeto da classe *BeanSimulacao*. Os parâmetros configurados através do formulário são repassados ao objeto da classe *BeanParam* pertencente a classe *BeanSimulacao* e a simulação é iniciada. Este e os demais fluxos de execução da ferramenta podem ser vistos na Figura 1.

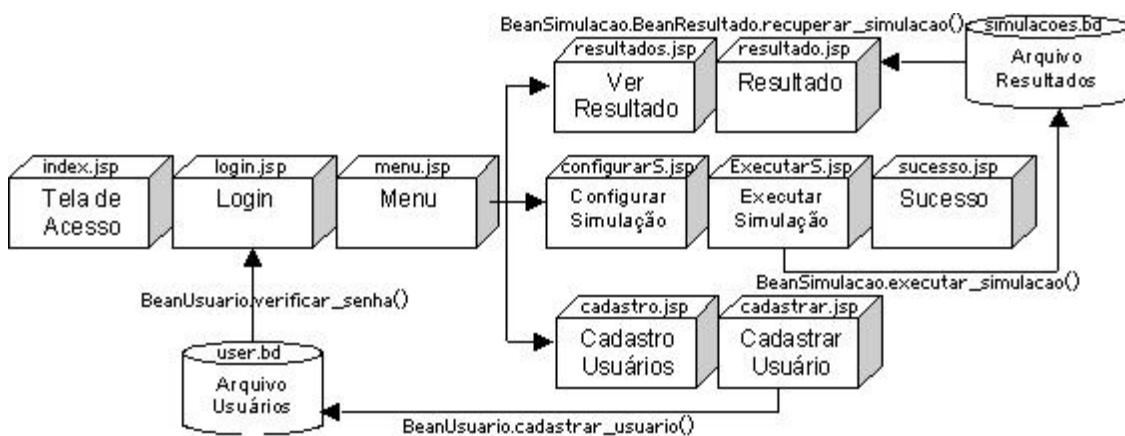


Figura 1 Fluxo de Execução da Ferramenta.

Para ter acesso ao simulador basta informar o endereço “http://enderedo_do_host:8080/Simulador” no navegador, onde o endereço do *host* pode ser endereço IP ou nome da máquina onde o *Tomcat* e a Ferramenta estão instalados.

Para utilizar a ferramenta, um usuário previamente cadastrado deve informar seu nome e senha de acesso na tela de acesso. Quando submete o pedido de acesso, o arquivo de login faz a verificação dos dados informados acessando o arquivo de cadastro (*user.bd*). Feito o acesso é disponibilizado ao usuário a tela de menu. Esta tela possui as opções disponíveis de acordo com a categoria do usuário (“Administrador” ou “Restrito”). Para usuários restritos não é disponibilizado o cadastro de usuários. Caso a opção de simulação seja escolhida, a tela de configuração da simulação é exibida contendo um formulário com os parâmetros a serem escolhidos. Quando a simulação é requisitada o arquivo *executarS.jsp* processa o pedido enviando-o a um objeto da classe *BeanSimulacao*, juntamente com os parâmetros já atribuídos ao objeto da classe *BeanParam* instanciado na própria classe *BeanSimulacao*. Se todos os parâmetros estiverem corretos uma tela de sucesso é exibida. Para ver o resultado da simulação o usuário deve acessar a tela de resultados a partir do menu, onde é exibida uma lista com as simulações requisitadas. Escolhida uma delas e desde que a mesma já tenha sido processada, serão exibidos os resultados obtidos. O tempo que uma simulação necessita para ser processada pode variar consideravelmente dependendo dos parâmetros escolhidos. Desta forma, dificilmente um resultado será fornecido imediatamente após a requisição. Caso a simulação ainda não tenha sido concluída, uma mensagem “*não processado*” aparecerá nos campos média e desvio padrão da tela de resultados.

O cálculo dos resultados é feito através de um *script Shell* que efetua um pós-processamento sobre os arquivos gerados pelo *script TCL* executado pelo NS. Destes arquivos são retirados os seguintes valores:

- *Arquivo TA*: arquivo que registra todos os eventos ocorridos durante a simulação. Dele são retirados os valores de *jitter* e *delay*, quando solicitado explicitamente;
- *Arquivo PQ*: arquivo que monitora a variação no tamanho da fila e na probabilidade de descarte. Fornece os valores da variação instantânea da probabilidade de perda e variação instantânea do tamanho da fila;

- *Arquivo MQ*: efetua o monitoramento da fila em intervalos de um segundo. Informa o tamanho da fila, taxa de utilização do enlace e taxa de perda do enlace;
- *Arquivo TT*: monitora cada fluxo gerado pela simulação. Dele é possível retirar os valores médios do *throughput*, *goodput*, RTT, *timeout* e *cwnd*.

Todos os arquivos descritos acima e os demais arquivos gerados pelo pós-processamento são guardados no diretório criado para o usuário, seguindo o padrão */TeCePe/scripts/projeto/usuario/cod_simulacao*, onde o projeto é informado no cadastro do usuário e o código da simulação é gerado automaticamente

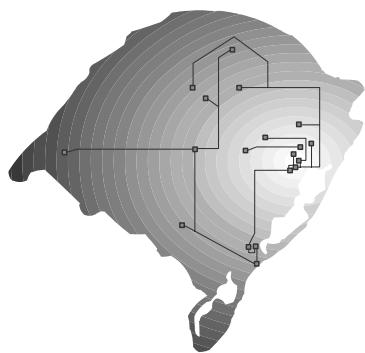
4 Considerações Finais

A ferramenta proposta permite o estudo em conjunto dos mecanismos de gerenciamento de filas, das diversas implementações TCP e do suporte a ECN de forma bastante simples e sem exigir conhecimentos aprofundados do usuário no que diz respeito ao simulador NS, que por sua vez exige outros conhecimentos como a linguagem TCL, necessária para geração dos scripts. Além disso, como a ferramenta foi desenvolvida em JSP, é possível utilizá-la através da Internet, possibilitando o acesso remoto a qualquer funcionalidade da mesma. Os resultados gerados pelas simulações, principalmente os gráficos, oferecem uma importante ajuda no estudo dos mecanismos de rede implementados, bem como nos possíveis mecanismos implementados futuramente.

Como trabalhos futuros propõe-se a automatização da ferramenta para inserção de novos mecanismos e o aumento da flexibilidade da topologia, permitindo assim a variação no número de nós, inserção de mais enlaces gargalo, etc.

Referências

- Allman, M.; Balakrishnan, H.; Floyd, S. “Enhancing TCP's Loss Recovery Using Limited Transmit”. RFC 3042, 2001.
- Allman, M.; Paxson, V.; Stevens, W. “TCP Congestion Control”. RFC 2581, 1999.
- Athuraliya, S. at al. “REM: Active Queue Management”. IEEE Network, Volume:15 Issue:3 2001. pp 48-53.
- Fall, Kevin; Varadhan, Kannan. “The ns Manual”. Disponível em <<http://www.isi.edu/nsnam/ns/doc/index.html>>.
- Floyd, S. at al. “Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management”, 2001.
- Floyd, S.; Henderson, T. “The NewReno Modification to TCP's Fast Recovery Algorithm”. RFC 2582, 1999.
- Floyd, S.; Jacobson, V. “Random Early Detection gateways for Congestion Avoidance”. IEEE/ACM Transactions on Networking, V.1 N.4, 1993, p. 397-413.
- Lima, Michele. M. A. E.; FONSECA, Nelson. L. S. “Controle de Tráfego Internet”. In: Congresso da Sociedade Brasileira de Computação, XXII, 2002, Florianópolis.
- Lin, D and Morris, R. “Dynamics of Random Early Detection”. In the Proceedings of SIGCOMM'97, 1997.
- Mathis, M. at al. “TCP Selective Acknowledgment Options”. RFC 2018, 1996.
- Ramakrishnan, K.; Floyd, S.; Black, D. “The Addition of Explicit Congestion Notification (ECN) to IP”. RFC 3168, 2001.



Sessão Técnica 10

Redes P2P e Grid I

Escambo: Um Modelo de Comportamento e Reputação para Sistemas Peer-to-Peer

Rafael da Rosa Righi, Felipe Rolim Pellissari, Carla Merkle Westphall

¹ Programa de Pós-Graduação em Ciência da Computação – PPGCC
Laboratório de Redes e Gerência (LRG) – Universidade Federal de Santa Catarina
Caixa Postal 476 - 88040-900, Florianópolis, SC

{rrighi,rolim,carla}@lrg.ufsc.br

Resumo. Os sistemas colaborativos Peer-to-Peer apresentam uma forma de computação colaborativa onde cada participante atua como cliente e servidor de recursos. Entre os principais desafios existentes nesse tipo de computação estão o desenvolvimento de técnicas para incentivar a colaboração dos usuários e minimizar o número de nós caronas que não auxiliam a rede e apenas sugam seus recursos. Este artigo define um modelo baseado em micropagamentos e na análise da reputação que objetiva aumentar a eficiência da rede Peer-to-Peer e diminuir o número de membros parasitas que a compõe. O protótipo desenvolvido valida o modelo e coloca em prática suas principais idéias.

1. Introdução

As redes Peer-to-Peer são sistemas distribuídos sem controle centralizado ou organização hierárquica, nas quais o programa que é executado em cada elemento é equivalente em funcionalidade. Esses sistemas possibilitam que os usuários sejam, além de consumidores de recursos, os próprios responsáveis por disponibilizá-los. Por minimizar o papel dos elementos centralizadores, os sistemas P2P tendem a ser imunes à censura, monopólios, regulamentos e outros exercícios atribuídos às autoridades centralizadoras [Agre, 2003].

O sucesso de uma rede Peer-to-Peer depende de fatores como o protocolo de comunicação utilizado, arquitetura de distribuição empregada (totalmente descentralizada ou não), tempo decorrido desde uma solicitação até o recebimento de respostas e o número de usuários participantes do sistema. Essa última característica é especialmente importante, pois é ela quem determina o tamanho da comunidade virtual e o volume de recursos presente na rede Peer-to-Peer.

Um dos problemas que os sistemas colaborativos enfrentam é a existência de usuários caronas (*free riders*), também chamados de parasitas ou sanguessugas, os quais não agregam nenhum valor à rede Peer-to-Peer, servindo somente para aumentar o congestionamento dos enlaces e concentrar as conexões sob aqueles nós que dispõem recursos no sistema [Strulo, 2004]. Os impactos dos caronas são diferentes nas várias arquiteturas Peer-to-Peer existentes. Por exemplo, na rede P2P Gnutella, além dos problemas citados, os caronas também contribuem para ao aumento do tempo de roteamento das solicitações, pois como nunca irão responder positivamente a um chamado, sempre passarão para seus vizinhos os pedidos por informações que recebem.

Os pesquisadores Adar e Huberman [Adar e Huberman, 2000] descobriram em seus estudos que quase 70 por cento dos usuários não compartilham recursos em uma rede Peer-to-Peer e aproximadamente 50 por cento de todas as respostas são retornadas por 1 por cento dos hospedeiros compartilhadores. Assim, eles confirmam a necessidade de haver mecanismos que incentivam a colaboração nas redes Peer-to-Peer e, consequentemente, as tornem mais justas e igualitárias.

O modelo Escambo exposto neste artigo define medidas que incentivam os usuários a disponibilizarem recursos na rede Peer-to-Peer e, como consequência, ele auxilia para atenuar o transtorno dos nós parasitas. Para chegar nesse objetivo o Escambo utiliza técnicas de micropagamentos (atribui um “preço” às comunicações) e reputação, juntamente com a seguinte premissa: “quem deseja informações da rede deve necessariamente também disponibilizar recursos”. O artigo apresenta também as técnicas utilizadas para proteger o modelo Escambo de usuários mal-intencionados (aqueles que desejam subverter o protocolo em benefício próprio) e o protótipo P2P desenvolvido durante a pesquisa.

O artigo está organizado em 4 seções. A seção 2 é responsável por exibir o modelo Escambo, como ele se protege de usuários maliciosos no ambiente Peer-to-Peer e os principais trabalhos relacionados com o tema pesquisado. A seção 3 descreve a aplicação construída para legitimar o modelo desenvolvido e aborda a maneira como as arquiteturas P2P podem se beneficiar do Escambo. O artigo encerra na seção 4 com a conclusão, a qual reúne as principais idéias e resultados da pesquisa, além de citar os possíveis complementos sobre ela, a cargo de trabalhos futuros.

2. Modelo Escambo

O modelo Escambo é responsável por controlar o fluxo de recursos entre os participantes da rede Peer-to-Peer. Ele baseia-se na idéia de troca de recursos, onde um nó apenas adquire acesso aos recursos que deseja caso possua outros recursos para disponibilizar no ambiente colaborativo. Dessa forma, participantes que são parasitas - aqueles que não colaboram com os demais - têm acesso restrito às vantagens que a rede proporciona e, então, são encorajados a saírem da condição de caronas para desfrutarem integralmente da comunidade Peer-to-Peer. O modelo Escambo modifica a estrutura normal das redes Peer-to-Peer para alcançar os seus objetivos. Nele, um nó que recebe uma solicitação por um recurso que detém deve, antes de permitir o acesso, verificar se o nó requisitor também dispõe recursos na rede P2P. O Escambo oferece uma sistemática que possibilita reconhecer quais são os nós caronas e quais não são.

O desenvolvimento de métodos para incentivar a cooperação entre os usuários da rede Peer-to-Peer está em constante pesquisa pela comunidade científica. Entre as técnicas existentes para solucionar o problema está a atribuição de preços aos recursos – utilizada no protocolo Mojo Nation [McCoy, 2002]. Nesse cenário, os nós que colaboram com o sistema P2P ganham mais dinheiro virtual; fato que estimula o compartilhamento de recursos¹. Outro trabalho relacionado é o *middleware* desenvolvido por Strulo [Strulo, 2004]. Ele divide os participantes da rede P2P em grupos e, nesse grupos, cada líder é responsável por definir regras de comportamento a serem seguidas pelos demais

¹O maior problema de utilizar dinheiro virtual é quantificar com clareza quanto vale cada recurso.

integrantes. O modelo Escambo une as principais vantagens dos dois métodos anteriores, além de acrescentar suas próprias idéias para a solução do problema dos nós caronas e da falta de colaboração nas redes Peer-to-Peer.

Para apresentar o funcionamento do Escambo, este artigo simula a procura de uma informação em uma rede Peer-to-Peer totalmente descentralizada. O nó cliente faz uma requisição por determinado dado e esse pedido é repassado para os seus vizinhos. Se o vizinho não possuir o dado, ele re-envia o pedido para frente. Caso contrário, ele deve informar uma resposta positiva (ela pode acontecer através do roteamento normal da internet ou através do caminho inverso percorrido pela solicitação). O Escambo define que junto com a resposta, o nó “servidor” deve também comunicar a sua **política de permissão aos recursos**.

A política de permissão aos recursos informa quais as condições que o nó cliente deve suprir para acessar os recursos deste nó servidor. Ela se divide em três categorias: (i) tamanho total dos recursos compartilhados (medido em bytes); (ii) número de arquivos oferecidos; (iii) tipo de recursos disponibilizados. Um exemplo de política de permissão é a seguinte: “os usuários que desejam acesso aos meus recursos devem compartilhar na rede no mínimo 1 Megabyte e 4 arquivos”.

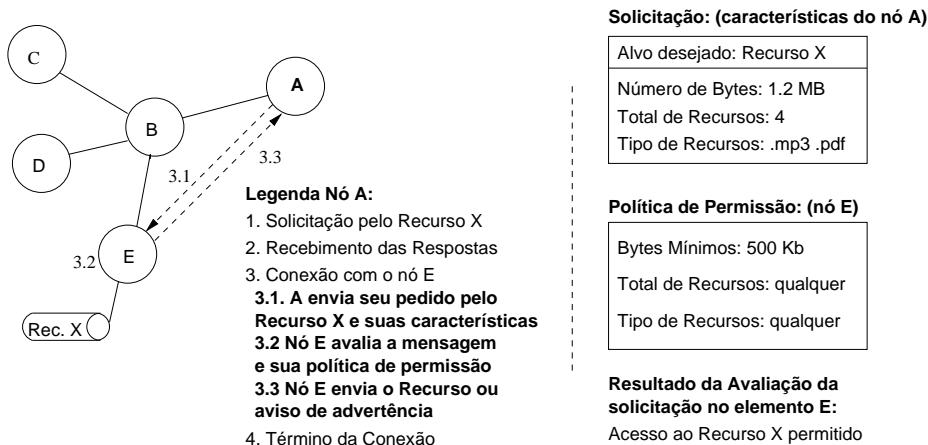


Figura 1: Rede Peer-to-Peer e o Modelo Escambo

O participante cliente recebe todas as respostas e analisa qual lhe parece a melhor (verifica as políticas que se encaixam em suas características) e abre uma conexão direta com o outro ponto - computação Peer-to-Peer - passando o identificador do recurso que almeja e as características dos recursos que oferece para o sistema colaborativo, assinalando o processo de reputação. O ponto servidor, ao receber a chamada por recursos, verifica a reputação do cliente avaliando suas categorias e, logo após, permite ou não o acesso aos seus recursos. Se as características dos recursos que o cliente compartilha não foram suficientes para lhe permitir o direito de acesso, ele pode tentar uma conexão a outro nó da rede com uma política menos restritiva ou disponibilizar mais recursos - fato que aumenta a sua reputação e favorece a qualidade da rede Peer-to-Peer. A Figura 1 apresenta a utilização do modelo Escambo em uma rede Peer-to-Peer. Nela estão presentes os identificadores da etapa de acesso ao recurso na rede. Os passos da procura por informações no sistema Peer-to-Peer encontram-se somente na legenda.

Para evitar que participantes maliciosos deturpem o modelo Escambo, é determi-

nado um nível máximo e outro mínimo para a política de permissão de acesso aos recursos. O grau mínimo não estabelece verificação e nele o sistema P2P se comporta como se não existisse o Escambo. O grau máximo é definido para prevenir que os usuários que administram os pontos P2P coloquem níveis “impossíveis” ou incomuns de exigências para a distribuição de seus recursos (ex: estabelecer uma política que requer que o cliente disponibilize 10 Gigabytes para a rede Peer-to-Peer). Uma política de permissão exemplo e o resultado de uma avaliação de reputação também estão presentes na Figura 1.

O Escambo utiliza um modo simplificado de micropagamento [Yang e Garcia-Molina, 2003]. O micropagamento introduz o conceito de pagamento pelo acesso a um recurso ou pedido de atividade. Esse pagamento pode ser de dois tipos: (*i*) aquele em que o nó servidor não recebe nenhum valor e o cliente paga-o geralmente com trabalho (cálculo de uma tarefa computacional complexa); (*ii*) o cliente utiliza algum sistema de pagamento (ex: dinheiro virtual) para pagar o detentor dos recursos, o qual tem acesso ao montante recebido. No caso específico do Escambo, o nó cliente paga o servidor através da oferta de recursos na rede Peer-to-Peer.

A reputação nos sistemas colaborativos Peer-to-Peer tradicionais informam quais participantes da rede são honestos ou bom servidores de recursos. A reputação dos nós é adquirida através das experiências dos próprios membros da rede e das trocas de informações de reputação entre os pares que confiam um no outro [Marti e Garcia-Molina, 2003]. O conceito de reputação encontrado no Escambo distancia-se do mencionado anteriormente. Nele, a reputação de cada nó está associada à quantidade e qualidade do material que ele disponibiliza na rede Peer-to-Peer e não depende da opinião de outros participantes do sistema. Um nó da rede Peer-to-Peer é o único responsável por sua reputação.

Além da verificação de **políticas de permissão** impróprias nos elementos da rede Peer-to-Peer, o modelo Escambo deve também evitar a falsificação das informações de reputação. O Escambo deve estar precavido contra usuários que alteram as mensagens passadas entre os nós em benefício próprio (um usuário não deve se passar por grande compartilhador de recursos quando na verdade não é). As seguintes medidas podem ser adotadas para colocar segurança no modelo definido: (*i*) ao solicitar um recurso, o cliente P2P passa, junto com a sua reputação, a lista de todos os recursos que oferece no ambiente; (*ii*) o servidor, além de aplicar sua política de permissão, escolhe um recurso aleatório na lista passada pelo cliente e tenta encontrá-lo na rede. Se ele encontrar há bons indícios que esse cliente seja honesto e a operação pode prosseguir (a anonimidade da busca na rede é essencial nesse processo). Essas ações auxiliam para a robustez do modelo Escambo e desencorajam o “atacante” a alterar os dados sobre seus recursos - sua reputação -, já que essa atitude apenas irá prejudicá-lo.

3. Aplicação Escambo

Esta seção apresenta o desenvolvimento do protótipo Escambo, o qual baseia sua operação no modelo de mesmo nome descrito na seção 2. O programa construído foi escrito na linguagem Java, é composto por seis classes e sua estrutura principal é semelhante àquela encontrada na aplicação P2P-Role [Righi et al., 2004], a qual define uma arquitetura de controle de acesso para redes Peer-to-Peer. As classes ControlClient e

ControlServer representam os módulos cliente e servidor existentes em cada nó P2P e estendem a classe Thread, ou seja, elas se comportam como fluxos de execução independentes. A classe Resource é responsável por duas tarefas: (i) a comparação da política de permissão do próprio nó com as características dos recursos do cliente P2P; (ii) capturar, para cada comunicação por busca de informações, a quantidade de bytes compartilhados, o número de recursos e seus tipos. Como mencionado no modelo conceitual, esses três itens acompanham a mensagem de solicitação enviada ao nó servidor.

A Figura 2 apresenta as classes que compõe um elemento na aplicação Escambo e um exemplo de conexões entre seus participantes (observe o sentido das conexões). O fluxo fornecedor de recursos abre um soquete do tipo servidor e espera por conexões na porta declarada na classe Configuration. Já o fluxo cliente conecta-se ao fluxo servidor do nó alvo e pede para o usuário digitar o nome do recurso (ou seu identificador) que procura. Logo após, o fluxo cliente chama a classe Resource para obter as informações que caracterizam o material oferecido pelo nó cliente à rede colaborativa. A mensagem enviada ao fluxo servidor é composta pelos itens mencionados anteriormente. A partir desse instante, o fluxo cliente permanece na espera por uma resposta do fluxo servidor. A resposta pode ser uma mensagem de advertência ou o recurso propriamente.

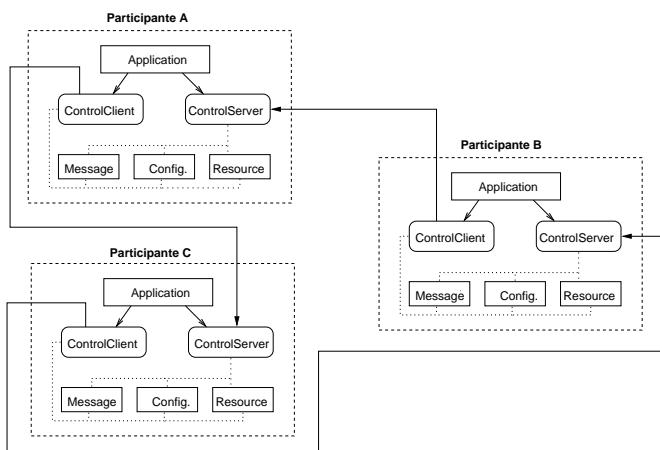


Figura 2: Classes Java existentes nos elementos da rede Peer-to-Peer

O fluxo servidor recebe a mensagem de solicitação da outra ponta da conexão e açãoa sua classe Resource, a qual determina se ele permite ou não o acesso a seu recurso. Nesse ponto o nó servidor verifica se o cliente se parece com um participante carona ou não. Caso a decisão seja de conceder o recurso, o mesmo é transportado até o participante cliente. Se o parecer for de negação, é comunicado o motivo de tal atitude. O motivo do bloqueio de acesso geralmente está associado ao não cumprimento das exigências mínimas da **política de permissão** adotada pelo servidor.

Cada membro da rede virtual possui sua própria política de permissão de recursos. Essa peculiaridade possibilita a existência de nós mais rígidos e outros mais permissivos. O Escambo trabalha com um limite inferior e outro superior para a política de permissão. Nesta implementação, o nível mínimo define que o fluxo servidor aceita todas as solicitações que chegam até ele (a ação é sempre permitir). No nível máximo a regra é permitir downloads dos clientes que tenham no mínimo 600 kilobytes de informações e 5 recursos (arquivos texto).

4. Conclusão

As redes Peer-to-Peer possibilitam a colaboração entre os elementos da rede e seu potencial é imenso, desde o compartilhamento de recursos até o comércio eletrônico entre organizações. O sucesso dessa classe de sistemas distribuídos está associado à responsabilidade e ao comportamento de seus membros. O modelo Escambo exposto nesse artigo colabora para a otimização do ambiente colaborativo através da redução no número de nós caronas da rede. E, consequentemente, auxilia para o aumento do volume de recursos no sistema e para a construção de uma comunidade de usuários comprometidos em colaborar com os demais.

A idéia principal adotada pelo Escambo baseou-se nos requisitos necessários para a troca de recursos. Nesse modelo os usuários que não compartilham recursos possuem acesso restrito às vantagens da rede Peer-to-Peer. Para acessar integralmente os recursos dispostos no sistema é necessário primeiro a colaboração com mais recursos. O Escambo definiu um esquema para o controle de participantes “parasitas” que se apóia nas arquiteturas de micropagamentos e reputação em aplicações P2P. Ele preocupou-se sobretudo com a segurança nas transações de reputação, especialmente em não permitir que usuários mal intencionados forjem as mensagens do modelo em benefício próprio.

O protótipo desenvolvido ajudou no entendimento dos conceitos existentes no modelo Escambo. Seu conjunto de classes e métodos podem servir para nortear a escrita de outras aplicações P2P, principalmente no âmbito acadêmico. Finalmente, como sugestão para trabalhos futuros indica-se a realização de comparações entre ambientes P2P com e sem o modelo Escambo. A diminuição no número de mensagens trocadas no Escambo e a avaliação de seu desempenho também apresentam-se como pesquisas futuras.

Referências

- Adar, E. e Huberman, B. (2000). Free Riding on Gnutella. Technical report, Xerox Palo Alto Research Center. Available: <http://citeseer.ist.psu.edu/adar00free.html>.
- Agre, P. E. (2003). P2P and the Promise of Internet Equality. *Communications of the ACM*, 46(2):39–42.
- Marti, S. e Garcia-Molina, H. (2003). Identity Crisis: Anonymity vs. Reputation in P2P Systems. In: *Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P'03)*, pages 134–141. Linkoping, Sweden.
- McCoy, J. (2002). Mojo Nation Responds. Available on-line: <http://www.openp2p.com/pub/a/p2p/2001/01/11/mojo.html>.
- Righi, R., Pellissari, F. e Westphall, C. (2004). P2P-Role: Uma arquitetura de Controle de Acesso Baseada em Papéis para Sistemas Colaborativos Peer-to-Peer. In: *IV Workshop de Segurança de Sistemas Computacionais (WSeg / SBRC)*, pages 285–296. Gramado, RS. ISBN: 85-88442-84-1.
- Strulo, B. (2004). Middleware to Motivate Co-operation in Peer-to-Peer Systems. *Peer-to-Peer Journal (P2PJ)*, 1(5):1–12. <http://www.p2pjurnal.com>.
- Yang, B. e Garcia-Molina, H. (2003). PPay: micropayments for peer-to-peer systems. In: *Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS)*, pages 300–310. Washington D.C., USA.

Controlando Tráfego Peer-to-Peer

Mell Fogliatto, Emerson Virti, Leandro Bertholdo, Liane Tarouco

Ponto de Presença da Rede Nacional de Ensino e Pesquisa do RS – POP-RS
Computer Emergency Response Team Rio Grande do Sul – CERT-RS
Centro de Processamento de Dados Universidade Federal do RS – UFRGS
Ramiro Barcellos, 2574 – Porto Alegre – RS – Brasil
`{mell,emerson,leandro,liane}@penta.ufrgs.br`

Abstract: This article provides an analysis of the main negative aspects of peer-to-peer (P2P) applications and their effects on the academic network of Rio Grande do Sul (Rede Tchê). We'll present a way to count and classify this kind of traffic by using the software Netflow and the NBAR resource, found in Cisco equipments. This paper also contains a sample of the volume of incidents related to distribution of illegal material registered with the Computer Emergency Response Team Rio Grande do Sul (CERT-RS). The solutions which were adopted by the state academic network and by the Universidade Federal do Rio Grande do Sul (UFRGS) to control traffic P2P will be broached as well. In addition, two new P2P systems that facilitate the exchange of information between users will be described.

Resumo: Nesse artigo serão analisados os principais aspectos negativos de aplicações peer-to-peer (P2P) e seus efeitos na rede acadêmica do Rio Grande do Sul. Será demonstrada uma forma de classificação e contabilização desse tráfego utilizando o software Netflow e o recurso NBAR existente em equipamentos Cisco. Esse trabalho contém também uma amostra do volume de incidentes relacionados à distribuição de material ilegal registrados junto ao Computer Emergency Response Team Rio Grande do Sul (CERT-RS). Serão também abordadas soluções utilizadas para tratamento do tráfego P2P na rede acadêmica estadual (Rede Tchê) e na Universidade Federal do Rio Grande do Sul (UFRGS). Nesse trabalho também descreveremos dois sistemas P2P que podem beneficiar a troca de informações entre usuários.

1. Introdução

O uso bastante difundido de aplicações P2P como meio de troca de arquivos MP3 e filmes, por exemplo, causa problemas aos administradores de redes como grande utilização de tráfego e o crescente número de reclamações referentes à violação de Copyright. As inúmeras aplicações e variedades de protocolos utilizados dificulta o controle e/ou bloqueio das mesmas.

Nesse trabalho serão abordados os principais impactos e riscos de tráfego P2P, para a rede e hosts, assim como uma forma de classificação e contabilização desse tráfego. São apresentados dados relacionados à distribuição de material ilegal.

Para mostrar como a arquitetura P2P pode ser utilizada para fins acadêmicos, serão apresentados dois exemplos do bom uso desta, incluindo um sistema de Voz sobre IP

(VoIP) que a utiliza. Finalmente, será explicada a experiência de controle adotada pela UFRGS.

2. Impactos e Riscos

Em 1998, o estudante Shawn Fanning, da Universidade de Massachussets, criou um programa para facilitar a troca de arquivos MP3 com seus colegas – o conhecido Napster. Desde então, esse tipo de programa tem se difundido consideravelmente e inúmeros outros softwares foram criados, dentre eles: Bearshare, BitTorrent, Earthstation, eDonkey, eMule, iMesh, KazaA, MIMac, SoulSeek, WinMX.

Logo quando foi criado o Napster, o bloqueio para as redes P2P era bastante simples. Bastava bloquear o acesso aos servidores que armazenavam os índices de arquivos compartilhados. Após uma melhoria nos protocolos P2P, esses servidores passaram a ser desnecessários e a filtragem do serviço tornou-se bem mais complexa, pois não poderia mais ser feita através do bloqueio dos IPs.

2.1. Vírus, Worms e Spywares

Uma das maiores preocupações atualmente são os vírus e worms que utilizam a rede P2P como meio de propagação.

O primeiro deles foi o Slapper que contaminou mais de 14.000 servidores Linux em 2002 (Symantec). Atualmente existem mais de 300 vírus/worms relacionados às aplicações P2P (Symantec). A maioria deles copia-se automaticamente para os diretórios que contenham a palavra ‘SHARE’ no nome, normalmente utilizados pelas aplicações em questão. Um exemplo seria o worm W32.HLLW.Sanker que, numa tentativa de aumentar as chances de download/propagação, assume nomes de filmes, programas, cracks, etc. A propagação de vírus via rede P2P é considerada extremamente eficiente e perigosa, já que os arquivos só podem ser verificados após o download completo dos mesmos.

Algumas aplicações, como Kazaa ou iMesh, também oferecem outros tipos de risco para os usuários por, em algumas versões, instalarem spywares acoplados ao programa. Spyware é qualquer programa que monitorea ações do usuário na internet e, sem que o usuário tenha conhecimento, transmite-as para um local pré-determinado. São informações normalmente colhidas para fins de propaganda, mas podem ser utilizadas ilicitamente.

2.2. Utilização dos Recursos da Rede

Outro problema é o elevado consumo de banda. Considerando estatísticas pesquisadas, o percentual de banda utilizado por estas aplicações – em redes que não possuem nenhum tipo de controle – pode chegar a 60% de toda a utilização da rede. Um exemplo mais preocupante seria a University of Florida que detectou uma queda de 85% na utilização da rede, após o bloqueio de aplicações P2P.

Como tentativa de contabilizar o consumo de banda, de forma a ter dados mais condizentes com a realidade regional, foram realizados alguns testes.

O primeiro deles consistiu em introduzir na rede uma máquina com uma aplicação P2P disponibilizando alguns filmes e músicas para download. Esse host foi deixado na rede por uma hora e meia e, durante esse tempo, essa máquina conseguiu gerar

aproximadamente 10 Mbps de tráfego, totalizando 16 Mbps. No momento em que o host foi desconectado, o tráfego retorna imediatamente ao normal, ou seja, em torno de 6 Mbps conforme mostra a Figura 1.

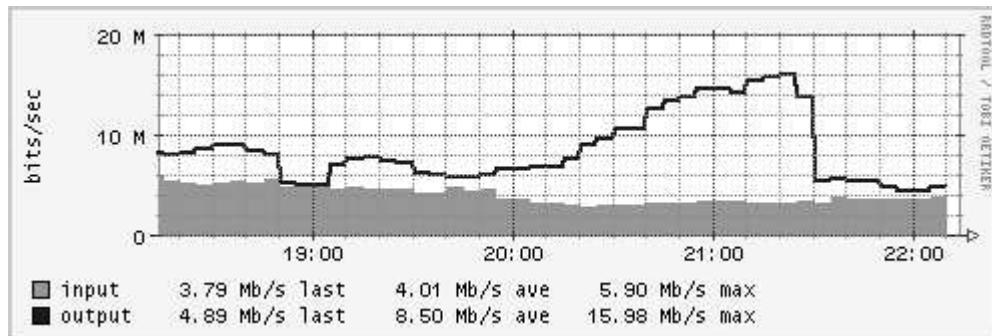


Figura 1. Gráfico demonstrando o tráfego gerado por uma aplicação P2P

A Figura 2 demonstra o impacto das redes P2P no tráfego normal de uma instituição. Às 18:45, é introduzido um filtro na rede e o mesmo é retirado às 19:05. Pode-se constatar uma queda de aproximadamente 3 Mbps.

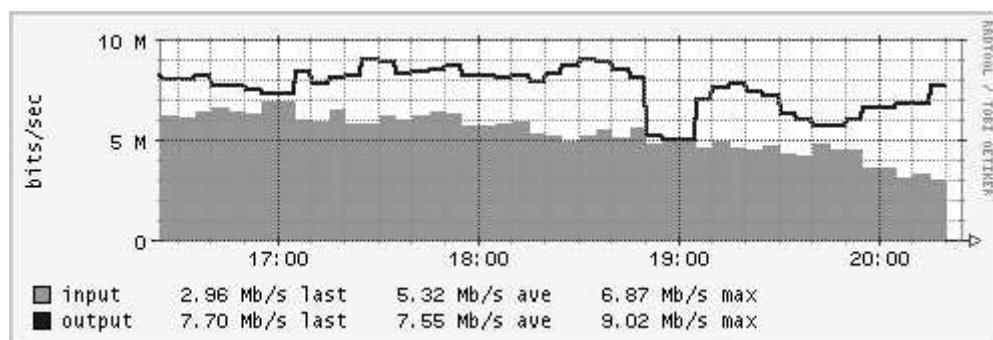


Figura 2. Gráfico demonstrando grande queda de tráfego quando filtradas aplicações P2P

2.3. Incidentes envolvendo distribuição ilegal

Através de um registro das ocorrências relevantes à segurança da rede realizado pelo CERT-RS, a cada trimestre é feita uma análise com base nesses dados e atualmente os incidentes em maior evidência são aqueles relacionados à violação de Copyright, ou seja, distribuição de material ilegal. Notou-se no último ano um aumento considerável nas formalizações de reclamações e denúncias. Isso é demonstrado na figura 3.

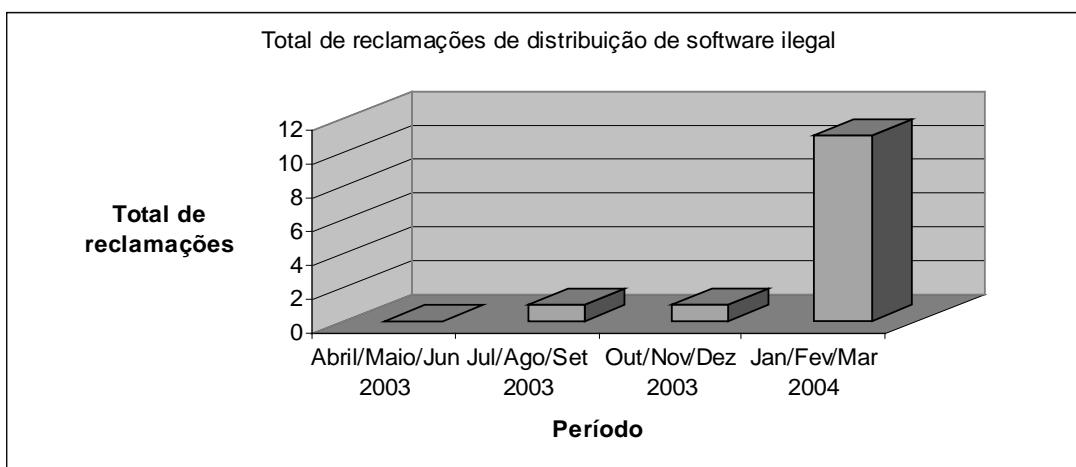


Figura 3. Gráfico demonstrando o aumento de formalizações de denúncias relacionadas à violação de Copyright

Vale ressaltar que algumas instituições americanas chegam a contabilizar cerca de 200 reclamações mensais.

3. Mensurando o Tráfego

Para fins de teste, foi escolhida uma instituição da rede Tchê que não fazia controle sobre aplicações P2P. Um dado significativo é que essa instituição havia dobrado a sua largura de banda – para 8 Mbps – em apenas um ano e meio.

Como a utilização de portas pelas aplicações é dinâmica, foi utilizado para análise o recurso NBAR – disponível em equipamentos Cisco – que, através da análise dos pacotes que trafegam pelo roteador, consegue identificar o tráfego de algumas aplicações P2P, incluindo Blubster, eDonkey, iMesh, Kazza Lite, LimeWire e Morpheus (aplicações testadas pela Cisco Systems). Através desse recurso foi possível implementar uma marcação nos pacotes P2P, utilizando o campo DSCP, para futura análise através do software Netflow. Também foi possível realizar uma filtragem de aplicações P2P no tráfego da referida instituição. Os resultados são demonstrados na Figura 4.

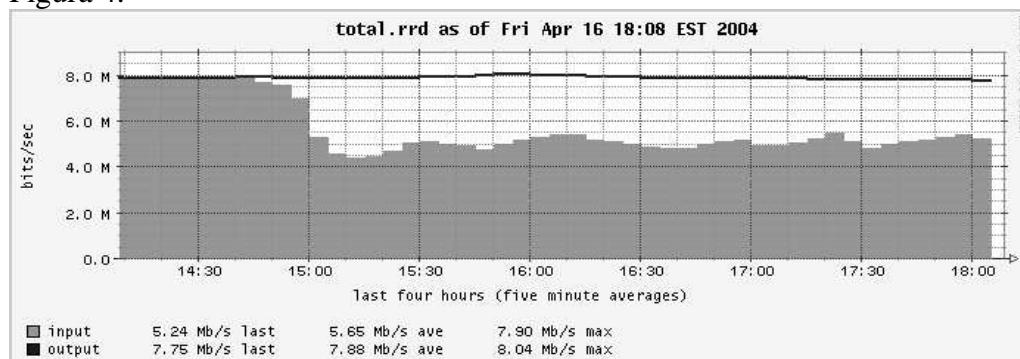


Figura 4. Gráfico demonstrando grande queda no tráfego de entrada quando filtradas aplicações P2P

4. Bom uso de Aplicações P2P

O bom uso da rede P2P permite várias novas aplicações como o XBrain (em desenvolvimento) e o Skype. Ambas serão analisadas a seguir.

4.1. Xbrain

Sistemas como o XBrain, utilizam a estrutura e protocolos P2P para indexar dados dos usuários a partir de servidores localizados na Rede Nacional de Ensino e Pesquisa (RNP). Funciona como um serviço para encontrar pessoas ou grupos que tenham possibilidade de oferecer ajuda em determinadas áreas de conhecimento previamente cadastradas.

As informações são organizadas em meta-dados (apontadores) que indexam dados dos usuários e a localização de recursos. Esses dados são armazenados em dois servidores diferentes, para que não haja possibilidade de perda de informações. Quando os usuários conectam-se em um dos servidores (chamados de XPeer), são mantidos registros de log para que a busca de pessoas seja local e, portanto, mais rápida. Segue ilustração da referida arquitetura.

Infra-estrutura e Aplicações

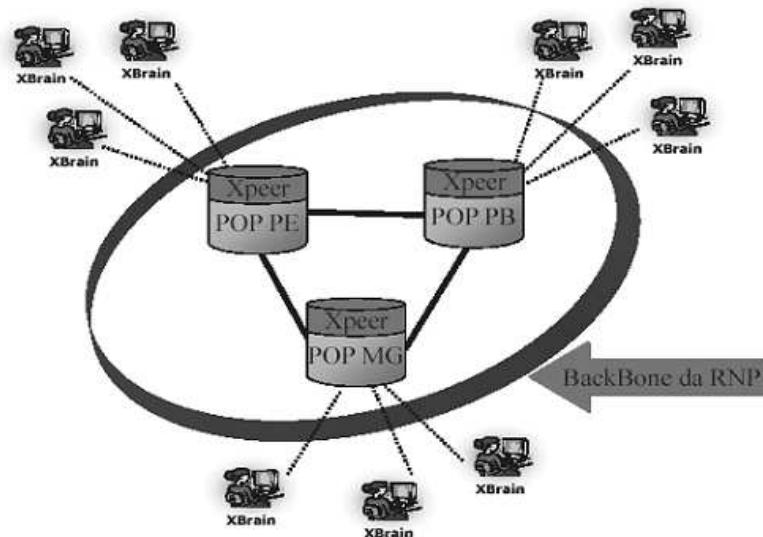


Figura 5. Arquitetura do Sistema XBrain

4.2. Skype

O Skype é um programa de telefonia sobre IP (VoIP) que utiliza a mesma arquitetura P2P do programa KazaA. Os dados dos usuários são armazenados em servidores chamados de supernodes. Quando é feita uma requisição de ligação, o programa busca o nome de usuário requerido na rede e são retornadas as informações necessárias para que se realize a conexão direta entre os usuários. Os dados são cifrados, garantindo privacidade nas conexões.

Possui um protocolo proprietário para transferência de dados onde o uso de portas é randômico, utiliza qualquer porta aberta acima de 1024, caso não encontre nenhuma livre, utiliza a porta 80 (porta normalmente utilizada para tráfego HTTP). O protocolo preferencial é UDP, mas pode utilizar também TCP.

5. A experiência da UFRGS

A solução adotada pela UFRGS baseia-se na conscientização dos usuários da rede acadêmica e no bloqueio dos IPs de máquinas reincidentes. Foi realizada uma reunião com os gerentes das diversas unidades da universidade a fim de divulgar a política de uso aceitável dos recursos computacionais.

O controle é feito através da combinação de scripts que utilizam a ferramenta ngrep para inspeção dos pacotes da rede. Em intervalos de 30 minutos, os scripts são executados e obtém informações das conexões da aplicação KazaA. Apartir dos dados armazenados, são gerados relatórios diários contendo os IPs coletados. Os relatórios são enviados aos gerentes das unidades e esses se encarregam de contactar os usuários a fim de alertá-los e conscientizá-los do impacto causado na rede da universidade. No caso de reincidência, o IP é automaticamente bloqueado. O desbloqueio é realizado através de contato com a Central de Atendimento.

6. Conclusão

Considerando os dados analisados, constatamos que as aplicações P2P são amplamente utilizadas no meio acadêmico e que esse uso (quando direcionado para troca de arquivos áudio-visuais) acarreta significativos problemas, como a diminuição da qualidade da rede, tendo em vista a elevada utilização da mesma, e o crescente número de reclamações de compartilhamento de arquivos ilegais.

Apesar disso, aplicações P2P podem ser benéficas para as instituições da rede acadêmica, se utilizadas com sabedoria. Como foi mostrado, podem ser desenvolvidas aplicações P2P para proporcionar maior interatividade entre os usuários e beneficiar o aprendizado.

Referências:

- Extreme Networks, ‘Meeting the Peer-to-Peer (P2P) Challenge in Higher Education – An Overview’,
http://www.extremenetworks.com/common/asp/frameHandler.asp?go=/LIBRARIES/whitepapers/technology/MeetingP2PChallenges_WP.pdf
- Extreme Networks, ‘Meeting the Peer-to-Peer (P2P) Challenge in Higher Education – A Network Designer’s View’,
http://www.extremenetworks.com/common/asp/frameHandler.asp?go=/LIBRARIES/whitepapers/technology/P2P_NDV_WP.pdf
- RNP, Grupo de Trabalho de P2P da RNP, <http://www.cin.ufpe.br/~gprt/gtp2p/>
- Gorgonio Araújo, ‘Tecnologias P2P e seu impacto na rede’,
http://www.rnp.br/_arquivo/sci/2003/p2p.pdf

Redes Peer-to-Peer como Ferramenta para o Gerenciamento Cooperativo de Redes

Diego Moreira da Rosa¹

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

diegoro@inf.ufrgs.br

Resumo. Popularizadas por ferramentas de compartilhamento de arquivos como Napster e Kazaa, as redes Peer-to-Peer vêm sendo utilizadas por um número cada vez maior de aplicações. Ao quebrar as barreiras impostas pelo tradicional modelo cliente-servidor, esses sistemas têm permitido a cooperação entre estações outrora isoladas em suas redes locais. Este trabalho tem por objetivo discutir a utilização de redes Peer-to-Peer na implementação de sistemas de gerenciamento cooperativo de redes, analisando como essa nova tecnologia e suas características poderiam auxiliar nas tarefas de gerência.

1. Introdução

A crescente complexidade das redes de comunicações de voz e dados bem como o aumento de sua popularidade e importância tem tornado cada vez mais crítica a tarefa de gerenciamento das mesmas. Ao mesmo tempo que o número de usuários aumenta, aumentam também a banda e a quantidade de serviços oferecidos pelas empresas de telecomunicações. Junto a isso, tem-se a popularização de uma variada gama de aplicações, as quais passam a exigir cada vez mais da rede instalada: videoconferências, comunicações de voz sobre IP, jogos em rede, etc. Somando-se ainda a heterogeneidade dos atuais sistemas de comunicações, torna-se claro que o tradicional paradigma de gerenciamento centralizado não será capaz de suprir necessidades emergentes das prestadoras de serviços, como a alocação de banda através de diferentes domínios, por exemplo [Calisti e Faltings 2000].

A partir da metade dos anos 90, nota-se uma crescente preocupação em se propor paradigmas de gerenciamento diferenciados do tradicional paradigma centralizado [Meyer et al 1995]. Nesse contexto, surgem as primeiras propostas de paradigmas de gerenciamento distribuído, como os paradigmas hierárquico e cooperativo [Goldszmidt e Yemini 1993]. Inicialmente baseadas no gerenciamento por delegação e na interação entre agentes inteligentes [Willmott e Calisti 2000], essas primeiras propostas surgiram visando cobrir as deficiências de escalabilidade e flexibilidade do gerenciamento centralizado. Apesar de aparentemente solucionarem diversas questões relativas à gerência de redes, esses modelos sofreram com a crescente dificuldade de comunicação entre dispositivos localizados em diferentes domínios administrativos. Mais recentemente, a introdução de novas tecnologias como os *Web Services* tem amenizado esses problemas de comunicação [Schönwälter et al 2003].

Outra tecnologia que tem sido muito utilizada para a comunicação entre estações localizadas em diferentes domínios são as redes Peer-to-Peer (P2P). Mundialmente popularizadas através de programas de troca de arquivos como Napster (www.napster.com) e Kazaa (www.kazaa.com), as redes P2P vêm sendo utilizadas por

um número cada vez maior de aplicações [Gao 2003], além de despertarem um interesse crescente junto aos grupos de pesquisa [Oram 2001]. Apesar desse rápido desenvolvimento dos sistemas P2P e das necessidades atuais do gerenciamento de redes expostas anteriormente, não foram encontrados, ao longo da elaboração desse trabalho, registros de pesquisas visando a utilização dessa tecnologia como ferramenta para o gerenciamento de redes.

Este trabalho tem por objetivo discutir a utilização de redes P2P como ferramenta para o gerenciamento cooperativo de redes. Será apresentada uma revisão dos principais conceitos do gerenciamento distribuído, bem como um panorama da tecnologia P2P. Espera-se ainda realizar uma análise de como as características das redes P2P poderiam auxiliar nas tarefas de gerência de redes, as quais vêm se tornando cada vez mais complexas e, ao mesmo tempo, mais importantes.

Este artigo está dividido em cinco seções incluindo esta introdução. Na seção 2, são apresentados os principais conceitos e desafios do gerenciamento cooperativo de redes. Na seção 3, é apresentado um panorama das redes P2P, passando por suas características e aplicações. Na seção 4, será apresentada uma discussão de como as redes P2P podem auxiliar no gerenciamento de redes. Por fim, na seção 5 são apresentadas algumas considerações finais.

2. Gerenciamento Cooperativo de Redes

No intuito de definir o que se entende por gerenciamento cooperativo de redes, pode-se destacar [Martin-Flatin 1997], no qual encontra-se uma tentativa de classificação dos diversos paradigmas de gerenciamento. Segundo o autor, o Gerenciamento Distribuído de Redes, ou *Distributed Network Management* (DNM), pode ser caracterizado como qualquer tipo de gerenciamento diferente do centralizado, no qual tipicamente uma única estação de gerenciamento gerencia uma série de agentes responsáveis apenas pela coleta dos dados. Mais precisamente, um sistema de gerenciamento distribuído é baseado em um modelo hierárquico ou em um modelo cooperativo, onde existe delegação entre as estações. Por delegação, entenda-se o processo de transferência de poder, autoridade e responsabilidade sobre uma tarefa para outra entidade. A delegação pode ser caracterizada ainda como vertical, no caso do modelo hierárquico, ou como horizontal, típica dos modelos cooperativos utilizados em inteligência artificial distribuída.

Existem ainda sistemas híbridos, os quais podem combinar características do modelo hierárquico com características do modelo cooperativo, como no caso de um sistema baseado em agentes inteligentes no qual um gerente delega uma atividade para um agente que coopera com outros agentes a fim de realizá-la. A Tabela 1 apresenta a classificação proposta por Martin-Flatin. Vê-se que, como exemplo de gerenciamento cooperativo, o autor apresenta apenas os sistemas baseados em agentes inteligentes.

Além do trabalho de Martin-Flatin, tem-se o trabalho de Strauss [Strauss 2000], que classifica sistemas cooperativos, entre outros critérios, de acordo com a conectividade entre os nodos. Segundo ele, em um sistema cooperativo, existe um grande número de nodos de gerenciamento, e quase não se pode observar uma hierarquia. Além disso, pode-se observar ainda uma grande conectividade entre os nodos. Nos trabalhos de ambos os autores, nota-se uma tendência em se definir gerenciamento cooperativo como um ambiente de cooperação entre aplicações, ou, mais especificamente, entre agentes.

Tabela 1. Classificação dos diversos paradigmas de gerenciamento segundo Martin-Flatin

	<i>Centralizado</i>	<i>Hierárquico</i>	<i>Cooperativo</i>
Não distribuído	SNMPv1, SNMPv2c		
Fracamente distribuído		RMON, SNMPv2, CMIP	
Fortemente distribuído		Web, código móvel, objetos distribuídos	Agentes inteligentes

Outros trabalhos também podem ser destacados na área de gerenciamento cooperativo. Ray, por exemplo, realiza um estudo sobre a cooperação entre pessoas envolvidas no gerenciamento da rede de uma grande empresa [Ray 1996]. Nesse trabalho, é destacada a importância das interações entre as pessoas envolvidas no gerenciamento. Já em [Ray 1995], o autor destaca a importância do conhecimento e dos recursos humanos no gerenciamento de redes e apresenta a modelagem de um sistema cooperativo de *troubleticketing*. Aqui, o compartilhamento de informações e a cooperação entre as pessoas se apresentam como os pontos fortes de ligação entre esse modelo de gerenciamento cooperativo mais amplo abordado por Ray e as redes Peer-to-Peer.

3. Redes Peer-to-Peer

Segundo [Gao 2003], computação P2P define uma interação recíproca e instantânea entre dois ou mais *peers*. *Peer* aqui representa qualquer equipamento conectado em rede, desde um mainframe, até um MP3 player, por exemplo. Outra caracterização clássica de redes P2P é aquela que contrasta o modelo P2P contra o modelo cliente-servidor. No modelo cliente-servidor, clientes com relativa baixa capacidade de processamento acessam informações, serviços e aplicações distribuídas através de servidores com endereços fixos e bem conhecidos e com uma capacidade de processamento maior. Esses clientes praticamente desconhecem a existência de outros clientes. Já no modelo P2P, as estações conectadas podem encontrar-se mutuamente através de esquemas de busca e comunicar-se diretamente entre si. No modelo P2P, dados, capacidade de processamento e aplicações não ficam concentrados em um único servidor, mas se encontram distribuídos ao longo dos *peers*.

Apesar da diversidade de definições existentes para sistemas P2P, em geral, é aceito pela comunidade que um sistema P2P deve suportar os seguintes requisitos [Rocha 2004]:

- *Peers* podem estar localizados nas bordas da rede;
- *Peers* com conectividade variável e endereços também variáveis;
- Capacidade de lidar com diferentes taxas de transmissão entre *peers*;
- *Peers* com autonomia parcial ou total em relação a um servidor centralizado;
- Capacidade dos *peers* comunicarem-se diretamente uns com os outros.

Como visto, é fundamental para um sistema P2P interconectar diretamente os *peers* componentes da rede. Uma questão preponderante aqui são os dispositivos de

proteção dispostos nas redes, os quais impedem a comunicação direta entre os *peers*. Os tipos de proteção mais utilizados atualmente são os *firewalls*, *NATs* (*Network Address Translators*) e servidores *proxy*. Uma técnica comumente utilizada pelas aplicações para realizar a comunicação através de *firewalls* é o tunelamento. Nessa técnica, os pacotes trocados entre os *peers* são encapsulados em pacotes HTTP ou outro protocolo que não esteja sendo filtrado pelo *firewall*. Para realizar a comunicação entre estações isoladas por *NATs*, uma técnica bastante utilizada é a utilização de um *relay peer*, ou seja, uma estação estrategicamente posicionada na rede que redireciona os pacotes enviados para um *peer* específico. Através dessas e outras técnicas [Rocha 2004], os sistemas P2P, na maioria dos casos, são capazes de interconectar estações mesmo que estas se encontrem em diferentes domínios administrativos. É justamente essa capacidade que torna possível a existência das redes P2P tão comuns atualmente.

Entre as principais aplicações de redes P2P atualmente, podemos citar:

- Compartilhamento de arquivos: troca e armazenamento de conteúdo. Napster (www.napster.com), Kazaa (www.kazaa.com);
- Troca de mensagens instantâneas ou *Instant Messaging* (IM): um usuário tem conhecimento quando outro usuário se conecta à rede e pode enviar mensagens em tempo real. ICQ (www.icq.com), MSN Messenger (messenger.msn.com);
- Computação distribuída: diversos *peers* colaboram para a execução de uma tarefa realizando parte da computação necessária para realizá-la. [SETI@home \(setiathome.ssl.berkeley.edu\)](mailto:SETI@home.setiathome.ssl.berkeley.edu);
- Trabalho colaborativo (*groupware*): aplicações que integram diversas funcionalidades como email, calendário e videoconferências a fim de aumentar a produtividade de um grupo de trabalho;
- Jogos em rede: a disputa de jogos em rede também vem se tornando uma aplicação bastante comum.

Dadas as características das redes P2P, facilitando a comunicação entre estações em diferentes domínios e permitindo a cooperação entre estações do mundo inteiro e dado o sucesso alcançado por tantas aplicações P2P de características tão diversas, por quê não utilizar redes P2P na implementação de um novo modelo de gerenciamento cooperativo de redes?

4. Redes Peer-to-Peer e o Gerenciamento de Redes

A partir da adoção de tecnologias P2P no gerenciamento de redes, o antigo modelo de gerenciamento cooperativo baseado em agentes inteligentes pode ser adaptado ou incrementado, dando origem a um novo modelo que agregue não apenas funcionalidades de delegação de tarefas e troca de informações entre nodos de gerenciamento, mas que também permita essas comunicações através de diferentes domínios administrativos e que permita o compartilhamento de conhecimento e recursos humanos através de diversos sistemas de gerência. Baseando-se nas características de redes P2P, esse novo modelo poderia incluir as seguintes funcionalidades:

- Compartilhamento de arquivos: diversos sistemas de gerenciamento podem compartilhar arquivos relacionados ao gerenciamento de redes, tais como manuais de dispositivos ou arquivos de configuração de equipamentos;
- Compartilhamento de informações: além de arquivos em geral, bases de dados inteiras podem ser mantidas de forma distribuída, alimentando, por exemplo, sistemas de *troubleticketing*;
- Compartilhamento de conhecimento: gerentes podem encontrar outros gerentes, realizando buscas de acordo com os conhecimentos técnicos dos mesmos e de acordo com as características da rede gerenciada. Dúvidas podem ser resolvidas através de sessões de chat ou videoconferência;
- Interconexão de nodos de gerenciamento através de diferentes domínios: utilizando as técnicas de interconexão adotadas nos sistemas P2P, é possível realizar a comunicação de nodos de gerenciamento através de diferentes domínios. Essa característica torna possível a cooperação entre gerentes de diferentes corporações, podendo solucionar o problema da alocação de banda ou até mesmo facilitando a gerência de um *backbone*. A combinação dessa característica com outras técnicas de gerenciamento, como por exemplo agentes inteligentes, pode dar origem a várias outras aplicações.

5. Considerações Finais

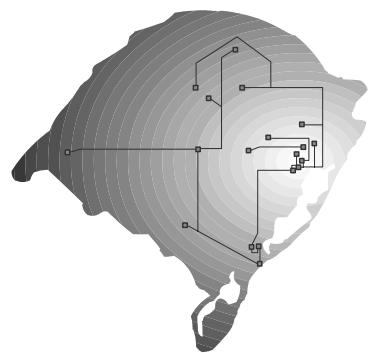
Foram apresentadas uma discussão a respeito da aplicabilidade de redes P2P no gerenciamento cooperativo de redes e uma revisão de alguns conceitos de gerenciamento distribuído, através da qual pôde-se observar que o tradicional modelo de gerenciamento cooperativo pode ser estendido, dando origem a um novo modelo mais amplo.

Além disso, foi apresentado um estudo sobre redes P2P e suas características, salientando-se a sua aplicabilidade na implementação de aplicações de gerência de redes. Mostrou-se que a utilização dessa tecnologia pode ajudar na implementação do modelo de gerenciamento proposto.

6. Referências

- Calisti, M. and Faltings, B. (2000) "Automated Allocation of Multiprovider Service Demands". International Conference on Trends towards a Universal Service Market (USM2000), Munique, Alemanha.
- Meyer, K., Erlinger, M., Betser, J., Sunshine, C., Goldszmidt, G. and Yemini, Y. (1995) "Decentralizing Control and Intelligence in Network Management". 4th International Symposium on Integrated Network Management, Santa Barbara, EUA.
- Goldszmidt, G. and Yemini, Y. (1993) "Evaluating Management Decisions via Delegation". IFIP International Symposium on Network Management Proceedings, San Francisco, EUA.
- Willmott, S. and Calisti, M. (2000) "An Agent Future for Network Control?". Swiss Journal of Computer Science (Informatik/Informatique).
- Schönwälter, J., Pras, A., Martin-Flatin, J. P. (2003) "On the Future of Internet Management Technologies". IEEE Communications Magazine, Vol. 41, No. 10, pp 90-97.

- Gao, Raymond (2003) "To P2P or P2P Too: A Discussion Of Peer-to-Peer and Related Technologies". P2P Journal.
- Oram, A. (2001) "Peer-to-Peer for Academia". O'Reilly Peer-to-Peer & Web Services Conference.
- Martin-Flatin, J. P. and Znaty, S. (1997) "A Simple Typology of Distributed Network Management Paradigms". IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM'97), Sydney, Australia, pp. 13-24.
- Strauss, F. (2000) "Distribution Models for Network Management Functions". Project report. Disponível em <http://www.ibr.cs.tu-bs.de/users/strauss/jasmin-dismod.ps.gz>. Acessado em Junho de 2004.
- Ray, P., Hawryszkiewycz, I. and Fry, M. (1996) "Group Cooperation in Network Operations and Management". IEEE International Communications Conference (ICC'96), Dallas, EUA.
- Ray, P., Loge, C., Gay, V. and Fry, M. (1995) "Cooperative Network Management from ODP Viewpoints". Accepted for IFIP Publication. Disponível em <ftp://ftp.cs.su.oz.au/bob/ULPAAC/112-RAY.ps.gz>. Acessado em Junho de 2004.
- Rocha, J., Domingues, M., Callado, A., Souto, E., Silvestre, G., Kamienski, C. e Sadok, D. (2004) "Peer-to-peer: Computação Colaborativa na Internet". Minicurso. Anais 22º Simpósio Brasileiro de Redes de Computadores (SBRC2004), Gramado, Brasil.



Sessão Técnica 11

Comunicação sem Fio II

Propostas de Pesquisa de Segurança em Redes Sem Fio

André Peres¹, Raul Fernando Weber²

¹Faculdade de Informática – Universidade Luterana do Brasil (ULBRA)

92420-280 – Canoas RS – Brasil

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)

Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{peres, weber} peres@ulbra.tche.br, weber@inf.ufrgs.br

Resumo. Este artigo descreve uma proposta de modelos para pesquisa sobre os aspectos de segurança em redes sem fio padrão IEEE 802.11. São apresentadas as características de funcionamento deste tipo de rede consideradas relevantes para a segurança, o modo de funcionamento do protocolo de segurança WEP (Wired Equivalent Privacy), alguns problemas relacionados a este protocolo levantados por outros estudos, e possíveis modelos de estudo para melhorias na segurança desse tipo de rede.

1. Introdução

As redes de computadores locais sem fio padrão IEEE 802.11[1] estão se tornando uma realidade para um grande conjunto de instituições e empresas. Estas redes permitem uma série de novas funcionalidades para troca de informações, tais como a facilidade de mobilidade de dispositivos, portabilidade e flexibilidade de conexões. As novas tecnologias de redes prometem o aumento da produtividade com custos relativamente baixos.

A forma de conexão e de compartilhamento é estabelecida de acordo com a arquitetura adotada, sendo definidas as arquiteturas de redes *ad hoc*; redes de infra-estrutura básica e; redes de infra-estrutura.

As redes *ad hoc* são compostas por estações independentes, sendo criadas de maneira espontânea por estes dispositivos. Este tipo de rede se caracteriza pela topologia altamente variável, existência por um período de tempo determinado e baixa abrangência. São denominadas redes IBSS (*Independent Basic Service Set*). Um exemplo de rede *ad hoc* está representado na figura 1a.

As redes de infra-estrutura básica são formadas por um conjunto de estações sem fio, controladas por um dispositivo coordenador denominado *Access Point* (AP). Todas as mensagens são enviadas ao AP que as repassa aos destinatários. Estas redes são denominadas BSS (*Basic Service Set*) e um exemplo deste tipo de rede é apresentado na figura 1b.

As redes de infra-estrutura são também denominadas ESS (*Extended Service Set*). Estas redes são a união de diversas redes BSS conectadas através de outra rede com ou sem fio. A estrutura deste tipo de rede é composta por um conjunto de APs interconectados, permitindo que um dispositivo migre entre dois pontos de acesso da rede. As estações vêem a rede como um elemento único. Um exemplo de rede ESS é apresentado na figura 1c.

A utilização de conexões sem fio implica na ausência de limites físicos definidos com precisão, já que a abrangência dos dispositivos é de difícil definição e na falta de controle da propagação de informações e de acesso físico.

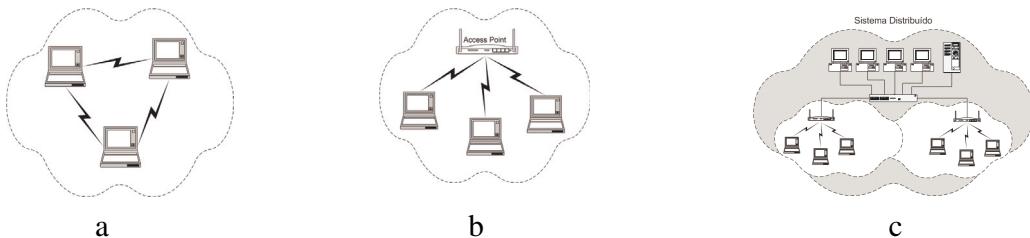


Figura 1 – redes *ad hoc* (a); BSS (b) e ESS (c)

Qualquer equipamento que possui acesso à área de abrangência da rede possui acesso aos dados sendo transmitidos. Isto significa que, para que uma rede sem fios possua as mesmas características de segurança de uma rede com fios, existe a necessidade de inclusão de mecanismos para o controle de autenticação e confidencialidade.

É importante salientar que a segurança que deve ser adicionada encontra-se no nível de enlace de dados. Isto se deve ao fato de que os aplicativos e protocolos de níveis superiores foram desenvolvidos contando com a segurança física disponível nas redes com fios. O nível de enlace das redes sem fio deve, então, prover características de segurança que compatibilizem estes dois tipos de conexão, e possibilitem a execução de aplicativos sem riscos.

A segurança no nível de enlace que deveria garantir a compatibilidade entre conexões com e sem fios foi prevista no padrão IEEE 802.11 através do protocolo WEP (*Wired Equivalent Privacy*). Este protocolo provou-se ineficiente em uma série de estudos [2] [3] e atualmente encontra-se em fase de reformulação por um grupo especial da IEEE, o grupo IEEE 802.11i. Os mecanismos de segurança previstos pelo padrão são a autenticação de dispositivos e a confidencialidade de dados.

2. Protocolo de segurança do IEEE 802.11 – WEP

O protocolo de segurança WEP tem por objetivo tornar a rede sem fios tão segura quanto uma rede com fios. Para realizar suas funções, o protocolo WEP possui os seguintes mecanismos:

- um segredo comum compartilhado entre as estações envolvidas;
- um algoritmo de criptografia baseado no RC4, utilizado para gerar o fluxo de dados cifrado.

O protocolo WEP funciona utilizando o gerador de números pseudo-aleatórios PRNG (*Pseudo-Random Number Generator*), do RC4. A semente para geração da chave é uma combinação do segredo compartilhado entre as estações com um vetor aleatório de 24 bits chamado IV (*Initialization Vector*).

A chave gerada pelo PRNG utilizando o segredo compartilhado concatenado com o IV como semente é utilizada em operações xor com a mensagem original, produzindo o texto cifrado [4].

Devido à grande taxa de perda de quadros de enlace, é inviável a utilização de uma chave única de sessão entre transmissor e receptor. A necessidade de sincronismo de chaves faz com que cada quadro de enlace utilize uma chave de criptografia diferente.

Para cada quadro, o protocolo WEP deve selecionar um IV diferente, permitindo que o segredo compartilhado permaneça o mesmo, mas a semente se altere [5].

Como o destinatário da mensagem deve criar a chave de decifragem a partir da mesma semente, o remetente envia o IV escolhido sem criptografia junto com o quadro. Desta forma, o destinatário pode unir o segredo compartilhado com o IV escolhido e utilizar estas informações como semente no PRNG.

Para verificar que os dados não foram alterados durante a comunicação, é utilizado um algoritmo redundante do tipo CRC-32 (*Cyclic Redundancy Check*) denominado ICV - *Integrity Check Value*.

A figura 2a apresenta o esquema de cifragem de mensagens do protocolo WEP, enquanto a figura 2b apresenta a decifragem.

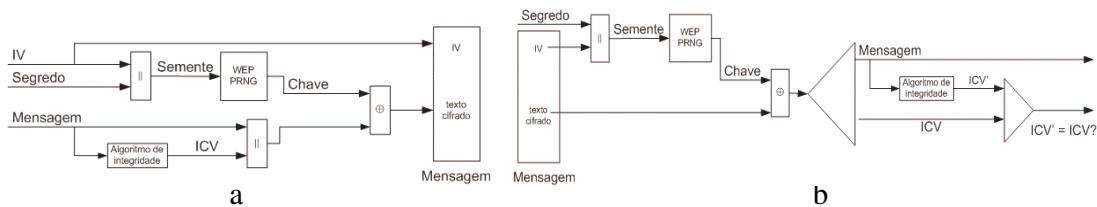


Figura 2 – WEP

2.1 Possíveis ataques ao protocolo WEP

Para obter acesso a uma rede sem fios, todas as estações devem ser autenticadas com o AP. Para realizar a autenticação, o padrão IEEE 802.11 especifica a autenticação *OpenSystem*, ou autenticação nula, na qual o AP aceita qualquer estação (esta é a autenticação padrão); ou autenticação *Shared Key* que implementa o WEP para autenticar as estações e trocar informações.

A autenticação *shared key* é implementada através da escolha de um texto desafio pelo AP que o repassa ao requisitante (cliente). O cliente deve cifrar este texto utilizando o segredo compartilhado e devolver o resultado ao AP. O AP pode agora verificar se o cliente possui o segredo compartilhado e o autenticar.

Em ambos modelos é possível realizar ataques na rede, porém, no restante deste trabalho serão consideradas as redes que implementam autenticação *Shared Key*.

A utilização de diferentes IVs não impede a repetição de chaves, pois o seu pequeno tamanho (24 bits) acarreta em repetições. Por exemplo, em uma rede onde o AP envia quadros de 1500 bytes a 11 Mbps ocorrerão repetições a cada: $(1500*8)*(2^{24})/(11*10^6)$ aproximadamente 18000 segundos, ou seja, a cada 5 horas.

O algoritmo RC4 ao ser utilizado nas redes sem fio, devido à repetição de IVs, apresenta fraquezas. O trabalho [6] apresenta os resultados de uma pesquisa sobre fraquezas do algoritmo RC4 utilizado nas redes sem fio. Estas fraquezas dizem respeito a um grande número de chaves fracas geradas pelo RC4, e pela possibilidade de descobrir bits do

segredo compartilhado a partir da análise/conhecimento dos primeiros bits da mensagem cifrada. Os primeiros bits são sempre conhecidos pelo atacante devido ao cabeçalho LLC (*Logical Link Control*), o qual inclui sempre a mesma informação (0xaa) no início do texto a ser cifrado. A partir dos resultados obtidos neste trabalho foram desenvolvidos programas como o AirSnort [7], o qual consegue obter o segredo compartilhado entre as estações através da análise do tráfego da rede. Ao obter o segredo compartilhado, a privacidade da rede e a autenticação ficam completamente comprometidas.

3. Propostas de solução/ aprimoramento da segurança

Alguns modelos propõem técnicas para acréscimo de segurança nas redes sem fio. Estes modelos estão atualmente fase de análise, sendo os que possuem maior destaque:

- utilização de mecanismos de proteção nas camadas superiores – uma solução para a maioria dos problemas nas redes sem fio seria a adoção de redes privativas virtuais VPNs (*Virtual Private Networks*). Existem alguns aspectos que devem ser considerados neste tipo de comunicação, principalmente sobre a degradação do desempenho da rede no estabelecimento de um grande número de conexões com um *gateway* VPN, além da perda de desempenho, consumo de baterias e processamento na adoção desta forma de comunicação em PDAs.
- utilização do padrão IEEE 802.1x para autenticação de dispositivos – propõe a utilização de variações do protocolo EAP (*Extensible Authentication Protocol*) para realizar a autenticação dos dispositivos [8]. Alguns estudos neste modelo apresentam problemas em relação a ataques do tipo *man-in-the-middle*, necessitando da criação de autoridades certificadoras (CAs) para as chaves trocadas entre os dispositivos, além da possibilidade de sequestro de seção (hijack) [9].

Existem outras soluções sendo propostas por fabricantes de equipamentos, mas nenhuma delas foi padronizada, ou é amplamente adotada na prática. Salienta-se o protocolo WPA (*Wi-Fi Protected Access*) que está atualmente em estudos, e já se encontra em alguns equipamentos comerciais (apesar de não ser padronizado).

4. Linhas de Pesquisa Sugeridas

Tendo em vista as fragilidades do protocolo atualmente utilizado para garantir a proteção das redes sem fio, e dos problemas envolvendo as propostas de solução, torna-se evidente a necessidade de novas pesquisas em mecanismos de proteção nestas redes.

Estas novas pesquisas devem possuir como objetivo a construção de alternativas que incrementem os aspectos de segurança das redes, além de apresentarem condições de utilização considerando o impacto no desempenho das redes e a possibilidade de uso de dispositivos móveis com baixo poder de processamento e que utilizem baterias (PDAs).

Cria-se então, a necessidade de linhas de pesquisa integradas no aprimoramento de segurança com garantia de usabilidade real.

4.1 Pesquisa em protocolos segurança de redes sem fio

Para a adição de segurança de redes sem fio, sugere-se a criação de protocolos alternativos ao WEP, considerando as características particulares do tipo de comunicação sem fio. Deve-se dar atenção ao fato da forma de segurança estar

localizada no nível de enlace e na flexibilidade necessária ao novo protocolo para utilização em uma grande quantidade heterogênea de equipamentos.

A pesquisa deve estar centrada na criação de metodologias de testes e verificações das alternativas propostas tentando abranger o maior número de situações de ataque. É sugerida a adoção de verificação formal dos protocolos propostos.

Os novos protocolos de criptografia podem ser desenvolvidos através da alteração do código fonte dos *drivers* de placas de rede sem fio. Caso deseje-se apenas uma validação do mecanismo de criptografia, é necessária apenas a configuração de um ambiente *ad-hoc*. Nesse ambiente é possível a criação de uma arquitetura formada por equipamentos que trocam informações cifradas.

A adoção de mecanismos de segurança mais complexos cria reflexos evidentes no desempenho da rede. Especialmente tratando-se de algoritmos de cifragem de dados no nível de enlace que envolvem a sua adoção em equipamentos com baixo poder de processamento e necessidades especiais de economia de energia.

Deve-se, para verificar se os protocolos propostos contemplam estas necessidades especiais, realizar a criação de modelos das redes sem fio contendo os diversos tipos de equipamentos implementando as propostas. Sugere-se, então, que estes modelos sejam submetidos a testes em simuladores, com o objetivo de verificar o impacto das propostas no desempenho da rede.

4.2 Adição de mecanismos de autenticação e gerência de segurança

Outra alternativa para aumentar as características de segurança em uma rede sem fios, é a construção de um AP contendo mecanismos de segurança atualmente disponíveis.

Existem *drivers* alternativos capazes de fazer com que um computador coloque uma placa de rede sem fio em modo *máster*, capaz de assumir as funcionalidades de um AP. Isto possibilita adição de funções de autenticação, gerência e filtros de pacotes. Ao adicionar estes mecanismos, é possível aumentar a segurança de uma rede baseada em AP.

Alguns mecanismos possíveis:

- filtros de pacotes: pode-se adicionar filtros de pacotes para controle do tráfego entre a rede sem fios e a rede com fios, ou entre os dispositivos sem fio;
- mecanismo de identificação de intrusão: mecanismos de IDS (*Intrusion Detection Systems*) são softwares capazes de realizar uma análise do tráfego da rede, em busca de alterações do comportamento considerado normal (configurado pelo administrador) ou de assinaturas de ataques conhecidos. Esses mecanismos são uma ferramenta de grande importância para a gerência de segurança de redes atualmente;
- VPN: é possível a adição de túneis VPN entre APs ou entre um AP e seus clientes.

Deve-se salientar que os APs são dispositivos que operam em *layer 2*, ou seja, os mecanismos de segurança que operam nos níveis superiores estão sujeitos à configuração do AP para que possa ser feita a filtragem e análise do tráfego.

Os mecanismos de autenticação e criptografia propostos podem ser adicionados também nesse dispositivo, graças à flexibilidade de configuração de um AP em um computador.

5. Conclusões

Atualmente, os mecanismos de segurança disponíveis para redes sem fio apresentam-se ineficazes para cumprir suas funções. O mecanismo de confidencialidade utilizado pelo WEP não é seguro e pode ser comprometido através da utilização de *softwares* de análise estatística de tráfego. Com isso, tanto a confidencialidade quanto a autenticidade não são garantidas.

A necessidade de busca de alternativas passa por dois processos: alteração do WEP; e/ou adição dos mecanismos hoje disponíveis para segurança de redes. A validação de novos mecanismos de criptografia é um processo que deve ser realizado através de simulações e testes que garantam tanto a confidencialidade, quanto a praticidade dos novos modelos.

Atualmente, os autores deste trabalho realizam testes com mecanismos adicionais em APs, em busca da adição de segurança em redes sem fio.

Bibliografia

- [1] IEEE 802.11. Part 11: **Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.** Disponível por www em: <http://standards.ieee.org/getieee802/802.11.htm>. 1999.
- [2] ARBAUGHT, William A; et al. **Your 802.11 Wireless Network has No Clothes.** Department of Computer Science - University of Maryland. Disponível por www em: <http://www.cs.umd.edu/~waa/wireless.pdf>
- [3] WALKER, Jesse R. **Unsafe at any key size; An analysis of the WEP encapsulation.** Intel Corporation, Oregon, 2000. Disponível por www em: <http://grouper.ieee.org/groups/802/11/Documents/DocumentHolder/0-362.zip>
- [4] SCHNEIER, Bruce. **Applied Cryptography. Second Edition: protocols, algorithms, and source code in C.** John Wiley & Sons Inc. 1996.
- [5] BORISOV, Nikita; et al. **Security of the WEP algorithm.** Disponível por www em: <http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html>
- [6] FLUHRER, Scott; et al. **Weaknesses in the Key Scheduling Algorithm of RC4.** Cisco Systems Inc. Disponível por www em: http://www.cs.umd.edu/~waa/class-pubs/rc4_ksaprocs.ps
- [7] AIRSNORT. Disponível por www em: <http://airsnort.shmoo.com/>
- [8] RSA Security. **Improving Wireless LAN Authentication – A Description of the Authentication in 802.1x Standard.** RSA Security Inc. Disponível por www em: <http://www.rsasecurity.com/go/slides2001Q4/wirelesslive/WirelessLANAuthentication2.pdf>
- [9] MISHRA, Arunesh; Arbaugh, William. **An Initial Security Analysis of the IEEE 802.1X Standard.** University of Maryland. Disponível por www em: <http://www.cs.umd.edu/~waa/1x.pdf>

Os novos padrões de segurança em redes wireless

Afonso Comba de Araujo Neto, Bruno Castro da Silva

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul

afonso@cpd.ufrgs.br, bcs@inf.ufrgs.br

Resumo. As tecnologias sem fio são cada vez mais importantes para empresas e usuários domésticos, na medida em que facilitam a interconexão de dispositivos móveis e reduzem custos de infra-estrutura. Entretanto, a rápida difusão de dispositivos wireless não foi acompanhada por um semelhante avanço nos métodos de segurança envolvidos. Este artigo apresentará uma revisão acerca das precariedades de segurança que motivam nosso trabalho, assim como um survey entre os novos padrões de segurança em redes wireless sendo propostos pelas organizações responsáveis.

1. Introdução

As redes *wireless* foram criadas a fim de resolver alguns problemas inerentes às redes tradicionais. Sem a necessidade de cabeamento físico, os dispositivos que necessitam de acesso à rede não mais ficam restritos a uma localização específica, o que é extremamente bom para *notebooks* e outros dispositivos móveis. Dada uma área de cobertura, computadores podem acessar a rede a partir de praticamente qualquer ponto geográfico, sem que ao usuário seja imposto nenhum esforço adicional.

Ao mesmo tempo em que o paradigma *wireless* oferece vantagens, traz consigo uma série de problemas, como o da falta de privacidade e de segurança. Da mesma maneira com que usuários legítimos têm acesso a rede, usuários ilegítimos também podem ter. Mais do que isso, sem a restrição física do acesso (ex: necessidade de acesso físico a um *hub*), um usuário malicioso pode ter acesso muito fácil a todos os dados que trafegam na rede. Para piorar ainda mais a situação, é bastante difícil, ou inviável, detectar indivíduos que estejam monitorando o tráfego de uma rede *wireless* [Borisov et al., 2001].

Essas características das redes *wireless* tornam claras a necessidade de mecanismos eficientes de criptografia e autenticação. Com isso em mente, o IEEE incluiu no padrão 802.11 para redes *wireless* um mecanismo de segurança chamado WEP, ou *Wireless Equivalent Privacy*. O objetivo deste mecanismo era garantir em uma rede sem fio a segurança implícita obtida pela restrição de acesso ao barramento em uma rede cabeada.

Entretanto, o WEP se mostrou extremamente deficiente em vários aspectos. Diversos trabalhos da comunidade científica demonstram as falhas do WEP e as razões pelas quais ele não é confiável. Inicialmente, o IEEE respondeu aos problemas de segurança com uma proposta de tecnologia que ficou conhecida como WPA, *WiFi Protected Access*¹. O WPA foi pensado como um passo transitório para uma solução definitiva, na medida em que poderia ser implantado sem que fosse necessário substituir todo o *hardware* já desenvolvido para suportar o WEP.

Nas seções seguintes serão apresentadas as deficiências do protocolo WEP e as soluções propostas pelo padrão 802.11i. Além disso, será feita uma crítica acerca de cada

¹O WPA utiliza conceitos já em discussão no GT IEEE-802.11i, grupo de trabalho responsável pela definição de um padrão de segurança em redes *wireless*.

solução, e as várias sugestões serão comparadas a fim de prover um bom panorama do estado da arte da segurança em redes *wireless*.

2. WEP: uma tecnologia falha

As redes *wireless* tradicionais são constituídas por dois elementos principais: o ponto de acesso, AP (*Access Point*), e as estações de trabalho. Os APs funcionam como *gateways* em uma rede *wireless*, fornecendo o primeiro ponto de contato para as estações de trabalho e são o ponto de conexão com outras redes. Para uma estação de trabalho ter acesso à rede ela precisa fazer a chamada *associação* com um ponto de acesso. O padrão 802.11 original, implementado na maior parte dos dispositivos *wireless* comercializados hoje em dia, especifica duas formas de associação: a **associação via sistema aberto** e **associação via autenticação por chave compartilhada**.

A associação via sistema aberto na verdade poderia ser chamada de associação sem autenticação. A estação de trabalho simplesmente requisita o acesso à rede e o ponto de acesso o concede, sem quaisquer restrições. É fácil perceber que este esquema implica uma rede completamente vulnerável: além de qualquer pessoa ter acesso aos recursos da rede e aos dados que trafegam na mesma, ainda é possível inserir pontos de acesso falsos, capazes de estender a rede de uma forma não originalmente prevista.

O protocolo WEP é empregado quando se utiliza o método de associação via autenticação por chave compartilhada. Toda a segurança do WEP está centrada em um único elemento: o compartilhamento de uma chave secreta entre todos dispositivos. Esta chave precisa ser configurada manualmente em todos os dispositivos que tiverem autorização para utilizar a rede. A partir dessa chave, a estação de trabalho se associa ao ponto de acesso através de um protocolo do tipo desafio-resposta. A partir daí, todos os pacotes passam a ser criptografados. A seguir será apresentado um resumo a respeito do funcionamento desse método de criptografia.

2.1. Criptografia no WEP

A chave secreta do WEP possui 40 ou 104 bits, dependendo da configuração dos dispositivos da rede. Como ela é configurada manualmente em cada dispositivo, sua transmissão pela rede não precisa ocorrer em nenhuma situação.

Além de permitir a associação dos dispositivos ao ponto de acesso, essa chave compartilhada é utilizada para cifrar todos os pacotes que trafegam na rede. A cifragem propriamente dita é feita através o algoritmo RC4². A cifragem dos pacotes se dá através dos seguintes passos:

1. é feito um cálculo de integridade do pacote através do algoritmo de CRC32. O resultado desse cálculo, chamado de ICV (*integrity check value*) é concatenado ao fim do pacote;
2. é gerado um vetor de inicialização de 24 bits, chamado de IV (*Initialization Vector*). Este IV, em princípio, deveria ser diferente para cada pacote gerado, mas na prática se repete em períodos de tempo que dependem do tráfego da rede e da forma como o algoritmo o determina;
3. o IV é concatenado com a chave secreta, gerando uma chave de cifragem de 64 ou 128 bits, dependendo do tamanho da chave secreta;

²RC4 é um cifrador de *stream* projetado por Ron Rivest. É um cifrador de chave variável com operação voltada a byte e baseado em permutação aleatória. Análises mostram que o período do encriptador é maior que 10^{100} .

4. o pacote é cifrado utilizando-se o algoritmo RC4 com a chave do passo anterior. O algoritmo gera, com base nessa chave, uma seqüência de *bytes* pseudo-aleatórios (também chamado de *stream* pseudo-aleatório), que são misturados com os *bytes* do pacote através da operação ou-exclusivo (XOR);
5. o pacote cifrado é enviado juntamente com o valor IV, sendo que este último transita em claro.

Ao receber um pacote cifrado, o receptor concatena a chave secreta com o IV que recebeu, de modo a reconstruir a seqüência pseudo-aleatória utilizada na cifragem do pacote utilizando novamente o RC4. O passo seguinte da decifragem consiste em aplicar novamente a seqüência de operações XOR a fim de reconstruir o conteúdo original do pacote. O receptor deve também recalcular o CRC32 do pacote, a fim de detectar alterações no conteúdo do mesmo.

2.2. Os problemas do WEP

Foi necessário pouco tempo após a publicação do padrão para que surgissem muitos dispositivos que já o implementassem. As deficiências do WEP foram descobertas na mesma velocidade. A seguir relaciona-se as principais fraquezas deste padrão:

O tamanho em bits do IV é muito pequeno Para redes com muito tráfego, o valor máximo delimitado pelos 24 bits do vetor de inicialização pode ser atingido em menos de um dia. Isso causa a reutilização dos vetores de inicialização, o que é um problema de segurança, pois pacotes com o mesmo vetor de inicialização utilizam o mesmo *stream* pseudo-aleatório para sua cifragem. Se for feito um ou-exclusivo entre dois pacotes com o mesmo IV, obtém-se o ou-exclusivo entre o valor em claro dos pacotes. Se o conteúdo de um deles for conhecido, o que não é incomum, pode-se descobrir o valor do outro. Da mesma forma, é possível obter o *stream* que foi utilizado para cifragem, o que então permitirá a decifragem de todos os pacotes subsequentes com o mesmo IV;

O padrão não define como o IV deve ser modificado O problema anterior é acentuado por não haver regras para a geração do IV. É uma prática comum entre os dispositivos a reinicialização do IV em zero, sempre que estes dispositivos forem ligados. Com isso, o número de IVs utilizados diminui drasticamente;

O CRC32 não é adequado para detectar alterações propositais nos pacotes O algoritmo CRC32 é um ótimo mecanismo para detecção de erros aleatórios. Entretanto, ele utiliza um cálculo linear sobre os dados, de modo que é relativamente simples implementar um método que permita modificar o pacote de maneira que o destinatário não seja capaz de detectar tal alteração através da simples conferência do *checksum*;

Por ser parte da chave, o conhecimento do IV fornece pistas da chave secreta Para alguns valores IV é possível obter parte da chave secreta através de um algoritmo probabilístico. Para o ataque, basta o conhecimento do *stream* RC4 utilizado. A descoberta da chave envolve apenas a tarefa de se obter na rede alguns pacotes com os valores IV adequados e cujo conteúdo do pacote (ou pelo menos parte dele) seja conhecido. Note que o fato de todos os dispositivos utilizarem a mesma chave secreta faz com que todos os pacotes que trafegam na rede possam ser utilizados no ataque. Em uma rede com bastante tráfego, é possível obter a chave secreta em apenas algumas horas [Fluhrer et al., 2001];

O padrão não define formas automatizadas da atualização da chave secreta A maioria dos problemas citados seriam menos críticos se houvesse uma política de troca das chaves secretas. Entretanto, o fato da troca ser manual não permite que isso seja feito regularmente, sendo que em redes com muitos dispositivos isso seja uma tarefa praticamente inviável. Além disso, se algum dos dispositivos for comprometido, ou se algum dos usuários da rede accidentalmente perder a chave, então toda a rede ficará comprometida.

É importante salientar que, por mais problemas que o WEP possa ter, o seu uso é recomendável quando a única alternativa é o abandono completo da criptografia. Entretanto, em ambientes corporativos, e mesmo para usuários domésticos que se importam com a privacidade, o WEP simplesmente não é suficiente. Outro problema para os ambientes corporativos é a falta de mecanismos de autenticação em nível de usuário. O acesso a um dispositivo que contenha a chave compartilhada, ou a simples configuração da chave em qualquer dispositivo não confiável, são suficientes para garantir acesso completo a todos recursos da rede.

3. O padrão IEEE 802.11i

Com o objetivo de resolver as deficiências do padrão WEP, a IEEE designou ao chamado *Task Group i* a missão de criar um novo padrão de segurança para redes *wireless*. O padrão ainda não foi completamente estabelecido, mas as suas principais características já foram definidas [Robinson, 2004].

O padrão 802.11i define duas grandes classes de solução: a primeira, que atualmente foi certificada pela *WiFi Alliance*, é comercialmente conhecida como WPA; a segunda solução provavelmente será chamada de WPA2 [Burns, 2004].

3.1. WPA: uma solução imediatista

Um dos objetivos da primeira solução proposta pelo padrão 802.11i foi tentar resolver os problemas de segurança do WEP da melhor maneira possível, mas de forma a não implicar a troca de *hardware* nos dispositivos que já suportam WEP. De fato, isso foi um dos pré-requisitos impostos ao grupo, pois era necessário entregar aos usuários da tecnologia WEP uma solução que exigisse, no máximo, uma atualização de *software* ou *firmware*.

A principal característica do WPA é a utilização do protocolo chamado TKIP (*Temporal Key Integrity Protocol*), o qual foi desenvolvido especialmente a fim de resolver os principais problemas do WEP. O TKIP é implementado como uma camada de software que funciona sobre o WEP, de forma a resolver os seus problemas, mas sem tornar o sistema incompatível com o padrão original. Ele acrescenta quatro características novas ao protocolo WEP:

- É adicionado um novo código de integridade dos pacotes, denominado *Michael*, que utiliza uma chave de autenticação e facilita a detecção de alterações proposicionais no pacotes;
- Passam a existir regras no seqüenciamento dos vetores de inicialização, a fim de dificultar ataques de repetição e o reuso de IVs. O tamanho do IV é aumentado para 48 bits;
- É utilizada uma função *hash* para o cálculo da chave, a fim de remover a correlação entre vetores de inicialização e a seqüência pseudo-aleatória gerada pelo RC4. Além disso, esta função passa a incluir o endereço MAC do dispositivo como parte da chave, fazendo com que cada dispositivo acabe utilizando uma chave diferente;
- A chave secreta passa a ser substituída a cada 10 mil pacotes enviados.

Note que o TKIP ainda utiliza o algoritmo de *stream* RC4. Entretanto, ele resolve os problemas do WEP implementando uma política de uso de chaves adequadas e a sua substituição periódica. Dessa forma, a segurança é obtida utilizando o mesmo *hardware* do WEP. Além disso, ataques de força bruta são dificultados na medida em que a chave efetiva passa de 104 bits para 128 bits, no WPA, devido ao uso da função *hash*.

Para a geração da seqüência de chaves que serão utilizadas para cifragem dos dados, o protocolo TKIP utiliza uma chave mestre como base. No WPA (e também no WPA2), existem duas formas de se obter essa chave mestre. A primeira é uma autenticação em nível de usuário e a segunda é a partir de uma chave cadastrada diretamente nos dispositivos.

O esquema de autenticação em nível de usuário do WPA presume a utilização de um servidor de autenticação, o qual provê sua funcionalidade através do uso de um protocolo chamado EAP (*Extensible Authentication Protocol*) [Haining, 2002]. A arquitetura de autenticação do WPA é baseada no padrão 802.1X, responsável pela definição de um *framework* para autenticação e gerência dinâmica de chaves tanto em redes cabeadas quanto em redes wireless.

Nesse esquema, ao tentar se associar a um ponto de acesso, o cliente estabelece uma sessão EAP com o servidor de autenticação. O padrão não define o método específico de autenticação a ser utilizado, podendo ser implementado desde autenticação por desafio-resposta até autenticação via certificados digitais. Isso, na prática, depende do servidor de autenticação em questão.

Após uma autenticação bem sucedida, é gerada uma chave de sessão entre o servidor de autenticação, o ponto de acesso e o dispositivo cliente. Essa chave é válida somente durante essa sessão, e é então utilizada pelo TKIP para geração das chaves temporárias.

Para o seu uso em ambientes domésticos, onde seria inviável ou desnecessário a utilização de um servidor de autenticação, o WPA funciona em um modo denominado PSK (*Pre Shared Key*). Esse modo é exatamente o mesmo empregado no WEP, ou seja, uma chave é cadastrada manualmente em cada um dos dispositivos autorizados a fazer parte da rede. A diferença para o WEP é que, no WPA, o algoritmo TKIP gera chaves temporárias a partir desta chave secreta, de modo que a chave secreta propriamente dita nunca é utilizada diretamente para cifragem do tráfego entre os dispositivos.

4. WPA2: a solução definitiva

Uma solução mais completa para o problema de segurança em redes *wireless* é o WPA2, também chamado de RSN (*Robust Security Network*). A IEEE e a WiFi Alliance pretendem unificar o WPA2 e o 802.11i de modo a reduzir o número de padrões concorrentes, e também facilitar a adoção de soluções integradas por parte de fabricantes de *hardware*.

O WPA2 irá fornecer mecanismos de pré-autenticação, através dos quais uma estação se autentica em um AP com o qual irá se comunicar no futuro de modo que, durante o *roaming*, ocorre uma diminuição significativa do tempo de perda de sinal. Além disso, será empregado o protocolo CCMP para cifragem, o qual é baseado no algoritmo de criptografia AES utilizado em modo de operação CCM (*Counter Mode with CBC-MAC*). O CCM, que é independente do algoritmo de criptografia utilizado, é um modo genérico de operação que fornece autenticação e cifragem ao mesmo tempo, utilizando uma chave de 128 bits [Schneier, 1996]. O uso do AES, ao invés do RC4, garante maior segurança devido exatamente à melhor qualidade deste algoritmo.

Assim como o WPA, o WPA2 será definido em duas instâncias: o *WPA2 enterprise*, o qual conterá todos os requisitos necessários para suportar a autenticação prevista pelo padrão 802.1X com o protocolo EAP, e o *WPA2 personal*, que implementará o modo PSK e será voltado para ambientes SOHO (*Small Office/Home Office*).

Espera-se que a migração para o WPA2 das redes que utilizam o WPA não envolva maiores dificuldades, já que ele prevê um modo de operação conhecido como “*mixed*-

mode”. Neste modo, um dispositivo WPA2 é capaz de tratar simultaneamente dispositivos que utilizem tanto tecnologia WPA quanto WPA2.

Uma questão importante é o fato de que, mesmo que o *framework* utilizado pelo WPA2 seja basicamente o mesmo do WPA, a adoção deste novo padrão não será transparente. Isso é devido a necessidade de atualização de *hardware* que ele implica. O *hardware* que implementa o WEP e o WPA não é capaz de oferecer criptografia AES com desempenho adequado. Provavelmente a migração para o WPA2 irá ocorrer lentamente, junto com o desenvolvimento da tecnologia *wireless*, e só acontecerá de maneira brusca em redes que exijam mecanismos de cifragem muito superiores ao RC4.

5. Conclusões

As deficiências do WEP foram prontamente detectadas pela comunidade científica e pelos usuários de redes *wireless*. A primeira reação por parte dos administradores foi a de implementar técnicas que transferissem a responsabilidade pela privacidade e autenticação para os níveis superiores da rede. Essas medidas retiravam do padrão 802.11 a responsabilidade por prover segurança, e consistiam na aplicação de mecanismos como VPNs (Virtual Private Networks) e IPSec (padrão da IETF para comunicação IP segura) [Schneier, 1996]. A abordagem por parte das organizações responsáveis pela definição dos padrões seguiu mais ou menos a mesma linha: primeiramente foram pensados mecanismos provisórios que sanassem os principais problemas do WEP. O padrão WPA tenta solucionar os problemas de segurança através do aumento do número de bits das chaves/IVs e também através da inserção de novos mecanismos na gerência e manipulação das chaves. O WPA2, por sua vez, exigirá maiores modificações em termos de *hardware*, provavelmente tornando os novos sistemas incompatíveis com a infra-estrutura atual. Entretanto, o pressuposto de que a segurança *wireless* deve ser provida pelo próprio padrão 802.11, e não por mecanismos externos adicionados ao gosto de cada administrador, exige que o paradigma atualmente utilizado seja substituído. Embora seja difícil prever a evolução dos sistemas de segurança, pode-se esperar que a adoção completa dos novos padrões ocorra aproximadamente no mesmo ciclo de substituição natural do *hardware* de redes *wireless*, caso o preço dos novos equipamentos acompanhe a crescente necessidade de sua adoção.

Referências

- Borisov, N., Goldberg, I., and Wagner, D. (2001). Intercepting mobile communications: The insecurity of 802.11. In *Proceedings of the Seventh Annual International Conference on Mobile Computing And Networking*, Roma, Itália.
- Burns, J. (2004). On the way: 802.11i and wpa2. *Communications News*.
- Fluhrer, S., Mantin, I., and Shamir, A. (2001). Weaknesses in the key scheduling algorithm of RC4. *Lecture Notes in Computer Science*, 2259:1–24.
- Haining, T. P. (2002). Enhanced wired equivalent privacy for ieee 802.11 wireless lans. *IEEE Document 802.11-00/362*.
- Robinson, F. (2004). Examining 802.11i and wpa: The new standards – up close. *NetworkComputing Document*.
- Schneier, B. (1996). *Applied Cryptography*. John Wiley and Sons, Inc., New York, 2nd edition.

Secure Roaming Wireless: Uma Abordagem de Segurança para Redes Locais 802.11 com Criptografia Forte

Marcos José Sarres de Almeida^{1,2}, MAR Dantas³

¹Aker Security Solutions – Departamento de Tecnologia – Brasília – DF - Brasil

²Departamento da Ciência da Computação (CIC) - Universidade de Brasília (UnB) - 70910-900 – Brasília – DF – Brasil

sarres@aker.com.br

³ Departamento de Informática e Estatística (INE) - Universidade Federal de Santa Catarina (UFSC) - Caixa Postal 476 - 88040-900 – Florianópolis-SC-Brasil

mario@inf.ufsc.br

Abstract - *In this article we present a proposal solution and an implementation for a more secure wireless LAN use based on the standard IEEE 802.11b. In the research project we have used the Aker Secure Roaming software package, which we have added the wireless feature. The environment originally helps clients to use some security facilities in a LAN such as authentication and cryptography. Our experimental results show more security level compared to the standard 802.11b, however with less network performance. In addition, we present some interfaces that we have developed that can be very friendly and efficient to users to configure the Aker Secure Roaming package in a more secure IEEE 802.11b environment.*

Resumo - *Neste artigo apresentamos uma proposta e uma implementação de solução para utilização com maior segurança de redes wireless baseadas no padrão IEEE 802.11b. Durante nosso trabalho de pesquisa, utilizamos um sistema de servidor de cliente de criptografia, o Aker Secure Roaming, com o objetivo de autenticar e cifrar todo o tráfego de dados de uma determinada rede. Nossos resultados experimentais indicam um grau de segurança muito superior ao padrão 802.11b, porém detectamos uma pequena perda de performance para a rede. Finalizamos o artigo apresentando algumas interfaces de nossa implementação que indicam a possibilidade de estabelecimento das abordagens de segurança e autenticação forte em uma rede 802.11b de maneira amigável e eficiente.*

1. Introdução

As redes *wireless* têm se apresentado como uma infra-estrutura interessante, que podem prover uma melhoria na conectividade dos dispositivos que com um determinado poder computacional desejam se conectar as redes convencionais. As facilidades de implementação e manutenção das redes *wireless* são diferenciais interessantes que motivam sua utilização. Todavia, apesar de alguns anos terem se passado após o desenvolvimento de inúmeras tecnologias *wireless* (exemplos são os padrões *Bluetooth*, 802.11a e 802.11b) (DANTAS, 2002), ainda não existe uma nítida sinalização no mercado de qual tecnologia será mais largamente utilizada. Este fato se deve principalmente aos problemas de segurança que surgiram junto com essas novas abordagens.

Quando um novo projeto de LAN *wireless* é desenvolvido, diversas vantagens aparecem de uma forma clara. Exemplos são: a colocação de novos pontos de rede com facilidade; a mudança de local de computadores com mais flexibilidade; ou ainda a manutenção dos pontos de rede de uma forma mais simples. Mas apesar de todas as vantagens expostas, as redes *wireless* ainda têm uma certa resistência de utilização por parte das empresas, pois os ambientes *wireless* não oferecem a segurança necessária para acesso a seu ambiente corporativo (PEIKARI et al., 2002).

O artigo é organizado da seguinte forma: na seção 2 fazemos uma breve apresentação do padrão IEEE 802.11 com especial ênfase aos aspectos de segurança; o ambiente de software Aker Secure Roaming e nossa extensão do pacote para redes *wireless* 802.11b são apresentados descritos na seção 3; finalizamos este artigo com nossas conclusões e sugestões para trabalhos futuros.

2. Segurança Padrão no IEEE 802.11

O padrão IEEE 802.11 é uma especificação que define um conjunto de protocolos baseados em Ethernet *wireless*. Todavia, dentro do padrão existem diversos conceitos que devem ser observados com bastante atenção para que se possa entender perfeitamente a especificação 802.11. O primeiro modelo do IEEE 802.11 foi criado em 1997 (IEEE Computer Society, 2003), abrangendo as versões 802.11a, 802.11b e 802.11g, cada uma com a sua taxa de acesso. Desta forma, dependendo do protocolo utilizado (exemplos de protocolos são TCP, UDP, ICMP) e a forma como estes estão configurados para operar, é possível conseguir melhores (ou piores) taxas de acesso (ANJUM et al., 2003). Por esta razão, a industria temendo uma falta de interoperabilidade na área, criou uma aliança de empresas denominada de *Wi-Fi Alliance*, para tratar da compatibilidade de seus produtos. Os produtos Wi-Fi podem funcionar em dois modos: modos de infra-estrutura (*infrastructure mode*) ou modo ad-hoc (BUZKO et al., 2001). Neste artigo, apenas será analisado o modo de infra-estrutura.

Nos primeiros cinco anos de criação, o *Wi-Fi* possuía apenas um método de segurança conhecido como WEP (*Wired Equivalent Privacy*) (EDNEY et al., 2003). Em 2000 as redes *wireless* começaram a ganhar popularidade, o que chamou a atenção da sociedade da segurança da informação que rapidamente detectou falhas no método padrão. Diversos sistemas para atacar configurações *wireless* foram aparecendo, mostrando diversas vulnerabilidades das redes *wireless*. Desta forma, outras tecnologias

foram desenvolvidas para agregar valor à segurança das redes *wireless* (ALLEN et al., 2002). Todavia, ainda hoje não existe uma forma de segurança padrão atenda aos usuários dos ambientes *wireless* de forma satisfatória, visando evitar problemas contra elementos mal intencionados.

Alguns analistas (PARK et al., 2002) criticaram o projeto do modelo de segurança proposto pelo IEEE, pelo fato do mesmo apresentar inúmeras falhas. Assim, vamos a seguir apresentar algumas características e as vulnerabilidades na abordagem WEP.

Características e Vulnerabilidades do WEP

O WEP trabalha com chaves compartilhadas (*shared key*), fazendo com que tanto a STA (estação de trabalho) como o AP (Access Point) conheçam a chave de criptografia que é instalada em cada máquina e no AP pelos administradores. Quando a autenticação segura esta implementada em um ambiente *wireless* o processo de autenticação ocorre na forma *challenge/response*. O algoritmo de criptografia utilizado pelo WEP é conhecido como RC4 (SCHNEIER, 1997). A principal operação de criptografia executada pelo RC4 é uma operação binária conhecida como XOR (*Exclusive OR*).

O padrão WEP possui várias falhas muito sérias que o tornam um padrão extremamente fraco do ponto de vista da segurança da informação. As redes *wireless* do padrão IEEE têm sua vulnerabilidade expostas, uma vez que os sinais de rádio podem ser captados fora da empresa, fato preocupante para o 802.11b que possui uma área de alcance maior. Já foram demonstrados ataques a redes *wireless* a mais de 40 km de distância, com a utilização de antenas de alta intensidade (SHIPLEY, 2001).

Apesar do método RC4 ser extremamente rápido, este foi quebrado em 1994, existindo um processo de engenharia reversa através da qual é possível conseguir o valor das chaves diante de um conjunto de textos em claros e textos cifrados. Como Fluhrer comenta (FLUHRER, 2000), devido ao fato do início de toda mensagem 802.11 possui características próprias (tais como campos fixos), uma análise estatística pode ser feita e assim se quebrar uma chave WEP em poucas horas. Na tabela I identificamos os principais problemas de segurança relativos à abordagem WEP (SOUMENDRA, 2002). Alguns criptógrafos analisaram as características do WEP e encontraram formas de se descobrir a chave do sistema (Walker, 2001).

Tabela 1. Principais Vulnerabilidade do WEP

1	WEP e chaves de 104 bits são opcionais	6	Após a autenticação, existe a possibilidade de roubo da sessão da STA (<i>session hijacking</i>).
2	Chaves idênticas em todas as máquinas na abordagem <i>default</i> .	7	Método de autenticação fraco, facilitando a criptoanálise.
3	Troca de chaves não é segura.	8	Controle de acesso feito apenas pelo endereço MAC.
4	Mesma chave para autenticação e codificação dos dados.	9	Possibilidade de ataques de repetição.
5	Fragilidade do algoritmo criptografia RC4.	10	Pequeno tamanho dos vetores de inicialização.

3. Projeto de Secure Roaming Wireless

Nesta seção vamos apresentar nossa contribuição, visando aumentar a segurança para o tráfego de rede de um ambiente 802.11 com WEP implementado. Como todo o padrão 802.11 é definido sobre a camada de enlace de dados, a idéia é agregar todo um sistema

de criptografia e autenticação sobre a camada superior. Em outras palavras, nossa abordagem adiciona uma maior segurança na camada de rede.

Decidimos utilizar o pacote aplicativo comercial Aker Secure Roaming (Aker, 2004), da empresa Aker Security Solutions, visando que nossa contribuição pudesse ser testada em um ambiente real. Este software implementa a criação de um canal cifrado entre o cliente e o servidor, utilizando chaves de até 256 bits e autenticação de usuários..

O Aker Secure Roaming Server possui um certificado digital com um conjunto de chave pública e chave privada. Para cada tentativa de conexão de um cliente ao servidor, o servidor envia a sua chave pública ao cliente. O servidor então escolhe um algoritmo de criptografia (em geral, AES 256) e gera uma chave de criptografia simétrica para operar o túnel criptográfico. Uma vez validado os pacotes o túnel de VPN está estabelecido. Ilustramos na figura 1, um exemplo de estabelecimento de conexão segura.

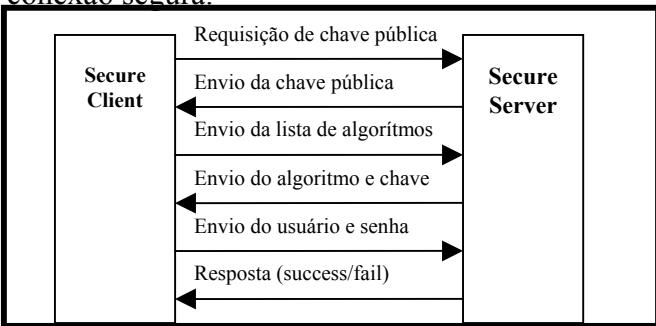


Figura 1: Estabelecimento de uma seção segura

3.1 Secure Roaming Wireless

Nesta seção apresentamos nossa contribuição de projeto que foi denominada de Secure Roaming Wireless (SRW). O ambiente utiliza as funcionalidades do Aker Secure Roaming para poder fornecer segurança forte para uma LAN wireless 802.11b. Conhecendo as falhas encontradas no WEP, foi importante definir alguns objetivos para a segurança da rede wireless, tais como privacidade e controle de acesso.

O modelo proposto para efetuar a segurança da rede necessita que a rede toda seja dividida em diversos departamentos, sendo cada um deles utilizando o Secure Roaming. Para poder acessar a rede local, o usuário precisa instalar um Secure Roaming Client na sua máquina e estabelecer uma conexão com o Secure Roaming Server.

Tabela 2 : Configuração IP do ambiente experimental

Roteador	192.168.200.254
Access Point	10.0.10.101 (para a rede interna wireless) e 10.0.20.101 (para a comunicação com o servidor)
Servidor	192.168.200.1 (para a comunicação com o roteador) e 10.0.20.1 (para comunicação com a AP)
Laptop	10.0.10.1 (para se comunicar com o AP através da placa wireless)

Configuração do sistema

O diagrama apresentado na figura 2 ilustra a comunicação da rede. Os endereços sobre as setas representam os IPs das placas de rede. Os endereços em itálico dentro das caixas representam os *gateways* padrão.

Um pacote que queria acessar a Internet, deve fazer o seguinte percurso: o pacote sai de STA (10.0.10.1) e vai através da rede wireless para o AP em 10.0.10.101 que envia para o servidor Windows 2000 Server, no IP 10.0.20.1, que repassa o pacote para o roteador ADSL no IP 192.168.200.254 (figura 2).

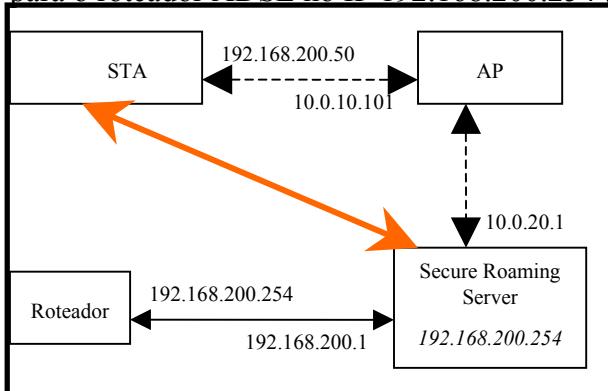


Figura 2: Diagrama da rede como Secure Roaming ativado

A configuração do Secure Roaming *Wireless* foi realizada para deixar a rede 10.0.20.0 uma rede segura. Tão logo o usuário faça acesso a uma máquina ele entra na tela de configuração do cliente e escolhe a opção de conectar. A partir deste momento ele está apto a trafegar na rede. Em adição, ainda existe a possibilidade de se estabelecer a sessão de criptografia de forma automática, agora evitando qualquer interação do usuário. Neste caso, tão logo o Windows realize o *logon*, o Aker Secure Roaming Client automaticamente se conecta com o Server e estabelece a sessão segura.

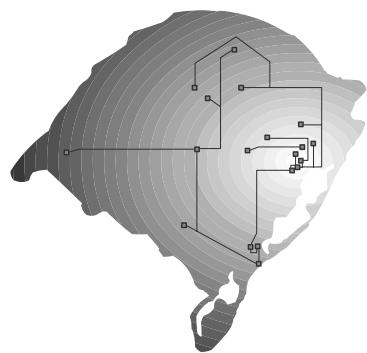
4. Conclusão e Propostas para Trabalhos Futuros

Com a crescente utilização das redes *wireless*, a necessidade de uma maior segurança do ambiente fica patente. Neste artigo, fizemos primeiramente uma pequena revisão da falta de segurança dentro do modelo padrão conhecido como WEP. Para resolver os problemas identificados utilizamos uma solução comercial. O pacote Aker Secure Roaming foi empregado e uma extensão *wireless* para o padrão IEEE 802.11b foi proposta e implementada.

Nossos resultados experimentais, após vários testes realizados, indicam que a proposta pode ser considerada uma abordagem bem sucedida, pois o ambiente pode contar com maior privacidade, melhor autenticação, controle de acesso por usuário, maior confiabilidade e integridade e proteção das chaves. Houve uma pequena perda de performance de 15% para o tráfego de rede, fato que tentaremos melhorar no futuro. Em adição, podemos dizer que a solução tem ainda como um diferencial a não requisição de nenhuma alteração de hardware ou atualização de *firmware*, sendo facilmente implementada em um ambiente de rede para suportar o padrão IEEE 802.11b.

5. Bibliografia

- Aker, Aker Secure Roaming, <http://www.aker.com.br>, 2004.
- ALLEN, J; WILSON, J; **Securing a Wireless Network**, Proceedings of the 30th annual ACM SIGUCCS conference on User services, 2002
- ANJUM, F.; TASSIULAS, L.; **Comparative Study of Various TCP Versions Over a Wireless Link with Correlated Losses**, IEEE/ACM Transactions on Networking, Vol. 11, No. 3, 2003.
- BUZKO, D.; LEE, W.; HELAL, A.; **Decentralized Ad-Hoc Groupware API and Framework for Mobile Collaboration**. ACM International Conference on Supporting Group, 2001.
- DANTAS, M: **Tecnologias de Redes de Comunicação de Computadores**, Axcel Books, 2002.
- EDNEY, J.; ARBAUGH, W.; **Real 802.11 Security – Wi-Fi Protected Access and 802.11i**, Addison-Wesley Pearson Education, 2003
- FLUHRER, S.; MANTIN, I.; SHAMIR, A.; **Weakness in the Key Scheduling Algorithm of RC4.**, Cisco Corporation White Papers, 2000
- PARK, J., DERRICK D.; **Wlan Security: Current and Future**, Syracuse University, 2002
- PEIKARI, C., FOGIE, S., NEILSON, B., LANNERSTROM, S; **Wireless Maximum Security**, SAMS Publishing, 2002.
- SCHNEIER, B.; **Applied Criptography**, Wiley, 1997.
- SHIPLEY, M.; **Presentation on War Driving: Open WLANS**, 2001
- SOUMENDRA N.; **Wireless Insecurity / How Johnny Can Hack Your WEP Protected 802.11b Network**, 2002
- WALKER, J.; **Unsafe at any Key Size: An Analisys of the WEP encapsulation**, IEEE 802.11-00/362, 2001.
- Wireless Lan Medium Access Control (MAC) and Phisical Layer (PHY) specifications**, IEEE Computer Society, 2003



Sessão Técnica 12

Redes P2P e Grid II

Obtenção de Tolerância a Falhas na ferramenta de computação MyGrid *

Hélio Antônio Miranda da Silva[†], Carolina Ming Chiao[‡]

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul
Caixa Postal 15064 – 90501-970 Porto Alegre, RS

{hamsilva, cchiao}@inf.ufrgs.br

Resumo. O avanço tecnológico permitiu o surgimento de um novo conceito que é o da computação em Grid e de ferramentas como o MyGrid que dão o suporte necessário para a execução de aplicações nestes ambientes. Este trabalho visa apresentar a utilização da recuperação por retorno como mecanismo para aumentar a disponibilidade e a confiabilidade da execução de tarefas bag of tasks em ambientes de computação em Grid utilizando a ferramenta MyGrid.

1. Introdução

Nos últimos tempos, os avanços tecnológicos consideráveis que ocorreram na computação, como o aumento na velocidade de processamento, o aumento na capacidade de armazenamento e alto desempenho das redes de comunicação, permitiram o surgimento de um novo conceito que é o da computação em Grid. A computação em Grid [Foster et al., 2001] tem emergido como uma área de pesquisa importante por permitir o compartilhamento de recursos computacionais geograficamente distribuídos entre vários usuários. Assim, a computação em Grid está relacionada ao compartilhamento coordenado de recursos e a solução de problemas em uma organização virtual dinâmica, multi-institucional [Foster, 2002].

A computação em Grid permite a criação de organizações virtuais com o compartilhamento de recursos computacionais e de dados, garantindo um acesso controlado e seguro a recursos como capacidade de processamento paralelo e de armazenamento. Todo o acesso de usuários a esses recursos, que estão geograficamente espalhados, é realizado através de uma visão unificada do sistema, isto é, os recursos compartilhados passam a ser vistos como um grande computador virtual. Também podem executar aplicações que demandam grande poder de processamento, compartilhar informações e permitir colaboração entre diferentes instituições. A computação em Grid pode ser utilizada para vários objetivos como, por exemplo, na obtenção de alto desempenho em aplicações que exigem uma grande demanda computacional como simulações climáticas, simulações de fenômenos astrofísicos ou em computações científicas em geral. Outros exemplos de áreas onde as tecnologias Grid podem ser aplicadas são a computação *peer-to-peer* (P2P) e a criação de ambientes virtuais colaborativos [Brunett et al., 1998].

O desenvolvimento de aplicações e ferramentas em ambientes de computação em Grid tem sua complexidade aumentada pela heterogeneidade e pelas freqüentes mudanças que ocorrem nestes ambientes em função do comportamento dinâmico dos seus recursos. Visando amenizar este problema, existem ferramentas como o *Globus Toolkit*

*Parcialmente financiado pelo projeto DepGriFE/HP Brasil P&D

[†]Bolsista financiado pela CAPES.

[‡]Bolsista financiada pelo projeto DepGriFE/HP Brasil P&D

[Foster and Kesselman, 1998] que é um conjunto de componentes de *software* projetados para dar suporte ao desenvolvimento de aplicações que serão executadas em ambientes de computação em Grid. Outro exemplo é a ferramenta *MyGrid* [Cirne et al., 2003b] onde as tarefas são lançadas e coordenadas através de um nó chamado *Home*, seguindo o modelo de computação *bag of tasks* [Cirne et al., 2003a], em que aplicações paralelas são compostas por tarefas independentes umas das outras.

Estes ambientes tratam de uma imensa variedade de recursos espalhados entre vários domínios, tornando o sistema extremamente heterogêneo. A execução de aplicações em ambientes Grid pode envolver centenas e até milhares de nós. Assim, a probabilidade de algum recurso falhar torna-se alta, já que falhas em nós e desconexões tornam-se muito freqüentes [Medeiros et al., 2003]. Por isso, a tolerância a falhas torna-se uma questão chave. Então, para que se tenha um maior rendimento, é necessário que estas plataformas apresentem uma confiabilidade e uma disponibilidade adequadas. Assim, torna-se necessário fazer com que o ambiente se recupere destas falhas e consiga completar sua execução ao mesmo na presença de falhas através de mecanismos que não prejudiquem significativamente o desempenho do sistema[Bosilca et al., 2002].

Este trabalho tem por objetivo definir o funcionamento de um mecanismo de recuperação através de *checkpoints*, que será utilizado no nó central da ferramenta *MyGrid*. Este nó central é responsável pelo escalonamento de tarefas e pelo gerenciamento do sistema. A ferramenta *MyGrid* é uma proposta de plataforma Grid desenvolvida na Universidade Federal de Campina Grande (UFCG). Na atual implementação do *software MyGrid*, existe uma máquina que centraliza o gerenciamento e o escalonamento chamada máquina *Home*. A ocorrência de falhas na máquina *Home* acarreta na perda das aplicações que estavam sendo executadas no Grid, podendo resultar em horas ou até mesmo dias de processamento perdido. Este trabalho, portanto, apresenta a implementação de uma solução para este problema, em que após a ocorrência de uma falha na máquina *Home*, uma nova máquina *Home* será lançada com o último estado consistente do escalonador.

Este trabalho mostra como está sendo realizada a implementação do mecanismo de recuperação no *MyGrid*. Na seção 2 são apresentados o modelo de sistema e o modelo de falhas adotado. Na seção 3 é definido, em linhas gerais, o funcionamento do *MyGrid*. Na seção 4 é mostrado o funcionamento do mecanismo de recuperação. E, por último, na seção 5 são apresentadas as conclusões referentes ao trabalho.

2. Modelo do Sistema

Um Grid pode ser composto por um conjunto de máquinas que estão espalhadas através de muitos domínios. O modelo de computação que irá ser utilizado neste trabalho é o modelo de computação *bag of tasks*. Nele existe um conjunto de tarefas no qual a execução de cada uma destas tarefas é completamente independente das demais, o que implica na inexistência de comunicação entre estas tarefas durante as suas execuções. Também é importante salientar que neste modelo de computação, as tarefas podem ser realizadas em qualquer ordem [Cirne et al., 2003a]. A ferramenta *MyGrid*, que servirá como base de estudos neste trabalho, utiliza este modelo de computação *bag of tasks*.

Quanto ao modelo de falhas, é adotado um modelo de *colapso*, onde, se uma tarefa que está sendo executada em uma máquina for considerada como em falha, toda a sua execução é perdida. A máquina em que esta tarefa estava sendo executada, a princípio não é excluída do Grid, pois a falha gerada durante a execução pode ter sido gerada por problemas no ambiente de execução ou por problemas de configuração da tarefa. Assim, outras tarefas poderão ser lançadas neste nó e ter a sua execução concluída com sucesso, já que existe a probabilidade de que este problema não volte a se repetir. Caso o nó torne-

se inatingível, devido a um *crash* na máquina ou devido a falhas na rede de comunicação, este nó não terá mais tarefas alocadas a ele e passará a ser considerado fora do sistema Grid.

3. Estrutura da ferramenta MyGrid

O *MyGrid* é uma ferramenta que tem por objetivo dar suporte à utilização de recursos em um Grid computacional. Ela visa contornar as dificuldades de gerenciamento e de escalonamento de tarefas em ambientes de computação em Grid. Esta ferramenta permite a execução de aplicações que seguem o modelo de computação *bag of tasks* nestes ambientes. Neste modelo, as tarefas são independentes umas das outras; além disso, a ordem em que são executadas não altera o resultado da computação. A total independência entre as tarefas faz com que não exista comunicação entre elas durante a computação [Cirne et al., 2003a]. A ferramenta *MyGrid* é implementada na linguagem de programação Java e em *scripts shell* e a sua implementação atual permite utilizá-la em várias versões do sistema operacional Linux.

3.1. Funcionamento do MyGrid

A estrutura do *MyGrid* está esquematizada na figura 1. A execução de uma aplicação paralela através do *MyGrid* é lançada através de um nó central do Grid que coordena o lançamento e o escalonamento das tarefas nas máquinas que realizam a computação. Este nó central é chamado de máquina *Home*. As outras máquinas que compõem a estrutura do Grid são chamadas máquinas *Grid*. Cada máquina *Grid* realizará uma, e somente uma, tarefa por vez. Toda comunicação existente dentro do sistema será iniciada pela máquina *Home*.

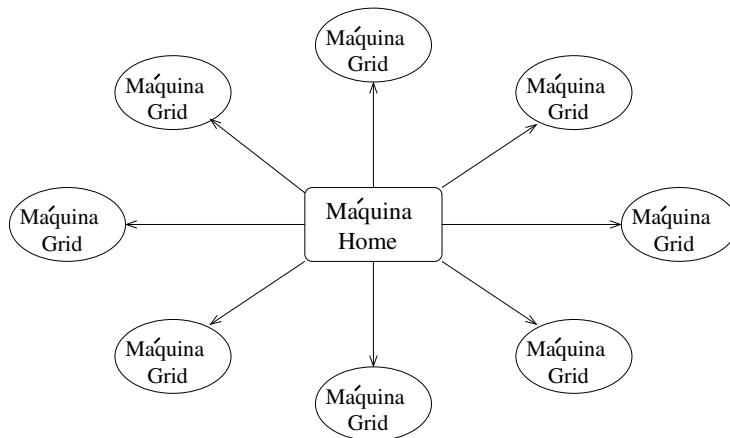


Figura 1: Estrutura do MyGrid

Existem três possibilidades de configuração para máquinas *Grid* que irão definir a estratégia de comunicação entre a máquina *Home* e as máquinas *Grid*. Na primeira delas, uma máquina *Grid* executa um processo chamado *UserAgent* que permite a comunicação do *MyGrid* com o nó. Esta é a estratégia de comunicação implementada pelo próprio *MyGrid*. Na segunda delas, as máquinas *Grid* possuem uma instalação do *Globus Toolkit*. Assim, o *MyGrid* realiza a interação com os serviços disponibilizados pelo *Globus*. E a última possibilidade de estratégia de comunicação é por tunelamento¹ utilizando o software SSH (*Secure Shell*).

¹Palavra usada como tradução do termo *tunneling* do inglês.

3.2. Escalonamento de tarefas

O *MyGrid* possui, como suas principais entradas, um arquivo que define as máquinas que compõem o Grid e um outro arquivo que define todas as tarefas que devem ser executadas. Este conjunto de tarefas é chamado de *Job*, e cada uma das tarefas que compõem o *Job* é chamada de *Task*.

O escalonamento do *MyGrid* utiliza um algoritmo chamado *Workqueue with Replication* (WQR). Neste algoritmo, não é utilizado nenhum conhecimento a respeito do ambiente ou a respeito do tempo de execução das tarefas. No WQR, cada nova tarefa é escalonada para uma máquina *Grid* ociosa, enquanto existirem máquinas ociosas. A cada vez que uma máquina termina a execução ela volta a ser considerada ociosa e pronta para receber mais tarefas. Neste escalonamento, são criadas réplicas das tarefas com o objetivo de obtenção de desempenho [Cirne et al., 2003a]. Assim, para cada tarefa são lançados várias tarefas réplicas, e, após o retorno de qualquer uma das cópias da tarefa, todas as réplicas desta tarefa têm a sua execução interrompida. Mesmo considerando que o objetivo principal da replicação de processos é a obtenção de desempenho, este mecanismo também oferece tolerância a falhas, já que, mesmo que a execução de uma tarefa seja interrompida por intermédio de uma falha, a execução da aplicação paralela poderá continuar, pois possivelmente alguma das outras réplicas desta tarefa não será afetada por esta falha e poderá continuar e concluir a execução desta tarefa em questão.

3.3. Máquina Home como um ponto único de falha

Na estrutura do *MyGrid*, a máquina *Home* apresenta-se como um ponto único de falha, pois caso uma falha ocorra neste nó a execução da aplicação paralela inevitavelmente será interrompida. Este trabalho propõe a realização de pontos de verificação (*checkpoints*) no estado da máquina *Home* como o intuito de sobrepor esta fragilidade do *software MyGrid*. Com isso, o estado da máquina *Home* poderá ser recuperado de forma consistente e a execução deste processo poderá ser continuada no mesmo nó ou em algum outro nó disponível no Grid.

4. Mecanismo de Recuperação

Com o objetivo de aumentar a disponibilidade da máquina *Home* do *MyGrid*, será utilizada a recuperação por retorno. A recuperação por retorno é uma técnica bem conhecida dentro da área tolerância a falhas, onde são realizados periódicos *checkpoints* de um processo. Estes *checkpoints* possuem as informações referentes ao estado deste dado processo. Com isso, a partir deste mecanismo, é possível retornar a execução de um processo a um estado anterior já computado toda vez que o sistema apresenta falha. Com isso, possíveis falhas no nó central do *MyGrid* poderão ser contornadas pelo mecanismo de recuperação, fazendo com que a disponibilidade de sistema seja aumentada, uma vez que este sistema permanecerá uma porção de tempo maior em funcionamento.

A implementação do mecanismo de recuperação será feita através de modificações e incrementações no código fonte do *MyGrid*. Isto é, serão inseridos os procedimentos necessários para que a própria máquina *Home* realize, em momentos necessários, o salvamento em arquivo das informações imprescindíveis para retomar a sua execução na ocorrência de falhas. Serão salvas somente as estruturas dados relevantes para a recuperação do contexto da execução da máquina *Home* permitindo, assim, a seu restabelecimento após qualquer falha.

Os *checkpoints* seriam salvos em um local considerado de armazenamento estável. Este local será em uma outra máquina do Grid ou em sistema de arquivos compartilhados. Assim, o estado do processo poderá ser recuperado em uma outra máquina.

4.1. Salvamento de estados da máquina Home

Conforme explicitado anteriormente, o salvamento de estado da máquina *Home* será realizado inteiramente em nível de aplicação através da modificações no código fonte do *MyGrid*. Nestas modificações são incluídos os procedimentos responsáveis pelo salvamento do estado do nó principal desta ferramenta. A princípio, as estruturas de dados que contêm as informações necessárias para definir o estado da máquina *Home* são as filas de tarefas a serem executadas, assim como os seus estados e as estruturas de dados referentes aos estados das máquinas que compõem o Grid computacional, isto é a composição do ambiente Grid. Portanto, com estas informações será possível recuperar o estado da máquina *Home* que é responsável pelo escalonamento e pelo lançamento das tarefas.

O processo de salvamento de estado será executado a cada mudança de estado do escalonador da máquina *Home*. Uma mudança de estado ocorre a cada nova tarefa lançada e também quando ocorre o retorno de alguma tarefa vindo de alguma réplica que estava sendo executada, seja este retorno realizado por intermédio de falhas ou por término da execução com sucesso. Isto é, cada vez que uma nova tarefa for lançada e a cada vez que ocorrer o retorno de uma execução, um novo salvamento de estado deverá ser realizado. Já que estes freqüentes salvamentos de estado poderiam tornar-se bastante onerosos para a execução do Grid, optou-se pela utilização de registros de *log* visando diminuir o impacto que os estes vários salvamentos de estado podem causar ao desempenho do sistema. Estes *logs* têm como objetivo armazenar mudanças ocorridas no processo da máquina *Home* entre dois salvamentos de estado. Com isso, os *checkpoints* serão realizados após realizados um determinado número de modificações no estado do processo *Home*, como, por exemplo, após a ocorrência de 10 modificações no estado da máquina *Home* será realizado um *checkpoint* e todos os *logs*, assim como, o antigo *checkpoint* serão excluídos. Assim, os registros de *logs* seriam utilizados para registrar os eventos que ocorreriam entre dois sucessivos *checkpoints*.

Um importante questão é quanto às conexões que estarão ativas quando o processo da máquina *Home* falhar. Para poder refazer estas conexões, é necessário que haja um controle de todas as conexões que estão abertas para que, em caso de falhas, estas conexões possam ser recriadas já sendo redirecionadas para a nova máquina *Home*, pois assim todos os nós que estão executando tarefas no Grid poderão retornar seus resultados para o novo processo que estará executando em uma nova máquina *Home*.

5. Conclusão

Dentro do conjunto de aplicações que podem se beneficiar da computação em Grid destacam-se as aplicações *bag of tasks*. O software *MyGrid* é uma plataforma que possui todo suporte necessário para a execução de aplicações deste tipo em ambientes de Grid. Contudo, a estrutura de funcionamento desta ferramenta apresenta a sua máquina *Home*, que centraliza o gerenciamento e o escalonamento de tarefas, como um ponto único de falha. Esta vulnerabilidade deste software merece atenção, pois falhas na máquina *Home* do *MyGrid* podem causar a perda de toda computação que estava sendo realizada no Grid. A recuperação por retorno é técnica bem conhecida dentro da área de tolerância a falhas que se propõe como uma boa solução para contornar esta fragilidade do ambiente de computação proposto pelo *MyGrid*. Assim, este trabalho expõe o projeto e a implementação de um mecanismo de recuperação por retorno neste nó central do *My-*

Grid, desta forma, aumentando a disponibilidade e a confiabilidade da utilização deste ambiente de computação.

Referências

- Bosilca, G., Bouteiller, A., Cappello, F., Djilali, S., Fedak, G., Germain, C., Herault, T., Lemarinier, P., Lodygensky, O., Magniette, F., Neri, V., and Selikhov, A. (2002). MPICH-V: Toward a scalable fault tolerant MPI for volatile nodes. In *SC'2002 Conference CD*, pages 1–18, Baltimore, MD - USA. IEEE/ACM SIGARCH.
- Brunett, S., Czajkowski, K., Fitzgerald, S., Foster, I., Johnson, A., Kesselman, C., Leigh, J., and Tuecke, S. (1998). Application experiences with the globus toolkit. In *Proceedings of 7th IEEE Symp. on High Performance Distributed Computing*, pages 81–89.
- Cirne, W., Brasileiro, F., Sauvé, J., Andrade, N., Paranhos, D., Santos-Neto, E., and Medeiros, R. (2003a). Grid computing for bag of tasks applications. *Proceedings of the Third IFIP Conference on E-Commerce, E-Business and E-Goverment*.
- Cirne, W., Paranhos, D., Costa, L., Santos-Neto, E., Brasileiro, F., Sauvé, J., da Silva, F. A. B., Barros, C. O., and Silveira, C. (2003b). Running bag-of-tasks applications on computational grids: The mygrid approach. In *Proceedings of the ICCP'2003 - International Conference on Parallel Processing*, page 407.
- Foster, I. (2002). What is the grid? a three point checklist. *GRIDToday*, 1(6).
- Foster, I. and Kesselman, C. (1998). The globus project: A status report. *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pages 4–18.
- Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150:1.
- Medeiros, R., Cirne, W., Brasileiro, F., and Sauvé, J. (2003). Faults in grids: Why are they so bad and what can be done about it? *Proceedings of the 4th International Workshop on Grid Computing*, page 18.

Estudo da Tecnologia P2P Visando sua Aplicação em Sistemas Baseados em Agentes

Elder Rizzon Santos

José Fernando de Lacerda Machado Junior

Universidade Federal do Rio Grande do Sul

Instituto de Informática

Av. Bento Gonçalves, 9500, Bloco IV, Prédio 43424 – Porto Alegre – RS

{ersantos,fmachadoj}@inf.ufrgs.br

Resumo

O presente trabalho visa relatar o panorama de redes peer-to-peer na atualidade e também expor aplicações de agentes inteligentes dentro da peculiar arquitetura das redes estudadas, dando uma visão superficial de arquitetura do sistema, melhoramentos possíveis e propostas com tecnologia de agentes.

1. Introdução

O presente trabalho visa estudar a tecnologia peer-to-peer (aqui tratados como P2P) para estabelecer pontos de convergência com os sistemas baseados em agentes. Esta convergência pode apresentar benefícios para ambas, já que algumas deficiências de infra-estrutura para sistemas baseados em agentes podem ser supridas por sistemas P2P e estes podem beneficiar-se das técnicas já estudadas e aceitas em Inteligência Artificial, nas quais estão incluídas os agentes inteligentes.

Inicialmente será apresentada uma introdução aos sistemas P2P; na seção 2 será apresentado um estudo mais detalhado sobre arquiteturas e funcionalidades dos sistemas P2P; a seção 3 apresenta uma reflexão sobre a união de sistemas P2P com sistemas baseados em agentes, além da apresentação de um projeto de implementação das tecnologias conjugadas; finalmente, são apresentadas as conclusões do estudo realizado.

2. Sistemas Peer-to-Peer

Apesar de bastante explorada, a tecnologia de P2P ainda gera controvérsias em relação a sua classificação. Segundo [MIL02], existe uma concepção errada de que sistemas de processamento distribuído não se caracterizam como P2P pois apresentam servidor que centraliza os recursos computacionais, sendo assim, os *peers* não funcionam como servidores e não ocorre comunicação entre eles. Porém, grande parte do processamento é realizado nos *peers*, com grande autonomia, o que permite que seja feita a caracterização de sistemas de processamento distribuído como P2P.

Porém [SAR01] defende que sistemas P2P se diferenciam de outros sistemas distribuídos devido à simetria do sistema, já que todos os *peers* atuam como servidor e cliente e não há uma centralização no serviço de troca, exceto na indexação.

Sistemas P2P geralmente não apresentam uma infra-estrutura centralizada, eles dependem da participação voluntária dos indivíduos (*peers*) para contribuir com recursos a

partir dos quais a infra-estrutura é estabelecida [SAR01]. A participação em um sistema P2P é *ad-hoc* e dinâmica, sendo assim, o desafio de tais sistemas é determinar um mecanismo e uma arquitetura para organizar os indivíduos de tal maneira que possam cooperar e, desta forma, fornecer um serviço útil para toda a comunidade de usuários.

Em [KAN02] são propostas cinco dimensões para a classificação de sistemas P2P sendo elas as seguintes: armazenamento de dados: centralizado ou distribuído; controle dos recursos: centralizado ou distribuído; utilização dos recursos: isolado ou colaborativo; controle do estado global: fraco ou forte; QoS: fraco, moderado ou forte. O autor também afirma que dificilmente um sistema P2P irá se enquadrar em algum destes valores extremos, já que estes são meramente ilustrativos, dependendo, então, de detalhes da implementação.

As dimensões para classificação são úteis para observar em detalhes a funcionalidade de um sistema P2P, além de servirem como base para comparações entre os diversos tipos de sistemas.

2.1. Peer-to-peer aplicada - Arquiteturas comuns e classificação

As arquiteturas peer-to-peer se permitem ser classificadas de duas maneiras, sendo a primeira quanto ao grau de centralização e a segunda quanto à estrutura da rede, tipos bastante parecidos, sendo um com foco na estrutura sistêmica e outro com foco na estrutura da camada de rede em nível de aplicação. O grau de centralização, como proposto por [AND02], pode se apresentar como:

- Puramente descentralizado: Arquiteturas P2P puramente descentralizadas, como o *Gnutella* original e o *Freenet*, são caracterizadas pelo mesmo grau de autonomia de todos os nós da rede, ou seja, todos os *peers* realizam as mesmas tarefas, atuando tanto como clientes como servidores, sem coordenação central de suas atividades. O termo SERVENTS é proposto para este tipo de aplicativo (SERVers+cliENTS), já que os clientes atuam como servidores também.
- Parcialmente centralizado: Exemplificada através de aplicativos famosos, como o *Kazaa* e o *Morpheus*, este tipo de arquitetura é bastante parecido com a puramente descentralizada, exceto pelo fato de existirem *peers* responsáveis pela indexação do que está sendo compartilhado pelos outros *peers*, ocorrendo uma diferenciação hierárquica entre os membros. Estes indexadores são designados dinamicamente, não comprometendo assim a segurança do sistema no caso de este vir a ser atacado ou em caso de falha.
- Hibridamente Descentralizado: A presença de um servidor central, onde são armazenados dados referentes a todos os arquivos compartilhados pelos outros *peers* diferencia este tipo de arquitetura, que tem como mais famoso representante o extinto *Napster*. Há então a necessidade de uma conexão direta de todos os *peers* a um serviço centralizador de busca.

Milojcic, em [MIL02], trata estas classificações de maneira sintética, como *modelo de diretório centralizado*, *modelo de requisição por inundação* e *modelo de roteamento de documento*, que vem a ser uma classificação mais objetiva em relação à aplicação.

Quanto à estrutura da camada de rede em nível de aplicação, redes P2P possuem grande complexidade topológica e dinamismo, sendo possível a diferenciação dos sistemas através da forma que as redes formam a camada que sobrepõe a estrutura física. Esta

diferenciação se dá pelo grau de estruturação da rede, que pode ser de *ad hoc* para uma com um grau de planejamento. A caracterização proposta por [AND02] enquadra-se em uma das três seguintes:

- Redes desestruturadas: aqui se enquadra a clássica rede *peer-to-peer*, uma rede *ad hoc*. A localização de arquivos independe da camada de rede em nível de aplicação, sendo assim, a busca é ineficiente e se dá de forma distribuída e randômica, inundando a rede com pacotes *query*, porém é possível acomodar um grande número de nós sem sobrecarregar um específico.
- Redes estruturadas: as redes P2P estruturadas permitem um melhor aproveitamento da banda de transmissão de dados, uma vez que a centralização do conteúdo dos *peers* permite uma busca mais precisa e sem a necessidade de sobrecarga da rede com *floods* de *queries*. O ponto negativo fica por conta da falha **total** da rede no caso da indisponibilidade do servidor de indexação.
- Redes parcialmente estruturadas: a criação de supernós, que ficam responsáveis pela indexação de parte do conteúdo compartilhado pelos *peers* permite que não haja sobrecarga de um único nó, porém o mecanismo de busca não é tão eficiente quanto ao de uma rede estruturada. A grande vantagem é que a promoção de *supernodes* é feita de forma dinâmica, permitindo uma tolerância grande a falhas e ataques.

2.2. Tecnologia Comparada

Comparando-se as diversas tecnologias P2P presentes, pode-se observar diferenças em relação aos recursos, na sua descoberta, gerenciamento e descrição, como proposto por [CIM02]. Quanto à descoberta, o autor propõe a diferenciação entre o tipo de recursos buscados por cada aplicação, sendo assim, alguns buscam serviços, como o JXTA [JXT03], outros buscam arquivos, como o *Gnutella* e o *Freenet*.

O projeto desenvolvido pela Sun Microsystems [JXT03] para a criação de serviços em um ambiente P2P serve como uma plataforma para a execução de aplicativos de forma distribuída, seja processamento distribuído ou interação colaborativa. Existem vários aplicativos disponíveis para execução sobre a plataforma JXTA que permitem a utilização dos recursos providos pela estrutura do sistema de diversas maneiras, sendo elas de maneira isolada, onde um *peer* isolado oferece recursos, ou então através de um *cluster* de *peers*, que podem prover um ou mais recursos, de maneira transparente.

Os outros clientes de troca de arquivos, aplicações P2P clássicas oferecem apenas serviço de troca de arquivos e indexação. Os mecanismos de busca também não são muito eficientes, sendo no máximo razoáveis para o fim que se propõem. Através desta observação, têm-se o projeto JXTA como base e inspiração da integração de SMA e redes P2P.

Quanto ao gerenciamento destes recursos, as arquiteturas que trabalham com arquivos devem monitorar a utilização de banda e evitar sobrecarga, uma vez que os sistemas deve responder ainda às buscas; já as arquiteturas que trabalham com serviços devem monitorar e manter uma lista dos mesmos, bem como resultados de medição de desempenho.

No que toca à descrição, arquiteturas que tratam com arquivos devem manter um identificador único, um *hash*, mantendo assim a integridade da transferência em caso de interrupção. Já arquiteturas de serviços possuem documentos XML com *metadados* identificando e descrevendo os serviços disponíveis e outros dados relevantes para os peers.

3. Aplicação da tecnologia P2P em sistemas baseados em agentes

Sistemas baseados em agentes geralmente estão relacionados aos Sistemas MultiAgentes (SMA), uma sub-área da Inteligência Artificial Distribuída (DAI) [SYC98]. Uma definição fundamental para qualquer estudo nesta área é o conceito de agente. Apesar da maioria dos pesquisadores de SMA, discordarem quanto a uma definição formal de agente, a maioria concorda que a sua propriedade principal é a capacidade de comunicar-se com outros agentes, independente de quem sejam e onde estejam. Em [FRA96] o autor apresenta uma discussão sobre diversas definições e reflete sobre o motivo para tal divergência.

Conforme apresentado anteriormente, um aspecto fundamental dos SMAs é a capacidade de interação dos agentes. Existem diversos modelos para a interação [YEP03]. A troca de mensagens utilizando a Linguagem de Comunicação entre Agentes¹ (ACL) é um dos modelos mais flexíveis e extensíveis.

Uma possibilidade de utilização de P2P em SMA decorre da própria meta de um SMA, que é, a partir de pequenas unidades resolvidoras de problemas – os agentes – ser capaz de resolver problemas complexos, através da cooperação, coordenação e planejamento. Considerando cada agente um *peer*, é possível mesclar as técnicas de IA presentes nos agentes às infra-estruturas P2P conhecidas, para, por exemplo, compartilhamento de recursos (podendo utilizar processamento de *peers* ociosos, obtenção de informações, etc.).

Um *peer*, isoladamente, através da sua autonomia de processamento pode ser capaz de criar ontologias e selecionar grupos de interesse de acordo com as informações recebidas de outros *peers*. Com isso espera-se a otimização de mecanismos de busca, como o proposto por [BER02] e derivações que podem advir destes conceitos.

3.1. Integração P2P + SMA – Estudo de Caso

Existem propostas de aplicações de agentes em redes P2P para busca de informações, como em [BER02], onde o sistema denominado SEWASIE (*SEmantic Webs and AgentS in Integrated Economies*), que tem como objetivo um mecanismo de busca avançado em fontes de dados heterogêneas via enriquecimento semântico, permite ser adaptado para funcionamento sobre um ambiente P2P. A proposta prevê um cliente onde o usuário se depara com uma interface de fácil utilização para busca de informações na internet, de diversas origens e em diversos formatos.

Inicialmente, o sistema foi projetado para utilização em ambientes estáticos. Apresenta três elementos, sendo eles um agente de busca (*query agent*), um agente negociador (*brokering agent*) e um SINode (*SEWASIE Information Nodes*). Os agentes são responsáveis pela manipulação de dados, tanto de descoberta como de informação aos nodos, enquanto o SINode fica encarregado de armazenar os dados e construir as

¹ A linguagem de comunicação é o meio no qual são comunicadas as atitudes referentes ao conteúdo sendo transmitido (uma solicitação, uma afirmação, uma consulta, etc.) [LAB99].

ontologias.

Os agentes de busca são responsáveis pela busca de informações dos clientes nos SINodes. Apenas transportam o conteúdo da busca do SINode para o cliente.

Os SINodes apresentam um repositório de informações (*Virtual Data Store*), que concentra informações sobre os dados disponíveis, um construtor de ontologias (*Ontology Builder*) responsável pela associação de dados e um gerenciador de busca (*Query Manager*), encarregado de administrar as chamadas dos clientes [BER02].

Já os *brokering agents* ficam encarregados de acessar os SINodes próximos para retirar informações sobre o conteúdo armazenado nesses e gerar informações para busca. Neste momento, os *brokering agents* assumem a responsabilidade de enriquecer o conteúdo do seu SINode, aumentando assim a eficiência da busca.

Há duas propostas de implementação em redes P2P apresentadas em [BER02], através da criação de redes alternativas, uma onde agiriam os agentes negociadores e uma rede onde estariam interligados os SINodes. Na primeira proposta vários agentes negociadores agiriam de forma a localizar os conteúdos e criar derivações. Na outra proposta, apenas um agente negociador teria conhecimento do sistema inteiro, sendo responsável por mapear as informações entre os SINodes envolvidos.

Porém ainda permanecem alguns desafios, observados em [BER02], como um protocolo consistente para a troca de informações, suporte semi-automático para geração de fórmulas de coordenação, as políticas de propagação de informações para os outros *peers* e a disponibilização destas informações.

4. Conclusão

Com o estudo realizado neste artigo foi possível observar alguns pontos de convergência entre a tecnologia P2P e SMA. A aplicação de SMA em P2P ainda é um paradigma novo na área de redes de computadores, porém permite propostas bastante interessantes, como o sistema de busca SEWASIE.

Observando a abordagem arquitetural, observa-se que as aplicações P2P clássicas oferecem apenas serviço de troca de arquivos e indexação. Os mecanismos de busca também não são muito eficientes, sendo no máximo razoáveis para o fim que se propõem. Através desta observação, têm-se o projeto JXTA como base e inspiração da integração de SMA e redes P2P.

O sistema de busca explorado por [BER02], o SEWASIE parece bastante eficiente e permite implementação em sistemas P2P. Dentro da proposta, é possível criar derivações dos modelos propostos pelo autor para o funcionamento do mecanismo de busca dentro de uma arquitetura P2P.

Como trabalho futuro, pode-se citar uma abordagem derivada do mecanismo de busca SEWASIE, devidamente enriquecido e com mecanismos de coordenação eficientes.

5. Referências

- [AND02] ANDROUTSELLIS-THEOTOKIS, S.. **A Survey of Peer-to-Peer File Sharing Technologies**. White paper, 2002.
http://www.eltrun.aueb.gr/whitepapers/P2P_2002.pdf

- [BER02] BERGAMASCHI, S, GUERRA, F. **Peer-to-peer Paradigm for Semantic Search Engine.** 2002. Disponível em <http://citeseer.ist.psu.edu/bergamaschi02peertopeer.html>
- [CIM02] CIMINIERA, L.; SANNA, A.; ZUNINO, C. **Survey on grid and peer-to-peer network technologies.** European Grid of Solar Observations (EGSO), Document EGSO-DE01_01D02-021001, Outubro 2002.
http://www.mssl.ucl.ac.uk/grid/twiki/pub/Main/AndreaSanna/Technology_Survey.pdf
- [FRA96] FRANKLIN S., GRAESSER A. **Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents.** In: Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, Springer-Verlag, 1996.
- [JXT03] **The JXTA Project Website.** Sun Microsystems, 2003 <http://www.jxta.org>
- [KAN01] KANT, K., MOHAPATRA, P. **Current Research Trends in Internet Servers,** PAWS-2001 proceedings (Performance Evaluation Reviews, Vol. 29, no 2), Sept 2001, pp 5-7.
- [KAN02] KANT, K., IYER, R., TEWARI, V. **A framework for classifying peer-to-peer Technologies,** 2nd IEEE/ACM Intl. Symposium on Cluster Computing and the Grid, May 21-26, 2002, Berlin, Germany.
- [LAB99] LABROU Y., FININ T., PENG Y. **Agent Communication Languages: the Current Landscape.** In: Intelligent Systems, volume 14, number 2, March/April 1999, IEEE Computer Society.
- [MIL02] MILOJICIC, D. *et al.* **Peer-to-peer computing.** Technical Report HPL-2002-57, HP Lab, 2002. <http://citeseer.ist.psu.edu/milojicic02peertopeer.html>
- [PET96] PETRIE, C. **Agent-Based Engineering, the Web, and Intelligence.** In: IEEE Expert: Intelligent Systems and Their Applications Volume 11 , Issue 6 (Dezembro de 1996) Pg: 24 – 29.
- [SAR01] Stefan Saroiu, P. Krishna Gummadi and Steven D. Gribble: **A Measurement Study of Peer-to-Peer File Sharing Systems.** Technical Report UW-CSE-01-06-02 University of Washington, Seattle, WA, USA, July 2001.
- [SYC98] SYCARA K. **Multiagent Systems.** In: AI Magazine volume 19, Nº 2 Intelligent Agents, 1998.
- [YAN01] YANG, B., GARCIA-MOLINA, H. **Comparing hybrid peer-to-peer systems.** In Proc. of the 27th International Conference on Very Large Data Bases, Rome, Italy, September 2001.
- [YPE03] YEPES I., BARONE D. **Inteligência Artificial distribuída: Uma Abordagem ao Comportamento Social Inteligente.** In: Sociedades Artificiais: A Nova Fronteira da Inteligência nas Máquinas, Porto Alegre: Ed. Bookman, 2003.

Uma Ferramenta para Execução de Simulações em Larga Escala

André Detsch, Gerson G. H. Cavalheiro, Luciano P. Gaspari

Programa de Pós-Graduação em Computação Aplicada (PIPCA)
Universidade do Vale do Rio dos Sinos (UNISINOS)

Resumo. Este artigo apresenta uma ferramenta com funcionalidades direcionadas à execução de um grande volume de simulações, atuando em todas as fases, desde a definição dos experimentos a serem executados até a geração de gráficos a partir dos resultados obtidos. A partir da especificação de um conjunto de experimentos, é possível realizar a sua execução de forma paralela em diversas máquinas de um laboratório, agilizando o processo como um todo. O desenvolvimento e a evolução da ferramenta baseou-se na experiência prática de uso em diversos estudos, mostrando-se bastante útil na automatização de etapas que, em geral, seriam realizadas de forma manual.

1. Introdução

A execução de simulações computacionais é uma das mais importantes formas de avaliação de novos protocolos e mecanismos de rede. Entretanto, estudos completos envolvendo simulação tipicamente exigem um grande poder computacional, especialmente pelo fato de que, durante o processo de avaliação, inúmeras execuções independentes se fazem necessárias, variando os parâmetros de entrada (por exemplo, número de *hosts* e largura de banda) e as sementes aleatórias (para que uma boa precisão estatística seja obtida).

Este artigo apresenta uma ferramenta que possibilita ao pesquisador uma fácil descrição e paralelização das simulações em um conjunto de estações e, ao mesmo tempo, propicia facilidades de geração automatizada de gráficos, utilizando linguagens de descrição simples e similares entre si. Por empregar a infra-estrutura já instalada de compartilhamento de arquivos via NFS (*Network File System*, [1]), sua implementação e, principalmente, seu uso se tornam extremamente simples, o que incentiva sua utilização mesmo por quem não possui conhecimentos avançados de processamento distribuído.

A Seção 2 apresenta a arquitetura da ferramenta, apresentando, além dos módulos e suas interações, as linguagens utilizadas. A Seção 3 apresenta um exemplo realístico de uso, descrevendo os passos típicos que envolvem a utilização da ferramenta. Por fim, a Seção 4 encerra o artigo com algumas considerações finais.

2. Arquitetura da ferramenta

A ferramenta é composta de três componentes básicos que são executados independentemente:

- *módulo de geração de tarefas*: interpreta o arquivo contendo a descrição dos experimentos e gera as tarefas correspondentes;
- *módulo de execução*: executado em diversas máquinas em paralelo, consome uma-a-uma as tarefas geradas, realizando a sua execução localmente;
- *módulo de geração de gráficos*: a partir das saídas geradas e de um arquivo de descrição, gera os gráficos no formato indicado.

A Figura 1 ilustra a interação entre os elementos da arquitetura, bem como o fluxo relativo ao processo de execução dos experimentos e plotagem dos resultados. Primeiramente (etapa 1 da figura) o usuário define textualmente no arquivo de descrição, através de linguagem específica (definida na Seção 2.1), os experimentos a serem executados. Acionando-se o módulo de geração de tarefas, o arquivo de descrição é interpretado, e as respectivas tarefas são geradas através da criação de arquivos (cujo nome representa o *id* único da tarefa) no diretório Waiting/Tasks (2). Este diretório (a exemplo dos demais envolvidos no

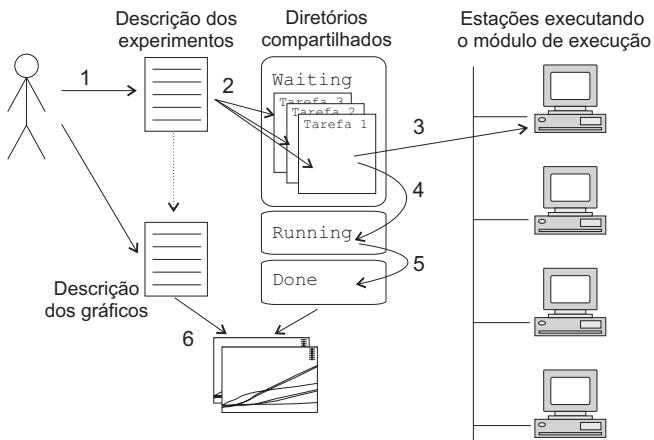


Figura 1: Diagrama de funcionamento da ferramenta

processo aqui descrito) é compartilhado através de NFS entre todas as máquinas que irão contribuir na execução dos experimentos. Nessas máquinas é acionado o módulo de execução, tipicamente com o auxílio de *login* remoto (*Secure Shell* - SSH, por exemplo). As instâncias do módulo monitoram o diretório *Waiting/Tasks* em busca de tarefas para executar (3). Assim que uma tarefa apta a ser executada é identificada, ela é transferida para o diretório *Running/Tasks* (4) e o comando relativo à tarefa é executado localmente. Nessa etapa, as saídas padrão (*stdout*) e de erro (*stderr*) do comando são redirecionadas para uma arquivo específico (identificado pelo *id* da tarefa) nos diretórios *Running/Output* e *Running/Error*, respectivamente. Quando a execução da tarefa é terminada, o respectivo arquivo é transferido para o diretório *Done/Tasks* (5), e os arquivos de saída para os diretórios *Done/Output* e *Done/Error*.

A criação dos gráficos (6) utiliza os resultados gerados (no diretório *Done/Output*) e o arquivo de descrição de gráficos. Esse arquivo é criado utilizando a linguagem apresentada na Seção 2.3, que se baseia fortemente na linguagem de descrição de experimentos (Seção 2.1), o que permite reaproveitar boa parte da codificação já realizada.

2.1. Linguagem para definição dos experimentos

O conjunto de simulações a ser realizado pela ferramenta é descrito através de uma linguagem simples, que provê recursos que facilitam a especificação. A seguir são listadas as suas características principais, tendo como base um exemplo fictício, onde é recriada a avaliação de um mecanismo ou aplicação qualquer variando-se o protocolo utilizado e o número de nodos envolvidos.

Definição de um bloco de tarefas

Durante a interpretação do arquivo, tarefas são geradas sempre que um comando `dtask` é encontrado no arquivo de descrição. Isto permite que diversos blocos de tarefas sejam definidos em um mesmo arquivo, de acordo com os diferentes comandos interpretados até a chamada de cada `dtask`. Opcionalmente, pode-se definir um nome para o bloco de tarefas sendo gerado (por exemplo `dtask BLOCO_TCP`). Isto permite a posterior definição de dependência com o bloco de tarefas, conforme ilustrado a seguir, na especificação de requisitos.

Comandos básicos

Os comandos básicos `parameters`, `defaults` e `commandbase` são obrigatórios na especificação de um conjunto de experimentos. `Parameters` define o nome dos parâmetros bem como a ordem na qual serão passados para o simulador, enquanto `defaults` atribui valores padrão para esses parâmetros. Já `commandbase` define o comando de chamada

do simulador. O Código 1 apresenta um exemplo que usa os três comandos. A interpretação do exemplo levaria a geração de apenas uma tarefa cujo comando chamado é “ns simulacao.tcl TCP 30 0”.

Código 1 Uso dos comandos básicos.

```

1 parameters protocolo nodos semente
2 defaults TCP 30 0
3 commandbase ns simulacao.tcl
4 dtask

```

Definição da variação dos parâmetros

A definição de diferentes simulações a serem executadas se dá através da variação dos valores dos parâmetros passados para o simulador. Essa variação é especificada através da primitiva **values**, passando-se o nome do parâmetro em questão e os valores desejados, conforme exemplificado nas linhas 4 e 5 do Código 2.

Código 2 Exemplo de variação de parâmetros.

```

1 parameters protocolo nodos semente
2 defaults TCP 30 0
3 commandbase ns simulacao.tcl
4 values protocolo TCP UDP
5 values nodos 10 20
6 dtask

```

Com a adição dessas duas linhas, passam a ser geradas não apenas uma tarefa, mas o resultado do produto cartesiano das variações definidas, no caso, quatro tarefas: “ns simulacao.tcl TCP 10 0”, “ns simulacao.tcl TCP 20 0”, “ns simulacao.tcl UDP 10 0” e “ns simulacao.tcl UDP 20 0”. Pode-se ainda utilizar comandos Python ([2]) que retornem uma lista de valores no lugar de escrevê-los um a um. Por exemplo, é possível variar o parâmetro “semente” utilizando o comando **values semente range(10)**, de forma que o valor de semente será variado dentre todos os valores inteiros no intervalo [0, 9].

Delimitação de escopo

Um recurso importante para facilitar a especificação de uma grande variedade de experimentos é a delimitação de escopos, utilizando os caracteres { e }. Sempre que um escopo é encerrado, todas as especificações realizadas dentro desse escopo são descartadas. Isto se mostra bastante útil quando conjuntos de experimentos diferentes compartilham apenas algumas especificações (por exemplo, os parâmetros e algumas variações), mas não outras. O Código 3 ilustra essa funcionalidade. A interpretação do código gera quatro tarefas: “ns simulacao.tcl TCP 30 0”, “ns simulacao.tcl UDP 30 0”, referentes ao primeiro bloco; e “ns simulacao.tcl TCP 10 0”, “ns simulacao.tcl TCP 20 0”, referentes ao segundo bloco.

Código 3 Exemplo de definição de escopo

```

1 parameters protocolo nodos semente
2 defaults TCP 30 0
3 commandbase ns simulacao.tcl
4 {
5     values protocolo TCP UDP
6     dtask
7 }
8 {
9     values nodos 10 20
10    dtask
11 }

```

Definição de macros

Macros são definidas através do comando **macro**. Por exemplo, **macro numeros_nodos 100 200** define uma macro **numeros_nodos** com o valor **100 200**. Macros podem

também conter diversas linhas, bastando delimitá-las com operadores de escopo (`{ e }`). Macros são acessadas utilizando o caracter `!` antes do identificador da macro. Por exemplo, o comando `values nodos !numero_nodos` seria expandido para `values nodos 100 200`.

Especificação de requisitos

Uma facilidade mais avançada na especificação de simulações é a definição de requisitos para um conjunto de tarefas. Os requisitos são especificados através da primitiva `requires`, seguido de um dos tipos de requisitos:

- **cpu:** freqüência mínima, em *megahertz*, da máquina na qual a tarefa pode ser executada. Exemplo: `requires cpu 1000`.
- **mem:** quantidade mínima de memória RAM exigida (em *megabytes*). Exemplo: `requires mem 256`.
- **ips:** lista de localidades (endereços IP) que podem executar a tarefa. Exemplo: `requires ips 10.16.165.152 10.16.165.153`.
- **tasks:** blocos de tarefas que devem ser executadas antes do bloco atual. Exemplo: `requires tasks BLOCO_TCP`.

2.2. Arquivos de tarefas

Os arquivos de tarefas são gerados automaticamente pelo módulo de geração de tarefas. O formato desses arquivos é bastante simples e tem como obrigatório apenas uma linha que contenha o comando a ser executado, no formato `RUN: <comando>`, por exemplo, `RUN: ns simulação.tcl TCP 20 0`. Adicionalmente, podem haver linhas de especificação de requisitos, como `REQUIRED_CPU: <MHz>` e `REQUIRED_MEM: <MB>`, inseridas (pelo módulo de geração de tarefas) de acordo com os requisitos especificados pelo usuário. Quando presentes, estas linhas são avaliadas pelo módulo de execução antes do processamento da tarefa. Caso algum requisito não seja atendido, a tarefa é mantida no diretório `Waiting/Tasks`, e outra tarefa é procurada.

2.3. Processamento de resultados e geração de gráficos

Aliado à geração e processamento distribuído das simulações, foi desenvolvida uma ferramenta (*front-end* para o Gnuplot, [3]) que, utilizando uma linguagem similar à apresentada na Seção 2.1, permite a fácil geração de gráficos a partir dos resultados obtidos após a execução das simulações. Enquanto na linguagem de definição de experimentos existe um comando `dtask` que, baseado nas variações definidas, gera as tarefas correspondentes, na linguagem de definição de gráficos existe um comando `dplot` que gera um gráfico, onde cada uma das combinações é representada por uma linha no gráfico gerado.

As funcionalidades de definição de macros, delimitação de escopo e variações dos parâmetros (comando `values`) são tratados da mesma forma. O comando `parameters` também possui a mesma funcionalidade, porém a lista definida deve conter não apenas os parâmetros de entrada do simulador mas também os campos de retorno, que contém os resultados. Os comandos `defaults`, `commandbase` e `requires` são ignorados. Por outro lado, existem comandos que são utilizados apenas na definição dos gráficos. São eles:

- **datafile:** define um ou mais arquivos que contém os dados de entrada para geração dos gráficos;
- **project:** define os dois parâmetros projetados nos eixos X e Y do gráfico. Tanto em X quanto em Y podem ser usadas expressões que definam o eixo, conforme exemplificado na linha 5 do código 4;
- **projectiontype:** especifica a semântica de agrupamento dos valores no eixo Y. O tipo padrão é `mean`, onde é plotada a média dos valores do parâmetro em Y para cada valor de X. Os tipos `max`, `min` e `count` projetam, respectivamente, o valor

máximo, mínimo e o número de ocorrências do valor em X. Já o tipo `confidence` plota, além de uma linha sobre a média dos valores, o intervalo de confiança para 95% de significância.

Adicionalmente, podem ser usados comandos Gnuplot diretamente, permitindo uma total liberdade na formatação dos gráficos, conforme exemplificado na linha 7 do Código 4.

Código 4 Exemplo de descrição de gráfico

```

1 parameters protocolo nodos semente bytes_enviados tempo
2 values protocolo TCP UDP
3 values nodos 10 20
4 datafile resultados1.txt resultados2.txt
5 project nodos bytes_transmitidos / tempo
6 projectiontype confidence
7 set title "Variação da vazão em função do número de nodos"
8 dplot grafico.eps

```

3. Utilização da ferramenta

Esta seção apresenta um exemplo ilustrativo do uso da ferramenta, utilizando o simulador ns-2. O exemplo é baseado nos experimentos realizados em [4], onde cinco modelos básicos de protocolos baseados em *polling* (PET, PeP, PSEW, RBP, PrP) foram avaliados, buscando-se observar e comparar o comportamento desses modelos com a variação, entre outros parâmetros, do número de receptores e tamanho da janela deslizante empregada. O produto final dos experimentos será apresentado em três gráficos: vazão em função do tamanho da janela, custo de rede e vazão em função do número de receptores. Cada um dos gráficos contém cinco linhas, uma para cada protocolo testado.

3.1. Preparação dos experimentos

O primeiro passo consiste no preparo do arquivo de descrição de experimentos. Para o estudo exemplificado nesta seção, o arquivo respectivo é apresentado no Código 5. De forma consistente com esse arquivo, o simulador deve receber os parâmetros na ordem indicada. No caso do ns-2, onde é utilizado um script tcl para instanciação dos experimentos, são utilizadas linhas do tipo `set protocolo [lindex $argv 0]`, uma para cada parâmetro.

Código 5 Arquivo de descrição de experimentos

```

parameters protocolo receptores janela semente
defaults PET 100 64 0
commandbase ns poll.tcl
values protocolo PET PSEW RBP PeP PrP
values semente range(0,10) # 10 valores, de 0 a 9
# avaliação do impacto da variação do tamanho da janela na vazão
{
    values janela 50 100 150 200 250 300 350 400 450 500
    dtask
}
# avaliação do impacto do número de receptores no custo e vazão
{
    {
        values receptores 1 50 100
        dtask
    }
    values receptores 200 300 400
    requires mem 512 dtask
    dtask
}

```

3.2. Instanciação dos experimentos

Uma vez preparado o arquivo de descrição, as respectivas tarefas podem ser geradas acionando-se o *módulo de geração de tarefas*, através do comando `dtask gen`

<arquivo_de_descricao>. Este comando pode ser executado em qualquer máquina com acesso aos diretórios de tarefas compartilhados. Para que a execução dos experimentos tenha início, deve-se ativar instâncias do *módulo de execução* em diferentes máquinas, através do comando `dtask server`.

3.3. Geração dos gráficos

O Código 6 apresenta o arquivo de descrição referente à geração dos gráficos do exemplo aqui utilizado. O arquivo de dados definido, `resultados.txt`, é construído concatenando-se os resultados escritos nos arquivos em `Done/Output`, tipicamente através de um comando `cat Done/Output/* > resultados.txt`. Vale ressaltar que o formato de saída dos resultados por parte do simulador deve ser consistente com esse arquivo, ou seja, as linhas exibidas devem conter os campos indicados no comando `parameters` do arquivo de descrição de gráficos. O acionamento do *módulo de geração de gráficos* é realizado utilizando-se o comando `dplot <arquivo_de_descricao>`, que pode ser executado em qualquer máquina com acesso ao arquivo de dados indicado.

Código 6 Arquivo de descrição dos gráficos

```
parameters protocolo receptores janela semente custo tempo
datafile resultados.txt
values protocolo PET PSEW RBP PeP PrP
{
    # um ponto por protocolo para cada tamanho de janela
    project janela custo/tempo      # janela X custo/tempo
    set xlabel "Tamanho da Janela"   # comando gnuplot
    set ylabel "Vazão (Kbits/s)"     # comando gnuplot
    dplot janela_vazao.eps          # gera o gráfico
}
set xlabel "Número de Receptores"
{
    project receptores custo
    set ylabel "Custo de Rede (Mbits)" "
    dplot receptores_custo.eps
}
project receptores custo/tempo
set ylabel "Vazão (Kbits/s)"
dplot receptores_vazao.eps
}
```

4. Considerações finais

Este artigo apresentou uma ferramenta moldada em função das necessidades específicas de estudos que envolvem simulação, em especial, na área de redes de computadores. Desenvolvida em paralelo com diversos projetos de pesquisa que fizeram uso de suas facilidades, o conjunto de módulos que a compõe se mostra bem adaptado aos diferentes requisitos de diferentes estudos. A ferramenta, bem como sua documentação e exemplos mais avançados, podem ser acessados através do site <http://www.gobolinus.org/~detsch/dtools>.

Referências

- [1] Network File System Protocol Specification, RFC 1094, <http://www.ietf.org/rfc/rfc1094.txt>
- [2] Python Programming Language, <http://www.python.org>.
- [3] Gnuplot, <http://www.gnuplot.info>.
- [4] M. Barcellos, A. Detsch, “Avaliação de Desempenho de Protocolos para Multicast com Conhecimento de Grupo baseados em Polling”. In: *SBRC2002 - XX Simpósio Brasileiro de Redes de Computadores*, 2002, Búzios.