UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GERSON BATTISTI

Modelo de Gerenciamento para Infra-Estruturas de Medições de Desempenho em Redes de Computadores

Tese apresentada como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação

Profa. Dra. Liane Margarida Rockenbach Tarouco
Orientadora

Porto Alegre, novembro de 2007

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Battisti, Gerson

Modelo de Gerenciamento para Infra Estruturas de Medições de Desempenho em Redes de Computadores / Gerson Battisti – Porto Alegre: Programa de Pós-Graduação em Computação, 2007.

131 f.:il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2007. Orientador: Liane Margarida Rockenbach Tarouco.

1. Infra-estruturas de Medição. 2. Gerenciamento de Redes 3. Serviços *Web*. I. Tarouco, Liane Margarida Rockenbach. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. José Carlos Ferraz Hennemann
Vice-Reitor: Prof. Pedro Cezar Dutra Fonseca
Pró-Reitora de Pós-Graduação: Profa. Valquiria Linck Bassani
Diretor do Instituto de Informática: Prof. Flávio Rech Wagner
Coordenadora do PPGC: Profª Luciana Porcher Nedel
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

Quero iniciar agradecendo aquele que muitas vezes é lembrado por último ou até mesmo às vezes esquecido: o Criador. Obrigado pela saúde e pela força interior nas muitas vezes que achei que não iria conseguir nestes últimos anos.

A minha orientadora Liane Tarouco por ter aceitado a orientação e me guiado nestes anos, bem como pelos merecidos "puxões de orelha".

Um agradecimento especial a dois professores:

Luciano Paschoal Gaspary que me incentivou e auxiliou na elaboração do projeto inicial para a entrada no doutorado, disponibilizando seu tempo no SBRC do Rio de Janeiro. Valeu!!!

Lisandro Zambenedetti Granville meu co-orientador extra-oficial que me atendia pelo MSN ajudando a definir os rumos do trabalho e ficando até madrugada para escrever artigo. Incluo neste agradecimento a Clarissa que participou do artigo ao ICT. Obrigado.

Aos professores que participaram da banca de proposta de tese e da defesa de tese, obrigado pela sua disposição e pelas contribuições que enriqueceram enormemente o texto final.

Agradeço a UNIJUI pelo apoio financeiro que me possibilitou fazer o curso e aos meus colegas de trabalho, aqueles que efetivamente estavam torcendo pelo sucesso nesta árdua empreitada. Não vou citar nomes, mas eles sabem quem são.

Os desafios são mais fáceis de serem superados quando temos pessoas que torcem fervorosamente pela gente. Estou falando da família. Da minha família biológica Pai, Mãe, Mano e Mana e da minha família "contraída" Sogro, Sogra, Cunhada, Vô, Oma, Tios, Tias e Primos. Vocês são muito especiais, obrigado pela força!!!!

A esposa merece um agradecimento muito especial, pois não foi uma única vez que tivemos que nos despedir em rodoviárias embarcando para POA. Foram vários anos de estudo e neste tempo passamos por momentos bons e outros nem tanto. Muitas coisas mudaram, até nos mudamos (inclusive de cidade), a Júlia  partiu e o Ingo  chegou, choveu e fez sol e sempre recebi apoio e incentivo incondicional. Muito Obrigado do fundo do coração. Espero retribuir. Iara Te Amo.

SUMÁRIO

LISTA DE ABREVIATURAS OU SIGLAS 7

LISTA DE FIGURAS 10

LISTA DE QUADROS 12

RESUMO 13

ABSTRACT 14

1 INTRODUÇÃO 15

2 GERENCIAMENTO E MONITORAÇÃO DE REDES 18

2.1 Gerenciamento OSI 18
2.2 Gerenciamento Internet 19
2.3 Paradigmas de Gerenciamento 19
2.3.1 Classificação de Leinwand 20
2.3.2 Classificação de Martin-Flatin 21
2.3.3 Classificação de Schönwälder 22
2.3.4 Gerenciamento por Delegação 24
2.3.5 Gerenciamento *WEB*/XML 24
2.3.6 Gerenciamento usando *Web Services* 25
2.4 Monitoração de Rede 26
2.4.1 Medição não intrusiva 26
2.4.2 Medição Intrusiva 26
2.5 Métricas de Desempenho 27
2.5.1 Retardo 27
2.5.2 Variação do Retardo 28
2.5.3 Perda de Pacotes 29
2.5.4 Largura de Banda 30
2.6 Considerações Parciais 31

3 INFRA-ESTRUTURAS DE MEDIÇÃO 33

3.1 Estado da Arte 33
3.2 GFD – General Framework Design 35
3.2.1 Serviço de Ponto de Medição (*Measurement Point Service*) 35
3.2.2 Serviço de Armazenamento de Medições (*Measurement Archive Service*) 35
3.2.3 Serviço de Publicação e Descoberta (*Lookup Service*) 36
3.2.4 Serviço de Autenticação e Autorização (*Authentication Service*) 36
3.2.5 Serviço de Proteção de Recurso (*Resource Protector Service*) 37

3.2.6 Serviço de Transformação (*Transformation Service*) 37
3.2.7 Serviço de Topologia (*Topology Service*) 37
3.2.8 perfSonar - protótipo do GFD 38
3.2.9 Gerenciamento GFD – perfSONAR 39
3.3 AMP 39
3.3.1 Arquitetura NLANR-AMP 40
3.3.2 Metodologia 40
3.3.3 Gerenciamento AMP 41
3.4 BW 41
3.4.1 Arquitetura BW 41
3.4.2 Metodologia 42
3.4.3 Gerenciamento BW 42
3.5 MonALISA 43
3.5.1 Arquitetura MonALISA 43
3.5.2 Gerenciamento MonALISA 47
3.6 NIMI 47
3.6.1 Arquitetura NIMI 48
3.6.2 Gerenciamento NIMI 50
3.7 NWS 51
3.7.1 Arquitetura NWS 51
3.7.2 Metodologia 52
3.7.3 Gerenciamento NWS 52
3.8 IEPM - PingER 53
3.8.1 Arquitetura PingER 53
3.8.2 Metodologia de Monitoração 54
3.8.3 Gerenciamento PingER 55
3.9 E2E piPEs 55
3.9.1 Arquitetura piPES 55
3.9.2 Ferramentas de Testes 57
3.9.3 Gerenciamento piPEs 59
3.10 RIPE NCC - TTM 59
3.10.1 Arquitetura RIPE NCC – TTM 59
3.10.2 Metodologia 60
3.10.3 Gerenciamento RIPE 61
3.11 SCRIPTROUTE 61
3.11.1 Arquitetura Scriptroute 61
3.11.2 Gerenciamento Scriptroute 63
3.12 Surveyor 63
3.12.1 Arquitetura Surveyor 65
3.12.2 Metodologia de Medição 66
3.12.3 Gerenciamento Surveyor 67
3.13 Comparativo das infra-estruturas 67
3.13.1 Quanto aos testes 68
3.13.2 Quanto ao sincronismo 68
3.13.3 Quanto ao Gerenciamento 68

4 GERENCIAMENTO DE INFRA-ESTRUTURAS DE MEDIÇÃO 72

4.1 Requisitos de Gerenciamento 72
4.1.1 Gerenciamento de Falhas 73
4.1.2 Gerenciamento de Configuração 74

4.1.3 Gerenciamento de Contabilização 74
4.1.4 Gerenciamento de Desempenho 74
4.1.5 Gerenciamento de Segurança 75
4.2 Modelo de Gerenciamento para Infra-estruturas de Medição 75
4.2.1 Módulo *Hardware* 77
4.2.2 Módulo Sistema 77
4.2.3 Módulo Testes 78
4.2.4 Módulo Específico 80
4.2.5 Módulo Comunicação 80
4.3 Arquitetura de Gerenciamento 82
4.3.1 Componente Serviço de Infra-estrutura 83
4.3.2 Componente Serviço de Domínio 84
4.3.3 Componente Agente de Domínio 84
4.3.4 Componente Agente de Infra-estrutura 84
4.4 Considerações Parciais 84

5 ASPECTOS DE IMPLEMENTAÇÃO 86

5.1 Serviços *Web* 86
5.2 Protótipo 87
5.2.1 Tecnologias Utilizadas 87
5.2.2 Ambiente de Testes 87
5.2.3 Aspectos de Segurança 89
5.2.4 Modelo de Implementação dos Serviços 91
5.2.5 Instalação 93
5.2.6 Configuração 94
5.2.7 Inclusão dos Pontos de Medição 96
5.2.8 Avaliação do Protótipo 99

6 CONSIDERAÇÕES FINAIS 102

6.1 Trabalhos Futuros 103

REFERÊNCIAS 105

ANEXO A GERENCIAMENTO OSI - FCAPS 111

ANEXO B SNMP 114

ANEXO C ARQUITETURA DOS SERVIÇOS *WEB* 120

LISTA DE ABREVIATURAS OU SIGLAS

ACL	Access Control List
AMP	Active Measurement Project
API	Application Programming Interface
AS	Authentication Service
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation.1
ATM	Asynchronous Transfer Mode
BDC	Banco de Dados Central
BW	Bandwidth
CCITT	Comité Consultatif International Téléphonique et Télégraphique
CORBA	Common Object Request Broker Architeture
CPOC	Configuration Point of Contact
CPU	Central Processing Unit
DAC	Data Analysis Client
DANTE	Delivery of Advanced Network Technology to Europe
DDSA	Dynamic Distributed Services Architecture
DES	Data Encryption Standard
DNS	Domain Name System
DOS	Denial of Service
E2EPI	End-To-End Performance Initiative
FDDI	Fiber Distributed Data Interface
FTP	File Transfer Protocol
GFD	General Framework Design
GPS	Global Positioning System
GUI	Graphics User Interface
HTTP	Hiper Text Transfer Protocol
IAB	Internet Architecture Board
ICMP	Internet Control Message Protocol

IEPM	Internet End-To-End Performance Monitoring
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPPM	Internet Protocol Performance Metrics
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
LS	Lookup Service
LUS	Lookup Discovery Services
MA	Measurement Archive Service
MAC	Medium Access Control
MC	Measurement Client
MD5	Message Digest 5
MIB	Management Information Base
MOAT	Measurement and Operations Analysis Team
MonALISA	- Monitoring Agents in a Large Integrated Services Architecture
MP	Measurement Point Service
NAI	Network Analysis Infrastructure
NAT	Network Address Translation
NCC	Network Coordination Center
NSF	National Science Foundation
NIMI	National Internet Measurement Infrastructure
NLANR	National Laboratory for Applied Network Research
NWS	Network Weather Service
ODMA	Open Distributed Management Architecture
OSI	Open Systems Interconnect
perfSONAR	- Performance focused Service Oriented Network monitoring Architecture
PIPES	Performance Initiative Performance Environment System
PMD	Ponto de Medição de Desempenho
QOS	Quality of Service
RFC	Request For Comments
RIPE	Réseaux IP Européens
RNP	Rede Nacional de Ensino e Pesquisa
RP	Resource Protector Service
RPC	Remote Procedure Call

RTT	Round Trip Time
SDSC	San Diego Supercomputer Center
SGBD	Sistema Gerenciador de Dados
SLAC	Stanford Linear Accelerator Center
SMI	Structure of Management Information
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOA	Service Oriented Architecture
SOAP	Service Oriented Architecture Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TMN	Telecommunication Management Network
TS	Transformation Service
TTL	Time To Live
TTM	Test Traffic Measurement
UBR	Universal Business Registry
UDDI	Universal Description Discovery and Integration
UDP	User Datagram Protocol
UTC	Universal Time Coordinated
VACM	View-Based Access Control Model
VoIP	Voice over IP
W3C	World Wide *Web* Consortium
WAN	Wide Area Network
WSD	*Web* Service Description
WSDL	*Web* Services Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition

LISTA DE FIGURAS

Figura 2.1: Modelo de Gerenciamento Centralizado 20
Figura 2.2: Modelo de Gerenciamento Hierárquico 20
Figura 2.3: Modelo de Gerenciamento Distribuído 21
Figura 2.4: Classificação dos paradigmas de gerenciamento 22
Figura 2.5: Modelos de Gerenciamento 23
Figura 2.6: *Jitter* – Variação no atraso de chegada dos pacotes 29
Figura 3.1: Escopo do GFD 34
Figura 3.2: Escopo de Interação do perfSONAR 38
Figura 3.3: NAI e os tipos de monitores 40
Figura 3.4: Arquitetura AMP 40
Figura 3.5: Topologia dos *site*s participantes 42
Figura 3.6: Visão Esquemática do Mecanismo de Coleta de Dados 44
Figura 3.7: Mecanismo de Registro e Descoberta 45
Figura 3.8: Interação dos clientes com os serviços utilizando os serviços de Proxy 46
Figura 3.9: Acesso de pseudo clientes à arquitetura MonALISA 46
Figura 3.10: Fluxo de Mensagens entre os componente da Arquitetura NIMI. 48
Figura 3.11: Exemplo de uma tabela de ACL. 49
Figura 3.12: Exemplo de um tipo de bloco para comunicação NIMI 50
Figura 3.13: Arquitetura NWS 51
Figura 3.14: NWS – Experimentos de atraso e largura de banda 52
Figura 3.15: Arquitetura PingER 53
Figura 3.16: Teste entre dois PMDs, uma visão parcial da rede 55
Figura 3.17: Visão dos cenários de testes 56
Figura 3.18: Componentes de *software* da arquitetura piPEs 57
Figura 3.19: Arquitetura OWAMP 58
Figura 3.20: Arquitetura BWCTL 58
Figura 3.21: Visão geral da arquitetura RIPE-NCC 60
Figura 3.22: Componentes do Servidor *Scriptroute* 61

Figura 3.23: Espaço de nomes *Scritptroute* 62
Figura 3.24: Linha de Tempo das Infra-Estruturas de Medição de Desempenho 67
Figura 4.1: Modelo de Gerenciamento de Infra-estruturas de Medição de Desempenho 76
Figura 4.2: Interação entre as entidades de gerenciamento 82
Figura 4.3: Entidades da Arquitetura 83
Figura 5.1: Cidades Utilizadas para Testes no Protótipo 88
Figura 5.2: Ambiente de testes do protótipo 89
Figura 5.3: Recebimento da chave pública assimétrica 89
Figura 5.4: Envio da chave simétrica ao gerente de domínio superior 90
Figura 5.5: Exemplo de solicitação de serviços com mensagens criptografadas com chaves simétricas. 90
Figura 5.6: Exemplo de Criptografia utilizando o OpenSSL e PHP. 91
Figura 5.7: Protótipo - Informações locais do Gerente de Domínio 93
Figura 5.8: Tela de Instalação do Protótipo 94
Figura 5.9: Tela de Configuração. 95
Figura 5.10: Continuação da Tela de Configurações. 96
Figura 5.11: Pedido de inclusão de um novo ponto de medição 97
Figura 5.12: Ativar um ponto de medição 98
Figura 5.13: Hierarquia dos pontos de medição no ambiente de testes 98
Figura 5.14: Atualização do arquivo de configuração do *owamp* 99
Figura 5.15: Falha no equipamento *ijui1* 99
Figura 5.16: Falha no *ssh* no equipamento *3passos* 100
Figura 5.17: Detecção da falha no equipamento *panambi* pelo gerente superior 101
Figura 5.18: Verificação da falha pelo gerente local (*panambi*) 101

LISTA DE QUADROS

Quadro 3.1: Relação das infra-estruturas e suas principais características de gerenciamento. 71
Quadro 4.1: Lista de serviços definidos no módulo *Hardware* 77
Quadro 4.2: Lista de serviços definidos no módulo *Sistema* 78
Quadro 4.3: Lista de serviços definidos no módulo *Testes* 79
Quadro 4.4: Lista de serviços definidos no módulo *Específico* 80
Quadro 4.5: Lista de serviços definidos no módulo *Comunicação* 81
Quadro C.6.1: Tipos de registros previstos na especificação UDDI v.3 130

RESUMO

A avaliação do comportamento das redes de computadores através de métricas de desempenho é útil para avaliação de protocolos, para o aperfeiçoamento de aplicações, na escolha de conteúdo, entre outros. A obtenção destas métricas, porém, continua sendo uma tarefa complexa porque na maioria das vezes não se tem instrumentos adequados para tal. O desempenho de uma rede é fácil de ser obtido quando as conexões envolvidas estão limitadas a uma Intranet, pois os administradores têm acesso irrestrito a todos os equipamentos envolvidos. Por outro lado, quando os equipamentos pertencerem a domínios administrativos diferentes, a avaliação de desempenho passa a ser uma tarefa difícil de ser efetuada.

As infra-estruturas de medição de desempenho foram propostas como uma forma de dotar as redes de computadores com instrumentos que facilitem a mensuração do desempenho. Uma infra-estrutura de medição de desempenho é um conjunto de equipamentos, espalhados por diversas redes, designados para interagirem uns com os outros quantificando o desempenho dos enlaces entre eles.

As vantagens da utilização de infra-estruturas de medição de desempenho são claras, porém a sua utilização em larga escala ainda é problemática. Isso ocorre por problemas de escalabilidade e da inabilidade de gerenciamento dos equipamentos que compõem a infra-estrutura, presentes em diferentes domínios administrativos.

Este trabalho avaliou um conjunto de infra-estruturas de medição de desempenho e concluiu que o tratamento dado ao gerenciamento de tais infra-estruturas é variado. De forma resumida, elas não possuem um conjunto de funções de gerenciamento padronizado. Com base nesta avaliação, o trabalho apresenta um modelo de gerenciamento para infra-estruturas de medição de desempenho. O modelo proposto mantém a independência administrativa dos pontos presentes na infra-estrutura, mas permite a interação entre os mesmos por meio de funções de gerenciamento específicas.

Um protótipo foi implementado e implantado em um ambiente real para validação do modelo de gerenciamento proposto. Durante o período de avaliação do protótipo foi possível confirmar a importância do gerenciamento dos componentes das infra-estruturas de medição. Os problemas ocorridos durante esse período foram rapidamente detectados e solucionados.

A experiência de uso do protótipo possibilitou observar os benefícios que podem ser obtidos com o gerenciamento de infra-estruturas de medição de desempenho.

Palavras-Chave: Infra-estruturas de Medição, Gerenciamento de Redes, Serviços *Web*

Management Model for Measurement Infrastructure in Computer Networks

ABSTRACT

The evaluation of computer network behavior through performance metrics is useful in protocol evaluation, application improvement and content choice, among others. The process of obtaining those metrics continues to be a complex task because most of the time there are no tools available for such. The performance of a network is easy to be obtained when the involved connections are limited to an Intranet. In this case, administrators have unrestricted access to all the involved equipments. On the other hand, when the equipments belong to different administrative domains, the performance evaluation becomes a very difficult task.

The measurement infrastructures appear as a form of endowing the computer networks with instruments that facilitate measuring performance. A measurement infrastructure is a group of equipments, spread through several networks, designated to interact with each other quantifying the performance of the connections among them.

The advantages of the use of measurement infrastructures are clear, although its use in wide scale is still problematic. That happens mainly due to scalability problems and to inability of equipment management belonging to different administrative domains.

This work evaluated a set of measurement infrastructures and concluded that the treatment given to the management of such infrastructures is varied.

In summary, there is not a set of standardized management functions. Based on this evaluation, this work presents a management model for measurement infrastructures. The proposed model maintains the administrative independence of the present points in the infrastructure but, also, allows the interaction among them by means of specific management functions.

A prototype was implemented and deployed in a real environment for validation of the proposed management model. During the evaluation prototype period it was possible to confirm the importance of the management of the components of the measurement infrastructures. The experience of use of the prototype made possible to observe the benefits that can be obtained with the management of measurement infrastructures.

Keywords: Measurement Infrastructure, Management Network, *Web* Services

1 INTRODUÇÃO

A cada ano, novas aplicações e novos usuários impulsionam o crescimento da Internet tanto em escala como em complexidade. A necessidade de compreender o comportamento dessa rede, diante desse crescimento, impulsiona também a evolução das técnicas de monitoração de rede. A forma mais utilizada para mapear o comportamento da rede é através da monitoração de desempenho da mesma.

A monitoração de desempenho possibilita detectar problemas de *hardware* nos equipamentos de rede, falha nos enlaces, problemas de configuração de equipamentos, gargalos de tráfego, ataques de negação de serviço (DoS - *Denial of Services*) entre outros (LAI, 2000). Além disso, a monitoração de desempenho é útil porque permite ao usuário checar se a banda disponível está de acordo com o que ele está pagando, verificar se a rede que ele acessa está suficientemente dimensionada e permite aos provedores detectar enlaces congestionados e/ou subutilizados, o que possibilita planejar de forma eficiente o futuro de uma rede (DOVROLIS, 2004).

O desempenho da rede é mais fácil de ser monitorado quando as conexões envolvidas estão limitadas a uma Intranet, pois os administradores têm acesso irrestrito a todos os equipamentos envolvidos. Por outro lado, quando equipamentos pertencentes a domínios administrativos diferentes influenciarem no desempenho de um fluxo, então a detecção de degradações se torna uma tarefa difícil de ser efetuada. Essa dificuldade ocorre porque os provedores de acesso normalmente não fornecem dados sobre a carga e desempenho das suas redes: ou porque não possuem ferramentas adequadas para isso, ou por representarem eventual divulgação de dados confidenciais.

A solução intuitiva é dotar os provedores de acesso de *hardwares* e/ou *softwares* especiais que possibilite a qualquer usuário visualizar o desempenho fim-a-fim da rede (equipamento de origem dos dados até o equipamento de destino), incluindo os pontos intermediários. Esta solução, chamada de infra-estrutura de medição, consiste em um conjunto de equipamentos com *softwares* dedicados para avaliar métricas que representam o desempenho de segmentos da rede.

Muitos são os motivos para a utilização de uma infra-estrutura de medição, tais como (MURRAY, 2001): estabelecimento de um padrão de comportamento da rede, histórico da rede, detecção de anomalias. Em resumo, uma infra-estrutura de medição pode facilitar a rápida detecção e diagnóstico de vários problemas produzindo benefícios para os usuários e operadores de rede.

Apesar da existência de pontos favoráveis para a utilização de infra-estruturas de medição elas ainda estão restritas a infra-estruturas de testes ou a pequenos grupos com interesses comuns. Nestes casos, é comum a existência de uma instituição centralizadora que organiza e gerencia o funcionamento da infra-estrutura, incluindo os equipamentos dos demais participantes.

As tarefas de administração, controle e configuração dessas infra-estruturas de medição são estáticas, ou seja, durante a instalação o administrador opta por algumas definições que permanecem até serem alteradas manualmente. O mesmo acontece com os testes: aqueles que são definidos para serem executados periodicamente têm seus parâmetros definidos manualmente.

Normalmente, as infra-estruturas não se preocupam com as condições do equipamento em que estão instaladas, ou seja, não monitoram o *hardware* e *software* que compõe um equipamento participante da infra-estrutura de medição. A falta de monitoração de características como o uso de CPU (*Central Processing Unit*), quantidade de disco disponível, estado da execução dos programas de teste (normal, *loop*, falhas) entre outras, permite que a ocorrência de erros comprometa o bom funcionamento do equipamento e provavelmente os resultados dos testes. Por exemplo, supondo que o programa *traceroute* seja utilizando por determinada infra-estrutura e, por algum motivo qualquer, ele seja removido do sistema, a falha somente será detectada quando o programa for utilizado novamente para um teste, o que certamente acarretará perda de informação.

Em uma infra-estrutura de medição em larga escala as mudanças nas ferramentas instaladas ou a instalação de novas ferramentas são freqüentes. Em ambos os casos, um longo tempo é necessário para que todos os integrantes estejam corretamente configurados e com as mesmas versões de *software*. Em algumas infra-estruturas de medição, por exemplo, esse processo de atualização é feito por meio do envio de e-mail para o administrador de cada ponto integrante, solicitando que este atualize as ferramentas.

Ainda considerando uma infra-estrutura de medição em larga escala, ela inevitavelmente estará presente em múltiplos domínios administrativos, ou seja, os componentes da infra-estrutura são heterogêneos e administrados por entidades diferentes. Este fato envolve diretamente questões de segurança entre os administradores. A solução onde o operador local fornece acesso a determinado operador remoto é dificilmente aceitável.

O tratamento dado pelas infra-estruturas de medição atuais para o gerenciamento é variado. Algumas propostas simplesmente ignoram a necessidade de gerenciamento, apenas definem um conjunto mínimo de *hardware* e *software* para compor a infra-estrutura. As que mais englobam gerenciamento prevêem um pequeno conjunto de funções, normalmente utilizadas para controle de usuários e para o controle das ferramentas de testes. Na prática não existe um conjunto de funções de gerenciamento específicas para uso em infra-estruturas de medição.

A utilização efetiva em larga escala das infra-estruturas de medição é dependente de um conjunto de funções de gerenciamento que garantam seu funcionamento ao longo do tempo, de forma organizada, segura e atualizada. É necessário garantir que uma infra-estrutura de medição não tenha seus serviços comprometidos com facilidade, não atendendo aos interesses dos usuários e operadores de rede.

O objetivo principal deste trabalho é propor um modelo de gerenciamento para infra-estruturas de medição de desempenho em redes de computadores. Este modelo apresenta algumas funcionalidades de gerenciamento de redes, disponibilizadas de forma segura, que facilitam a manutenção e uso das infra-estruturas de medição. Outro objetivo deste trabalho é apresentar as infra-estruturas públicas de medição, suas características e funcionalidades.

A organização do texto deste trabalho é o que segue:

No capítulo 2 são apresentados os conceitos básicos de gerenciamento de redes e os paradigmas de gerenciamento atuais. Na seqüência do capítulo estão relacionadas as métricas utilizadas para quantificar o desempenho em uma rede de computadores. Estas métricas são utilizadas pelas ferramentas de testes que compõem as infra-estruturas de medição. O capítulo 3 apresenta o estado da arte nas propostas de infra-estruturas de medição de desempenho e um conjunto de infra-estruturas de medição públicas, sua metodologia, funcionamento e características de gerenciamento. São consideradas públicas aquelas que disponibilizam informações sobre seu funcionamento ou resultados das medições, por artigos ou páginas *Web*. No capítulo 4 inicialmente são apresentados os requisitos de gerenciamento que devem ser observados pelo modelo. Na seqüência é apresentado o modelo de gerenciamento para infra-estruturas de medição, seus componentes, os serviços que devem ser implementados e o funcionamento do sistema de gerenciamento. O capítulo 5 apresenta os aspectos de implementação do protótipo do modelo. O capítulo inicia com as tecnologias utilizadas na implementação até o funcionamento e utilização do protótipo. No final do capítulo são relacionados alguns casos que demonstram que a utilização do sistema de gerenciamento facilita a manutenção das infra-estruturas de medição. As conclusões e considerações finais do trabalho são apresentadas no capítulo 6.

2 GERENCIAMENTO E MONITORAÇÃO DE REDES

Neste capítulo são apresentados os conceitos iniciais sobre gerenciamento de redes, sobre os paradigmas de gerenciamento e sobre as métricas para avaliação de desempenho. Estes conceitos são importantes para um melhor entendimento do funcionamento das infra-estruturas de medição que serão apresentadas no próximo capítulo. Além disso, a apresentação das áreas funcionais do gerenciamento de redes fornece a base para a compreensão das funções de gerenciamento previstas no modelo que será apresentado neste trabalho.

Concebidas inicialmente como um meio de compartilhar dispositivos periféricos mais caros como impressoras e modems de alta velocidade, as redes de computadores passaram a fazer parte do cotidiano dos usuários como uma ferramenta que oferece um conjunto de recursos imprescindíveis. A manutenção do funcionamento deste ambiente de produção de forma suave, quase imperceptível aos usuários é, em última análise, a função do gerenciamento de redes. Existem várias definições na literatura para gerenciamento de redes (BRISA, 1993), (PRAS, 1995), (LEINWAND, 1996). Uma definição detalhada é feita por SAYDAM apud KUROSE (2004, p.591), que diz:

> Gerenciamento de rede inclui a disponibilização, a integração e a coordenação de elementos de *hardware*, *software* e humanos, para monitorar, testar, consultar, configurar, analisar, avaliar e controlar os recursos da rede, e de elementos, para satisfazer às exigências operacionais, de desempenho e de qualidade de serviço em tempo real a um custo razoável.

Neste capítulo serão abordados os modelos de gerenciamento OSI (*Open Systems Interconnect*), apresentando as suas áreas funcionais e o modelo de gerenciamento Internet, baseado no protocolo SNMP (*Simple Network Management Protocol*).

2.1 Gerenciamento OSI

O Modelo de gerenciamento OSI teve seu início em 1989 com a publicação do *OSI Management Framework* e em 1992 com a publicação do *OSI System Management Overview*. Estas duas publicações, feitas em conjunto pelos organismos ISO e CCITT (Comité Consultatif International Téléphonique et Télégraphique), atual ITU-T (*International Telecommunication Union – Telecommunication Standardization Sector*), são os marcos iniciais para o padrão de gerenciamento OSI.

O Modelo de Referência OSI apesar de definir os protocolos de gerenciamento acabou não sendo utilizado em redes de computadores, principalmente as baseadas em TCP/IP (*Transmission Control Protocol/Internet Protocol*). Porém, as definições das suas áreas funcionais continuam válidas. As subdivisões previstas no modelo funcional, em cinco áreas, definem o escopo de um sistema de gerenciamento de redes e são

normalmente aceitas independente do modelo de gerenciamento (STALLINGS, 1993). As áreas são conhecidas pela sigla “FCAPS”, que é um acrônimo, significando:

Fault Mangement (Gerenciamento de Falhas)

Configuration Management (Gerenciamento de Configuração)

Accounting Management (Gerenciamento de Contabilização)

Performance Management (Gerenciamento de Desempenho)

Security Management (Gerenciamento de Segurança)

No Anexo A são detalhadas as funções de cada uma das áreas FCAPS prevista no modelo de gerenciamento OSI. Estas áreas funcionais serão utilizadas neste trabalho para apresentar os requisitos de gerenciamento do modelo.

2.2 Gerenciamento Internet

A necessidade de estruturação e padronização do gerenciamento na década de 80, devido ao crescimento no número de dispositivos e a complexidade das redes, incentivaram o surgimento de novas propostas de gerenciamento (TANENBAUM, 2003). A primeira proposta real foi lançada em 1988 com a definição do protocolo SNMP. Devido à sua concepção simples, o protocolo SNMP tornou-se rapidamente a solução de gerenciamento mais utilizada, sendo suportada por diferentes fabricantes. O protocolo SNMP está detalhado no Anexo-B.

2.3 Paradigmas de Gerenciamento

O modelo inicial de gerenciamento de rede (SNMP) era baseado no modelo gerente-agente. Neste modelo o gerente concentra as funções de controle e monitoração e é o responsável pelo acesso aos diversos agentes da rede. Os agentes são simples fornecedores das variáveis (da MIB), enquanto o gerente, através do mecanismo de *polling*, monitora a rede efetuando operações de controle, quando necessário. O objetivo era simplificar o agente permitindo um desenvolvimento rápido e exigindo poucos recursos dos equipamentos. Além disso, não era definido nenhum mecanismo para a comunicação entre gerentes. Esta abordagem sobrecarregava o gerente com todas as funções, gerando grande tráfego na rede e degradando o tempo de resposta na detecção de problemas.

Com o crescimento das redes de computadores, em tamanho e complexidade, os sistemas de gerenciamento baseados em um único gerente, responsável por todas as funções de gerenciamento, tornam-se inapropriados devido ao volume das informações que devem ser tratadas e que podem pertencer a localizações geograficamente distantes do gerente.

Desta forma, evidencia-se a necessidade da distribuição do gerenciamento na rede, através da divisão das responsabilidades gerenciais entre gerentes locais que controlam domínios distintos e da expansão das funcionalidades dos agentes.

Naturalmente, com o surgimento de novas tecnologias, várias abordagens para o gerenciamento surgiram, produzindo um vasto conjunto de paradigmas. Vários autores tentaram classificar estes paradigmas, mas ainda não existe um consenso entre os administradores sobre qual classificação utilizar. Na seqüência são apresentadas três

classificações propostas para os paradigmas de gerenciamento. As propostas são de Leinwand (1996), Martin-Flatin (1999) e Schönwälder (2000).

2.3.1 Classificação de Leinwand

Segundo Leinwand (1996) as arquiteturas de gerenciamento podem ser divididas em: Centralizada, Hierárquica e Distribuída.

Na **arquitetura centralizada** apenas um gerente é o responsável pelos procedimentos de gerenciamento em todos os equipamentos. Esse modelo usa um banco de dados de gerenciamento centralizado. Para tolerância a falhas, o banco de dados deve ser replicado para outro sistema.

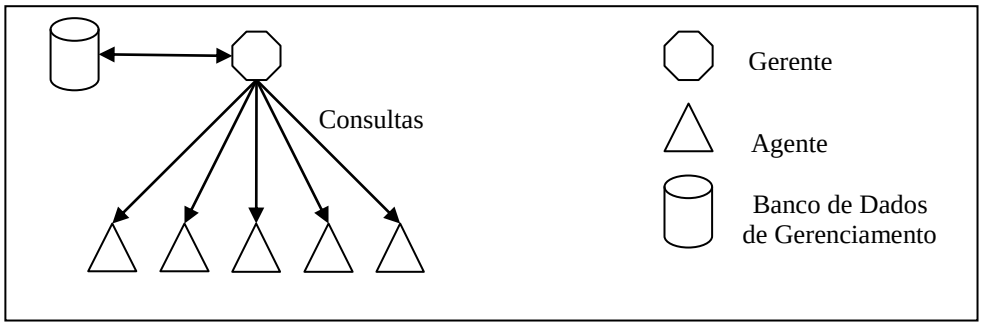

Figura 2.1: Modelo de Gerenciamento Centralizado

A vantagem deste modelo é possuir um único local para visualização das informações da rede, como alertas e eventos. A questão de segurança também é uma vantagem, pois a tarefa de proteger o gerente é simplificada quando é único. Como desvantagem tem-se a dependência de um único ponto: em caso de falha, o sistema fica inoperante. Outra desvantagem significante é o fato de concentrar todo o tráfego gerado para o gerenciamento no enlace de acesso ao elemento central. Este é um dos motivos do modelo não ser escalável.

Já uma **arquitetura hierárquica** usa múltiplos gerentes, onde um atua como servidor central e os outros como clientes. As funções de gerenciamento são divididas entre o servidor e os clientes. Uma arquitetura hierárquica possui as seguintes características: não dependência de um ponto central; distribuição das tarefas de gerenciamento; monitoração distribuída pela rede; informações de gerenciamento armazenadas de forma centralizada.

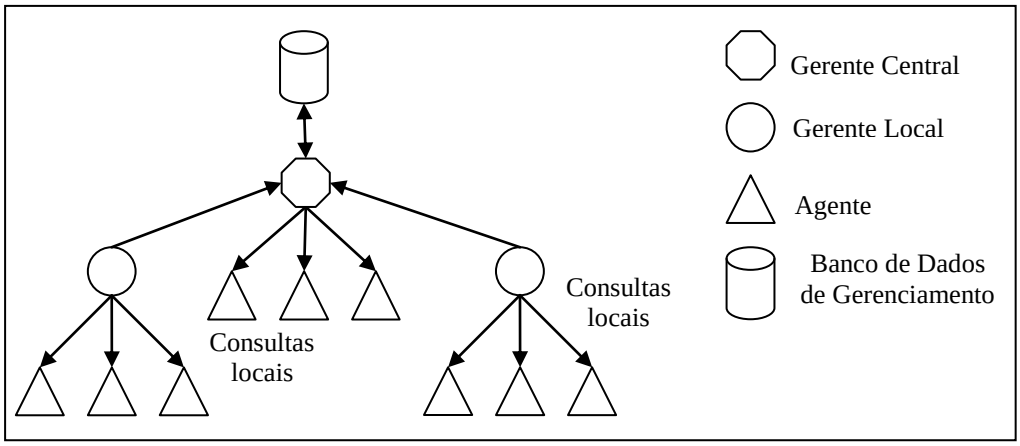

Figura 2.2: Modelo de Gerenciamento Hierárquico

Nesta arquitetura os clientes não possuem um banco de dados local, eles utilizam o modelo cliente/servidor para enviar os dados para o banco de dados central.

A **arquitetura distribuída** combina as arquiteturas centralizada e hierárquica considerando cada plataforma um ponto de gerenciamento. Cada ponto individual possui um banco de dados completo dos dispositivos gerenciado na rede e, além de efetuar várias tarefas devem reportar os resultados ao sistema central. As vantagens do modelo são: único local para todas as informações da rede (alertas e eventos); único local para acesso a todas as aplicações de gerenciamento; não dependência de um ponto central; distribuição de tarefas de gerenciamento; monitoração distribuída pela rede.

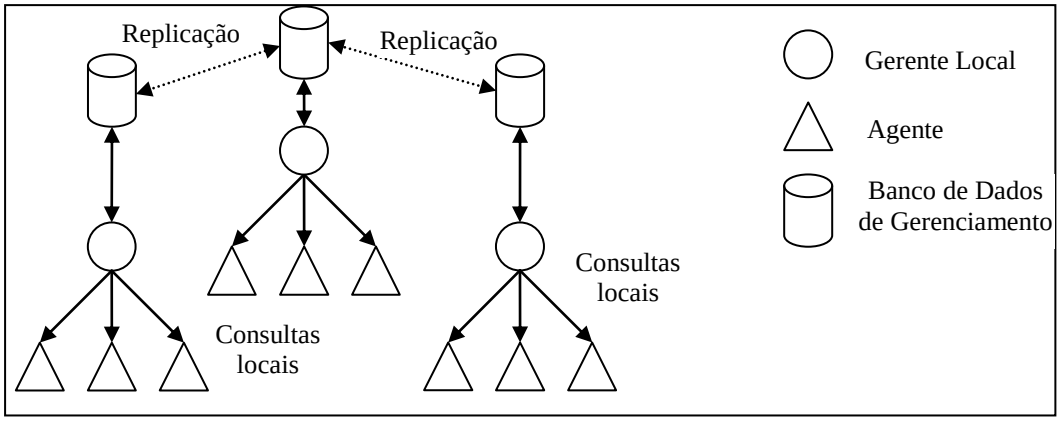

Figura 2.3: Modelo de Gerenciamento Distribuído

A replicação dos bancos de dados entre os pontos é a principal desvantagem, pois exige um consumo extra de recursos da rede para manter o sincronismo das bases de dados de gerenciamento.

2.3.2 Classificação de Martin-Flatin

A classificação proposta por MARTIN-FLATIN (1999) obedece a três princípios:

- Separa os paradigmas entre centralizados e distribuídos
- Isola as diferenças entre os paradigmas novos e tradicionais
- Distingue os paradigmas baseados em delegação vertical e horizontal

Os *paradigmas tradicionais* são constituídos pelos paradigmas centralizados e pelos paradigmas hierárquicos fracamente distribuídos. Nestes paradigmas estão incluídas as arquiteturas SNMP, OSI, e TMN (*Telecommunication Management Network*). Estes paradigmas caracterizam-se pelo fato do processamento realizado para gerenciar a rede ficar concentrado em um grupo pequeno de estações de gerenciamento e os agentes serem meros coletores de informação.

A arquitetura SNMP foi apresentada na seção anterior. A arquitetura OSI (não é utilizada para gerenciamento Internet) e a arquitetura TMN (destinada às telecomunicações) conseqüentemente não serão apresentadas por estarem fora do escopo do trabalho. Maiores informações sobre estas arquiteturas são encontradas em HEMMEN (2000).

A figura 2.4 sintetiza a classificação dos paradigmas de gerenciamento proposta por Martin-Flatin.

	Paradigmas Centralizados	**Paradigmas Hierárquicos**	**Paradigmas Cooperativos**
Não Distribuído	SNMPv1		
Fracamente Distribuído		RMON, RMONv2, SNMPv2, SNMPv3, CMIP(OSI) e TMN	
Fortemente Distribuído		Código Móvel, Objetos Distribuídos.	Agentes Inteligentes

Figura 2.4: Classificação dos paradigmas de gerenciamento

Nos *paradigmas hierárquicos fortemente distribuídos* encontram-se os baseados em *Código Móvel* e *Objetos Distribuídos*. Os paradigmas baseados em *código móvel* visam prover flexibilidade transferindo dinamicamente programas para os agentes onde serão executados. Duas abordagens utilizam estes conceitos, *redes ativas* e *gerenciamento por delegação*. As *redes ativas* permitem que os usuários injetem programas nos nodos da rede que podem manipular o fluxo de dados do usuário que trafega pelo nodo. A proposta do *gerenciamento por delegação* é transferir funcionalidades para agentes remotos ou gerentes delegando a execução dinâmica de tarefas. O gerenciamento por delegação será apresentado com mais detalhes nas próximas seções.

Nos paradigmas de gerenciamento baseado em *objetos distribuídos* estão as tecnologias CORBA (*Common Object Request Broker Architeture*) e ODMA (*Open Distributed Management Architecture*). Essas tecnologias foram uma iniciativa da indústria para os problemas de interoperabilidade da orientação a objetos que acabou sendo utilizada para o gerenciamento de rede.

Os *paradigmas cooperativos fortemente distribuídos* são baseados em *agentes inteligentes*. Oriundos da Inteligência Artificial distribuída (multi-agentes), os agentes deixam de serem meros coletores de dados e passam a efetuar processamento sobre os dados coletados. Essa abordagem permite dividir as tarefas de gerenciamento entre um grupo de entidades autônomas, posicionadas o mais próximo possível do equipamento gerenciado, reduzindo o volume de informações de gerenciamento que precisam trafegar pela rede.

2.3.3 Classificação de Schönwälder

Para classificar os paradigmas de gerenciamento, SCHÖNWÄLDER (2000) estabelece uma relação entre o número de agentes e gerentes. Os autores definem quatro classes de sistemas ou formas de gerenciamento de rede: *gerenciamento centralizado* quando o sistema possuir apenas um gerente; *gerenciamento fracamente distribuído* quando o sistema possuir dois ou mais gerentes, porém com um número bem inferior ao número total de componentes (soma de agentes e gerentes); *gerenciamento fortemente distribuído* quando existe no sistema muito mais do que dois gerentes, mas em número inferior ao total de componentes; *gerenciamento cooperativo* quando o número de gerentes é aproximado ao número total de componentes.

A figura 2.5 apresenta a classificação com as relações entre agentes, gerentes intermediários e gerentes.

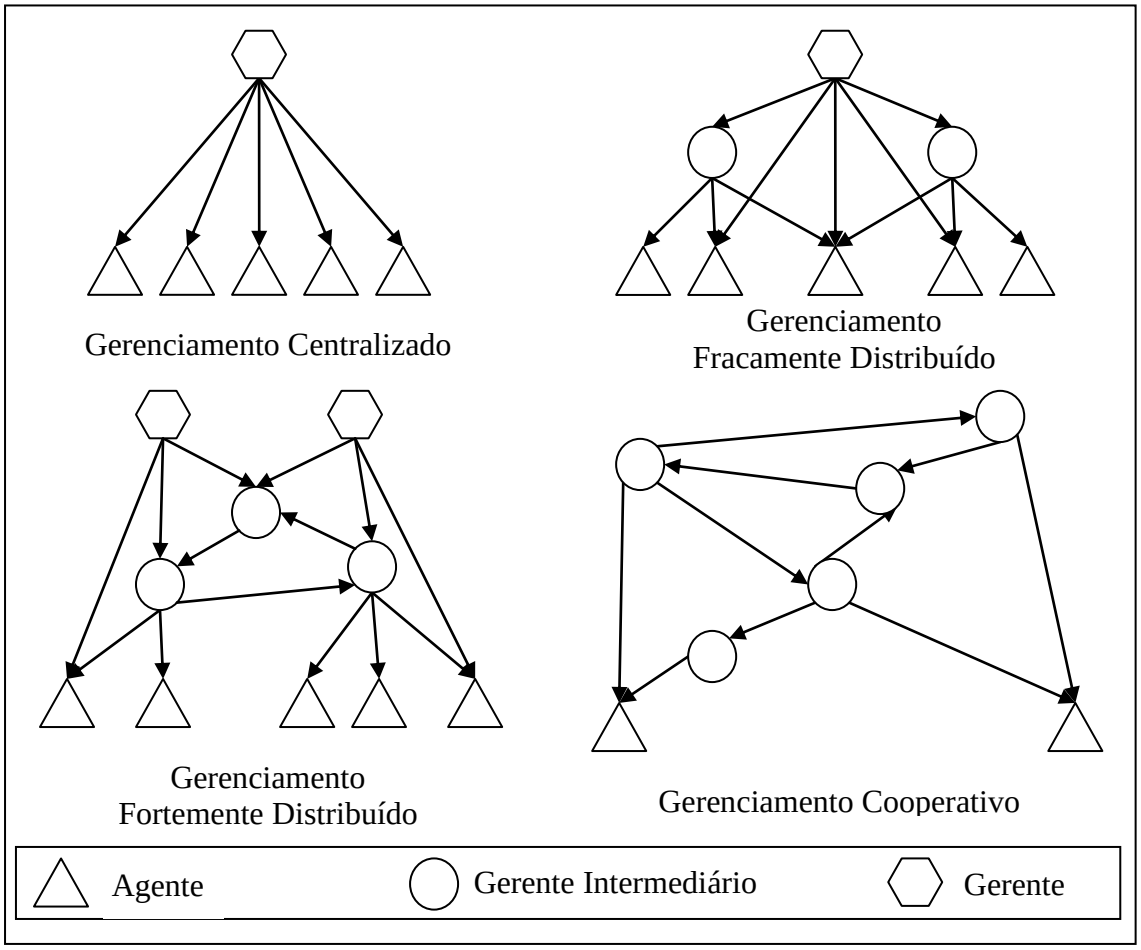

Figura 2.5: Modelos de Gerenciamento

Ainda, segundo SCHÖNWÄLDER (2000), visualizando as classes de gerenciamento percebe-se que à medida que se direciona do gerenciamento centralizado para o cooperativo, a simplicidade do gerenciamento vai diminuindo e, em contrapartida, a *escalabilidade* e *flexibilidade* vai aumentando.

- Escalabilidade: A escalabilidade do sistema está em três razões básicas. Primeira, a carga computacional do gerente é reduzida pela delegação das funções de gerenciamento. Os gerentes intermediários monitoram um subconjunto de agentes e repassam aos gerentes informações agregadas. Segunda, a redução da carga computacional na rede. A localização dos gerentes intermediários, mais próximos dos agentes, permite enviar dados processados e compactados para os gerentes. Além disso, com delegação e mobilidade de processos, o código é movido pela rede e não os dados. Terceira, a redução da necessidade de armazenamento nos nodos. Com a delegação dinâmica das funções de gerenciamento ou mobilidade de processos, apenas os procedimentos ativos necessitam ser armazenados.
- Flexibilidade: como as tarefas de gerenciamento podem ser atribuídas dinamicamente ou os processos podem ser movidos, o sistema de gerenciamento pode ser mais flexível. As tarefas dos agentes e os gerentes

intermediários podem ser alterados livremente e novas tarefas podem ser definidas rapidamente.

2.3.4 Gerenciamento por Delegação

Uma aplicação típica de gerenciamento distribuído de rede consiste em gerentes, agentes e entidades capazes de realizar tarefas de ambos, também conhecidas como gerentes intermediários. Tradicionalmente, o gerente é caracterizado por disparar procedimentos de gerenciamento e o agente, por realizar tarefas bastante simples como coletar dados e prover aos gerentes acesso a eles. Atualmente, entretanto, a distinção entre as tarefas associadas a gerentes e agentes começa a ficar menos nítida, uma vez que os agentes de gerenciamento têm assumido um conjunto crescente de responsabilidades.

O gerenciamento por delegação foi definido por GOLDSZMIDT (1995) tendo como idéia básica a possibilidade do gerente delegar para o agente a execução de programas de gerenciamento. Os agentes podem monitorar e controlar efetivamente os objetos locais sem a necessidade de envolvimento dos gerentes remotos. As entidades envolvidas no gerenciamento por delegação são:

- Servidor elástico: é um processo *multithread*, cujo código do programa e estado do processo podem ser modificados, estendidos e/ou contratados durante sua execução. O ambiente de execução de um processo elástico suporta: a tradução e a ligação dinâmica de agentes delegados; o controle remoto dos agentes; e a comunicação entre agentes.
- Agente delegado: pode ser transferido dinamicamente para um servidor elástico remoto em execução. O código do agente é instanciado como uma *thread* dentro do espaço de endereçamento do processo elástico. Os agentes delegados podem ser implementados em qualquer linguagem.
- Protocolo de delegação: um protocolo de delegação é utilizado para transferir um agente para um servidor elástico remoto e controlar sua execução.

Uma das propostas do IETF para o gerenciamento por delegação é a MIB Script. Algumas propostas de uso da MIB Script para o gerenciamento distribuído são apresentadas por SHÖNWÄLDER (2000).

2.3.5 Gerenciamento *WEB*/XML

O gerenciamento *Web* inicialmente era caracterizado pela existência de servidores *Web* embarcados nos dispositivos de rede, o que possibilitava o acesso as informações via navegador. Essa proposta de gerenciamento oferecia algumas vantagens aos administradores como acesso amigável, interface padronizada e acesso remoto, porém este modelo possui sérias limitações de escalabilidade. O uso desta tecnologia para configurar centenas de equipamentos via navegador é inviável, portanto o uso da *Web* para gerenciamento de rede é uma alternativa de acesso as informações dos dispositivos, mas está bem distante de prover uma solução que possa substituir uma arquitetura de gerenciamento como SNMP.

O gerenciamento de rede usando XML, por sua vez utiliza a linguagem de marcação XML para codificar os dados a serem transferidos, provendo um excelente mecanismo de transmissão de dados complexos usados no gerenciamento de rede (JUNIPER, 2005).

O uso de XML no gerenciamento de rede difere do gerenciamento *Web* porque além de fornecer uma nova forma de acesso as informações também permite definir, criar, excluir estruturas de dados dinamicamente. As principais vantagens do uso do gerenciamento XML são (JUNIPER, 2005) e (STRAU, 2003):

- Os dados de gerenciamento podem ser representados como um documento XML, que possui formato texto, facilitando o processamento e até mesmo a direta compreensão humana;
- Os protocolos utilizados para transferir os dados entre os equipamentos são amplamente conhecidos (TCP/HTTP);
- As operações de gerenciamento de alto nível podem ser definidas através de uma linguagem de descrição (WSDL).
- Possibilidade de extensão e compatibilidade com aplicações já definidas. Os clientes e servidores podem ajustar entre si qual versão vão usar para transferência das informações.

A grande interoperabilidade com os agentes SNMP tradicionais por meio dos gateways XML/SNMP impulsiona a adoção do gerenciamento XML. Para atender bem as questões de escalabilidade e eficiência, consideradas problemas no modelo SNMP, o gerenciamento XML necessita de outras tecnologias de acesso a *Web*. A adoção do XML associado aos serviços *web* atende bem essas necessidades.

2.3.6 Gerenciamento usando *Web Services*

Os *Web Services* ou serviços *web* é uma arquitetura baseada em XML que possui mecanismos de processamento distribuído para pesquisa e publicação dos serviços de gerenciamento. A arquitetura distribuída dos serviços *web* aliada a definição de interfaces complexas e flexíveis viabiliza a sua aplicação na comunicação entre os gerentes em diferentes domínios.

As vantagens do uso dos serviços *web* no gerenciamento de rede são (MANNAERT, 2005):

- Os serviços *web* não definem a arquitetura de gerenciamento, mas oferece os mecanismos para sua implementação;
- Com os serviços *web* os recursos da rede podem ser gerenciados localmente ou remotamente por meio de mensagens bem definidas;
- O modelo convencional de gerenciamento (gerente-gerente) pode ser facilmente mapeado para o uso com serviços *web*;
- Possibilidade de uso do protocolo HTTP(S) para transporte dos dados permite passar pela maioria das redes (firewalls);
- Outra grande vantagem dos serviços *web* é a reutilização de código, pois a migração ou adaptação das plataformas existentes é facilitada;

Uma das grandes críticas ao modelo de gerenciamento usado serviços *web* e XML está no desempenho da arquitetura. Diversos estudos, como por exemplo, PRAS (2004) demonstra que para pequenas quantidades de dados a arquitetura SNMP é mais eficiente que os serviços *web*, porém quanto maior a quantidade de informações a ser transferida, mais próximo um do outro fica o desempenho.

2.4 Monitoração de Rede

A monitoração é indissociável do gerenciamento de rede, ela é a responsável pela coleta das informações que serão analisadas nos sistemas de gerenciamento. A monitoração pode ser dividida em três partes (LEINWAND, 1996):

- **Informações a serem monitoradas**: como definir uma informação a ser monitorada e como obter essa informação do recurso gerenciado.
- **Projeto dos mecanismos de monitoração**: Qual a melhor forma de obter as informações dos recursos gerenciados.
- **Aplicação da informação monitorada**: definir como a informação vai ser usada nas várias áreas funcionais de gerenciamento.

Basicamente, a coleta das informações para o gerenciamento de desempenho de uma rede de computadores pode ser feita de duas maneiras: de forma passiva (ou não intrusiva) e de forma ativa (ou intrusiva).

2.4.1 Medição não intrusiva

O método de medição não intrusiva mede o comportamento da rede através da observação da taxa de chegada de pacotes em um determinado sistema. A monitoração passiva não utiliza nenhum recurso da rede e não introduz nenhuma perturbação na mesma.

Os coletores passivos devem obrigatoriamente derivar o sinal físico para evitar a introdução de latência na transmissão de dados.

Simplesmente monitorar um dos lados de uma troca de dados arbitrária não acrescenta muita informação de valor e pode resultar em diversas interpretações dos resultados. Para obter informações mais precisas deve-se conhecer a origem dos dados que estão sendo observados.

2.4.2 Medição Intrusiva

Medição intrusiva refere-se à injeção controlada de pacotes na rede e a subseqüente coleta destes pacotes. A troca de pacotes Ping (*ICMP (Internet Control Message Protocol) echo request* e *ICMP echo reply*) é um exemplo deste método de medição. Enviando seqüências de pacotes através deste comando em intervalos regulares, a estação de medição pode medir parâmetros, tais como: alcançabilidade, tempo de resposta RTT (*Round Trip Time*) de transmissão para uma estação remota e expectativa de perda de pacotes no caminho de ida e volta.

O problema da monitoração intrusiva é que os pacotes de teste podem saturar enlaces com pouca largura de banda, além de atrapalhar outras aplicações que estão no mesmo *host* ou que utilizam a mesma rede.

A vantagem da monitoração intrusiva é a facilidade e rapidez com que os resultados podem ser obtidos. Considerando uma técnica que requer *n* pacotes transmitidos para calcular a largura de banda, usando monitoração ativa, basta gerar os pacotes necessários. No caso de monitoração passiva é preciso aguardar que os pacotes sejam gerados por alguma fonte da rede.

A execução de um determinado número de medições considerando o comportamento da fila dentro dos comutadores ou roteadores, combinando estas

medidas com a medida de perda de pacotes e o *jitter*, pode-se fazer algumas inferências sobre a largura de banda disponível entre dois pontos e o nível de congestionamento.

2.5 Métricas de Desempenho

As métricas de desempenho permitem criar uma base para avaliação dos diferentes componentes da rede. O grupo de trabalho IPPM - *IP Performance Metrics*, do IETF - *Internet Engineering Task Force* tem como objetivo desenvolver um conjunto de métricas padronizadas que possam ser aplicadas para avaliação da qualidade, desempenho e confiança dos serviços de troca de dados na Internet. Essas métricas são desenvolvidas de forma que possam ser executadas por operadores de rede, usuários finais ou grupos independentes de testes.

Segundo PAXSON (1998) na Internet operacional há diversas quantidades relacionadas ao desempenho e à confiabilidade da Internet que necessitamos saber o valor. Quando tal quantidade é cuidadosamente especificada, a quantidade é denominada métrica.

Na seqüência será apresentado um conjunto de métricas utilizadas para avaliar o desempenho da rede.

2.5.1 Retardo

O retardo ou atraso, em redes de comunicação de dados, normalmente vem associado ao tempo necessário para uma informação atravessar a rede. O IPPM define duas RFCs para medir os retardos: o retardo de ida-e-volta (*Round-Trip Delay*) (Almes, 1999-C) ou retardo unidirecional (O*ne-Way Delay*) (Almes, 1999-A).

- **Retardo de ida-e-volta**: Definido como sendo o tempo decorrido desde a inserção do primeiro bit de um pacote IP1 de tamanho D1, até a chegada do último bit de outro pacote IP2 de tamanho D2 ao emissor. Nesta medida é necessária a utilização de dois pacotes IP diferentes: o primeiro pacote (IP1) é gerado pelo emissor, e o segundo é enviado pelo receptor em resposta à requisição do primeiro pacote (IP1) recebido.
- **Retardo Unidirecional**: É definido como sendo o tempo decorrido desde a inserção do primeiro bit de um pacote IP de tamanho D até a chegada do último bit do mesmo pacote IP ao destino. Na Internet a rota da origem até um destino pode ser diferente da rota do destino para a origem (rotas assimétricas). Neste caso seqüências de roteadores diferentes estão sendo usadas para as rotas. Conseqüentemente as medidas de retardo de ida-e-volta medem realmente o desempenho de dois trajetos distintos juntos. Medir cada trajeto destaca independentemente a diferença do desempenho entre os dois trajetos. Por exemplo, numa transferência de arquivos o desempenho depende na maior parte do desempenho no sentido do fluxo de dados do que no sentido inverso.

As principais motivações para medir o retardo na rede são apresentadas na RFC 2681 (Almes, 1999-C) e na RFC 2679 (Almes, 1999-A):

- As aplicações não interativas são pouco afetadas pelo retardo introduzido pelas redes geograficamente distribuídas – WAN (*Wide Area Network*). Porém, os usuários de aplicativos interativos como Telnet, Voz Sobre IP –

VoIP (*Voice over IP*), videoconferência e comércio eletrônico esperam um retardo limitado mínimo na recepção de uma resposta da rede.

- Retardos longos também afetam os protocolos da camada de transporte. Quanto maior o valor médio do atraso, mais difícil será a manutenção de altas taxas de transmissão nos protocolos da camada de transporte (camada TCP).

Segundo KUROSE (2004), o retardo entre dois equipamentos nas redes de comunicação, pode ser subdividido em quatro categorias: retardo de processamento, retardo de enfileiramento, retardo de transmissão e retardo de propagação. Cada uma dessas categorias de retardo possui uma origem diferente e a sua soma é o valor total do retardo.

- **Retardo de processamento:** é o retardo introduzido quando um pacote é encaminhado por equipamentos de interconexão de redes, tais como pontes, *switches* ou roteadores. Inclui o tempo necessário para examinar o cabeçalho do pacote e determinar para onde direcioná-lo. Além disso, agrega o tempo necessário para verificar a existência de erros nos bits do pacote. Neste caso, o retardo depende da velocidade dos circuitos internos, da arquitetura do dispositivo de comutação e do tamanho das filas de pacotes existentes dentro de tais dispositivos.
- **Retardo de Enfileiramento**: é definido pelo tempo decorrido entre a entrada do pacote na fila e a sua transmissão no enlace. O retardo para um pacote específico depende da quantidade de outros pacotes que chegaram antes e que já estão na fila. Este retardo pode variar significativamente de pacote para pacote, pois é o mais afetado pelos congestionamentos da rede.
- **Retardo de transmissão**: pode ser definido como o intervalo de tempo entre o início e o fim da transmissão de um pacote, ou seja, é o tempo necessário para por os dados digitais sobre uma linha de transmissão. Ele depende apenas do tamanho do pacote e da taxa de transmissão do canal.
- **Retardo de propagação**: pode ser definido como o intervalo de tempo entre o início da transmissão de um bit e sua chegada do outro lado do canal. Ele depende da velocidade de propagação do sinal no canal e da distância a ser percorrida pelo sinal.

2.5.2 Variação do Retardo

Segundo Demichelis (2002) a variação do retardo implica que o atraso de chegada dos pacotes não é constante. A variação do retardo também é conhecida como *jitter*. A figura 2.6 exemplifica como ocorre o *jitter*, neste caso T1 e T2 são os tempos de partida dos pacotes 1 e 2 respectivamente, enquanto R1 e R2 são os tempos de chegada ao destino dos pacotes 1 e 2 respectivamente.

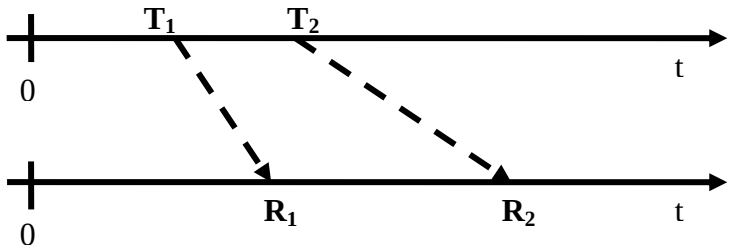

Figura 2.6: *Jitter* – Variação no atraso de chegada dos pacotes

Na seqüência estão listados alguns dos motivos para mensurar o *jitter* (Demichelis 2002).

As aplicações não interativas não são muito afetadas pelo *jitter*, porém os usuários de aplicativos que envolvem mídia contínua (áudio e vídeo) são extremamente prejudicados. O *jitter* é potencialmente mais danoso à qualidade dos serviços de mídia contínua (áudio e vídeo) do que o retardo total fim-a-fim. O retardo apenas introduz, no caso do transporte de voz, um intervalo entre o final da fala de uma pessoa e o início da fala da outra. Os pacotes que carregam mídias contínuas são normalmente gerados em suas origens a taxas constantes, mas ao atravessarem a rede, sofrem retardos diferentes (gerando o *jitter*), e chegam ao destino não sincronizados. Para amenizar esse efeito, os equipamentos e aplicativos que envolvem mídias contínuas possuem, no lado do receptor, um *buffer* para armazenamento de alguns pacotes e posterior envio ao usuário na taxa originalmente transmitida à rede de dados pelo transmissor. Este buffer é chamado de buffer de *dejitter*, que é utilizado para atenuar a diferença de tempo de chegada dos bits.

O *jitter* pode ser causado por diversos fatores, tais como: a variação do tamanho das filas de transmissão nos nós intermediários da rede, alteração das rotas entre origem e destino ou uso de sistemas de comunicação com diferentes tempos de propagação.

A variação do tamanho das filas de transmissão nos nós da rede pode ser minimizada priorizando-se o tráfego de mídias contínuas em relação aos outros tipos de tráfego e impedindo a grande variação dos tamanhos de pacotes, usando-se a técnica de fragmentação dos pacotes. Neste caso, os pacotes IP são divididos em datagramas com um tamanho máximo especificado, minimizando a variação do tempo necessário para serialização dos mesmos.

2.5.3 Perda de Pacotes

Um pacote é considerado perdido quando o destino não o receber após um tempo limite pré-determinado, conhecido como *timeout*. O *timeout* irá depender da distância, do caminho por onde passa o pacote e do tamanho das filas. As duas formas mais utilizadas de avaliar a perda de pacotes são Perda de Pacotes Unidirecional (*One-Way Packet Loss) (Almes, 1999-B)* e Perda de Pacotes de Ida-e-Volta (*Round-trip Packet Loss).*

- **Perda de Pacotes Unidirecional**: É a relação entre a quantidade de pacotes que são considerados perdidos e a quantidade de pacotes recebidos com sucesso apenas no caminho de ida entre o emissor e o destino. Esta medida geralmente é apresentada em porcentagem de um universo de pacotes enviados em certo período de tempo.
- **Perda de Pacotes de ida-e-volta**: É a relação dos pacotes que são considerados perdidos em relação aos pacotes que são recebidos com sucesso

pelo emissor até o destino e novamente do destino até o emissor. Esta medida, geralmente, é feita em percentagem de um universo de pacotes enviados em certo período de tempo.

Ao contrário do retardo fim-a-fim e do *jitter*, a perda de pacotes é mais danosa às aplicações de transferência de dados que às aplicações de transferência de voz e vídeo. Nas aplicações de transferências de dados a perda de pacotes é mais danosa porque, quando se está transferindo dados, todos os pacotes são igualmente importantes. Se um deles não chegar, o arquivo ou a mensagem que está sendo transmitida não está completo e será necessário que o pacote perdido seja retransmitido. Nas aplicações multimídia (dependendo da codificação de áudio e vídeo) os limites para perda de pacotes são mais relaxados, porque com a perda de alguns pacotes, a qualidade da conversação ou do vídeo cai, mas ainda assim o entendimento é mantido. Além disso, aplicações de voz ou vídeo não solicitam retransmissão de pacotes considerados perdidos. Isto porque, tais aplicações funcionam em tempo real e esperar pela retransmissão de um pacote perdido seria mais danoso para a qualidade da conversação ou do vídeo do que simplesmente descartá-lo. Algumas codificações de áudio e vídeo, porém não toleram perdas de pacotes acima de um patamar máximo que permita a inteligibilidade da transmissão.

Diversos fatores geram perdas de pacotes nas redes de comunicação, dente eles:

- ***Overflow* de filas nos roteadores**: este tipo de perda, em geral, está associada ao congestionamento do canal;
- **Descarte pelos protocolos**: por exemplo, o campo TTL (*Time To Live*) é usado para limitar o tempo de transmissão dos pacotes. Esse campo recebe um valor inicial quando o pacote é criado e sempre que um roteador repassa este pacote ele decrementa o valor desse campo. Quando o valor de TTL chega a zero, o pacote é descartado. Em geral, isto acontece quando o pacote está em *loop*, mas também pode acontecer quando há um grande congestionamento, ou até mesmo quando há um erro na atribuição do valor inicial do campo.
- **Troca de bits no meio físico**: em geral é ocasionada por distorção dos sinais, tais como ruídos, atenuação e ecos. Quando o pacote chega ao próximo nó da rede, o *checksum* do pacote é verificado e o erro é detectado. Uma vez detectado o erro, o pacote é descartado.

2.5.4 Largura de Banda

A largura de banda é a avaliação da capacidade de transmissão em determinado enlace e pode ser avaliada de diferentes formas, como as listadas abaixo (LAI, 2000):

- **Largura de Banda de Contenção (*Bottleneck Bandwidth ou Capacity*)**: É a taxa máxima que um fluxo pode alcançar em um caminho sem a presença de tráfego (sem carga).
- **Largura de Banda Disponível (*Available Bandwidth*)**: É a taxa máxima que um fluxo pode alcançar em um caminho na presença de tráfego (com carga).
- **Largura de Banda Utilizada (*Bandwidth Utilization*)**: É o agregado de todos os tráfegos presentes no enlace.

- **Largura de Banda Alcançável (Achievable Bandwidth)**: É a taxa de transferência entre dois pontos, considerando um conjunto de outras condições como: o protocolo de transmissão (TCP ou UDP (*User Datagram Protocol*)), *hardware* do *host* (velocidade do processador, velocidade do barramento e velocidade da placa de rede), sistema operacional e configuração do sistema (*buffers* TCP).

O conhecimento da largura de banda de determinado enlace pode auxiliar a administradores, usuários e aplicações a tomarem decisões para a melhor utilização do mesmo. O conhecimento da capacidade do enlace pode ser útil em alguns casos, como os que seguem (LAI, 2000):

- **Adaptação de Aplicações:** o exemplo mais tradicional da possibilidade de adaptação são as aplicações multimídia (tipicamente de áudio e vídeo). Essas aplicações poderiam adaptar o conteúdo a ser enviado ao usuário, de acordo com a capacidade de transmissão dos enlaces. Outro exemplo é um servidor *Web*, o qual poderia escolher tamanho, qualidade das imagens e os sons que seriam enviados ao cliente.
- **Análise de protocolos:** obtendo informações confiáveis da rede os protocolos podem ser analisados e aperfeiçoados para maximizar os recursos disponíveis.
- **Roteamento:** quando existem múltiplos caminhos para determinada rota a escolha pode ser mais eficiente com o conhecimento das condições da rede. Além disso, no caso de tráfego *multicast* o conhecimento das condições da rede permite a construção de uma árvore de distribuição eficiente.

Quando a *largura de banda de contenção* de determinado enlace é insuficiente para a quantidade de tráfego gerado ocorre um congestionamento. Em um congestionamento pacotes são descartados e atrasados com impacto direto nas métricas relacionadas anteriormente.

2.6 Considerações Parciais

Neste capítulo foram apresentados conceitos básicos sobre gerenciamento de rede, o modelo de gerenciamento Internet e as principais classificações dos paradigmas de gerenciamento. Estas definições serão úteis para a melhor compreensão do modelo de gerenciamento que será apresentado no capítulo 4.

As métricas que são utilizadas para quantificar o desempenho são de difícil mensuração quando as técnicas necessitam de cooperação entre origem e destino e os equipamentos envolvidos estão em domínios administrativos diferentes. Porém, são inegáveis os benefícios que a avaliação precisa das mesmas traria para o desenvolvimento de novas aplicações. Para este objetivo uma vasta coleção de técnicas e ferramentas se propõe a avaliá-las sem a necessidade de cooperação entre as entidades envolvidas. A maioria dessas técnicas (por exemplo, pares de pacotes ou trens de pacotes) baseia-se no tempo entre o envio de pacotes para um destino e o recebimento das respectivas respostas. Um dos problemas é que quanto mais nodos intermediários existirem entre origem e destino maior a probabilidade da inferência das medições apresentarem erros.

Como uma infra-estrutura de medição permite avaliar o desempenho da rede por segmentos, o uso destas mesmas técnicas pode ser mais eficiente. Além disso, essas

técnicas isoladamente não conseguem prover a mesma quantidade e qualidade de informações quando associadas a uma infra-estrutura de medição.

No próximo capítulo é apresentado um conjunto de infra-estruturas de medição, detalhando suas características e seu funcionamento para visualizar um de seus maiores problemas quando utilizada em larga escala, o gerenciamento.

3 INFRA-ESTRUTURAS DE MEDIÇÃO

A avaliação de desempenho da rede é importante por diversos motivos, porém a sua mensuração é uma tarefa complexa. O funcionamento descentralizado da Internet impossibilita qualquer tipo de monitoração centralizada e, monitorar em uma das pontas da rede pode não ser eficiente, pois não se tem conhecimento absoluto do caminho dos dados. Uma solução é dotar os provedores de acesso de *hardwares* e/ou *softwares* especiais que possibilitem analisar o desempenho entre esses pontos. Esta solução, chamada de infra-estrutura de medição de desempenho, permite avaliar o desempenho e pode facilitar a rápida detecção e diagnóstico de vários problemas. Com uma infra-estrutura de medição pode-se (MURRAY, 2001):

- estabelecer um padrão de comportamento da rede, para diferenciar e detectar comportamentos anormais;
- detectar e localizar problemas específicos na Internet (ataques de DoS, problemas de configuração em roteadores, falhas em enlaces ou *hardwares*);
- identificar gargalos;
- manter os dados coletados por um longo período (histórico) para análises posteriores (tendências);
- possibilitar a coleta de informações para eventos específicos (um experimento).

Considerando que as infra-estruturas de medição são soluções importantes para avaliação de desempenho, neste capítulo são apresentadas infra-estruturas de medição que disponibilizam suas informações de forma pública. Inicialmente será apresentado o "estado da arte" das infra-estruturas de medição. Na seqüência apresentamos (em ordem alfabética) as infra-estruturas de medição encontradas na literatura. Para cada infra-estrutura são apresentados seu funcionamento e os principais componentes. Ao final do capítulo é apresentada uma comparação das infra-estruturas descritas.

3.1 Estado da Arte

Após o levantamento das infra-estruturas de medição (publicamente disponíveis), nas quais foram detectados os problemas que justificam esse trabalho, iniciou-se o processo de busca por trabalhos com os mesmos objetivos dessa proposta. Essa pesquisa foi infrutífera, pois não foram encontrados na literatura trabalhos cujo objetivo era efetivamente o gerenciamento de infra-estruturas de medição.

Por outro lado, pode-se constatar que os grupos de pesquisa que trabalham com infra-estruturas de medição, percebendo os problemas nas infra-estruturas existentes,

vêm propondo novas soluções e incorporando funções de gerenciamento para alguns de seus componentes.

Nesse sentido, a proposta mais atual e que conseqüentemente prevê o maior número de funções que podem ser consideradas de gerenciamento é a proposta do grupo JRA1(JRA1 2006) da GÉANT2. A proposta denominada GFD (*General Framework Design*) tem como objetivo definir e desenvolver capacidades de medição e monitoração para permitir que usuários e grupos de usuários possam compreender o desempenho das redes que utilizam, normalmente redes que envolvem múltiplos domínios (BOOTE, 2007).

O GFD é uma proposta de arquitetura em três níveis, que apresentados de baixo para cima são: O primeiro nível consiste nas ferramentas de medição e nas bases de dados que armazenam as informações dos testes. Esses componentes tipicamente não estão disponíveis fora do domínio administrativo que estão instalados. O segundo nível (acima do nível anterior) consiste efetivamente no escopo do GFD, provendo uma interface (baseada em serviços) para acesso de forma ubíqua as ferramentas de medição e as bases de dados. O último nível consiste nas ferramentas de visualização e alerta, baseados em informações repassadas pelo nível inferior. A figura 3.1 sintetiza o escopo da proposta do GFD.

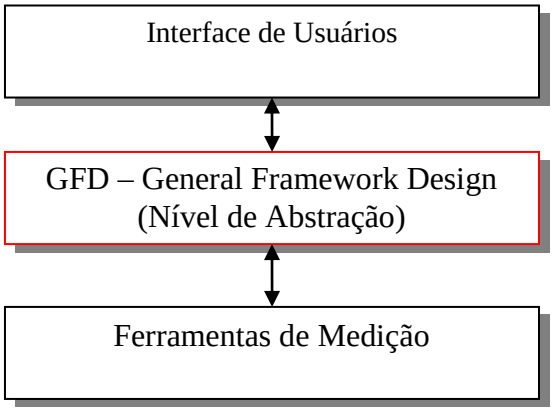

Figura 3.1: Escopo do GFD

Apesar de semelhantes na escolha da tecnologia de implementação (orientação a serviços) e de implementar um conjunto de funções que permite gerenciar alguns componentes da infra-estrutura, a proposta deste trabalho e do GFD têm enfoques diferentes.

O GFD é uma nova infra-estrutura de medição que intercepta as requisições dos usuários e interage com as ferramentas de medição. A arquitetura proposta neste trabalho é complementar a uma infra-estrutura de medição, sendo genérica o suficiente para trabalhar em conjunto com a própria proposta GFD, por exemplo. Sendo o GFD uma nova infra-estrutura de medição ela ainda não atende todos os requisitos de gerenciamento que julgamos necessários para garantir o funcionamento dos componentes de uma infra-estrutura de medição, como por exemplo, as verificações no *hardware* disponibilizado para a infra-estrutura.

As próximas seções apresentam em detalhes as principais infra-estruturas de medição de desempenho encontradas na literatura.

3.2 GFD – General Framework Design

Como visto na seção anterior, o GFD é uma proposta de infra-estrutura de medição de desempenho em um ambiente multi-domínio e inteoperável com outras infra-estruturas, permitindo o uso de diversas ferramentas de medição de desempenho. O GFD propõe um conjunto de serviços para a infra-estrutura que é acessada por interfaces bem definidas. A proposta GFD prevê a existência de um conjunto de serviços, divididos em serviços básicos e serviços opcionais (BOOTE, 2007).

O GFD define como **Serviços Básicos** um conjunto mínimo de serviços que precisam ser definidos para permitir que a estrutura possa atender a maioria das necessidades atuais, como: recuperação de dados, realização de testes, autorização e autenticação de usuários e localização de um ponto de medição. Os serviços básicos são: Serviço de Ponto de Medição, Serviço de Armazenamento de Medições, Serviço de Publicação e Descoberta e o Serviço de Autenticação e Autorização. Os **Serviços Adicionais** são serviços opcionais que executam alguma tarefa específica e não são requeridos por todos os serviços. Os serviços adicionais são: Serviço de Proteção de Recurso, Serviço de Transformação e Serviço de Topologia. Estes serviços são detalhados na seqüência.

3.2.1 Serviço de Ponto de Medição (*Measurement Point Service*)

O serviço de Ponto de Medição (MP) é responsável por manipular as ferramentas de medição, realizando testes de forma ativa ou passiva. De forma resumida, é uma interface padrão para as diferentes ferramentas de medição.

Esse serviço não armazena nem faz qualquer processamento com os dados coletados, estas são funções de outros serviços (descritos na seqüência). Diferentes tipos de medição podem ser implementados. Inicialmente seis tipos foram definidos:

- Medição de forma ativa do Atraso, Perda e Variação do Atraso dos pacotes;
- Medições coletadas diretamente nos equipamentos de rede;
- Medição baseada em fluxo;
- Medidas de largura de banda no gargalo;
- Medidas de largura de banda disponível;
- Captura passiva de pacotes.

Um ponto de medição pode ser um *hardware* localizado no núcleo da rede para coletar dados e determinar o comportamento da rede ou pode ser um *software* instalado na borda da rede em algum host para efetuar medições fim-a-fim.

3.2.2 Serviço de Armazenamento de Medições (*Measurement Archive Service*)

O Serviço de Armazenamento de Medições (MA) é usado para armazenar e publicar dados históricos de monitoração produzidos pelo Serviço de Ponto de Medição ou pelo Serviço de Transformação. Este serviço não cria ou transforma nenhum dado e, é essencial por muitas razões, incluindo eficiência, para prover dados históricos para análises de tendências, para reduzir a carga sobre os MPs e para garantir alta disponibilidade dos dados de monitoração.

Devido aos diferentes tipos de dados obtidos nas medições, são necessárias múltiplas bases de dados para armazená-los. O serviço deve conhecer o tipo de informação que está armazenado em cada base de dados e utilizar essa informação para armazenamento e recuperação dos dados.

Para obter alta disponibilidade dos dados são necessários ao serviço recursos de backups, redundância de dados e tolerância a falhas.

3.2.3 Serviço de Publicação e Descoberta (*Lookup Service*)

O serviço de Publicação e Descoberta (LS) permite aos usuários descobrir e publicar outros serviços (MP, MA, serviço de Autenticação, etc.), de acordo com as suas necessidades. O LS deve fornecer aos solicitantes todas as informações necessárias para localizar e contatar outros serviços diretamente, possibilitando utilizar as funcionalidades providas pelos mesmos.

O LS funciona como um serviço de diretório, onde serviços podem ser anunciados e os solicitantes podem localizar os serviços que necessitam. O acesso às informações é dependente do tipo de autorização do usuário. Por exemplo, de acordo com o interesse de quem fornece as informações, algumas delas podem estar ocultas a determinados tipos de usuário.

O serviço de Publicação e Descoberta é um elemento fundamental na estrutura proposta no GFD porque permite o tratamento dos serviços de forma independente, como uma parte visível do sistema. Isso permite a publicação de novos serviços ou a retirada de outros sem interromper o oferecimento dos demais, característica que torna dinâmica e flexível o acesso e uso dos serviços. O LS deve aceitar o processo de descoberta por tipo ou características dos serviços, permitindo aos usuários a possibilidade de consultas múltiplas e exaustivas. Além disso, para o ambiente dinâmico que é a Internet, o LS deve atender às características de Disponibilidade, Escalabilidade, Segurança e Tolerância a Falhas.

O LS está implementando uma função de "*heartbeat*" que possibilite verificar se um determinado serviço continua disponível.

3.2.4 Serviço de Autenticação e Autorização (*Authentication Service*)

O serviço de Autenticação e Autorização (AS) provê a funcionalidade de autenticação para o framework e o fornecimento de autoridade de atributo (autoridade de atributo é um componente do serviço de Autorização que decide quais atributos de um determinado recurso podem ser descobertos). O GFD define um conjunto de requisitos para esse serviço.

O serviço deve suportar clientes com múltiplas identidades, como por exemplo, usuários ligados a mais de uma organização ou que têm papéis diferentes em tempos diferentes.

- Utilizar autenticação baseada em "papéis" permitindo proteger a privacidade do usuário;
- Deve permitir a federação dos serviços de autenticação para formar comunidades com aceitação mútua;
- Os detalhes de federação devem estar escondidos dos outros serviços dentro de um determinado domínio administrativo. A relação de "confiança" dentro

de um domínio existente deve ser entre os serviços no domínio e o serviço de Autenticação do mesmo;

As solicitações autenticadas permitem aos serviços fazerem consultas reversas para a entidade autenticadora. Isto possibilita cada serviço determinar que direitos deve ter um solicitante de um determinado recurso sem divulgar a identidade do mesmo.

3.2.5 Serviço de Proteção de Recurso (*Resource Protector Service*)

O serviço de Proteção de Recurso (RP) é usado para controlar o consumo de recursos limitados, como por exemplo, largura da banda de rede. Possui também um componente de agendamento para controlar o consumo de recursos dependentes de tempo. Os serviços consumidores de recursos contatarão o RP para adquirir as permissões necessárias. Um exemplo que necessita de permissão é o MP.

Devido à redução de flexibilidade, o uso de RP deve ser restrito para proteger recursos limitados, ou seja, nem todos os serviços necessitam solicitar autorização ao RP. Por questões de flexibilidade, faz sentido permitir aos MPs solicitarem recursos de uma lista de RPs (potencialmente diferentes, dependendo dos recursos necessários para executar determinado teste). Adicionalmente, cada RP pode ser configurado com uma lista de RPs adicionais para proteger recursos de outros tipos, permitindo uma organização hierárquica dos serviços de RP, característica que pode ser útil para alguns tipos de proteção de recurso.

O administrador do MP configura o RP de acordo com as suas informações locais, como por exemplo, CPU, memória do MP ou de acordo com a autorização para determinado teste.

3.2.6 Serviço de Transformação (*Transformation Service*)

O serviço de Transformação (TS) é usado para canalizar os dados entre os outros serviços da estrutura. Pode ser usado para executar qualquer função sobre os dados. Esse serviço fica entre os produtores de dados (por exemplo, MPs, MAs, ou outro TS) e consumidores de dados (por exemplo, MAs, outro TS, ou clientes). Ele é ao mesmo tempo um consumidor e produtor de dados.

Existe uma variedade de funções potenciais que um TS pode prover, incluindo agregação, correlação, filtro e conversão de dados. Um TS, por exemplo, pode receber dados de múltiplos produtores e criar uma correlação específica ou compactar um conjunto de dados obtidos recentemente e publicá-los em um serviço de MA.

A arquitetura previne que um TS possa subscrever seus próprios dados, evitando laços e problemas potenciais de segurança.

3.2.7 Serviço de Topologia (*Topology Service*)

O serviço de Topologia é usado para coletar informações de topologia da rede e torná-las disponíveis para todos os serviços da estrutura. O Serviço de Topologia é um exemplo específico de um Serviço de Transformação. O serviço coleta informações de topologia de uma variedade de fontes (por exemplo, serviços MPs) e usa algoritmos para deduzir a topologia de rede.

O Serviço de Topologia deve refletir as múltiplas camadas de rede, ou seja, a topologia deve ser descrita desde o nível de domínio administrativo até o nível de comprimentos de onda. Adicionalmente, a topologia de rede poderia conter informações geográficas, por exemplo, usando coordenadas de GPS.

Conhecendo a topologia da rede, o sistema de medição pode aperfeiçoar sua operação. Por exemplo, o LS confia no Serviço de Topologia para determinar quais serviços de MP estão mais próximos. Da mesma forma, um componente de serviço pode perguntar pela proximidade de nó. Como a definição de proximidade pode variar, dependendo do contexto da solicitação, é desejável deixar em aberto a natureza exata da consulta de proximidade e simplesmente permitir ao cliente fazer várias consultas ao serviço LS, de acordo com suas necessidades particulares.

Adicionalmente, o Serviço de Topologia pode ser usado para gerar mapas que ilustram a rede com dados das medições pertinentes.

3.2.8 perfSonar - protótipo do GFD

O perfSONAR (*Performance focused Service Oriented Network monitoring Architecture*) é o desenvolvimento de um protótipo para o *framework* do GFD. O perfSONAR iniciou seu desenvolvimento pelo JRA1 com a colaboração do Internet2 (USA) e ESnet(USA). Recentemente a RNP, por intermédio do GT-Medições passou a integrar o desenvolvimento do protótipo.

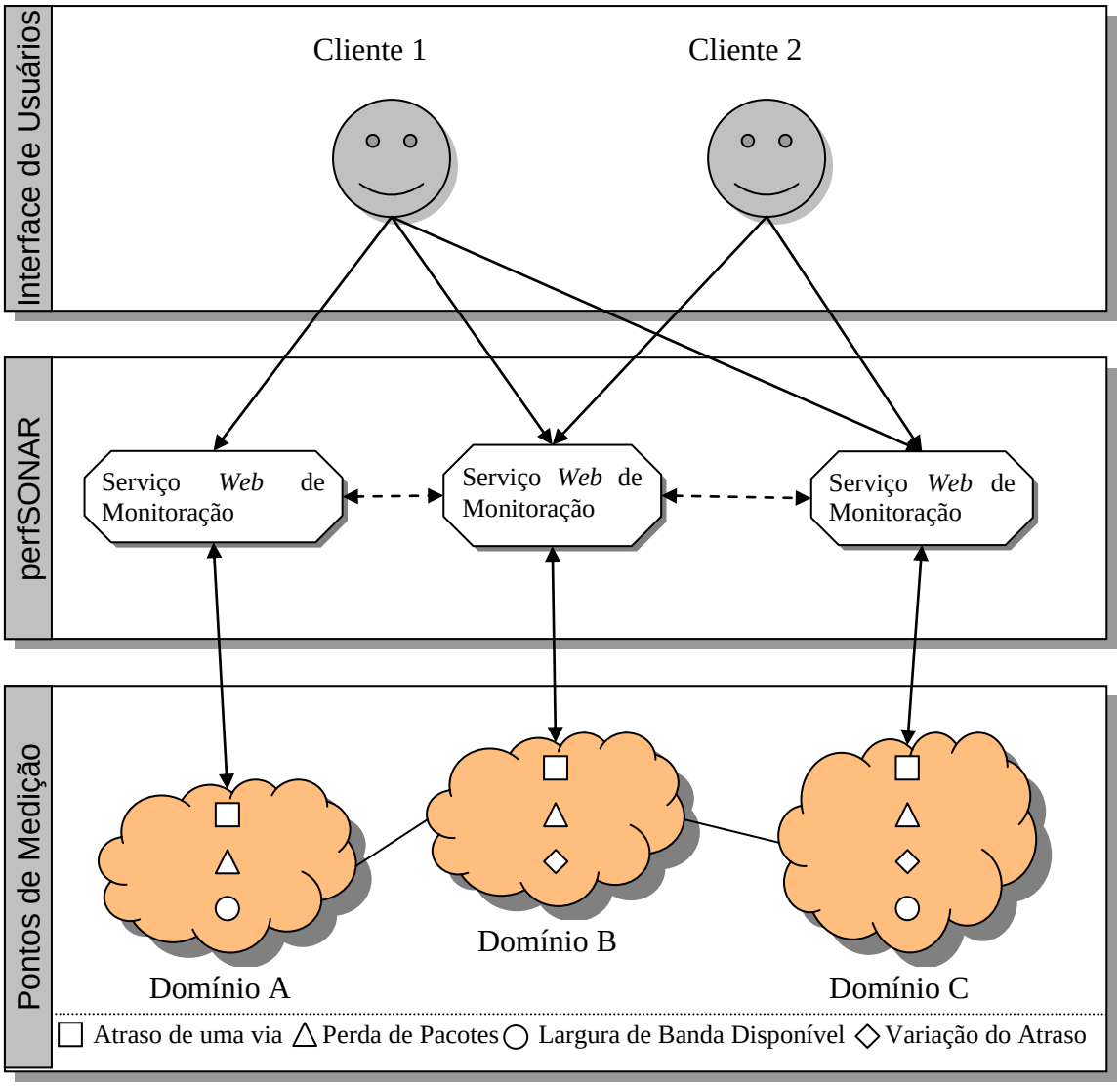

Figura 3.2: Escopo de Interação do perfSONAR

Segundo HANEMANN (2005), o perfSONAR possui três contextos:

- é um "consórcio" de organizações que buscam construir *middleware* de desempenho de rede que de forma integrada em múltiplas redes seja útil para análise de desempenho intra-rede e inter-rede. Uma das metas principais é facilitar a detecção e solução de problemas de desempenho fim-a-fim em enlaces que passam por várias redes.
- é um protocolo. Possui um conjunto de regras definindo a sintaxe e semântica para comunicação entre os componentes. O protocolo é baseado em mensagens XML/SOAP.
- é um conjunto de código exemplo (implementação de serviços) que implementa uma infra-estrutura de medição de desempenho de forma interoperável. Os conjuntos de código são desenvolvidos por diferentes entidades participantes do projeto. Os serviços desenvolvidos atuam como uma camada intermediária entre as ferramentas de medição de desempenho e as aplicações de diagnóstico ou visualização. A figura 3.2 sintetiza o escopo de interação do perfSONAR.

3.2.9 Gerenciamento GFD – perfSONAR

A proposta do GFD – perfSONAR prevê funções para a infra-estrutura que podem ser consideradas como funções de gerenciamento. Na área de gerenciamento de falhas prevê a existência de *backups* e técnicas de recuperação de falhas para os dados coletados nas medições. Na área de configuração está previsto um controle do uso dos recursos dos pontos de medição de acordo com regras de configuração e da capacidade do equipamento. Na área de segurança prevê controles de autenticação e autorização para todos os usuários da infra-estrutura de medição. Inclusive com o uso de federações e de usuários com múltiplas identidades.

3.3 AMP

O *MOAT (Measurement and Operations Analysis Team)* dentro do NLANR [1] (*National Laboratory for Applied Network Research)* desenvolveu uma infra-estrutura para análise da rede, conhecida como *NAI - Network Analysis Infrastructure.* O objetivo deste projeto foi criar uma infra-estrutura que suporte medições e análises através da coleta e publicação de dados “brutos”, além da visualização das medições (McGREGOR, 2000). A infra-estrutura proposta é apresentada na figura 3.3 e permitindo coletar informações por meio de monitoração passiva, monitoração ativa e monitoração de controle (são informações de controle as rotas ou gerenciamento).

O *AMP – Active Measurement Project* é parte da NAI, cujo foco foi efetuar medições ativas entre os *site*s participantes.

[1] O NLANR foi encerrado em 2006.

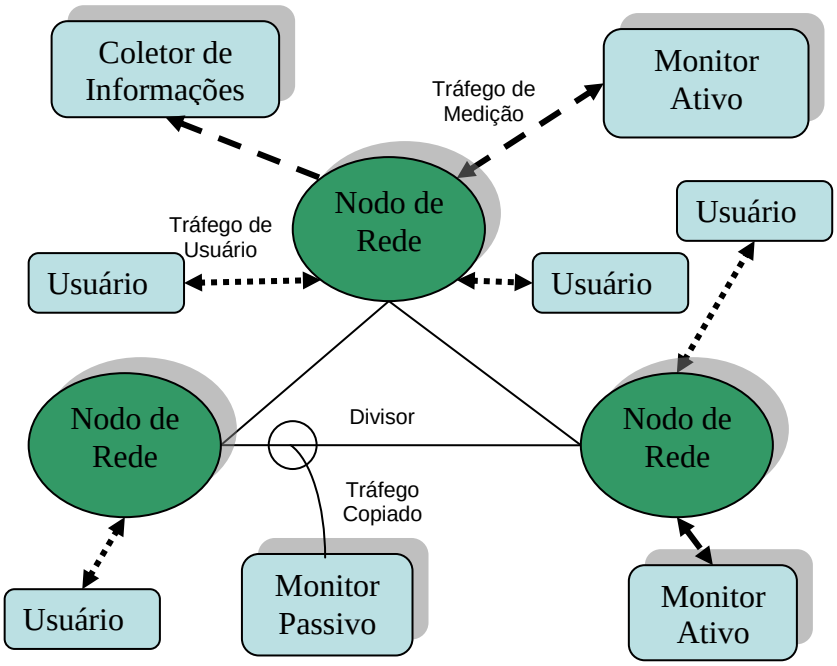

Figura 3.3: NAI e os tipos de monitores

3.3.1 Arquitetura NLANR-AMP

A arquitetura AMP que é apresentada na figura 3.4, mostra que os monitores são configurados para interagirem apenas com os equipamentos remotos selecionados, por uma questão de escala é inviável efetuar os testes entre todos os monitores (todos com todos). As métricas utilizadas são tempo de ida-e-volta, pacotes perdidos, topologia e largura de banda. Cada um dos sistemas envia os dados para um coletor central que disponibiliza as informações.

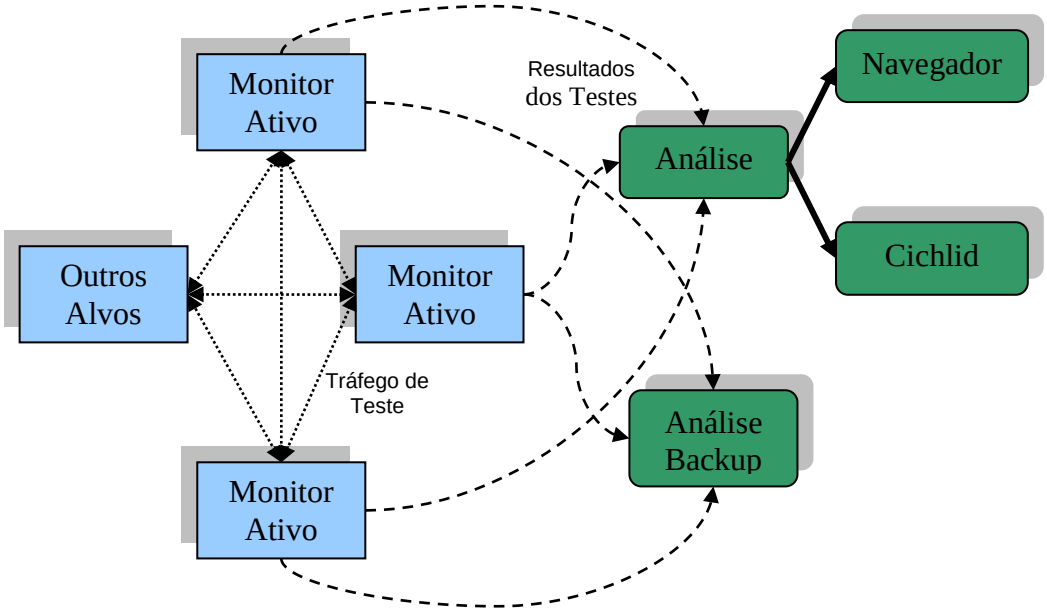

Figura 3.4: Arquitetura AMP

3.3.2 Metodologia

Cada monitor envia um pacote ICMP (*Internet Control Message Protocol*) para cada um dos outros monitores a cada minuto (mais 15 segundos aleatórios) e grava o tempo

(ou a ausência) do retorno do pacote. Um pacote é considerado perdido quando 4 pacotes *ICMP echo request* são enviados e não é obtido nenhum retorno. Adicionalmente, a cada 10 minutos (mais 15 segundos aleatórios) a rota para cada outro monitor é armazenada, usando o *traceroute.* Os testes de largura de banda somente são executados sob demanda, uma vez que possui um alto custo em termos de tráfego de rede. Os testes de largura de banda disponíveis são: *ping –F, treno*, transferência de dados TCP em massa e transferência de dados UDP em massa.

Os dados coletados eram enviados para o *site* central AMP, no *SDSC* (*San Diego Supercomputer Center*), para processamento e publicação na *Web*. Para aumentar a robustez do sistema, dois equipamentos centrais eram usados. Os dados eram enviados independentemente para cada um dos monitores.

3.3.3 Gerenciamento AMP

A infra-estrutura AMP não previa nenhum mecanismo de gerenciamento de seus componentes. A única tarefa de gerenciamento que se pode considerar é o fato de possuir dois equipamentos para recepção e análise dos dados (espelho ativo) que, em caso de falha, aciona o equipamento reserva. De acordo com as áreas de gerenciamento OSI, a infra-estrutura AMP implementa apenas uma função na área de falhas.

3.4 BW

O objetivo do projeto *Internet End-to-end Performance Monitoring – Bandwidth (IEPM-BW)* é o desenvolvimento de uma infra-estrutura leve e baseada em padrões de tecnologias abertas, permitindo produzir aplicações de medição e predição de desempenho fim-a-fim nas redes monitoradas (IEPM, 2005).

3.4.1 Arquitetura BW

A arquitetura IEPM-BW é composta basicamente de dois tipos de equipamentos, que são: Equipamento de Medição e Equipamento Remoto.

Equipamento de Medição: é o responsável por executar as ferramentas de medição, denominadas *sensors*, armazenar os dados recebidos dos sensores, extrair, analisar e reportar as informações via *web*.

Equipamento Remoto: este equipamento executa o servidor, responsável por responder aos testes iniciados pelos sensores.

Como as respostas dos sensores são armazenadas num sistema de arquivos de rede, as funções do equipamento de medição podem ser divididas em dois equipamentos, um que efetua as medições e outro que extrai, analisa e disponibiliza os dados na *web*.

Cada *site* presente na arquitetura trabalha com seus colaboradores para decidir quais os equipamentos remotos que cada um vai monitorar, tipicamente múltiplos equipamentos remotos são monitorados por um equipamento de monitoração. Para cada equipamento remoto é necessária uma conta com acesso via SSH (*Secure Shell*), para o equipamento de monitoração. Após o acesso ser permitido o equipamento de monitoração configura a conta remotamente e armazena estas informações em uma base de dados local.

A figura 3.5 apresenta as ligações entre os participantes do projeto.

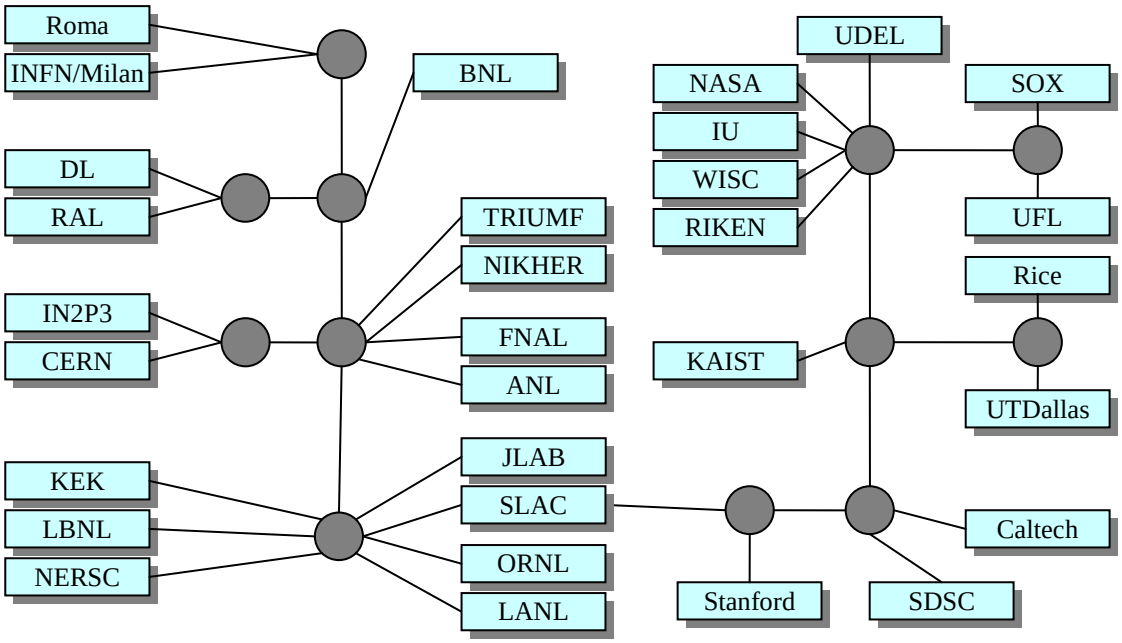

Figura 3.5: Topologia dos *site*s participantes

3.4.2 Metodologia

O equipamento de monitoração inicia um conjunto de medições em intervalos regulares gerenciados pelo *cron* do sistema Unix. Tipicamente os intervalos são da ordem de uma ou duas horas. Os intervalos atuais dependem da carga aceitável dos enlaces dos equipamentos monitorados e da quantidade de tempo necessária para efetuar a medição em todos os equipamentos remotos. O início da medição é de forma aleatória, adicionada de uma espera de até 15 minutos, obedecendo a uma distribuição aleatória uniforme.

Para cada conjunto de testes, o equipamento de medição seleciona cada equipamento remoto e executa um *ping* por 10 segundos, depois executa um *traceroute* (com uma única medida por salto) e após faz o teste usando o Iperf (NLANR, 2004).

Para prover robustez, os servidores são remotamente iniciados e encerrados para cada medição. Os comandos do sensor (por exemplo, Iperf) são executados em uma tarefa separada, assim ela pode ter seu tempo de execução monitorado e em caso de problemas pode ser encerrada. Os resultados de alguns testes são observados antes de continuarem os outros testes, por exemplo, se um *ping* para um equipamento, que normalmente responde, falhar, os próximos testes não são efetuados.

Os dados são extraídos dos *logs* e armazenados em tabelas (com formatos documentados) e disponibilizados na *web*. A análise é executada nos dados das tabelas e são produzidas páginas *web* com gráficos de séries temporais, histogramas e tabelas estatísticas.

3.4.3 Gerenciamento BW

Nesta seção são usadas as áreas funcionais do modelo OSI para descrever as funções de gerenciamento presentes na infra-estrutura BW. No gerenciamento de Falhas, ela implementa um registro das falhas, que associado a uma ferramenta de análise notifica o contato na ocorrência de falhas repetidas. No gerenciamento de Configuração,

implementa um mecanismo automático de inicialização baseado em uma configuração pré-estabelecida. Além disso, possui uma ferramenta que a cada inicialização coleta as informações de configuração do sistema remoto e envia para um equipamento de administração central. Quanto ao gerenciamento de Segurança, a infra-estrutura exige o cadastro do cliente no equipamento remoto para efetuar os testes. Já no gerenciamento de Contabilização e *Performance* a infra-estrutura BW não implementa nenhuma função.

3.5 MonALISA

A proposta da arquitetura MonALISA, desenvolvida no *Caltech – Califórnia Institute of Technology,* é prover um serviço global para monitorar, controlar e otimizar sistemas complexos, como Grids de alcance global.

A arquitetura MonALISA, abreviatura de *Monitoring Agents in a Large Integrated Services Architecture* é descrita como um conjunto de subsistemas autônomos, *multi-threads*, baseado em agentes que são registrados como serviços dinâmicos e podem colaborar e cooperar executando um grande número de tarefas de monitoração em aplicações distribuídas de larga escala. Estes serviços permitem ser descobertos e usados por outros serviços ou clientes que requerem tal informação. A arquitetura MonALISA foi desenvolvida para facilmente integrar procedimentos e ferramentas de monitoração existentes e prover as informações de uma maneira auto-descrita e dinâmica a qualquer outro serviço ou cliente (NEWMAN, 2003).

O desenvolvimento foi baseado em uma arquitetura de serviços dinâmica e distribuída (DDSA – *Dynamic Distributed Services Architecture*) implementada usando Java/JINI e tecnologias de serviços *Web*. O *framework* DDSA usa um conjunto de estações servidoras, uma para cada *site* ou aplicação, para gerenciar de forma robusta e escalável sistemas globais. Um serviço no *framework* DDSA é um componente que interage autonomamente com outros serviços por *proxies* dinâmicos ou por agentes que usam protocolos autodescritos. O paradigma de mobilidade de código (agentes móveis ou *proxies* dinâmicos) usados no DDSA permite estender as possibilidades dos paradigmas de chamada de procedimento remoto – RPC (*Remote Procedure Call*) e cliente – servidor, pois neste caso, tanto o código como os parâmetros apropriados são baixados para sistema.

Segundo NEWMAN (2003) este paradigma de mobilidade de código possui várias vantagens, dentre as quais:

- comunicação assíncrona otimizada;
- execução paralela dinâmica;
- mobilidade autônoma;
- interação remota e adaptabilidade.

3.5.1 Arquitetura MonALISA

Os pontos principais da arquitetura são (TOARTA, 2005): o Mecanismo de Coleta de Dados; o Mecanismo de Registro e Descoberta; os Agentes de Alarme, Filtros e Predicados; o Serviço de *Proxy* para clientes; Repositórios e Pseudoclientes; e Serviços de Administração.

Mecanismo de Coleta de dados

No centro do mecanismo de monitoração está um sistema *multi-thread* usado para executar muitas tarefas de coleta de dados em paralelo e de forma independente. A utilização de *threads* permite monitorar nodos heterogêneos, com variados tempos de respostas e evita que a ocorrência de problemas ou falhas em uma das tarefas de monitoração interfira na execução das demais. Para finalizar tarefas com problemas e re-escalonar tarefas não completadas, o sistema mantém uma *thread* dedicada de controle. Uma fila de prioridades é mantida para as tarefas que necessitam ser executadas periodicamente. A figura 3.6 apresenta uma visão esquemática do mecanismo de coleta.

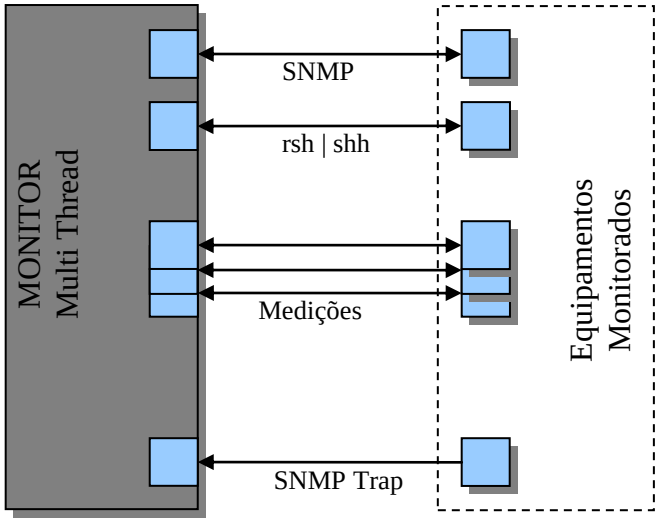

Figura 3.6: Visão Esquemática do Mecanismo de Coleta de Dados

Um Módulo de Monitoração é uma unidade carregada dinamicamente que executa um procedimento (roda um *script*, programa ou efetua uma requisição SNMP) para coletar um conjunto de métricas. Em geral, um Módulo Monitor é uma classe que usa um procedimento para obter um conjunto de valores e apresentá-los de forma simples e padronizada. Um módulo monitor pode ser carregado dinamicamente de um *site* centralizado, quando necessário. Isso facilita a atualização de grandes sistemas de monitoração e de prover novas funcionalidades. Assim, os usuários podem implementar facilmente qualquer novo módulo dedicado e usá-lo no *framework*.

Mecanismo de Registro e Descoberta

Os serviços MonALISA são registrados em um conjunto *JINI LUS (Lookup Discovery Services)* como parte de um grupo e com um conjunto de atributos. Os LUSs também são serviços JINI e, como tal, podem se registrar mutuamente. No caso de dois LUSs possuírem grupos comuns, qualquer mudança de estado detectada por um deles é imediatamente repassada para o outro. Com isso, é possível construir uma rede confiável e distribuída para registro dos serviços, além de permitir a adição e remoção dinâmica de LUSs no sistema. O registro é baseado no mecanismo de “aluguel”, que é verificado periodicamente. Caso a renovação do aluguel falhe, o serviço é removido do LUSs e uma notificação é enviada a todos os clientes e serviços que estão subscritos para receber estes eventos. Qualquer serviço pode usar o *Lookup Discovery Service*

(LUS) para localizar todos os serviços MonALISA que estão rodando como parte de um ou mais grupos ou "comunidades".

O Mecanismo de Descoberta é usado para notificação do início de novos serviços ou da indisponibilidade de outros. A comunicação entre os serviços ou clientes interessados é baseada no mecanismo de notificação de eventos remotos, o qual também suporta subscrição.

A aplicação cliente conecta diretamente com cada serviço do qual está interessada em receber informações de monitoração. Para fazer isso, é necessário efetuar o *download* do Proxy para o serviço de seu interesse de uma lista de possíveis endereços *Web*, especificados com atributos de cada serviço, depois deve instanciar as classes necessárias para a comunicação com o mesmo. A figura 3.7 apresenta o mecanismo de registro e descoberta.

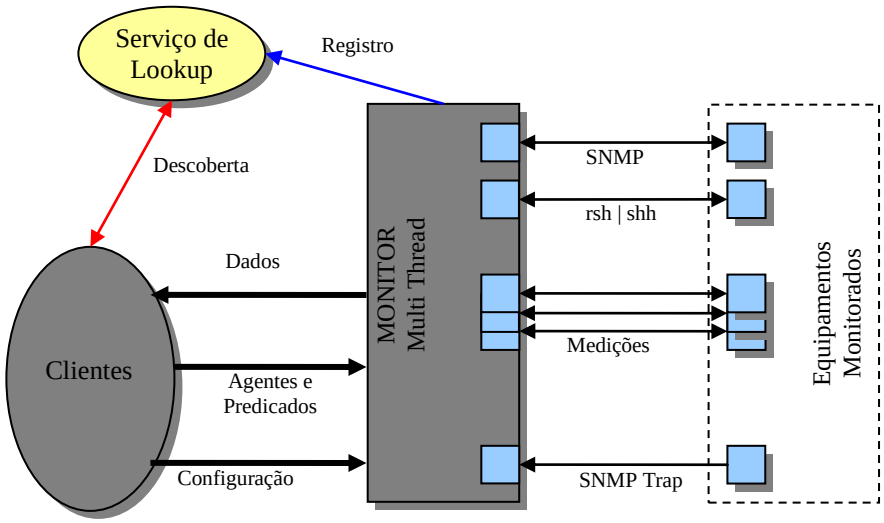

Figura 3.7: Mecanismo de Registro e Descoberta

Agentes de Alarme, Filtros e Predicados

Os clientes podem buscar qualquer dado atual ou histórico usando o mecanismo de predicados. Estes predicados são baseados em expressões regulares usadas para comparar a descrição dos atributos dos valores solicitados pelo cliente. Eles também podem ser usados para impor condições adicionais ou limitar os valores selecionados. No caso de requisições históricas os predicados são usados para gerar consultas SQL (*Structured Query Language*) para o banco de dados local. A requisição de subscrição irá criar uma *thread* dedicada para cada cliente, e irá executar os testes de comparação para todos os predicados submetidos pelo cliente. A mesma *thread* é responsável por enviar os resultados selecionados de volta para o cliente como objetos serializados e compactados. Há uma *thread* independente por cliente sempre enviando a informação que ele necessita, de maneira confiável e rápida, não sendo afetada por erros que podem ocorrer com outros clientes. No caso de problemas de comunicação estas *threads* vão tentar restabelecer a conexão ou liberar a subscrição de um cliente ou serviço que não está mais ativo.

A requisição de dados monitorados com o mecanismo de predicados também é possível usando WSDL (*Web Services Description Language*) /SOAP (*Simple Object Access Protocol*) por clientes e serviços escritos em outras linguagens. As classes para os predicados e os métodos usados são descritas em WSDL e qualquer cliente pode criar dinamicamente e instanciar os objetos necessários para a comunicação.

O Serviço Proxy para Clientes

A arquitetura provê um serviço de *Proxy* que é utilizado pelos clientes para conectar serviços diferentes (o serviço de *Proxy* também é um serviço JINI). Um modelo de descobrimento mútuo entre serviço e *proxy* é usado para detectar se um certo serviço está sendo executado sob um *firewall* ou NAT (*Network Address Translation*). Neste caso o serviço inicia as conexões para todos os *proxies* disponíveis para uma comunidade e auto registra nos LUSs. Qualquer cliente pode agora interagir com qualquer serviço via os serviços do Proxy, que faz uma multiplexação dos dados subscritos por múltiplos clientes. São executados múltiplos serviços de Proxy para redundância e balanceamento de carga entre os clientes.

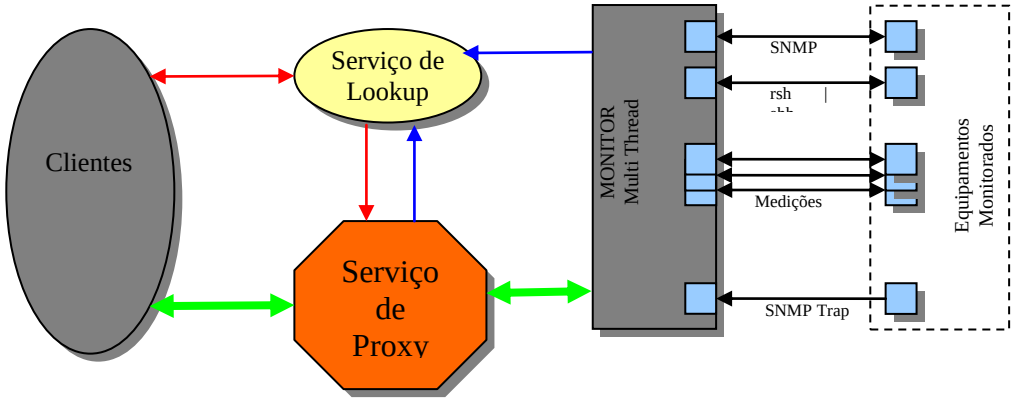

Figura 3.8: Interação dos clientes com os serviços utilizando os serviços de Proxy

Repositórios e Pseudo Clientes

Os pseudo clientes utilizam os LUSs para encontrar todos os serviços MonALISA ativos para um conjunto de grupos específicos e subscrever estes serviços com uma lista de predicados e filtros, como apresentado pela figura 3.9.

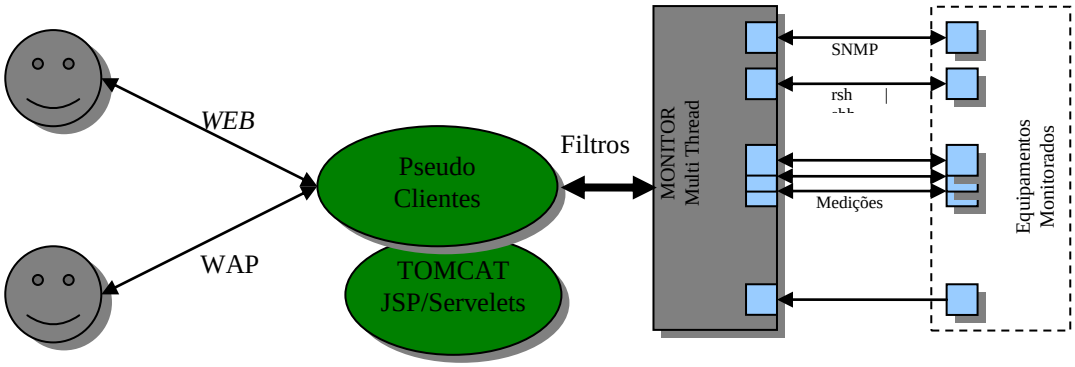

Figura 3.9: Acesso de pseudo clientes à arquitetura MonALISA

Os predicados e filtros especificam as informações que os pseudoclientes desejam coletar de todos os serviços. Um pseudocliente armazena todos os valores recebidos dos serviços que estão rodando em um banco de dados local (MySQL) e usa procedimentos escritos em Java para compactar dados antigos. Uma *servlet* baseada em Tomcat é usada para prover uma maneira flexível de apresentar dados globais e construir gráficos

rapidamente com os dados atuais ou históricos, de acordo com a demanda. Outras *servlets* dedicadas são usadas para gerar páginas WAP (*Wireless Access Protocol*) contendo as mesmas informações para usuários de telefones móveis.

Administração dos Serviços

A arquitetura MonALISA usa um mecanismo seguro SSL (*Secure Sockets Layer*) com certificados X.509, para permitir a configuração dinâmica dos elementos e para o suporte de serviços de nível mais alto visando administrar o conjunto de aplicações distribuídas ou para otimizar o *workflow*. A administração é feita por meio de uma GUI (*Graphics User Interface*) dedicada, que pode ser iniciada de qualquer cliente, pois quando o administrador carrega sua chave privada para o cliente, ele automaticamente ganha direitos administrativos sobre os serviços que importarem seu certificado.

3.5.2 Gerenciamento MonALISA

A infra-estrutura MonALISA é aquela que mais funções de gerenciamento possui, graças em grande parte à tecnologia usada na sua implementação. A MonaALISA implementa funções nas áreas funcionais de Falhas, Configuração e Segurança. No gerenciamento de falhas, por exemplo, todos os serviços possuem monitoração e recuperação automática de falhas. Esta monitoração é feito de forma automática pela tecnologia Jini, usada na implementação.

No gerenciamento de configuração tem-se o mecanismo de atualização automática, o qual periodicamente verifica por atualizações nos componentes do sistema. O processo de atualização é feito na inicialização do sistema e periodicamente pelo *cron* do sistema operacional.

No gerenciamento de segurança, a MonALISA prevê o uso de SSL e certificados X.509 para conexões seguras e autenticação.

3.6 NIMI

A infra-estrutura de medição NIMI (*National Internet Measurement Infrastructure)* foi apoiada pela NSF (*National Science Foundation)* e consiste num conjunto dedicado de servidores de medição (chamados de NIMI *Probe*). A meta chave do projeto NIMI é a escalabilidade, potencialmente milhares de NIMI *probes* em uma simples infra-estrutura. Aumentando o número de *probes* aumenta o número de caminhos mensuráveis, na proporção N^2.

A solicitação de uma medida (que pode ser composta de *n* outras medidas individuais) é inserida na infra-estrutura por meio de um Cliente de Medição – MC (*Measurement Client*), o qual pode estar rodando em uma máquina comum. À medida que os resultados vão sendo coletados, o foco da medição pode ser alterado para métricas diferentes ou refinado pelo MC.

Um aspecto importante nesta infra-estrutura é que cada NIMI *Probe* deve se reportar a um CPOC (*Configuration Point Of Contact)* que foi configurado pelo dono do NIMI Probe. Normalmente, haverá um CPOC por domínio administrativo, isso evita que os problemas ocorridos ultrapassem um domínio limitado. A arquitetura NIMI pode facilmente delegar parte dos serviços de medição de uma *probe*, oferecendo um controle mais severo, quando necessário, sobre os serviços delegados.

A infra-estrutura NIMI foi projetada para ser segura. Ela usa autenticação e criptografia em todas as comunicações entre os componentes da infra-estrutura. Cada *probe* é configurada pelo seu CPOC para autorizar um conjunto particular de operações por credencial. Isso permite o controle total das ações de uma credencial pelo dono da *probe*.

Para permitir uma maior flexibilidade na forma de efetuar as monitorações, o NIMI foi projetado para trabalhar com qualquer ferramenta de monitoração. Estas ferramentas são empacotas em módulos para serem utilizadas na infra-estrutura. Segundo PAXSON (1998), O NIMI não é uma ferramenta de monitoração, mas um sistema de comando e controle para gerenciamento de ferramentas de medição.

3.6.1 Arquitetura NIMI

A arquitetura NIMI foi dividida em um conjunto de ferramentas e componentes com funcionalidades distintas.

O componente mais engenhoso é o *NIMI probe*, ou *nimid*, cuja tarefa exclusiva é processar e armazenar os pedidos de medição quando eles chegam, executá-los no momento apropriado, empacotar, enviar os resultados para o Cliente de Análise de Dados - DAC (*Data Analysis Client*) e apagar os resultados quando solicitado. Além disso, o NIMI Probe é subdividido em dois processos distintos, o *nimid* e *scheduled*. O *nimid* é responsável pela autenticação, codificação e autorização de pedidos e transporte de resultados. O *scheduled* simplesmente enfileira, executa e empacota (para transportar) os pedidos de medição.

O próximo componente principal é o CPOC. O CPOC age como o ponto administrativo de contato para um conjunto de NIMI Probe, dentro da zona de controle do CPOC (domínio administrativo). O trabalho do CPOC é simplesmente prover as políticas iniciais para cada NIMI e com o passar do tempo, prover as atualizações das políticas. O CPOC também agirá como dispensador de chaves públicas e eventualmente um repositório de *software* para componentes e módulos de medição do NIMI.

O MC é o componente que é operado pelo usuário final. Ele é responsável por agrupar as informações de uma ou mais medições solicitadas pelo usuário e enviar para o NIMI Probe, para que este efetue a solicitação.

O DAC funciona como um repositório e um pós-processador dos dados devolvidos pelo NIMI Probe em resposta a uma medição. Este componente pode ser executado pelo MC para coletar resultados imediatos ou ser executado como um processo para coletar medidas contínuas.

A figura 3.10 descreve o fluxo de mensagens entre componentes de NIMI.

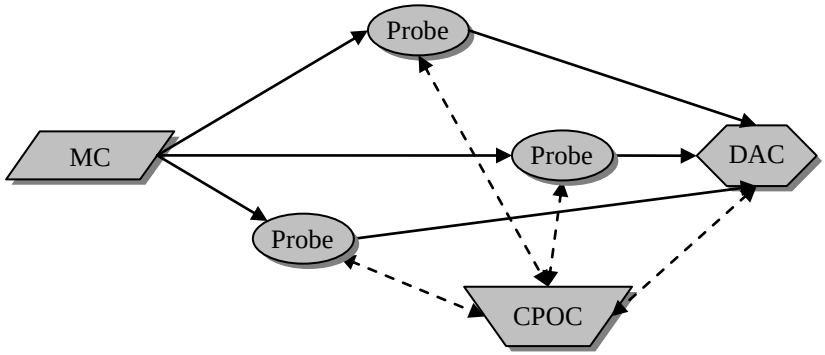

Figura 3.10: Fluxo de Mensagens entre os componente da Arquitetura NIMI.

Módulos de Medição

A flexibilidade da arquitetura foi conseguida tratando as ferramentas de medição como módulos que podem ser adicionados no NIMI. O NIMI não tem nenhum conhecimento das ferramentas de medição, quando uma NIMI Probe recebe uma requisição ela simplesmente executa a ferramenta que foi nomeada na requisição.

A execução das ferramentas é feita por meio de um *script*. O uso de scripts traz três benefícios. Primeiro, empacota os resultados gerados das ferramentas de medição de forma unificada para um processamento mais fácil do DAC. Segundo, permite uma interface comum (API – *Application Programming Interface*) entre o *scheduler* e as ferramentas de medição. E terceiro, o *script* não precisa ser *setuid root* e assim permite mais flexibilidade, pois não necessita um administrador de sistema com privilégios de superusuário para implementar uma mudança no *script*.

Para adicionar uma nova ferramenta de medição, como um módulo, é necessário simplesmente gerar um *script* e propagá-lo junto com a ferramenta para todos os NIMI Probes (isso é feito via SSH).

Política de Controle

O NIMI suporta diversas políticas de controle por meio do uso de Listas de Controle de Acesso (ACL – Access Control List) que abriga um único conjunto de regras para um determinado componente NIMI.

Cada componente NIMI tem sua própria ACL, que é uma tabela onde as colunas representam requisições ou tarefas e as linhas apresentam as credenciais. A interseção de uma requisição e uma credencial é um valor boleano. Eventualmente, um script pode ser aplicado nos argumentos de uma requisição para gerar um valor boleano.

Um NIMI Probe recebe sua ACL do CPOC no momento de sua inicialização. O CPOC pode também delegar alguma administração de ACL para outros componentes NIMI para, por exemplo, permitir que um MC inclua alguma ACL para outro MC. A figura 3.11 apresenta o modelo de uma tabela formando uma ACL.

	ACL_ADD	ACL_DEL	TEST_ADD	TESTE_DEL	DATA_GET	DATA_DEL	SAND_BOX	INHERITENCE
CPOC_KEY	T	T	T	T	T	T	F	F
MC_A_KEY	F	F	T	T	T	T	F	F
MC_B_KEY	T	T	T	T	T	T	mc_b	F
MC_B_KEY_2	F	F	TReno	T	F	F	Mc_b2	MC_B_KEY
MC_C_KEY	T	T	<512B	T	T	F	mc_c	F
MC_C_KEY2	F	F	Zing	F	F	F	Mc_zc2	MC_C_KEY

Figura 3.11: Exemplo de uma tabela de ACL.

Controle e Gerenciamento Local

Para obter maior segurança na execução das solicitações, os arquivos de ACL, os pedidos e os resultados são armazenados no sistema de arquivos local.

A tabela de ACL para um componente NIMI existe como um arquivo de texto plano dentro do subdiretório de “acl/”. Se o administrador de sistema para um *nimid* local resolver remover uma política particular ele simplesmente remove uma linha correspondente à tupla que identifica a Acl.

Além disso, quando chega uma requisição, o *Schedule* move o pedido para um arquivo de texto localizado no subdiretório "pendente/". Quando chega o tempo para a execução o pedido é movido para um outro arquivo no subdiretório "ativo/".

Quando o pedido de uma medição é finalizado tanto o pedido e como os resultados são colocados em um arquivo *tar* e este é movido para o subdiretório “dock/”, aguardando para ser enviado ao DAC apropriado. Após a entrega do arquivo para o DAC pelo *nimid*, este é movido para o subdiretório “completed/”.

A qualquer instante o status de qualquer medição pode ser obtido simplesmente pela observação do sistema de arquivos.

Segurança e Comunicação

Para efetuar a comunicação entre os componentes do NIMI foi desenvolvido um protocolo de troca de mensagens.

As mensagens usadas no NIMI são codificadas por um par de chaves públicas e privadas e passam entre os componentes do NIMI via TCP/IP (*Transmission Control Protocol/Internet Protocol*). Uma mensagem consiste em um cabeçalho que contém as informações necessárias para decifrar o corpo da mensagem (com a chave apropriada) e um corpo de mensagem codificado.

O corpo da mensagem consiste em um ou mais blocos de mensagem. Cada bloco ou tipo de bloco mapeia para uma tarefa ou requisição que um componente NIMI pode aceitar, assim como possui os dados necessários para completar a requisição.

Quando uma mensagem é recebida, o componente NIMI decifrará, autenticará a mensagem e extrairá o primeiro bloco do corpo de mensagem. O componente NIMI usará o tipo de bloco do primeiro bloco (junto com quaisquer dados dentro do bloco) e pesquisa na tabela de ACL para autorizar o pedido (tipo de bloco). Se a autorização tiver sucesso, o bloco será passado para uma função de processamento. A figura 3.12 apresenta um exemplo de um bloco do tipo TEST_ADD.

```
TEST_ADD Handle-id treno-exec nimi1.psc.edu 200001050000 dac.psc.edu "-q
nimi.washington.edu"
```

Figura 3.12: Exemplo de um tipo de bloco para comunicação NIMI

3.6.2 Gerenciamento NIMI

Na infra-estrutura NIMI, a área funcional de gerenciamento de falhas efetua o controle da execução dos testes. Todos os estados possíveis de um pedido de teste, desde o seu agendamento até a conclusão é determinado pela sua localização no sistema de arquivos.

Na área funcional de gerenciamento de segurança, a infra-estrutura NIMI possui um controle sobre usuários e testes. Todos os testes são autenticados e os usuários submetidos a uma lista de controle de acesso.

3.7 NWS

O NWS (*Network Weather Service) - University of Califórnia (San Diego)* é um sistema distribuído que periodicamente monitora e dinamicamente prevê o desempenho de vários recursos computacionais e de rede. O serviço opera um conjunto distribuído de sensores de desempenho (monitor de rede, monitor de CPU) dos quais coleta as condições instantâneas e, usando modelos numéricos, gera previsões das condições da rede para um determinado espaço de tempo. Esta funcionalidade foi pensada como sendo análoga à "previsão do tempo" e por este motivo o sistema herda seu nome. O NWS foi desenvolvido para facilitar o uso de escalonadores dinâmicos em grades computacionais (WOLSKI, 1998).

3.7.1 Arquitetura NWS

A construção do NWS envolve quatro processos diferentes (WOLSKI, 1999):

Estado Persistente – processo que armazena e recupera medições de forma persistente, capaz de recuperação em caso de falha do processo em memória.

Servidor de Nomes – implementa as capacidades de diretório, usado para ligar processos e nomes de dados com informações de baixo nível, por exemplo, número de porta TCP e endereços IP.

Sensor – processo para obter a medida de desempenho de um recurso específico.

Previsão – processo que gera e apresenta a previsão de desempenho para um intervalo de tempo específico, para cada recurso.

Cada componente é apresentado na figura 3.13, e pode se comunicar com outro processo apenas através de mensagens. Isso permite que as implementações sejam melhoradas ou substituídas por outras implementações padronizadas, quando disponíveis.

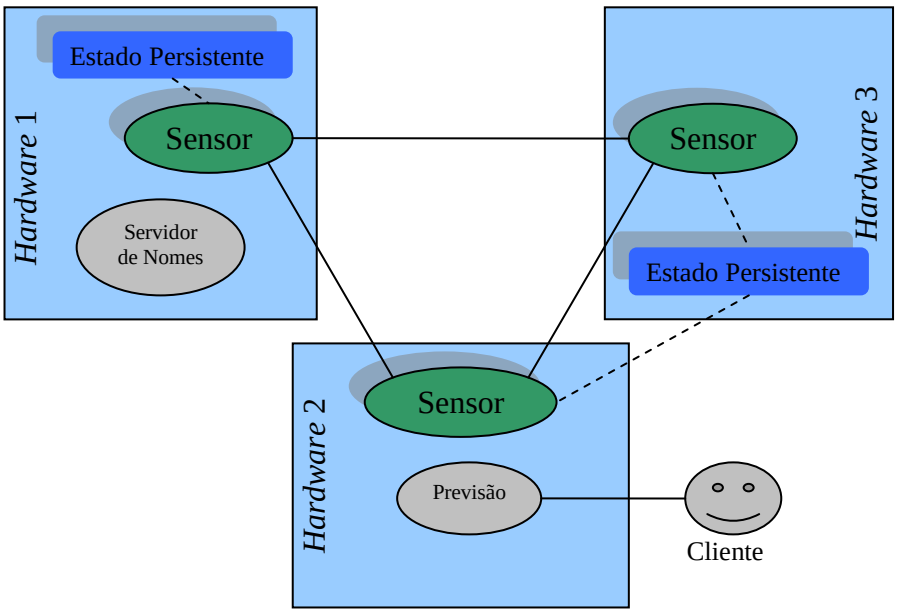

Figura 3.13: Arquitetura NWS

3.7.2 Metodologia

A implementação atual suporta sensores para medir: fração de tempo de CPU usada para um novo processo, tempo de uma conexão TCP, atraso TCP fim-a-fim, e largura de banda TCP fim-a-fim. Para este trabalho, será detalhado apenas o sensor de rede, responsável pela coleta das informações de nosso interesse.

Os sensores de rede medem o desempenho da rede, com as seguintes características: tempo de ida-e-volta de mensagens pequenas, processamento de mensagens grandes, tempo de conexão e desconexão do *socket* TCP.

As mensagens pequenas consistem em 4bytes TCP que são enviados por uma conexão previamente estabelecida. Assim, o tempo computado, entre o envio e a resposta do sensor de destino, exclui o estabelecimento e encerramento da conexão.

A transferência de mensagens grande é computada entre a transferência da mensagem, usando TCP, e a chegada do reconhecimento enviado pelo receptor. O tamanho da mensagem, o tamanho do *buffers* de envio e recepção do *socket* e das chamadas *send()* e *recv()* são parametrizados para cada sensor. Empiricamente os valores definidos foram respectivamente, 64k, 32k e 16kBytes. A figura 3.14 detalha os experimentos de desempenho efetuados pelo NWS.

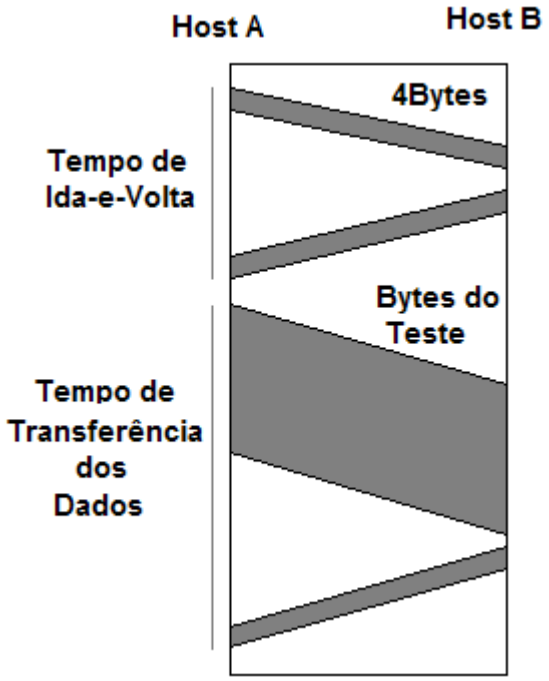

Figura 3.14: NWS – Experimentos de atraso e largura de banda

Para tornar os dados dos experimentos disponíveis para o processo de predição, os sensores contatam o processo persistente para armazenar a informação. A localização do processo persistente que um sensor irá usar para cada medida é especificada quando o sensor é configurado. Quando o sensor é iniciado, ele registra a localização no processo persistente, que armazena seus dados de medição, com o serviço de nomes, permitindo a localização de uma medida por nome.

3.7.3 Gerenciamento NWS

O gerenciamento previsto nesta infra-estrutura diz respeito à periodicidade com que os testes são efetuados. Um servidor somente pode efetuar um teste quanto possuir um

token. A velocidade de repasse desse *token* é definida pelo administrador de forma que não sobrecarregue o sistema, logo podemos incluir essa função no gerenciamento de performance. Nas outras áreas funcionais não é prevista nenhuma outra função.

3.8 IEPM - PingER

A arquitetura PingER , foi proposta em meados de 1996 pelo IEPM - *Internet End-to-end Performance Monitoring*, no *SLAC – Stanford Linear Accelerator Center*, com a finalidade de permitir que diversas instituições norte americanas pudessem avaliar a qualidade das conexões Internet entre elas. Esta arquitetura foi denominada PingER, significando *Ping End-to-End Reporting*. Como indicado no próprio nome, o PingER é baseado no tradicional programa *ping*, largamente conhecido pelos administradores de rede. Um *ping* é o envio de um pacote *ICMP echo request* para um equipamento remoto que deve responder com um pacote *ICMP echo reply*. Opcionalmente, podem ser enviados bits adicionais (*payload*) no pacote e aguardar seu retorno. O tempo de ida e volta dos pacotes (*rtt*) é contabilizado.

3.8.1 Arquitetura PingER

A arquitetura PingER, apresentada na figura 3.15, define três tipos de equipamentos que interagem para efetuar a monitoração. Eles são (MATTHEWS, 2000): equipamento de monitoração (*Monitoring*), equipamento remoto (*Remote*) que está sendo monitorado e equipamento para armazenamento e análise (*Archive*).

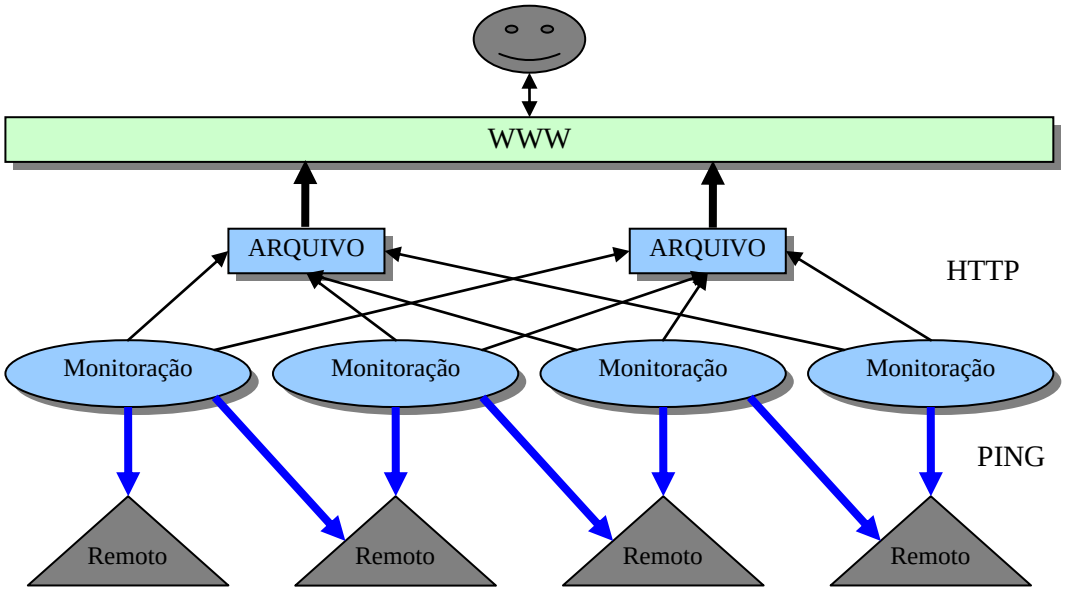

Figura 3.15: Arquitetura PingER

Cada equipamento de monitoração configura quais os equipamentos remotos que deseja monitorar. A cada 30 minutos o equipamento de monitoração envia para o equipamento remoto, onze *pings* com intervalos de 1 segundo, cada um com 100Bytes adicionais. Posteriormente, envia mais dez *pings*, também intercalados em 1 segundo, com 1000Bytes adicionais. O primeiro *ping* é descartado, pois é presumido que este seja o mais lento, pois sofre atrasos para preencher os *caches* do sistema, como DNS (*Domain Name Server*), por exemplo. São armazenados localmente os tempos de ida e volta (rtt) do pacote, pacotes perdidos, fora de ordem e ou duplicados. Diariamente estes dados são transferidos para o equipamento de armazenagem e análise. Estes equipamentos também provêem ferramentas para análise interativa e apresentação dos

dados na *web*. Estas ferramentas incluem navegação, organização e agregação dos dados por tempo, país, região, afinidade ou métrica.

3.8.2 Metodologia de Monitoração

O PingER define cinco métricas para estimar o desempenho da rede. As métricas são (MATTHEWS, 2000): Perda de Pacotes (*Packet Loss*), Tempo de Ida e Volta (*Round-Trip Time*), Inatingibilidade (Unreachability), Tranqüilidade (*Quiescence*), Imprevisibilidade (*Unpredictability*).

Perda de Pacotes – Muitos pacotes são enfileirados nos *buffers* dos roteadores para posterior processamento, caso o *buffer* esteja cheio o pacote será descartado. O número de pacotes perdidos pode ser uma boa indicação que esta parte da conexão está congestionada. O desempenho de uma aplicação usando TCP se deteriora significativamente quando a taxa de perda de pacotes ultrapassa 3%, porém o efeito sentido pelo usuário varia de acordo com a aplicação. Por exemplo, uma videoconferência pode ser comprometida com uma moderada taxa de perda, em contrapartida um e-mail pouco é afetado mesmo com altas taxas de perdas.

Tempo de Ida-e-Volta – É o tempo gasto para o pacote sair do transmissor, passar pelas filas nos roteadores, chegar no destino e retornar. O valor mínimo do tempo de ida-e-volta é determinado pelas leis da física, ou seja, mesmo que o pacote não sofra atrasos nas filas dos roteadores ele terá um tempo mínimo imposto pelo meio de transmissão em relação à distância. Quando o pacote é enfileirado nos roteadores ele sofre um atraso adicional ao tempo mínimo de transmissão, este atraso adicional pode ser interpretado como um congestionamento. Uma alteração nos valores mínimos pode indicar uma mudança de rota.

Inatingibilidade – Se o nodo remoto não responder a nenhum dos pacotes enviados ele é considerado inatingível. Para uma análise mais precisa do desempenho da rede é necessário identificar se o motivo da inatingibilidade é ou não, devido ao desempenho da rede. As respostas não foram recebidas porque os computadores falham, ficam indisponíveis ou por alguma outra razão de desempenho da rede? A identificação da Inatingibilidade é extremamente difícil de ser programada. Além disso, pode ser utilizada variação estatística para determinar se um nodo está ou não inatingível, digamos no caso de que o número de pacotes perdidos ultrapasse 90% e desconsiderar se as perdas forem inferiores a 1%. Na prática permanece o julgamento do analista de determinar se o nodo é verdadeiramente inalcançável devido a problemas de rede.

Tranqüilidade – Se todos os pacotes enviados ao nodo remoto receber um retorno, a rede é considerada tranqüila ou não ocupada. A não ocorrência de pacotes perdidos pode ser uma indicação do uso da rede. Por exemplo, uma rede que esteja ocupada durante as oito horas diárias de trabalho e tranqüila nas demais possui uma percentual de tranqüilidade de 75%. Se a rede stiver ocupada durante todo o dia é considerada “pobre” e provavelmente necessita ser ampliada.

Imprevisibilidade – é derivada de um cálculo baseado na variabilidade do número de pacotes perdidos e no tempo de ida e volta. O valor derivado é a medida de estabilidade e consistência do enlace. Um enlace com baixo desempenho pode ter baixa imprevisibilidade se os valores de perda de pacotes e tempo de ida e volta não sofrerem variações drásticas. Porém, um enlace de alto desempenho e com grandes

variações nas taxa de perda de pacotes e de tempo de ida-e-volta será caracterizado como imprevisível.

3.8.3 Gerenciamento PingER

A infra-estrutura proposta pelo projeto PingER é a de mais simples implementação. Ela utiliza uma variação do programa ping (largamente conhecido) e simplesmente consolida os resultados em algumas métricas. O PingER não define nenhuma função de gerenciamento da infra-estrutura.

3.9 E2E piPEs

A iniciativa E2Epi da Internet2 [2] (E2Epi *End-to-End Performance Initiative*) (INTERNET2, 2004), vem atuando na compreensão e solução dos problemas de desempenho fim-a-fim. O principal projeto do grupo é chamado PES (*Performance Environment System)*, conhecido comumente como piPEs (*Performance Initiative Performance Environment System*), que é um *framework* para avaliação de desempenho baseado em medições realizadas por diversas ferramentas (E2EPI, 2004-c).

O principal objetivo do projeto piPEs é permitir a usuários finais e operadores de rede determinar as capacidades de desempenho fim-a-fim (E2E), localizar problemas de desempenho e contatar a pessoa correta para solução do problema (E2EPI, 2004-c).

3.9.1 Arquitetura piPES

A arquitetura piPEs considera que o caminho fim-a-fim é um conjunto de segmentos compostos de equipamentos de nível 3, ou seja, roteadores e os equipamentos finais, como mostra a figura 3.16. Em cada segmento supõe-se a existência de um PMD (Ponto de Medição de Desempenho), sendo que o conjunto de PMDs tornará possíveis testes de performance entre quaisquer dois pontos da rede.

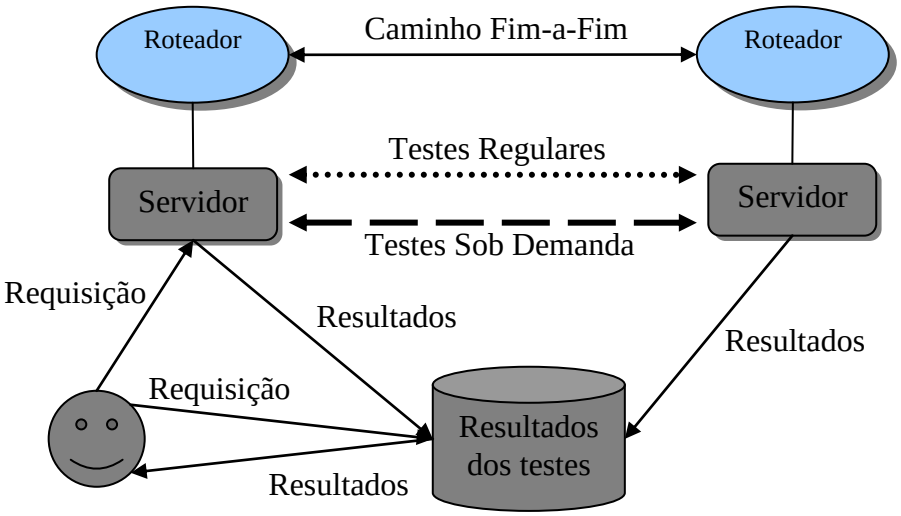

Figura 3.16: Teste entre dois PMDs, uma visão parcial da rede

[2] O grupo Internet2 passou a trabalhar posteriormente em conjunto com a GÉANT2 no desenvolvimento do GFD e do perfSONAR

Com o objetivo de otimizar o uso da rede, evitando que os próprios testes de desempenho a congestionem, a arquitetura prevê uma série de testes regulares entre os principais pontos (PMD) e os seus resultados são armazenados em bases de dados disponíveis aos usuários. Nestes pontos, o usuário não necessita efetuar um novo teste, simplesmente faz uma consulta à base de dados. Os pontos não atingidos por testes regulares são complementados por testes sob demanda. Na figura 3.17, podemos ver os diferentes tipos de testes que podem ser efetuados.

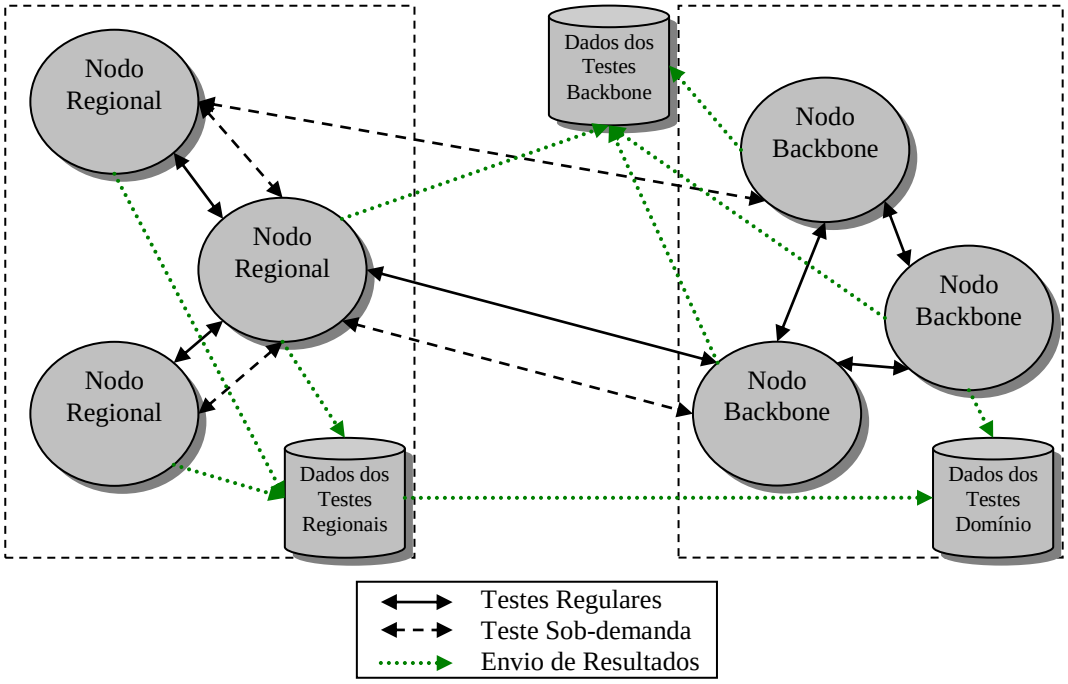

Figura 3.17: Visão dos cenários de testes

A figura 3.18 apresenta os *softwares* que compõem a arquitetura piPEs, uma breve descrição de cada módulo está apresentada abaixo:

- **Detecção (Detect)** – este módulo inclui as ferramentas que devem detectar os problemas, usando os dados obtidos pelas medições e apresentá-los em um formato amigável;
- **Interface (Interface)** – Os resultados e os dados armazenados devem estar disponíveis de forma pública, por meio de acessos padronizados;
- **Autorização (Authorize)** – Como o acesso à infra-estrutura de medições deve ser público, este módulo deve fornecer os mecanismos de autenticação e autorização para o uso das ferramentas;
- **Agendamento dos Testes (Schedule)** – Permite a programação da execução dos testes. Usado para avaliar o comportamento da rede em dia e horário específico, normalmente relacionado a algum evento na rede.
- **Testes (Test)** – Módulo que agrega as ferramentas de medição, instaladas nos PMDs. As ferramentas OWAMP e BWCTL já estão sendo utilizadas (totalmente desenvolvidas), outras estão em fase experimental e ainda outras

surgem como propostas. Este modelo não é fechado, ele permite a integração e inclusão de novas ferramentas.

- **Armazenamento (Store)** – as ferramentas de testes devem armazenar os dados provenientes das medições em um SGBD (Sistema Gerenciador de Banco de Dados), para posterior recuperação.

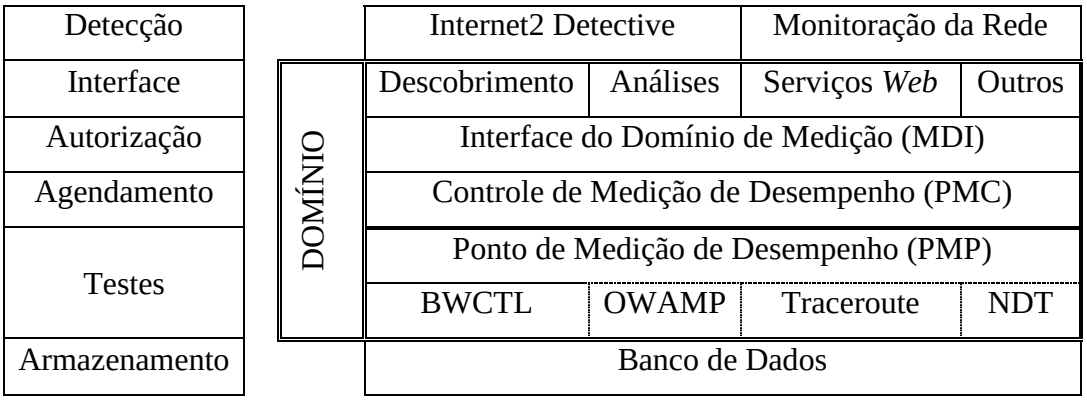

Figura 3.18: Componentes de *software* da arquitetura piPEs

A proposta desta arquitetura não está completamente definida, alguns módulos estão previstos, mas devido a mudança de foco do projeto (integração com o perfSonar) não possuem implementação.

3.9.2 Ferramentas de Testes

OWAMP

A ferramenta OWAMP (E2EPI, 2004-b) é a implementação do protocolo de mesmo nome *One-Way Active Measurement*, que foi padronizado pela RFC 4656 (SHALUNOV, 2006). O protocolo OWAMP permite estimar os atrasos e as perdas de pacotes em ambos os sentidos da comunicação (one-way).

O protocolo é composto de duas partes, apresentadas na figura 3.19:

- OWAMP-Control
- OWAMP-Test

O OWAMP-Control é usado para iniciar a conexão, definir a configuração e a transferência dos resultados. O seu funcionamento é baseado no modelo tradicional cliente servidor, utilizando o protocolo TCP. Nesta conexão, ambos os lados negociam como será a execução do teste.

O OWAMP-Test é responsável pelos testes propriamente dito, utilizando o protocolo UDP. Os parâmetros como tamanho do pacote e tempo entre os envios dos pacotes podem ser configurados, simulando uma possível aplicação.

As métricas são avaliadas observando a diferença entre a marca de tempo inserida pelo transmissor e o tempo na chegada da mensagem ao receptor. O aplicativo exige que o relógio esteja sincronizado via NTP (*Network Time Protocol*) ou que o *clock* seja gerado localmente por um equipamento confiável como um GPS, por exemplo. A precisão depende da fonte de sincronização.

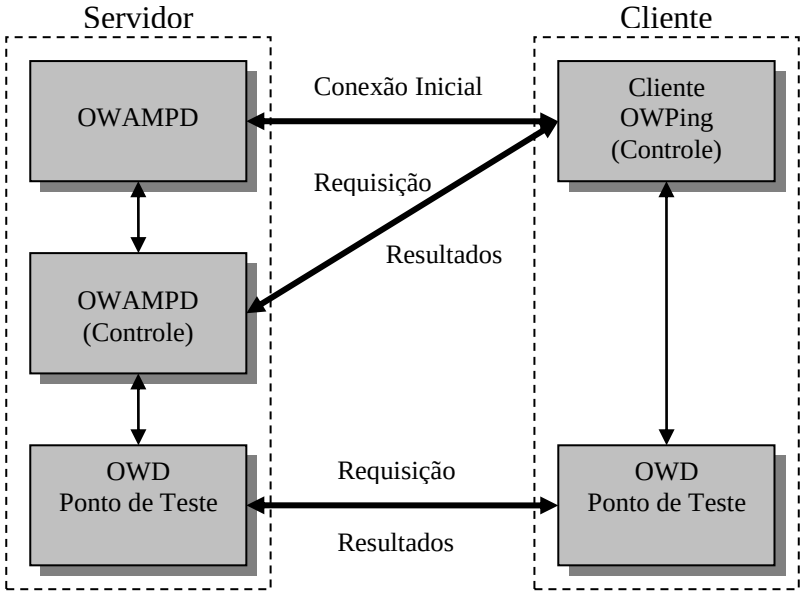

Figura 3.19: Arquitetura OWAMP

BWCTL

O BWCTL (E2EPI, 2004-a) é uma aplicação cliente, trabalhando em linha de comando, que monitora e agenda execuções do programa IPERF (NLANR, 2004). O programa IPERF permite determinar a largura de banda disponível entre dois *hosts*, no caso da arquitetura piPEs, dois PMDs.

O BWCTL trabalha de forma integrada ao IPERF, evitando a execução de múltiplas instâncias do IPERF, casos em que um teste pode interferir no outro.

O cliente BWCTL pode estabelecer testes também entre dois servidores, sendo que um deles pode ou não estar localizado na mesma máquina do cliente, como mostra a figura 3.20.

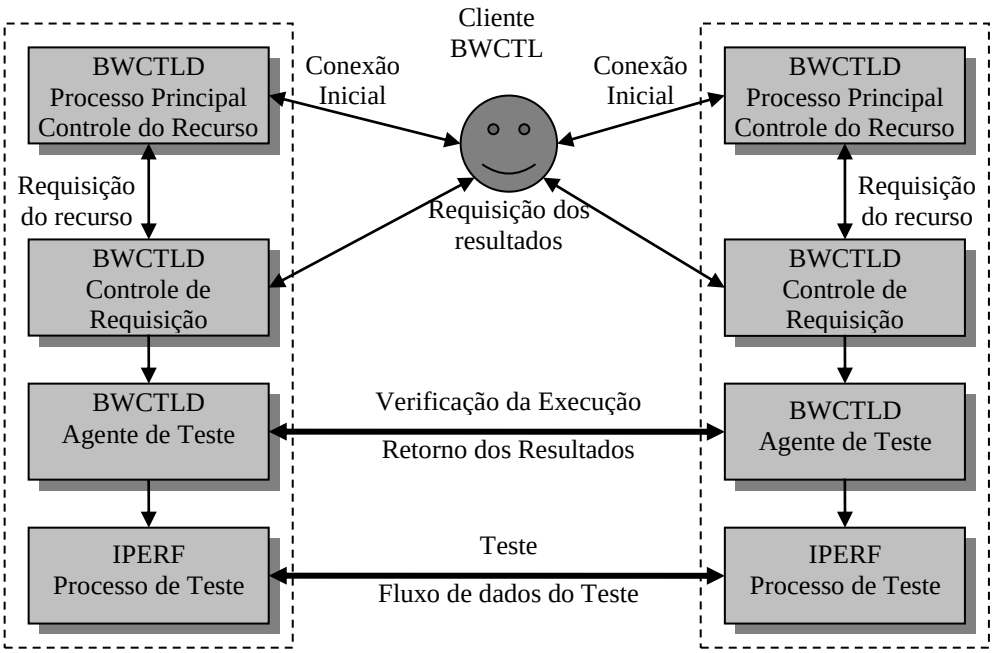

Figura 3.20: Arquitetura BWCTL

3.9.3 Gerenciamento piPEs

A infra-estrutura de medição piPEs teve seu foco direcionado para o desenvolvimento conjunto do perfSonar. As ferramentas externas OWAMP e BWCTL permitem o uso de autenticação entre os testes, além de permitirem a configuração de limites de utilização. A infra-estrutura de forma integral não prevê um módulo para o gerenciamento da própria infra-estrutura.

3.10 RIPE NCC - TTM

O *RIPE-NCC Réseaux IP Europeen - Network Coordination Center* foi fundado em 1992 como uma organização sem fins lucrativos, com o objetivo de prover serviços para o benefício das operadoras de redes baseadas em IP, na Europa e arredores (GEORGATOS, 2001).

Em 1997 os membros do RIPE NCC decidiram iniciar pesquisas sobre medição de desempenho entre os participantes. Este projeto foi chamado de *Test Traffic*, mais tarde denominado de TTM (*Test Traffic Measurement).* O principal objetivo do TTM é efetuar medições de desempenho de acordo com padrões bem definidos por grupos de padronização, como o IETF.

Os objetivos adicionais do projeto são (GEORGATOS, 2001):

Medição de uma via: cada vez mais, as rotas entre os equipamentos são assimétricas (o caminho dos pacotes que viajam de A para B é diferente de B para A), logo é necessário mensurar o desempenho separadamente em ambos os sentidos.

Infra-estrutura de medição dedicada: não utilizar nenhuma infra-estrutura previamente existente e definir um equipamento, como um *black box*, evitando alterações pelo *site* que está utilizando. Este equipamento foi chamado de "*test-box"*.

Medição ativa: o tráfego deve ser gerado pelo próprio equipamento de medição, não confiando em tráfegos de outras fontes. Além disso, permite um maior controle sobre o formato dos dados usados na medição.

Tráfego de teste "real": O tráfego de teste gerado dever ser o mais próximo do tráfego real. Por exemplo, as técnicas baseadas em ICMP, como o *ping*, possuem tratamento diferente nos roteadores do que tráfego TCP ou UDP.

Somente redes entre os provedores: as técnicas definidas são previstas para utilização apenas na rede entre os provedores.

Métricas padronizadas: as medições devem obedecer a um conjunto de métricas bem definidas e cientificamente defensáveis.

3.10.1 Arquitetura RIPE NCC – TTM

A arquitetura RIPE NCC – TTM é composta dos equipamentos de medição, chamados de *test-box* que são instalados nos provedores participantes e de um equipamento central instalado no RIPE NCC. O *test-box* é um equipamento do tipo IBM PC equipado com uma placa de GPS (para prover a precisão desejada nas marcas de tempo inseridas nos pacotes) e sistema operacional FreeBSD.

Para efetuar as medidas de atraso e perda entre dois provedores, os dispositivos de medição (*test-box)* devem ser instalados o mais próximo possível dos roteadores de borda. A figura 3.21 apresenta uma visão geral da arquitetura RIPE NCC.

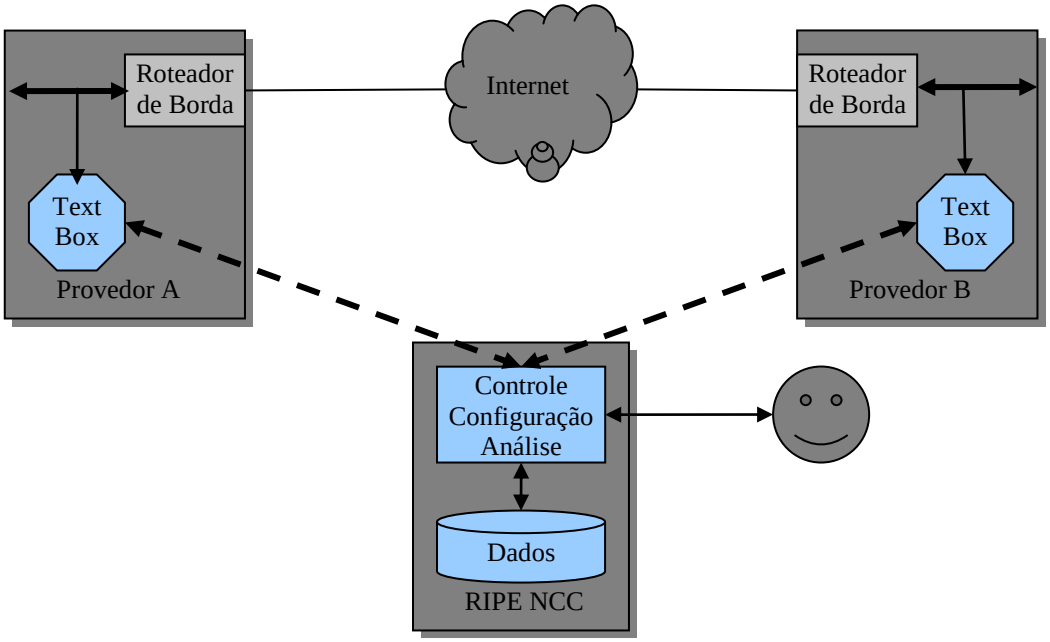

Figura 3.21: Visão geral da arquitetura RIPE-NCC

Os dados sobre as medições ficam armazenados localmente em cada *test-box* de forma segmentada, ou seja, o *test-box* que enviou os pacotes de teste tem apenas informações dos pacotes que foram enviados e o *test-box* receptor tem informações apenas dos pacotes que chegaram até ele. Neste caso, para obter todas as informações necessárias sobre o resultado de um teste, os dados são coletados e armazenados diariamente pelos equipamentos da central RIPE NCC. Os equipamentos centrais também controlam o experimento, enviando os dados de configuração necessários para os *test-boxes* (por exemplo, a freqüência do teste), iniciam e encerram os processos de teste e atuam também como um repositório dos *softwares* de medição.

3.10.2 Metodologia

Considerando uma medição entre os provedores A e B, figura 3.21, uma marca de tempo é coletada quando o pacote deixa o *test-box* A e também quando ele chega no *test-box* B. Como os *test-boxes* são providos de GPS, para sincronização, basta subtrair as marcas de tempo para determinarmos o atraso. O número de pacotes enviados e o número de pacotes recebidos determinam a quantidade de pacotes perdidos. Para determinar a rota que o pacote percorreu é utilizado o programa *traceroute*.

Diariamente um processo de coleta e análise é iniciado na central. Este processo efetua a coleta dos dados de todos os *test-boxes,* posteriormente os dados são eliminados dos *test-boxes.* Após a coleta dos dados, efetua-se a análise das informações, comparando os pacotes enviados com os pacotes recebidos entre dois provedores calculando-se o atraso e a perda. Depois de terminado o cálculo, os dados são formatados e disponibilizados.

3.10.3 Gerenciamento RIPE

As funções de gerenciamento presentes nessa arquitetura fazem parte das áreas funcionais de falhas e de configuração. No gerenciamento de falhas a infra-estrutura prevê a monitoração de todos os processos presentes no "test-box", incluindo os processos necessários para a manutenção, como *ssh* e *login*. No gerenciamento de configuração, a infra-estrutura prevê que a instalação de um novo sistema seja efetuada por meio de um disco específico contendo todos os arquivos e as configurações necessárias.

3.11 SCRIPTROUTE

O *Scriptroute – University of Washington* é um sistema que permite a usuários comuns da Internet efetuarem medições na rede. Este sistema procura combinar a flexibilidade encontrada em sistemas de medição dedicados com a popularidade dos servidores de *traceroute* baseado em *Web*. Para acessar o *Scriptroute* os clientes utilizam o DNS para encontrar os servidores de medição e então submetem um *script* de medição que será executado no servidor, em um ambiente com recursos limitados por questões de segurança, chamado de *sandbox* (caixa de areia). A proposta do sistema sugere a substituição dos atuais servidores de *traceroute*, com funcionalidades limitadas, pelo sistema chamado *Scriptroute*, que é mais flexível, englobando as funções do *traceroute* tradicional.

3.11.1 Arquitetura Scriptroute

O conjunto de componentes do sistema são apresentados na Figura 3.22 e foram separados por questões de robustez e segurança, ou seja, cada parte desempenha tarefas simples e em caso de falha em uma delas, a outra não é afetada (SPRING, 2003).

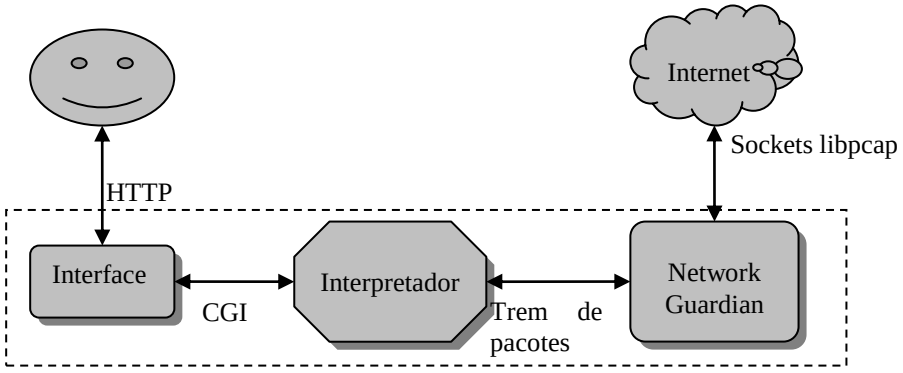

Figura 3.22: Componentes do Servidor *Scriptroute*

A tarefa do *Front-End* é passar o *script* enviado pelo cliente para a execução pelo interpretador. O interpretador é executado em um ambiente restrito e pode falhar caso exceda os limites dos recursos ou pela ocorrência de erros no *script* enviado. O *Network Guardian* é o único componente que necessita de permissões especiais para ler e escrever pacotes de baixo nível.

System Management (Sistema de Gerenciamento)

Os servidores *Scriptroute* publicam sua existência atualizando dinamicamente a base de dados do DNS. Isso permite aos clientes encontrar os servidores pelos seus respectivos nomes. Um exemplo da forma de organização do espaço de nomes do

Scriptroute é apresentado na figura 3.23. O espaço de nomes está separado em duas sub-árvores: servidores (*servers)* e política (*policy*).

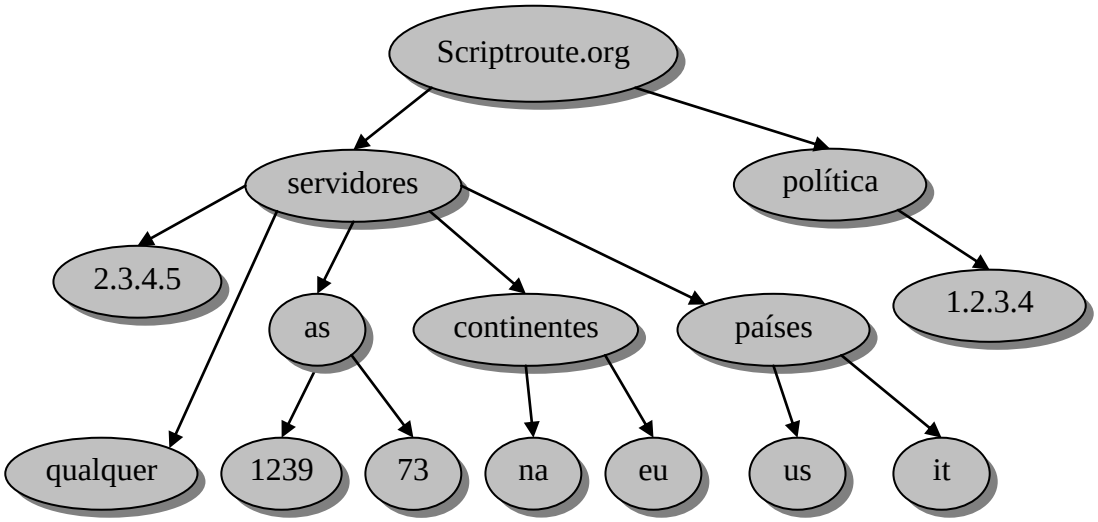

Figura 3.23: Espaço de nomes *Scritptroute*

A sub-árvore de servidores retorna uma lista pseudo-aleatória de servidores *Scriptroute*, opcionalmente escolhida por AS, país ou continente. A base de dados completa pode ser acessada usando uma página *web* gerada dinamicamente.

A sub-árvore de política inclui entradas para destinos de medição para os quais o tráfego de medição deve ser bloqueado. Há dois modos para atualizar este banco de dados. A primeira forma é bloquear um destino individual. Neste caso, basta se conectar no servidor e bloquear o tráfego de medição de retorno para o IP selecionado. A outra forma é um e-mail do administrador para bloquear um prefixo inteiro de IPs.

A interface *Web* provê uma atualização oportuna quando está claro, pelo estabelecimento da conexão TCP, que o usuário da máquina de destino pediu um filtro e, estas mudanças são propagadas imediatamente na sub-árvore de políticas do DNS. Para a inserção ou remoção de filtros mais complexos é necessária a verificação humana.

Server Front-End

Cada servidor *Scriptroute* funciona como um servidor *web* comum, sob a porta 3355 e provê acesso para submissão de *scripts* e tarefas administrativas. O servidor possui três páginas principais: submissão de trabalhos, registros e informações.

A página de submissão de trabalhos provê uma interface para o envio do *script* e apresenta o retorno da medição. O IP do cliente é obtido e validado no estabelecimento da conexão TCP. O servidor utiliza o programa thttpd que possui controle de tempo de execução, tamanho e saída do *script*. Em caso de exceção a conexão é encerrada e o erro é informado ao cliente.

A página de registros permite acesso aos *logs* do sistema, especificamente aos pacotes no formato *tcpdump* enviados para um endereço IP particular de determinado cliente.

A página de informações provê informações sobre os tipos de tráfegos de medição suportados, como contatar o administrador do servidor, como obter mais informações sobre o *Scriptroute* e como adicionar filtros para tráfegos indesejados.

Script Interpreter

A linguagem de script utilizada na implementação foi a linguagem *Ruby*, porém qualquer outra poderia ser utilizada desde que implementasse como segurança uma *sandbox.*

Cada trabalho submetido é interpretado pelo interpretador usando um novo processo, isso evita a interferência entre os diferentes trabalhos. Um controle do uso de recurso é utilizado para prevenir o excesso de consumo de recursos por determinado processo, caso isso ocorra o processo é terminado abruptamente.

O interpretador se comunica com o *Network Guardian* usando a API *Send-train*, que permite efetuar a maioria das medições, enviando um conjunto de pacotes de teste e coletando as respostas. A operação *Send-train* recebe como argumento um vetor contendo dois valores (atraso, pacotes de teste) e retorna um vetor com quatro valores (marca de tempo, pacote de teste, marca de tempo, pacote de resposta), desta forma atende à maioria dos testes.

Network Guardian

O "guardião da rede" é responsável por limitar a taxa do tráfego de medição e regular os tipos de pacotes que são enviados. Ele combina os filtros para bloquear determinados endereços (armazenados no DNS) com o limite da taxa de tráfego e filtros configurados localmente pelo administrador. Ele também é responsável por comparar as sondas e suas respostas, protegendo os *hosts* de tráfegos não solicitados.

Este módulo efetua o acesso a *sockets* de baixo nível e a captura de pacotes pelo *kernel* e, neste caso, deve estar sendo executado como *root* ou com uma configuração especial. Finalmente, este módulo armazena um histórico de todos os pacotes enviados e recebidos.

3.11.2 Gerenciamento Scriptroute

Como o foco da infra-estrutura Scriptroute está no cliente não autenticado, ou seja, para uso da infra-estrutura não é necessário nenhum tipo de identificação, é esperado, portanto, que ela possua um conjunto de funções para prover segurança para o sistema. Neste sentido, a infra-estrutura monitora um conjunto de parâmetros visando detectar anomalias. Para cada teste são monitorados os limites de duração, quantidade de memória utilizada, tempo de processador e a quantidade de tráfego.

No gerenciamento de desempenho a infra-estrutura efetua o controle de requisições para o sistema, por cliente e o número total de requisições.

3.12 Surveyor

O objetivo do projeto Surveyor[3] – *Advanced Network* foi criar tecnologia e infra-estrutura para que usuários e provedores de serviços pudessem ter uma compreensão comum e precisa sobre o desempenho e confiabilidade dos caminhos na Internet. As métricas usadas neste projeto são *one-way delay* (atraso de uma via) e *one-way loss* (perda de uma via), ambas definidas pelo grupo de trabalho IPPM (*Internet Protocol Performance Metrics*) do IETF (*Internet Engineering Task Force*). Para atingirem seus objetivos o projeto observa os seguintes princípios (KALIDINDI, 1999):

[3] O projeto Surveyor já foi encerrado.

Uso de métricas padronizadas – sem o uso de métricas padronizadas e metodologias bem compreendidas, os resultados podem ser de difícil compreensão ou de impossível comparação com outras implementações. Neste projeto são usadas as métricas desenvolvidas pelo grupo de trabalho IPPM do IETF, mais precisamente, atraso unidirecional e perda de pacote unidirecional.

Medição unidirecional – estudos passados mostraram que muitas rotas na Internet são assimétricas, ou seja, a seqüência de roteadores que um pacote passa para atravessar a rede da origem até o destino é diferente da seqüência para atravessar a rede no sentido contrário (destino até origem). Em casos de rotas assimétricas a medida tradicional *round trip time (RTT)* acaba igualando o desempenho em ambos os sentidos da comunicação, enquanto que as medidas de uma via permitem avaliar separadamente cada sentido da comunicação. Mesmo no caso de caminhos simétricos, a carga pode ser radicalmente diferente entre os sentidos da comunicação, justificando o uso de medidas de uma via.

Uso de máquinas dedicadas para medição precisa – os equipamentos de testes foram desenvolvidos pelo próprio projeto e então colocados nos *site*s participantes para efetuar a medição. Cada máquina de medição Surveyor pode ser considerada como um instrumento de medição dedicado. O uso de um *hardware* dedicado ocorreu por três razões: primeiro, um *hardware* dedicado assegura que os equipamentos são uniformes e são utilizados com uma carga controlada; segundo, foi utilizado um *hardware* especial para sincronizar os relógios e este *hardware* é mais fácil de instalar e manter usando computadores dedicados. Finalmente, um *hardware* dedicado provê um alto nível de segurança, pois é necessário habilitar somente os serviços essenciais. Alta segurança é essencial para assegurar que as medidas não serão comprometidas e que os instrumentos de medição não poderão ser comprometidos e serem fontes de ataque.

Medição continuada - o projeto Surveyor foi desenvolvido para ser uma infra-estrutura de medição que pode monitorar constantemente a rede. Embora seja importante não inundar a rede com tráfego de medição, uma medição contínua, de baixa intromissão, pode mostrar tendências e reduzir a possibilidade de perder eventos importantes da rede. Coletar dados apenas em tempos determinados podem mostrar o estado atual da rede, mas pode perder as características e não detectar alertas da mesma.

Armazenar dados de desempenho por um longo período – os dados obtidos ao longo do tempo trazem benefícios, pois ajudam no planejamento de capacidades e engenharia de rede. Além disso, os dados armazenados por um longo período podem ser utilizados em pesquisas sobre o comportamento das redes.

Prover acesso em tempo real aos dados de desempenho – *troubleshooting* ou detecção de eventos da rede necessita de acesso aos dados atuais de desempenho e não dados históricos. O projeto provê dados de performance “próximos ao tempo real”, com uma diferença de no máximo cinco minutos, embora os dados já estejam disponíveis no equipamento de coleta há alguns minutos, eles ainda não estão disponíveis na central.

Medições de desempenho fim-a-fim – normalmente, provedores de diferentes locais estão interessados no desempenho do caminho entre eles, isso, atualmente, implica em atravessar uma série de provedores. Idealmente, seria necessário medir individualmente cada um dos provedores intermediários, requerendo sua colaboração para localizar problemas de desempenho. O que é inviável no curto prazo e neste caso medidas fim-a-fim seriam úteis a ambos os provedores.

3.12.1 Arquitetura Surveyor

A arquitetura Surveyor é composta por três componentes principais: equipamento de medição, o banco de dados, e o servidor de análise. Os equipamentos de medição são as máquinas dedicadas distribuídas entre os *site*s participantes. Os equipamentos de medição remetem as informações para um banco de dados central, que cataloga todos os dados de desempenho. Finalmente, análises são executadas e disponibilizadas pelo servidor de análise. O servidor de análise é acessado pelo protocolo HTTP (*HyperText Transfer Protocol*).

Equipamento de Medição

Os equipamentos de medição são PCs de mesa, com processadores de diferentes velocidades. Além do PC básico, cada máquina tem uma placa de rede apropriada e um cartão de GPS. Inicialmente são suportados os padrões de rede Ethernet (10BaseT e 100BaseT), FDDI (*Fiber Distributed Data Interface*), Conexão Ótica (OC-3) e ATM (Asynchronous Transfer Mode). Os cartões de GPS armazenam localmente o tempo atual e basta ler o barramento para obter o tempo. O sistema operacional é o BSDI (sistema Unix BSD modificado), escolhido pela sua utilização por muitos provedores e pela facilidade de construção de um *driver* para o GPS.

A precisão da medição é determinada em parte pelo grau de sincronização entre os equipamentos de medição. Usando cartões de GPS é obtida uma sincronização da ordem de dois microssegundos.

A métrica *One-Way Delay* ou Atraso de uma via, do IPPM é definida como "*wire times*" e significa o tempo entre a transmissão do primeiro bit do pacote de teste até o recebimento do último bit pela placa de rede do receptor. Acontece que o *software* que efetua a medição está no nível de usuários e as marcas de tempo coletadas neste nível são chamadas "*host time*". A diferença entre as duas medidas é o *overhead* introduzido por *hardware* e *software*. O *software* de medição é mais fácil de ser implementado e gerenciado em nível de usuário, porém é sensível à carga do equipamento, ou seja, a precisão pode ser comprometida devido à carga de medição no equipamento.

Esta sensibilidade à carga representa um problema de escala e para solucionar este problema foi necessária uma modificação no *driver* de rede do sistema operacional. A modificação é simples e envolve o uso de uma *flag* para indicar que determinado socket está sendo usado para um teste de performance. No código do *kernel* esta *flag* é verificada e se estiver ligada uma marca de tempo é efetuada no próprio *kernel*, não necessitando esperar que o processo de usuário receba o pacote e só aí insira a marca de tempo.

Banco de Dados Central

Todas as medições são transferidas para um Banco de Dados Central (BDC) para posterior análise e armazenagem (histórico). Os equipamentos de medição coletam os dados de desempenho e os armazenam localmente. A cada cinco minutos o banco de dados central consulta estes equipamentos para verificar a existência de novos dados de medição, caso existam eles são coletados. Assim como a consulta, o envio dos dados é feito via *ssh*. Por questões de segurança, todos os equipamentos de medição confiam no Banco de Dados Central e este não confia em nenhum equipamento de medição.

O banco de dados central na verdade não é um banco de dados tradicional (relacional ou orientado a objetos), mas arquivos dentro da estrutura de diretórios do

sistema operacional. Os dados provenientes das medições são armazenados em formato binário e os dados das rotas são no formato ASCII (*American Standard Code for Information Interchange*).

Servidor de Análise

Os dados coletados pelos equipamentos de medição e enviados ao BDC agora são sumarizados estatisticamente. Para cada rota são produzidos gráficos com as estatísticas sumarizadas das últimas 24 horas de coleta. As estatísticas diárias são baseadas no UTC (*Universal Time Coordinated*) e a geração começa à meia noite. Os gráficos, os dados do *traceroute* e as estatísticas estão disponíveis usando o servidor http.

Para cada rota um registro, que consiste em um par <*tempo_que_o_pacote_foi_enviado*, *atraso_de_uma_via_observado*>, é armazenado no BDC. Se o pacote for perdido, o atraso observado é gravado como infinito. Adicionalmente, para cada rota e para cada dia, estatísticas sumarizadas de cada minuto são geradas e armazenadas no banco de dados. Neste caso, o dia foi dividido em 1440 minutos e como a maioria das rotas é mensurada pela média de duas amostras por segundo, temos aproximadamente 120 amostras por minuto. Para cada minuto são computadas as seguintes estatísticas:

- Atraso mínimo durante o minuto;
- Atraso durante o minuto (percentil 50%);
- Atraso durante o minuto (percentil 90%);
- Percentual de pacotes perdidos durante o minuto.

3.12.2 Metodologia de Medição

Atraso e Perda

O atraso e perda são avaliados usando o mesmo tráfego ativo de teste. Os pacotes de teste são enviados segundo uma distribuição de Poisson. O uso de duas amostras por segundo foi escolhido com um compromisso pragmático, obter o máximo de detalhes com testes mais freqüentes, porém não tão freqüentes que comprometessem as medições. Adicionalmente, existia um limite de espaço em disco: dois pacotes por segundo são 178800 medidas por dia (por caminho) requerendo mais de 2 megabytes por dia, mais o overhead do banco de dados relacional.

Cada pacote de testes possui o mínimo de tamanho: 12 bytes de dados de usuário, essencialmente um número de seqüência e uma marca de tempo. Os pacotes de teste usam o protocolo *User Datagram Protocol (UDP)*, assim o tamanho do pacote, excluindo o cabeçalho MAC (*Medium Access Control*) é de 40 bytes. O pacote mínimo foi escolhido devido ao objetivo inicial de obter um bom entendimento do comportamento dos caminhos, obtendo uma linha base do comportamento.

Uma vez que as máquinas de medição são equipadas com GPS para sincronização de tempo, a marca de tempo levada pelos pacotes tem validade global. A máquina receptora determina o atraso do pacote de teste simplesmente subtraindo o tempo recebido no pacote do tempo atual. Quanto o atraso for superior a 10 segundos o pacote é considerado perdido. Os pacotes perdidos são tratados como pacotes enviados com atraso infinito.

Informações de Rotas

As informações de rotas são obtidas para os mesmos caminhos de obtenção dos valores de atraso e perda. As informações são obtidas usando uma versão modificada do *traceroute*. São três modificações: Primeira, o programa é mais persistente na presença de falha. Se o tempo de vida (TTL) do pacote ICMP expirar sem resposta, são feitas 10 tentativas, ao invés de 3, que é o valor padrão. Segunda, é menos persistente no caso de sucesso. Ao invés de enviar 3 sondas em todos os casos, o programa deixa de enviar as demais em caso de sucesso das anteriores. Terceira, o programa não mantém nenhuma informação sobre o tempo dos pacotes. Os tempos gerados pelo *traceroute* são descartados.

O intervalo entre as amostras é gerado baseado em um processo de Poisson truncado. As amostras deveriam ser geradas em intervalos por volta de 10 minutos, porém na prática o maior intervalo entre amostras pode chegar a mais de uma hora, tornando as informações menos úteis. Então, o processo foi programado para efetuar pelo menos uma amostra a cada 10 minutos, ou antes, se o processo assim determinar.

3.12.3 Gerenciamento Surveyor

O gerenciamento previsto na infra-estrutura Surveyor é mínimo, define apenas o uso de *ssh* para transferência dos dados entre os bancos de dados.

3.13 Comparativo das infra-estruturas

Nas seções anteriores foram apresentas as infra-estruturas de medição que possuíam informações públicas disponíveis, perfSONAR, AMP, BW, MonALISA, NIMI, NWS, PingER, piPES, RIPE, Scriptroute e Surveyor. Esta seção encerra o capítulo fazendo um comparativo entre as infra-estruturas de medição quanto às suas características.

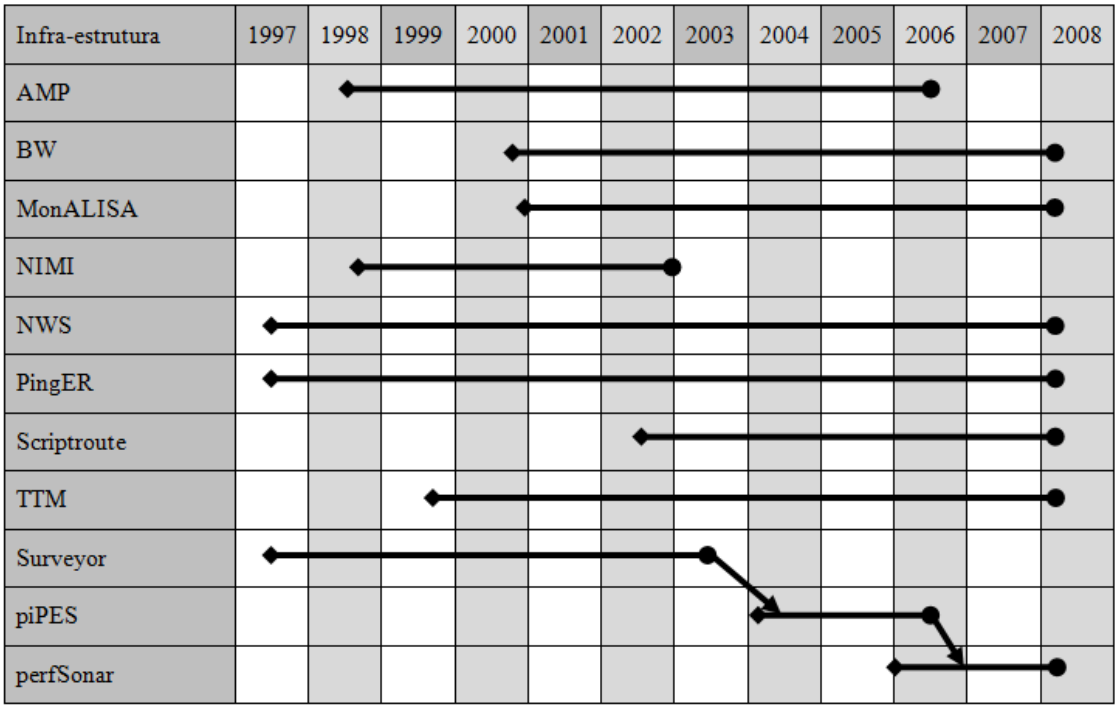

Figura 3.24: Linha de Tempo das Infra-Estruturas de Medição de Desempenho

A figura 3.24 apresenta a linha do tempo das infra-estruturas avaliadas. Nela podemos perceber que algumas já tiveram seus projetos encerrados. A infra-estrutura AMP, por exemplo, tinha prazo determinado para o financiamento do projeto e ao fim do mesmo as suas atividades foram encerradas. Outras infra-estruturas foram simplesmente encerradas por dificuldade de uso, não estavam atualizadas ou migraram para o desenvolvimento de outros projetos.

Um caso interessante ocorre com as infra-estruturas Surveyor, piPES e perfSonar. Neste caso, as pessoas envolvidas nos projetos migraram, ou seja, direcionaram seus esforços para o desenvolvimento de uma proposta mais atualizada. Como visto na figura 3.24, o grupo de desenvolvimento da infra-estrutura Surveyor passou a integrar o desenvolvimento da infra-estrutura piPES e atualmente o grupo de trabalho do piPES passou a trabalhar na proposta do perfSonar.

Mesmo que a grande maioria das infra-estruturas continue em atividade, apenas algumas efetivamente estão em evolução no presente momento. São elas: MonALISA, TTM e perfSonar.

3.13.1 Quanto aos testes

Analisando-se os testes de avaliação de desempenho, podemos classificar as infra-estruturas de medição em três grupos: baseadas em atraso de ida e volta (*rtt*), atraso de uma via (*one-way*) e dinâmicas (os testes podem ser construídos ou inseridos de forma dinâmica).

As infra-estruturas que usam testes baseados em atraso de ida e volta são PingER, BW, AMP e NWS. Os testes utilizados por estas infra-estruturas utilizam o programa *ping* ou variações. O tempo entre o envio de um pacote e a resposta do mesmo é contabilizado e relacionado com o tamanho do pacote.

As infra-estruturas Surveyor, RIPE e piPES são baseadas em atraso de uma via e implementam as técnicas propostas pelo IPPM para medir *one-way delay*. Neste caso, exige que os relógios dos equipamentos estejam sincronizados.

As infra-estruturas que possibilitam a inclusão de testes dinamicamente são MonALISA, NIMI e Scriptroute. A forma de inclusão de um teste pode ser na forma de scripts ou como um pacote que é executado pela infra-estrutura. Estas infra-estruturas estão mais sujeitas a problemas devido à possibilidade de inconsistências nos testes.

O perfSONAR trabalha com um conjunto específico de testes, a inclusão de novos testes é possível porém não dinamicamente. Esta infra-estrutura está incorporando os testes previstos na infra-estrutura piPEs.

3.13.2 Quanto ao sincronismo

Para a avaliação de medidas *one-way,* a sincronização dos relógios dos equipamentos é um fator decisivo. Para manter os relógios sincronizados as infra-estruturas Surveyor e RIPE utilizam um equipamento externo do tipo GPS. A infra-estrutura piPES utiliza o protocolo NTP para sincronismo, porém pressupõe que o servidor NTP esteja sincronizado com um relógio externo confiável.

3.13.3 Quanto ao Gerenciamento

O ponto mais relevante para este trabalho são as características de gerenciamento presentes nas infra-estruturas. Este aspecto foi levantado individualmente para cada uma

das infra-estruturas neste capítulo e fica evidente o tratamento variado dado pelas infra-estruturas para o gerenciamento. Algumas propostas simplesmente ignoram a necessidade de gerenciamento e apenas definem um conjunto mínimo de *hardware* e *software* para compor a infra-estrutura. As que mais englobam gerenciamento prevêem um pequeno conjunto de funções que normalmente são utilizadas para controle de usuários e para o controle das ferramentas de testes.

Por falta de um conjunto de ações de gerenciamento bem definido as infra-estruturas de medição de desempenho avaliadas podem apresentar problemas durante o seu funcionamento que inviabilizem a execução dos testes. Além disso, a falta de gerenciamento dificulta a manutenção da infra-estrutura. Abaixo segue algumas deficiências das infra-estruturas analisadas que podem comprometer seu funcionamento correto. São elas:

a) Desconhecimento dos componentes de *hardware*. As infra-estruturas definem os requisitos mínimos para os equipamentos que farão parte da infra-estrutura, mas não possuem nenhum mecanismo que possa informar se houve alterações no *hardware*, nem se o mesmo está funcionando corretamente. A exceção é para a infra-estrutura TMN que o *hardware* é no formato "*black-box*";

b) Desconhecimento dos componentes de *software*. Assim como os componentes de *hardware*, após a instalação inicial da infra-estrutura, ela não possui mecanismos para avaliar os *softwares* e as respectivas versões. Por exemplo, se são os corretos para a o bom funcionamento da infra-estrutura. Isso ocorre principalmente com os *softwares* adicionais, como aplicações do sistema operacional e programas de apoio aos testes. A exceção é para a infra-estrutura TTM que o *hardware* é no formato "*black-box*" e para a MonALISA que os programas estão centralizados e são enviados dinamicamente para os diferentes pontos da infra-estrutura;

c) Em geral as infra-estruturas se preocupam apenas com a coleta dos resultados dos testes, em raros casos a própria execução do teste é avaliada. Um problema eventual na aplicação que efetua o teste pode produzir valores errados;

d) A inclusão, exclusão ou atualização dos testes não é dinâmica (exceto a infra-estrutura MonALISA que distribui os testes de um ponto centralizado);

e) O administrador local e o administrador da infra-estrutura não são notificados quando ocorre algum problema de *hardware* ou *software*. Qualquer problema desse tipo que ocorra num equipamento pertencente a uma infra-estrutura de medição só será de conhecimento do administrador quando algum usuário perceber o problema e informá-lo;

f) O administrador local não tem ação sobre a infra-estrutura. No caso da infra-estrutura TTM que o equipamento é um "Black-box", o serviço é pago e o administrador local não tem nenhum acesso ao equipamento. As demais infra-estruturas exigem a instalação local de *softwares* pelo administrador local ou por meio de uma conta de acesso privilegiada. Em geral, após a instalação o administrador local não tem mais liberdade de atuação sob a infra-estrutura, apenas será chamado quando for necessária alguma atualização nos componentes da infra-estrutura.

g) Atualização manual. Com exceção da infra-estrutura MonALISA que possui os testes centralizados, as demais, que necessitam de instalação local necessitam de intervenção manual para atualização. A atualização pode ser feita por um

administrador local ou remotamente, mas neste caso é necessário acesso privilegiado ao equipamento, podendo comprometer a segurança do domínio.

h) Testes com parâmetros fixos. De forma geral as infra-estruturas definem os programas de testes para serem instalados e uma configuração padronizada. Essas configurações são definidas no ato da instalação e somente podem ser modificadas manualmente. Este fato impossibilita a execução de testes personalizados ou com parâmetros definidos dinamicamente. As infra-estruturas que permitem a utilização de testes dinâmicos também permitem executar testes com diferentes parâmetros;

i) Inexistência de monitoração preventiva. O fato das infra-estruturas não definirem funções de gerenciamento, impossibilita que ações preventivas possam ser efetuadas para facilitar a manutenção, antecipando-se a possíveis problemas;

A proposta de gerenciamento que será apresentada no próximo capítulo procura preencher esta lacuna de falta de gerenciamento das infra-estruturas de medição de desempenho e com isso tentar solucionar os problemas listados acima.

O quadro 3.1 relaciona as infra-estruturas e suas principais características de gerenciamento. As características estão organizadas de acordo com as áreas funcionais definidas no modelo de gerenciamento OSI.

Infra-Estrutura	Áreas Gerenciamento				
	Falhas	**Configuração**	**Contabilização**	**Performance**	**Segurança**
GFD	Previsão de *backups* e técnicas de recuperação de falhas para os dados coletados nas medições. Heartbeat para verificação da disponibilidade do serviço.	Controle dos recursos dos pontos de medição de acordo com regras de configuração e da capacidade do recurso			Autenticação e autorização para todos os usuários (em implementação)
AMP	Controle de falhas por um espelho ativo banco de dados				
BW	Notificação de falhas repetidas nos testes	Inicialização automática (pré-estabelecida) e coleta da configuração a cada inicialização.			Clientes previamente cadastrados
MonALISA	Em caso de falha o serviço é reiniciado automaticamente.	Atualização automática (a cada inicialização)			Conexões seguras (SSL)
NIMI	Estado dos testes é armazenado no sistema de arquivos (pode ser reiniciado de onde parou).				Testes são autenticados e usuários possuem controle de acesso.
NWS				Controle de tempo entre a execução dos testes.	
PingER					
piPEs			As ferramentas Owamp e Bwctl permitem a definição de limites de execução (disco).		Uso de autenticação nos testes
Scritproute			Número de requisições ao sistema, por cliente e total. Quantidade de memória, processador e tráfego.		
Surveyor					Define que a transferência dos dados é de forma segura.
TTM		Instalação pré-configurada, disco especial com arquivos e configuração.	Monitoração dos processos.		

Quadro 3.1: Relação das infra-estruturas e suas principais características de gerenciamento.

71

4 GERENCIAMENTO DE INFRA-ESTRUTURAS DE MEDIÇÃO

Uma infra-estrutura de medição é importante para avaliação do comportamento da rede: quanto mais abrangente ela for, melhor é o conjunto de informações obtidas. O uso eficiente destas infra-estruturas em larga escala é um dos objetivos mais difíceis de ser alcançado. Por exemplo, uma das propostas já antecipa os problemas e informa que da forma como a infra-estrutura está sendo proposta não é escalável e não poderá ser ampliada para a Internet (McGREGOR, 2000). Um dos problemas encontrados nas tentativas de implementação de infra-estruturas de medição em escala global é o gerenciamento dos componentes em diferentes domínios administrativos. A complexidade começa já na inclusão de um novo elemento na infra-estrutura, sendo necessário estabelecer um conjunto de procedimentos e autorizações. Posteriormente, são as dificuldades na manutenção e atualização dos sistemas.

O capítulo anterior apresentou as infra-estruturas de medição públicas e suas características de gerenciamento. De forma geral, o funcionamento das infra-estruturas depende dos esforços pessoais dos administradores para manter o serviço em funcionamento. O nível de tarefas de gerenciamento automatizadas é mínimo e pequenas falhas podem deixar os testes inoperantes até que o administrador faça uma verificação manual ou até que seja notificado por um usuário.

Este capítulo apresenta um modelo de gerenciamento específico para infra-estruturas de medição. Inicialmente é apresentado um conjunto de funções de gerenciamento necessárias para o funcionamento das infra-estruturas de medição, constituindo os requisitos do sistema. Na seqüência, o modelo de gerenciamento será apresentado, bem como seus componentes e interações.

4.1 Requisitos de Gerenciamento

Uma modelo de gerenciamento para atender as infra-estruturas de medição considerar alguns requisitos:

- A infra-estrutura pode ser composta por centenas de componentes (pontos de medição, de coleta e de análise);
- Os componentes pertencem a domínios administrativos diferentes e, conseqüentemente, serão heterogêneos;
- Os componentes podem ser inseridos ou retirados dinamicamente;

Considerando os requisitos citados acima, para atender efetivamente as necessidades de gerenciamento das infra-estruturas de medição, o modelo deve possuir algumas funcionalidades:

- Monitorar CPU, memória, disco, rede e outros recursos locais;

- Manusear configurações do sistema operacional;
- Manusear as configurações das ferramentas de testes;
- Enviar um alarme sobre certas circunstâncias (sendo estas configuráveis pelo usuário);
- Interferir o mínimo possível (em termos de CPU, memória e rede) nos equipamentos;
- Permitir visualizar os estado de todos os componentes da infra-estrutura;
- Ser escalável;
- Ser expansível;

O modelo proposto organiza as funções de gerenciamento de acordo com as áreas funcionais definidas no modelo de gerenciamento OSI, que continuam sendo válidas para outros modelos de gerenciamento. Nesta seção, as áreas funcionais são utilizadas para organizar a apresentação dos principais requisitos de gerenciamento do modelo que está sendo proposto.

4.1.1 Gerenciamento de Falhas

Como apresentado no capítulo 2 o gerenciamento de falhas tem a responsabilidade de monitorar os estados dos recursos, da manutenção de cada um dos objetos gerenciados e das decisões que devem ser tomadas para restabelecer as unidades do sistema que venham a dar problemas.

Neste caso, cada ponto presente numa infra-estrutura de medição possui vários componentes que podem apresentar falhas e comprometer o funcionamento local e, em última instância, a infra-estrutura como um todo. Na seqüência são apresentados os componentes que devem ser monitorados para que falhas eventuais sejam detectadas o mais cedo possível e, em alguns casos, tomar uma ação imediata visando retomar o funcionamento normal. Os componentes são:

Hardware: O *hardware* como um todo deve ser monitorado: CPU, discos e a interface de comunicação (placa de rede). No caso da placa de rede, um dos objetivos é evitar que uma interface que esteja funcionando com problemas possa comprometer os resultados dos testes.

Sistema Operacional: Encontra-se em relatos de implantação de infra-estruturas (PAXSON, 2002) que problemas simples como falta de espaço em disco comprometeram o funcionamento dos testes. Além disso, é necessário monitorar a presença das aplicações do sistema operacional necessárias para o funcionamento da infra-estrutura, por exemplo, *deamon ssh*, servidor de banco de dados, *cron* e outros. A relação de aplicações necessárias varia de acordo com a infra-estrutura a ser gerenciada.

Ferramentas de Testes: Uma infra-estrutura de medição possui um conjunto de aplicações que são responsáveis pelos testes. Tomando, como exemplo, o teste de atraso de uma via, usando o protocolo OWAMP, ele necessita interação entre os pontos de medição. Essa interação é efetuada baseada no modelo cliente-servidor ou P2P, portanto a monitoração da execução da aplicação servidora é essencial para garantir o funcionamento dos testes.

4.1.2 Gerenciamento de Configuração

As funções do gerenciamento de configuração correspondem ao conjunto de facilidades que permitem controlar os objetos gerenciados, identificando, coletando e disponibilizando dados sobre estes objetos. Nesta área é necessário monitorar os seguintes pontos:

Ferramentas de Testes: Como o nome da área sugere, o gerenciamento de configuração deve registrar e verificar a configuração das ferramentas. O registro é importante para gerar "discos de configuração" que armazenam as configurações customizadas, facilitando novas instalações ou recuperação em caso de falhas. Outra tarefa importante do gerenciamento de configuração é atualização das ferramentas. Por exemplo, o administrador pode definir que as atualizações serão manuais e que deseja apenas ser notificado quando uma nova versão for disponibilizada.

Sistema Operacional: A principal tarefa do gerenciamento de configuração no sistema operacional é registrar e armazenar a configuração dos componentes necessários para o funcionamento da infra-estrutura. Por exemplo, as infra-estruturas de medição normalmente utilizam o *openssh* como um componente de segurança. O registro e armazenamento da sua versão e configuração podem evitar problemas de compatibilidade.

4.1.3 Gerenciamento de Contabilização

As funções do gerenciamento de contabilização provêem meios para se medir e coletar informações a respeito da utilização dos recursos e serviços de uma rede. O conhecimento da taxa de uso dos recursos é fundamental para questões de tarifação e para o planejamento das capacidades da rede. Nesta área os seguintes fatores podem ser contabilizados:

Usuários: A contabilização das atividades dos usuários pode evitar abusos no uso da infra-estrutura de medição. Alguns testes de largura de banda geram grandes quantidades de pacotes e, um usuário mal intencionado pode usar este mecanismo como uma forma de ataque de negação de serviço, procurando congestionar o enlace até o ponto de medição. A contabilização associada ao estabelecimento de limites de utilização permite fazer uso racional da infra-estrutura de medição.

Ferramentas de Testes: A monitoração da quantidade de solicitações de cada teste, associado ao consumo de recursos por cada um deles é importante para o planejamento das capacidades do ponto de medição.

4.1.4 Gerenciamento de Desempenho

O gerenciamento de desempenho é o conjunto de funções responsável pela manutenção e exame dos registros de desempenho com o objetivo de serem usados na análise de tendências e no conseqüente planejamento do sistema.

Uma infra-estrutura de medição normalmente integra um sistema de gerenciamento de desempenho e seus resultados são utilizados para avaliação e planejamento da rede, porém pode-se também avaliar o desempenho dos componentes das infra-estruturas de medição, tais como as ferramentas de testes.

Ferramentas de Testes: A monitoração da execução de determinado teste implica em avaliar o tempo de execução, a quantidade de memória utilizada, quanto de processador e quantidade de tráfego gerado. Este monitoração visar identificar comportamentos anormais das ferramentas, por exemplo, *loops* de programação ou

outras falhas. Outro fator importante é estabelecer uma relação entre o *hardware* consumido pelos testes e o *hardware* existente. O administrador necessita ser notificado se esta relação ultrapassar valores pré-determinados. Este gerenciamento permite avaliar e definir quais são e qual o número de testes simultâneos que podem ser efetuados.

4.1.5 Gerenciamento de Segurança

A função do gerenciamento de segurança é dar subsídios à aplicação de políticas de segurança, protegendo os objetos gerenciados e o sistema de acessos indevidos de intrusos. O gerenciamento de segurança nas infra-estruturas de medição pode ser aplicada em:

Usuários: Numa infra-estrutura de medição estão presentes diferentes ferramentas de testes, com diferentes funcionalidades e diferentes requisitos. É necessário gerenciar a relação entre usuários e ferramentas de teste. As ferramentas de avaliação de largura de banda, por exemplo, geram muito tráfego na rede e, portanto, seu uso deve ser racional. Seria normal acreditar que estas ferramentas tenham seu uso restrito pelo administrador.

Aplicações: Uma das preocupações dos administradores é a possibilidade de ataque ao equipamento destinado para a infra-estrutura de medição e o posterior uso em ataque a outros equipamentos. A monitoração das permissões e da identificação das ferramentas de teste visa detectar um possível funcionamento anormal do programa.

Dados: Os resultados dos testes são armazenados e podem ser disponibilizados em diferentes níveis, de acordo com os usuários. Por exemplo, os dados brutos de medição são de acesso restrito e dados consolidados são de acesso público. É tarefa do projetista da infra-estrutura identificar e definir estas diferenças.

4.2 Modelo de Gerenciamento para Infra-estruturas de Medição

Uma infra-estrutura de medição de alcance global é inevitavelmente distribuída entre diferentes domínios administrativos. Obviamente, o modelo de gerenciamento deve estar de acordo com essa característica.

O modelo considera que cada domínio administrativo participante da infra-estrutura de medição deve disponibilizar pelo menos um elemento para participar dos testes. Este elemento será identificado de agora em diante no texto como *ponto de medição*.

Um *ponto de medição* é constituído por um *hardware* (dedicado ou não, dependendo da infra-estrutura) e um conjunto de *softwares* que permitem elaborar testes de desempenho entre os participantes da infra-estrutura, além de armazenar e disponibilizar os resultados.

Cada domínio administrativo deve possuir um gerente que responde pelo domínio e que fará interação com os gerentes dos demais domínios. As relações entre os gerentes são apresentadas na seqüência do capítulo.

Como visto no capítulo anterior, o maior problema encontrado nas infra-estruturas apresentadas anteriormente é a falta de gerenciamento apropriado dos componentes pertencentes à infra-estrutura.

O modelo proposto visa simplificar os processos de inclusão e manutenção de domínios numa infra-estrutura de escala global. Um aspecto chave para o sucesso da

proposta é trazer benefícios mútuos para usuários e administradores, sempre considerando os aspectos de segurança.

A estratégia base do modelo proposto é permitir que cada novo domínio gerencie seus dispositivos locais ao mesmo tempo em que provê informações para serem exportadas ou trocadas com outros domínios participantes da mesma infra-estrutura de medição. Cada administrador de domínio (humano) pode definir quais são as interações possíveis com outros domínios, considerando as suas políticas locais.

Cabe ressaltar que as infra-estruturas de medição não possuem o mesmo conjunto de funcionalidades, e neste caso, o conjunto de funções definidas deve atender às especificidades de cada infra-estrutura. Para prover essa funcionalidade, a proposta foi dividida em um conjunto de módulos genéricos.

De forma geral podemos decompor as infra-estruturas de medição em quatro módulos chave: *hardware*, sistema operacional, testes, específico (para aplicações diferentes em cada infra-estrutura). Para cada módulo chave que compõe uma infra-estrutura, foi definido um módulo de gerenciamento associado.

Os módulos de gerenciamento são compostos de um conjunto básico funções para o gerenciamento de infra-estruturas de medição. Para possibilitar a expansão do sistema e para atender demandas não contempladas inicialmente, foi definido um módulo denominado "expansão". Além disso, se necessário, um novo módulo pode ser incorporado ao modelo no futuro. A figura 4.1 apresenta o modelo de gerenciamento com um conjunto inicial de módulos de gerenciamento.

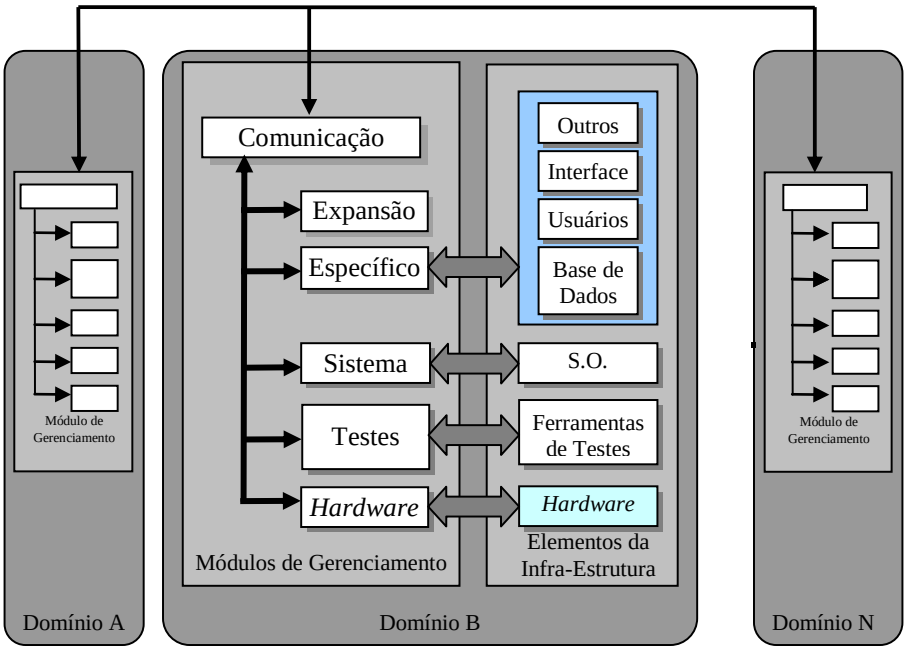

Figura 4.1: Modelo de Gerenciamento de Infra-estruturas de Medição de Desempenho

Adicionalmente aos módulos anteriores, para permitir a comunicação entre os diferentes domínios, foi definido um módulo denominado "comunicação". As funcionalidades de cada módulo de gerenciamento são descritas na seqüência.

4.2.1 Módulo *Hardware*

O módulo *Hardware* é o responsável pela monitoração de todos os componentes de *hardware* presentes no ponto de medição que podem comprometer a infra-estrutura de medição (CPU, disco, interface de redes). Este monitoração procura prevenir que o funcionamento inadequado destes componentes possa comprometer os testes (por exemplo, uma falha no disco ou uma interface de rede danificada). Em resumo, este módulo é o responsável por detectar e notificar falhas de *hardware* nos componentes da infra-estrutura do domínio. O quadro 4.1 apresenta os serviços inicialmente definidos para este módulo.

Serviço	Obrigatório /Protótipo	Descrição	Entradas	Saída
CheckNIC	Sim/Sim	Verifica o funcionamento correto da placa de rede	Literal (endereço do agente)	Real (percentual)
CheckDisc	Sim/Sim	Verifica o espaço disponível em disco	Literal (endereço do agente)	Real (percentual)
CheckCPU	Não/Não	Verifica o uso de CPU do equipamento	Literal (endereço do agente)	Real (percentual)
CheckMem	Não/Não	Verifica o uso da memória do equipamento	Literal (endereço do agente)	Real (percentual)
CheckForce	Não/Não	Verifica os estado da fonte de alimentação ininterrupta (No-break)	Literal (endereço do agente)	Real (percentual)

Quadro 4.1: Lista de serviços definidos no módulo *Hardware*

4.2.2 Módulo Sistema

Como citado na seção 4.1, existem relatos de que pequenos problemas podem comprometer a execução dos testes. Adicionalmente existe um conjunto de aplicações relacionadas ao sistema operacional do equipamento, por exemplo, *cron* e *ssh* que devem ser monitorados para manter a execução dos testes de maneira correta. Outra função importante deste módulo é registrar e armazenar as configurações dos componentes necessários para o funcionamento da infra-estrutura. Por exemplo, quando uma infra-estrutura utiliza a ferramenta *openssh*, como um componente de segurança, é necessário o registro da versão e configuração deste componente para poder evitar problemas de compatibilidade. O quadro 4.2 apresenta os serviços inicialmente definidos para este módulo.

Serviço	Obrigatório /Protótipo	Descrição	Entrada	Saída
checkApp	Sim/Sim	Verifica a execução de determinada aplicação (como ssh, cron)	(Literal, Inteiro) (endereço do agente, ID da Aplicação)	Booleano (estado da execução)
setConf	Não/Sim	Salva a configuração de determinada aplicação	(Literal, Inteiro) (endereço do agente, ID da Aplicação)	Booleano (retorno da operação)
getConf	Não/Não	Recupera a configuração de determinada aplicação	(Literal, Inteiro) (endereço do agente, ID da Aplicação)	Literal (configuração do teste)
getVersion	Não/Não	Recupera a versão de determinada aplicação	(Literal, Inteiro) (endereço do agente, ID da Aplicação)	Literal (versão da aplicação)

Quadro 4.2: Lista de serviços definidos no módulo *Sistema*

4.2.3 Módulo Testes

Uma infra-estrutura de medição possui um conjunto de aplicações que são responsáveis pelos testes de desempenho entre os domínios. Por exemplo, o teste *one-way delay* efetuado pelo protocolo OWAMP necessita uma interação entre os pontos de medição. Esta interação é baseada no modelo cliente-servidor. Portanto, monitorar a execução do lado servidor da aplicação é essencial para garantir a operação do teste. Este módulo de gerenciamento é o responsável pelo gerenciamento da configuração e monitoração da execução dos testes.

As tarefas de gerenciamento envolvem o registro e monitoração da configuração das ferramentas de teste. O registro é importante para gerar “discos de configuração”, para armazenar configurações customizadas, facilitando novas instalações, recuperação em caso de falhas ou atualizações. Por exemplo, o administrador local (humano) pode definir que as atualizações serão manuais e que deseja apenas ser notificado quando uma nova versão estiver disponível.

A monitoração da execução dos testes é importante porque ele fornece informações para o planejamento do ponto de medição. A monitoração de determinado teste implica em avaliar o tempo de processamento, a quantidade de memória e processador usado, e a quantidade de tráfego gerado. Esta tarefa permite identificar comportamentos anormais das ferramentas de testes.

Outra importante função deste módulo de gerenciamento é estabelecer uma relação entre a quantidade de recursos consumidos por um teste e a quantidade de recursos disponíveis. O administrador pode ser notificado se um valor pré-determinado de consumo de recursos for ultrapassado. Isso ajuda ao administrador definir a quantidade máxima de testes que pode ser executado simultaneamente. O quadro 4.3 apresenta os serviços inicialmente definidos para este módulo.

Serviço	Obrigatório /Protótipo	Descrição	Entradas	Saídas
ListTools	Sim/Sim	Lista todas as ferramentas (testes) disponíveis	Literal (endereço do agente)	Literal (lista de ferramentas)
updadeConfig	Sim/Sim	Recebe uma nova versão de configuração de uma determinada ferramenta de teste.	(Literal, Arquivo) (endereço do agente, arquivo de configuração)	Booleano (resultado da operação)
updadeNotifica	Sim/Sim	Recebe a notificação da existência de uma nova versão de determinada ferramenta.	(Literal, Literal) (endereço do agente, Notificação)	Booleano (resultado da operação)
checkTools	Sim/Sim	Verifica a existência e a execução de determinada ferramenta.	(Literal, Inteiro) (endereço do agente, ID da Aplicação)	Booleano (resultado da operação)
checkAllTools	Sim/Sim	Verifica todas as ferramentas de determinada infra-estrutura.	Literal (endereço do agente)	Booleano (resultado da operação)
getConfig	Sim/Não	Captura as informações de configuração da infra-estrutura	Literal (endereço do agente)	Literal (configuração)
getApplication	Não/Não	Verifica quais as ferramentas estão instaladas.	Literal (endereço do agente)	Literal (lista de aplicações)
getConfigTest	Não/Não	Captura as informações de configuração dos testes	(Literal, Inteiro) (endereço do agente, ID da Aplicação)	Literal (configuração)
monTest	Não/Não	Controla a execução de determinado teste (consumo de recursos).	(Literal, Inteiro) (endereço do agente, ID da Aplicação)	Real (tempo de execução)

Quadro 4.3: Lista de serviços definidos no módulo *Testes*

4.2.4 Módulo Específico

Este módulo é definido para incorporar todas as funcionalidades que não são comuns a todas as infra-estruturas. Como apresentado na figura 4.1, o módulo específico pode, por exemplo, contabilizar as atividades do usuário ou gerenciar a base de dados, além de monitorar a relação entre os usuários e os testes.

Este módulo gerencia o acesso aos testes e as atividades de usuário. Os resultados dos testes são armazenados e podem ser publicados em diferentes níveis, de acordo com os usuários. Por exemplo, os dados "brutos" devem ser restritos e os dados consolidados podem ter acesso público.

A contabilização das atividades dos usuários pode evitar o uso abusivo do ponto de medição. Assim, este módulo provê mecanismos para lidar com as funcionalidades específicas de cada infra-estrutura de medição. O quadro 4.4 apresenta os serviços definidos para este módulo.

Serviço	Obrigatório /Protótipo	Descrição	Entradas	Saídas
getInfo	Sim/Sim	Informações gerais sobre o ponto de medição.	Literal (endereço do agente)	Literal (Informações)
mapUser	Não/Não	Contabiliza os acessos dos usuários a infra-estrutura de medição.	(Literal, Inteiro) (endereço do agente, ID do Usuário)	Inteiro (número de acessos)
mapUserTool	Não/Não	Contabiliza a relação de usuários e testes por ele executados.	(Literal, Inteiro, Inteiro) (endereço do agente, ID do Usuário, ID da Aplicação)	Inteiro (número de execuções do teste)

Quadro 4.4: Lista de serviços definidos no módulo *Específico*

4.2.5 Módulo Comunicação

Este módulo é o responsável pela comunicação entre os gerentes de diferentes domínios. Este módulo permite, por exemplo, que um gerente verifique a disponibilidade de determinado teste em outro gerente. Na verdade, todas as interações entre os gerentes devem ser feitas por meio deste módulo. A próxima seção detalha o funcionamento deste módulo.

O módulo Comunicação é o mais complexo e o possui o maior número de serviços definidos. O quadro 4.5 apresenta os serviços definidos para este módulo.

Serviço	Obrigatório /Protótipo	Descrição		Protótipo
ListTools	Sim/Sim	Lista todas as ferramentas (testes) disponíveis	(Literal, Literal) (endereço do gerente, endereço do agente)	Literal (Informações)
updadeConfig	Sim/Sim	Recebe uma nova versão de configuração de uma determinada ferramenta de teste.	(Literal, Literal, Inteiro) (endereço do gerente, endereço do agente, ID da Aplicação)	Booleano (resultado da operação)
checkHW	Sim/Sim	Checagem dos componentes de *hardware* local	(Literal, Literal) (endereço do gerente, endereço do agente)	Booleano (resultado da operação)
checkSW	Sim/Sim	Checagem dos componentes de *software* local	(Literal, Literal) (endereço do gerente, endereço do agente)	Booleano (resultado da operação)
remInf	Sim/Sim	Busca informações gerais de outros domínios	(Literal, Literal) (endereço do gerente, endereço do agente)	Booleano (resultado da operação)
remHW	Sim/Sim	Verifica os componentes de *hardware* em outros domínios	(Literal, Literal) (endereço do gerente, endereço do agente)	Literal (Informações)
remSW	Sim/Sim	Verifica os componentes de *software* em outros domínios	(Literal, Literal) (endereço do gerente, endereço do agente)	Literal (Informações)
register	Sim/Sim	Permite o registro em um dos gerentes	(Literal, Literal) (endereço do gerente, endereço do agente)	Booleano (resultado da operação)

Quadro 4.5: Lista de serviços definidos no módulo *Comunicação*

4.3 Arquitetura de Gerenciamento

Como apresentado na seção anterior, o módulo Comunicação é o responsável pela comunicação entre os pontos de medição. Cada administrador local pode determinar quais os módulos de gerenciamento do ponto de medição (*hardware*, sistema e outros) estão disponíveis para integrar a infra-estrutura de medição, quais os serviços estão disponíveis e também a forma de funcionamento do módulo Comunicação.

Este módulo pode trabalhar de acordo com três "papéis" da entidade de gerenciamento: Gerente de Domínio (GD), Gerente de Infra-estrutura (GI) e Gerente de Infra-estrutura Intermediário (GII).

Estas entidades estão organizadas de forma hierárquica como apresentado na figura 4.2. Um GD é um gerente local de um domínio administrativo cujo objetivo é manipular todos os módulos disponíveis na infra-estrutura. Esta é a entidade de mais baixo nível na arquitetura. Um GI é a entidade de nível mais alto, responsável pela requisição e recebimento das respostas sobre os pontos de medição.

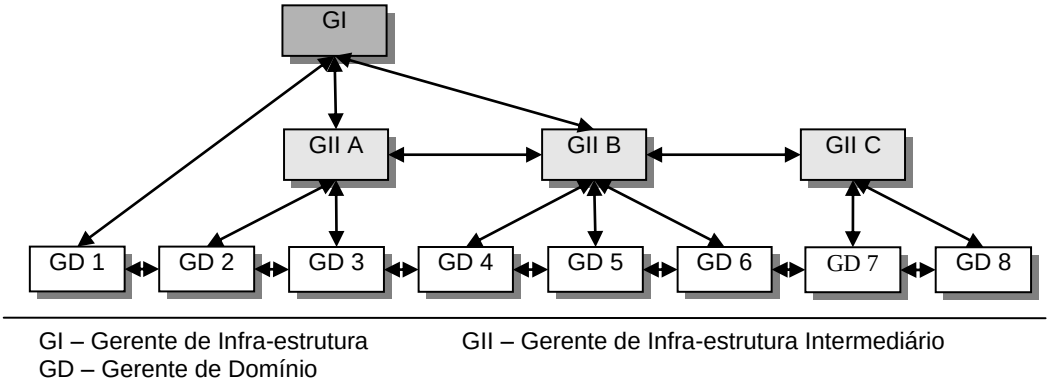

Figura 4.2: Interação entre as entidades de gerenciamento

Considerando uma infra-estrutura de medição em escala global, a centralização em um único gerente superior é possível, mas não aconselhável devido aos problemas de escalabilidade. Para garantir a escalabilidade, foi definida uma entidade intermediária chamada GII que permite receber tarefas dos GI e repassá-las aos GDs. Como ilustrado na figura 4.2, os GD são capazes de se comunicar com outros GDs e com entidades de nível superior, isto é, com as entidades GI ou GII. Uma entidade GII pode comunicar-se com outras entidades GII de mesmo nível, e como entidades de nível superior (GI) ou com GD no nível mais baixo. A comunicação entre os gerentes deve garantir a independência administrativa do domínio. Cada operador de domínio determina quais as interações são possíveis baseado na disponibilidade de seus dispositivos.

Uma característica importante dessa arquitetura é a modularidade e flexibilidade do ponto de medição. Por exemplo, GII A é um gerente de nível superior para o GD 2 e GD 3 e possui as características dessa entidade, mas também ao mesmo tempo possui as características de GD, onde um GI pode ver o GII como um GD.

Ainda, como forma de prover maior modularidade e flexibilidade da arquitetura, cada módulo Comunicação é formado por um conjunto de componentes da arquitetura de gerenciamento. Estes componentes são dissociados dos "papéis" da entidade, isto é, o mesmo ponto de medição pode possuir as características das três entidades. Além

disso, cada componente é projetado de maneira isolada e pode ser habilitado de acordo com a necessidade.

Foram definidos quatro componentes para serem utilizados no módulo Comunicação: Serviço de Infra-estrutura (SI), Agente de Infra-estrutura (AI), Serviço de Domínio (SD) e Agente de Domínio (AD). Os componentes SI e SD expõem as características das entidades GD e GI, respectivamente, enquanto AI e AD são os usados pelos componentes anteriores no desenvolvimento de suas tarefas de gerenciamento. A figura 4.3 apresenta os componentes da arquitetura e o modo como eles estão relacionados.

De modo intuitivo, a entidade GI é composta pelos componentes SI e AI, enquanto a entidade GD inclui os componentes SD e AD. Porém, a entidade GII pode ser formada por todos os componentes projetados e, neste caso, repassar as tarefas de gerenciamento para a entidade GD. O próprio ponto de medição pode gerenciar seus módulos, definindo o modo de funcionamento das suas entidades.

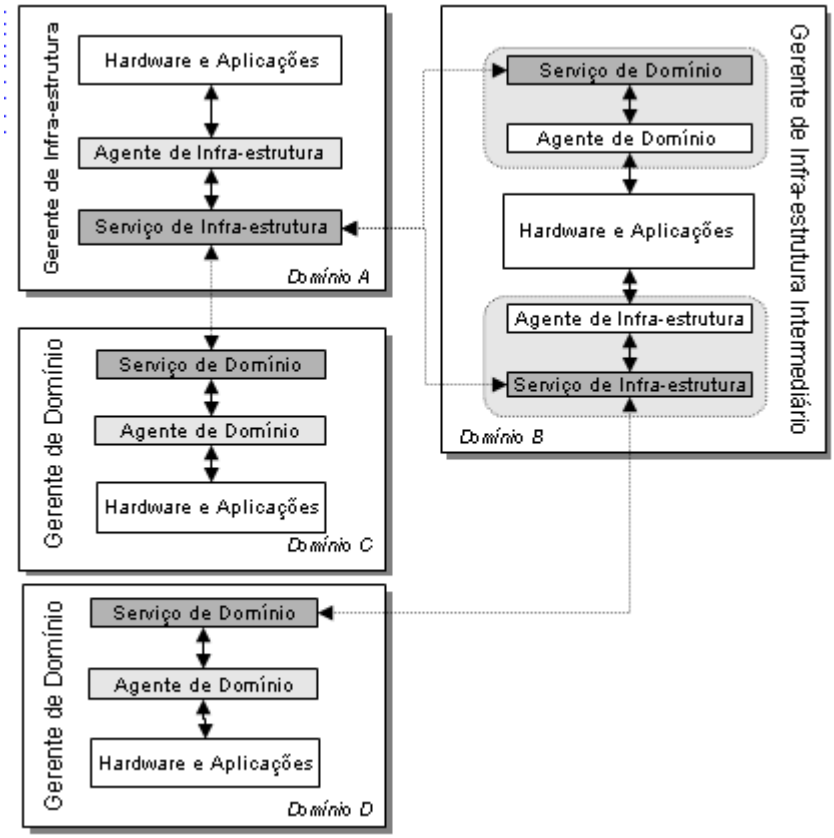

Figura 4.3: Entidades da Arquitetura

A figura 4.3 mostra o contexto onde o GII possui todos os componentes. Neste caso, além de repassar as tarefas de gerenciamento para a entidade GD, a entidade GII na figura também atua como um GD, isto é, pode retornar informações para um GI, em nível superior, baseado na execução dos componentes SD e AD. Cada um dos componentes do módulo Comunicação são descritos em detalhes na seqüência.

4.3.1 Componente Serviço de Infra-estrutura

O componente *serviço de infra-estrutura* (SI) é o responsável pelo gerenciamento de toda a infra-estrutura de medição. As principais funções são: manter as informações

consolidadas de gerenciamento sobre a infra-estrutura; gerenciar informações sobre os integrantes ativos; monitorar seus componentes locais; e notificar outras entidades como GIs, GIIs e GDs sobre atualizações dos elementos internos dos módulos presentes na infra-estrutura.

4.3.2 Componente Serviço de Domínio

O componente *serviço de domínio* (SD) é o responsável pelas tarefas de gerenciamento em seu domínio administrativo. O SD consulta seus agentes periodicamente e efetua ações, se necessário, para garantir a correta operação dos componentes da infra-estrutura de medição. O SD interage com os agentes do domínio de acordo como o modelo centralizado de gerenciamento, isto é, este componente é o responsável por consultar todos os ADs e receber suas respostas. Além disso, deve receber todas as notificações dos ADs.

As tarefas SD não estão restritas a seu domínio. Ele também deve interagir com outros SDs e com o SI. A interação entre SDs de diferentes domínios permite a troca de informações sem a necessidade da intermediação do SI (o que também é possível).

4.3.3 Componente Agente de Domínio

O agente de domínio é o responsável pela monitoração de todos os componentes necessários para o funcionamento do ponto de medição. Estes componentes incluem as aplicações específicas do ponto de medição, as ferramentas do sistema operacional e o *hardware*. A definição do que deve ser monitorado e de que forma é definido pelo SD. O agente deve interagir com as aplicações e o *hardware* do ponto de medição para responder às solicitações do SD. Além da monitoração o agente deve notificar o SD sobre comportamentos anormais dos componentes monitorados.

Na ocorrência de mais de um AD no mesmo domínio administrativo, todos devem reportar as informações para um SD centralizado, dentro do domínio. Isto ocorre quando mais de um ponto de medição está presente no mesmo domínio administrativo.

4.3.4 Componente Agente de Infra-estrutura

Um agente de infra-estrutura possui as mesmas características de um agente de domínio, incluindo a forma de interação com o gerente de infra-estrutura. Além disso, o agente de infra-estrutura deve monitorar os componentes específicos presentes no controle da infra-estrutura de medição. Por exemplo, são considerados componentes específicos do controle as aplicações para receber e consolidar os dados de medição e aplicações para disponibilizar as informações.

4.4 Considerações Parciais

Na literatura não foram encontradas propostas de gerenciamento específicas para infra-estruturas de medição de desempenho, no entanto existem propostas para gerenciamento distribuído, principalmente para *grids*.

Algumas das ferramentas para *grids* foram analisadas para verificar se poderiam ser utilizadas para o gerenciamento de infra-estruturas de medição. As principais são: Ganglia (Matthew, 2004) e Parmon (BUYYA, 2004).

A ferramenta Ganglia foi desenvolvida para monitoração de *clusters* e *grids* de forma distribuída e escalável. Um módulo centralizador coleta e atualiza as informações,

enquanto que cada nó mantém uma cópia do estado corrente do sistema. Os dados coletados podem ser visualizados graficamente através de uma interface *Web*. Com *Ganglia* é possível monitorar qualquer tipo de informação, uma vez que o usuário pode definir métricas específicas através de outra aplicação, além daquelas já coletadas pelo próprio sistema.

A ferramenta *Parmon* foi projetada para ser portável, flexível, interativa, escalável e ambiente compreensível para monitorar grandes clusters, segue a metodologia cliente/servidor e provê acesso transparente a todos os nós, para serem monitorados de uma máquina de monitoração.

As principais carências destas ferramentas é não possuir rotinas para coleta de informações das aplicações, como a quantidade de recursos utilizados. Além disso, utilizam tecnologias proprietárias e um protocolo baseado em *multicast* para difusão dos dados que, pode não passar por roteadores e firewalls.

Em contrapartida a arquitetura proposta foi projetada para ser modular e extensível, isto é, outros módulos, entidades, e componentes podem ser definidos e integrados as modelo. Além disso, a arquitetura proposta dá liberdade para o administrador local inspecionar as aplicações presentes em seu domínio administrativo, o que não ocorre nas ferramentas apresentadas anteriormente.

No próximo capítulo será apresentado o protótipo desenvolvido para a validação do modelo.

5 ASPECTOS DE IMPLEMENTAÇÃO

Como visto anteriormente o uso eficiente das infra-estruturas em larga escala é um objetivo difícil de ser alcançado. O fator determinante reside no fato de que os participantes estão em diferentes domínios administrativos o que gera atrito sobre a segurança de seus domínios entre os administradores.

O capítulo anterior apresentou um modelo de gerenciamento específico para infra-estruturas de medição, um conjunto de funções de gerenciamento, componentes e interações necessárias para o funcionamento das infra-estruturas de medição. O sucesso do modelo de gerenciamento depende da facilidade de administrar os componentes independentemente do domínio administrativo em que se encontram. Neste caso, a tecnologia utilizada na implementação tem papel relevante e, atualmente, a SOA - *Service Oriented Architecture* é a tecnologia que contempla as necessidades do modelo de definição de interfaces para as ações de gerenciamento de forma hierárquica e distribuída.

Uma arquitetura orientada a serviços não é uma tecnologia em si, mas um conjunto de princípios e metodologia para o desenvolvimento de "serviços" que podem ser utilizados através da rede. No caso desta proposta, os serviços são as funções de gerenciamento disponibilizadas entre os gerentes.

Este capítulo contempla os conceitos e o referencial teórico das tecnologias utilizadas na implementação (arquitetura baseada em serviços). No final do capítulo encontra-se o protótipo do sistema de gerenciamento que, desenvolvido em pequena escala, permite comprovar a viabilidade do sistema.

5.1 Serviços *Web*

Os Serviços *Web* (*Web Services*) permitem a integração entre sistemas e compatibilidade de aplicações. Com esta tecnologia é possível que novas aplicações possam interagir eficientemente com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Numa aplicação, cada módulo pode ser de um fornecedor diferente e o usuário monta sua própria solução customizada e, se houver problema em algum módulo, ele não afetará o resto do sistema. Os serviços *web* permitem integrar aplicações, mudar a interface de usuário e evoluir sem reescrever uma única linha de código. De forma resumida, os serviços *web* possibilitam a composição de RPCs (*Remote Procedure Call* ou Chamada a Procedimento Remoto) de objetos pela rede. Os serviços *web* não são a primeira tecnologia a permitir esse tipo de operação, porém diferentemente das demais utilizam padrões neutros de plataforma, como HTTP e XML. As definições completas dos Serviços *Web* estão no anexo C.

5.2 Protótipo

Os capítulos anteriores deram o embasamento teórico necessário para implementar um protótipo capaz de facilitar o gerenciamento de infra-estruturas de medição em uma escala global. Com o auxílio da arquitetura baseada em serviços podemos tornar públicos e de fácil acesso, serviços que antes eram apenas de disponibilidade dos administradores.

O objetivo do protótipo é disponibilizar serviços, utilizando-se dos serviços *Web*, de forma que qualquer aplicação, independente da plataforma ou linguagem de programação, possa fazer parte do sistema de gerenciamento. Portanto, os serviços presentes no protótipo podem ser acessados por qualquer linguagem de programação que tenha suporte a serviços *Web*.

5.2.1 Tecnologias Utilizadas

Um dos principais fatores que tornam os serviços *Web* um padrão em ascendência é o fato das tecnologias utilizadas serem de caráter público e de fácil acesso na *Web*. Partindo deste pressuposto, utilizaram-se tecnologias semelhantes no desenvolvimento do protótipo.

As tecnologias utilizadas são as seguintes:

- Apache2: Servidor *Web* gratuito.
  Obtido no endereço http://httpd.apache.org/.
- Linux SuSE10.1: Sistema Operacional gratuito.
  Obtido no endereço http://www.opensuse.com.
- PHP: Linguagem de programação livre.
  Obtida no endereço http://www.php.net.
- MySQL: sistema gerenciador de banco de dados gratuito.
  Obtido no endereço http://www.mysql.com.
- OpenSSL: implementação de código aberto dos protocolos SSL e TLS.
  Obtido no endereço http://www.openssl.org

A utilização da linguagem PHP neste trabalho se deve a quatro principais qualidades desta linguagem: velocidade, estabilidade, segurança e simplicidade.

O PHP passou a ter suporte aos serviços *Web* a partir da versão 5, através da extensão SOAP. Esta extensão suporta as especificações SOAP 1.1, SOAP 1.2 e WSDL 1.1, permitindo a construção e utilização dos serviços *Web*.

5.2.2 Ambiente de Testes

Para os testes do protótipo foram utilizados equipamentos presentes na Universidade Regional do Noroeste do Estado do Rio Grande do Sul – UNIJUI, nos diversos campi universitários. As cidades escolhidas para participarem do protótipo foram Ijuí, Panambí, Santa Rosa e Três Passos. Em cada uma destas cidades existe um administrador local da rede e representam no protótipo os diferentes domínios administrativos.

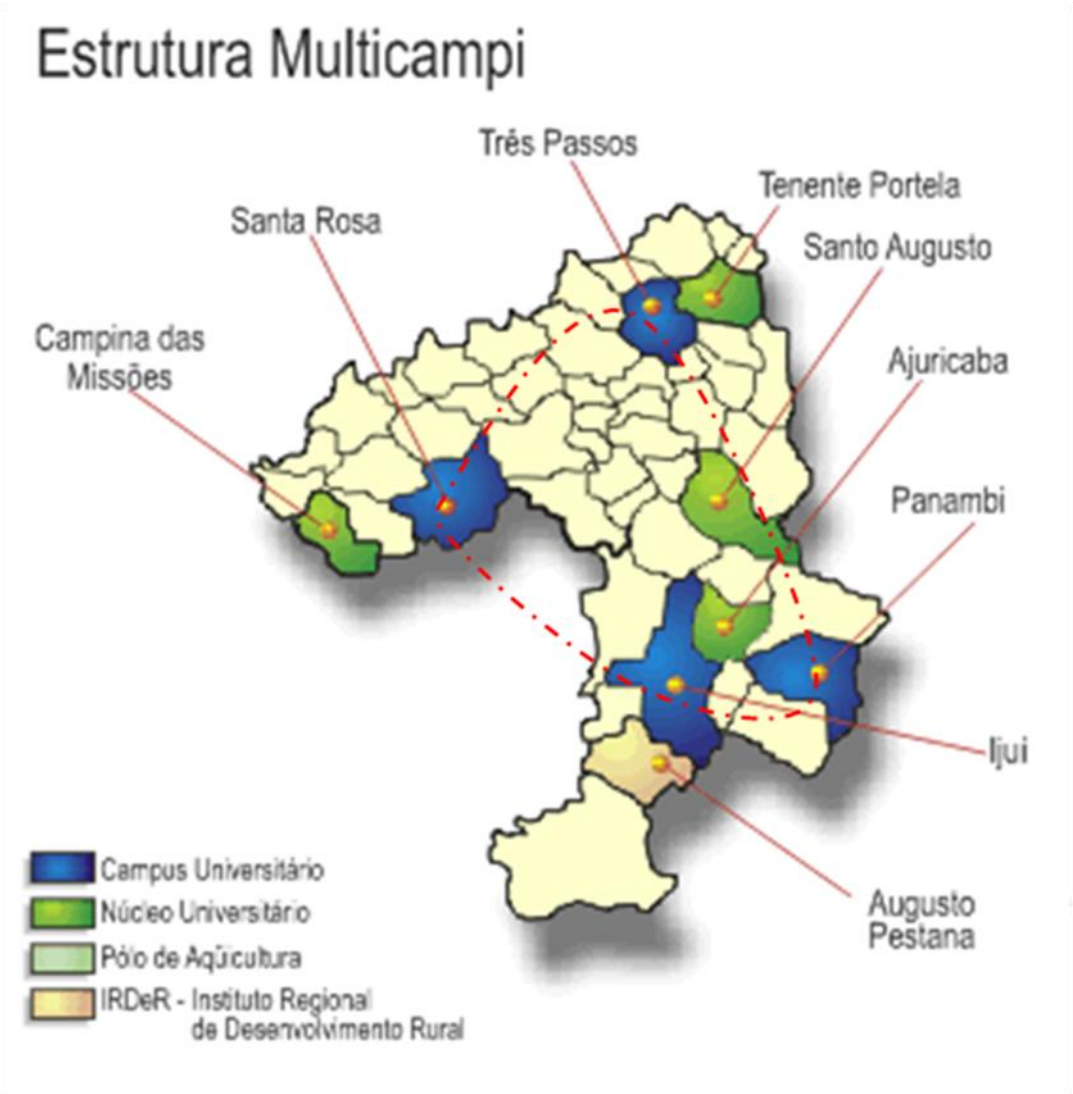

Figura 5.1: Cidades Utilizadas para Testes no Protótipo

Como pode ser visto na figura abaixo, cada equipamento foi configurado em um nível de gerenciamento descrito neste trabalho, tendo como Gerente de Infra-Estrutura principal a máquina presente no campus UNIJUI – Santa Rosa.

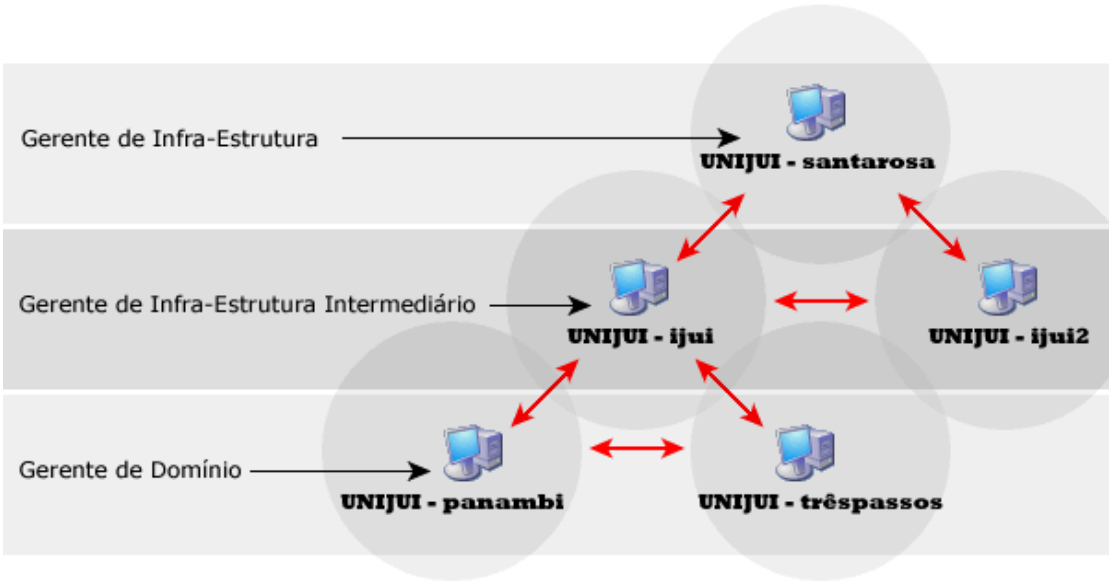

Figura 5.2: Ambiente de testes do protótipo

Para os testes foi escolhida a infra-estrutura de medição pIPEs (apresentada na seção 3.9). Foram instalados em todos os equipamentos os *softwares* necessários para o funcionamento da infra-estrutura (OWAMP, BWCTL, IPERF).

5.2.3 Aspectos de Segurança

Devido a não preocupação do protocolo SOAP quanto à segurança no tráfego de informações presentes nas suas mensagens, como visto anteriormente neste trabalho, foi necessária implementação de uma classe capaz de preencher esta lacuna.

Primeiramente, foi implementada uma classe utilizando-se da biblioteca *mcrypt*, presente no PHP5, para gerar chaves simétricas, as quais estariam disponibilizadas em todos os pontos de medição dos domínios administrativos. Apesar de satisfatórios resultados, esta implementação não é segura o suficiente por utilizar outros meios sem segurança para a troca das chaves.

A solução encontrada para este problema foi a utilização em conjunto das chaves assimétricas e simétricas. Como visto na figura 5.3 e figura 5.4, primeiramente a chave simétrica é gerada de forma independente em cada gerente de domínio. Anteriormente ou posteriormente é solicitado em forma de um serviço o recebimento da chave pública ao gerente de domínio superior ao qual se está cadastrado.

Figura 5.3: Recebimento da chave pública assimétrica

Após, utilizando-se a chave pública assimétrica recebida do gerente de domínio superior, o enviamos a chave simétrica criptografada através de um novo serviço, onde, por sua vez o gerente de domínio superior a salva no respectivo campo de cadastro do gerente emissor.

Figura 5.4: Envio da chave simétrica ao gerente de domínio superior.

Sempre que o gerente de domínio gerar novas chaves assimétricas, a chave pública assimétrica é repassada através de um serviço a todos os gerentes de domínio inferiores cadastrados, facilitando o sincronismo das informações e evitando um custo elevado na troca de informações.

Por fim, os serviços que serão utilizados com segurança são criptografados com a chave simétrica do respectivo emissor. Este solicita um serviço ao gerente de domínio superior, que localiza em sua base de dados a respectiva chave, criptografa os resultados e a retorna. A figura 5.5 apresenta a troca de mensagens de forma segura, usando chaves simétricas.

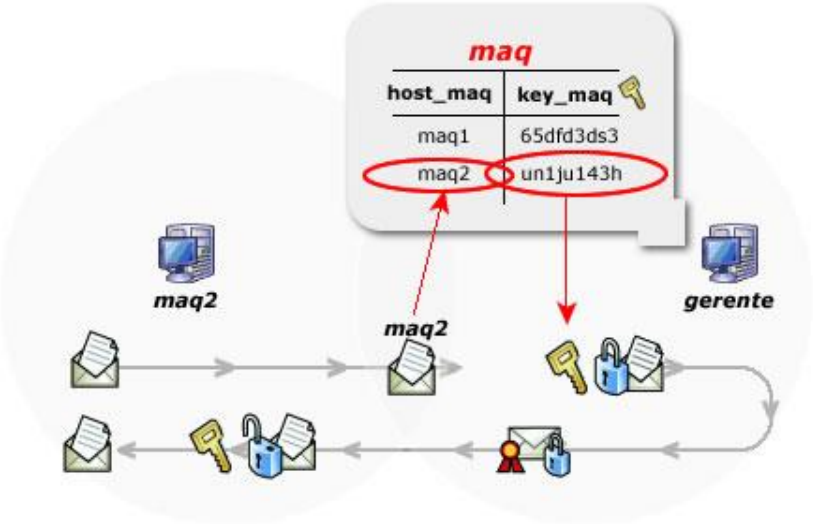

Figura 5.5: Exemplo de solicitação de serviços com mensagens criptografadas com chaves simétricas.

A figura 5.6 apresenta um exemplo da utilização do OpenSSL na geração das chaves assimétricas através da linguagem de programação PHP, onde a palavra “teste” foi

criptografada com a chave pública gerada e decriptografada utilizando-se a chave privada.

Figura 5.6: Exemplo de Criptografia utilizando o OpenSSL e PHP.

5.2.4 Modelo de Implementação dos Serviços

Após a confirmação do registro, o gerente de domínio pode requisitar a descrição dos serviços que o mesmo deve implementar (arquivo WSDL) para poder atuar no sistema de gerenciamento ou pode utilizar os códigos já desenvolvidos. Seguindo os padrões definidos pelos serviços *web*, os serviços disponibilizados pelos gerentes de domínio podem ser implementados em qualquer linguagem de programação.

Para exemplificar, o trecho de código abaixo apresenta como o serviço que retorna as informações do ponto de medição (*getInfo*) é implementado no servidor.

O código está dividido em três partes: a primeira cria uma instância de um servidor SOAP; a segunda é o serviço propriamente dito; e, a terceira informa ao servidor que deve passar a suportar o serviço informado.

Os valores retornados pelo serviço *getIinfo* são informações obtidas da MIB usando o protocolo de gerenciamento SNMP.

```
<?php
      // → parte 1
     //cria uma instância do servidor
      $server = new SoapServer
            (null, array('uri' => "http://guru.unijui.tche.br/ws/"));

      // → parte 2
      //definição do serviço que retorna informações sobre o PMD
      function getInfo() {
            $host = "localhost";
```

```
            $community = "public";
            snmp_set_quick_print(1);

            $sysName = snmpget($host, $community, "sysName.0");
            $sysDescr = snmpget($host, $community, "sysDescr.0");
            $sysLocation = snmpget($host, $community, "sysLocation.0");
            $sysContact = snmpget($host, $community, "sysContact.0");
            $resp =
                  "sysName#".$sysName.
                  "|sysDescr#".$sysDescr.
                  "|sysLocation#".$sysLocation.
                  "|sysContact#".$sysContact;
            return $resp;
      }

      // → Parte 3
$server->addFunction("getInfo");

if ($_SERVER["REQUEST_METHOD"]== "POST") {
            $server->handle();
            } else {
            $functions = $server->getFunctions();
            foreach ($functions as $func) {
                  echo $func. "<br>";
            }
      }
?>
```

O trecho de código abaixo apresenta como um cliente pode fazer uso de um serviço disponível. Para isso basta direcionar o cliente para o endereço do serviço (parte inicial do código) e efetuar uma chamada para a função desejada (segunda parte). O tratamento dado para as respostas de determinado serviço é uma questão de implementação e, neste caso, foi suprimido por questões de visibilidade.

As informações sobre os serviços disponíveis, parâmetros de entrada e tipo de retorno podem ser definidos dinamicamente via WSDL.

```
try {
            //→ parte 1
            $client   =   new   SoapClient(null,   array('location'   =>
            "http://guru.unijui.tche.br/ws/serverSOAP.php",
            'uri' => "http://guru.unijui.tche.br"/ws",
            'trace'    => 1,
            "connection_timeout"=>2));

      try {
            // → parte 2
                  $result = $client->getInfo();

            } catch (SoapFault $exception) {}

} catch (SoapFault $exception) {}
```

A Figura 5.7 apresenta a tela do protótipo com a resposta do sistema para este serviço.

Figura 5.7: Protótipo - Informações locais do Gerente de Domínio

5.2.5 Instalação

De acordo com a proposta de gerenciamento, a definição de um gerente de infra-estrutura é o ponto de partida na constituição da infra-estrutura de medição. Este gerente deve implementar serviços que permitam aos demais gerentes registrarem-se para integrar a infra-estrutura. O gerente de infra-estrutura pode redirecionar o registro para um gerente intermediário (para evitar sobrecarga).

A primeira ação a ser feita assim que o protótipo é salvo em sua máquina, é rodar o arquivo “install.php”, presente em sua pasta de origem.

Figura 5.8: Tela de Instalação do Protótipo

A figura 5.8 apresenta a tela de instalação do protótipo. Nela um formulário solicita os dados básicos para configuração do sistema e criação da base de dados. Ao preencher os dados e enviá-los, um arquivo denominado "conecta.php" é criado com as informações necessárias para a conexão com o banco de dados, as tabelas do banco são criadas com os respectivos dados de configuração e, por último, o arquivo "install.php" é excluído para uma maior segurança.

5.2.6 Configuração

Após a instalação do protótipo, o primeiro passo a ser efetuado é a sua configuração. A figura 5.9 apresenta a tela do sistema, onde na lateral esquerda da tela inicial, encontra-se o *menu* de opções, nele possui um item denominado de "Configurações". Ao clicar neste item é aberto um formulário para preenchimento com as informações necessárias.

Figura 5.9: Tela de Configuração.

O item Gerente Superior é preenchido quando deseja conectar-se a um gerente superior. É necessário cadastrar-se primeiramente neste Gerente Superior, aguardar o recebimento de uma confirmação via e-mail do seu cadastro. Após esta confirmação, é preenchido o endereço no formulário de Configurações, que é salva no banco de dados.

Assim que o cadastro estiver efetuado, ao clicar no item "*Update*", no mesmo quadro da página de configurações, é solicitado um serviço ao endereço do Gerente Superior cadastrado. Este serviço tem como objetivo retornar a chave pública vigente, item indispensável para a troca de informações seguras entre as infra-estruturas.

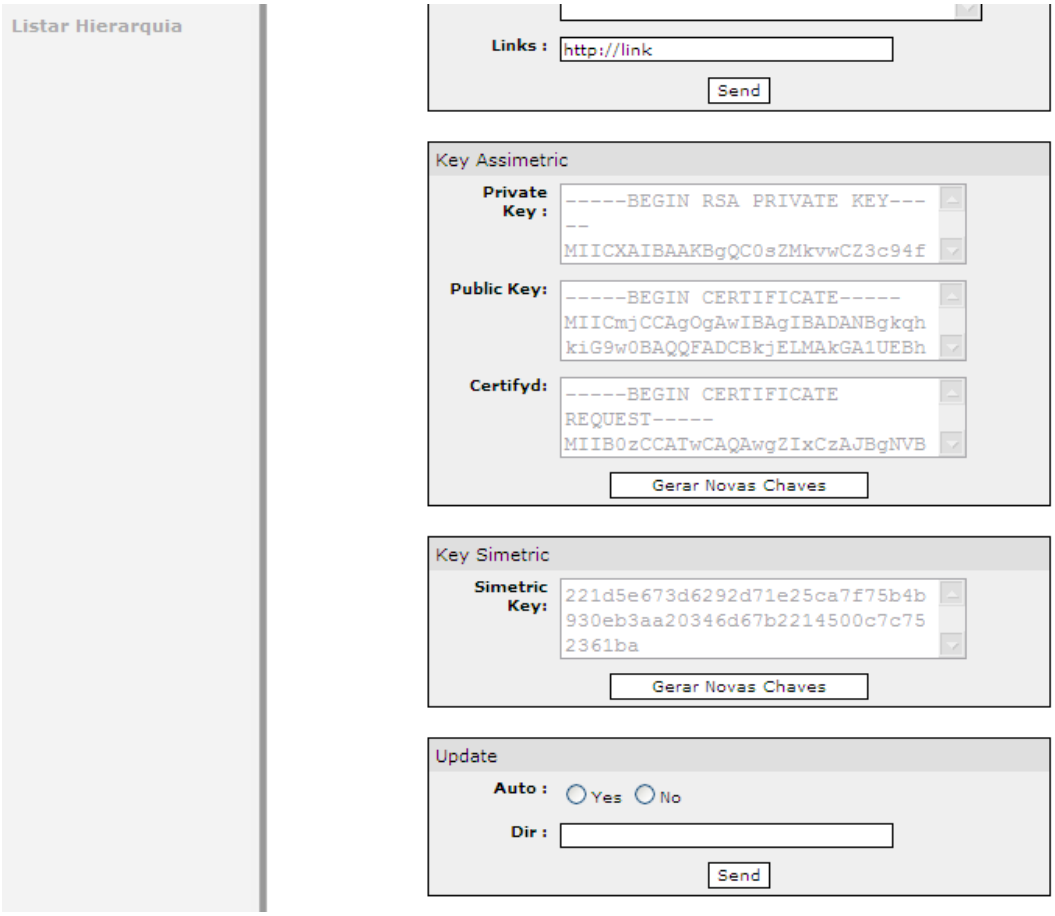

Figura 5.10: Continuação da Tela de Configurações.

Outro passo importante para a troca de informações é a geração das chaves assimétricas e simétricas. Para uma troca segura de informações entre os gerentes é obrigatório.

Ao gerar as chaves, clicando nos ícones “Gerar Novas Chaves”, é questionado se deseja enviar as mesmas aos respectivos membros aos quais elas são designadas. No caso da chave simétrica, ao Gerente Superior, e a chave pública (assimétrica) aos membros inferiores a sua infra-estrutura. O mesmo processo se equivale para a atualização das informações de configuração, como visto na figura 5.10.

5.2.7 Inclusão dos Pontos de Medição

Quando o ponto de medição solicita a sua inclusão no sistema de gerenciamento, a solicitação aparece na janela principal do gerente a qual a solicitação foi feita.

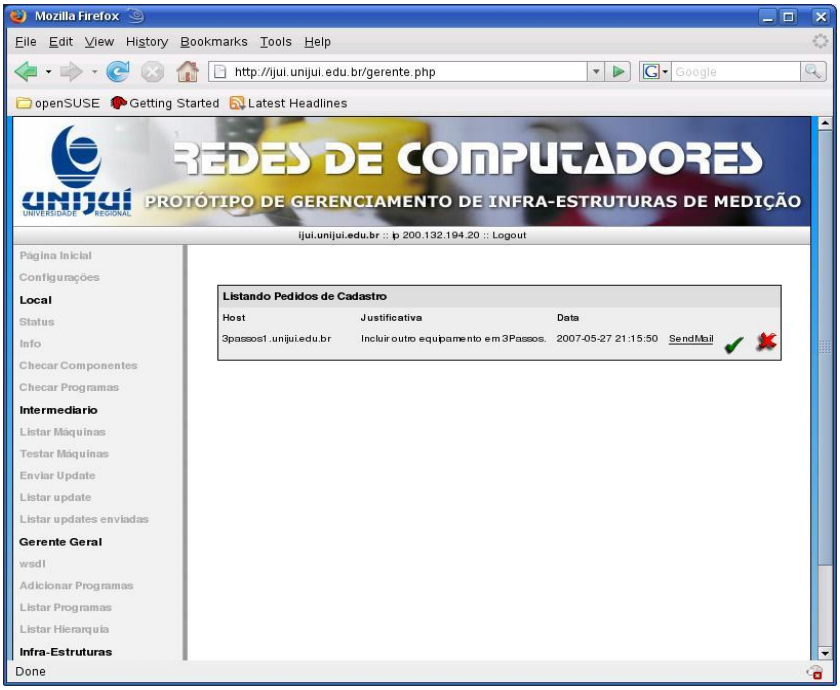

Figura 5.11: Pedido de inclusão de um novo ponto de medição

A figura 5.11 apresenta a janela onde o gerente tem a opção de aceitar ou excluir a solicitação ou enviar um e-mail para o solicitante (solicitando maiores informações, por exemplo). Quando o gerente aceita a solicitação de inclusão é redirecionado para enviar um e-mail para o solicitante informando do aceite. Após o aceite, o ponto de medição faz parte do sistema, porém ainda de forma não ativa. O gerente deve ativá-lo para que o mesmo possa fazer parte do gerenciamento da infra-estrutura. A figura 5.12 apresenta a tela do sistema que lista os pontos de medição presentes nas infra-estruturas, ativos e inativos.

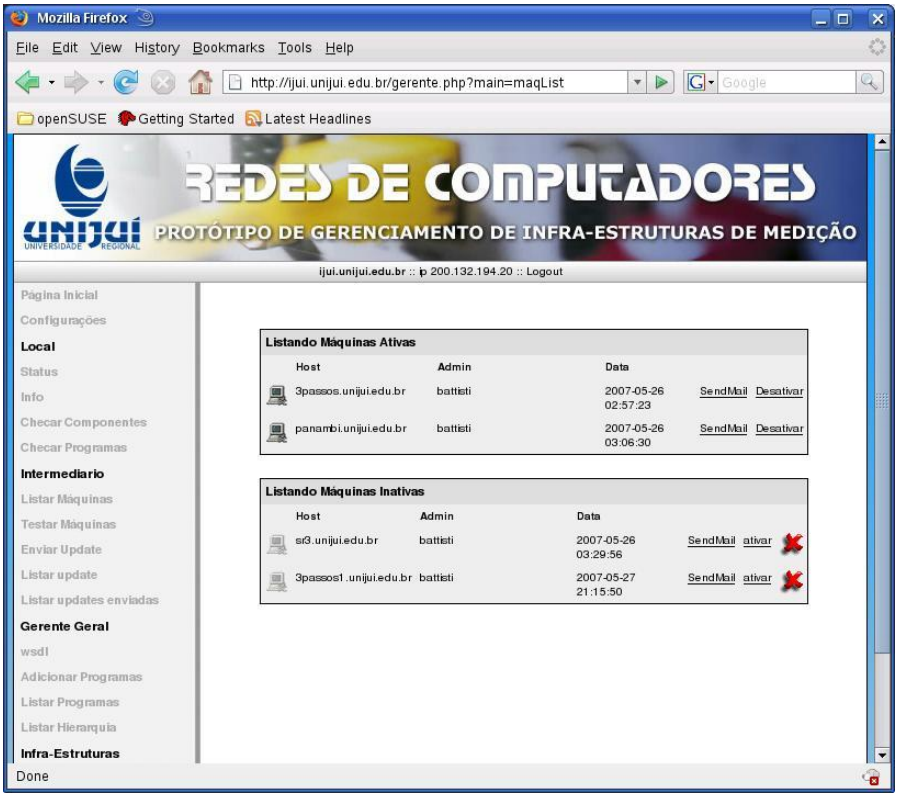

Figura 5.12: Ativar um ponto de medição

O gerente de infra-estrutura pode visualizar todos os pontos de medição presentes no sistema de gerenciamento pela opção listar hierarquia. A figura 5.13 apresenta esta opção, nela pode-se ver os gerentes intermediários e os gerentes de domínio.

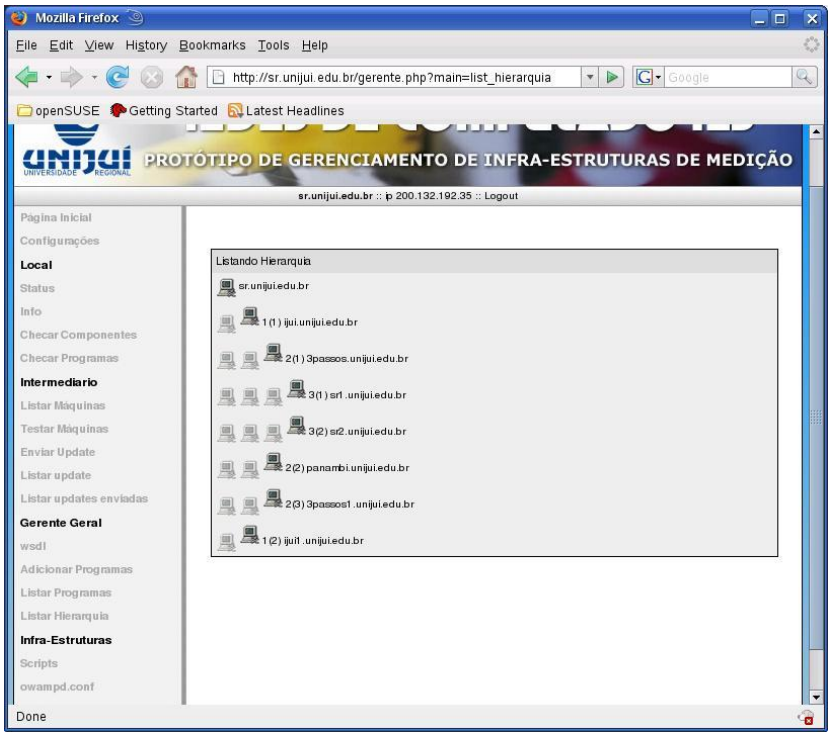

Figura 5.13: Hierarquia dos pontos de medição no ambiente de testes

A figura 5.14 mostra a possibilidade de alteração nos arquivos de configuração das ferramentas de testes. Neste caso específico, a configuração do *owampd* pode ser feita pelo formulário na página ou pelo envio de um arquivo contendo a configuração.

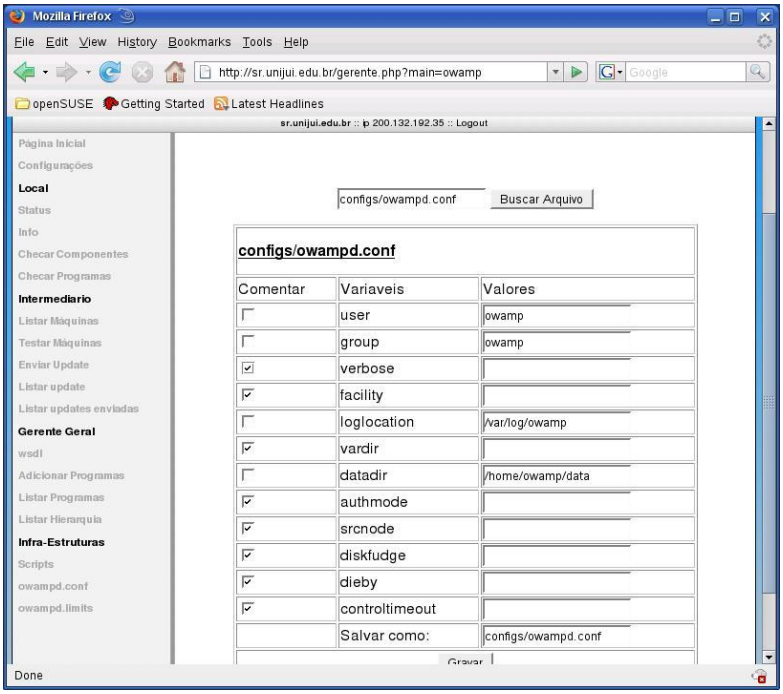

Figura 5.14: Atualização do arquivo de configuração do *owamp*

5.2.8 Avaliação do Protótipo

O ambiente de testes foi configurado e foram efetuados vários testes na forma de simulação de problemas. Nestes testes foram simulados problemas em aplicações e em componentes de *hardware*, nestes casos, o sistema reportou os problemas corretamente.

Após o período de simulação o ambiente de testes avaliado durante 60 dias em "situação real", ou seja, sem nenhum tipo de simulação, apenas avaliando os acontecimentos no uso corriqueiro dos sistemas. Durante este período de avaliação destacam-se três eventos que demonstram o funcionamento do protótipo.

O primeiro evento anormal detectado pelo protótipo foi a ausência de resposta do equipamento "*ijui1*". A figura 5.15 mostra que o acesso aos serviços *web* não estavam respondendo no referido equipamento. Ao iniciar o processo de verificação dos *softwares* do equipamento percebeu-se a ausência total de acesso remoto. Este fato exigiu um contato com o administrador do laboratório em que o equipamento estava fisicamente localizado. Uma pequena verificação e constatou-se que o cabo de rede do equipamento estava com problemas, após a substituição do mesmo, tudo voltou a funcionar corretamente.

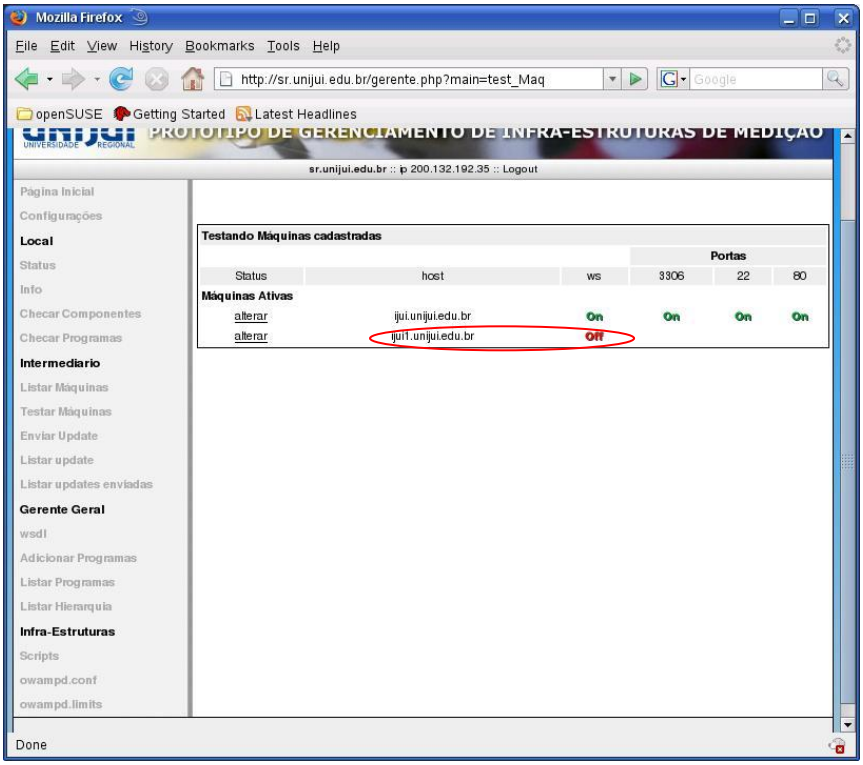

Figura 5.15: Falha no equipamento *ijui1*

O segundo evento de falha ocorreu com o servidor do serviço de *login* remoto (SSH). O equipamento era de uso múltiplo, outros usuários também utilizam o acesso remoto ao equipamento via ssh. Acredita-se que em um destes acessos tenha ocorrido um erro, finalizando a execução do processo. A causa da falha não foi diagnosticada, porém a

solução foi simplesmente a reinicialização do processo. A figura 5.16 apresenta como o erro foi apresentado no protótipo.

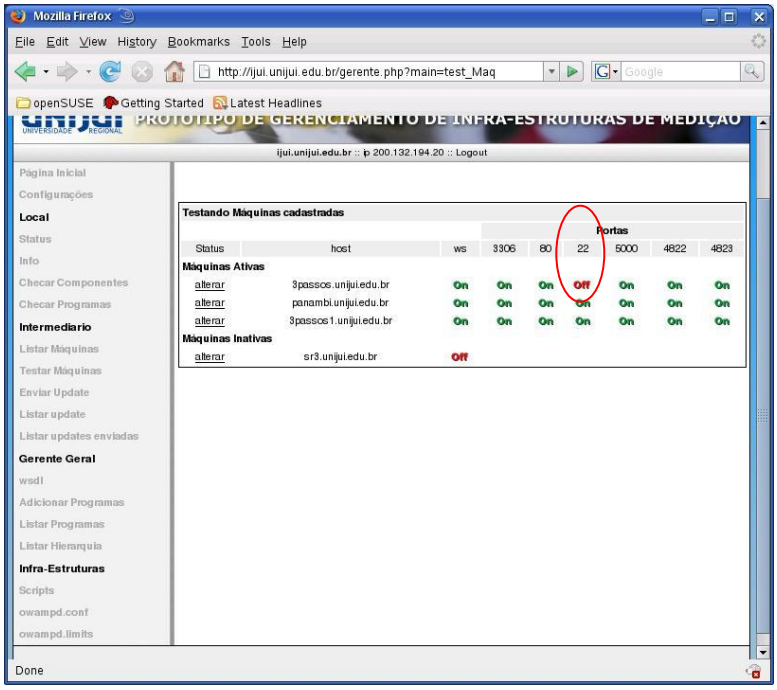

Figura 5.16: Falha no *ssh* no equipamento *3passos*

O terceiro evento foi a detecção da falha nos programas servidores das ferramentas de testes da infra-estrutura. A figura 5.17 e figura 5.18 apresentam a detecção do problema nos programas servidores IPERF, Owampd e Bwctl. Como os serviços *web* e outros servidores estavam respondendo corretamente ficou claro o problema na inicialização dos servidores das ferramentas de testes. O problema foi detectado no script de inicialização de todos os programas servidores das ferramentas de testes da infra-estrutura. Foram feitas atualizações no script, porém não foi atualizado neste ponto de medição.

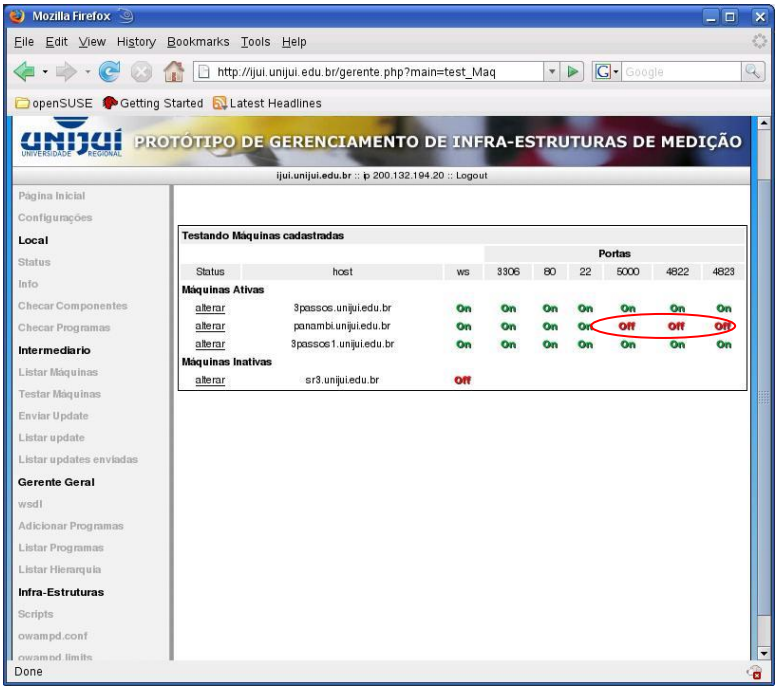

Figura 5.17: Detecção da falha no equipamento *panambi* pelo gerente superior

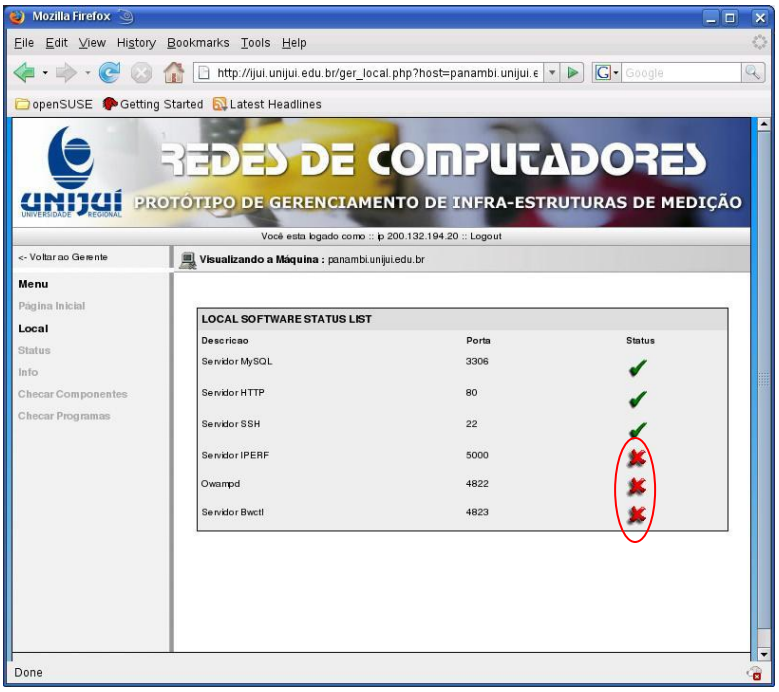

Figura 5.18: Verificação da falha pelo gerente local (*panambi*)

Antes deste período de avaliação em ambiente real, foram efetuados vários testes na forma de simulação de problemas. Nestes testes foram simulados problemas em aplicações e em componentes de *hardware*, nestes casos, o sistema reportou os problemas corretamente.

No período de avaliação real, porém não foram detectados problemas de *hardware* nos pontos de medição. Evidentemente, os componentes de *hardware*, por serem menos dinâmicos estão menos propensos a problemas. Acredita-se que em um período mais longo de avaliação certamente ocorreriam problemas de hardware e seriam detectados pelo sistema, assim como no processo de simulação.

6 CONSIDERAÇÕES FINAIS

A compreensão das características de desempenho dos serviços e aplicações na Internet é um processo importante na evolução das técnicas e protocolos, visando atender as demandas dos usuários. Sabe-se que uma infra-estrutura de medição é útil para uma melhor compreensão do comportamento da rede. Por meio dela pode-se determinar um padrão de comportamento da rede e com isso detectar anomalias. Além disso, o registro histórico do padrão de comportamento permite avaliar tendências, facilitando o planejamento das capacidades da rede.

Para a melhor compreensão dos testes realizados pelas infra-estruturas de medição para inferir o comportamento da rede, foi apresentado no início do trabalho um conjunto de métricas de desempenho.

Apesar da existência de pontos favoráveis para a utilização de infra-estruturas de medição, elas ainda estão restritas a infra-estruturas de testes ou a pequenos grupos com interesses comuns. O uso em larga escala das infra-estruturas de medição somente vai ocorrer quando estas atenderem às necessidades de usuários e administradores de rede de uma forma eficiente.

Neste trabalho foi investigado um conjunto de infra-estruturas de medição, apresentando suas características, seu funcionamento e suas deficiências quanto ao gerenciamento de seus componentes. De forma geral as infra-estruturas avaliadas não estavam preparadas para uso em larga escala e não gerenciavam seus componentes. Esta inabilidade ou a despreocupação com o gerenciamento acaba dificultando a sua manutenção e conseqüentemente restringindo a sua utilização.

Para suprir a lacuna de gerenciamento existente nas infra-estruturas de medição, foi proposto um modelo de gerenciamento específico para estas infra-estruturas. A forma distribuída e autônoma do gerenciamento proposto procura minimizar a principal dificuldade de ampliação das infra-estruturas de medição, ou seja, o uso integrado em diferentes domínios administrativos. Nesse caso, o modelo prevê que cada administrador local gerencie a sua parte da infra-estrutura e disponibilize para os demais gerentes funções que permitam um funcionamento integrado e colaborativo de toda a infra-estrutura de medição.

De forma evidente, a tecnologia utilizada na implementação da proposta de gerenciamento tem fator preponderante para alcançar os objetivos. Neste ponto a arquitetura orientada a serviços contempla facilmente todas as necessidades do modelo. Como visto no capítulo 5 uma arquitetura orientada a serviços não é uma tecnologia em si, mas um conjunto de princípios e metodologia para o desenvolvimento de “serviços” que podem ser utilizados através da rede. Essa metodologia de desenvolvimento foi utilizada na construção do protótipo.

O protótipo construído permite observar a facilidade com que os serviços *web* podem ser utilizados para integrar aplicações presentes em diferentes domínios. Uma vez definidos os serviços que devem ser prestados pelo ponto de medição, os administradores locais tem liberdade para construir a sua própria implementação. É necessário apenas que ele observe o conjunto de entradas e a saídas que devem ser produzidas.

Apesar de ser possível o desenvolvimento dos próprios códigos para compor o sistema, foram utilizados em todos os locais os códigos desenvolvidos no protótipo. Os códigos foram bem aceitos pelos administradores pelo fato de serem baseados em tecnologias abertas e com a possibilidade de inspeção dos códigos fontes. Evitando dúvidas sobre o funcionamento dos mesmos, principalmente sobre segurança.

O ambiente de utilização do protótipo conseguiu reproduzir fielmente um ambiente multidomínios, uma vez que em cada cidade (campus) onde foram instalados os equipamentos possuem um administrador (humano) local.

Durante o período de avaliação do protótipo alguns eventos (apresentados na seção 5.4.9) indicam que o sistema é eficiente na detecção de anomalias nos componentes do ponto de medição.

As falhas ocorreram nos componentes de *software* e não nos componentes de *hardware*. De fato, os componentes de *hardware* exigem um tempo maior de acompanhamento para capturar a ocorrência de eventos de falha, uma vez que os componentes são mais estáticos. Devido ao maior dinamismo dos componentes de *software*, isto é, processos de instalação, atualização e alteração de configuração eles são mais propensos a apresentarem falhas.

Após a detecção das falhas o processo de solução foi rápido, no contato com o administrador local bastava informar a solução, pois a falhas já tinha sido identificada, agilizando a solução. Em apenas um dos casos o administrador local teve que identificar o problema (falha no cabeamento).

Finalmente, os serviços *web*, baseados unicamente em tecnologias não proprietárias, mostraram-se estáveis, não causando nenhum tipo de erro ou falha no protótipo pelo uso dessa tecnologia. O comportamento do protótipo foi dentro do esperado para a validação da proposta, atendendo aos requisitos do modelo de gerenciamento, garantindo o bom funcionamento da infra-estrutura de medição.

6.1 Trabalhos Futuros

Como trabalhos futuros relacionamos o aprimoramento dos serviços atuais, o incremento do número de serviços implementados pelo protótipo e a simplificação do suporte a novas infra-estruturas.

Alguns serviços já implementados podem ser aprimorados, como por exemplo, a possibilidade de manter e manipular diversos arquivos de configuração para os testes. Isso pressupõem a alteração dos parâmetros de execução dos testes que pode ser feita dinamicamente com o uso de scripts ou chamadas de sistema dentro da linguagem de programação.

O incremento no número dos serviços disponível no protótipo é necessário para atender a todos os serviços previstos no capítulo 4. Apesar de alguns serviços previstos

serem considerados de implementação não obrigatória para os gerentes de domínio, eles devem ser implementados e testados para futura distribuição.

A simplificação do processo de inclusão de novas infra-estruturas no sistema de gerenciamento. Essa simplificação pode ser alcançada com o uso de uma interface mais genérica que permita o gerente superior descrever todos os componentes gerenciáveis da nova infra-estrutura e repassar essas informações em bloco para os gerentes intermediários de forma automática.

REFERÊNCIAS

ADAMS, A.; MATHIS, M. A System for Flexible Network Performance Measurement. In: INET, 2000, Yokohama, Japan. **Proceedings…**. [S.l.:s.n.], 2000. Disponível em: <http://www.isoc.org/isoc/conferences/inet/00/cdproceedings/1d/1d_1.htm>. Acesso em: mar. 2005.

ALMES, G.; KALIDINDI, S.; ZEKAUSKAS, M. **A One-way Delay Metric for IPPM:** RFC 2679. [S.l.]: Internet Engineering Task Force, Network Working Group, 1999.

ALMES, G.; KALIDINDI, S.; ZEKAUSKAS, M. **A One-way Packet Loss Metric for IPPM:** RFC 2680. [S.l.]: Internet Engineering Task Force, Network Working Group, 1999.

ALMES, G.; KALIDINDI, S.; ZEKAUSKAS, M. **A Round-trip Delay Metric for IPPM:** RFC 2681. [S.l.]: Internet Engineering Task Force, Network Working Group, 1999.

BLUMENTHAL, U.; WIJNEN, B. **The User-Based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3)**: RFC 3414. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

BOOTE, L. et al. **Deliverable DJ.1.2.1:** GEANT2 General Monitoring Framework Design. Disponível em: <http://wiki.perfsonar.net/jra1-wiki/images/9/95/GN2-05-057v5.pdf> Acesso em: maio 2007.

BRISA. **Arquitetura de Redes de Computadores OSI e TCP/IP**. São Paulo: Makron Books, 1994. 669p.

BRISA. **Gerenciamento de Redes** : uma abordagem de sistemas abertos. São Paulo: Makron Books, 1993.

BUYYA, R. PARMON: A portable and scalable monitoring system for clusters. ***Software*** **Practice and Experience**, London, v. 30, n. 7, p.723–739, 2000.

CASE, J. et al. **Simple Network Management Protocol**: RFC 1157. [S.l.]: Internet Engineering Task Force, Network Working Group, 1990.

CASE, J. et al. **Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)**: RFC 3412. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

CASE, J. et al. **Introduction to Community-based SNMPv2**: RFC 1442, [S.l.]: Internet Engineering Task Force, Network Working Group, 1996.

CASE, J. et al. **Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)**: RFC 1442, [S.l.]: Internet Engineering Task Force, Network Working Group, 1993.

CHOI, M. J. et al. XML-based configuration management for IP network devices. **IEEE Communications Magazine**, New York, v.42, n.7, p.84-91, 2004.

CISCO. **Internetworking Technology Handbook**. Disponível em: <http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/index.html> Acesso em: jul. 2005.

COTTRELL, L.; LOGG, C. **Overview of IEPM-BW Bandwidth Testing of Bulk Data Transfer**. 2003. Disponível em: <http://www.slac.stanford.edu/cgi-wrap/getdoc/slac-pub-9202.pdf>. Acesso em: mar. 2005.

COTTRELL, L.; MATTHEWS, W.; LOGG, C. **Tutorial on Internet Monitoring & PingER at SLAC.** Disponível em: <http://www.slac.stanford.edu/comp/net/wan-mon/tutorial.html#pinger>. Acesso em: mar. 2005.

DEMICHELIS, C.; CHIMENTO, P. **IP Packet Delay Variation Metric for IP Performance Metrics (IPPM):** RFC 3393, [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

DOVROLIS, C.; MOORE, D. Packet-Dispersion Techniques and a Capacity-Estimation Methodology. **IEEE/ACM Transactions on Networking**, Piscataway, v.12, n.6, p.963-977, Dec. 2004.

E2EPI. **BWCTL:** Bandwidth Control. Disponível em: <http://e2epi.internet2.edu/bwctl/> Acesso em: nov. 2004.

E2EPI. **OWAMP:** One-Way Active Measurement Protocol. Disponível em: <http://e2epi.internet2.edu/owamp/ > Acesso em: nov. 2004.

E2EPI. **piPES:** Internet2 E2E Performance Initiative Performance Environment System. Disponível em: <http://e2epi.internet2.edu/E2EpiPEs/index.html> Acesso em: out. 2004.

FIOREZE, T.; GRANVILLE, L. Z.; ALMEIDA, M. J.; TAROUCO, L.R. Comparing *web* services with SNMP in a management by delegation environment. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM. IM, 9., 2005, Nice, France. **Integrated Network Management IX**. Piscataway: IEEE, 2005. p.601-614.

GEORGATOS, F. et al. Providing Active Measurement as a Regular Service for ISP's. In: A WORKSHOP ON PASSIVE AND ACTIVE MEASUREMENTS, 2001, Amsterdam. **Proceedings…** Disponível em: <http://www.ripe.net/projects/ttm/Documents/Papers/PAM2001.pdf>. Acesso em: mar. 2005.

GOLDSZMIDT, G.; YEMINI, Y. Distributed management by delegation. In: INTERNATIONAL CONFERENCE OF DISTRIBUTED COMPUTING SYSTEMS, 1995. **Proceedings …**[S.l.:s.n.], 1995. p. 333-340.

HANEMANN, A. et al. Perfsonar: A service oriented architecture for multi-domain network monitoring. In: INTERNATIONAL CONFERENCE ON SERVICE-ORIENTED COMPUTING, ICSOC, 3., 2005. **Proceedings…** Berlin: Springer, 2005. p. 241-254. (Lecture Notes in Computer Science, v.3826).

HARRINGTON, D.; PRESUHN, R.; WIJNEN, B. **An Architecture for describing Simple Network Management Protocol (SNMP) Management Frameworks**: RFC 3411. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

HEMMEN, L.J.G.T. Models Supporting the Network Management Organization. **International Journal of Network Management**, New York, v.10, n.6, p. 299-314, Nov. 2000.

HENDRICKS, M. et al. **Profissional Java *Web* Services.** Rio de Janeiro: Alta Books, 2002.

IEPM. **BW:** Methodology. Disponível em: <http://www-iepm.slac.stanford.edu/bw/methodology.html>. Acesso em: mar. 2005.

INTERNET2. **E2EPI:** Internet2 End-to-End Performance Initiative. Disponível em: <http://e2epi.internet2.edu> Acesso em: out. 2004.

ISO. **ISO 8824.** Specification of Abstract Syntax Notation One (ASN.1). Suiça, 1987.

ITU-T. **Recommendation G.114:** One-Way Transmission Time. Suiça, 2000. Disponível em: <http://ftp.tiaonline.org/tr-41/tr411/Public/2003-05-LakeBuenaVista/TR41.1-03-05-057L-Draft-ITU-TG.114.doc>. Acesso em: mar. 2003.

JUNIPER NETWORKS. **XML-based Network Management**. 2001. Disponível em: <http://www.juniper.net/solutions/literature/white_papers/200017.pdf>. Acesso em: 15 jan. 2008.

KALIDINDI, S.; ZEKAUSKAS, M. J. **Surveyor:** An Infrastructure for Internet Performance Measurements, 1999. Disponível em: <http://www.isoc.org/inet99/proceedings/4h/4h_2.htm>. Acesso em: abr. 2005.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet:** uma nova abordagem. São Paulo: Pearson, 2004.

LAI, K.; BAKER, M. Measuring link bandwidths using a deterministic model of packet delay. In: CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES AND PROTOCOLS FOR COMPUTER COMMUNICATION, 2000. **Proceedings...** New York:ACM Press, 2000. p. 283-294.

LEINWAND, A.; CONROY, K.F. **Network Management:** A pratical perspective. 2nd ed. Massachusetts:Addison-Wesley, 1996. 338p.

LEVI, D.; MEYER, P.; STEWART, B. **Simple Network Management Protocol (SNMP) Applications**: RFC 3413. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

MANNAERT , H.; ADRIAENSSENS, P. *Web* Services Based Systems for Network Management and Provisioning: A Case Study. In: AICT; SAPIR; ELETE, 2005. **Proceedings…** Los Alamitos: IEEE Computer Society, 2005. p. 442-445.

MARTIN-FLATIN, J. P.; ZNATY, S.; HUBAUX, J. P. **A** Survey of Distributed Enterprise Network and Systems Management Paradigms. **Journal of Network and Systems Management**, New York, v. 7, n. 1, p. 9-26, 1999.

MASSIE, M.L.; CHUN, B.N.; CULLER, D.E. The ganglia distributed monitoring system: design, implementation, and experience. **Parallel Computing**, Amsterdam, v. 30, n. 7, p. 817-840, 2004.

MATTHEWS, W.; COTTRELL, L. The PingER Project: Active Internet Performance Monitoring for the HENP Community. **IEEE Communications Magazine**, New York, v.38, n.5, p.130-136, May 2000.

MAURO, D.; SCHMIDT K. **Essential SNMP.** Beijing: O'Reilly, 2001. 291p.

McCLOGHRIE, K. et al. **Structure of Management Information Version 2 (SMIv2)**: RFC 2578. [S.l.]: Internet Engineering Task Force, Network Working Group, 1999.

McCLOGHRIE, K. et al. **Textual Conventions for SMIv2**: RFC 2579. [S.l.]: Internet Engineering Task Force, Network Working Group, 1999.

McCLOGHRIE, K. et al. **Conformance Statements for SMIv2**: RFC 2580. [S.l.]: Internet Engineering Task Force, Network Working Group, 1999.

McCLOGHRIE, K.; ROSE, M. **Management Information Base for Network Management of TCP/IP-based internets:MIB-II**: RFC 1156. [S.l.]: Internet Engineering Task Force, Network Working Group, 1990.

McCLOGHRIE, K.; ROSE, M. **Management Information Base for Network Management of TCP/IP-based internets**: RFC 1156. [S.l.]: Internet Engineering Task Force, Network Working Group, 1991.

McGREGOR, T. BRAUN, H-W. BROWN, J. The NLANR network analysis infrastructure. **IEEE Communications Magazine**, New York, v. 38, n. 5, p. 122-128, May 2000.

MILLER, M. C. **Managing Internetworks with SNMP:** The definitive guide to simple network management protocol, SNMPv2, RMON and RMON2. 3rd ed. New York:Wiley, 1999.

MURRAY, M. CLAFFY, K. Measuring the Immeasurable**:** Global Internet Measurement Infrastructure. In: PASSIVE AND ACTIVE MEASUREMENTS, 2001, Amsterdam, Netherlands. **Proceedings…**[S.l.:s.n.], 2001. p.159-167.

NEWMAN, H.B. et al. MonALISA**:** A Distributed Monitoring Service Architecture. In: CHEP, 2003, La Jola, California. **Proceedings...** Disponível em: <http://monalisa.cacr.caltech.edu/> Acesso em: fev. 2005.

NLANR/DAST. **Iperf**. Disponível em: <http://dast.nlanr.net/Projects/Iperf/> Acesso em: 19 nov. 2004.

OASIS. **UDDI Version 3.0.2**. Disponível em: <http://uddi.org/pubs/uddi_v3.htm>. Acesso em: dez. 2005.

PAPAZOGLOU, M. P.; HEUVEL, W. *Web* Services Management: A Survey. **IEEE Internet Computing**, Piscataway, NJ, USA, v. 9, n. 6, p. 58-64, 2005.

PAXSON, V.; ALMES, G.; MAHDAVI, J. Mathis, M. **Framework for IP Performance Metrics:** RFC 2330. [S.l.]: Internet Engineering Task Force, Network Working Group, 1998.

PAXSON, V.; ADAMS, A.; MATHIS, M. Experiences with NIMI. In: SYMPOSIUM APPLICATIONS AND THE INTERNET, SAINT, 2002. Nara, Japan. **Proceeding…** Los Alamitos: IEEE Computer Society, 2002. p.108-118.

PAXSON, V.; MAHDAVI, J.; ADAMS, A.; MATHIS, M. An architecture for large scale Internet measurement. **IEEE Communications Magazine,** New York, v.38, n.8, p.48-54, Aug. 1998.

PRAS, A. **Network Management Architectures**. 1995. Ph. D-thesis – University Twente, Enschede, Netherlands.

PRAS, A. et al. Comparing the Performance of SNMP and *Web* Services-Based Management. **IEEE Transactions on Network and Service Management**, [S.l.], v.1, n.2, 2004.

PRESUHN, R. et al. **Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)**: RFC 3418. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

PRESUHN, R. et al. **Transport Mappings for the Simple Network Management Protocol (SNMP)**: RFC 3417. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

PRESUHN, R. et al. **Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)**: RFC 3416. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

ROSE, M. **A Convention for Defining Traps for use with the SNMP**: RFC 1215. [S.l.]: Internet Engineering Task Force, Network Working Group, 1991.

ROSE, M.; McCLOGHRIE, K. **Concise MIB Definitions**: RFC 1212. [S.l.]: Internet Engineering Task Force, Network Working Group, 1991.

ROSE, M.; McCLOGHRIE, K. **Structure and Identification of Management Information for TCP/IP-based internets**: RFC 1155. [S.l.]: Internet Engineering Task Force, Network Working Group, 1990.

SAYDAM, T.; MAGEDANZ, T. From Networks and Network Management into Service and Service Management**. Journal of Networks and System Management**, New York, v.4, n.4, p. 345-348, Dec. 1996.

SCHÖNWÄLDER, J.; QUITTEK, J.; KAPPLER, C. Building Distributed Management Applications with the IETF Script MIB. **IEEE Journal on Selected Areas in Communications**, New York, v.18, n.5, May 2000.

SHALUNOV, S. et al. **A one-way active measurement protocol (OWAMP):** RFC 4656. [S.1]: Internet Engineering Task Force, IP Performance Metrics Work Group, 2006.

SOARES, L. F. G.; LEMOS, G.; COLCHER, S. **Das LANs, MANs e WANs às Redes ATM.** 2.ed. Rio de Janeiro: Campus, 1995. 705p.

SPRING, N.; WETHERALL, D.; ANDERSON, T. Scriptroute: A Public Internet Measurement Facility. In: USENIX SYMPOSIUM ON INTERNET TECHNOLOGIES AND SYSTEMS, 4., 2003, Seattle, Washington, USA. **Proceedings…** Disponível em: <http://www.usenix.org/events/usits03/tech/spring.html>. Acesso em: abr. 2005.

STALLINGS, W. SNMP and SNMPv2: The Infrastructure for Network Management. **IEEE Communications Magazine**, New York, v.36, n.3, p.37-43, May 1998.

STALLINGS, W. **SNMP, SNMPv2 and CMIP:** the practical guide to network management standards. Reading: Addison-Wesley, 1993.

STALLINGS, W. **SNMP, SNMPV2, SNMPV3 and RMON 1 and 2**. 3rd ed. Reading: Addson-Wesley, 1999. 619p.

STEVENS, W.R. **TCP/IP Illustrated.** Reading: Addison-Wesley, 1994. v.1.

STRAU, F.; KLIE, T. Towards XML oriented internet management. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 8., 2003, Colorado Springs. **Proceedings …** [S.l.: s.n], 2003.

TANENBAUM, A. S. **Redes de Computadores**. 4.ed. Rio de Janeiro: Campus, 2003. 945p.

TOARTA, M. et al. **MonALISA:** Monitoring Agents in A Large Integrated Services Architecture. Disponível em: <http://monalisa.cacr.caltech.edu/> Acesso em: fev. 2005.

VORUGANTI, R.R. A global network management framework for the '90s. **IEEE Communications Magazine**, New York, v.32, n.8, p.74-83, Aug. 1994.

W3C, World Wide *Web* Consortium. ***Web* Services Architecture.** Disponível em: <http://www.w3.org/TR/ws-arch/> Acesso em: dez. 2005.

WIJNEN, B.; PRESUHN, R.; McCLOGHRIE, K. **View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)**: RFC 3415. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

WOLSKI, R. Dynamically forecasting network performance using the Network Weather Service. **Cluster Computing**, Hingham, v. 1, n. 1, p. 119-132, May 1998.

WOLSKI, R.; SPRING, N.; PETERSON, C. Implementing a performance forecasting system for metacomputing: The Network Weather Service. In: ACM/IEEE CONFERENCE ON HIGH PERFORMANCE NETWORKING AND COMPUTING, 1997, San Jose, CA. **Proceedings…** New York: ACM, 1997. p. 1-19.

WOLSKI, R.; SPRING, N. T.; HAYES, J. The network weather service: a distributed resource performance forecasting service for metacomputing. **Future Generation Computer Systems**, Amsterdam, v. 15, n. 5-6, p. 757-768, Oct. 1999.

ANEXO A GERENCIAMENTO OSI - FCAPS

Gerenciamento de Falhas

O gerenciamento de falhas compreende as tarefas de detecção e localização de problemas, isolamento e, se possível, sua solução. Existem duas formas principais de detecção de falha, na primeira o sistema de gerenciamento efetua uma consulta a um objeto gerenciado verificando o seu funcionamento anormal, a outra, é pela recepção de notificações enviadas pelos equipamentos para o sistema de gerenciamento. A ação do sistema de gerenciamento quando detecta uma falha pode ir desde a simples notificação para o administrador até a tomada de ações automáticas, visando a correção do problema.

As principais funções previstas para o gerenciamento de falhas são: receber notificações de falhas; armazenar e analisar registros de falhas; examinar os registros e agir; identificar e corrigir falhas; realizar testes de diagnóstico; e, prevenir falhas.

Gerenciamento de Configuração

Esta área é representada pelo conjunto de operações necessárias para a inicialização, alteração, término e armazenamento das configurações dos equipamentos gerenciados. As modificações dos atributos de configuração do sistema podem ser de maneira preventiva, por necessidade, por eventos imprevisíveis ou simplesmente para atender a demanda de um usuário específico. O gerenciamento de configuração permite a coleta de informações sobre os equipamentos para obtenção da documentação atualizada da configuração.

As principais funções do gerenciamento de configuração são: coletar informações de configuração; armazenar a configuração; atribuir os valores iniciais de configuração aos elementos gerenciados; alterações de configuração dos elementos gerenciados, armazenando a alteração.

Gerenciamento de Contabilização

A principal função do gerenciamento de contabilização é definir e monitorar os recursos que são disponibilizados para os usuários. Esta área funcional deve garantir que os recursos definidos para determinado usuário estejam disponíveis em quantidade suficiente. Os recursos normalmente monitorados são: uso de CPU, espaço em disco, quantidade de tráfego na rede e tempo de uso.

As tarefas principais são: monitorar e armazenar o uso dos recursos pelos usuários; garantir os limites definidos para determinado usuário; estabelecer cotas de utilização; e, estabelecer escalas de tarifação.

Gerenciamento de Desempenho

A principal tarefa desta área de gerenciamento é monitorar o desempenho da rede por meio de métricas de desempenho. Os resultados obtidos são fundamentais na otimização e no planejamento da rede. Alguns exemplos dessas métricas são: taxa de utilização, taxa de erros, vazão e tempo de resposta. Essa área deve assegurar que a rede tenha capacidade para suportar as suas aplicações e seus usuários.

As principais funções desta área funcional são: monitoração dos parâmetros de QoS (*Quality of Service*) como atraso, *jitter* e vazão; monitoração de disponibilidade; monitoração do tempo de resposta de equipamentos e aplicações; e, registro e notificação de anormalidades no desempenho da rede.

Gerenciamento de Segurança

Esta é a área responsável pelo controle de acesso aos recursos da rede, utilizando técnicas de autenticação, políticas de autorização, controle de chaves, senhas e manipulação dos logs de segurança. É o conjunto de funções que permite ao gerente identificar e proteger equipamentos e dados da rede, de ataques e violações de pessoas não autorizadas (*hackers* e *crackers*). Para manter a segurança deve-se limitar o acesso a equipamentos e dados com ferramentas adequadas e *softwares* de segurança como *proxy* e *firewall*. O gerenciamento de segurança é responsável pela aplicação da política de segurança da instituição.

A área de gerenciamento de segurança engloba tarefas como: controle do acesso aos recursos da rede; manutenção da integridade dos dados; monitoração dos usuários; manutenção de registros (logs) de segurança; e, monitoração do uso dos recursos.

ANEXO B SNMP

SNMP

O funcionamento do protocolo SNMP é baseado no conceito de agentes e gerentes, com uma interação igual ao modelo cliente-servidor. Neste modelo, os agentes coletam junto aos objetos gerenciados informações relevantes para o gerenciamento da rede e os gerentes buscam e processam as informações recolhidas pelos agentes. Com as informações, o gerente procura detectar a presença de falhas no funcionamento dos componentes da rede (*hosts*, *gateways*, processos, protocolos de comunicação, etc.), para que possam ser tomadas providências no sentido de contornar os problemas ocorridos como conseqüência das falhas.

A figura b. mostra a interação entre agentes e gerentes no gerenciamento Internet.

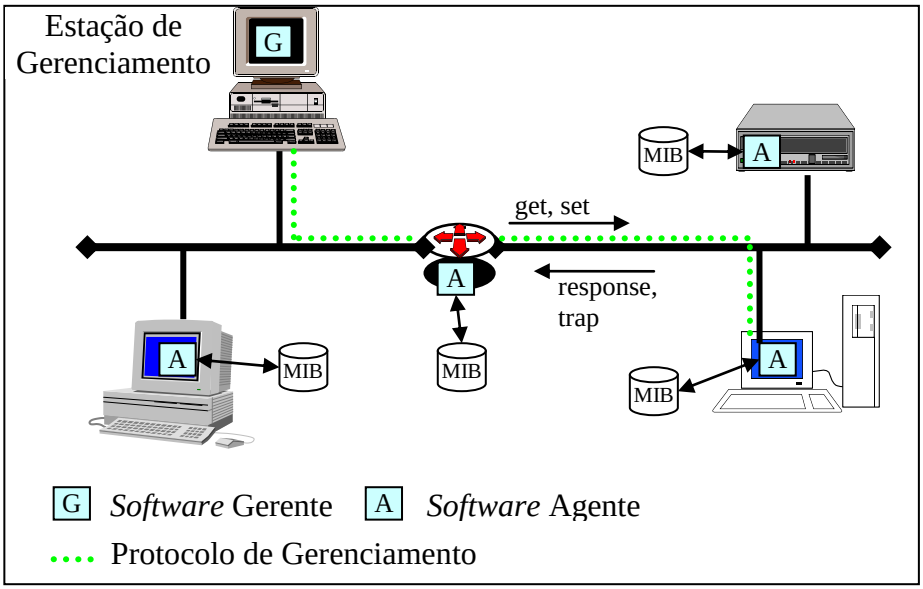

Figura B.1: Modelo de gerenciamento SNMP

A arquitetura de gerenciamento SNMP (RFC (*Request For Comments*) 1157) (CASE, 1990) pressupõe quatro elementos fundamentais que são: a Estação de Gerenciamento; o Agente de Gerenciamento; a Base de Informações Gerenciais (MIB); e um Protocolo de Gerenciamento de Rede.

Estação de Gerenciamento

A estação de gerenciamento é o dispositivo responsável pela interface entre o operador de rede e o sistema a ser gerenciado. Este dispositivo deve possuir no mínimo:

- Um conjunto de aplicações de gerenciamento para análise de dados, recuperação de falhas, entre outras;
- Uma interface através da qual o gerente possa monitorar e controlar a rede;
- Um protocolo através do qual a estação de gerenciamento e as entidades gerenciadas possam trocar informações de gerenciamento e controle;
- Uma base de dados com informações extraídas a partir das bases de informação de gerenciamento de todas as entidades gerenciadas.

Apenas os dois últimos requisitos devem obrigatoriamente seguir a especificação do SNMP. A forma de implementação dos demais componentes é livre, ficando a cargo do administrador.

Agente de Gerenciamento

A maioria dos equipamentos como *hosts*, pontes, roteadores e *hubs* podem ser equipados com agentes SNMP, e com isso podem ser gerenciados de uma estação de gerenciamento. O agente de gerenciamento responde a pedidos de informações e ações requisitadas pela estação de gerenciamento. Além disso, pode enviar de maneira assíncrona à estação de gerenciamento, informações importantes, apesar de não solicitadas.

Base de Informações Gerenciais

Os recursos gerenciados em uma rede são representados como objetos. A coleção de objetos é referenciada como uma MIB (*Management Information Base*). Cada objeto presente na MIB é essencialmente uma variável que representa um aspecto do sistema gerenciado. A estação de gerenciamento monitora um equipamento recuperando valores de objetos da MIB e, pode realizar uma ação no agente, modificando e/ou alterando os valores de variáveis específicas.

Protocolo de Gerenciamento de Rede

A estação de gerenciamento e os agentes trocam informações através do protocolo de gerenciamento SNMP. Este protocolo possui três mensagens básicas que permitem a interação entre agentes e gerentes. São elas:

- Get: possibilita que a estação de gerenciamento recupere o valor de objetos do agente;
- Set: possibilita que a estação de gerenciamento altere o valor de objetos do agente;
- Trap: possibilita que o agente notifique a estação de gerenciamento sobre a ocorrência de eventos importantes.

A RFC 1155 (ROSE, 1990), Estrutura das Informações de Gerenciamento - SMI (*Structure of Management Information*), define um conjunto de regras usadas para definir e identificar os objetos gerenciados. A SMI especifica regras e restrições para manter os protocolos de gerenciamento o mais simples e independente de plataforma quanto possível. A SMI especifica os objetos gerenciados usando a linguagem ASN.1 (*Abstract Syntax Notation 1*) da ISO (ISO 1987).

A ASN.1 é uma linguagem formal que possui algumas características especiais: uma delas é a utilização de uma notação de fácil leitura e entendimento pelos usuários; a outra é o uso de uma representação codificada bastante compacta. A grande capacidade da ASN.1 é prover uma notação formal e precisa que elimine qualquer ambigüidade tanto na representação quanto no significado.

Além de manter documentos livres de ambigüidade, a ASN.1 também permite a implementação de protocolos e aplicações com estruturas que os deixam completamente interoperáveis, não dependente de máquina.

A RFC 1156 (McCLOGHRIE, 1990) definiu a primeira MIB padrão Internet, a MIB-I. Com a necessidade de novos objetos para o gerenciamento, o IAB (*Internet*

Architecture Board) propôs uma extensão da MIB-I, a MIB-II, atualmente definida pela RFC 1213, mantendo a compatibilidade com a anterior.

SNMPv2C

O protocolo SNMP tornou-se rapidamente o esquema de gerenciamento mais utilizado. Porém, devido ao uso em larga escala, as suas deficiências também ficaram aparentes, dentre elas (STALLINGS, 1998): a falta de comunicação entre gerentes; a inabilidade para transferência de grandes quantidades de dados e a falta de segurança.

Visando corrigir problemas de segurança e desempenho da primeira versão, surgiu então a versão 2 do SNMP o SNMPv2. Juntamente com a nova versão do SNMP o IETF lançou uma nova SMI (RFC 1442), novas mensagens e novos objetos na MIB, além da manutenção da compatibilidade com a versão anterior.

Duas novas mensagens foram adicionadas às existentes na versão anterior:

- InformRequest: utilizada para comunicação entre gerentes distribuídos, permitindo o gerenciamento distribuído.
- GetBulkRequest: funcionamento idêntico ao Get-Next-Request, porém otimiza o acesso recuperando mais de um objeto ao mesmo tempo.

Visando a segurança, os pacotes passaram a ser criptografados com o DES (*Data Encryption Standard*) e autenticados pelo protocolo MD5 (*Message Digest 5*). Os contadores foram incrementados em 32 bits, ou seja, passaram de 32 para 64bits, permitindo um maior tempo de utilização do equipamento antes que o contador chegue ao máximo e retorne a zero.

A primeira proposta do SNMPv2 surgiu em 1993, porém não teve a aceitação esperada pelos seus criadores. As questões de segurança foram consideradas demasiadamente complexas pelos desenvolvedores. Uma revisão em 1996 manteve os aprimoramentos nos aspectos funcionais, mas foram retiradas as questões de segurança.

A RFC 1901 definiu um *framework* administrativo que associa cada mensagem do protocolo SNMP a um valor de comunidade. O valor de comunidade é uma string e funciona como uma espécie de senha para interação entre agentes e gerentes. O uso deste *framework* administrativo com o SNMPv2 passou a ser conhecido como "*Community-based SNMPv2*" ou SNMPv2C.

As RFC atuais do protocolo SNMPv2C são 1901 (histórica) (CASE, 1996), RFCs 2578 a 2580 (McCLOGHRIE, 1999) e RFCs 3416 a 3418 (PRESUHN, 2002).

SNMPv3

A terceira versão do protocolo SNMP define os mecanismos de segurança, retirados das especificações na versão anterior.

O SNMPv3 definiu uma nova arquitetura para suportar as funções de gerenciamento de maneira modular. Dessa forma, foi possível integrá-lo com as versões anteriores (SNMPv1 e SNMPv2c) e prover um *framework* para futuras revisões e ampliações do protocolo. Uma entidade SNMPv3 pode ser definida como um gerente ou agente e cada entidade pode incluir uma ou mais aplicações que utilizam um *Motor* SNMP. O *Motor* SNMP possui quatro componentes: *Despachante*; *Subsistema de Processamento de*

Mensagens; *Subsistema de Segurança*; e *Subsistema de Controle de Acesso*. A figura b.2 apresenta os componentes do *motor* SNMPv3.

O *Despachante* é o responsável pelo envio e recebimento de mensagens. Quando uma mensagem é recebida o Despachante tenta determinar o número de versão da mensagem e então passa para o sistema de processamento adequado.

O *Subsistema de Processamento de Mensagens* é o responsável por preparar as mensagens a serem enviadas e extrair os dados de mensagens recebidas. Ele é constituído de um ou mais modelos de processamento de mensagens, por exemplo, SNMPv1, SNMPv2, SNMPv3 e outros. Novos modelos podem ser acrescentados no futuro.

O *Subsistema de Segurança* fornece os serviços de autenticação, criptografia de mensagens. Concebido de forma modular, possui um subsistema de segurança baseado em comunidade para dar suporte às versões SNMPv1 e SNMPv2c, um modelo de segurança baseado em usuário, com criptografia e autenticação, para as mensagens SNMPv3, além de prover um modelo aberto de segurança para possíveis atualizações ou novas versões.

O *Subsistema de Controle de Acesso* basicamente determina se o acesso a um determinado objeto gerenciado deve ou não ser permitido. O modelo de controle de acesso definido atualmente é o VACM (*View-Based Access Control Model*), sendo possível controlar quais os usuários e quais operações podem ter acesso a quais objetos gerenciados.

Cabe ressaltar que o SNMPv3 não substitui o SNMPv1 e SNMPv2C, mas define mecanismos que permitem que possam ser utilizados em conjunto. As RFC referentes ao protocolo SNMPv3 são a RFC 3411(HARRINGTON, 2002), a RFC 3415(WIJNEN, 2002), além das RFCs 2578 a 2580 (McCLOGHRIE, 1999) do SNMPv2C e as 3416 a 3418 (PRESUHN, 2002) da SMIv2.

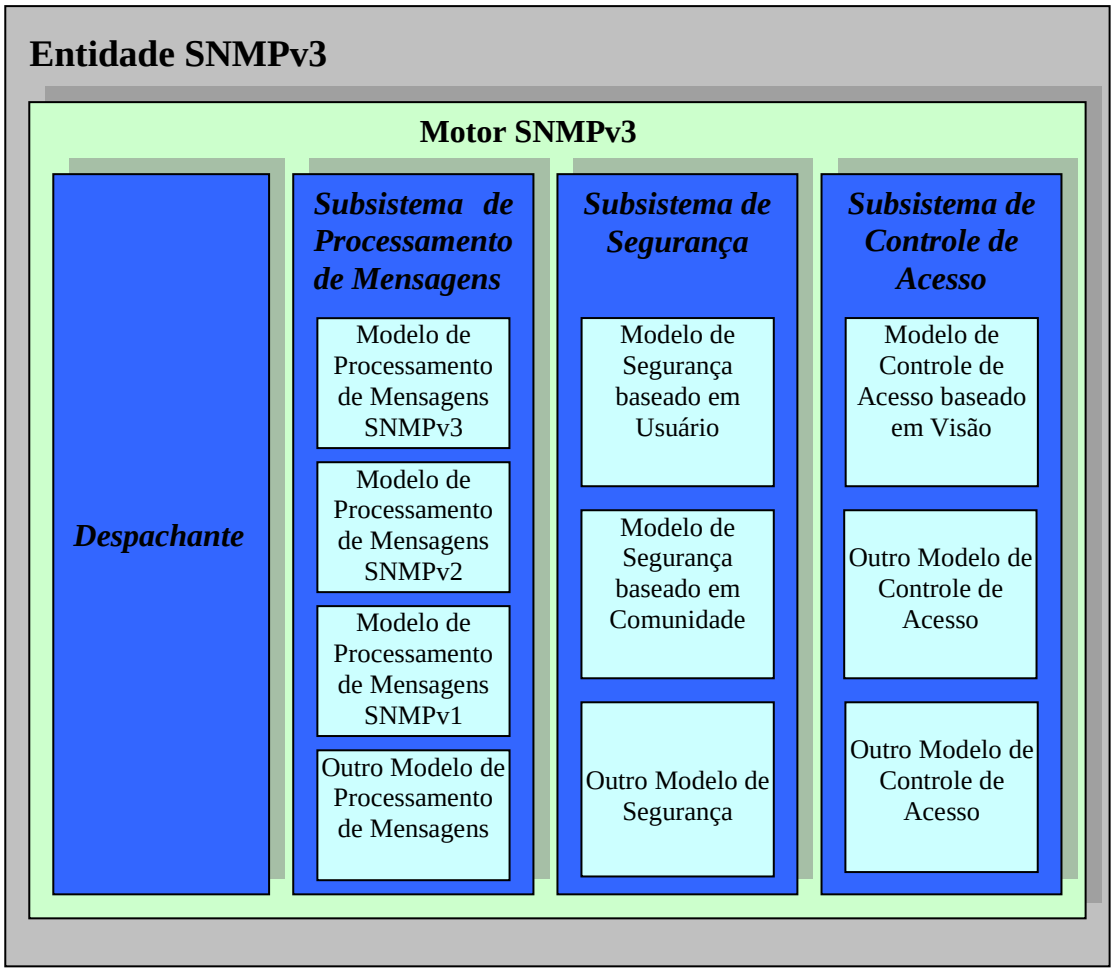

Figura B.2: Componentes do SNMPv3

ANEXO C ARQUITETURA DOS SERVIÇOS *WEB*

1. Arquitetura dos Serviços *Web*

Segundo o W3C (W3C, 2005) um serviço *web* é um sistema de *software* projetado para suportar interações entre máquinas sobre a rede. Ele possui uma interface descrita em um formato processável por máquina (especificamente WSDL). Um sistema interage com um serviço *Web* de uma maneira predeterminada pela sua descrição, usando mensagens SOAP tipicamente codificadas em XML e transportadas usando HTTP.

Em Fevereiro de 2004, o W3C publicou um documento que define a *Web Services Architecture (WSA)*. A finalidade da WSA é fornecer meios padronizados que permitam a interação entre *softwares* de diferentes aplicações, independentemente das várias plataformas e/ou estruturas encontradas no ambiente. Os serviços *web* provêem um meio padronizado de interoperabilidade entre diferentes aplicações de *software*, rodando sobre uma variedade de plataformas e/ou *frameworks*. A arquitetura não tenta especificar como devem ser implementados serviços *web* e não impõe nenhuma restrição de como poderiam ser combinados. Ela descreve um conjunto mínimo de características que são comuns a todos os serviços *web* e características que são necessárias para muitos deles. Essa arquitetura define a interoperabilidade identificando os serviços que são globais aos serviços *web*, a fim de garantir total comunicação entre eles. Os principais conceitos e relacionamentos entre os componentes principais da arquitetura estão descritos abaixo (W3C, 2005).

1.1. Agentes e Serviços

Um serviço *web* é uma noção abstrata que deve ser implementada por um agente concreto. O ***agente*** é uma parte de *software* ou *hardware* que envia e recebe mensagens, enquanto o ***serviço*** é o recurso caracterizado pelo conjunto abstrato de funcionalidades que é provido. Para ilustrar a distinção, seria possível implementar um serviço *web* usando diferentes linguagens de programação. Assim, embora o agente possa ter mudado, os serviços providos permanecem os mesmos.

1.2. Requisitantes e Provedores

O propósito de um serviço *web* é prover alguma funcionalidade para seu dono, uma pessoa ou organização, como um negócio ou um indivíduo. A ***Entidade Provedora*** [4]é a pessoa ou organização que provê um agente apropriado para implementar um serviço particular. A ***Entidade Requisitante*** é a pessoa ou organização que deseja fazer uso de uma entidade provedora de um serviço *web*. Ela usará um agente requisitante para trocar mensagens com uma entidade provedora e seu respectivo agente provedor.

Na maioria dos casos, o agente requisitante é quem inicia a troca de mensagens, mas isto não é obrigatório. Todavia, para consistência é utilizado o termo ***Agente Requisitante*** para o agente que interage com o ***Agente Provedor***, até mesmo nos casos em que é o agente provedor que inicia a troca. Para que a troca de mensagens tenha

[4] *Muitos documentos usam o termo Provedor de Serviço para referir-se à entidade provedor e/ou o agente provedor. Similarmente, podem usar o termo Requisitante de Serviço para se referir à entidade requisitante e/ou agente requisitante. Porém, como estes termos são ambíguos – às vezes se referindo ao agente e às vezes para a pessoa ou organização que possuem os agentes – o documento do W3C utiliza os termos* ***Entidade Requisitante***, ***Entidade Provedora***, ***Agente Requisitante*** *e* ***Agente Provedor***.

êxito, a entidade requisitante e a entidade provedora devem concordar sobre a semântica e os mecanismos da troca de mensagens.

1.3. Descrição do Serviço

Os mecanismos de troca de mensagens estão documentados em uma *Web Service Description (WSD).* A WSD é uma especificação da interface de um serviço *web*, escrita em *Web Services Description Language (WSDL)*. Ela define o formato das mensagens, os tipos dos dados, o protocolo de transporte e o formato da serialização de transporte que deve ser usado entre o agente requisitante e o agente provedor. Também especifica um ou mais locais de rede que um agente provedor pode ser invocado. Pode prover algumas informações sobre o padrão de troca de mensagens que é esperado. Em essência, a descrição de serviço representa um acordo que governa os mecanismos de interação entre os serviços.

1.4. Semântica

A semântica de um serviço *web* é o compartilhamento da definição do comportamento de um serviço em particular, em resposta a mensagens que são enviadas para este. Este é um contrato entre a entidade requisitante e a entidade provedora relativo ao propósito e conseqüências da integração. Embora esse contrato represente o acordo global entre a entidade requisitante e a entidade provedora sobre como e porque seus respectivos agentes irão interagir, isto não é necessariamente escrito ou explicitamente negociado. Isto pode estar explícito ou implícito, oral ou escrito, processável por máquina ou orientado a humanos e pode ser um acordo legal ou informal. Enquanto a descrição do serviço representa um contrato que governa os mecanismos de interação com um serviço particular, a semântica representa um contrato que governa o significado e o propósito daquela interação. A linha divisória entre elas não é rígida.

Quanto mais semanticamente rica for a linguagem usada para descrever os mecanismos de interação, mais informações essenciais podem migrar da semântica informal para a descrição do serviço. À medida que esta migração ocorre, mais automatizado pode ser o trabalho requerido para alcançar o sucesso da interação.

1.5. Funcionamento dos componentes

Há muitos modos nos quais uma entidade requisitante pode contratar e usar um serviço *web*. Em geral as etapas necessárias, vistas na figura c, são:

1) as Entidades Requisitante e Provedora tornam-se conhecidas uma da outra (ou pelo menos um saiba do outro);
2) as Entidades Requisitante e Provedora, de alguma forma, concordam sobre a descrição e a semântica que governarão a integração entre o Agente Requisitante e o Provedor;
3) a descrição do serviço e semântica são compreendidos pelo Agente Requisitante e o Agente Provedor;
4) os Agentes Requisitante e Provedor trocam mensagens e executam alguma tarefa em nome das Entidades Requisitante e Provedora. Isto é, a troca de mensagens com o Agente Provedor representa a manifestação concreta de interagir com a Entidade Provedora do serviço *web*.

Alguns destes passos podem ser executados manualmente ou automatizados.

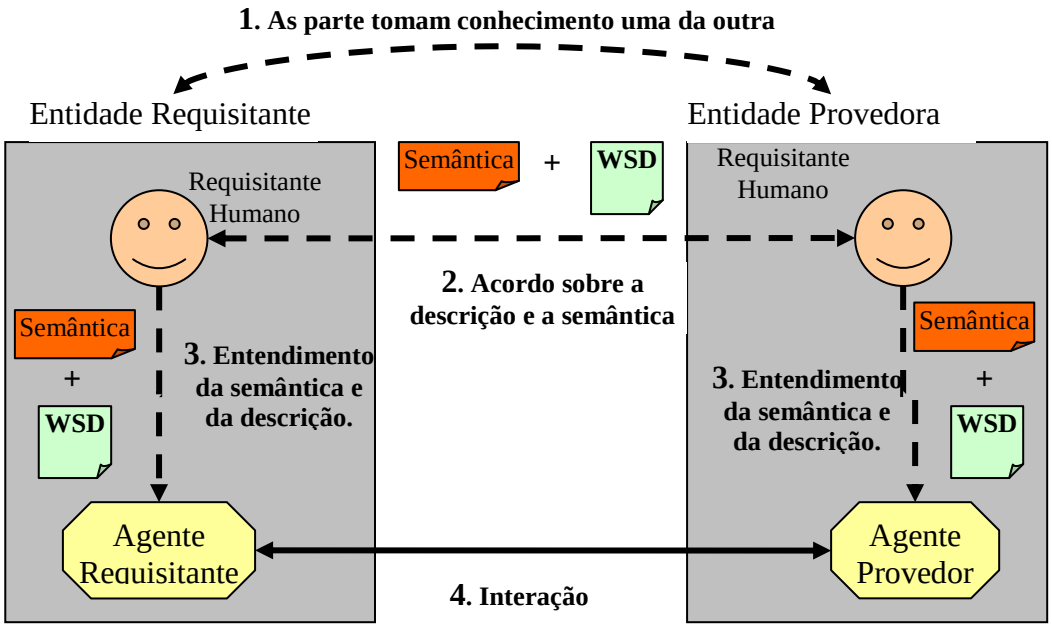

Figura C.1: Relacionamento entre componentes de um serviço *web*

2. Tecnologias que compõem os Serviços *Web*

Existem muitas tecnologias presentes na arquitetura de serviços *web* e há muitas maneiras de visualizá-las. A figura c.2 apresenta uma ilustração da hierarquia das principais tecnologias que fundamentam os serviços *web*. Os blocos à esquerda são conceituais e os blocos da direita são tecnologias reais que estão presentes em cada camada. Com exceção das tecnologias de comunicação que são amplamente conhecidas, as demais são detalhadas nas próximas subseções.

Serviço de Publicação e Descoberta	UDDI
Serviço de Descrição	WSDL
Mensagens	SOAP
Comunicação	HTTP, SMTP, FTP, etc

Figura C.2: Componentes presentes na Arquitetura de Serviços *Web*

2.1. SOAP

Em 1998, um grupo de pesquisadores de empresas como Microsoft, DevelopMentor e UserLand *Software* estavam envolvidos em criar o SOAP, um protocolo que inicialmente simplesmente definia um mecanismo para a transmissão de documentos XML que contivessem comandos capazes de disparar operações ou respostas em sistemas remotos ou RPCs. Para conseguir este objetivo eram necessários uma linguagem de esquema e um sistema de tipos para XML que até então não existia. O SOAP deveria ter sido lançado em 1998 como uma especificação, mas por divergências políticas isso não aconteceu (HENDRICKS, 2002).

No final de 1999 foi definida a especificação SOAP 1.0. A versão 1.1 foi submetida ao *World Wide Web Consortium (W3C)* para discutir a viabilidade de um padrão formal.

Um grupo de trabalho foi definido e em 2003 foi divulgada a recomendação da versão 1.2 do SOAP, a atual versão.

A recomendação SOAP 1.2 é dividida em duas partes principais: A primeira parte da especificação define um framework de mensagens. As mensagens SOAP podem ser carregadas por variados protocolos de rede, como HTTP, SMTP, FTP, RMI/IIOP ou um protocolo de mensagem proprietário. A segunda define três componentes opcionais: um conjunto de regras de codificação para expressar instâncias dos tipos de dados definidos pela aplicação, uma convenção para representar RPCs e respostas e um conjunto de regras para usar SOAP com HTTP/1.1.

Na versão atual, o SOAP não é mais definido com um único significado, existem duas expansões para este termo, que refletem diferentes caminhos nos quais a tecnologia pode ser interpretada, que são (W3C, 2005):

- ***Service Oriented Architecture Protocol:*** no caso geral, uma mensagem SOAP representa a informação necessária para invocar um serviço ou analisar o resultado de uma chamada e contém informações específicas da definição da interface do serviço.

- ***Service Object Access Protocol:*** quando é usada uma representação opcional do SOAP RPC, a mensagem SOAP representa um método de invocação de um objeto remoto e a serialização da lista de argumentos do método que precisam ser movidos do ambiente local para o ambiente remoto.

De maneira simplificada, o SOAP é um protocolo de computação superficialmente distribuído que permite a troca de informações em um ambiente descentralizado e distribuído. A especificação SOAP define um *framework* para a transmissão de mensagens entre sistemas distribuídos, além de convenções para a transmissão de chamadas e respostas de procedimentos remotos. Entretanto, o SOAP não define a semântica das mensagens, fornece, apenas, o *framework* (HENDRICKS, 2002).

O SOAP também é uma especificação para requisitar métodos de negócio, como documentos XML, e suporta outros protocolos como HTTP e SMTP. Uma mensagem SOAP é um fragmento XML bem formado, encapsulado por um par de elementos SOAP.

2.2. Mensagem SOAP

De acordo com o W3C (W3C, 2005), uma mensagem SOAP é um documento XML que pode conter, no máximo, três partes:

- envelope SOAP (obrigatório)
- cabeçalho SOAP (opcional)
- corpo SOAP (obrigatório)

A figura c.3 representa uma mensagem SOAP, as partes são detalhadas na seqüência.

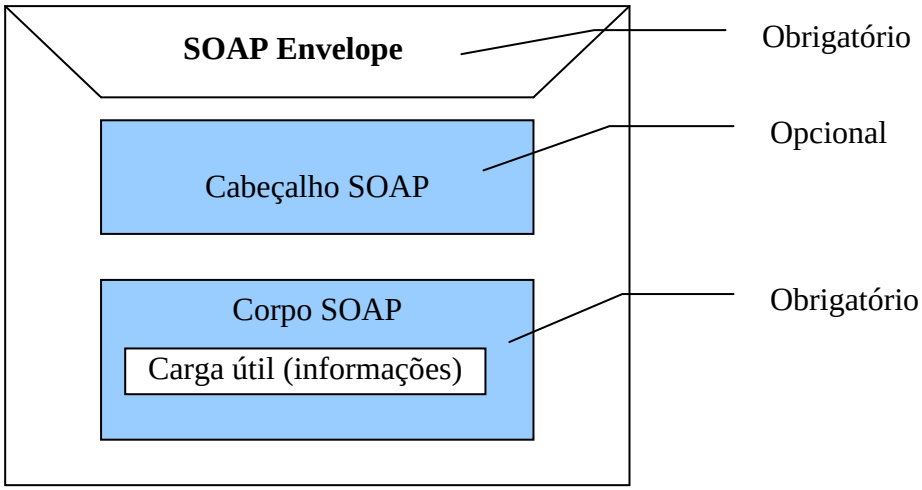

Figura C.3: Representação gráfica de uma mensagem SOAP

O **Envelope SOAP** é o elemento mais externo de uma mensagem SOAP, podendo ser interpretado analogamente a um envelope usado para enviar uma carta pelo correio convencional. O envelope SOAP é designado em uma mensagem SOAP pelo elemento *SOAP-ENV:Envelope.*

Um **Cabeçalho SOAP** não é obrigatório, porém se for usado deve vir após o envelope SOAP. Um cabeçalho SOAP pode conter elementos filhos, que devem ser representados como entradas de cabeçalho dentro da especificação SOAP. O uso de cabeçalhos SOAP permite uma maneira flexível de adicionar recursos como autenticação e gerenciamento de transações a uma mensagem SOAP. O cabeçalho SOAP é determinado pelo elemento *SOAP-ENV:Header.*

O **corpo SOAP** de uma mensagem é especificado usando o elemento *SOAP-ENV:Body* . O corpo SOAP contém as informações (carga útil) que devem ser recebidas pelo receptor. Em teoria, um corpo SOAP pode conter qualquer tipo de informação, mas tipicamente contém uma chamada RPC ou resposta RPC ou relatório de erro.

Abaixo segue um código exemplo de uma mensagem SOAP:

```
<SOAP-ENV:Envelope>
// Elemento raiz do SOAP, definindo que é uma mensagem SOAP
    <SOAP-ENV:Header>      //      Opcional
    // Informações específicas, como autenticação – opcional
    </SOAP-ENV:Header>     //      Opcional
    <SOAP-ENV:Body>
    // Elemento que contém o corpo da mensagem (incluindo possíveis erros)
            <SOAP-ENV:Fault>
            // Elemento fault contém os erros que podem ocorrer
            </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2.3. WSDL

Um dos fortes argumentos para o uso de serviços *web* é o fato de permitir que as aplicações conversem umas com as outras sobre qualquer rede, independentemente da linguagem em que foram desenvolvidas. Para que isso seja viável, é necessária uma maneira padronizada para descrever os serviços que são disponibilizados por

determinada aplicação. Desta forma, uma aplicação externa pode se adequar dinamicamente a um serviço. Além disso, a descrição deve ser independente de linguagem e plataforma.

A ***Web Services Description Language (WSDL)*** é uma linguagem baseada em XML com a finalidade de descrever os serviços *web* através de XML, documentando o serviço para que os clientes possam utilizá-lo de forma automatizada. A WSDL foi definida num esforço conjunto da IBM, Microsoft e Ariba e submetida ao W3C, para padronização. A versão atual é a WSDL 1.1 e a versão 2.0 encontra-se em fase de padronização.

Segundo a W3C (W3C, 2005), as definições dos serviços *web* podem ser mapeadas para qualquer linguagem de implementação, plataforma, modelo de objeto ou sistema de mensagens. A aplicação poderia ser implementada usando COM, JMS, CORBA, COBOL ou qualquer outra solução de integração proprietária. A forma de implementação dos serviços *web* é irrelevante, desde que o remetente e receptor concordem na descrição de serviço (arquivo WSDL).

2.4. Especificação WSDL

Um documento WSDL representa a interface externa de um serviço *Web* e também contém a localização ou extremidade do serviço. Para dar maior flexibilidade a WSDL descreve um serviço *web* em duas partes, uma delas especifica a ligação com o nível de transporte da rede e a segunda uma noção abstrata e independente do protocolo de transporte. A figura c.4: apresenta graficamente os componentes da especificação WSDL.

<table>
<tr><td>Ligação</td><td rowspan="4">Definição Abstrata</td></tr>
<tr><td>Tipo de Porta</td></tr>
<tr><td>Mensagem</td></tr>
<tr><td>Tipo</td></tr>
<tr><td>Serviço</td><td rowspan="2">Implementação Específica</td></tr>
<tr><td>Porta</td></tr>
</table>

Figura C.4: Elementos de uma descrição WSDL

A figura c.5, apresenta a estrutura básica de um documento WSDL. Os seus componentes são detalhados na seqüência.

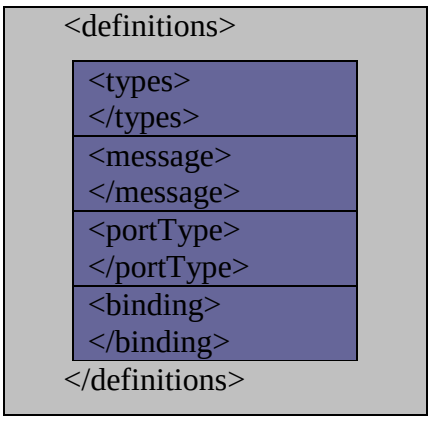

Figura C.5: Estrutura de um documento WSDL

O elemento **<definitions>** é o elemento raiz de um documento WSDL, onde tipicamente são definidos os *namespaces* usados no documento. Um exemplo típico é:

```
<?xml version ='1.0' encoding ='ISO-8859-1' ?>
<definitions name='Exemplo'
      targetNamespace='http://example.org/Exemplo'
      xmlns:tns='http://example.org/Exemplo'
      xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
      xmlns:xsd='http://www.w3.org/2001/XMLSchema'
      xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
      xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
      xmlns='http://schemas.xmlsoap.org/wsdl/'>
</definitions>
```

O elemento **<types>** contém os tipos de dados que estarão presentes na mensagem. Um serviço deve informar a seus clientes qual o tipo de dado esperado e qual o tipo de dado que eles podem esperar como retorno. Os tipos podem ser simples, como uma *string* ou *float* ou, dependendo da necessidade, tipos complexos também podem ser definidos no elemento *types.* Abaixo é apresentado um exemplo da definição de um tipo complexo, neste caso, um tipo endereço é composto por três campos (rua, cidade e código postal), todos definidos como *strings*.

```
<types>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="Endereco">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element name="rua" type="xsd:string"/>
                        <xsd:element name="cidade" type="xsd:string"/>
                        <xsd:element name="cep" type="xsd:string"/>
                  </xsd:sequence>
            </xsd:complexType>
      </xsd:element>
</xsd:schema>
</types>
```

No elemento **<message>** estão especificados os dados que serão transmitidos entre o cliente e o serviço. Uma mensagem pode ser composta de alguns argumentos, parâmetros, que são representados por uma única mensagem. Para isso, cada elemento **<message>** contém um ou mais elementos **<part>** que especificam o conteúdo da mensagem representando os parâmetros passados para o serviço e a resposta que ele retorna. Na seqüência um exemplo de duas mensagens, a primeira contém o nome de uma pessoa na forma de *string* e a segunda mensagem contém o nome de uma pessoa e o seu endereço. Vale ressaltar que o tipo Endereço é um tipo complexo e apresentado anteriormente.

```
<message name="nomeMsg">
      <part name="nome" type="xsd:string/>
</message>
<message name="enderecoMsg">
      <part name="nomePart" type="xsd:string"/>
      <part name="enderecoPart" type="tns:Endereço"/>
</message>
```

Com as mensagens definidas deve-se especificar a seqüência em que as mesmas devam ser utilizadas. A especificação WSDL chama isso de operação <**operation**>. Existem quatro tipos de mensagens diferentes de operações, que são:

- **Operação Unidirecional**: define que uma mensagem é enviada de um cliente para um serviço sem que ocorra nenhuma resposta por parte do serviço.
- **Solicitação-Resposta**: neste caso o cliente envia uma solicitação para um serviço e recebe uma mensagem como resposta para a solicitação. É o tipo mais comum de operação.
- **Pedido-Resposta**: define uma mensagem enviada do serviço para o cliente, resultando em uma mensagem de retorno do cliente para o serviço.
- **Notificação**: neste caso o serviço envia uma mensagem ao cliente, sem necessariamente receber uma resposta.

No código abaixo é apresentado um exemplo de uma operação do tipo solicitação-resposta, consistindo em uma mensagem de entrada e outra de saída. Na especificação WSDL pode-se agrupar uma ou mais operações usando o elemento <**portType**>. A opção de agrupar ou não as operações é uma decisão de projeto.

```
<portType name=”LivroEndereco”>
      <operation name=”buscaEndereco”>
            <input message=”nomeMsg”/>
            <output message=”endereçoMsg”/>
      </operation>
</portType>
```

O elemento mais complexo é o <**binding**>, o qual descreve o mecanismo utilizado por um serviço para se comunicar com um cliente. A WSDL é definida de forma independente do mecanismo efetivo para acessar um serviço. A especificação menciona SOAP, HTTP e MIME. O exemplo abaixo mostra o uso do protocolo SOAP como mecanismo de acesso ao serviço.

```
<binding name='ExemploBinding' type='tns:ExemploPortType'>
   <soap:binding style='rpc'
   transport='http://schemas.xmlsoap.org/soap/http'/>

   <operation name='buscaEndereco'>
         <soap:operation soapAction='exemplo#getNome'/>
         <input>
               <soap:body     use='encoded'     namespace='exemplo'
         encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
         </input>

         <output>
               <soap:body use='encoded' namespace='exemplo'
         encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
         </output>
   </operation>
</binding>
```

A última informação que um arquivo WSDL deve conter é o destino da solicitação real. O elemento <**port**> no elemento <**service**> contém essa informação. Diversas

portas podem ser especificadas, cada uma é específica para o tipo de vínculo que foi descrito no elemento <binding>. No exemplo abaixo a solicitação é enviada para um servidor local.

```
<service name='ExemploService'>
   <port name='ExemploPort' binding='ExemploBinding'>
         <soap:address location='http://127.0.0.1/servidor.php'/>
   </port>
</service>
```

2.5. UDDI

Segundo (OASIS, 2004) o *Universal Description, Discovery, and Integration (UDDI)* é um elemento central do grupo de padrões que compõe a pilha de componentes dos serviços *web*. A especificação UDDI define um método padronizado para publicação e descoberta de componentes de *software* em uma arquitetura baseada em serviços.

A primeira versão do UDDI (1.0) foi anunciado pela Microsoft, IBM e Ariba em setembro de 2000. Atualmente o UDDI representa tanto uma especificação proposta para padronização, quanto um consórcio de apoiadores ao projeto (UDDI.org). A versão atual é a 3.0, declarada oficialmente como padrão em fevereiro de 2005.

Quando o UDDI foi lançado, as atenções voltaram-se para o conceito de *Universal Business Registry (UBR)* que representa um diretório público de serviços disponíveis. A metáfora mais utilizada para descrever o UBR são as listas telefônicas, devido à forma de estruturação dos registros UBR. São elas (HENDRICKS, 2002):

- **Páginas Brancas**: neste tipo de registro o UDDI contém informações sobre nomes, endereços, números de telefone, além de outras informações sobre os negócios.
- **Páginas Amarelas**: contém listagens comerciais baseadas nos tipos desses negócios. Da mesma maneira, o UDDI permite a inserção de dados classificados por um tipo de negócio ou pela indústria em que ela opera.
- **Páginas Verdes**: são usadas para indicar os serviços oferecidos por cada negócio, incluindo todas as informações técnicas envolvidas na interação com o serviço, ou com a sua utilização, como parâmetros, valores da extremidade, etc.

Com o passar do tempo, a realidade mostrou que na maioria das aplicações os serviços *web* não eram para uso público, mas fortemente para uso interno das organizações ou sócios empresariais confiáveis. Devido a este fato, o enfoque da especificação UDDI evoluiu para permitir variadas implementações do padrão, inclusive registros públicos como o UBR e registros privados. Estes registros podem ser implementados em produtos de *software* como servidores de aplicação ou outros produtos *stand-alone* que podem ser desenvolvidos dentro de determinada empresa. O quadro c.6.1 apresenta os tipos de registros disponíveis na versão 3 do UDDI.

Quadro C.6.1: Tipos de registros previstos na especificação UDDI v.3

Tipo de Registro	Descrição	Analogia com a *Web*
Privado	É um registro interno, protegido por um firewall, isolado da rede pública. O acesso às tarefas administrativas e aos dados do registro é de forma segura. Os dados não são compartilhados com outros registros.	Intranet
Semi-Privado	É um registro desenvolvido dentro de um ambiente controlado. O acesso para o mundo externo é controlado e é compartilhado apenas com sócios externos confiáveis. Tarefas administrativas podem ser delegadas a partes confiáveis. Podem ser compartilhados dados com outros registros de um modo controlado.	Extranet
Público	Da perspectiva de um usuário final, um registro público parece ser um serviço em uma "nuvem". Ainda que possam ser efetuadas funções administrativas de forma segura, o acesso aos dados dos registros é essencialmente aberto e público. Os dados podem ser compartilhados ou transferidos para outros registros.	*Site Web*

A figura c.6 apresenta os possíveis relacionamentos entre os registros armazenados pelo UDDI versão 3.

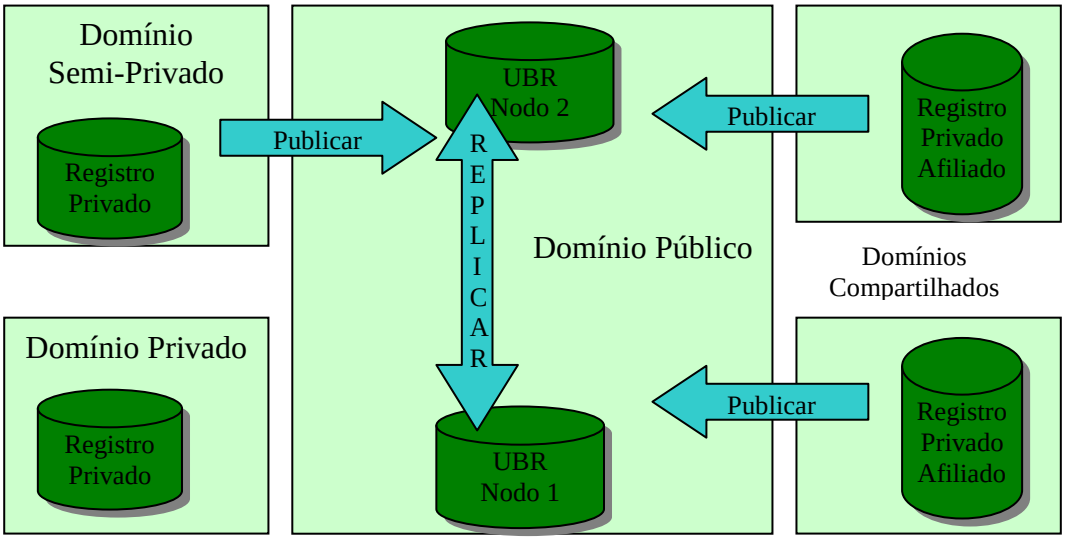

Figura C.6: Diagrama de interação dos registros UDDI versão 3.

O UDDI descreve o registro de um serviço *Web* e uma interface para publicar, recuperar e gerenciar informações sobre o serviço descrito. A especificação UDDI define serviços que possibilitam a descrição e descoberta de negócios, organizações e

outros provedores de serviço. Além disso, disponibiliza o serviço *web* e uma interface técnica para acessar e gerenciar este serviço. O UDDI é baseado em vários outros padrões de indústria já estabelecidos, inclusive HTTP, XML, XML Schema (xsd), SOAP e WSDL.

2.6. Modelo de dados do UDDI

O núcleo do modelo de informação usado por um registro de UDDI é definido em vários *schemas* de XML. O XML foi escolhido porque oferece uma plataforma-neutra de dados e permite descrever relações hierárquicas de um modo natural. O XSD foi escolhido por causa do seu suporte a um rico conjunto de tipos de dados e a sua habilidade de facilmente descrever e validar informações baseadas nos modelos representados nos *schemas*. O UDDI XSDS define alguns tipos centrais de centro de informação que provê os tipos de informação que os usuários e aplicações precisariam conhecer por usar um serviço de rede particular. Junto, estes formam um modelo básico de informação e um framework para interação de registros UDDI (OASIS, 2004). Eles são:

- uma descrição da função de negócio de um serviço (*businessService*);
- informação sobre a organização que publicou o serviço (*businessEntity*);
- os detalhes técnicos do serviço (*bindingTemplate*), inclusive uma referência para a interface do serviço ou API; e
- vários outros atributos ou metadados como taxonomia, transportes, assinaturas digitais, etc (*tModels*).

A versões 2 e 3 do UDDI adicionaram dois tipos para facilitar a afiliação dos registros, são eles respectivamente:

- relacionamento entre entidades no registro (*publisherAssertion*); e,
- localizar mudanças para uma lista de entidades (*subscription*).

Estes, como todos os tipos de dados UDDI, são expressos em XML e é armazenado persistentemente por um registro de UDDI. Dentro de um registro de UDDI, para cada estrutura de dados é definido um identificador de acordo com um esquema padrão. Este identificador é chamado de UDDI *key*.

A forma como os serviços são definidos no documento UDDI é chamado *Type Model* ou *tModel*. Em muitos casos, o *tModel* contém o arquivo WSDL que descreve a interface SOAP do serviço *web*, mas o *tModel* é flexível o suficiente para descrever quase todo tipo de serviço. A figura c.7 apresenta graficamente o modelo de dados do UDDI. O diretório UDDI também inclui várias maneiras de procurar os serviços, por exemplo, pode-se procurar por fornecedores de um serviço em uma localização geográfica específica ou por negócios de um tipo específico.

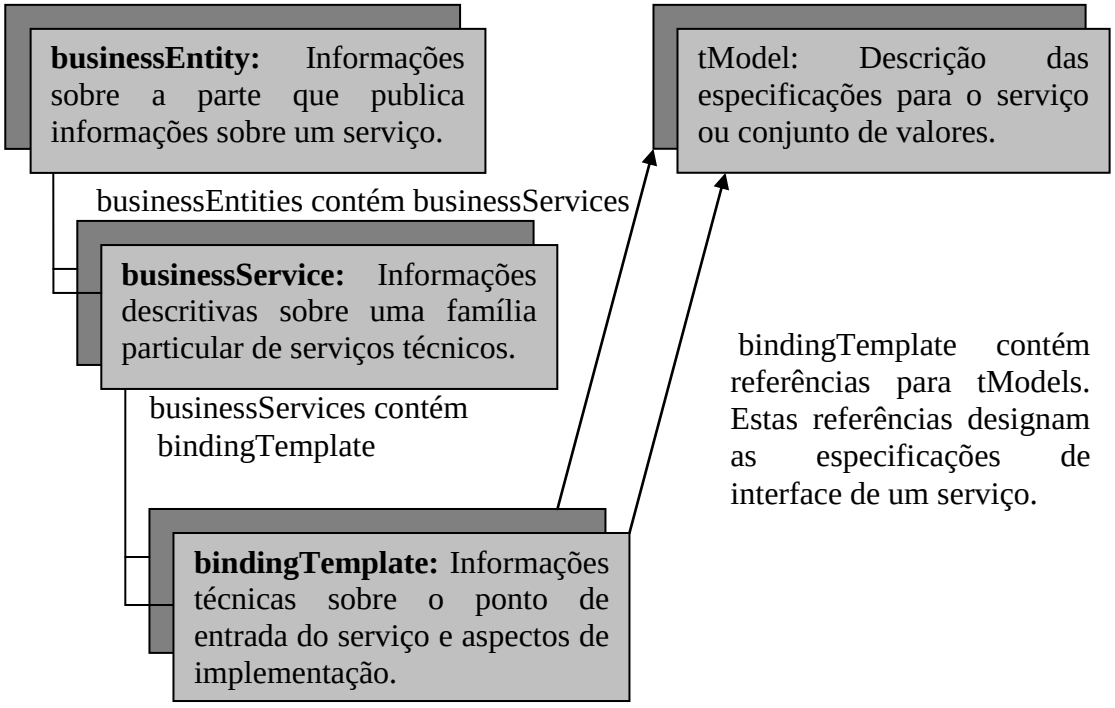

Figura C.7: Modelo de dados UDDI