

**ADAGA: sistemA de Detecção de Anomalias para o
Gerenciamento Autônomo de redes virtuais**

Pedro Silveira Pisa

DEL / POLI / COPPE / UFRJ

Projeto submetido para a obtenção do título de

Engenheiro de Computação e Informação

à Escola Politécnica da UFRJ

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
COPPE - ESCOLA POLITÉCNICA
DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

**ADAGA: sistema de Detecção de Anomalias para o
Gerenciamento Autônomo de redes virtuais**

Autor:

Pedro Silveira Pisa

Orientadores:

Prof. Otto Carlos Muniz Bandeira Duarte, Dr.Ing.

Natalia Castro Fernandes, M.Sc.

Examinadores:

Prof. Igor Monteiro Moraes, D.Sc.

Prof. Marcelo Gonçalves Rubinstein, D.Sc.

Prof. Miguel Elias Mitre Campista, D.Sc.

Prof. Pedro Braconnot Velloso, Dr.

Engenharia de Computação e Informação

Março de 2011

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Escola Politécnica - Departamento de Eletrônica e de Computação

Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária

Rio de Janeiro - RJ CEP 21949-900

Este exemplar é de propriedade da Universidade Federal do Rio de Janeiro, que poderá incluí-lo em base de dados, armazenar em computador, microfilmear ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

À minha família e à minha namorada Claudia

Agradecimentos

Aos meus pais, Wilson e Lucia, por todo amor, incentivo e orientação passados em todos os anos de minha vida. À minha irmã Heloisa, pelo companheirismo e amizade. À minha avó, Mercedes, pelo amor, pelos cuidados e pela dedicação. À minha namorada, Claudia, pelo amor, pelos momentos felizes e por toda a compreensão nos momentos de dificuldade. Agradeço a todos pela compreensão com a hora de chegada depois de longas reuniões no GTA. Sem eles, a conclusão dessa importante fase nunca aconteceria.

Aos amigos da Graduação, em especial para o Hugo Eiji, Daniel Vega, Diogo Ferrazani, João Pedro Francese, Pedro Coutinho e Leonardo Arnt, pelos momentos de diversão, como as risadas e brincadeiras dentro e fora da sala de aula, e de concentração, como nos estudos para a prova e reuniões de trabalhos e por toda a ajuda que me deram durante a graduação.

Ao professor e orientador Otto, responsável por grande parte da minha formação acadêmica e profissional, e a co-orientadora Natalia Castro Fernandes, por seus conselhos e orientação. Aos professores Luís Henrique, Miguel e Igor pelos ensinamentos e pelas dicas. Aos professores do Departamento de Eletrônica e de Computação e dos programas de Sistemas, Elétrica, Produção e Civil da COPPE, que ministraram aulas me ensinando mais do que o conhecimento da Engenharia; ensinaram lições de vida e me deram diferentes de visões de mundo, enriquecendo minha formação pessoal e profissional.

Aos professores Otto Carlos Muniz Bandeira Duarte, Igor Monteiro Moraes, Marcelo

Gonçalves Rubinstein, Miguel Elias Mitre Campista, Pedro Braconnot Velloso e à Natalia Castro Fernandes pela presença na banca examinadora e pela contribuição neste trabalho

Aos amigos e colegas do GTA, Natalia, Marcelo, Lino, Hugo, Diogo, Carlo, Rafael, pelos conselhos, pela troca de experiências e pela grande ajuda neste trabalho. A todos os alunos de iniciação científica, que ajudaram no desenvolvimento deste trabalho com ideias desprendidas de preconceitos e padrões pré-estabelecidos.

A todos que me incentivaram, contribuindo de forma direta ou indireta, para a minha formação pessoal e profissional e pelo aprendizado absorvido nesses cinco anos de graduação.

A FINEP, CNPq, PIBIC, CAPES, FUNTTEL e FAPERJ pelo financiamento da pesquisa.

Resumo

Este trabalho propõe um sistema de geração de alarmes baseado na detecção de anomalias para o gerenciamento autônomo de redes virtuais. O sistema é composto por mecanismos de monitoramento de redes virtuais, de caracterização de tráfego e de avaliação de características dos roteadores físicos e virtuais. Os mecanismos visam evitar a revelação de dados privados dos usuários, processando e armazenando apenas estatísticas dos dados coletados. O comportamento do roteador é representado em séries temporais, extraídas das estatísticas obtidas. Dois mecanismos preditores de séries temporais foram avaliados: um simples e outro mais complexo, que decompõe as séries temporais em tendência, sazonalidade e ruído. Um protótipo foi desenvolvido para analisar o sistema proposto em um roteador real. Os resultados obtidos mostram o impacto da utilização de diversos parâmetros tais como tráfego de rede, processos em execução e uso de memória sobre a detecção correta das anomalias da rede, analisando-se os falsos positivos e os falsos negativos. Com o mecanismo preditor mais complexo, obteve-se 2,5% de falsos positivos e 0% de falsos negativos na melhor configuração de parâmetros avaliada.

Palavras-Chave

Internet do Futuro

Gerenciamento

Sistema de Detecção de Anomalia

Sistema de Detecção de Intrusão

Autonomia

Detecção

Anomalia

Abstract

This work proposes an anomaly-detection-based alarm generator system for the autonomous management of virtual networks. The system consists of mechanisms for virtual network monitoring, traffic characterization, and evaluation of physical and virtual router characteristics. The proposal does not expose private data of users because only processes and stores statistics of collected data. Time series are extracted from the statistics of collected data to represent the router behavior. We evaluate two time series predictors: a simple one and another one that decomposes time series in trend, seasonality, and noise. The results of our prototype show the impact of using different characteristics of routers such as network traffic, running processes, and memory usage on the correct anomaly detection by analyzing false positives and false negatives percentages. Our results show 2,5% of false positives and 0% of false negatives in the best configuration of the parameters.

Keywords

Future Internet

Network Management

Anomaly Detection System

Intrusion Detection System (IDS)

Autonomous System

Detection

Anomaly

Lista de Abreviaturas

ACID	<i>Atomic, Consistent, Isolated, and Durable</i> , p. 31
API	<i>Application Programming Interface</i> , p. 29
CABO	<i>Concurrent Architectures are Better than One</i> , p. 8
CIA	<i>Central Intelligence Agency</i> , p. 4
DHCP	<i>Dynamic Host Configuration Protocol</i> , p. 5
DNS	<i>Domain Name Service</i> , p. 5
DOM	<i>Document Object Model</i> , p. 29
DoS	<i>Denial of Service</i> , p. 13
GUI	<i>Graphical User Interface</i> , p. 28
IP	<i>Internet Protocol</i> , p. 1, 3, 13, 29, 35
ISP	<i>Internet Service Providers</i> , p. 8
MAC	<i>Medium Access Control</i> , p. 35
NAT	<i>Network Address Translation</i> , p. 5, 30, 53
P2P	<i>Peer to Peer</i> , p. 3
RAM	<i>Random Access Memory</i> , p. 44

LISTA DE ABREVIATURAS

SGBD	Sistema Gerenciador de Banco de Dados, p. 31, 37
SNMP	<i>Simple Network Management Protocol</i> , p. 4
SSH	<i>Secure Shell</i> , p. 35, 45, 46
TCP	<i>Transmission Control Protocol</i> , p. 1, 16, 29, 35, 45, 46
TLS	<i>Transport Layer Security</i> , p. 53
UDP	<i>User Datagram Protocol</i> , p. 35
W3C	<i>World Wide Web Consortium</i> , p. 29
WWW	<i>World Wide Web</i> , p. 3
XML	<i>eXtensible Markup Language</i> , p. 18, 29, 36

Sumário

Resumo	vi
Abstract	viii
Lista de Abreviaturas	x
Lista de Figuras	xv
Lista de Tabelas	xvii
I Introdução	1
I.1 Conceitos sobre a Internet	3
I.1.1 Arquitetura para a Internet do Futuro	4
Modelo Purista	7
Modelo Pluralista	7
I.2 Arquitetura de Redes Virtuais	8
I.2.1 Monitoramento de Redes Virtuais	9
I.3 Características Gerais	11
I.4 Organização do Projeto	14
	xii

SUMÁRIO

II Sistema ADAGA	15
II.1 Arquitetura	16
II.2 Coleta e Representação	18
II.3 Séries Temporais	20
II.4 Mecanismos de Predição	21
II.4.1 <i>Exponential Smoothing</i>	22
II.4.2 <i>Holt-Winters Seasonal</i>	23
II.5 Geração de Alarmes	25
III Detalhamento da Implementação	27
III.1 Tecnologias Utilizadas	28
III.2 Comunicação Sistema/Gerente	31
III.3 Coleta de Dados	34
III.4 Representação dos Dados	36
III.5 Gerência de Séries Temporais	39
III.6 Mecanismos de Predição	40
III.6.1 Adição de Novos Mecanismos de Predição	40
III.6.2 Mecanismos Preditores Implementados	41
Mecanismo Base	41
<i>Exponential Smoothing</i>	42
<i>Holt-Winters Seasonal</i>	42
III.7 Gerenciamento dos Alarmes	42

SUMÁRIO

IV Avaliação do Sistema	44
IV.1 Resultados	46
V Conclusão	53
Referências Bibliográficas	57
A Trechos de Código Fonte	62
A.1 Estrutura Básica da Mensagem do ADAGA	62
A.2 Sistema de Comunicação	63
A.3 Séries Temporais	70
A.4 Mecanismos Preditores	72

Lista de Figuras

I.1	Atividades de controle de um sistema autônomo para a Internet.	6
II.1	Arquitetura do ADAGA.	16
II.2	Possibilidades de relacionamento entre gerentes de monitoramento de sistemas monitorados no sistema ADAGA.	17
II.3	Estrutura de representação dos elementos da rede no sistema ADAGA. . .	19
II.4	Influência das observações passadas no cálculo da predição da amostra $t = 50$ para a característica pacotes transmitidos por todas as interfaces. .	23
III.1	Diagrama de relacionamento das classes do servidor TCP executado no sistema monitorado.	32
III.2	Modelo de dados do banco de dados usado para armazenar as observações e cálculos realizados no ADAGA.	37
IV.1	Cenário de testes para análise dos falsos positivos e falsos negativos do sistema ADAGA.	45
IV.2	Evolução temporal da característica “recepção de pacotes na porta 22 do TCP” de cada mecanismo para α igual a 0,1.	47
IV.3	Evolução temporal para o mecanismo <i>Holt-Winters Seasonal</i> de duas características não correlatas diretamente com o tráfego de rede.	48

LISTA DE FIGURAS

- IV.4 Análise para a característica “recepção de pacotes na porta 22 do TCP”
para o mecanismo *Exponential Smoothing*, variando o parâmetro η para
diversos valores de α 49
- IV.5 Análise para a característica “recepção de pacotes na porta 22 do TCP”
para o mecanismo *Holt-Winters Seasonal*, variando o parâmetro η para
diversos valores de α , β e γ 50

Lista de Tabelas

IV.1 Tabela comparativa das taxas de falsos positivos para as configurações de α , β e γ apresentadas nas Figuras IV.5(a), IV.5(c) e IV.5(e) com $\eta = 5$. Valores percentuais.	51
IV.2 Tabela comparativa das taxas de falsos negativos para as configurações de α , β e γ apresentadas na Figura IV.5(b), IV.5(d) e IV.5(f) com $\eta = 5$. Valores percentuais.	51

Lista de Códigos Fonte

A.1	Definição da estrutura XML do cabeçalho da mensagem de requisição esperada pelo servidor construído.	62
A.2	Servidor da comunicação entre sistema monitorado e gerente de monitoramento. Executa no sistema monitorado.	63
A.3	Cliente da comunicação entre sistema monitorado e gerente de monitoramento. Executa no gerente de monitoramento.	68
A.4	Métodos da classe de controle das séries temporais.	70
A.5	Método que processa os alarmes reportados pelas séries temporais.	71
A.6	Métodos responsáveis pelo carregamento dinâmico de mecanismos preditores sem que o sistema precise ser reiniciado.	72
A.7	Mecanismo de predição base para outros mecanismos de predição.	73
A.8	Métodos adicionais e alterados no mecanismo <i>Exponential Smoothing</i> em relação ao mecanismo preditor base.	76
A.9	Métodos adicionais e alterados no mecanismo <i>Holt Winters Seasonal</i> em relação ao mecanismo preditor base.	77

Capítulo I

Introdução

O modelo de núcleo simples e inteligência nas extremidades e os protocolos TCP/IP foram os maiores responsáveis pelo grande sucesso da Internet. No entanto, este mesmo modelo engessa o núcleo da rede, pois não é possível configurar ou inserir novos elementos no núcleo da rede. Além disso, a baixa gerenciabilidade da rede implica configurações manuais, difícil depuração de erros e barreiras na implantação de novas tecnologias. Existe um consenso na comunidade científica sobre a necessidade de uma nova Internet [1], a Internet do Futuro. A nova Internet é baseada na tecnologia de virtualização, que permite diversas redes virtuais, executando protocolos distintos, funcionarem ao mesmo tempo e de forma isolada sobre o mesmo substrato físico [2, 3]. Na Internet do Futuro, o gerenciamento autônomo se apresenta como uma solução para o gerenciamento escalável das redes de computadores, pois, dessa forma, as redes se autoconfiguram, autorreparam e autoaperfeiçoam [4]. Para que um sistema seja autônomo, é necessário que ele possa coletar e analisar dados do ambiente, decidir a melhor solução e aplicá-la aos equipamentos da rede [5]. O monitoramento e a coleta de dados em redes virtualizadas introduzem certas especificidades e ainda é pouco abordado. Dentre elas podemos citar o aumento da complexidade de monitoramento e análise, pois a quantidade de características monitoradas é a soma das características monitoradas em cada rede virtual, além do monitoramento do sistema físico e da própria plataforma de virtualização. Além disso, correlações nas medidas observadas em redes virtuais diferentes podem acontecer, sobretudo devido ao

compartilhamento do meio físico.

Este projeto descreve o ADAGA, sistema de Detecção de Anomalias para o Gerenciamento Autônomo de redes virtuais, que provê mecanismos de coleta e análise dos dados em um ambiente de redes virtuais. Através do ADAGA, gerentes autônomos de monitoramento podem observar os sistemas monitorados na rede, tais como servidores e roteadores, físicos e virtuais. O objetivo do sistema proposto é a emissão de alarmes relatando possíveis anomalias nos equipamentos da rede e, para isso, este trabalho caracteriza o comportamento anômalo na rede como mudanças de curto prazo nas observações coletadas que são inconsistentes com o passado.

O ADAGA utiliza séries temporais para prever o valor atual segundo o passado da série e confrontá-lo com a nova observação. O ADAGA considera todas as séries temporais iniciadas em zero. Para isso, decrementa de todas as observações, o valor inicial da série, permitindo erro zero na inicialização e nenhuma influência da condição inicial nas previsões futuras. A inicialização correta dos preditores é uma importante configuração, pois impacta o desempenho de todo o sistema [6]. Dois mecanismos preditores foram implementados e avaliados, mas o sistema permite a inclusão dinâmica de novos mecanismos preditores. Dos preditores analisados, ambos utilizam o conceito da média móvel para a previsão do próximo valor. No entanto, o primeiro, mais simples, aplica o conceito de média móvel para o valor completo da série e o segundo decompõe a série em tendência, sazonalidade e ruído, aplicando a cada uma dessas componentes uma equação de média móvel.

A análise dos preditores é realizada com base nos falsos positivos e falsos negativos conforme são variados os diversos parâmetros dos preditores. Os resultados deste projeto analisam o comportamento de características observadas nos sistemas monitorados e seu impacto na emissão de alarmes em uma situação de anomalia. Os testes são realizados em um roteador real e anomalias foram geradas para simular uma sobrecarga no roteador. Os resultados mostram que o sistema ADAGA é capaz de detectar a anomalia com taxas de falsos positivos de 2,5% e taxas de falsos negativos de 0% nas melhores configurações de parâmetros avaliadas

I.1 Conceitos sobre a Internet

A Internet é um projeto da década de 1970, que surgiu a partir de redes militares e universitárias. Tinha por objetivo principal o envio de mensagens eletrônicas entre as diversas entidades participantes da rede, que eram usuários especializados. Em seu projeto original, priorizou-se a simplicidade no núcleo da rede, que representa os elementos intermediários necessários para prover a conectividade entre os sistemas finais, responsáveis pelas aplicações que utilizavam a rede. Dessa forma, toda a inteligência da rede se localizava nas extremidades e o núcleo da rede ficou responsável pelo encaminhamento dos pacotes transmitidos, semelhante a um sistema de correio convencional. O protocolo responsável pelo endereçamento dos pacotes é utilizado até hoje e é conhecido como *Internet Protocol* (IP). Esse modelo possibilita a absorção de novas aplicações e novos protocolos na Internet, pois todo o núcleo da rede permanece inalterado, necessitando apenas que os sistemas finais, nas extremidades da rede, recebam a implementação dessas novas aplicações. Tal característica permitiu o rápido crescimento de aplicações como a *World Wide Web* (WWW) e as redes Par a Par (P2P), que são um sucesso.

No entanto, a maior vantagem da Internet é também a sua principal desvantagem. Diversas aplicações, como a mobilidade e a correção de falhas, se aproveitariam de mecanismos inteligentes no núcleo da rede se eles existissem. A restrição para a mobilidade reside na sobrecarga semântica do endereço IP, que acumula as funções de identificador e localizador. Dessa forma, um endereço IP está associado a uma localização na rede, sendo necessária a sua mudança quando o usuário se move. Ao mudar o endereço IP, as conexões criadas entre as extremidades se perdem. Sistemas inteligentes e capazes de realizar o procedimento de *handover* auxiliariam na manutenção da comunicação de forma transparente para o usuário e para a rede. Protocolos como o *Host Identity Protocol* (HIP) [7, 8] visam remover essa sobrecarga semântica permitindo a mobilidade na rede.

As falhas que ocorrem na rede não são corretamente relatadas, pois os elementos do núcleo da rede são simples e não informam sobre o funcionamento da rede. Além da própria inexistência de registros das falhas na rede, a Internet é fragmentada em sistemas

I.1 Conceitos sobre a Internet

autônomos, cada um responsável por gerir um pedaço da rede [9]. Essa fragmentação da gerência da rede representa que as informações coletadas por um sistema autônomo podem não estar disponíveis para outros sistemas autônomos. Nesse cenário, mecanismos de diagnóstico de rede necessitam ser inseridos nas extremidades para descobrir o real problema da rede. Sistemas monitores inseridos e disseminados no núcleo da rede e com capacidade de comunicação entre si podem oferecer informações mais concretas sobre as falhas ocorridas na rede. O sistema ADAGA, proposto neste trabalho, visa a coleta de informações no núcleo da rede através de agentes inteligentes que se comunicam para obter os dados de cada elemento de rede.

Na década de 1980, o *Simple Network Management Protocol* (SNMP) [10] foi definido e se tornou o protocolo padrão para o gerenciamento da Internet. Atualmente, os equipamentos de rede implementam o protocolo SNMP, mas sua atuação é limitada ao monitoramento dos equipamentos, sem atuar na gerência das configurações dos protocolos e aplicações utilizados nos equipamentos. Com as demandas atuais, ele é visto como ultrapassado, complexo, lento e inseguro pelos administradores de rede. O sistema ADAGA utiliza mecanismos próprios para a coleta de dados, pois os agentes específicos criados para a coleta dos dados são mais flexíveis, já que podem aprender novas métricas de coletas.

I.1.1 Arquitetura para a Internet do Futuro

Atualmente, a Internet se consagra com um sucesso inquestionável. Segundo a Agência Central de Inteligência (*Central Intelligence Agency* - CIA) dos Estados Unidos, em 2009, existiam mais de 1 bilhão e 800 milhões de usuários de Internet no mundo¹. O Brasil era o quarto país em número de usuários da Internet, com quase 76 milhões de usuários, atrás apenas da China, dos Estados Unidos e do Japão. Com esse sucesso, o objetivo principal da Internet e o nível de especialização dos seus usuários mudaram radicalmente. O seu sucesso é a exposição das suas maiores fraquezas. Demandas por novas aplicações surgem

¹<https://www.cia.gov/library/publications/the-world-factbook/fields/2153.html>

I.1 Conceitos sobre a Internet

constantemente e os requisitos que nortearam a criação Internet são colocados à prova. A evolução para a rede atual ocorreu com base em “remendos” nos sistemas criados originalmente, como o *Network Address Translation* (NAT), o *Dynamic Host Configuration Protocol* (DHCP) e o *Domain Name Service* (DNS). O NAT e o DHCP são mecanismos criados para postergar a escassez de endereços IP disponíveis. O NAT [11] permite que diversos dispositivos se conectem à Internet utilizando um mesmo endereço IP, quebrando premissas da Internet como o endereço IP global e a comunicação fim-a-fim, já que é necessário um elemento intermediário na rede para fazer a tradução do endereço interno do NAT para o endereço aparente na Internet. O DHCP [12] é um mecanismo de alocação dinâmica de endereços e permite que dispositivos diferentes utilizem um mesmo endereço IP em momentos distintos, otimizando a distribuição dos endereços disponíveis para um dado domínio da rede. O DNS [13] é um protocolo de mapeamento criado para facilitar a memorização dos endereços da Internet na forma de nomes relacionados ao destino da mensagem.

Assim, existe um consenso entre os pesquisadores de que a arquitetura da Internet precisa ser reformulada [1]. Considera-se que a arquitetura original da Internet a engessa, dificultando a inovação no núcleo da rede. A comunidade científica defende que a reformulação seja completa, ignorando a compatibilidade com a Internet atual, como no projeto *Clean Slate* [14]. Dessa forma, uma nova Internet deve ser criada do zero, considerando os novos requisitos da Internet e mantendo os itens que garantiram o enorme sucesso que é a Internet, como a facilidade na implantação de novos serviços e a adaptabilidade dos protocolos da rede a esses novos serviços.

Na nova arquitetura para a Internet, a palavra chave é autonomia. Devido à escala e à complexidade da Internet, os equipamentos do núcleo da rede devem ser capazes de se autogerir, autoconfigurar, autorecuperar e autoaperfeiçoar. Todas essas características compõem o gerenciamento autônomo da Internet e um sistema que as possua consiste de mecanismos inteligentes e de uma base de conhecimento que determinam as ações dos sistemas autônomos frente a um desafio. O ciclo natural de todo sistema autônomo, conforme descrito por Dobson *et. al.* [5], consiste em monitorar o ambiente, analisar os dados

I.1 Conceitos sobre a Internet

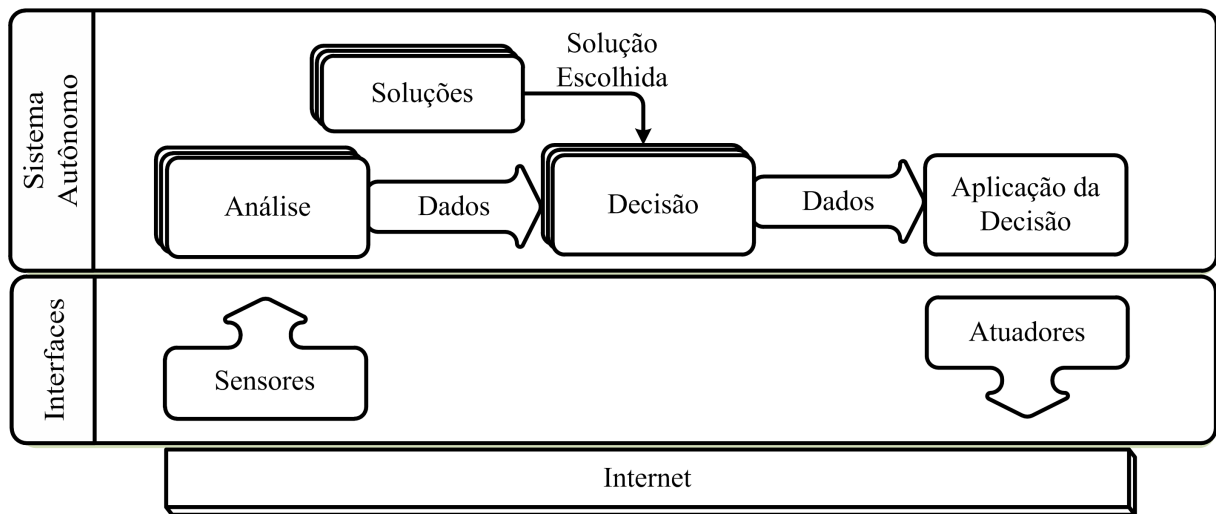


Figura I.1: Atividades de controle de um sistema autônomo para a Internet.

coletados, decidir por realizar uma ação e atuar no ambiente. A Figura I.1 representa o laço de controle típico de um sistema autônomo. No entanto, um sistema autônomo executa diversos mecanismos de análise e tomada de decisão, os quais possuem seus mecanismos próprios de sensoriamento da rede. Esses diversos mecanismos de monitoramento da rede representam uma sobrecarga para o equipamento da rede, sobretudo uma vez que podem monitorar as mesmas variáveis de rede, como, por exemplo, a quantidade de bytes recebidos em um dado instante. O ADAGA é um sistema para o gerenciamento autônomo para a Internet do Futuro que consiste em agregar em um único sistema, as etapas de sensoriamento da rede e análise dos dados coletados. Através do sistema ADAGA, sistemas autônomos específicos para a decisão e atuação na rede podem ser simplificados, uma vez que não monitoram a rede nem precisam detectar uma anomalia. Os mecanismos de decisão e atuação na rede são acionados apenas quando um alarme é gerado pelo sistema ADAGA, reduzindo a sobrecarga de controle nos equipamentos da rede.

Portanto, no contexto da concepção da nova Internet, pautada nos princípios que levaram o sucesso à Internet atual, sem a preocupação com a compatibilidade, e com novos conceitos como a autonomia, existem dois modelos principais para a construção da arquitetura flexível da Internet do Futuro: o modelo purista e o modelo pluralista.

I.1 Conceitos sobre a Internet

Modelo Purista

Os defensores do modelo purista acreditam que uma arquitetura para a Internet do Futuro deve possuir a flexibilidade em sua concepção. Essa arquitetura deve englobar todas as ferramentas necessárias para assegurar a integração de soluções específicas para diversos desafios de qualidade de serviço, roteamento e gerenciamento, entre outras. Atualmente, diversas propostas para problemas específicos da rede existem e são incompatíveis. No modelo purista, elas serão integradas na arquitetura da rede. Dessa forma, para buscar a flexibilidade na arquitetura, os puristas não esperam impactos imediatos para os usuários da rede, pois a rede tem a capacidade de absorver gradualmente as inovações. Tecnologias como as redes sobrepostas (*overlays*) e a virtualização são encaradas como ferramentas para agregar as novas funcionalidades à arquitetura.

Modelo Pluralista

Os defensores do modelo pluralista acreditam na coexistência de múltiplas pilhas de protocolo ao mesmo tempo na Internet. Nessa visão, técnicas como a virtualização de redes e as redes sobrepostas são fundamentais para a arquitetura. Assim, cada rede instanciada objetiva a solução de um problema específico, facilitando a implantação e permitindo a percepção imediata das inovações no núcleo da rede. Além disso, os pluralistas defendem que a arquitetura é mais dinâmica e permite evoluções maiores, já que não requer a integração das soluções propostas para problemas específicos, visto que cada uma das propostas é instanciada em redes sobrepostas ou multiplexadas por uma ferramenta de virtualização simultânea e isoladamente. No modelo pluralista, a migração para a nova Internet é automática, uma vez que a Internet atual pode ser instanciada e compor uma das pilhas de protocolos utilizadas.

O sistema ADAGA, proposto e detalhado neste projeto, é concebido para o modelo de redes pluralista, no qual as redes virtuais são parte da arquitetura da nova Internet. O modelo pluralista é adotado nesse trabalho devido às vantagens de implantação, evolução e coexistência de novas soluções para a Internet. Assim, o ADAGA é projetado para o

I.2 Arquitetura de Redes Virtuais

monitoramento de redes virtuais em um ambiente pluralista.

I.2 Arquitetura de Redes Virtuais

A Internet atual é fragmentada em sistemas autônomos, também chamados de *Internet Service Providers* (ISP). Assim, os sistemas autônomos são responsáveis por prover tanto os serviços da rede quanto a infraestrutura de acesso. Além disso, os ISPs possuem gerência apenas sobre a sua rede, inviabilizando a oferta de serviços diferenciados, pois não detém o controle de todo o caminho fim-a-fim. Com o objetivo de flexibilizar este cenário, propostas surgem com o objetivo de separar os papéis acumulados pelo ISP. O projeto *Concurrent Architectures are Better than One* (CABO) [15] defende a utilização de roteadores virtuais, sob responsabilidade dos provedores de serviço, executando sobre os roteadores físicos, geridos pelos provedores de infraestrutura. Sob o aspecto dos provedores de infraestrutura, continua sendo inviável um mesmo provedor deter o controle de todo o caminho fim-a-fim, devido aos aspectos técnicos, sobretudo de escala da infraestrutura que seria necessária. No entanto, para os provedores de serviço, o controle de todo o caminho fim-a-fim é possível, uma vez que os roteadores virtuais são instâncias de software executados sobre o substrato físico dos provedores de infraestrutura. Com essa arquitetura, cada provedor de serviço pode alterar a configuração de todos os seus equipamentos ou até mesmo utilizar uma pilha completamente nova de protocolos, sem que os demais provedores de serviços sejam afetados.

Outras propostas semelhantes são apresentadas, como é o caso do projeto 4WARD [16, 17], que defende a existência de três entidades: os provedores de infraestrutura, os provedores de redes virtuais e os operadores das redes virtuais. A organização é semelhante à proposta pelo projeto CABO, mas subdivide o provedor de serviço em duas entidades distintas. O provedor de redes virtuais seria responsável por contratar redes virtuais e garantir os contratos com os provedores de infraestrutura. O operador de redes é a entidade usuária da rede, quem realmente proverá o serviço.

I.2 Arquitetura de Redes Virtuais

O elemento comum das propostas é a dissociação entre o responsável pela infraestrutura e o provedor de serviço através da virtualização de redes. Nesse contexto, a virtualização das redes recebeu um importante papel na Internet do Futuro. Com ela possibilita-se a realização das propostas de arquitetura para a rede, sobretudo no modelo pluralista. Com a utilização da virtualização, é possível *criar*, *instanciar*, *mover* e *remover* roteadores virtuais da rede sob demanda e com agilidade. Tais primitivas permitem agregar à rede uma série de características como a manutenção de equipamentos físicos sem queda no serviço, economia de energia, garantia de qualidade de serviço dentre outras. Esse trabalho se concentra na área de virtualização de redes e no modelo pluralista de redes. Mais especificamente, os roteadores virtuais são desenvolvidos sobre a plataforma de virtualização Xen [18], que foi projetado para a virtualização de servidores em *data-centers*. Por esse motivo, o Xen utiliza a plataforma de computadores pessoais padrão de mercado, com arquitetura x86 e amd64, dentre outras.

I.2.1 Monitoramento de Redes Virtuais

A utilização das primitivas *criar*, *instanciar*, *mover* e *remover* da virtualização para a gerenciar as redes virtuais conforme a demanda e os contratos de qualidade de serviços (*Service Layer Agreements* - SLA) requer que os dispositivos atuadores da rede sejam disparados de forma autônoma. Os sistemas autônomos possuem o ciclo de controle apresentado na Figura I.1 e as primitivas da virtualização estão localizadas no grupo da Aplicação das Soluções. No começo do ciclo de controle estão os sensores que monitoram os sistemas. Neste trabalho, apresenta-se o ADAGA, sistema de Detecção de Anomalias para o Gerenciamento Autônomo de redes virtuais, que monitora e analisa os dados em redes virtuais para o gerenciamento autônomo.

Diversos outros sistemas já foram propostos e analisados para o monitoramento da rede e a detecção de anomalias [19]. No entanto, nenhum, ao contrário do ADAGA, é concebido para a análise multidimensional da rede em ambiente virtualizado. Tal concepção definida em tempo de projeto é um fator importante para determinar a validade do

I.2 Arquitetura de Redes Virtuais

sistema de detecção de anomalias, pois os sistemas virtualizados possuem peculiaridades em relação aos sistemas tradicionais provocados pela simples coexistência de múltiplas redes simultaneamente em dado equipamento.

Como peculiaridade mais imediata, tem-se a maior complexidade do monitoramento e da análise para detecção de anomalia, pois o sistema deve analisar, em tempo real, diversas características de cada uma das redes presentes em cada sistema monitorado. Assim, a quantidade de características analisadas é o produto entre número de características monitoradas nas redes virtuais e a quantidade de redes virtuais mais o número de características monitoradas na rede física. Tal aumento de complexidade requer um sistema eficiente para permitir a reação às anomalias em tempo real. O sistema ADAGA é capaz de analisar uma rede virtual com 40 características em 10^{-4} segundos em um computador pessoal padrão de mercado, com processador Intel Core i7 950 de 3.06 GHz, devido ao tratamento de cada característica em uma *thread* separada e ao armazenamento de todos os dados usados para cada cálculo em memória. Assim, o ADAGA favorece o processamento rápido em detrimento do uso de memória. Essa desvantagem não se apresenta como um fator complicador para o sistema ADAGA uma vez que a quantidade de redes virtuais em cada sistema físico é limitada e, por consequência, a quantidade de redes virtuais monitoradas por uma instância de monitoramento do sistema ADAGA é limitada.

Além do aumento da complexidade de análise, a coexistência de várias redes em um mesmo equipamento significa que uma rede está influenciando nas outras, uma vez que não há isolamento dos dispositivos de entrada e saída no Xen [3]. Dessa forma, alarmes devem ser gerados para uma rede mesmo que não exista anomalia relacionada com o funcionamento da rede em si, pois pode haver uma anomalia em outra rede virtual ou mesmo na rede física que influencie o funcionamento da rede virtual em questão. Portanto, sistemas de monitoramento específicos para ambientes virtualizados devem ser construídos com base em requisitos adicionados aos já presentes em sistemas de detecção de anomalias. Um exemplo desses novos requisitos é a correlação existente entre os dados coletados em redes virtuais diferentes que compartilham um mesmo equipamento físico.

I.3 Características Gerais

Monitoramento e gerência de redes virtuais baseados em detecção de anomalia é um assunto pouco explorado. O monitoramento de cenários virtualizados é diferente do monitoramento de sistemas tradicionais, pois se devem observar os recursos físicos e virtuais. Dessa forma, a complexidade da análise aumenta, uma vez que o número de características monitoradas agrega todas as características monitoradas em todas as redes virtualizadas, a rede física e a própria plataforma de virtualização. O aumento de características monitoradas requer um sistema de monitoramento e análise mais eficiente para que o processamento seja feito em tempo real. Além disso, como as redes virtuais compartilham os recursos do *hardware* físico, fatores externos à rede virtual podem ocasionar alarmes em uma dada rede, obrigando o sistema a reconhecer alarmes provocados pelo funcionamento da rede em si, por outras redes ou ainda por problemas ocorridos na rede física. Portanto, o monitoramento de redes virtuais apresenta novos desafios para a área de gerência de rede.

As técnicas de detecção de anomalia são comumente usadas na área de segurança como, por exemplo, na detecção de intrusão [20]. A detecção de anomalias também tem sido usada em sistemas autônomos de gerência de rede que são acionados quando uma anomalia é detectada e um alarme, emitido. Anomalias são classificadas em três tipos [21]: anomalias por operação de rede, que consistem em falhas em equipamentos da rede ou mudanças de configurações; anomalias por *flash crowd*, que acontecem normalmente quando uma informação em dado local é requisitada por muitos usuários ao mesmo tempo, como a distribuição da nova versão de um sistema operacional ou um vídeo viral; e, por fim, anomalias por abuso da rede, como ataques de negação de serviço e varredura de portas. O sistema proposto considera todas essas anomalias, pois elas são relevantes para a satisfação dos usuários das redes virtuais gerenciadas autonomamente. Além dos tipos tradicionais de anomalias, em ambientes virtualizados existe o que classificamos como anomalias correlatas, que consistem nas anomalias detectadas em uma rede que não são provenientes do funcionamento da rede em si, mas de outras redes virtuais que compartilham o mesmo substrato físico ou do próprio mecanismo de virtualização.

I.3 Características Gerais

Brutlag usa séries temporais e mecanismos de predição para a detecção de anomalias em redes de computadores [6]. Esse trabalho foca na análise do tráfego de rede de um roteador para a geração de alarmes de anomalias. Uma única configuração de parâmetros do preditor é testada e a sua inicialização é feita em zero, mas nenhum pré-processamento é feito na série temporal, ao contrário do que é feito no ADAGA para se obter erro zero na inicialização. Além disso, a principal diferença entre sistema de Brutlag e o sistema ADAGA proposto é a análise de diversas características da rede, como uso de memória e de processamento, realizada no ADAGA. Por analisar várias características, o ADAGA realiza uma análise multidimensional, na qual os cálculos de séries temporais são aplicados a cada uma das características. O monitoramento de características como o uso de memória e de processamento, que não são monitoradas no trabalho de Brutlag [6] são importantes em cenários virtualizados, pois com o baixo isolamento dos recursos de rede nesses cenários, as anomalias no tráfego de rede não impactam apenas o desempenho da rede, mas impactam também o uso de processamento e memória em outras redes virtuais [3]. Portanto, todas essas características adicionais monitoradas pelo ADAGA são importantes em ambientes virtualizados. Além disso, operações de gerência da rede, como a migração de máquinas virtuais, impactam na operação e no monitoramento dos equipamentos da rede [22], pois, em um dado momento, máquinas virtuais que eram monitoradas no cenário de um sistema físico passam para outro sistema físico, o que pode gerar alarmes nos mecanismos de predição, já que o passado histórico da série temporal possui o padrão obtido no cenário da máquina física de origem.

O trabalho de Lucena e Moura analisa o tráfego de rede com foco em uma observação baseada em fluxos de pacotes [23]. Os autores definem por fluxo, um conjunto com mesmo endereço IP de origem, endereço IP de destino, porta de origem e porta de destino. A abordagem por fluxos utilizada define diversos tipos de anomalias, como a Negação de Serviço (DoS - *Denial of Service*), falhas de configuração e os *flash crowds*. No sistema ADAGA, os fluxos não são agrupados da maneira convencional. O sistema agrupa os pacotes por serviços, pois o objetivo é gerenciar a rede do ponto de vista da sua função, uma vez que a abordagem das redes pluralistas considera redes virtuais por serviços na Internet do Futuro [1]. Portanto, no ADAGA, considera-se um fluxo de pacotes pelo

I.3 Características Gerais

protocolo de transporte e a porta de destino, pois são as características dos pacotes que ajudam a definir o serviço do qual fazem parte.

Diversos trabalhos correlatos à detecção de anomalias apresentam intervalos de observação da ordem de unidades ou dezenas de minutos [6, 23–25], o que reduz os requisitos de processamento e armazenamento. Intervalos de medição desta magnitude, no entanto, não permitem reagir de forma rápida a importantes anomalias que ocorrem por um curto período de tempo. O sistema ADAGA propõe observações que permitam reagir rapidamente a anomalias e, para isso, o intervalo de observação utilizado foi uma distribuição de Poisson com média em 15 segundos. Os testes realizados com o protótipo apresentam desempenho satisfatório, visto que o processo de coleta e análise é feito em tempo real e a análise de 40 características diferentes de um sistema monitorado toma 10^{-4} segundos em um computador pessoal padrão de mercado, com processador Intel Core i7 950 de 3,06 GHz, permitindo a sua utilização em ambientes virtualizados com milhares de redes virtuais. A eficiência na análise multidimensional é importante em cenários virtualizados, pois o número de características analisadas é multiplicado pela quantidade de redes virtuais, em adição ao monitoramento da rede física e da plataforma de virtualização.

Além do intervalo de amostragem, outra importante característica dos sistemas detectores de anomalia é a taxa de amostragem de pacotes. Sistemas que requerem o processamento por pacote apresentam altas cargas de processamento e armazenamento se avaliarem todos os pacotes. Por esse motivo, diversos trabalhos realizam amostragens de pacotes [25–27]. No entanto, a amostragem de pacotes introduz distorções e ruídos nas observações realizadas [28]. Propostas recentes buscam solucionar esse problema através de marcações de pacotes potencialmente anômalos para posterior análise em outros equipamentos da rede [29] ou através de filtros mais eficientes do que a amostragem aleatória [30]. O sistema ADAGA não executa nenhuma estratégia de amostragem de pacotes, não sofrendo influência dessas perturbações. Para evitar o alto processamento e armazenamento exigido para o processamento pacote a pacote, o sistema ADAGA utiliza as estatísticas obtidas através de contadores e sensores presentes nos sistemas monitorados. O processamento do pacote propriamente dito permite realizar análise com base em seu

I.4 Organização do Projeto

conteúdo, o que não será possível no ADAGA, pois o sistema proposto está interessado em anomalias de rede e não das aplicações que utilizam a rede.

Seguindo o ciclo de controle do gerenciamento autônomo descrita por Dobson *et. al.* [5] e apresentada na Figura I.1, ou seja, coleta, análise, decisão e atuação, após a detecção de anomalia requer-se a descoberta da causa geradora da anomalia. A causa das anomalias é obtida nas observações mais recentes da rede [24, 31]. O ADAGA não aborda a descoberta da causa das anomalias, mas envia as últimas observações com todos os dados coletados para serem processadas em outros mecanismos de descoberta da causa das anomalias.

I.4 Organização do Projeto

Este projeto está organizado da seguinte forma. O Capítulo II descreve o sistema desenvolvido e seus módulos enquanto o Capítulo III apresenta os detalhes da implementação do sistema. O Capítulo IV apresenta o cenário de testes com o protótipo desenvolvido e os resultados obtidos. Por fim, o Capítulo V conclui este trabalho e apresenta direções para trabalhos futuros.

Capítulo II

Sistema ADAGA

O ADAGA, sistema de Detecção de Anomalias para o Gerenciamento Autônomo de redes virtuais, provê mecanismos de coleta e análise dos dados em um ambiente de redes virtuais, dando suporte ao gerenciamento autônomo de redes virtualizadas. O objetivo do sistema é permitir a detecção de anomalias nas redes virtuais para acionar mecanismos de correção de anomalias. O ADAGA é um sistema de detecção de anomalias baseado em séries temporais e na geração de alarmes. O processamento dos alarmes indica a existência de anomalias e, por consequência, define os momentos em que a rede necessita de intervenção. A Figura II.1 apresenta a arquitetura do ADAGA, bem como a interligação dos seus módulos e a comunicação entre os sistemas monitorados e o gerente de monitoramento. Cada uma das seções deste capítulo, apresenta explicações para os blocos apresentados na Figura II.1, exceto o módulo de visualização, que é responsável por apresentar gráficos como os mostrados no Capítulo IV.

A Seção II.1 apresenta a estrutura de distribuição do sistema monitorado e do gerente de monitoração. Uma ontologia para representar e estruturar o conhecimento da rede é definida na Seção II.2, que apresenta tanto a estrutura de dados utilizada para representar os elementos da rede quanto os mecanismos utilizados para coletar os dados. O administrador da rede pode selecionar as características relevantes para o monitoramento que deseja realizar dentre todas as monitoradas e representadas na ontologia. Cada um

II.1 Arquitetura

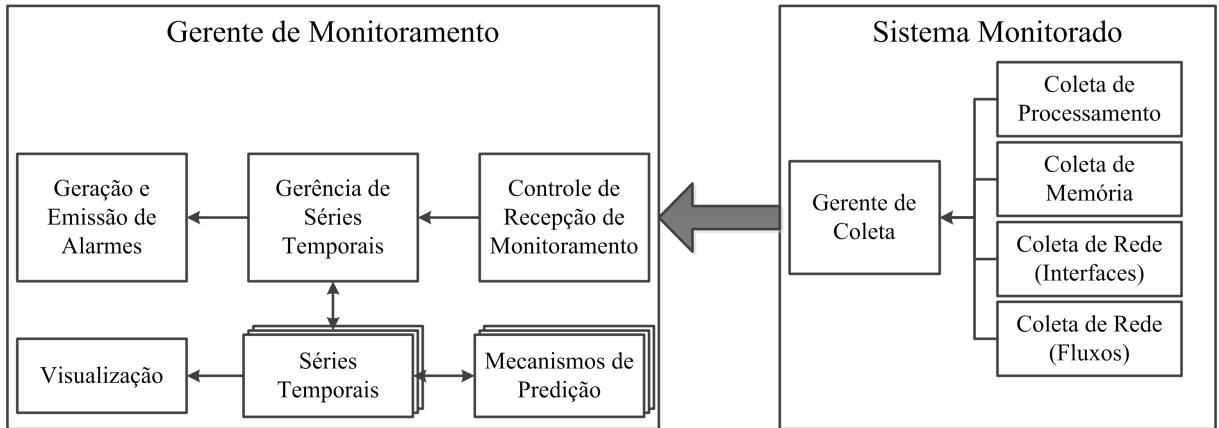


Figura II.1: Arquitetura do ADAGA.

dos valores observados das características definidas para monitoramento são estruturados em séries temporais, como definido na Seção II.3. Cada uma das séries temporais definidas possui um mecanismo de predição associado, que são descritos na Seção II.4 e são responsáveis por quantificar o desvio da medida atual ao passado daquela característica, gerando os alarmes descritos na Seção II.5. O módulo de visualização é responsável por gerar os gráficos de evolução da série temporal e os gráficos de erro no preditor, como os gráficos apresentados nos resultados na Seção IV.1.

II.1 Arquitetura

O sistema ADAGA é concebido para um ambiente distribuído de monitoramento. Dessa forma, o sistema monitorado e o gerente de monitoramento podem ser executados em dispositivos distintos. A comunicação entre ambas as entidades é realizada pela rede, através do protocolo de camada de transporte TCP. Tal formato de implementação permite, sem restringir, o uso de diversas arquiteturas de monitoramento. Assim, não existe um mapeamento pré-definido entre os sistemas monitorados e os gerentes de monitoramento. Tanto pode haver diversos sistemas monitorados pelo mesmo gerente de monitoramento (ver Figura II.2(b)) quanto diversos gerentes de monitoramento observam um mesmo sistema monitorado (ver Figura II.2(c)).

II.1 Arquitetura

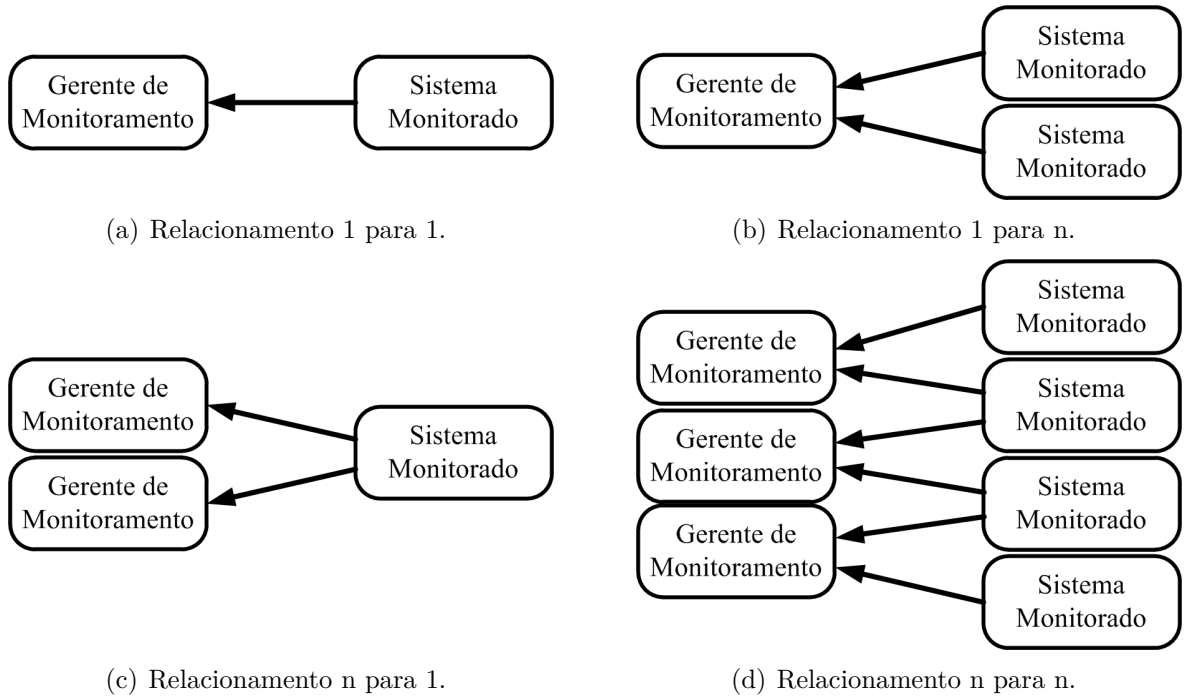


Figura II.2: Possibilidades de relacionamento entre gerentes de monitoramento de sistemas monitorados no sistema ADAGA.

O sistema ADAGA permite um ecossistema de monitoramento na Internet do Futuro através da distribuição de gerentes de monitoramento nas redes física e virtuais. Nesse contexto, a distribuição dos gerentes de monitoramento na rede se apresenta como um problema de otimização para a rede. Liotta *et. al.* [32] propuseram uma solução baseada em agentes móveis para a alocação dos gerentes de monitoramento. Nessa proposta, apenas um agente capaz de gerenciar o monitoramento de toda a rede é lançado pelo administrador da rede em um nó próximo ao administrador, pois este agente pode se duplicar e migrar caso perceba que não está numa posição ótima para realizar o monitoramento. Dessa forma, uma função de otimização, que considera as métricas de roteamento da rede e o consumo de banda da rede usado pelo monitoramento, é calculada pelo agente para verificar se sua posição é ótima para o monitoramento dos sistemas monitorados sobre sua responsabilidade. Se o agente julgar que não é ótimo, ele realiza os procedimentos de duplicação e migração para que o monitoramento seja ótimo. No sistema de Liotta *et. al.* [32], a posição é ótima é obtida com base nos cálculos do protocolo de roteamento em execução na rede. Assim, a função de custo de um monitoramento se baseia no custo de

II.2 Coleta e Representação

se gerenciar o monitoramento a partir do nó escolhido somado ao custos de roteamento de cada nó monitorado. Uma solução de distribuição semelhante pode ser realizada no sistema ADAGA, pois o monitoramento é realizado remotamente.

Outras propostas para monitoramento foram desenvolvidas, sobretudo para o monitoramento de redes *Ad Hoc*, e, devido à natureza distribuída do sistema ADAGA, podem ser aplicadas ao sistema proposto neste projeto. Fernandes *et. al.* [33] propõe que cada elemento monitore todos os seus vizinhos próximos e, quando uma anomalia é detectada, os relatórios de alarmes são enviados para um conjunto de controladores definido para cada nó da rede. Na proposta de Fernandes *et. al.*, todos os nós da rede acumulam a função de gerente de monitoramento, sistema monitorado e de controlador de nós. Esta última função é responsável por executar ações quando uma anomalia é detectada. Propostas como essa, focadas no monitoramento do comportamento malicioso em redes *Ad Hoc* podem ser aplicadas para a arquitetura de monitoramento do sistema ADAGA.

II.2 Coleta e Representação

O ADAGA realiza a coleta de dados dos equipamentos da rede através de requisições remotas de dados no formato *eXtensible Markup Language* (XML). O gerente de monitoramento requisita os dados para os sistemas monitorados, que podem ser tanto os elementos físicos quanto os elementos virtuais da rede. Cada um dos sistemas monitorados mantém em execução o agente de monitoração que, ao receber as requisições, aciona diversos agentes para obter dados sobre o uso de processamento, de memória e de rede do sistema monitorado, conforme apresentado na Figura II.1. Dessa forma, o ADAGA realiza monitoramento passivo, uma vez que não envia sondas de monitoramento na rede [19].

A definição de métricas para observação é fundamental no cenário de monitoramento de redes [19]. Nosso protótipo coleta as observações através de ferramentas disponíveis no sistema operacional Linux, utilizado como base para o sistema, e na plataforma de virtualização Xen [18], escolhida para implementação da rede virtualizada, por ser a plataforma

II.2 Coleta e Representação

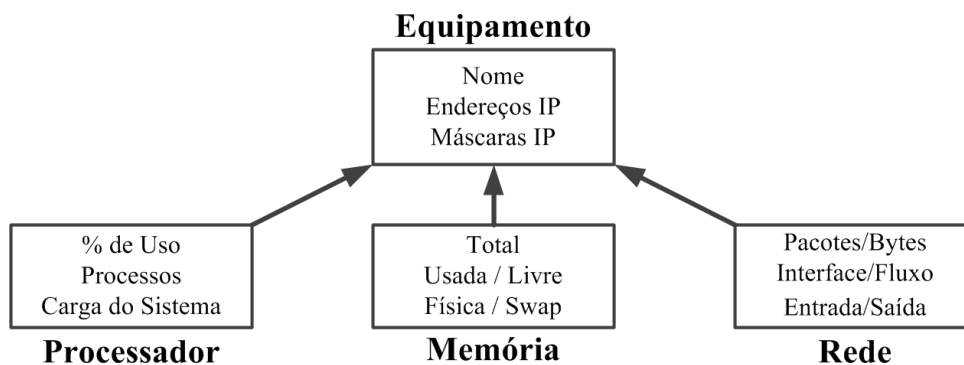


Figura II.3: Estrutura de representação dos elementos da rede no sistema ADAGA.

de virtualização de hardware com código aberto muito utilizada no meio acadêmico. A Figura II.3 apresenta a estrutura simplificada de representação dos elementos de rede no ADAGA.

A modelagem considera processadores de múltiplos núcleos de processamento, diversos dispositivos de memória RAM, como memória física e memória virtual, e diversas interfaces de rede presentes no sistema monitorado. O modelo de dados do equipamento de rede inclui dados de identificação e localização do equipamento e conexões com diversos modelos de processadores, memória e rede. Cada uma das características representadas no modelo participa da detecção de anomalia sendo processada como uma série temporal distinta. No sistema ADAGA, os cálculos descritos na Seção II.4 são aplicados para cada uma das características do modelo independentemente.

No ADAGA, assume-se que um sistema de detecção de anomalias eficiente deve ser capaz de detectar uma anomalia apenas analisando contadores e estatísticas do sistema e da rede, sem que seja necessário analisar todos os pacotes trafegados na rede. Essa propriedade permite ao sistema proposto maior eficiência em processamento e armazenamento dos dados coletados, além de evitar problemas com a privacidade dos dados trafegados na rede, pois não processa nem armazena os dados da rede. De fato, sem a análise do conteúdo da camada aplicação dos pacotes, não é possível a detecção de anomalias em aplicações, o que não é escopo do sistema proposto.

II.3 Séries Temporais

Neste trabalho, séries temporais são usadas como estrutura para o cálculo das previsões e, por conseguinte, a detecção das anomalias. Segundo Brockwell e Davis [34], as séries temporais são um conjunto de observações s_t , cada uma delas realizada em um momento específico $t \in T$, onde T é o conjunto definido dos tempos de medição. A diferença entre as séries temporais e um conjunto comum de valores é que, nas séries temporais, a ordenação das observações é importante. Existem dois tipos de séries temporais classificadas com base na obtenção das observações: as séries temporais discretas e as séries temporais contínuas. Nas séries temporais discretas, as observações são pontuais e realizadas em momentos específicos do tempo. Nas séries temporais contínuas, a observação é realizada continuamente durante um intervalo de tempo. Neste trabalho, utilizamos as séries temporais discretas, pois os dados são coletados em instantes definidos de tempo. Uma série temporal contínua é mais adequada para o processamento analógico. Em sistemas digitais, a discretização é uma exigência e, portanto, o sistema ADAGA utiliza séries temporais discretas.

Nas medições realizadas neste trabalho, define-se o conjunto T pela sequência de tempos de observação com o intervalo entre as observações seguindo a distribuição de Poisson com média em 15 segundos. Segundo Paxson [35], a utilização de intervalos fixos de medidas pode não observar um comportamento periódico na rede ou ainda estar sincronizado com algum evento imprevisível, acarretando redução do desempenho medido ou ofuscação de alguma anomalia.

Em séries temporais, em um dado tempo t , é possível prever o próximo valor s_{t+1} da série baseado no histórico da série (s_1, s_2, \dots, s_t) . Com base nessa previsão e no valor realmente observado no tempo $t + 1$, pode-se definir o erro de predição por

$$\epsilon_t = |\hat{s}_{t+1} - s_{t+1}|, \quad (\text{II.1})$$

onde \hat{s}_{t+1} é o valor predito e s_{t+1} é o valor real do instante $t + 1$, para o qual a predição está sendo realizada. Caso este erro seja maior que a tolerância definida para a previsão,

II.4 Mecanismos de Predição

pode-se gerar um alarme, conforme descrito na Seção II.5.

No sistema ADAGA, o módulo de Gerência de Séries Temporais é responsável por alimentar e controlar as séries temporais na medida em que as observações são recebidas. Esse módulo define as características monitoradas de cada máquina, os parâmetros dos preditores de cada característica e a inserção de novos valores na série temporal das características. Para que o cálculo dos preditores não apresente aumento de custo computacional com o decorrer do tempo e as novas observações coletadas, uma janela de observação é definida para cada série temporal. A janela de observação consiste no subconjunto de s_t das w últimas observações que são consideradas para o cálculo nos mecanismos de predição. O módulo de gerência de séries temporais é responsável por garantir que a janela de observação não ultrapasse o seu tamanho máximo.

II.4 Mecanismos de Predição

A previsão do próximo valor de cada série temporal é calculada através de mecanismos de predição. Neste trabalho, foram utilizados dois mecanismos de predição. O mais simples, que não considera eventos periódicos nas séries, é denominado *Exponential Smoothing*. O outro mecanismo implementado e analisado nesse trabalho é o *Holt-Winters Seasonal*, que considera as componentes sazonais e de tendência das séries temporais. Nesse trabalho, foram escolhidos esses dois mecanismos de predição por serem de simples implementação e de comprovado sucesso em outros trabalhos [6, 23] para a detecção de anomalias em redes de computadores.

No sistema ADAGA, o módulo gerenciador de séries temporais permite que cada característica monitorada de cada sistema monitorado possa ser predita com um mecanismo preditor diferente, sem que uma característica influencie na predição das demais. Além disso, o sistema permite a troca do mecanismo de predição enquanto o monitoramento é feito sem que o sistema precise ser reiniciado. Mesmo mecanismos de predição não implementados ou integrados ao sistema no momento da inicialização, podem ser carregados

II.4 Mecanismos de Predição

dinamicamente.

II.4.1 *Exponential Smoothing*

O mecanismo preditor *Exponential Smoothing*, ou Aproximação Exponencial, é um algoritmo simples para cálculo de próximo valor em uma série temporal, pois é baseado na média móvel do histórico da série [6]. Considere s_1, s_2, \dots, s_t como a série temporal para a característica monitorada em questão e \hat{s}_{t+1} a previsão para o próximo valor, denotado por $t + 1$, desta série. Para o cálculo de cada próximo valor \hat{s}_{t+1} , o valor atual medido s_t e a previsão realizada para o valor atual \hat{s}_t são expressos por

$$\hat{s}_{t+1} = \alpha s_t + (1 - \alpha) \hat{s}_t, \quad (\text{II.2})$$

onde $0 < \alpha < 1$ e $\hat{s}_1 = s_1$.

O parâmetro α é a ponderação do valor atual medido em relação ao histórico da série. Assim, quanto maior o valor de α menor é a influência do passado da série no cálculo do valor previsto. Segundo a análise de Lucena e Moura [23], no cenário de redes, valores apropriados para o parâmetro α devem ser inferiores a 0,1, mas não muito próximos de zero. Caso o valor de α seja superior a 0,1, o estimador se adequa à anomalia, inviabilizando a sua sinalização. Por outro lado, se o valor de α for muito próximo de zero, após uma anomalia, a captura do padrão de normalidade da característica analisada demora mais. As avaliações deste trabalho são feitas com valores de α iguais a 0,05, 0,10 e 0,15, onde este último foi escolhido para validar os resultados obtidos por Lucena e Moura[23].

Devido aos cálculos sucessivos da média móvel, cada uma das observações passadas influencia na predição do próximo valor. Assim, o desdobramento da Equação II.2 resulta em [34]

$$\hat{s}_t = \left[\sum_{j=0}^{t-2} \alpha(1 - \alpha)^j s_{t-j} \right] + (1 - \alpha)^{t-1} s_1. \quad (\text{II.3})$$

Assim, cada parcela do cálculo representa a influência de cada observação passada e essa

II.4 Mecanismos de Predição

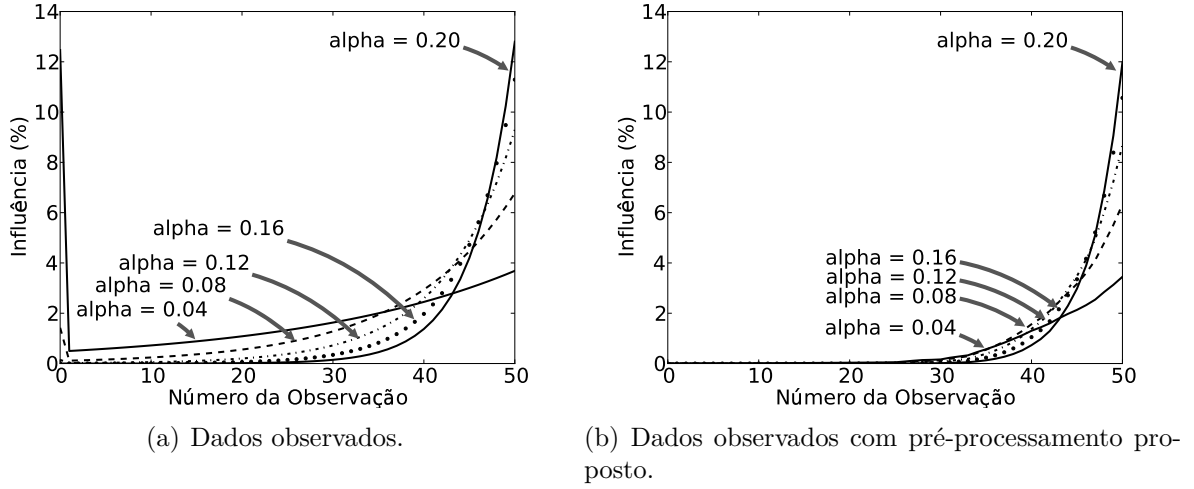


Figura II.4: Influência das observações passadas no cálculo da predição da amostra $t = 50$ para a característica pacotes transmitidos por todas as interfaces.

influência é reduzida exponencialmente ao longo do tempo a partir do instante $t = 2$, como mostra a Figura II.4. Portanto, a condição inicial definida possui grande influência nos resultados, conforme se observa na Figura II.4(a). Para inibir o impacto da condição inicial na previsão dos valores, este trabalho propõe a aplicação de um pré-processamento nos valores da série, que é definido por

$$s_t = s_t - s_1, \forall s_t, \quad (\text{II.4})$$

onde o novo valor de s_t é definido pela subtração do primeiro valor da série pelo valor coletado no instante t . Dessa forma, neste trabalho, a configuração inicial é $\hat{s}_1 = 0$ e não há erro na predição inicial, como apresentado na Figura II.4(b).

II.4.2 *Holt–Winters Seasonal*

O mecanismo *Holt–Winters Seasonal* objetiva a previsão de séries temporais que apresentem sazonalidades. As sazonalidades são comportamentos periódicos na série temporal. Brutlag [6] definiu um modelo de sazonalidade na utilização das redes de computadores, o qual consiste em um aumento do uso ao longo da manhã e uma menor utilização à

II.4 Mecanismos de Predição

noite. Em cenários com sazonalidades, o mecanismo *Exponential Smoothing* não é adequado, pois ele considera a linearidade dos valores da série, uma vez que aproxima o valor seguinte da série utilizando a média móvel do histórico da série.

O *Holt-Winters Seasonal* decompõe a série temporal em tendência, sazonalidade e ruído. Cada uma dessas componentes é tratada como uma variação do método *Exponential Smoothing*. Existem duas formas de agregar essas componentes em um valor de previsão: a aditiva e a multiplicativa [36]. Agregam-se essas componentes de forma aditiva quando a variação estatística do período não depende da tendência da série. Quando depende, as componentes são multiplicadas. O trabalho de Lucena e Moura [23] defende que, para redes de computadores, a composição aditiva das componentes apresenta melhores resultados. Dessa forma,

$$\hat{s}_{t+1} = R_t + T_t + P_{t+1-m} \quad (\text{II.5})$$

representa o resultado da previsão do próximo valor por esse método e é a definição utilizada neste trabalho. O termo T_t representa a tendência da série temporal, P_{t+1-m} a componente periódica, no qual m é o período da sazonalidade, e R_T é o ruído agregado à série. Assim, as equações de previsão de cada um desses valores são dadas por

$$R_t = \alpha (s_t - P_{t-m}) + (1 - \alpha) (R_{t-1} + T_{t-1}) \quad (\text{II.6})$$

$$T_t = \beta (R_t - R_{t-1}) + (1 - \beta) T_{t-1} \quad (\text{II.7})$$

$$P_t = \gamma (s_t - R_t) + (1 - \gamma) P_{t-m}. \quad (\text{II.8})$$

Os parâmetros α , β e γ , tal que α, β e $\gamma \in \mathbb{R}$ e α, β e $\gamma \in [0, 1]$, representam as constantes de esquecimento para cada uma das parcelas do valor predito. Do mesmo modo que no *Exponential Smoothing*, esses parâmetros representam a importância do passado da série no cálculo do valor predito. Quanto maior o valor das constantes, menor será a influência do passado da componente na previsão.

II.5 Geração de Alarmes

A geração dos alarmes no sistema ADAGA ocorre quando o erro da predição é superior ao valor calculado como margem de aceitação. A margem de aceitação é recalculada para cada nova observação e é definida por

$$\epsilon_t = \delta \Psi_t, \quad (\text{II.9})$$

onde δ é uma constante de ampliação da margem de aceitação do erro e Ψ_t é dependente do mecanismo de predição. Brutlag [6] defende que valores ótimos para δ pertencem ao intervalo de números reais tal que $2 < \delta < 3$. O sistema ADAGA usa $\delta = 2$, pois objetiva ser um sistema mais sensível a anomalias e $\delta = 2$ gera uma faixa de aceitação menor.

Para o *Exponential Smoothing*, Ψ_t é definido pelo desvio padrão dos valores considerados na janela de observação em relação ao próximo valor predito. Já no mecanismo *Holt-Winters Seasonal*, o valor de Ψ_t é definido por

$$\Psi_t^{HOLT} = \gamma(|s_t - \hat{s}_t|) + (1 - \gamma)(\Psi_{t-m}^{HOLT}). \quad (\text{II.10})$$

No ADAGA, propõe-se o controle na emissão de relatórios de alarmes através da acumulação de alarmes. Dessa forma, nem todos os alarmes gerados são emitidos. O objetivo dessa técnica é a eliminação de alarmes pontuais no sistema. A implementação desse método no ADAGA utiliza uma histerese para definir a emissão dos alarmes gerados. Caso o sistema detecte uma anomalia nas observações, esta não é emitida imediatamente. Somente após o acúmulo de η alarmes gerados, que os alarmes começam a ser emitidos. O acúmulo de alarmes antes de gerar o relatório representa um aumento no tempo de reação às anomalias enquanto pode reduzir falsos positivos. A correta escolha desse parâmetro representa um compromisso entre o tempo de reação e a eficácia do preditor. O valor de η é definido pela administração da rede e sua variação é analisado nos resultados referentes a eficácia do mecanismo preditor.

II.5 Geração de Alarmes

A emissão de um alarme significa o envio de um relatório para o sistema de decisão e atuação na rede. Esse relatório consiste dos valores de todas as características analisadas na rede nas últimas cinco observações e mais os dados da análise da observação da característica que gerou o alarme, como o valor predito, o valor real e a faixa de confiança, identificando a característica.

Capítulo III

Detalhamento da Implementação

Um protótipo do sistema ADAGA proposto foi construído como prova de conceito e avaliação do sistema. Neste capítulo, apresentam-se os detalhes técnicos envolvidos na implementação do protótipo. Na Seção III.1 será apresentada a base tecnológica utilizada neste protótipo, tais como as bibliotecas e ferramentas. Na Seção III.2, a implementação da comunicação entre o sistema monitorado e o gerente de monitoração é descrita. A Seção III.3 discute os detalhes de cada um dos agentes de coleta utilizados e a Seção III.4 apresenta o modelo de dados utilizado no armazenamento dos dados obtidos para posterior utilização. A Seção III.5 aborda a construção do mecanismo de controle das séries temporais, bem como a comunicação *interthread* implementada no protótipo. O mecanismo que obtém as observações do sistema monitorado e o gerenciador das séries temporais estão em *threads* separadas para que a coleta das observações ocorra em intervalos determinados pela distribuição de Poisson utilizada sem influência do tempo gasto pelas funções do gerenciador das séries temporais. O gerenciador de séries temporais é responsável pela inserção dos valores observados nas séries temporais correspondentes e pelo cálculo das predições, que é realizado pelos mecanismos de predição implementados, tal como é descrito na Seção III.6. A Seção III.7 discute a implementação do mecanismo de geração de alarmes e do sistema de emissão de relatórios implementado no sistema ADAGA.

III.1 Tecnologias Utilizadas

Python - Python é uma linguagem de programação que permite o desenvolvimento ágil e a integração mais efetiva entre os sistemas¹. A linguagem Python foi utilizada para a construção de todo o protótipo desenvolvido neste projeto.

Biblioteca Qt - Qt é um arcabouço multiplataforma para desenvolvimento de aplicações com interface gráfica. Além das funções para interfaces gráficas do usuário (*Graphical User Interface* - GUI), a biblioteca oferece suporte a interação com diversos bancos de dados, processamento de XML, gerenciamento de *threads*, suporte a operações de rede e tratamento de arquivos em diversas plataformas de forma uniformizada. Neste projeto, foi utilizada para o gerenciamento e a comunicação das *threads* utilizadas na recepção das observações e no gerenciamento das séries temporais.

Biblioteca Os - Esta biblioteca da linguagem Python permite realizar operações dependentes do sistema operacional de uma maneira uniformizada. Assim, o código desenvolvido se mantém multiplataforma mesmo utilizando funções específicas do sistema operacional, como a obtenção de estatísticas de um *path* da estrutura de diretórios ou a execução de um processo filho da aplicação corrente. No projeto do protótipo do sistema ADAGA, este módulo foi utilizado para executar as ferramentas de coleta de dados no sistema monitorado, como as ferramentas Top e Ifconfig.

Biblioteca NumPy - NumPy é uma biblioteca fundamental para computação científica em Python. Contém ferramentas como estruturas de dados manipulação de listas multidimensionais e algoritmos úteis para álgebra linear, transformadas de Fourier e cálculo de números aleatórios². No projeto, a biblioteca NumPy foi utilizada para a obtenção de números aleatórios seguindo a distribuição de Poisson para calcular o intervalo de coleta das observações.

¹Adaptado de: <http://www.python.org>

²Adaptado de: <http://numpy.scipy.org/>

III.1 Tecnologias Utilizadas

Biblioteca Collections - A biblioteca Collections da linguagem Python implementa estruturas de dados de alta performance para agrupamento de dados como listas, filas e dicionários. Nesse projeto, foi utilizado a estrutura de dados *deque*, abreviação para *double-ended queue*, para o armazenamento das séries temporais. A grande vantagem desta estrutura de dados é a possibilidade de se adicionar e remover elementos da fila por qualquer uma das extremidades com complexidade $O(1)$. A lista padrão da linguagem Python oferece remoção e inserção de elementos com complexidade $O(n)$.

Biblioteca Re - A biblioteca Re da linguagem Python é responsável por operações com expressões regulares. Uma expressão regular especifica um conjunto de *strings* que correspondem às regras definidas. As funcionalidades desta biblioteca consistem em checar se uma *string* corresponde a uma dada expressão regular ou se uma expressão regular corresponde a uma *string* dada. Além disso, este módulo filtra uma *string* por uma expressão regular para obter apenas o trecho desejado. Neste projeto, as expressões regulares foram utilizadas com o objetivo de filtrar os dados obtidos das ferramentas de coleta de dados apresentadas abaixo e obter os dados observados do sistema monitorado.

Biblioteca Socket - A biblioteca Socket provê acesso a interface de manipulação de sockets do BSD. A interface Python que o módulo oferece é uma transliteração das chamadas nativas do sistema Unix em uma interface no estilo de orientação a objetos, usado no Python. No projeto ADAGA, esta biblioteca foi utilizada para a comunicação entre o sistema monitorado e o gerente de monitoramento, através de conexão TCP com suporte a IP versão 6.

Biblioteca Minidom - A biblioteca Minidom é uma implementação leve da especificação *Document Object Model* (DOM) para análise e processamento de XML. Seu objetivo é ser mais simples e menor que a implementação completa da especificação DOM. DOM é uma API multiplataforma criada pelo *World Wide Web Consortium* (W3C) para acessar e modificar documentos XML. Implementações que seguem a especificação DOM apresenta um documento XML como uma estrutura em árvore.

III.1 Tecnologias Utilizadas

Essa estrutura dá acesso aos dados do documento XML através de um conjunto de objetos com interfaces bem definidas. O uso do DOM é indicado para aplicações que requerem o acesso aleatório aos elementos do documento XML, pois a implementação do DOM consiste no processamento prévio de todo o documento, armazenando-o em memória. Uma vez que se tem o objeto do documento DOM, podem-se acessar as partes do documento XML diretamente através dos atributos e métodos do objeto do documento DOM. Neste projeto, a biblioteca Minidom foi utilizada para o processamento do XML das mensagens recebidas, tanto as requisições recebidas pelo sistema monitorado quando as respostas recebidas pelo gerente de monitoramento.

Ferramenta Top - O programa Top oferece uma visualização em tempo real do sistema executado. Ele apresenta um resumo das informações do sistema como a lista de processos gerenciados pelo sistema operacional³. Neste projeto, esta ferramenta foi utilizada para o sistema de coleta obter os dados de uso de processamento, processos e carga do sistema monitorado.

Ferramenta Ifconfig - O Ifconfig utiliza os dados armazenados no arquivo “*/proc/net/dev*” e atualizados pelo sistema operacional Linux. Informações sobre cada interface de rede tais como o IP, a máscara de rede, a quantidade de pacotes recebidos e transmitidos entre outros são obtidos com a ferramenta. Neste projeto, é utilizada para se obter os dados das interfaces de rede do sistema monitorado.

Ferramenta Iptables - O programa Iptables é uma ferramenta de administração para filtragem de pacotes IP versão 4 e configuração do *Network Address Translation* (NAT) em sistemas Linux. Ele é usado para configurar, manter e inspecionar as tabelas de regras de filtragem de pacotes do Kernel do Linux. Diferentes tabelas podem ser definidas e cada tabela contém cadeias definidas previamente ou pelos administradores. Cada cadeia é uma lista de regras sobre as quais os pacotes são testados. Cada regra especifica o que fazer uma ação para ser realizada quando os pacotes atendem a esta regra. As ações são chamadas de “*target*” ou “alvo”, que

³Adaptado da página de manual da ferramenta Top no Linux.

III.2 Comunicação Sistema/Gerente

pode ser uma outra cadeia na mesma tabela ou comandos para o Iptables, como aceitar ou descartar os pacotes⁴. No sistema ADAGA, o Iptables foi utilizado para obter contadores de pacotes e bytes para fluxos específicos de pacotes. Definiu-se regras para cada fluxo que se deseja monitorar e cada pacote correspondente a alguma regra definida é contado na coleta de dados do sistema ADAGA.

Base de Dados SQLite - SQLite é uma biblioteca de software multiplataforma que implementa um Sistema Gerenciador de Banco de Dados (SGBD) baseado em transações, autocontido, sem necessidade de servidor e com nenhuma configuração necessária. As transações do SQLite são atômicas, consistentes, isoladas e duráveis (ACID), mesmo após quebras de sistemas e falhas de energia. Uma das grandes diferenças do SQLite em relação a outros sistemas gerenciados de bancos de dados é o fato do SQLite armazenar toda a base de dados em um único arquivo, que pode ser armazenado em qualquer diretório⁵. O SQLite foi utilizado no protótipo desenvolvido para a persistência dos dados coletados.

III.2 Comunicação Sistema/Gerente

A comunicação entre o sistema monitorado e o gerente de monitoramento é realizada em rede, para que seja possível a distribuição do gerente de monitoramento na rede e a construção de variadas arquiteturas de monitoramento. Nesta seção, a implementação dos módulos de conexão entre o sistema monitorado (servidor) e o gerente de monitoramento (cliente) é discutida.

O servidor desenvolvido é subdividido em três classes principais: *BindServer*, *BindServerThread* e *SelectApplication*. O relacionamento entre essas classes é apresentado no diagrama da Figura III.1. O objeto da classe *BindServer* constitui na configuração do servidor propriamente dita, pois define a porta TCP que receberá as conexões deste serviço e constitui o laço principal de recepção das requisições remotas. Para evitar que o

⁴Adaptado da página de manual da ferramenta Iptables no Linux

⁵Adaptado de: <http://www.sqlite.org/>

III.2 Comunicação Sistema/Gerente

servidor fique bloqueado quando uma requisição é recebida e está sendo processada, todo o seu processamento é feito em outra *thread*, reduzindo ao máximo o bloqueio do servidor para requisições concorrentes.

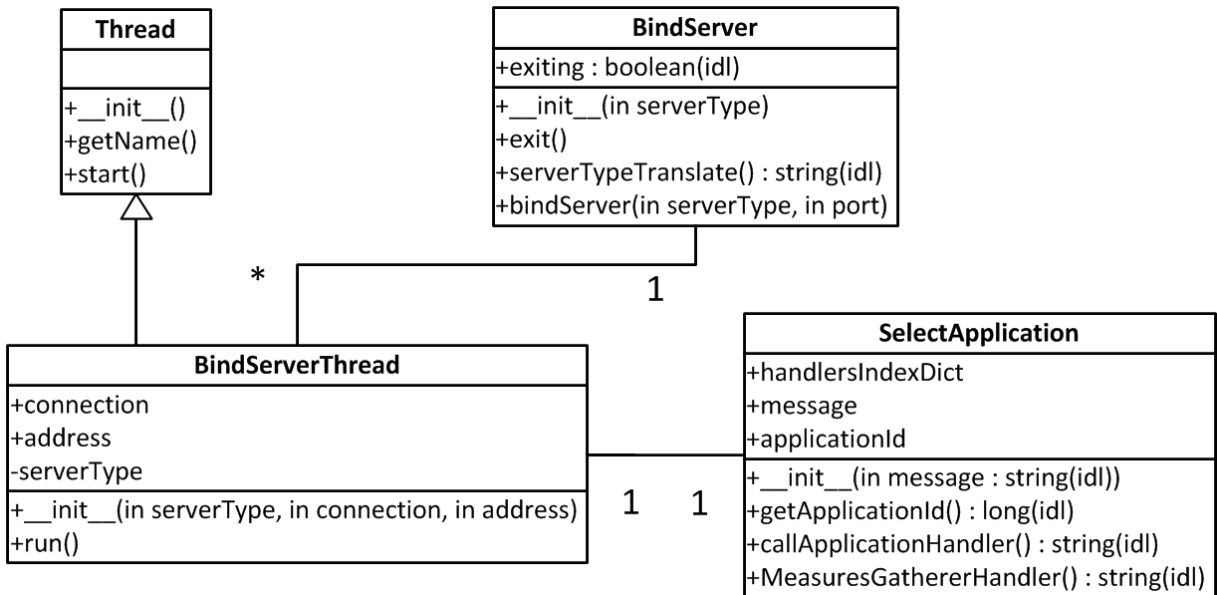


Figura III.1: Diagrama de relacionamento das classes do servidor TCP executado no sistema monitorado.

A classe *BindServerThread* processa a requisição recebida. O objeto desta classe se mantém vivo até que toda a requisição seja processada e a resposta enviada para o cliente. O servidor desenvolvido para o protótipo do sistema ADAGA foi projetado para ser genérico, podendo ser utilizado por outros serviços que necessitem de interação com o sistema monitorado, como possíveis funções de atuação como, por exemplo, migrar ou criar novos roteadores virtuais. Devido a esse requisito, a classe *SelectApplication* foi construída e é responsável por detectar qual aplicação do servidor deve ser carregada de acordo com a mensagem de requisição recebida. Essa classe ainda pode ser especializada para funções específicas de roteadores físicos e virtuais, conforme desejado. É responsabilidade do objeto da classe *BindServerThread* definir qual das classes especializadas da classe *SelectApplication* deve ser instanciada naquele sistema monitorado. Essa decisão é realizada com base no parâmetro *serverType* recebido.

A classe *SelectApplication* base é utilizada neste protótipo, pois a obtenção de medi-

III.2 Comunicação Sistema/Gerente

das é realizada tanto nos roteadores físicos quanto nos roteadores virtuais. Ela possui um dicionário (*handlersIndexDict*) que mapeia o identificador numérico da aplicação no método que deve ser invocado para manipular a mensagem de requisição. Cada desenvolvedor de aplicação deve criar seu próprio método para que sua aplicação seja incorporada ao servidor. O método referente à aplicação de coleta dos dados do sistema monitorado é o *MeasuresGathererHandler*. Para ser possível um servidor único, um formato padrão de mensagem foi definido. Assim, o método *getApplicationId* da classe *SelectApplication* consegue entender a parte inicial da mensagem, extraindo o número de identificação da aplicação e invocando o método correspondente. A estrutura inicial da mensagem de requisição definida pode ser observada no Código Fonte A.1, no Apêndice A. A classe *SelectApplication* extrai a informação do campo *ApplicationId* que possui o número de identificação da aplicação que deve ser executada. O campo *NodePK* identifica o gerente de monitoramento, pois esse campo guarda a chave pública do emissor de todas as mensagens utilizadas no protótipo. No caso da aplicação de monitoramento desenvolvida, o número identificador é 1.

O cliente desenvolvido é constituído pela classe *clientConnect*, que deve ser instanciada para realizar a comunicação. Esta classe recebe a mensagem que deve ser enviada e os dados de conexão com o servidor, como endereço IP e porta TCP. Após a definição desses dados, uma conexão de rede é realizada e a resposta é aguardada, caso necessário. O retorno do método *connectToServer* depende do método *processingAnswer* implementado. No caso da implementação utilizada neste protótipo, o método *processingAnswer* apenas retorna a mensagem sem nenhum processamento. Contudo, seguindo o conceito de implementação de clientes e servidores genéricos, outros clientes podem utilizar esta classe para conectar com o servidor no caso de outras aplicações. Para tal, basta o desenvolvedor da aplicação criar outra classe, que herda da classe *clientConnect*, e sobrescrever o método *processingAnswer* para a aplicação desenvolvida.

O Código Fonte A.2, no Apêndice A, apresenta o servidor executado no sistema monitorado e o Código Fonte A.3, no Apêndice A, o cliente executado pelo gerente de monitoramento quando este deseja coletar uma observação do sistema monitorado.

III.3 Coleta de Dados

A coleta de dados do sistema ADAGA consiste em um sistema multiagente, implementado em Python. O agente gerente de coleta é responsável por receber as requisições de coleta e ativar cada um dos agentes especialistas para observar as diversas características do elemento de rede monitorado. Cada agente especialista coleta dados de uma ferramenta específica do Linux ou do Xen, como a ferramenta Top e a ferramenta Ifconfig. No sistema implementado, existem quatro agentes especialistas: Coleta de Processamento, Coleta de Memória, Coleta de Rede por Interface e Coleta de Rede por Fluxos. Todos os agentes de monitoramento podem ser utilizados nos sistemas virtualizados ou no substrato físico da rede. A coleta de dados no ADAGA é realizada de maneira intrusiva. Portanto, os agentes de coleta executam dentro da máquina virtual, que precisa conhecer e implementar o protocolo TCP/IP, para transmitir os dados coletados para o gerente de monitoramento.

O agente coletor de processamento utiliza a ferramenta *Top* do Linux. Ele coleta dados como o uso de processamento por usuários, pelo sistema e por tarefas de entrada e saída separadamente, oferecendo percepção sobre o estado atual do processamento do elemento de rede monitorado. Além dos dados sobre o processamento, o agente de coleta também obtém informações sobre a carga média do sistema⁶ em três granularidades: média do último minuto, dos últimos cinco minutos ou dos últimos quinze minutos. Assim, o sistema pode monitorar séries temporais distintas que representam as médias em intervalos curtos e longos da carga do sistema, possibilitando detectar tendências na carga do sistema e prever comportamentos futuros com base no cruzamento das médias curta e longa processadas. Por último, o coletor de processamento observa a quantidade de processos gerenciados pelo sistema operacional do elemento de rede monitorado. A quantidade de processos em execução, dormindo e parados é monitorada, pois seu monitoramento

⁶A carga média do sistema é a média do número de processos que estão em estado de execução normal ou ininterrupta. Um processo em estado de execução normal está tanto utilizando CPU ou esperando o escalonador permitir o uso da CPU. Um processo em estado de execução ininterrupta espera por dispositivos de entrada e saída, como o acesso à rede. O valor obtido não é normalizado pelo número de núcleos de processamento da máquina. Adaptado da página de manual da ferramenta no Linux.

III.3 Coleta de Dados

é importante para a detecção de anomalias do tipo *fork bomb*, ou seja, anomalias que dupliquem indefinidamente os processos.

O agente responsável pela coleta de informações da memória⁷ do sistema monitora o arquivo “*/proc/meminfo*” do Linux, alimentado com informações de uso de memória pelo sistema operacional. Nele, é possível encontrar a quantidade total de memória do sistema, a quantidade usada, memória virtual, páginas de memória, memória em cache dentre outras.

O agente responsável pela coleta de informações de rede nas interfaces⁸ permite obter informações do tráfego de rede de cada interface de rede, física ou virtual, do sistema através da ferramenta *ifconfig* do Linux. Esta ferramenta provê contadores de bytes recebidos e transmitidos e pacotes recebidos, transmitidos, descartados e com erro, dentre outras. Este agente de coleta agrega os dados de rede por interface de rede e, portanto, não permite controlar um dado fluxo específico, como por exemplo, o tráfego de pacotes recebidos do protocolo Secure Shell (SSH). Para permitir essa granularidade, foi implementado no sistema ADAGA outro agente coletor de informações de rede, mas que utiliza o conceito de fluxo. No sistema ADAGA, a sequência unidirecional de pacotes entre dois pontos da rede que seguem por um mesmo caminho é considerado um fluxo de rede. Para um dado elemento desse caminho, pode-se definir um fluxo como sendo a sequência de pacotes que possuem os mesmos endereços de origem e destino nas camadas de enlace (endereço MAC), de rede (endereço IP), e de transporte (portas do TCP ou do UDP).

Essa definição de fluxo adiciona o endereço MAC, pois a multiplexação dos pacotes para as máquinas virtuais é realizada através do endereço MAC. No sistema ADAGA, o agente de Coleta de Rede por Fluxos utiliza a ferramenta *iptables* do Linux para contar os bytes e pacotes de cada fluxo. Para tal, são definidas regras do *iptables* para cada fluxo a ser monitorado. Como esta ferramenta pode agir como um *firewall*, todas as regras são associadas a ações. Assim, o sistema ADAGA cria uma cadeia de regras própria para permitir o isolamento com as demais regras definidas pelo administrador do elemento de

⁷Agradeço ao aluno de mestrado Carlo Fragni que desenvolveu este agente de coleta e o forneceu para uso neste projeto final.

⁸Agradeço ao aluno de mestrado Carlo Fragni que desenvolveu este agente de coleta e o forneceu para uso neste projeto final.

III.4 Representação dos Dados

rede. As regras do agente de Coleta de Rede por Fluxos executam a ação RETURN, que representa apenas retornar para a cadeia de regras padrão, sem aceitar ou descartar os pacotes, mas realizando as estatísticas que serão monitoradas e transmitidas pelo gerente de coleta.

Todos os agentes coletores implementados são gerenciados pelo gerente de coleta, conforme apresentado na Figura II.1. Este agente aguarda requisições do gerente de monitoramento, que envia mensagens em formato XML informando quais grupos de informação deseja receber. Quando uma dada requisição é recebida, o gerente de coleta processa-a e decide os agentes de coleta que devem ser ativados para atender à requisição recebida. Após a coleta de dados ser realizada por todos os agentes de coleta, o gerente monta a mensagem XML de resposta com as informações desejadas. Esse processo de coleta de todas as informações é definido no sistema ADAGA como observação do elemento de rede monitorado. A observação constitui um elemento das séries temporais e consecutivas observações são realizadas para construir as séries temporais, responsáveis pelo monitoramento. No sistema ADAGA, as observações não são espaçadas entre si por um período constante, mas sim por valores que seguem uma distribuição de Poisson [35] com centro em quinze segundos, pois a coleta periódica dos dados pode ocultar a observação de algum evento periódico ou de eventos que se sincronizem com a observação. A distribuição de Poisson é obtida através da biblioteca *NumPy* para Python.

III.4 Representação dos Dados

Os dados coletados no sistema monitorado são representados dinamicamente através da ontologia definida na Figura II.3. No entanto, os objetos que representam as observações precisam ser persistidos para que as observações passadas possam ser utilizadas em análises futuras. No sistema ADAGA, optou-se pelo uso de banco de dados para a persistência dos dados coletados na rede. Para permitir a simplicidade de operação e redução da complexidade do ambiente de monitoramento, foi adotado o SQLite. Como a biblioteca de acesso ao Sistema Gerenciador de Banco de Dados (SGBD) foi a biblioteca

III.4 Representação dos Dados

Qt, que permite acesso a diversos SGBDs, a alteração do sistema utilizado é trivial e não requer nenhuma modificação no código do protótipo.

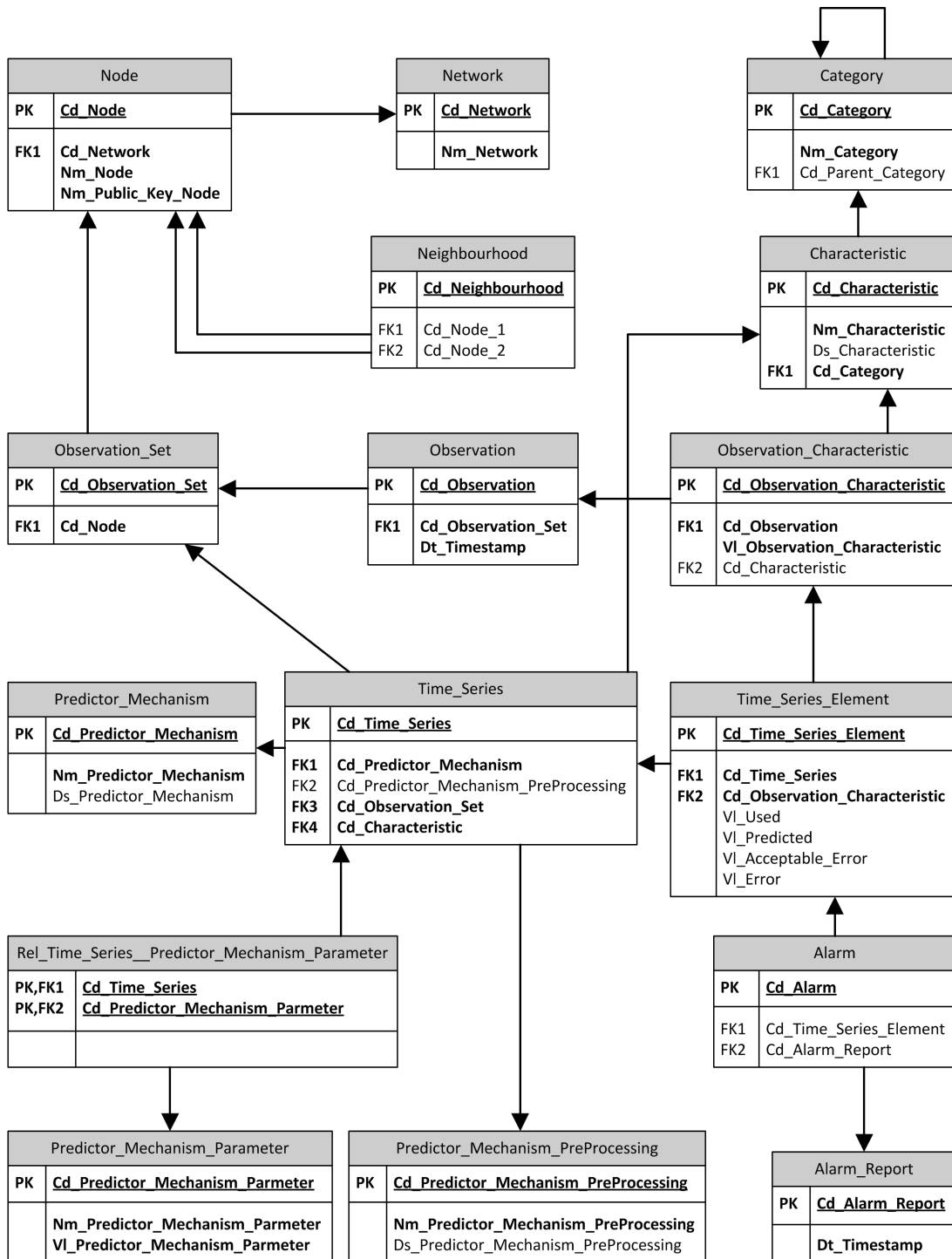


Figura III.2: Modelo de dados do banco de dados usado para armazenar as observações e cálculos realizados no ADAGA.

III.4 Representação dos Dados

O diagrama de entidade-relacionamento do modelo de dados utilizado para no sistema ADAGA é apresentado na Figura III.2. O modelo foi projetado para permitir a personalização do monitoramento pelo administrador da rede. Portanto, as características monitoradas na rede não são definidas a priori no modelo. A entidade *Characteristic* é responsável por armazenar as características que já foram monitoradas pelo sistema, juntamente com a sua descrição e o indicativo da categoria a qual pertencem. As categorias são agrupadas na entidade *Category*, que, para permitir uma árvore de categorias e favorecer a estrutura apresentada na ontologia da Figura II.3, possui um autorelacionamento, através da coluna *Cd_Parent_Category*, que referencia a chave primária da categoria pai da categoria em questão. Apenas uma categoria não possui uma categoria pai, pois é a raiz da árvore de categorias. No caso do projeto ADAGA, essa categoria é denominada *Equipamento* e é inserida no modelo logo na inicialização do sistema, caso não haja.

Os nós e a topologia da rede são armazenados nas entidades *Node* e *Neighbourhood*. Como as redes virtuais são isoladas, cada nó pertence a apenas uma rede, referenciando-a através da coluna *Cd_Network* da entidade *Node*. Além disso, a entidade *Node* também armazena o nome do nó e sua chave pública, presente no cabeçalho de todas as mensagens recebidas daquele sistema monitorado. A vizinhança do nó é tida como todos os seus vizinhos. Assim, uma entrada da entidade *Neighbourhood* representa que os dois nós referenciados nas colunas *Cd_Node_1* e *Cd_Node_2* são vizinhos.

Expostas as entidades de identificação dos nós e das características monitoradas, as demais entidades compõem três grupos principais. O primeiro grupo representa o que foi realmente observado na rede e possui as entidades *Observation_Set*, *Observation* e *Observation_Characteristic*. O segundo grupo representa o cálculo das séries temporais com seus mecanismos de predição, envolvendo as entidades *Time_Series*, *Time_Series_Element*, *Predictor_Mechanism*, *Predictor_Mechanism_Parameter*, *Predictor_Mechanism_PreProcessing* e *Rel_Time_Series_Predictor_Mechanism_Parameter*, uma entidade que representa o relacionamento n para n entre as entidades *Time_Series* e *Predictor_Mechanism_Parameter*. E, por último, o grupo responsável por armazenar os alarmes gerados na rede, representados pelas entidades *Alarm* e *Alarm_Report*.

III.5 Gerência de Séries Temporais

O módulo de gerência das séries temporais é responsável por receber os dados de monitoramento, obter os dados monitorados apenas das características que estão sendo monitoradas e adicionar os novos valores nas séries temporais corretas. Além disso, é através dele que o administrador da rede pode interagir com o sistema, definindo as características da rede que serão monitoradas e o mecanismo de predição usado para cada série temporal.

No Código Fonte A.4, no Apêndice A, é apresentado um trecho da classe *TimeSeriesController*, responsável pela gerência das séries temporais. O método *addMonitoring* é chamado quando uma nova observação é recebida. A lista de características monitoradas é consultada e apenas os dados dessas características são recuperados do objeto de monitoramento recebido para serem adicionados nas séries temporais correspondentes. Esse método ainda é responsável por perceber os alarmes em cada série temporal. Caso haja algum alarme, o método *alarmReceived*, que será melhor explicado na Seção III.7, é chamado.

Os dois outros métodos, *monitorVariable* e *forgetVariable*, estão relacionados às características que devem ser monitoradas. O administrador da rede utiliza esses métodos para adicionar ou remover, respectivamente, características da rede a serem monitoradas. No caso da remoção de variáveis monitoradas, basta excluir a variável monitorada para cada um dos sistemas monitorados e retirá-la da lista de variáveis monitoradas. Contudo, a situação oposta, a adição de novas variáveis a serem monitoradas, requer que, além da criação das séries temporais observações recebidas anteriormente no conjunto de observações atual sejam carregadas na ordem em que foram recebidas para preencher a janela de observação. O Objetivo desse carregamento é aproveitar ao máximo os dados que já foram coletados anteriormente para que não seja necessária uma nova fase de aprendizado para a característica que passa a ser monitorada.

III.6 Mecanismos de Predição

Os mecanismos preditivos são os responsáveis pelo cálculo das previsões para as séries temporais e, por consequência, dos erros encontrados pelos preditivos. Diversos mecanismos podem ser utilizados para o mesmo objetivo. Cada um suas especificidades, vantagens e desvantagens. Devido a essa variabilidade possível, esse módulo do sistema ADAGA foi projetado de modo que novos mecanismos preditivos pudessem ser carregados de forma simples e dinâmica, sem que o sistema necessitasse ser reiniciado. Para permitir a conexão dinâmica dos mecanismos preditivos, uma interface foi definida e três mecanismos preditivos foram implementados e serão discutidos.

III.6.1 Adição de Novos Mecanismos de Predição

Em Python, novos módulos podem ser carregados dinamicamente com o sistema em execução. Utilizando-se a função `__import__`, pode-se carregar um módulo a partir de uma *string* com o nome do módulo. Essa possibilidade é usada pelo sistema ADAGA para carregar os mecanismos preditores. O Código Fonte A.6, no Apêndice A, apresenta os dois métodos criados para permitir o *late bind* dos mecanismos preditores. O método `loadPredictiveMechanisms` é responsável por carregar todos os módulos da pasta “*predictive_mechanism*”, onde devem ser armazenados os módulos dos mecanismos preditores. Após carregar todos os módulos, são extraídas as classes que implementam os mecanismos preditores. Após o carregamento dinâmico dos mecanismos preditores, existe o objeto `predictiveMechanisms`, que é um dicionário com todas as classes de mecanismos preditores carregados indexado pelo nome do mecanismo. O método `choosePredictiveMechanism` recebe como parâmetro um trecho do nome do mecanismo preditor e busca-o no dicionário de mecanismo preditores. Se encontrado um mecanismo preditor com aquele nome, instancia um objeto desse mecanismo preditor e o referencia no atributo `chosenPredictiveMechanismObj`. Se o mecanismo for diferente do anterior, é necessário recalcular todos os valores da série para este novo mecanismo, para que não seja necessário um novo período de aprendizagem.

III.6 Mecanismos de Predição

Novos mecanismos de predição herdam do mecanismo base, descrito na Seção III.6.2 e sobrescrevem os métodos *calculate_next_value* e *calculate_confidence_band* para que sejam utilizados pelo sistema ADAGA.

III.6.2 Mecanismos Preditores Implementados

Como prova de conceito, três mecanismos preditores foram implementados no sistema ADAGA. O primeiro deles, denominado de Mecanismo Base e apresentado na Seção III.6.2, tem por objetivo definir os métodos básicos para os mecanismos preditores. Assim, todos os mecanismos preditores devem herdar desse mecanismo base. Os outros dois mecanismos implementados são preditores propriamente ditos, herdam do mecanismo base e sobrescrevem apenas os métodos *calculate_next_value* e *calculate_confidence_band*, conforme definido na interface.

Mecanismo Base

Este é o mecanismo base para a criação de outros mecanismos preditores. O maior esforço de implementação deste mecanismo não está no cálculo das previsões ou da faixa de confiança, mas sim nos métodos de adição e remoção de valores nos mecanismos preditores, o controle da janela de observação, o cálculo do erro de predição e a verificação de emissão dos alarmes. Cada uma dessas funções pode ser observada no Código Fonte A.7, no Apêndice A, referente a este mecanismo preditor. Por ser o mecanismo base, ele não foi avaliado nos resultados do sistema ADAGA.

Os métodos *calculate_next_value* e *calculate_confidence_band* são definidos de forma simplória, considerando o próximo valor igual ao atual e a faixa de confiança compreendendo desde o menor até o maior valor presente na janela de observação.

III.7 Gerenciamento dos Alarmes

Exponential Smoothing

A implementação do mecanismo preditor *Exponential Smoothing* se concentra nos métodos de cálculo da predição do próximo valor e da faixa de confiança e é apresentada no Código Fonte A.8, no Apêndice A. Os demais métodos são herdados do mecanismo base tal como estão. No método *calculate_next_value*, temos a implementação da Equação II.2 e, no método *calculate_confidence_band*, é calculado o desvio padrão dos valores da janela de observação da série temporal.

Holt-Winters Seasonal

A implementação do mecanismo preditor *Holt-Winters Seasonal*, exposta no Código Fonte A.9, no Apêndice A, foi realizada com um processo semelhante ao do mecanismo *Exponential Smoothing*, ou seja, com a herança do mecanismo base e a sobrescrita dos métodos *calculate_next_value* e *calculate_confidence_band*. No entanto, nesta classe, novos métodos foram criados com finalidades específicas deste mecanismo preditor. O cálculo do próximo valor segue a Equação II.5, e as suas parcelas são mais complexas que as parcelas do mecanismo *Exponential Smoothing*. Assim, métodos como o *calculateA*, *calculateB* e *calculateC* foram implementados para facilitar a depuração dos cálculos e tornar a implementação mais manutenível. O cálculo da faixa de confiança é realizada segundo a Equação II.10.

III.7 Gerenciamento dos Alarmes

Quando um alarme é emitido pelo mecanismo preditor, ou seja, após o acúmulo de alarmes consecutivos em quantidade igual ao valor definido pelo administrador da rede, o sistema de gerenciamento de alarmes é acionado. A percepção dos alarmes é ocorre na classe *TimeSeriesController*, responsável pela gerência das séries temporais. Quando um valor é adicionado à série temporal gera a emissão de um alarme, essa classe aciona o método *alarmReceived*, apresentado no Código Fonte A.5, no Apêndice A.

III.7 Gerenciamento dos Alarmes

Esse método recebe um objeto da classe *Alarm* e o preenche com dados úteis para a identificação da causa da anomalia. A descoberta da causa da anomalia está fora do escopo deste projeto, mas outras ferramentas e propostas com esse intuito podem ser agregadas ao sistema ADAGA facilmente. Ao objeto da classe *Alarm* recebido, são adicionados objetos com todas as características coletadas nas últimas observações, inclusive a observação que gerou a anomalia. A quantidade de observações anteriores adicionadas no relatório do alarme é definida pelo administrador da rede e o valor padrão usado é cinco. Atualmente, por não existir um mecanismo de tratamento dos alarmes integrado ao sistema ADAGA, o relatório de alarme é apenas persistido na base de dados para posterior análise dos falsos positivos e falsos negativos.

Capítulo IV

Avaliação do Sistema

Neste capítulo, o ADAGA é avaliado pela sua capacidade de detectar anomalias. A avaliação ocorre através das taxas de falsos positivos e falsos negativos da detecção. Por falsos positivos, entendem-se todos os alarmes que foram emitidos em momentos sem anomalias e por falsos negativos, considera-se a quantidade de observações de momentos com anomalias que não geraram alarmes, incluindo, portanto, os alarmes acumulados durante o período de histerese.

O protótipo desenvolvido permite analisar o sistema ADAGA. Pode-se observar o cenário de testes realizado para este trabalho na Figura IV.1. O gerente de monitoramento executa os mecanismos de análise do sistema ADAGA, consistindo na gerência das séries temporais, previsões e geração de alarmes. O sistema monitorado possui mecanismos de coleta de informações sobre sua execução. O envio das informações é periódico com período variável seguindo uma distribuição de Poisson com média 15 segundos [35]. Nos testes realizados, o sistema monitorado é um roteador de rede sem fio utilizado em uma rede real implementado em um computador pessoal com processador Intel Pentium IV de 3,20 GHz, 1,5 GB de memória RAM e conexão com a Internet a 100 Mb/s. O computador que executa o gerente de monitoramento possui processador Intel Core i7 950 de 3,06 GHz, 6,0 GB de memória RAM e conexão com o sistema monitorado a 100 Mb/s. A escolha desse cenário de testes em detrimento de um cenário virtualizado ocorre devido a utilização



Figura IV.1: Cenário de testes para análise dos falsos positivos e falsos negativos do sistema ADAGA.

de tráfego real. Como o ponto de acesso estava sendo utilizado normalmente pelos seus usuários, o sistema ADAGA foi avaliado em condições reais, permitindo a análise da eficácia do sistema com perturbações aleatórias próprias do tráfego de produção.

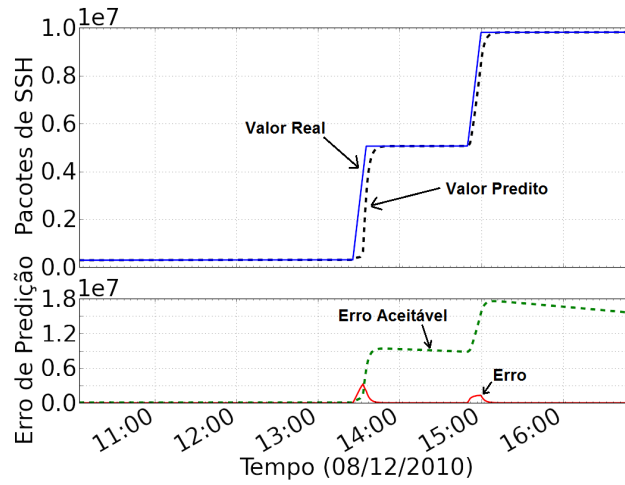
Após, aproximadamente, dois dias de monitoramento, dois nós iniciaram duas transferências consecutivas de um arquivo de 15 gigabytes para o roteador, utilizando o protocolo *Secure Shell* (SSH). Essa transferência fez com que o roteador ficasse inacessível para acesso remoto. O objetivo é que o sistema ADAGA detecte essa transferência e emita relatórios corretamente. Além disso, é avaliado o impacto de cada característica na detecção dessa anomalia. Observa-se que as séries temporais que representam o tráfego de rede da interface de recepção e o fluxo na porta 22 do TCP, usada para o serviço SSH, permitem detectar a anomalia e as séries temporais que representam outras características como número de processos e uso de memória não permitem detectar a anomalia. Observa-se ainda que variáveis como a carga do sistema, embora apresentem comportamento peculiar, com muitos picos, detectam a anomalia, pois a transferência realizada oferece alta carga ao roteador. Neste trabalho, janelas de observação de diversos tamanhos foram avaliadas e apenas resultados com 2000 observações na janela foram apresentados, demais tamanhos de janelas de observações com o mesmo intervalo de observação não apresentaram mudanças significativas nos resultados.

IV.1 Resultados

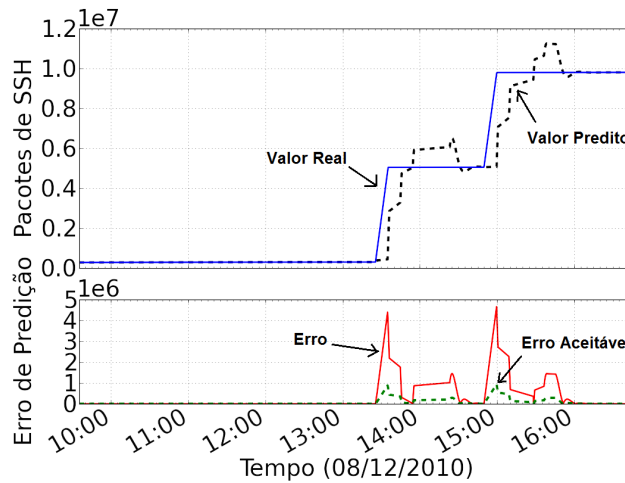
A evolução histórica das observações de recepção de pacotes TCP do serviço SSH, pode ser observada na Figura IV.2. A Subfigura IV.2(a) mostra os resultados para o mecanismo *Exponential Smoothing* e a Subfigura IV.2(b), para o mecanismo *Holt-Winters Seasonal*. Na parte superior dos gráficos, têm-se a evolução do valor real, em linha cheia, e do valor predito pelo mecanismo, em linha tracejada. Desse gráfico, pode-se observar que o mecanismo *Exponential Smoothing* se adapta mais facilmente a evolução do valor real, o que torna a detecção de anomalias, mesmo dessa magnitude, dificultada. Já o mecanismo *Holt-Winters Seasonal* possui maior dificuldade em acompanhar a curva dos valores observados, apresentando melhor detecção da anomalia. Quando um preditor é capaz de prever mudanças bruscas, ele terá baixa eficácia para a detecção das anomalias, que são mudanças bruscas, pois o mecanismo preditor irá acompanhar a mudança de comportamento da série, estimando a anomalia e fazendo com que o sistema não detecte as perturbações na rede. A parte inferior dos gráficos da Figura IV.2 está diretamente ligada à detecção das anomalias. Pode-se observar, em linha cheia, o erro de predição de cada mecanismo, calculado segundo a Equação II.1, e, em linha tracejada, o limite aceitável do erro de predição, obtido pela Equação II.9. Para efeitos de melhor visualização neste projeto, estes gráficos exibem apenas instantes próximos da anomalia gerada.

Na Figura IV.3, observa-se a evolução histórica de duas características monitoradas pelo sistema ADAGA, a carga média do sistema nos cinco minutos antes de cada coleta dos dados, na Subfigura IV.3(a), e o percentual de uso de processador, na Subfigura IV.3(b). A partir da análise dos gráficos, observa-se que variáveis como a carga do sistema sofrem influência em anomalias do tráfego de rede, confirmando a importância do monitoramento multidimensional dos equipamentos de rede. Essas sinalizações em diversas características são úteis para o rastreamento da causa das anomalias, funcionalidade ainda não disponível no ADAGA, mas que é objeto de estudo nos trabalhos futuros. Características como o uso de processamento não foram influenciadas pela anomalia gerada, conforme mostra a Figura IV.3(b).

IV.1 Resultados



(a) Mecanismo *Exponential Smoothing*.

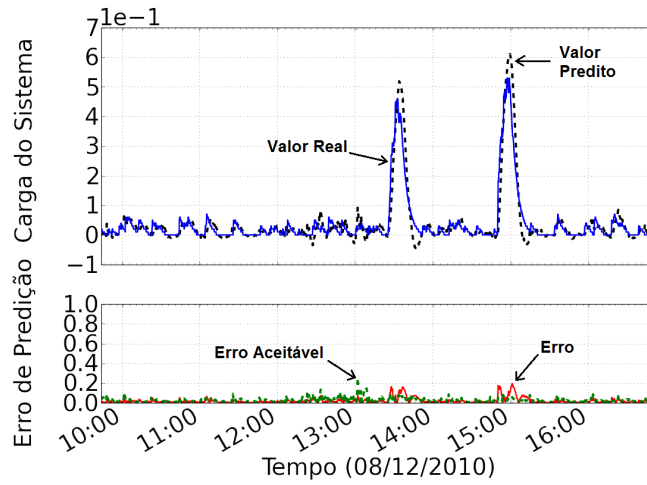


(b) Mecanismo *Holt-Winters Seasonal*.

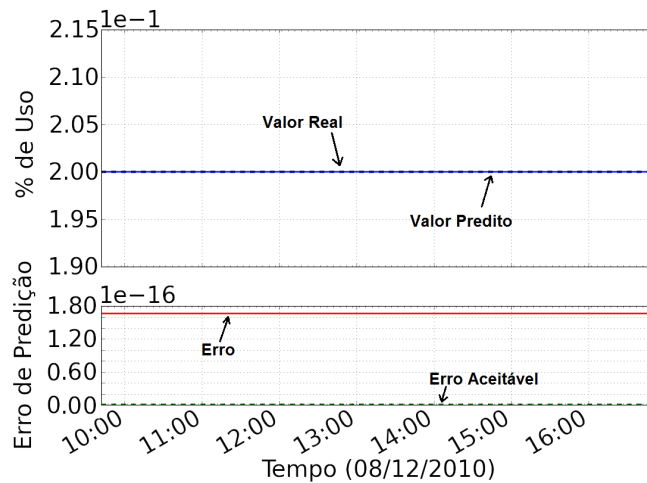
Figura IV.2: Evolução temporal da característica “recepção de pacotes na porta 22 do TCP” de cada mecanismo para α igual a 0,1.

A partir da análise dos gráficos da Figura IV.2, espera-se que as taxas de falsos positivos e falsos negativos do mecanismo *Holt-Winters Seasonal* sejam melhor do que as taxas do mecanismo *Exponential Smoothing*. De fato, pelos gráficos da Figura IV.4, nota-se a melhora nas taxas de falsos positivos e negativos do mecanismo *Exponential Smoothing* e pelos gráficos da Figura IV.5, as taxas do mecanismo *Holt-Winters Seasonal*. Nas Figuras IV.4 e IV.5, pode-se observar o percentual de falsos positivos e falsos negativos de

IV.1 Resultados



(a) Carga média do sistema nos cinco minutos antes de cada observação.

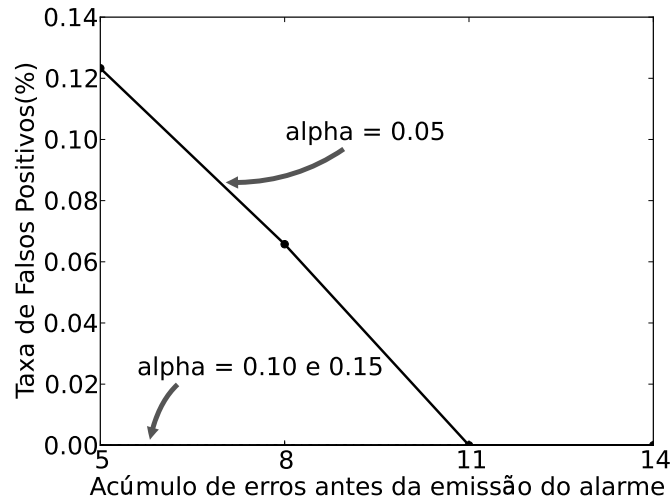


(b) Percentual de uso de processador no roteador.

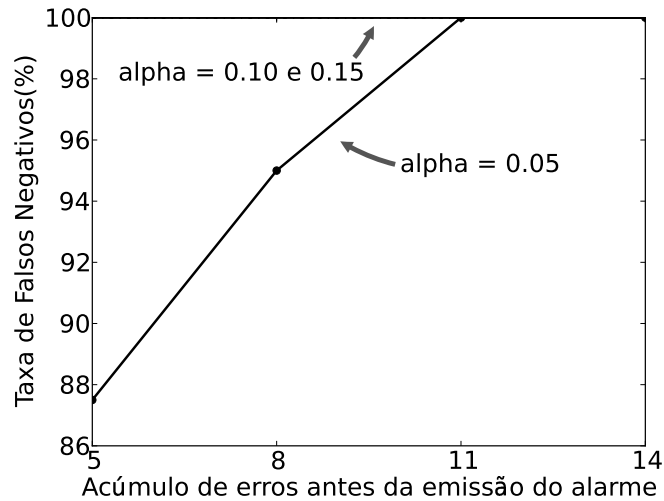
Figura IV.3: Evolução temporal para o mecanismo *Holt-Winters Seasonal* de duas características não correlatas diretamente com o tráfego de rede.

diversas configurações dos mecanismos de predição conforme varia-se o parâmetro η , que representa a quantidade de alarmes acumulados antes da emissão do relatório. Cada uma das curvas apresentadas representa uma configuração de parâmetros para os mecanismos preditivos. No caso do *Exponential Smoothing*, os valores de α testados foram 0,05, 0,10 e 0,15. A escolha desses valores é devido ao resultado obtido por Lucena e Moura [23], que defende o uso de valores menores que 0,10 para a detecção de anomalias em redes

IV.1 Resultados



(a) Falsos positivos.



(b) Falsos negativos.

Figura IV.4: Análise para a característica “recepção de pacotes na porta 22 do TCP” para o mecanismo *Exponential Smoothing*, variando o parâmetro η para diversos valores de α .

de computadores. Foram avaliados valores próximos acima e abaixo do valor indicado por Lucena e Moura. Os mesmos valores foram aplicados aos parâmetros α , β e γ do mecanismo *Holt-Winters Seasonal* (0,05, 0,10 e 0,15), o que resultou em 27 curvas, divididas nas subfiguras da Figura IV.5. As curvas foram divididas em subfiguras através do parâmetro α . Portanto, nas Subfiguras IV.5(a) e IV.5(b) estão as curvas para os resultados com o parâmetro $\alpha = 0.05$, nas Subfiguras IV.5(c) e IV.5(d), $\alpha = 0.10$ e nas

IV.1 Resultados

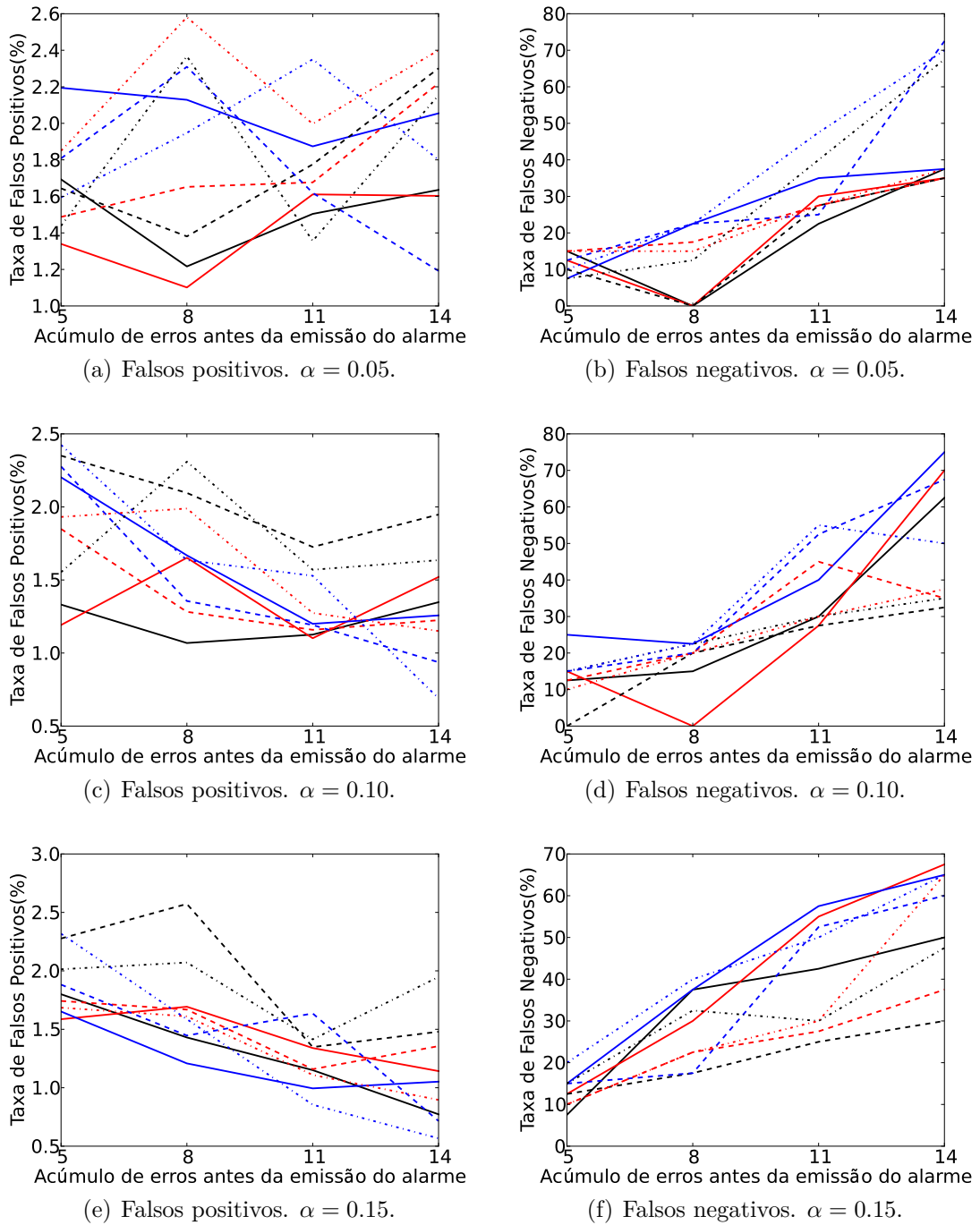


Figura IV.5: Análise para a característica “recepção de pacotes na porta 22 do TCP” para o mecanismo *Holt-Winters Seasonal*, variando o parâmetro η para diversos valores de α , β e γ .

Subfiguras IV.5(e) e IV.5(f), $\alpha = 0.15$. Em todas as figuras, os parâmetros β e γ são expressados através de código de estilo de linhas e de cores, respectivamente. Assim, as

IV.1 Resultados

Tabela IV.1: Tabela comparativa das taxas de falsos positivos para as configurações de α , β e γ apresentadas nas Figuras IV.5(a), IV.5(c) e IV.5(e) com $\eta = 5$. Valores percentuais.

β	0.05			0.10			0.15		
$\alpha \setminus \gamma$	0.05	0.10	0.15	0.05	0.10	0.15	0.05	0.10	0.15
0.05	1,69	1,34	2,19	1,64	1,49	1,81	1,44	1,85	1,59
0.10	1,33	1,19	2,20	2,35	1,85	2,28	1,55	1,93	2,42
0.15	1,80	1,59	1,65	2,28	1,74	1,88	2,01	1,68	2,32

Tabela IV.2: Tabela comparativa das taxas de falsos negativos para as configurações de α , β e γ apresentadas na Figura IV.5(b), IV.5(d) e IV.5(f) com $\eta = 5$. Valores percentuais.

β	0.05			0.10			0.15		
$\alpha \setminus \gamma$	0.05	0.10	0.15	0.05	0.10	0.15	0.05	0.10	0.15
0.05	15,0	12,5	7,5	10,0	15,0	12,5	7,5	15,0	10,0
0.10	12,5	15,0	25,0	0	12,5	15,0	15,0	10,0	15,0
0.15	7,5	12,5	15,0	12,5	10,0	15,0	15,0	10,0	20,0

linhas cheias representam os resultados para $\beta = 0.05$, as linhas tracejadas, para $\beta = 0.10$ e as linhas traço-ponto, para $\beta = 0.15$. Da mesma forma, para as cores, as linhas em preto representam $\gamma = 0.05$, as linhas em vermelho, $\gamma = 0.10$ e as linhas em azul, $\gamma = 0.15$. Os gráficos das Figuras IV.4 e IV.5 são considerados a partir de uma janela de observação de 2000 amostras. Testes realizados para outros tamanhos de janela de observação e com o mesmo intervalo de amostragem apresentaram resultados semelhantes para a detecção da anomalia gerada.

Nas análises realizadas, a taxa de falsos positivos foi calculada em cima de todas as observações coletadas, buscando-se obter a quantidade de alarmes emitidos em momentos nos quais não havia anomalias. A taxa de falsos negativos foi obtida considerando-se os alarmes não emitidos durante o período em que havia a anomalia em relação ao total das observações coletadas durante o momento da anomalia.

Com base na definição adotada, comprova-se a baixa eficácia do mecanismo *Exponential Smoothing*, pois, apesar de ter baixas taxas de falsos positivos (Figura IV.4(a)), ele não foi capaz de detectar a anomalia gerada, obtendo altas taxas de falsos negativos

IV.1 Resultados

(Figura IV.4(b)). No entanto, todas as variações de parâmetros testadas no mecanismo *Holt-Winters Seasonal* apresentaram boa eficácia, com taxas de falsos positivos inferiores a 2,5% e taxas de falsos negativos adequadas com o acúmulo de alarmes definidos. No intervalo em que ocorreram as anomalias, foram coletadas 40 amostras. Dessa forma, uma taxa de falsos negativos igual a $1/40$ do valor de acúmulo de erros representa que todas as observações em que haviam anomalias foram detectadas pelo sistema ADAGA no uso do mecanismo *Holt-Winters Seasonal*. De fato, é o que ocorre, pois, por exemplo, para o valor $\eta = 5$, tem-se que a taxa de falsos negativos esperada é em torno de 12,5%, valor que concentra grande parte das curvas analisadas.

Para uma análise mais detalhada, pode-se observar a taxa de falsos positivos na Tabela IV.1 e de falsos negativos na Tabela IV.2 para cada uma das 27 configurações de parâmetros testada para o mecanismo *Holt-Winters Seasonal*. As linhas demonstram os valores de falsos positivos e negativos para cada valor de α e as colunas apresentam os valores de falsos positivos e falsos negativos para cada valor de γ agrupados pelos valores de β . A partir dessas tabelas, conclui-se que existe grande semelhança dos resultados para as diferentes configurações. No entanto, um pequeno ganho pode ser observado para valores menores de β . Conforme visto na Subseção II.4.2, o parâmetro β está relacionado com a tendência da série. Portanto, quando o preditor aceita mudanças mais drásticas na tendência da série temporal, ou seja, β grande, mais rapidamente o preditor se adapta à anomalia e, por consequência, menores são as taxas de acertos.

Capítulo V

Conclusão

De fato, a Internet é um sucesso. Com mais de 1,86 bilhões de usuários em todo o mundo em 2009, a Internet é o maior sistema complexo do mundo. No entanto, a forma em que se desenvolveu, desde o seu surgimento na década de 1970 até os dias de hoje, promoveu um verdadeiro engessamento da rede. Pouco se modificou na arquitetura da rede, seu núcleo permaneceu simples e as tarefas mais complexas são realizadas nas extremidades da rede. A comutação de pacotes, formato de envio de informação na rede que permite o meio ser compartilhado por diversos usuários, ao mesmo tempo e sob demanda, não permite garantir valores máximos de atraso ou de perda na rede. Dessa forma, aplicações sensíveis a atrasos e variação no atraso, como por exemplo, uma cirurgia realizada à distância, não podem ser realizadas na Internet, visto que esta oferece o serviço de melhor esforço na entrega de pacotes e não garante nem a entrega dos pacotes nem um prazo para a entrega. Outras deficiências da Internet como a quantidade de endereços disponíveis e a necessidade de segurança em algumas aplicações foram corrigidas através de “remedios” na arquitetura da rede, como o *Network Address Translation* (NAT) e o *Transport Layer Security* (TLS), respectivamente. Soluções como o NAT, inclusive, ferem princípios originais da Internet, como o endereço global e único para as estações finais.

Na comunidade acadêmica, existe um consenso de que a Internet precisa ser completamente reformulada para formar uma nova Internet, a “Internet do Futuro”. Diversas

são as propostas para a Internet do Futuro, como as que defendem o modelo purista e as que defendem o modelo pluralista, mas a tecnologia de virtualização aplicada ao contexto de virtualização de rede se apresenta como a mais promissora. Nesse contexto, a caracterização da rede é importante, pois permite o conhecimento do comportamento dos elementos da rede e do tráfego que utiliza os enlaces da rede. A implantação da virtualização na rede promove o aumento da complexidade do processo de caracterização da rede, pois novas o espaço de monitoramento é maior, uma vez que devem ser monitoradas todas as redes virtuais e a plataforma de virtualização em adição a já monitorada rede física. Além disso, peculiaridades do monitoramento de redes virtuais existem, pois um fenômeno observado em uma rede virtual pode ser gerado por fatos ocorridos em outras redes virtuais ou mesmo na rede física. Dessa forma, o monitoramento e a análise para a caracterização da rede em ambientes virtualizados requer sistemas de monitoramento e análise projetados para esses ambientes e ser capaz de considerar a possível correlação nos fenômenos observados em cada uma das diversas redes virtuais em execução sobre a rede física, em um dado momento.

Neste trabalho, propôs-se o ADAGA, sistema de Detecção de Anomalias para o Gerenciamento Autônomo de redes virtuais, que visa a coleta e análise de dados de redes virtuais. O sistema ADAGA coleta dados de ferramentas próprias do sistema operacional dos nós monitorados, como quantidade de pacotes transmitidos, percentual de uso de processador, quantidade de processos em execução entre outras, e do tráfego de rede que é recebido ou transmitido pelo nó monitorado. O sistema permite a análise dos dados coletados em mecanismos de predição para séries temporais. Quando o mecanismo de predição erra o próximo valor da série, o sistema pode gerar um alarme, detectando a anomalia. Um sistema de histerese na geração dos alarmes foi projetado para evitar a emissão de alarmes em erros pontuais dos mecanismos preditores. Da análise dos mecanismos preditores, conclui-se que um bom estimador de séries temporais não é um bom mecanismo para detectar anomalias, pois esse mecanismo acertará as predições, mesmo em situações extremas de anomalias, onde o comportamento dos dados observados muda drasticamente. Esse fato impossibilita a detecção das anomalias por mecanismos preditores bons.

As principais vantagens do sistema ADAGA são o monitoramento multidimensional, o baixo intervalo de monitoramento, o qual é responsável por baixos tempos de reação, e o fato de não usar estratégias de amostragem para o processamento dos pacotes. Como o sistema não considera os dados dos pacotes como um todo, mas apenas suas estatísticas e contadores, a sobrecarga é baixa, o que torna o processamento de todos os pacotes viável. O sistema ADAGA possui dois mecanismos para a predição de séries temporais e permite que novos mecanismos sejam integrados dinamicamente. O mecanismo foi avaliado em roteador de acesso a rede sem-fio, que estava em uso por usuários normais da rede e, portanto, sujeito a anomalias e a comportamentos pontuais naturais dos usuários da rede. Uma anomalia foi gerada manualmente e o sistema ADAGA foi testado na sua capacidade de detectar a anomalia com boa eficácia. Os resultados obtidos mostram que, no cenário de teste, o mecanismo *Exponential Smoothing* não é adequado para a predição de redes de computadores, pois as redes possuem sazonalidades não consideradas por esse mecanismo. Esse mecanismo de predição não detectou as anomalias geradas, apresentando 100% de taxa de falsos positivos. Os resultados mostram ainda que a multidimensionalidade da análise realizada no ADAGA se apresenta como uma forte aliada ao gerenciamento autônomo, visto que várias características do equipamento monitorado podem sinalizar anomalias ao mesmo tempo, facilitando o rastreamento da causa das anomalias, que é objeto dos trabalhos futuros. A análise dos falsos positivos e falsos negativos mostra que, no cenário avaliado, o mecanismo *Holt-Winters Seasonal* possui melhor eficácia, pois detectou as anomalias da rede com baixas taxas de falsos positivos, de 2,5% no melhor caso, e com taxas de falsos negativos compatíveis com o sistema de histerese na emissão de alarmes presente no ADAGA.

Os trabalhos futuros desse projeto se enquadram nas diversas etapas do ciclo de um sistema autônomo: sensoriamento, análise, decisão e atuação. Com relação ao sensoriamento dos sistemas monitorados, inclui-se a implementação de novos mecanismos de coleta tanto para novas características de roteadores virtuais quanto o sensoriamento de equipamentos de redes de nova geração, como celulares, sensores, veículos. Para a etapa da análise dos dados coletados, considera-se a implementação e a avaliação de novos mecanismos preditores e novas possibilidades de pré-processamento das séries temporais.

Componentes de tomada de decisão e atuação na rede não estão incluídos no sistema ADAGA, mas a construção de elementos que permitam a descoberta da causa das anomalias e a escolha de soluções computacionais adequadas para a resolução dos problemas detectados, além da integração com interfaces de atuação existentes, são possibilidades futuras para transformar o sistema ADAGA em um sistema autônomo completo.

Além disso, a distribuição dos gerentes de coleta na rede se apresenta como um problema complexo e que deve ser estudado nos trabalhos futuros. Deve-se definir de forma distribuída, a quantidade de gerentes de monitoramento, em que nós da rede cada um dos gerentes deve ser alocado e quais nós da rede cada gerente deve monitorar e analisar. Tal decisão deve considerar a sobrecarga de mensagens na rede, promovendo a proximidade com entre o gerente de monitoramento e o sistema monitorado, e a carga de processamento do nó que aloca o gerente de monitoramento, pois a análise dos sistemas monitorados não deve afetar o funcionamento normal do nó.

Referências Bibliográficas

- [1] M. Moreira, N. Fernandes, L. Costa e O. Duarte, “Internet do futuro: Um novo horizonte”, *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2009*, pp. 1–59, 2009.
- [2] T. Anderson, L. Peterson, S. Shenker e J. Turner, “Overcoming the Internet impasse through virtualization”, *Computer*, vol. 38, no. 4, pp. 34–41, 2005.
- [3] N. Fernandes, M. Moreira, I. Moraes, L. Ferraz, R. Couto, H. Carvalho, M. Campista, L. Costa e O. Duarte, “Virtual networks: Isolation, performance, and trends”, *Annals of Telecommunications*, pp. 1–17, 2010.
- [4] J. Kephart e D. Chess, “The vision of autonomic computing”, *Computer*, vol. 36, no. 1, pp. 41–50, 2003.
- [5] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt e F. Zambonelli, “A survey of autonomic communications”, *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 1, no. 2, pp. 223–259, 2006.
- [6] J. Brutlag, “Aberrant behavior detection in time series for network monitoring”, *Proceedings of the 14th USENIX conference on System administration*, 2000.
- [7] R. Moskowitz e P. Nikander, “Host Identity Protocol (HIP) Architecture”. RFC 4423 (Informational), maio de 2006.

-
- [8] Lyno Henrique Gonçalves Ferraz, “Uma Avaliação do Protocolo HIP para Provisão de Mobilidade na Internet”, Projeto Final, DEL/POLI/UFRJ, março de 2010.
- [9] D. D. Clark, C. Partridge, J. C. Ramming e J. T. Wroclawski, “A knowledge plane for the Internet”, *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '03)*, New York, NY, USA, pp. 3–10, ACM, 2003.
- [10] J. Case, M. Fedor, M. Schoffstall e J. Davin, “Simple Network Management Protocol (SNMP)”. RFC 1157 (Historic), maio de 1990.
- [11] K. Egevang e P. Francis, “The IP Network Address Translator (NAT)”. RFC 1631 (Informational), maio de 1994. Obsoleted by RFC 3022.
- [12] R. Droms, “Dynamic Host Configuration Protocol”. RFC 2131 (Draft Standard), março de 1997. Updated by RFCs 3396, 4361, 5494.
- [13] P. Mockapetris, “Domain names - implementation and specification”. RFC 1035 (Standard), novembro de 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966.
- [14] A. Feldmann, “Internet clean-slate design: what and why?”, *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 59–64, julho de 2007.
- [15] N. Feamster, L. Gao e J. Rexford, “How to lease the Internet in your spare time”, *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 61–64, janeiro de 2007.
- [16] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel e L. Mathy, “Network virtualization architecture: Proposal and initial prototype”, *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pp. 63–72, ACM, 2009.
- [17] M. e. a. Achemlal, “Virtualisation approach: Concept”, Relatório Técnico, The FP7 4WARD Project - Architecture and Design for the Future Internet., 2010.

-
- [18] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt e A. Warfield, “Xen and the art of virtualization”, *Proceedings of the nineteenth ACM symposium on operating systems principles*, pp. 164–177, ACM, 2003.
- [19] A. Ziviani e O. Duarte, “Metrologia na Internet”, *Minicursos do XXIII Simpósio Brasileiro de Redes de Computadores, SBRC*, pp. 285–329, 2005.
- [20] A. Patcha e J. Park, “An overview of anomaly detection techniques: Existing solutions and latest technological trends”, *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.
- [21] P. Barford e D. Plonka, “Characteristics of network traffic flow anomalies”, *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pp. 69–73, ACM, 2001.
- [22] P. Pisa, N. Fernandes, H. Carvalho, M. Moreira, M. Campista, L. Costa e O. Duarte, “OpenFlow and Xen-Based Virtual Network Migration”, *Communications: Wireless in Developing Countries and Networks of the Future*, pp. 170–181, 2010.
- [23] S. de Lucena e A. de Moura, “Análise dos Estimadores EWMA e Holt-Winters para Detecção de Anomalias em Tráfego IP a partir de Medidas de Entropia”, *CSBC2009*, 2009.
- [24] F. Silveira e C. Diot, “URCA: Pulling out anomalies by their root causes”, *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, IEEE, 2010.
- [25] F. Silveira, C. Diot, N. Taft e R. Govindan, “ASTUTE: Detecting a different class of traffic anomalies”, *Proceedings of the ACM SIGCOMM 2010 Conference*, Nova Deli, India, ACM, September 2010.
- [26] A. Soule, K. Salamatian e N. Taft, “Combining filtering and statistical methods for anomaly detection”, *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, p. 31, USENIX Association, 2005.

-
- [27] P. Barford, J. Kline, D. Plonka e A. Ron, “A signal analysis of network traffic anomalies”, *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*, pp. 71–82, ACM, 2002.
- [28] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May e A. Lakhina, “Impact of packet sampling on anomaly detection metrics”, *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pp. 159–164, ACM, 2006.
- [29] S. Ali, I. Haq, S. Rizvi, N. Rasheed, U. Sarfraz, S. Khayam e F. Mirza, “On mitigating sampling-induced accuracy loss in traffic anomaly detection systems”, *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 3, pp. 4–16, 2010.
- [30] D. Brauckhoff, K. Salamatian e M. May, “A signal processing view on packet sampling and anomaly detection”, *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, IEEE, 2010.
- [31] I. Paredes-Oliva, X. Dimitropoulos, M. Molina, P. Barlet-Ros e D. Brauckhoff, “Automating root-cause analysis of network anomalies using frequent itemset mining”, *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 467–468, 2010.
- [32] A. Liotta, G. Pavlou e G. Knight, “Exploiting agent mobility for large-scale network monitoring”, *Network, IEEE*, vol. 16, no. 3, pp. 7–15, 2002.
- [33] N. Fernandes, M. Moreira e O. Duarte, “A self-organized mechanism for thwarting malicious access in ad hoc networks”, *INFOCOM, 2010 Proceedings IEEE*, pp. 1–5, IEEE, 2010.
- [34] P. Brockwell e R. Davis, *Introduction to time series and forecasting*. Springer Verlag, 2002.
- [35] V. Paxson, “On calibrating measurements of packet transit times”, *SIGMETRICS/PERFORMANCE: Joint International Conference on Measurement and Modeling of Computer Systems*, Madison, WI, 1998.

-
- [36] A. Koehler, R. Snyder e J. Ord, “Forecasting Models and Prediction Intervals for the Multiplicative Holt-Winters Method”, *Monash Econometrics and Business Statistics Working Papers*, 1999.

Apêndice A

Trechos de Código Fonte

Neste apêndice, são apresentados os trechos de código fonte descritos nas seções do Capítulo III, que versa sobre a implementação do Sistema ADAGA.

A.1 Estrutura Básica da Mensagem do ADAGA

Código Fonte A.1: Definição da estrutura XML do cabeçalho da mensagem de requisição esperada pelo servidor construído.

```
1 <?xml version="1.0" encoding="utf-8"?>
2
3 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.w3schools.com" xmlns="http://www.
  w3schools.com" elementFormDefault="qualified">
4   <xs:element name="XenPrototype" >
5     <xs:complexType><xs:sequence>
6       <xs:element name="NodePK" type="xs:string" />
7       <xs:element name="Application" >
8         <xs:complexType><xs:sequence>
9           <xs:element name="ApplicationId" type="xs:integer
            " />
10          <xs:element name="AplicationRequest" >
```

A.2 Sistema de Comunicação

```
11         <xs:complexType><xs:sequence>
12             <xsd:annotation>Depends on application.</
13                 xsd:annotation>
14         </xs:sequence></xs:complexType>
15     </xs:element>
16 </xs:sequence></xs:complexType>
17 </xs:element>
18 </xs:sequence></xs:complexType>
19 </xs:element>
20 </xs:schema>
```

A.2 Sistema de Comunicação

Código Fonte A.2: Servidor da comunicação entre sistema monitorado e gerente de monitoramento. Executa no sistema monitorado.

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  from threading import Thread
5  from xml.dom import minidom
6  import traceback
7  import sys
8
9  from xenprototype.utils.generalConst import GeneralConst
10
11 class BindServer(object):
12
13     def __init__(self, serverType):
14         self.exiting = False
15         return
16
17     def exit(self):
18         self.exiting = True
19         return
```

A.2 Sistema de Comunicação

```
20
21 def bindServer(self, serverType, port=GeneralConst.DEFAULT_PORT):
22     # Echo server program
23     import socket
24     import sys
25
26     HOST = None                # Symbolic name meaning all
27                               available interfaces
28     PORT = port                # Arbitrary non-privileged port
29     s = None
30     for res in socket.getaddrinfo(HOST, PORT, socket.AF_UNSPEC,
31                                   socket.SOCK_STREAM, 0, socket.AI_PASSIVE):
32         af, socktype, proto, canonname, sa = res
33         try:
34             s = socket.socket(af, socktype, proto)
35         except socket.error, msg:
36             print msg
37             s = None
38             continue
39         try:
40             s.bind(sa)
41             s.listen(1)
42         except socket.error, msg:
43             print msg
44             s.close()
45             s = None
46             continue
47         break
48     if s is None:
49         print 'could not open socket'
50         sys.exit(1)
51     print "The server is ready in the " + self.
52         serverTypeTranslate(serverType) + " mode. Waiting for
53         conections..."
54
55     while not(self.exiting):
```

A.2 Sistema de Comunicação

```
52         conn, addr = s.accept()
53         print 'Connected by', addr
54         communication = bindServerThread.BindServerThread(
55             serverType, conn, addr)
56         communication.start()
57         return
58     def serverTypeTranslate(self, serverType):
59         serverTypeDict = {"d": "default", "c": "controller", "p": "
60             physical node", "v": "virtual node"}
61         return serverTypeDict[serverType]
62 class BindServerThread (Thread):
63     '''This class process the remote requests. It is inside another
64         thread in order not to block the server while the request is
65         being processed.'''
66     SERVER_TYPE_DICT = {"d": selectApplication.SelectApplication, "c":
67         selectApplication.SelectApplicationController, "p":
68         selectApplication.SelectApplicationPhysicalNode, "v":
69         selectApplication.SelectApplicationVirtualNode}
70
71     def __init__(self, serverType, connection, address):
72         Thread.__init__(self)
73         self.connection = connection
74         self.address = address
75         self.serverType = serverType
76         return
77
78     def run(self):
79         try:
80             print self.getName(), " processing request."
81             message = ""
82             while 1:
83                 message += self.connection.recv(1024)
84                 if "</XenPrototype>" in message: break
```

A.2 Sistema de Comunicação

```
81         if self.serverType == "c":
82             applicationSelector = self.SERVER_TYPE_DICT[self.
                serverType](message, self.address[0])
83         else:
84             applicationSelector = self.SERVER_TYPE_DICT[self.
                serverType](message)
85         self.connection.send(applicationSelector.
                callApplicationHandler())
86     except Exception as detail:
87         print 'Handling run-time error in', self.getName(), ':',
                type(detail)
88         print 'Handling run-time error in', self.getName(), ':',
                detail
89         exc_type, exc_value, exc_traceback = sys.exc_info()
90         traceback.print_exception(exc_type, exc_value,
                exc_traceback, file=sys.stdout)
91
92     finally:
93         self.connection.close()
94
95 class SelectApplication():
96
97     TAG_APP = "Application"
98     TAG_APP_ID = "ApplicationId"
99     TAG_XEN_PROTO = "XenPrototype"
100
101     def __init__(self, message):
102         '''Constructor of SelectApplication class. It is responsible
                for receiving the XML message and discovering the
                application ID of this message.'''
103
104         self.handlersIndexDict = {1:self.MeasuresGathererHandler}
105
106         self.message = message
107         self.getApplicationId()
108     return
```

A.2 Sistema de Comunicação

```
109
110     def getApplicationId(self):
111         #Parsing xml file
112         dom = minidom.parseString(self.message)
113
114         #Retrieving root node
115         xen = dom.getElementsByTagName(self.TAG_XEN_PROTO)[0]
116
117         #Retrieving the application node
118         app = xen.getElementsByTagName(self.TAG_APP)[0]
119
120         #Retrieving the application id node
121         appIdElement = app.getElementsByTagName(self.TAG_APP_ID)[0]
122
123         #Retrieving the application id data
124         appId = appIdElement.firstChild.data
125
126         self.applicationId = int(appId)
127         return self.applicationId
128
129     def callApplicationHandler(self):
130         """Each application has to have a handler. This method
131           returns True if the transation was successfull or False if
132           don't."""
133
134         if self.applicationId in self.handlersIndexDict.keys():
135             return self.handlersIndexDict[self.applicationId]()
136         else:
137             return "<XenPrototype>ApplicationIdError</XenPrototype>"
138
139     #Application Handlers
140     # MeasuresGatherer Handler
141     def MeasuresGathererHandler(self):
142         from xenprototype.VirtualMachineMeasurements.src.
143             measuresGathererHandler import MeasuresGathererHandler
144         handler = MeasuresGathererHandler(self.message)
```

A.2 Sistema de Comunicação

```
142         return handler.getResponseMessage()
```

Código Fonte A.3: Cliente da comunicação entre sistema monitorado e gerente de monitoramento. Executa no gerente de monitoramento.

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  from xenprototype.utils.generalConst import GeneralConst
4  import socket
5
6  class ClientConnect(object):
7
8      def __init__(self, message = "", host = "localhost", port =
9          GeneralConst.DEFAULT_PORT):
10         self.message = message
11         self.host = host
12         self.port = port
13         return
14
15     def connectToServer(self, message = None, host = None, port =
16         None, waitForResponse = True):
17         self.waitForResponse = waitForResponse
18
19         if not (host == None):
20             self.host = host
21         if not (port == None):
22             self.port = port
23         if not (message == None):
24             self.message = message
25
26         HOST = self.host    # The remote host
27         PORT = self.port    # The same port as used by the server
28         s = None
29
30         for res in socket.getaddrinfo(HOST, PORT, socket.AF_UNSPEC,
31             socket.SOCK_STREAM):
32             af, socktype, proto, canonname, sa = res
```

A.2 Sistema de Comunicação

```
30
31     try:
32         s = socket.socket(af, socktype, proto)
33     except socket.error, msg:
34         s = None
35         print "Error Message: ", msg
36         continue
37     try:
38         s.connect(sa)
39     except socket.error, msg:
40         s.close()
41         s = None
42         print "Error Message: ", msg
43         continue
44     break
45
46     if s is None:
47         print 'could not open socket'
48         return False
49     s.send(self.message)
50     self.answer = ""
51     if self.waitForResponse == True:
52         while 1:
53             self.answer += s.recv(1024)
54             if "</XenPrototype>" in self.answer: break
55         s.close()
56
57         return self.processingAnswer()
58     else:
59         s.close()
60         return
61
62 def processingAnswer(self):
63     return self.answer
```

A.3 Séries Temporais

Código Fonte A.4: Métodos da classe de controle das séries temporais.

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 '''
4 @author: pisa
5 '''
6
7 from threading import Thread
8 import time, cPickle, sys, os
9 from temporal_serie_list import TemporalSerieList
10 from constants import Constants
11 global constantsObj
12
13 class TemporalSerieController(Thread):
14     '''
15     Controls the time series of each monitored system
16     '''
17     def addMonitoration(self, machineObjList):
18         machineObjList[2].verifyVariableType()
19         alarm = None
20         try:
21             alarm = self.temporalSerieListBySource[machineObjList
22                 [0][0]].addElement(machineObjList[2],
23                 machineObjList[1])
24         except KeyError:
25             self.temporalSerieListBySource[machineObjList[0][0]] =
26                 TemporalSerieList(self.constantsObj)
27             for monitoringVariable in self.monitoringVariables:
28                 self.temporalSerieListBySource[machineObjList[0][0]].
29                     monitorVariable(monitoringVariable)
30             self.temporalSerieListBySource[machineObjList[0][0]].
31                 setPredictiveMechanism()
32             alarm = self.temporalSerieListBySource[machineObjList
33                 [0][0]].addElement(machineObjList[2],
```

A.3 Séries Temporais

```
        machineObjList[1])
28     if (alarm != None):
29         self.alarmReceived(alarm, machineObjList)
30     return
31
32     def monitorVariable(self, variable):
33         if (variable not in self.monitoringVariables):
34             for temporal_serie_list_by_source in self.
                 temporalSerieListBySource.values():
35                 temporal_serie_list_by_source.ts_list[variable] =
                     TemporalSerie(variable, self.constantsObj)
36                 validVariable = True
37                 for machine in temporal_serie_list_by_source.machines
                     :
38                     validVariable = (temporal_serie_list_by_source.
                         addValueToTS(machine, variable) != None)
39                     if not(validVariable):
40                         break
41                     if not(validVariable):
42                         temporal_serie_list_by_source.forgetVariable(
                             variable)
43                 return
44                 self.monitoringVariables.append(variable)
45     return
46
47     def forgetVariable(self, variable):
48         if (variable in self.monitoringVariables):
49             for temporal_serie_list_by_source in self.
                 temporalSerieListBySource.values():
50                 del temporal_serie_list_by_source.ts_list[variable]
51                 self.monitoringVariables.remove(variable)
```

Código Fonte A.5: Método que processa os alarmes reportados pelas séries temporais.

```
1 def alarmReceived(self, alarm, machineObjList):
2     alarm.setActualMachineObj(machineObjList)
```

A.4 Mecanismos Preditores

```
3     alarm.setNumberOfNearMachines(self.constantsObj.  
        ALARM_NEAR_MACHINES)  
4     count = 0  
5     for machineIndex in range(len(self.temporalSerieListBySource [  
        machineObjList [0][0]].machines), 0, -1):  
6         count += 1  
7         alarm.addNearMachineObjects(self.temporalSerieListBySource [  
            machineObjList [0][0]].machines [machineIndex - 1])  
8         if(count >= alarm.numberOfNearMachines):  
9             break  
10    alarm.process()  
11    alarm.persist()  
12    return
```

A.4 Mecanismos Preditores

Código Fonte A.6: Métodos responsáveis pelo carregamento dinâmico de mecanismos preditores sem que o sistema precise ser reiniciado.

```
1 def loadPredictiveMechanisms(self):  
2     package = "predictive_mechanism"  
3     mechanisms = os.listdir(package)  
4     mechanism_name = []  
5     for mechanism in mechanisms:  
6         if(mechanism[-3:] == ".py" and mechanism[1] != "_"):  
7             mechanism_name.append(".".join(mechanism.split('.')[:-1])  
                )  
8     mechanisms_import_obj = __import__(package, fromlist=  
        mechanism_name)  
9     for mechanism in mechanisms_import_obj.__dict__.keys():  
10        if(mechanism[0] != '_'):  
11            self.predictiveMechanisms [mechanisms_import_obj.__dict__[  
                mechanism].__name__] = mechanisms_import_obj.__dict__[  
                    mechanism]  
12    return
```

A.4 Mecanismos Preditores

```
13
14 def choosePredictiveMechanism(self, pieceOfName):
15     oldPredictiveMechanism = self.chosenPredictiveMechanismObj
16     for mechanism_name in self.predictiveMechanisms.keys():
17         if(pieceOfName in mechanism_name):
18             self.chosenPredictiveMechanism = self.
19                 predictiveMechanisms[mechanism_name]
20             break
21     if(self.chosenPredictiveMechanism == None):
22         return False
23     for value in inspect.getmembers(self.chosenPredictiveMechanism):
24         if(value[0][0] != '_'):
25             if(type(value[1]) == type(".__class__")):
26                 tmpObj = value[1]()
27                 if(isinstance(tmpObj, predictive_mechanism.
28                     base_mechanism.BaseMechanism)):
29                     if(type(tmpObj) != type(oldPredictiveMechanism)):
30                         self.chosenPredictiveMechanismObj = value
31                             [1](self.constantsObj)
32                             self.chosenPredictiveMechanismObj.
33                                 input_new_values(self.values)
34                             self.dumpToBePlotterActive()
35                     break
36     return True
```

Código Fonte A.7: Mecanismo de predição base para outros mecanismos de predição.

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 '''
4 @author: pisa
5 '''
6
7 from collections import deque
8 from constants import Constants
9
10 class BaseMechanism(object):
```

A.4 Mecanismos Preditores

```
11
12     def __init__(self, constantsObj = Constants(), windowSize=None,
13                 errorThreshold=None):
14
15         if windowSize == None:
16             self.windowSize = self.constantsObj.MONITORING_SIZE
17         else:
18             self.windowSize = windowSize
19
20         if errorThreshold == None:
21             self.errorThreshold = self.constantsObj.ERROR_THRESHOLD
22         else:
23             self.errorThreshold = errorThreshold
24
25         self.values = deque()
26         self.predicted_values = deque()
27         self.confidence_bands = deque()
28         self.errors = deque()
29
30
31         self.next_value = 0
32         self.predictive_error = 0
33
34         self.errorCount = 0
35         return
36
37     def input_new_values(self, values):
38         if (type(values) == type([]) or type(values) == type(deque()))
39             :
40             for value in values:
41                 self.__add_new_value(value)
42         elif (type(values) == type(1.0) or type(values) == type(1) or
43              type(values) == type(long(1))):
44             self.__add_new_value(values)
45         return
```

A.4 Mecanismos Preditores

```
44
45     def __add_new_value(self, value):
46         self.values.append(value)
47         if(len(self.values) > self.windowSize):
48             self.removeValue()
49         self.calculate_next_value()
50         self.calculate_confidence_band()
51         return
52
53     def removeValue(self):
54         self.values.popleft()
55         self.confidence_bands.popleft()
56         self.errors.popleft()
57         self.predicted_values.popleft()
58         return
59
60     def get_next_value(self):
61         return self.next_value
62
63     def calculate_next_value(self):
64         """
65         In the base mechanism, it returns the last value
66         """
67         self.next_value = self.values[-1]
68         return
69
70     def calculate_confidence_band(self):
71         self.confidence_bands.append([min(self.values), max(self.
72             values)])
73         return
74
75     def __verify_next_value(self, next_value):
76         self.predictive_error = abs(self.next_value - next_value)
77         self.predicted_values.append(self.next_value)
78         self.errors.append(self.predictive_error)
```

A.4 Mecanismos Preditores

```
79         if(len(self.confidence_bands) == 0):
80             return True
81         if(next_value >= self.confidence_bands[-1][0]) and (
82             next_value <= self.confidence_bands[-1][1]):
83             self.errorCount = 0
84             return True
85
86         self.errorCount = self.errorCount + 1
87         if(self.errorCount >= self.errorThreshold):
88             return False
89         else:
90             return True
91
92     def get_error(self, next_value):
93         return (self.__verify_next_value(next_value), self.
94             predictive_error)
95
96     def getParametersDict(self):
97         return {"errorThreshold": self.errorThreshold, "windowSize":
98             self.windowSize}
```

Código Fonte A.8: Métodos adicionais e alterados no mecanismo *Exponential Smoothing* em relação ao mecanismo preditor base.

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  '''
4  @author: pisa
5  '''
6  from base_mechanism import BaseMechanism
7  from constants import Constants
8
9  class ExponentialSmoothing(BaseMechanism):
10     '''
11     The Exponential Smoothing predictive mechanism is based on:
12     Next_value → Exponential Smoothing [  $Y_{t+1} = \alpha * Y_t + (1 -$ 
13         alpha) * valor_previsto  $Y_t$  ]
```

A.4 Mecanismos Preditores

```
13         Confidence Band → Standard Deviation
14     '''
15
16     def __init__(self, constantsObj = Constants(), windowSize = None,
17                 alpha = None):
18         BaseMechanism.__init__(self, constantsObj, windowSize)
19         if alpha == None:
20             self.alpha = self.constantsObj.ES_ALPHA
21         else:
22             self.alpha = alpha
23         return
24
25     def calculate_next_value(self):
26         first_part = self.alpha * self.values[-1]
27         second_part = (1-self.alpha) * self.next_value
28         self.next_value = first_part + second_part
29         return
30
31     def calculate_confidence_band(self):
32         deviation = []
33         count = 0
34         for valueIndex in range(1, len(self.values)):
35             deviation.append(abs(float(self.values[valueIndex]) -
36                               self.next_value)**2.0)
37             count += 1
38         if count <= 0:
39             stdev = 0
40         else:
41             stdev = (sum(deviation) / (count+1))**(0.5)
42
43         self.confidence_bands.append([self.next_value - 2*stdev, self
44                                     .next_value + 2*stdev])
45     return
```

A.4 Mecanismos Preditores

Código Fonte A.9: Métodos adicionais e alterados no mecanismo *Holt Winters Seasonal* em relação ao mecanismo preditor base.

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 '''
4 @author: pisa
5 '''
6 from base_mechanism import BaseMechanism
7 from constants import Constants
8 import sys
9 import math
10 import random
11 from collections import deque
12
13 class HoltWintersForecasting(BaseMechanism):
14     '''
15     The mean predictive mechanism is based on:
16     Next_value → Holt-Winters Forecasting [ A[-1] + B[-1] + C[
17     len - m] ], assuming m < window size and m the seasonal
18     cycle in samples
19     A[t] = Alpha(Y[t] - C[len - m - 1]) + (1 - Alpha)(A[t-1]
20     + B[t-1])
21     B[t] = Beta(A[t] - A[t-1]) + (1 - Beta)(B[t-1])
22     C[t] = Gama(Y[t] - A[t]) + (1 - Gama)(C[t-m])
23     Confidence Band → Holt-Winters Forecasting
24     D[t] = Gama(|Y[t] -  $\hat{Y}[t]$ |) + (1 - Gama)(D[t - m])
25     Confidence Band = (Y[t] - Delta*D[t], Y[t] + Delta*D[t])
26     Parameters:
27     Alpha = ?
28     Beta = ?
29     Gama = ?
30     Delta = 2
31
32     window size = m + 1. Taking m to 1 day. window size =
33     240*24 + 1 = 5761
34 '''
```

A.4 Mecanismos Preditores

```
31         threshold → number of points out of the trend before
32             sending an alarm                 suggestion: 5
33     '''
34     def __init__(self, constantsObj = Constants(), windowSize = None,
35                 errorThreshold = None, alphaPercentil = None, betaPercentil =
36                 None, gamaPercentil = None):
37         '''
38         Constructor
39         '''
40
41         self.delta = 2#random.uniform(2,3)
42         self.initialThreshold = 5
43
44         BaseMechanism.__init__(self, constantsObj, windowSize,
45                               errorThreshold)
46
47         if alphaPercentil == None:
48             self.alphaPercentil = self.constantsObj.
49                 HW_ALPHA_PERCENTIL
50
51         else:
52             self.alphaPercentil = alphaPercentil
53
54         if betaPercentil == None:
55             self.betaPercentil = self.constantsObj.HW_BETA_PERCENTIL
56
57         else:
58             self.betaPercentil = betaPercentil
59
60         if gamaPercentil == None:
61             self.gamaPercentil = self.constantsObj.HW_GAMA_PERCENTIL
62
63         else:
64             self.gamaPercentil = gamaPercentil
65
66         self.m = self.windowSize - 2
67
68         self.aTemporal = deque()
69         self.bTemporal = deque()
70         self.cTemporal = deque()
```

A.4 Mecanismos Predictores

```
62         self.dTemporal = deque()
63
64         self.aTemporal.append(0)
65         self.aTemporal.append(0)
66         self.bTemporal.append(0)
67         return
68
69     def calculate_next_value(self):
70         #Calculate parameters
71         self.calculateA()
72         self.calculateB()
73         self.calculateC()
74         self.calculateD()
75
76         #Clean the temporalSeries
77         if (len(self.aTemporal) > self.windowSize):
78             self.aTemporal.popleft()
79             self.bTemporal.popleft()
80             self.cTemporal.popleft()
81             self.dTemporal.popleft()
82
83         self.next_value = self.aTemporal[-1] + self.bTemporal[-1]
84         try:
85             cValue = self.cTemporal[-self.m]
86         except IndexError:
87             cValue = 0
88         self.next_value = self.next_value + cValue
89         return
90
91     def calculate_confidence_band(self):
92         if (len(self.values) < self.initialThreshold):
93             super(HoltWintersForecasting, self).
94                 calculate_confidence_band()
95         else:
96             self.confidence_bands.append([self.next_value - self.
97                 delta*self.dTemporal[-1],
```

A.4 Mecanismos Preditores

```
96         self.next_value + self.delta*self
97             .dTemporal[-1]))
98
99
100     def getAlpha(self):
101         return self.alphaPercentil
102
103     def getBeta(self):
104         return self.betaPercentil
105
106     def getGama(self):
107         return self.gamaPercentil
108
109     def getDelta(self):
110         return self.delta
111
112     def calculateA(self):
113         """
114          $A[t] = \text{Alpha}(Y[t] + C[\text{len} - m - 1]) + (1 - \text{Alpha})(A[t-1] + B$ 
115              $[t-1])$ 
116         """
117         alpha = self.getAlpha()
118         cValue = 0
119         try:
120             cValue = self.cTemporal[-self.m-1]
121         except IndexError:
122             cValue = 0
123         first_part = (alpha)*(self.values[-1] - cValue)
124         second_part = (1 - alpha)*(self.aTemporal[-1] + self.
125             bTemporal[-1])
126         self.aTemporal.append(first_part + second_part)
127         return
128
129     def calculateB(self):
130         """
```

A.4 Mecanismos Preditores

```
129          $B[t] = \text{Beta}(A[t] - A[t-1]) + (1 - \text{Beta})(B[t-1])$ 
130         """
131         beta = self.getBeta()
132         first_part = (beta)*(self.aTemporal[-1] - self.aTemporal[-2])
133         second_part = (1 - beta)*(self.bTemporal[-1])
134         self.bTemporal.append(first_part + second_part)
135         return
136
137     def calculateC(self):
138         """
139          $C[t] = \text{Gama}(Y[t] + A[t]) + (1 - \text{Gama})(C[t-m])$ 
140         """
141         gama = self.getGama()
142         cValue = 0
143         try:
144             cValue = self.cTemporal[-self.m]
145         except IndexError:
146             cValue = 0
147             gama = 1
148
149         first_part = (gama)*(self.values[-1] - self.aTemporal[-1])
150         second_part = (1 - gama)*(cValue)
151         self.cTemporal.append(first_part + second_part)
152         return
153
154     def calculateD(self):
155         """
156          $D[t] = \text{Gama}(|Y[t] - \hat{Y}[t]|) + (1 - \text{Gama})(D[t - m])$ 
157         """
158         gama = self.getGama()
159         dValue = 0
160         try:
161             dValue = self.dTemporal[-self.m]
162         except IndexError:
163             dValue = 0
164             gama = 1
```

A.4 Mecanismos Preditores

```
165
166     first_part = (gama)*(abs(self.values[-1] - self.next_value))
167     second_part = (1 - gama)*(dValue)
168     self.dTemporal.append(first_part + second_part)
169     return
```