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

RAUL CERETTA NUNES

Adaptação Dinâmica do *timeout* de Detectores de Defeitos através do Uso de Séries Temporais

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

Profa. Dra. Ingrid Jansch-Pôrto
Orientadora

Porto Alegre, setembro de 2003

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Nunes, Raul Ceretta

Adaptação Dinâmica do *timeout* de Detectores de Defeitos através do Uso de Séries Temporais / Raul Ceretta Nunes. – Porto Alegre: Programa de Pós-Graduação em Computação, 2003.

164 f.: il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2003. Orientadora: Ingrid Jansch-Pôrto.

1. Tolerância a Falhas. 2. Detectores de Defeitos. 3. Séries Temporais. 4. Preditores. 5. Atrasos de Comunicação. I. Jansch-Pôrto, Ingrid. II. Título.

"Aos meus pais Inocêncio e Irma."

AGRADECIMENTOS

O apoio, o carinho, a orientação, o amor e o incentivo de várias pessoas foram fundamentais na realização deste trabalho. Por isto agradeço:

à Ingrid, minha orientadora, por suas sensatas e fundamentais colocações, as quais me servirão para o resto da vida na orientação pessoal e profissional;

ao Juergen, Taisy, Lúcia Lisboa e Rafael Rocha, os quais foram importantes incentivadores no processo de construção desta tese, além de sempre demonstrarem boa vontade e disposição para contribuir;

ao Fernando Boeira (Estatística-UFRGS) e ao Luis Felipe (Estatística-UFSM), pelo valioso auxílio na compreensão dos conceitos fundamentais da teoria de séries temporais. Ao Luis Felipe também agradeço por sua disposição para me ajudar na elaboração de alguns testes estatísticos;

aos amigos e colegas, Adenauer, Cadinho, Cechin, Emerson, Hércules, João Baptista, Jugurta, Marcia, Patrícia, Pilla, Rogério e Ronaldo que, de perto, estiveram presentes me dando forças para vencer as fases difíceis deste curso, mas que acima de tudo compõem a força que nos dá o sentido da vida;

aos colegas de Departamento, pelo importante auxílio concedido nos momentos em que tive que conciliar atividades docentes com atividades discentes;

aos meus amigos Alexandre, Maria, André, Raquel, Daniel, Rochele, Mauro, Vera, Ronaldo e Marilene que, em Santa Maria ou Porto Alegre, sempre estiveram presentes no coração da minha familia na forma do combustível indispensável para se ter uma vida saudável e alegre;

à Anilza, Vaneza, Wilson e Arselle, pela permanente torcida para que tudo desse sempre certo. A Anilza também agradeço pela dedicação e carinho ofertados de diversas formas;

aos meus pais Inocêncio e Irma que, com muito amor, carinho e fé, sabiamente me ensinaram a conduzir a vida, da qual faz parte este trabalho; e

à Viviany, Rauani e Rafaela ... faltam palavras. De forma breve, em poucas palavras, agradeço pela energia positiva emanada no convívio diário, pelo constante apoio, carinho e amor, e pela compreensão e paciência que tiveram nos momentos em que precisei sacrificar horários de lazer para realizar tarefas profissionais.

A realização do curso também teve apoio institucional sobre o qual agradeço:

à UFSM, por conceder-me afastamento integral das minha atividades didáticas sem prejuízo aos meus proventos;

à CAPES, por conceder-me uma bolsa de estudos, a qual foi imprescindível para viabilizar a realização desta tese; e

à UFCG, UFRGS, UFSM e ao Ponto de Presença da RNP no estado do Pará, por possibilitarem a realização da coleta de dados necessária ao desenvolvimento deste trabalho.

Por fim, mas nem por isto de menor apreço, agradeço aos demais que me ajudaram direta ou indiretamente na realização deste trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 9

LISTA DE FIGURAS 11

LISTA DE TABELAS 14

RESUMO 15

ABSTRACT 16

1 INTRODUÇÃO 17
1.1 Sistemas tolerantes a falhas e detectores de defeitos 18
1.2 Detectores de defeitos e o uso de *timeouts* 18
1.3 Estratégias para ajustar o *timeout* 20
1.4 Definição do problema 21
1.5 Contribuições deste trabalho 22
1.6 Organização do texto 22

2 SOBRE O AJUSTE DO *TIMEOUT* 24
2.1 Modelo de Sistema Distribuído 24
2.2 O detector de defeitos 25
2.2.1 *Timeout* no detector do estilo *push* 27
2.2.2 *Timeout* no detector do estilo *pull* 28
2.3 Estratégias para Ajuste do *timeout* 29
2.3.1 *Timeout* em protocolos de *heartbeat* acelerados 29
2.3.2 *Timeout* no serviço de grupo Jgroup 31
2.3.3 *Timeout* no Object Group Service - OGS 32
2.3.4 *Timeout* no trabalho de pesquisa de Chen e Toueg 32
2.3.5 Indicador do Tempo de Conectividade - CTI 33
2.3.6 Timeout no TCP 34
2.3.7 Outros trabalhos relacionados 35
2.4 Conclusões Parciais 36

3 SÉRIES TEMPORAIS 39
3.1 Conceitos básicos 39
3.1.1 Definição 39
3.1.2 Estacionariedade e ergodicidade 39
3.2 Modelos Matemáticos para Séries Temporais 40
3.2.1 Modelos de erro ou de regressão 40
3.2.2 Modelos ARIMA 40
3.2.3 Classificação de acordo com o número de parâmetros envolvidos . . . 42
3.3 Metodologia de previsão 42

3.3.1 Método automático . . . 43
3.3.2 Método interativo . . . 43
3.3.3 Algoritmo geral de previsão . . . 43
3.4 Conclusões Parciais . . . 44

4 REPRESENTAÇÃO E CARACTERÍSTICAS DO ATRASO DE COMUNICAÇÃO . . . 45
4.1 Modelagem do atraso de comunicação de ida e volta como uma série temporal . . . 46
4.1.1 Modelo discreto do *rtt* . . . 46
4.1.2 A interface para construir uma série de tempo . . . 49
4.1.3 Descrição formal da interface . . . 52
4.2 Análise da estacionariedade do *rtt* . . . 54
4.2.1 Coleta de dados . . . 55
4.2.2 Metodologia da análise . . . 56
4.2.3 Resultados da análise . . . 60
4.3 Conclusões Parciais . . . 65

5 MODELOS DE PREDIÇÃO DO *RTT* . . . 67
5.1 Especificação dos Preditores . . . 67
5.1.1 Preditor LAST . . . 68
5.1.2 Preditor MEAN . . . 68
5.1.3 Preditor WINMEAN . . . 68
5.1.4 Preditor LPF . . . 68
5.1.5 Preditor DMA . . . 69
5.1.6 Preditor BROWN . . . 71
5.1.7 Preditor ARIMA . . . 71
5.2 Configuração do preditor ARIMA . . . 74
5.2.1 Período de identificação/ajuste do modelo ARIMA . . . 74
5.2.2 Tamanho da série temporal . . . 76
5.2.3 Identificação automática do modelo ARIMA . . . 76
5.3 Métrica de comparação . . . 79
5.4 Metodologia de análise . . . 79
5.5 Análise dos preditores com padrões de amostragens . . . 82
5.5.1 Definição dos padrões . . . 82
5.5.2 Análise da precisão dos preditores . . . 84
5.5.3 Análise comportamental das curvas de previsão . . . 85
5.6 Análise dos preditores com amostragens reais . . . 89
5.6.1 Análise das amostragens com 86400 amostras . . . 90
5.6.2 Análise das amostragens com 64800 amostras . . . 94
5.6.3 Análise das amostragens com tamanhos variados . . . 95
5.7 Conclusões Parciais . . . 97

6 INFLUÊNCIA DO PREDITOR NA QUALIDADE DO DETECTOR DE DEFEITOS . . . 99
6.1 Métricas de QoS . . . 99
6.2 Ambiente de simulação . . . 102
6.3 Computação das métricas no simulador . . . 102
6.3.1 Computação dos instantes de transição . . . 102

6.3.2 Computação das métricas de QoS 104
6.4 Margem de segurança 106
6.4.1 Separando a margem em dois fatores 107
6.4.2 Margem variável de acordo com o erro médio da previsão 108
6.4.3 Margem variável em função do intervalo de confiança da previsão . . . 108
6.5 Relação das métricas de QoS com o *timeout* 109
6.6 Análise da qualidade de serviço 110
6.6.1 QoS em padrões de amostragens 111
6.6.2 QoS com amostragens reais 128
6.7 Conclusões Parciais 137

7 UM SERVIÇO DE DETECÇÃO DE DEFEITOS ADAPTATIVO . 139
7.1 Detector de defeitos como um serviço 139
7.2 Arquitetura configurável e flexível para um serviço de detecção 141
7.2.1 Dissociando algoritmos de detecção e de previsão 142
7.2.2 Novas interfaces orientadas à aplicação 143
7.2.3 Flexibilizando a arquitetura do serviço 145
7.3 AFDService - um serviço de detecção de defeitos adaptativo . 148
7.3.1 Estratégia global de adaptação do *timeout* 148
7.3.2 O módulo de detecção de defeitos 151
7.3.3 O módulo de previsão 153
7.4 Conclusões Parciais 155

8 CONCLUSÃO 156

REFERÊNCIAS 159

LISTA DE ABREVIATURAS E SIGLAS

ANOVA	Análise da variância - método estatístico, baseado na análise de variações amostrais, que serve para testar a igualdade de três ou mais médias populacionais
ARIMA	*Autoregressive Integrated Moving Average* - preditor para séries temporais não estacionárias, o qual modela a série como um processo estocástico auto-regressivo integrado de médias móveis
ATM	*Asynchronous Transfer Mode*
BROWN	Preditor para séries temporais não estacionárias, o qual aplica o modelo de alisamento exponencial duplo de Brown
CTI	Indicador do Tempo de Conectividade
DMA	*Double Moving Average* - preditor para séries temporais não estacionárias, o qual aplica o modelo de médias móveis de segunda ordem
Duncan	Teste de Duncan - método estatístico, baseado na análise de variações amostrais, que serve para comparar a igualdade entre múltiplas médias. Como resultado o teste classifica as médias em grupos.
ep	Margem de segurança variável computada em função do erro de previsão
FD	*Failure Detector*
ic	Margem de segurança variável computada em função do intervalo de confiança da previsão
Jgroup	Sistema de Comunicação de Grupo Confiável da Universidade de Bolonha (Itália). Programado em Java
LAN	*Local Area Network*
LAST	Preditor baseado somente no último valor amostrado
LPF	*Low Pass Filter* - preditor baseado num filtro passa baixa
MEAN	Preditor baseado na média aritmética de todas as amostras
OGS	*Object Group Service*
P_A	Probabilidade de uma resposta correta
RFC	*Request For Comments*
RPS	*Resource Prediction System*
rtt	*round-trip time*
TCP	*Transfer Control Protocol*

T_D	Tempo médio para detecção de um defeito
T_D^u	Tempo máximo para detecção de um defeito
t_i	Período de interrogação de estado num detector do estilo *pull*. No texto também é utilizado como período de envio de mensagens de estado num detector do estilo *push*
T_M	Tempo médio de duração de um defeito
T_{MR}	Tempo médio de recorrência ao erro
t_o	Tempo máximo de espera por uma dada mensagem (*timeout*)
UDP	*Universal Datagram Protocol*
WAN	*Wide Area Network*
WINMEAN	*Window Mean* - preditor baseado na média aritmética dos últimos 30 valores amostrados
α	Constante de alisamento
Δ	Margem de segurança global (inclui atrasos extras e perdas de mensagens)
β	Fator da margem de segurança Δ para sobrepor atrasos extras
δ	Fator margem de segurança Δ para sobrepor perdas de mensagens
λ	Taxa de erros

LISTA DE FIGURAS

Figura 4.1: Modelo de comunicação do detector de defeitos. 46
Figura 4.2: Amostragem dos atrasos de comunicação. 47
Figura 4.3: Modelo teórico para um atraso de comunicação. 48
Figura 4.4: Possíveis cenários num período estável. 49
Figura 4.5: Possíveis cenários de um período instável. 50
Figura 4.6: Construção da série temporal através de uma interface que converte uma amostragem aperiódica em periódica. 53
Figura 4.7: Algoritmo que implementa a interface. 53
Figura 4.8: Distribuição geográfica dos nós utilizados na coleta de dados. . . . 56
Figura 4.9: Algoritmo para determinar a estacionariedade e a ordem de integração. 59
Figura 4.10: Algoritmo para determinar se acf declina muito lentamente. . . . 59
Figura 4.11: Séries temporais para amostragens do rtt durante 24 horas. . . . 61
Figura 4.12: Séries temporais para amostragens do rtt durante uma hora (12-13h). 61
Figura 4.13: Função de autocorrelação amostral típica para séries não estacionárias. 62
Figura 4.14: Funções de autocorrelação amostral para séries diárias estacionárias. 62
Figura 4.15: Distribuição percentual estacionário `x` não-estacionário por hora do dia considerando séries de uma hora. 64
Figura 4.16: Relação estacionário `x` não-estacionário considerando para diferentes tamanhos de série temporal. 64
Figura 4.17: Relação estacionário `x` não-estacionário por canal de comunicação. 65

Figura 5.1: Algoritmo de previsão embasado no preditor ARIMA. 73
Figura 5.2: Erro quadrático médio de previsão para diferentes tamanhos de série temporal. 77
Figura 5.3: Padrões de amostragem fictícios. 83
Figura 5.4: Comportamento dos preditores num período aleatório de baixa variância (Padrão 1). 86
Figura 5.5: Comportamento dos preditores num período aleatório de alta variância (Padrão 3). 86
Figura 5.6: Comportamento dos preditores num período com alternância de baixa variância (Padrão 2). 87
Figura 5.7: Comportamento dos preditores num período com alternância de alta variância (Padrão 4). 87
Figura 5.8: Comportamento dos preditores num período que alterna a tendência a cada 15 amostras (Padrão 5). 88
Figura 5.9: Comportamento dos preditores num período que alterna o nível a cada 15 amostras (Padrão 6). 89

Figura 5.10: Comportamento dos preditores num período que alterna a tendência e nível de acordo com um padrão senoidal (Padrão 7). . . . 90
Figura 5.11: Curva de previsão do preditor MEAN numa quarta-feira. 91
Figura 5.12: Curva de previsão dos preditores LAST, MEAN, WINMEAN e DMA em situações de alta e baixa variância do *rtt*. 92
Figura 5.13: Curva de previsão dos preditores DMA, BROWN, ARIMA e LPF em amostragens de 86400 amostras sob situações de alta e baixa variância. 93
Figura 5.14: Curva de previsão dos preditores DMA, BROWN, ARIMA e LPF em amostragens de 64800 amostras sob situações de alta e baixa variância. 95
Figura 5.15: Curvas de previsão dos preditores DMA, BROWN, ARIMA e LPF na comunicação UFRGS-POP/PA (20/jun/2001) sob dois instantes de inicialização. 96
Figura 5.16: Erro quadrático médio para amostragens de tamanhos diferentes. 97

Figura 6.1: Algoritmo que computa o histórico de transições de estado. 104
Figura 6.2: Comportamento dos detectores LAST e ARIMA num período de alternância com baixa variância (padrão 2). 113
Figura 6.3: Comportamento dos detectores BROWN e MEAN num período com alternância de nível (padrão 6). 114
Figura 6.4: Comportamento do *timeout* no padrão 4. 117
Figura 6.5: Comportamento do *timeout* no padrão 5. 117
Figura 6.6: Comportamento do *timeout* no padrão 6. 117
Figura 6.7: Comportamento do *timeout* no padrão 7. 118
Figura 6.8: Métricas T_M e T_{MR} para o padrão 2 com diferentes margens de segurança. 119
Figura 6.9: Métricas T_M e T_{MR} para o padrão 4 com diferentes margens de segurança. 120
Figura 6.10: Métricas T_M e T_{MR} para o padrão 1 com diferentes margens de segurança. 121
Figura 6.11: Métricas T_M e T_{MR} para o padrão 3 com diferentes margens de segurança. 121
Figura 6.12: Métricas T_M e T_{MR} para o padrão 5 com diferentes margens de segurança. 123
Figura 6.13: Métricas T_M e T_{MR} para o padrão 6 com diferentes margens de segurança. 125
Figura 6.14: Curva de previsão para o ARIMA e LAST no padrão 7. 126
Figura 6.15: Métricas T_M e T_{MR} para o padrão 7 com diferentes margens de segurança fixas. 127

Figura 7.1: Diagrama de classes das interfaces do serviço de monitoramento de objetos utilizado por Felber et al. (1999). 140
Figura 7.2: Arquitetura dos módulos de detecção e previsão. 142
Figura 7.3: Arquitetura de interfaces para um serviço de detecção configurável. 144
Figura 7.4: Interface abstrata `Predictor`. 146
Figura 7.5: Estrutura do padrão de projeto *Strategy*. 146
Figura 7.6: Arquitetura versátil para o módulo de detecção. 147
Figura 7.7: Arquitetura versátil para o módulo de previsão. 147

Figura 7.8: Diagrama de classes do módulo de detecção. 152
Figura 7.9: Diagrama de classes do módulo de predição. 154

Figura 8.1: Sugestão de interface para construção de uma série temporal dos atrasos de comunicação de mensagens de *heartbeat*. 158

LISTA DE TABELAS

Tabela 2.1: Classes de detecção de defeitos definidas em função das propriedades de abrangência e precisão. 26

Tabela 4.1: Amostragens diárias coletadas. 57

Tabela 5.1: Erro quadrático médio dos preditores por padrão do *rtt*. 84
Tabela 5.2: Erro quadrático médio por preditor. 91
Tabela 5.3: Resultado do teste de Duncan a 5% de significância para amostragens com 86400 amostras. 91
Tabela 5.4: Resultado do teste de Duncan a 5% de significância para amostragens com 64800 amostras. 94
Tabela 5.5: Resumo dos resultados do teste de Duncan a 5% de significância. 97

Tabela 6.1: Tempo médio (em segundos) para detecção de um defeito do tipo colapso sem o uso de margem de segurança. 112
Tabela 6.2: Limite superior do tempo (em segundos) para detecção de um defeito do tipo colapso sem o uso de margem de segurança. 113
Tabela 6.3: Tempos para detecção de um defeito no padrão 7 para diferentes margens de segurança fixa. 115
Tabela 6.4: Tempos para detecção de um defeito para margens de segurança variáveis. 116
Tabela 6.5: Precisão dos detectores para amostragens com alternância adjacente regular quando a margem de segurança é nula. 119
Tabela 6.6: Precisão dos detectores para a amostragem aleatória do padrão 1. 120
Tabela 6.7: Precisão dos detectores para a amostragem aleatória do padrão 3. 121
Tabela 6.8: Precisão dos detectores para o padrão 5 quando a margem de segurança é nula. 122
Tabela 6.9: Precisão dos detectores para o padrão 6. 124
Tabela 6.10: Precisão dos detectores para o padrão 7. 126
Tabela 6.11: T_D e T_D^u em amostragens de 24 horas. 129
Tabela 6.12: T_D e T_D^u em amostragens de 18 horas. 130
Tabela 6.13: Métricas de precisão para amostragens com 24 horas. 132
Tabela 6.14: Métricas de precisão para amostragens com 18 horas. 132
Tabela 6.15: T_D e T_D^u em amostragens de pequena duração. 134
Tabela 6.16: Métricas de precisão para amostragens de pequena duração. . . . 135

RESUMO

Uma aplicação distribuída freqüentemente tem que ser especificada e implementada para executar sobre uma rede de longa distância (*wide-area network* - WAN), tipicamente a Internet. Neste ambiente, tais aplicações são sujeitas a defeitos do tipo colapso (falha geral num dado nó), temporização (flutuações na latência de comunicação) e omissão (perdas de mensagens). Para evitar que estes defeitos gerem conseqüências indesejáveis e irreparáveis na aplicação, explora-se técnicas para tolerá-los. A abstração de detectores de defeitos não confiáveis auxilia a especificação e trato de algoritmos distribuídos utilizados em sistemas tolerantes a falhas, pois permite uma modelagem baseada na noção de estado (suspeito ou não suspeito) dos componentes (objetos, processos ou processadores) da aplicação. Para garantir sua terminação, os algoritmos de detecção de defeitos costumam utilizar a noção de limites de tempo de espera (*timeout*). Adicionalmente, para minimizar seu erro (falsas suspeitas) e não comprometer seu desempenho (tempo para detecção de um defeito), alguns detectores de defeitos ajustam dinamicamente o *timeout* com base em previsões do atraso de comunicação. Esta tese explora o ajuste dinâmico do *timeout* realizado de acordo com métodos de previsão baseados na teoria de séries temporais. Tais métodos supõem uma amostragem periódica e fornecem estimativas relativamente confiáveis do comportamento futuro da variável aleatória.

Neste trabalho é especificado uma interface para transformar uma amostragem aperiódica do atraso de ida e volta de uma mensagem (rtt) numa amostragem periódica, é analisado o comportamento de séries reais do rtt e a precisão de sete preditores distintos (três baseados em séries temporais e quatro não), e é avaliado a influência destes preditores na qualidade de serviço de um detector de defeitos do estilo *pull*. Uma arquitetura orientada a objetos que possibilita a escolha/troca de algoritmos de previsão e de margem de segurança é também proposta.

Como resultado, esta tese mostra: (i) que embora a amostragem do rtt seja aperiódica, pode-se modelá-la como sendo uma série temporal (uma amostragem periódica) aplicando uma interface de transformação; (ii) que a série temporal rtt é não estacionária na maioria dos casos de teste, contradizendo a maioria das hipóteses comumente consideradas em detectores de defeitos; (iii) que dentre sete modelos de predição, o modelo ARIMA (*autoregressive integrated moving-average model*) é o que oferece a melhor precisão na predição de atrasos de comunicação, em termos do erro quadrático médio; (iv) que o impacto de preditores baseados em séries temporais na qualidade de serviço do detector de defeitos não é significativo em relação a modelos bem mais simples, mas varia dependendo da margem de segurança adotada; e (v) que um serviço de detecção de defeitos pode possibilitar a fácil escolha de algoritmos de previsão e de margens de segurança, pois o preditor pode ser modelado como sendo um módulo dissociado do detector.

Palavras-chave: Tolerância a Falhas, Detectores de Defeitos, Séries Temporais, Preditores, Atrasos de Comunicação.

ABSTRACT

Often a distributed application must be specified and implemented to run in a wide-area network (WAN), typically over the Internet. In this environment, it is well known that such applications are subject to crash failures (a component stops to work forever), temporization failures (fluctuations happen in the latency of communication) and omission failures (messages can be lost). To minimize the failure impact on application, fault-tolerant systems explore techniques to tolerate failures. A failure detector is an important abstraction to solve fundamental problems in fault-tolerant distributed computing. It allows a modeling based on the notion of state (suspect or not suspect). Encapsulating the notion of time, the failure detector algorithms commonly use timeouts to ensure its termination property. Thus, to minimize its error (wrong suspicion) and do not get poor performance (detection time), some detectors dynamically adjust their timeout based on forecast of the communication delay.

This thesis explores the use of time series based predictors in the dynamic adjustment of the timeout of failure detectors. An efficient time series predictor considers a periodic sampling and offers high quality predictions. To explore these predictors, this work: specifies a lightweight interface to transform a non-periodic sampling of the round-trip communication delay (rtt) in a periodic sampling (a time series); it analyses the behavior of the real rtt time series; it analyses the accuracy of seven predictors (three based on time series); and it evaluates the influence of these predictors in the quality of service of a *pull*-style failure detector. It also proposes a flexible object-oriented architecture where prediction algorithms, or safety margin algorithms, can be interchanged.

As result, this thesis shows that: (i) despite non-periodic sampling, the rtt can be modeled as a time series by applying a transformation interface; (ii) the resultant time series is not stationary in most of test cases, contradicting the common assumptions used by failure detector; (iii) among seven prediction models, the model ARIMA (*autoregressive integrated moving-average model*) offers the best precision, in terms of the mean quadratic error; (iv) the predictors based on time series do not result in a significant increase of the quality of service of failure detectors, if compared with common used models, but depending on the adopted safety margin they can get better results; and (v) by separating the predictor's and detector's semantics a failure detection service can make it possible an easy choice of prediction and safety margins algorithms.

Keywords: Fault Tolerance, Failure Detectors, Time Series, Predictors, Communication Delay.

1 INTRODUÇÃO

"*Um sistema distribuído é um sistema no qual um defeito de um computador cuja existência você ignorava, pode tornar seu computador inutilizável.*"[1] (LAMPORT, 1987). Com esta manifestação, há muito tempo, Leslie Lamport indicava a fragilidade de um sistema distribuído: as incertezas sobre o comportamento dos componentes que compõem o sistema, tais como nós e canais de comunicação, tornam uma aplicação distribuída suscetível a defeitos. No contexto deste trabalho, um defeito é a percepção do erro gerado por uma falha.

Na prática, uma falha e a conseqüente indisponibilidade de um serviço distribuído pode provocar catástrofes ou gerar grandes prejuízos financeiros. Por isto, o interesse por computação distribuída robusta tem crescido significativamente nos últimos anos, principalmente em aplicações implementadas sobre redes de longa distância (*wide-area networks* - WANs), tal como a Internet, onde os atrasos de comunicação não são previsíveis (BAUER; SICHITIU; PREMARATNE, 2001).

Para evitar que uma falha gere conseqüências indesejáveis ou irreparáveis, exploram-se técnicas para tolerá-la, ou seja, técnicas que fazem com que os erros provocados como resultado desta falha, no hardware ou no software, não afetem o funcionamento global do sistema (PRADHAN, 1996). Um sistema que, para poder continuar resolvendo corretamente suas tarefas previamente especificadas, faz uso de tais técnicas é chamado *sistema tolerante a falhas*. Entretanto, para implementar técnicas de tolerância a falhas, o uso de *redundância* é necessário (GARTNER, 1999).

Num sistema distribuído, onde os nós podem ser tipicamente considerados independentes do ponto de vista de falhas, a distribuição tem sido um *framework* usual para prover tolerância a falhas. Entretanto, a distribuição do estado da aplicação e a característica não confiável da comunicação entre os nós envolvidos motiva a exploração de técnicas de tolerância a falhas para sistemas distribuídos. Uma importante característica destas técnicas é que o processamento do erro e o tratamento da falha são implementadas principalmente por software via algoritmos distribuídos baseados em troca de mensagens (POWELL, 1998).

Neste contexto, uma importante abstração para a construção de sistemas distribuídos tolerantes a falhas é o que se chama de detector de defeitos não confiável (CHANDRA; TOUEG, 1996). Tal abstração procura encapsular o indeterminismo do atraso de comunicação entre duas entidades distribuídas e é normalmente implementada com o uso de limites de tempo de espera, ou *timeouts*, como são conhecidos. Esta tese explora o uso de séries temporais (BOWERMAN; O'CONNEL, 1993), um modelo matemático para previsão de valores, como alternativa para a determinação dinâmica dos *timeouts* de detectores de defeitos.

Inicialmente, para contextualizar este trabalho, discutem-se as relações entre

[1] Literalmente, "*A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.*".

os sistemas tolerantes a falhas e os detectors de defeitos (seção 1.1), as relações entre os detectores de defeitos e o uso de *timeouts* (seção 1.2) e as relações entre as diferentes estratégias que vêm sendo aplicadas para ajustar o *timeout* dos detectores de defeitos (seção 1.3). Então, define-se o problema atacado nesta tese (seção 1.4) e salienta-se as contribuições deste trabalho (seção 1.5). A seção 1.6 apresenta a estrutura do restante do texto.

1.1 Sistemas tolerantes a falhas e detectores de defeitos

Em sistemas distribuídos, a redundância natural do hardware (nós do sistemas) costuma ser explorada por técnicas de tolerância a falhas para implementar redundância de software na forma de replicação de componentes críticos (JALOTE, 1994). A replicação torna o sistema mais robusto (confiável + disponível) do que um sistema centralizado, mas exige a resolução de *protocolos de acordo* para manter a consistência das informações replicadas (GUERRAOUI; SCHIPER, 1997).

A característica aleatória do atraso de comunicação entre os componentes de um sistema distribuído (BAUER; SICHITIU; PREMARATNE, 2001) aliada à possibilidade de percepção de defeitos destes componentes, torna a resolução do acordo uma tarefa não trivial, pois não é possível distinguir um defeito gerado por uma falha real num dado componente replicado, de um defeito gerado por uma falha no meio de comunicação, ou mesmo de um gerado por uma sobrecarga em algum ponto do sistema. Como conseqüência, o consenso, um bloco de construção básico para protocolos de acordo (GUERRAOUI; SCHIPER, 1996), não pode ser resolvido deterministicamente (FISCHER; LYNCH; PATERSON, 1985).

Para sobrepor o problema do indeterminismo, as especificações de protocolos de acordo para sistemas distribuídos tolerantes a falhas não costumam adotar como hipótese um modelo assíncrono puro, mas sim um modelo: síncrono (DOLEV; DWORK; STOCKMEYER, 1987), parcialmente síncrono (DWORK; LYNCH; STOCKMEYER, 1988), assíncrono temporizado (CRISTIAN; FETZER, 1997), ou assíncrono com detectores de defeitos não confiáveis (CHANDRA; TOUEG, 1996).

Num modelo assíncrono com detectores de defeitos não confiáveis, a abstração de um detector de defeitos encapsula o problema do indeterminismo do atraso de comunicação. O encapsulamento possibilita a especificação de algoritmos de acordo baseada na noção de estado (um processo pode ser *confiável* ou *suspeito*), facilitando a validação do algoritmo. Com a noção de estado, a terminação de um protocolo de acordo pode ser garantida quando existir uma maioria de processos confiáveis (CHANDRA; TOUEG, 1996). As propriedades fornecidas pelo encapsulamento e pela noção de estado tornam o modelo de sistema distribuído assíncrono acrescido de um detector de defeitos um *framework* bem aceito no projeto de sistemas distribuídos tolerantes a falhas baseados em réplicas (GUERRAOUI; SCHIPER, 1997).

1.2 Detectores de defeitos e o uso de *timeout*s

Um detector de defeitos FD é um algoritmo distribuído composto por um conjunto de módulos de detecção de defeitos fd_i, cada um anexado a um componente de um sistema distribuído sob monitoramento. Cada módulo local fd_i monitora o estado de um subconjunto de componentes do sistema e mantém a informação de estado. Num dado instante t, as informações contidas nos módulos locais podem

diferir, pois adota-se como hipótese um sistema distribuído parcialmente síncrono. Baseado nas informações dos módulos locais, o detector de defeitos FD provê informações sobre o estado funcional dos componentes monitorados.

De acordo com Chandra e Toueg (1996), detectores de defeitos podem ser especificados através de duas propriedades axiomáticas: abrangência (*completeness*) e precisão (*accuracy*). Informalmente, a propriedade de abrangência diz respeito à capacidade do detector para suspeitar dos processos que tenham realmente falhado, enquanto que a propriedade de precisão diz respeito à capacidade do detector para evitar erros. Enquanto a propriedade de abrangência pode ser facilmente atendida, mesmo em sistemas assíncronos, a de precisão só pode ser atendida em sistemas parcialmente síncronos (LARREA; ARÉVALO; FERNÁNDEZ, 1999), ou seja, em sistemas onde se adota como hipótese a existência de limites de tempo de comunicação e de processamento. Chandra e Toueg adotam um modelo onde estes limites não são previamente conhecidos, entretanto supõem que eles serão conhecidos após um tempo finito mas desconhecido, chamado tempo de estabilização global (GST - *Global Stabilization Time*).

Do ponto de vista da especificação de algoritmos de mais alto nível (aplicação), as propriedades axiomáticas de um detector de defeitos, dadas pela classe a que pertence o detector, garantem comportamentos bem definidos que auxiliam nesta atividade. Entretanto, o problema do indeterminismo persiste, só que agora ele está encapsulado no detector, justificando o termo "não confiável" em sua denominação.

Assumindo por hipótese um sistema parcialmente síncrono, na prática, a implementação de um detector de defeitos não confiável pode garantir a terminação do algoritmo, através do uso de *timeouts* para progredir em suas atividades (CHANDRA; TOUEG, 1996). Entretanto, o ajuste inadequado deste valor de tempo pode produzir um número maior de erros.

Devido à característica "não confiável" do detector de defeitos, os algoritmos distribuídos que fazem uso desta abstração costumam ser projetados para tolerar erros do detector (falsas suspeitas). Uma falsa suspeita referente a algum membro correto normalmente desencadeia um processo indesejável na aplicação, tal como a exclusão do membro correto ou a execução de rodadas adicionais desnecessárias. Por isto, deve ser evitada.

Num detector de defeitos baseado em *timeouts*, para evitar erros pode-se aumentar a magnitude do *timeout*. Entretanto, esta ação aumenta o tempo para detecção de um defeito real, podendo não interessar à aplicação. Logo, a qualidade de um detector de defeitos deve ser medida tanto pela sua capacidade de evitar a ocorrência de falsas suspeitas quanto pela sua capacidade de detectar um defeito real no menor tempo possível. Neste sentido, Chen, Toueg e Aguilera (2002) propuseram um conjunto de métricas de qualidade de serviço para detectores de defeitos as quais auxiliam tanto no projeto quanto na escolha de um detector de defeitos.

Na prática, a tentativa de minimizar os erros do detector a um custo (desempenho) satisfatório, costuma refletir na forma usada para garantir a terminação do algoritmo de detecção, independentemente da estratégia de detecção de defeitos. Como na maioria dos casos a terminação é garantida via *timeout*, o ajuste deste é um ponto crítico no projeto de detectores de defeitos.

Embora detectores de defeitos possam ser projetados sem o uso de *timeout* como por exemplo o detector Heartbeat (AGUILERA; CHEN; TOUEG, 1997), que se baseia na quantidade de mensagens recebidas, neste trabalho, considera-se

somente a classe de detectores de defeitos baseados em *timeouts*.

1.3 Estratégias para ajustar o *timeout*

Para atender propriedades axiomáticas ou requisitos de qualidade de serviço em detectores de defeitos baseados em *timeouts*, distintas formas de ajustes do *timeout* foram propostas na literatura, sendo os dois principais enfoques: o ajuste *off-line*, realizado com base numa análise criteriosa do ambiente de execução (RAYNAL; TRONEL, 1999; RENESSE; MINSKY; HAYDEN, 1998), e o ajuste em tempo de execução (*on-line*), realizado de acordo com o comportamento dinâmico do sistema (MONTRESOR, 2000; MACÊDO, 1998, 2000; CHEN, 2000; BERTIER; MARIN; SENS, 2002).

Foram encontradas as seguintes estratégias de ajuste em tempo de execução:

(A) fazer um ajuste *off-line* do *timeout* (fixo) e, em tempo execução, incrementá-lo com um valor constante k cada vez que for detectada uma suspeita indevida (MONTRESOR, 2000) – a idéia fundamental desta estratégia é garantir, na prática, as propriedades axiomáticas do detector (especificadas em termos de comportamentos que serão observáveis em tempos finitos - *eventual behavior*). Desta forma, o uso de um *timeout* garante que um defeito tipo colapso (*crash*) sempre será detectado num tempo finito e, sempre que for detectada uma falsa suspeita, o uso de um incremento garante que todo processo correto não será mais erroneamente suspeito, num tempo finito;

(B) fazer o *timeout* variar de acordo com o tempo de ida e volta de uma mensagem[2] (*round-trip time - rtt*) (FELBER, 2000) – a idéia fundamental desta estratégia é fazer o *timeout* variar de acordo com o comportamento da rede de comunicação;

(C) ajustar o *timeout* com base em duas componentes: o tempo de comunicação fornecido por um módulo indicador do tempo de comunicação (ITC) (MACÊDO, 1998), e a constante k, incrementada sempre que for detectado uma suspeita indevida (MACÊDO, 1998, 2000) – a idéia fundamental desta estratégia é fazer o *timeout* variar de acordo com o comportamento da rede de comunicação, mas também garantir as propriedades formais do detector; e

(D) ajustar o *timeout* com base em duas componentes: o tempo estimado para a chegada da próxima mensagem de *heartbeat*[3], calculado por um módulo de previsão que estima o comportamento probabilístico de recepção das mensagens de *heartbeat*, e uma margem de segurança constante, que deve acomodar possíveis atrasos extras na mensagem (CHEN, 2000; BERTIER; MARIN; SENS, 2002) – a idéia fundamental desta estratégia é fazer o *timeout* variar de acordo com o comportamento probabilístico da rede de comunicação, buscando atender requisitos de qualidade de serviço previamente definidos pelo usuário. Nesta abordagem a margem de segurança pode ser fixa (CHEN, 2000) ou variável (BERTIER; MARIN; SENS, 2002).

[2] O *timeout* é fixado empiricamente em 3 vezes o último tempo de ida e volta de uma mensagem.

[3] *Heartbeat* é uma mensagem do tipo "I am alive!" enviada periodicamente para fins de monitoramento de defeitos.

As estratégias A e B podem ser consideradas duas estratégias de extremos. A primeira, preocupada em garantir as propriedades axiomáticas do detector só incrementa o *timeout*, nunca o decrementa. A desvantagem é que, após um longo período, o *timeout* cresce tanto que o tempo para detecção de um defeito (uma função do *timeout*) torna-se proibitivo. Já a segunda estratégia, preocupada somente em se adaptar ao meio de comunicação, adapta dinamicamente o *timeout* somente em função da última observação do *rtt*, ou seja, baseia seu ajuste numa única observação de uma variável aleatória, o que faz o *timeout* também ser aleatório.

A estratégia C é a união das estratégias A e B, ou seja, garante as propriedades axiomáticas do detector através do incremento do *timeout*, cada vez que uma falsa suspeita for verificada, ao mesmo tempo em que adapta o *timeout* ao comportamento da rede. De maneira similar à estratégia B, a adaptação ao comportamento da rede considera somente o último tempo de comunicação (MACÊDO, 1998), só que agora amostrado pelo protocolo SNMP (*Simple Network Management Protocol*) (CASE et al., 1990).

A estratégia D, ao contrário das listadas em A e C, não se preocupa em garantir propriedades axiomáticas para sistemas puramente assíncronos, mas sim em garantir a melhor qualidade de serviço possível, considerando requisitos temporais da aplicação. Para tanto, utiliza estatísticas sobre o comportamento do sistema a fim de melhor se adaptar a ele. Estimativas do tempo de chegada da próxima mensagem são realizadas com base na média aritmética dos n últimos atrasos de comunicação, ou seja, com base numa distribuição de probabilidades Normal.

Em qualquer das estratégias, sempre que um método de previsão é aplicado para estimar o próximo atraso de comunicação, o *timeout* também é composto por uma margem de segurança. A margem de segurança tem a função de possibilitar a sobreposição de variações no atraso. Em estratégias de ajuste do *timeout* que não consideram históricos de atrasos, tal como as estratégias A, B e C, a margem costuma ser um valor pré-fixado, enquanto que em abordagens que consideram históricos ela pode ser fixa ou variável.

1.4 Definição do problema

Conforme apresentado nas seções anteriores, ajustar adequadamente o tempo máximo de espera (*timeout*) numa comunicação entre dois ou mais processos distribuídos num sistema assíncrono sujeito a falhas, tem sido um desafio constante. Requisitos temporais da aplicação ainda tornam mais desafiadora esta tarefa. No campo da computação tolerante a falhas, este desafio repercute no trabalho realizado em várias pesquisas científicas, conforme apresentado na seção anterior.

Entretanto, embora a precisão no ajuste do *timeout* dos detectores de defeitos seja uma peça fundamental para alcançar uma boa qualidade de serviço do detector, foram encontrados poucos trabalhos que fazem previsão baseada nas estatísticas dos atrasos de comunicação. Além disto, estes restringem-se à hipótese de que os atrasos apresentam uma distribuição de probabilidades Normal, ou então que pode ser aproximada à Normal considerando um histórico de $n > 30$ amostras. Na área de detecção de defeitos, não foi identificado qualquer trabalho que explore o uso de técnicas de previsão de valores reconhecidamente mais eficientes em outros domínios, tal como séries temporais na área de econometria (VASCONCELLOS; ALVES, 2000).

1.5 Contribuições deste trabalho

Esta tese contribui na resolução do problema de ajustar adequadamente o *timeout* de um detector de defeitos, explorando uma nova abordagem para previsão do atraso de comunicação no contexto de um mecanismo de detecção de defeitos não confiável: séries temporais (BOWERMAN; O'CONNEL, 1993; BOX; JENKINS; REINSEL, 1994). Especificamente, a tese contribui:

(*i*) demonstrando que a amostragem do atraso de comunicação (*rtt*) de uma mensagem, embora aperiódico, pode ser modelado como sendo uma série temporal, um modelo matemático para amostragens periódicas;

(*ii*) mostrando que a série temporal do *rtt* é, na maioria dos casos, não estacionária durante a maior parte do período diurno e vespertino;

(*iii*) mostrando que o preditor que modela o *rtt* como uma série temporal não estacionária via um modelo autoregressivo integrado de médias móveis (ARIMA), oferece melhor precisão do que os preditores tradicionalmente utilizados nos detectores de defeitos auto-ajustáveis;

(*iv*) mostrando que, num detector de defeitos auto-ajustável, a escolha do preditor deve ser realizada em conjunto com a escolha da margem de segurança para se obter uma melhor qualidade de serviço;

(*v*) propondo o uso de uma nova margem de segurança variável de acordo com o intervalo de previsão, a qual permite obter um *timeout* que sobreponha o atraso de comunicação com uma probabilidade pré-determinada;

(*vi*) propondo uma arquitetura orientada a objetos que permite a construção de serviços de detecção de defeitos flexíveis, ou seja, que podem alterar o tipo de preditor e da margem de segurança em tempo de execução.

1.6 Organização do texto

O capítulo 2 revisa os principais conceitos sobre os detectores de defeitos baseados em *timeouts* e descreve brevemente as estratégias utilizadas para ajustar dinamicamente o *timeout*.

O capítulo 3 faz uma breve revisão dos conceitos e modelos de séries temporais.

O capítulo 4 especifica uma interface para converter a amostragem aperiódica de um detector de defeitos do estilo *pull* numa amostragem periódica. O capítulo também mostra que o tempo de ida e volta de uma mensagem (*rtt*) apresenta um comportamento predominantemente estacionário durante a noite e não estacionário durante as horas úteis do dia.

O capítulo 5 especifica sete preditores distintos, dentre os quais três são baseados em séries temporais, e analisa comparativamente a capacidade preditiva, em termos do erro quadrático médio, de cada um deles.

O capítulo 6 propõe um novo tipo de margem de segurança (variável de acordo com o intervalo de previsão) e avalia o impacto de cada um dos preditores especificados no capítulo 5 na qualidade de serviço de um detector de defeitos. A avaliação considera tanto margens fixas como variáveis para cada preditor.

O capítulo 7 propõe o uso do padrão de projeto *Strategy* (GAMMA et al., 1994) para flexibilizar a implementação de um detector de defeitos e seu módulo de previsão.

Finalmente o capítulo 8 apresenta as conclusões deste trabalho.

2 SOBRE O AJUSTE DO *TIMEOUT*

Este capítulo revisa os conceitos básicos de detectores de defeitos e disserta sobre as estratégias de ajuste do *timeout*. Inicialmente o capítulo apresenta o modelo de sistema distribuído adotado neste trabalho (seção 2.1) e os modelos básicos de funcionamento de um detector de defeitos (seção 2.2), dando ênfase nas referências temporais de cada modelo de detecção e na determinação do *timeout* nos dois principais estilos de implementação. Em seguida, o capítulo faz uma síntese das estratégias de ajuste do *timeout* que vêm sendo praticadas (seção 2.3).

2.1 Modelo de Sistema Distribuído

Escolher um modelo de sistema apropriado não depende somente das propriedades das camadas de software e da rede de computadores que estão sob a aplicação, mas também do uso que se pretende fazer do modelo (SCHNEIDER, 1993). Por exemplo, se alguém deseja implementar aplicações distribuídas com restrições temporais (caracterizadas como de *tempo real*), então o modelo deveria fornecer a noção de tempo, a fim de poder representar (expressar) as restrições de tempo da aplicação. Se o desejo for implementar aplicações que mantenham adequadamente seus possíveis estados, sem restrições rígidas de tempo, então o modelo deveria fornecer a noção de estado, facilitando a demonstração de que os algoritmos da aplicação são válidos. Neste trabalho, o desejo é implementar uma aplicação distribuída (detector de defeitos) que execute em uma rede de longa distância (WAN) e que forneça a noção de estado aos algoritmos que interagem com o detector através da solicitação de informações de estado (*request*) ou da espera por informações (*callback*).

Em sistemas distribuídos, a manutenção de um estado consistente recai na necessidade de acordo distribuído, impossível de ser resolvido num sistema assíncrono puro sujeito a defeitos (FISCHER; LYNCH; PATERSON, 1985). Entretanto, pode ser resolvido num sistema parcialmente síncrono, onde em toda execução existem limites (*bounds*) finitos referentes à velocidade relativa dos processos e ao atraso máximo para que uma mensagem seja entregue, mas estes limites são desconhecidos (modelo $M1$) ou somente válidos após um instante de tempo desconhecido (modelo $M2$), chamado instante de estabilização global (*Global Stabilization Time* - GST) (DWORK; LYNCH; STOCKMEYER, 1988). De acordo com Chandra e Toueg (1996), o acordo também pode ser resolvido num sistema com sincronismo parcial fraco (modelo $M3$), onde em toda execução, de acordo com o modelo $M1$, existem limites finitos, mas desconhecidos, e, de acordo com o modelo $M2$, tais limites só valem após um instante de tempo finito (GST), também desconhecido. Observa-se que antes do instante de estabilização global, tanto no modelo $M2$ como no modelo $M3$, mensagens podem ser perdidas, ou seja, a rede pode ser não confiável.

Neste trabalho, o sistema de comunicação está sujeito tanto à perda de mensagens (defeitos de omissão) como a defeitos de temporização. Tais defeitos podem

ser gerados por sobrecarga ou falhas temporárias, tanto na rede de comunicação como nos nós envolvidos. Assim, o modelo de sistema distribuído adotado neste trabalho é o modelo de sincronismo parcial fraco ($M3$).

Observa-se que, mesmo com um canal de comunicação não confiável, após o GST, os limites de velocidade relativa dos processos e do atraso de comunicação valem permanentemente, permitindo a resolução do acordo em detectores de defeitos com semânticas de abrangência forte (*strong completeness*). Na prática, não é necessário que tal condição valha para sempre, apenas que valha por um período suficiente para que o acordo seja resolvido (CHANDRA; TOUEG, 1996). Com atrasos de comunicação finitos, a rede é não particionável.

Por hipótese, os processos do sistema obedecem a uma semântica de defeitos por colapso (*crash*). Defeitos arbitrários (ou bizantinos), sejam eles nos nós, canais de comunicação ou processos, não são considerados.

2.2 O detector de defeitos

O conceito de detectores de defeitos não confiáveis foi introduzido por Chandra e Toueg (1996) para contornar o problema da impossibilidade de resolver o consenso num sistema assíncrono sujeito a falhas (FISCHER; LYNCH; PATERSON, 1985), o que, em outras palavras, decorre da impossibilidade de distinguir um processo lento de um que realmente está falho.

A idéia básica foi criar a abstração de um *oráculo* que fornecesse a noção de *estado* aos processos de um sistema distribuído, livrando as implementações dos algoritmos de consenso da noção não determinística de tempo (especificamente do atraso de comunicação, que é desconhecido). De acordo com Chandra e Toueg, a noção de estado introduzida, mesmo que não confiável, é suficiente para resolver o consenso com uma maioria de processos corretos.

Para gerar a noção de estado, um detector de defeitos (um algoritmo distribuído) consiste de um conjunto de módulos locais de detecção de defeitos anexados aos componentes (processos, objetos ou nós) de um sistema distribuído. Cada módulo monitora, usualmente via *timeouts*, um subconjunto de componentes no sistema e pode, assim, prover uma informação sobre o estado dos componentes que ele monitora.

Internamente, o estado pode ser mantido na forma de uma lista de componentes *suspeitos* (CHANDRA; TOUEG, 1996), uma lista de componentes *confiáveis* (AGUILERA; CHEN; TOUEG, 1998) ou uma lista de contadores de mensagens recebidas (AGUILERA; CHEN; TOUEG, 1997) (na lista, há um contador para cada componente monitorado). Além disto, para o detector de defeitos, o estado que é informado à aplicação pode corresponder à percepção do módulo de detecção local ou a percepção global dos módulos de detecção. No último caso, um acordo entre os módulos locais é necessário para formar uma lista global. Neste trabalho, o estado é mantido na forma de uma lista de componentes suspeitos e corresponde a percepção do módulo de detecção local.

Do ponto de vista da aplicação, a informação sobre o estado dos componentes pode ser fornecida pelo detector de defeitos de maneira síncrona (a aplicação requisita) ou assíncrona (a aplicação recebe um *callback*) e deve ser computada de acordo com propriedades axiomáticas[1] (CHANDRA; TOUEG, 1996) ou de acordo

[1]Neste texto propriedades que especificam comportamentos alcançáveis (*eventual behaviors*) são

Tabela 2.1: Classes de detecção de defeitos definidas em função das propriedades de abrangência e precisão.

Abrangência (*Completeness*)	**Precisão** (*Accuracy*)			
	Strong	Weak	Eventual Strong	Eventual Weak
Strong	Perfect $\mathcal{P}$	Strong $\mathcal{S}$	Eventually Perfect $\diamond\mathcal{P}$	Eventually Strong $\diamond\mathcal{S}$
Weak	Quasi-Perfect $\mathcal{Q}$	Weak $\mathcal{W}$	Eventually Quasi-Perfect $\diamond\mathcal{Q}$	Eventually Weak $\diamond\mathcal{W}$

com requisitos de qualidade de serviço (CHEN; TOUEG; AGUILERA, 2002), ambos previamente especificados.

Com o intuito de provar que detectores de defeitos podem ser utilizados na resolução do consenso em sistemas distribuídos assíncronos com possibilidade de falhas, Chandra e Toueg (1996) especificaram detectores de defeitos de acordo com duas propriedades axiomáticas: abrangência (*completeness*) e precisão (*accuracy*). A propriedade de abrangência especifica a capacidade do detector em suspeitar dos componentes que tenham realmente falhado, enquanto que a propriedade de precisão especifica a capacidade do detector em evitar erros. Considerando duas variantes para a propriedade de abrangência e quatro variantes para a propriedade de precisão, Chandra e Toueg classificaram detectores de defeitos em oito classes, conforme ilustra a tabela 2.1.

Do ponto de vista abrangência, um detector de defeitos pode garantir que, num tempo finito, todo componente que falha por colapso será permanentemente suspeito por *todo* (propriedade forte - *strong*), ou por *algum* (propriedade fraca - *weak*), componente correto. A propriedade de abrangência pode ser facilmente atendida através do uso de um *timeout*.

Do ponto de vista precisão, onde o sentido de propriedade forte e fraca segue o exposto no parágrafo anterior, detectores podem ser enquadrados numa classe perpétua ou não perpétua (*Eventual*). Numa classe perpétua, *nenhum* componente correto pode ser erroneamente suspeito pelo detector (classes $\mathcal{P}$ e $\mathcal{Q}$) ou *pelo menos um* componente não pode ser erroneamente suspeito (classes $\mathcal{S}$ e $\mathcal{W}$). Numa classe não perpétua, a garantia é que haverá um momento após o qual *todos* os componentes corretos não serão mais erroneamente suspeitos por outros componentes corretos (classes $\diamond\mathcal{P}$ e $\diamond\mathcal{Q}$) ou haverá um momento após o qual *pelo menos um* componente correto não será mais erroneamente suspeito por *pelo menos um* dos outros componente corretos (classes $\diamond\mathcal{S}$ e $\diamond\mathcal{W}$). Diferentemente da propriedade de abrangência, a propriedade de precisão é difícil de ser implementada, podendo ser atendida somente num sistema parcialmente síncrono (LARREA; ARÉVALO; FERNÁNDEZ, 1999). Além disto, nenhuma das quatro classes de detectores de defeitos perpétuos (P, S, Q e W) pode ser implementada num sistema parcialmente síncrono, onde os limites (bounds) dos atrasos de comunicação do sistema existem mas não são conhecidos (LARREA, 2000).

Se a especificação indicar requisitos de qualidade de serviço, ao invés de propriedades axiomáticas, a informação de estado mantida pelo detector deve ser gerada obedecendo aos requisitos, os quais podem ser garantidos por um mecanismo de adaptação dinâmica de parâmetros de configuração tal como a variação da periodicidade de envio de mensagens (CHEN; TOUEG; AGUILERA, 2002).

---

referidas como propriedades axiomáticas, ou seja, que podem servir como axiomas.

Na prática, para cumprir sua função, os módulos locais de detecção de defeitos necessitam trocar mensagens de controle ou de *heartbeat*. De acordo com o fluxo destas mensagens, uma dada implementação do detector de defeitos é classificada como sendo do estilo (FELBER et al., 1999):

- *push*, implementação baseada no envio periódico de mensagens de estado corrente ($I\_am\_alive!$ ou *heartbeat*); ou
- *pull*, implementação baseada no envio periódico de mensagens de solicitação de estado ($Are\_you\_alive?$) com a conseqüente confirmação ($Yes\_I\_am!$).

Naturalmente, outros estilos de implementações também podem ser derivados da composição dos dois anteriores. O exemplo clássico neste contexto é o estilo *dual*, onde o detector é projetado para trabalhar em duas fases. Na primeira, o detector utiliza o modo *push* para a detecção até que ocorra uma suspeita, momento em que passa para a próxima fase. Na segunda fase, o detector tenta refutar a suspeita da primeira fase utilizando o modo *pull* para estimular seu parceiro (componente monitorado) a responder a uma requisição de estado. A informação sobre suspeitas só será passada à aplicação após a confirmação da suspeita na segunda fase. Observa-se que a segunda fase pode servir para contornar o problema de uma possível perda de mensagem e conseqüente suspeita incorreta.

2.2.1 *Timeout* no detector do estilo *push*

Baseado em duas referências temporais, no intervalo de envio das mensagens de $I\_am\_alive!$ (t_h) e no tempo limite (*timeout*) para aguardar o recebimento de uma mensagem, o funcionamento de um detector do estilo *push* é simples e ditado pelas seguintes regras:

R1. A cada t_h unidades de tempo, cada componente p envia uma mensagem $I\_am\_alive!$ ($hbMsg$) a todos os outros componentes que o monitoram.

R2. Se um componente p receber uma mensagem $hbMsg$ de q antes de expirar o respectivo *timeout*, então p reinicializa o *timeout* associado a q.

R3. Se um componente p não receber uma mensagem $hbMsg$ de q dentro de um dado *timeout*, então p adiciona q à sua lista de suspeitos ($SuspectList_p$).

Convém observar que, para evitar falsas suspeitas, o *timeout* utilizado deve ser igual a $t_h+\Delta$, onde Δ é uma margem de segurança que engloba todas as possíveis variações no tempo de transmissão D da mensagem. Entretanto, para um D não conhecido, a margem tende ao infinito, o que não é admissível na maioria dos casos.

Para contornar este problema, que ocorre também no detector do estilo *pull*, a propriedade de precisão é enfraquecida, ou seja, considera-se a possibilidade de ocorrerem falsas suspeitas. A abordagem mais conhecida para implementar a nova semântica é incrementar o *timeout* de um valor fixo k a cada detecção de uma falsa suspeita. Desta maneira, pode-se assumir como hipótese que existirá um momento em que não haverá mais falsas suspeitas. Esta hipótese é válida num modelo de sistema distribuído parcialmente síncrono onde, em toda execução, há limites (*bounds*) na velocidade relativa dos processos e nos atrasos de transmissão de mensagens,

mas estes limites serão conhecidos e válidos somente após um tempo desconhecido, chamado tempo de estabilização global (CHANDRA; TOUEG, 1996). A regra R4 a seguir complementa o funcionamento no estilo *push* segundo esta abordagem.

R4. Se um componente p receber uma mensagem $hbMsg$ de outro componente q que já é suspeito, p remove q da sua lista de suspeitos e adiciona ao respectivo *timeout* uma constante k.

Uma outra abordagem, adotada quando não se conhece o atraso de comunicação e se deseja minimizar o tempo para detecção de um defeito, é adaptar dinamicamente o *timeout* de acordo com as flutuações dos atrasos de comunicações já observados. Para isto, a regra R2 deve ser alterada para:

R2. Se um componente p receber uma mensagem $hbMsg$ de q antes de expirar o respectivo *timeout*, então p reinicializa o *timeout* associado ao q de acordo com uma estratégia de adaptação baseada na predição do atraso de comunicação.

Desta forma, a regra R2 transforma o detector *push* num *detector de defeitos auto-ajustável*, podendo o auto-ajuste ser realizado em relação às variações nos atrasos de comunicação.

Deve-se observar que as regras não eliminam a possibilidade de falsas suspeitas, mas dependendo da qualidade da estratégia de adaptação na regra R2, a relação entre o número de falsas suspeitas e o tempo para detecção de um defeito (relação *custo* x *benefício*) pode ser trabalhada.

2.2.2 *Timeout* no detector do estilo *pull*

Um detector de defeitos do estilo *pull* é baseado em mensagens de controle do tipo $Are\_you\_alive?$ e $Yes\_I\_am!$, daqui para frente referidas por req e ack, respectivamente, e possui duas referências temporais: a periodicidade (t_i) com que ele requisita o estado corrente do componente monitorado e o prazo máximo (*timeout*) para espera pela resposta. Seu funcionamento segue as seguintes regras:

R1. A cada t_i unidades de tempo, cada componente p envia uma mensagem de req a todos os outros componentes monitorados.

R2. Se um componente p receber uma mensagem req de outro componente q, ele imediatamente envia para q uma mensagem ack.

R3. Se um componente p não receber uma mensagem ack de q dentro de um dado *timeout*, então p adiciona q na sua lista de suspeitos ($SuspectList_p$).

De maneira similar ao estilo *push*, para evitar falsas suspeitas o *timeout* deve ser igual a $rtt+\Delta$, onde rtt é o tempo de ida e volta de uma mensagem de requisição de estado (*round-trip time*). Considerando que a variação do rtt é desconhecida, a implementação de um detector *pull* também só é possível com uma propriedade de precisão enfraquecida, o que na prática se reflete numa regra similar à regra R4 do estilo *push*, com a correspondente substituição do tipo de mensagem de $hbMsg$ para ack. Especificamente, para fazer o detector do estilo *pull* alcançar o momento de estabilização global, onde os limites dos atrasos são conhecidos, adiciona-se a seguinte regra:

R4. Se um componente p receber uma mensagem *ack* de outro componente q que já é suspeito, p remove q da sua lista de suspeitos e adiciona ao respectivo *timeout* uma constante k.

Para tornar o detector auto-ajustável, a recepção das mensagens de *ack* deve obedecer a seguinte regra:

R5. Se um componente p receber uma mensagem $ackMsg$ de q antes de expirar o respectivo *timeout*, então p reinicializa o *timeout* associado ao q de acordo com alguma estratégia de adaptação baseada na predição do atraso de comunicação.

A qualidade das estratégias de adaptação utilizada na regra R5 influencia diretamente na qualidade de serviço do detector de defeitos *pull*.

A característica do detector *pull* de ter o controle de envio e recebimento num mesmo componente, o componente monitor, favorece a representação da amostragem de atrasos de comunicação na regra R5 como uma série temporal. No capítulo 6, mostra-se que o uso de séries temporais na regra R5 de um detector *pull* pode melhorar a sua qualidade de serviço. A adaptação realizada na regra R4 é tratada somente na arquitetura do serviço de detecção de defeitos proposta (capítulo 7).

2.3 Estratégias para Ajuste do *timeout*

A revisão dos estilos de detecção discutidos na seção anterior mostra que o *timeout* dos detectores pode ser ajustado dinamicamente em qualquer dos dois principais estilos de detecção. Nesta seção, faz-se uma síntese das principais estratégias empregadas no ajuste dinâmico do *timeout* de detectores de defeitos, a citar, as estratégias utilizadas nos serviços de grupo Jgroup (MONTRESOR, 2000) e OGS (FELBER, 1998), bem como nos trabalhos de pesquisa de Chen e Toueg (2000) e Macêdo (2000). A seção também discute como o *timeout* vem sendo tratado no protocolo de *heartbeat* acelerado proposto por Gouda e McGuire (1998) e discute como funciona o mecanismo de adaptação de *timeout* adotado no protocolo de controle de transmissão (TCP) (JACOBSON, 1988). O princípio de ajuste utilizado no TCP tem sido adotado em protocolos de detecção de defeitos em sistemas distribuídos, conforme será discutido na seção 2.3.7.

2.3.1 *Timeout* em protocolos de *heartbeat* acelerados

Um protocolo de *heartbeat* faz com que processos de um programa distribuído periodicamente troquem mensagens de controle (*beats*). No protocolo, tão logo um processo p percebe o recebimento de um *beat* de um processo q, o processo p considera que tanto o processo q como o meio de comunicação entre eles estão funcionando corretamente. Por outro lado, se p não recebe nenhuma mensagem de controle de q por um longo período, controlado por um *timeout*, p suspeita de q. Tal protocolo pode ser utilizado em procedimentos de diagnóstico de sistemas, de obtenção de acordo, de computação móvel e, naturalmente, de detecção de defeitos, e tem o *timeout* como um ponto crítico em seu projeto (GOUDA; MCGUIRE, 1998).

Para atender a aplicação com uma boa qualidade de serviço, em situações práticas, os protocolos de *heartbeat* devem alcançar um bom compromisso entre três objetivos: reduzir a taxa de envio dos *beats*, reduzir o tempo de detecção (uma

função do *timeout*) e ainda manter baixa a probabilidade de terminação prematura (uma função da precisão e também do *timeout*). Entretanto, segundo Gouda e McGuire (1998), estes objetivos são contraditórios. Reduzir a taxa de envio dos *beats* produz a diminuição do número de mensagens de controle, mas aumenta o tempo para detecção de um defeito (a latência do erro aumenta com a ampliação do intervalo de envio de mensagens). Reduzir o tempo de detecção, diminuindo o *timeout*, aumenta a probabilidade de uma terminação prematura[2], ou seja, de uma falsa suspeita.

Tentando atender os objetivos da melhor maneira possível, Gouda e McGuire apresentaram uma estratégia para acelerar a terminação do algoritmo. Na estratégia, o tamanho de cada período monitorado por um *timeout* depende dos eventos que ocorreram no período anterior, de acordo com as seguintes regras:

R1. Se, num período, um processo p enviar uma mensagem de controle para outro processo q e receber uma mensagem de controle de q, então p faz o tamanho do próximo período ser igual a um limite superior pré-especificado t_{max} (independente do tamanho do período corrente).

R2. Se, num período, p enviar uma mensagem de controle para q, mas não receber uma mensagem de controle de q, então p faz o tamanho do próximo período ser a metade do período corrente.

R3. Se o tamanho do próximo período for menor do que um valor pré-especificado t_{min}, então p torna-se inativo (bloqueia-se) e pára de enviar mensagens de controle para q (e para qualquer outro processo que esteja lhe escutando).

A regra 1 é adotada para garantir que, quando os dois processos e o meio de comunicação estão funcionando corretamente (uma situação típica), a taxa de envio de mensagens de controle é pequena. As regras 2 e 3 são adotadas quando o p suspeita de um defeito ou terminação. Inicialmente, pela regra 2, p tenta refutar esta suspeita várias vezes enviando novas mensagens de controle em períodos cada vez menores (metade do período anterior) antes de finalmente aceitar sua suspeita e bloquear-se, regra 3. Esta atuação é referida como *exponential speed-up*. As duas últimas regras garantem um tempo de detecção menor, se comparado com o algoritmo sem o *exponential speed-up* e uma baixa probabilidade de terminação prematura.

Deve ser observado que esta estratégia depende de duas referências temporais: o t_{max} e o t_{min}. Estes valores são duas constantes, e devem ser escolhidos de tal forma que a probabilidade de uma falsa suspeita seja mínima.

Fazendo R denotar o número máximo de rodadas[3] consecutivas sem receber mensagens de controle, a relação entre t_{max} e o t_{min} é dada por:

$$2^{R-1}.t_{min} \leq t_{max} < 2^{R}.t_{min}$$

[2] O termo *terminação prematura* foi utilizado por Gouda e McGuire (1998) porque, no protocolo de *heartbeat* proposto, ao detectar um defeito o componente pára de enviar mensagens de controle, o que gera a terminação do algoritmo num tempo finito.

[3] Neste contexto uma rodada inicia no instante do envio de uma mensagem de controle e termina no instante anterior ao envio da próxima mensagem de controle.

O protocolo termina se ele passa por R rodadas sem receber nenhuma mensagem de controle. Assim, a probabilidade da rodada ser terminal é dada por:

$$P_{terminal} = (1 - (1 - P_{loss})^2)^R$$

onde P_{loss} é a probabilidade de perda de uma mensagem de controle.

A probabilidade de terminação prematura $P_{premature}$ para qualquer execução do protocolo é dado por:

$$P_{premature} = \begin{cases} 0 & \text{if } r \leq 2 \\ \sum_{i=1}^{r-2}(1 - P_{terminal})^{i-1}.P_{terminal} & \text{if } r > 2 \end{cases}$$

onde, $r = \frac{T}{t_{max}}$ é o número de rodadas com recepção de mensagens e T é o intervalo de tempo onde se pretende verificar a probabilidade de terminação prematura.

Em termos de implementação, com este método supõe-se que os parâmetros t_{min} e P_{loss} possam ser estimados das características da rede. O t_{min} poderia corresponder ao limite superior do rtt ou ao rtt médio adicionado de uma margem de confiança, enquanto que P_{loss} poderia ser computado a partir do número de mensagens perdidas, contabilizadas pelos números de seqüência estampados nas mensagens. Já o t_{max}, poderia ser vinculado a um requisito de qualidade de serviço como, por exemplo, o tempo máximo aceitável para detectar um defeito. O problema é que as características da rede não são constantes, o que dificulta o ajuste prévio dos limites.

Por outro lado, a técnica *exponencial speed-up* mostra que o uso repetido de refutação dentro do período de interrogação não aumenta o tempo máximo para detecção de um defeito e conseqüentemente não compromete a qualidade de serviço do protocolo. Em síntese, do trabalho de Gouda e McGuire, percebe-se que, de maneira similar ao estilo de detecção *dual*, o uso de uma fase de refutação após o estouro de um dado *timeout* (nesta seção chamado t_{max}) pode reduzir a probabilidade de terminação prematura ou, em outras palavras, a probabilidade de uma falsa suspeita. O uso de refutação aumenta o número de mensagens mas é importante para modelos que consideram a possibilidade de defeitos de omissão (perda de mensagens).

2.3.2 *Timeout* no serviço de grupo Jgroup

As condições de estabilidade do serviço de grupo Jgroup (MONTRESOR, 2000) dependem de um módulo de detecção de defeitos baseado em *timeout*. No Jgroup, a especificação deste módulo é uma extensão do trabalho de Chandra e Toueg (1996) para sistemas particionáveis (sistemas onde a rede não é confiável). Segundo o autor, em sistemas particionáveis, a especificação das propriedades dos detectores de defeito deve ser baseada na *alcançabilidade* entre pares de objetos, ao invés de objetos individuais corretos ou falhos.

Seguindo a classificação de Chandra e Toueg, o detector de defeitos do Jgroup é da classe *eventually perfect* e tem as seguintes propriedades:

- **Strong Completeness** - se algum objeto q permanece inacessível a um objeto correto p, então, em um intervalo de tempo finito, p suspeitará de q.

- **Eventual Strong Accuracy** - se algum objeto q permanece acessível a um objeto correto p, então p não suspeitará de q após um tempo finito.

Para satisfazer a propriedade *strong completeness*, o Jgroup utiliza um mecanismo de *ping* periódico (um detector de defeitos do estilo *pull*), enquanto que, para satisfazer a propriedade *eventual strong accuracy*, duas condições foram necessárias: C1 - deixar a aplicação definir o *timeout* do teste de acessibilidade; e C2 - assumir por hipótese que os canais de comunicação são simétricos, ou seja, que não há possibilidade de um cenário onde são sempre perdidas apenas mensagens que trafegam em um dado sentido.

Assumindo C1 e C2, sempre que uma falsa suspeita for percebida, o *timeout* é incrementado de k unidades de tempo (um valor constante), ou seja, sempre que um membro q receber uma mensagem de um membro p que está na sua lista de suspeitos, ele incrementa k unidades de tempo ao último *timeout* associado a p. Se p não pertencer à lista de suspeitos, então o *timeout* não é alterado.

Do Jgroup percebe-se que as propriedades de Chandra e Toueg também podem ser descritas em função de testes de acessibilidade, importantes para modelos que assumem a hipótese de defeitos de temporização, pois com esta hipótese não há como diferenciar uma partição de um defeito de temporização. Além disto, o Jgroup também mostra que para garantir propriedades axiomáticas de precisão, tal como a *Eventual Strong Accuracy*, a cada percepção de uma falsa suspeita é necessário incrementar o *timeout*.

2.3.3 *Timeout* no Object Group Service - OGS

O protocolo usado no *Object Group Service* (OGS) para implementar uma detecção de defeitos do estilo *pull* está baseado num mecanismo de *timeout* adaptativo (FELBER, 1998) baseado no *rtt* mensurado. Assim, dependendo do tempo de resposta dos objetos (*rtt*), o *timeout* é incrementado ou decrementado. Em geral, o OGS tenta ajustar o *timeout* para $3.rtt$, mas este valor é empírico (FELBER, 2000). Em síntese, a estratégia de ajuste do *timeout* no OGS consiste em seguir o último atraso amostrado, o que fará o *timeout* variar de maneira desconhecida.

2.3.4 *Timeout* no trabalho de pesquisa de Chen e Toueg

Com o intuito de validar métricas de qualidade de serviço, Chen, Toueg e Aguilera (2002) propuseram várias implementações de um detector de defeitos adaptativo baseado em *heartbeats*. Dentre elas, o ajuste do *timeout* varia de acordo com as hipóteses sobre a sincronização de relógios e sobre o comportamento probabilístico da rede. Nesta seção, aborda-se somente a implementação para sistemas onde os relógios não são sincronizados, processos falham por colapso e o meio de comunicação pode atrasar ou perder mensagens segundo uma distribuição de probabilidade desconhecida.

O algoritmo de detecção de defeitos proposto por Chen faz com que um processo p envie mensagens de *heartbeat* $m_1, m_2, ...$ para um outro processo q a cada t_i unidades de tempo[4]. Para determinar se p está falho, q usa uma seqüência de pontos de tempo $\tau_1, \tau_2, ...$, chamados *freshness points*, calculados por:

$$\tau_{l+1} = EA_{l+1} + \Delta$$

onde EA_{l+1} é a estimativa do tempo de chegada da próxima mensagem de *heartbeat*, o índice l indica o maior número de seqüência já recebido e Δ é uma margem de

[4]No original t_i é representado por η.

segurança fixa[5] (função do tempo máximo permitido para detectar um defeito) usada para acomodar possíveis atrasos extras ou perdas de mensagens. Deste modo, em algum instante de tempo $t \in [\tau_i, \tau_{i+1})$, um processo q confia em p no instante t se e somente se q já tiver recebido uma mensagem de *heartbeat* m_i, ou alguma outra enviada posteriormente. Em outras palavras, enquanto o *timeout*, chamado τ_{i+1} não expirar, um processo q confia em p.

A estimativa do tempo de chegada para a próxima mensagem de *heartbeat* é computada por:

$$EA_{l+1} \approx \frac{1}{n}(\sum_{i=1}^{n} A'_i - t_i.mseq_i) + (l+1)t_i$$

onde $mseq_i$ é o número de seqüência da mensagem m_i, A'_i é o instante de recebimento da mensagem de *heartbeat* m_i, e n é o tamanho da amostra considerada para o cálculo da estimativa. Percebe-se que o primeiro termo da fórmula normaliza cada instante de recebimento de uma mensagem de *heartbeat* A'_i, deslocando-o para trás $t_i.mseq_i$ unidades de tempo, e realiza a média dos A'_is normalizados. O segundo termo adiciona $(l+1)t_i$ unidades de tempo à média computada, a fim de chegar a um valor que corresponda a estimativa do próximo instante de recebimento de uma mensagem de *heartbeat*.

Para que o estimador EA_{l+1} não seja viciado, Chen indica que deve ser usado $n > 30$ a fim de poder usar as propriedades do teorema da amostragem[6]. Em outras palavras, embora o modelo considere uma distribuição de probabilidades desconhecida, a previsão do atraso médio considera, por aproximação, que a distribuição é Normal.

Do trabalho de Chen e Toueg, percebe-se que para modelos de sistema distribuídos mais realistas (relógios não sincronizados, atrasos de comunicação desconhecidos e possibilidade de perda de mensagens) hipóteses sobre modelos probabilísticos são necessárias. Embora a distribuição de probabilidades seja desconhecida, por aproximação os cálculos do atraso de comunicação consideram uma distribuição Normal. Além disto, embora modelos probabilísticos favoreçam o uso de margens de segurança variáveis, Chen e Toueg adotam uma margem fixa, repassando para a aplicação, ou projetista do protocolo, a responsabilidade de estipulá-la, o que é um desafio.

2.3.5 Indicador do Tempo de Conectividade - CTI

Partindo de um modelo de sistema distribuído onde os canais de comunicação são confiáveis, os processos falham por colapso e os atrasos de comunicação não são mensuráveis, Macêdo (2000) propôs o ajuste do *timeout* dos detectores de acordo com a informação passada por um módulo Indicador do Tempo de Conectividade (CTI) (MACÊDO, 1998). Especificamente, considerando como aplicação um algoritmo de *membership* de grupo, o *timeout* corresponde à soma do tempo necessário para produzir um *heartbeat* com o tempo de conectividade D, definido como o tempo requerido para uma mensagem trafegar de um processo p até outro processo q (ou

[5] No original Δ tem outro significado, mas para manter a coerência com outras seções, neste texto Δ equivale à margem de segurança α citada por Chandra, Toueg e Aguilera (2002).

[6] O teorema da amostragem indica que, se n for grande, qualquer distribuição de probabilidade pode ser aproximada para uma distribuição Normal. Entretanto, por senso comum (*rule of thumb*), os estatísticos indicam que o teorema é válido se $n > 30$ (ALLEN, 1990, p. 434).

vice-versa). Como o valor de D depende do comportamento real da rede de comunicação, ele pode variar de 0, se $p = q$, a ∞, se p e q estão realmente desconectados. Macêdo ainda salienta que D pode ser definido como constante, caso a rede possua um comportamento previsível como é o caso de sistemas de tempo real estrito (*hard real-time*).

No CTI, D é calculado pela soma de três tempos: T1 - o tempo que uma mensagem leva de um processo da aplicação até o *driver* de comunicação local; T2 - o tempo que uma mensagem leva do *driver* de comunicação local até o *driver* de comunicação remoto; e T3 - o tempo que uma mensagem leva do *driver* de comunicação remoto até o processo da aplicação remota.

Os tempos T1 e T3 são computados por um analisador de carga do sistema operacional, um processo de alta prioridade, que tenta estimar o tempo de comunicação entre um dado processo e o respectivo *driver* de rede da máquina local. Nenhuma técnica estatística é utilizada neste caso, ou seja, a estimativa é definida pela última medida. O tempo T2 é computado por um analisador do estado corrente das conexões entre as máquinas monitoradas, baseado no protocolo de gerência de redes SNMP. O tempo de conectividade (soma dos três tempos computados) é constantemente atualizado numa base de dados do CTI, sendo que cada atualização sobrescreve a informação prévia.

Antes de utilizar o CTI, a aplicação deve registrar, na base de dados do CTI, cada par de processos que devem ser monitorados. Após registrar o par de processos, a aplicação pode consultar a base de dados por demanda (*polling*) ou indicar que deseja ser informada de maneira assíncrona (por interrupção) sobre alterações no tempo de conectividade.

Considerando como aplicação um protocolo de consenso que exige propriedades axiomáticas, Macêdo ainda adiciona ao *timeout* um fator de ajuste chamado $recoveryTime$, o qual tem a função de suprir o tempo necessário para uma eventual recuperação do processo. Tal fator é incrementado por uma constante de k unidades de tempo a cada percepção de falsa suspeita, semelhantemente ao que foi realizado no Jgroup (seção 2.3.2).

Embora não tenha sido o objetivo do autor, a proposição do CTI sinaliza que a tarefa de computar o tempo de conectividade, ou atraso de comunicação, pode ser dissociada da tarefa de garantir propriedades ao algoritmo de detecção de defeitos. Dissociando estas tarefas, pode-se construir arquiteturas flexíveis como a que é proposta na seção 7.2.

Em termos do ajuste realizado no módulo CTI, assim como no OGS, a última medida do tempo de conectividade descreve o comportamento da variável aleatória D, o que também fará o *timeout* variar de maneira desconhecida.

2.3.6 Timeout no TCP

Considerando que o tempo de comunicação sofre influência de fatores tais como velocidade da rede, mudanças de tráfego, congestionamento e mudanças de rotas, o protocolo TCP (*Transport Control Protocol*) estabelece que o *timeout* deve ser ajustado de acordo com o rtt, medido a cada transmissão e respectivo recebimento da resposta.

Seguindo a recomendação de Jacobson (1988), no TCP a estimativa do próximo rtt ($\widehat{rtt}$) é computada através de um método de previsão simples baseado num

filtro linear (uma média exponencialmente ponderada), ou seja:

$$\widehat{rtt} \leftarrow \alpha.rtt + (1 - \alpha).\widehat{rtt}$$

onde α é uma constante de alisamento, ou de ponderação, com um valor recomendado de $0,125$ (ou 12,5%). Em outras palavras, o $\widehat{rtt}$ é atualizado, a cada nova medida do rtt, para um valor que corresponde a 87,5% da estimativa prévia mais 12,5% da nova medida. O resultado é uma previsão fortemente dependente da média e que minimiza o impacto de grande flutuações.

Entretanto, para poder operar sob situações de alta variância no rtt, sem causar muitas retransmissões desnecessárias, o TCP calcula o *timeout* de retransmissão (rt_o) baseando-se tanto na média como na variância do rtt, de acordo com as seguintes equações:

$$\begin{aligned} Err &= rtt - \widehat{rtt} \\ \widehat{rtt} &\leftarrow \widehat{rtt} + g.Err \\ Dev &\leftarrow Dev + g.(\mid Err \mid - Dev) \\ rt_o &= \widehat{rtt} + 4.Dev \end{aligned}$$

onde Err é o erro do estimador corrente (diferença entre o último valor medido e seu estimador), Dev é o desvio médio exponencialmente ponderado, e g é a constante de ponderação utilizada no cômputo do $\widehat{rtt}$ e Dev. A constante de ponderação g aplicada no cálculo é ajustada para $g = 0,125$ (potência de 2 mais próxima ao valor de $0,1$ sugerido para α no RFC 793).

Como a constante de ponderação para o Dev não necessita ser a mesma do $\widehat{rtt}$, para forçar uma mudança rápida de rt_o em resposta a mudanças no rtt, a constante de ponderação para o Dev pode ser maior do que g. Assim, mantendo uma relação em potência de 2 para facilitar o cálculo (por deslocamento), a constante para o Dev é ajustada para $h = 2.g = 0.25$, e a expressão do desvio médio é dada por:

$$Dev \leftarrow Dev + h.(Err - Dev)$$

Para oferecer confiabilidade, toda vez que o rt_o expira, o TCP utiliza um algoritmo de retransmissão que funciona de acordo com uma política chamada *exponential backoff*. Esta política estabelece que, a cada retransmissão de um pacote, o valor do *timeout* (rt_o) seja multiplicado por uma potência de 2, ou seja, na primeira retransmissão, o rt_o é multiplicado por 2, na segunda por 4, e assim por diante. Esta política permite que o *timeout* inicial seja um valor mínimo, priorizando a transmissão veloz de dados. Ao mesmo tempo, a política procura reduzir o problema de possíveis congestionamentos, diminuindo por potências de 2 a periodicidade da retransmissão até que o intervalo entre retransmissões chegue a um valor máximo, quando então a aplicação é sinalizada sobre a impossibilidade de transmissão.

2.3.7 Outros trabalhos relacionados

Bertier, Marin e Sens (2002), adotando o estilo de detecção *push*, ajustam o *timeout* do detector de defeitos usando uma combinação das estratégias sugeridas por Chen, Toueg e Aguilera (2002) (seção 2.3.4) e por Jacobson (1988) (seção 2.3.6). A estimativa do *timeout* segue o modelo básico de estimativa proposto por Chen e Toueg, que consiste na soma da estimativa do tempo de chegada da próxima mensagem de *heartbeat* (EA_{l+1}) com uma margem de segurança Δ. Entretanto, ao

invés de adotar uma margem de segurança fixa, adotam uma margem de segurança variável de acordo desvio médio, mas com uma constante de alisamento h igual a $0,1$, ao invés de $0,25$ como sugerido por Jacobson. Esta diminuição de h faz a margem reagir de maneira mais conservadora quanto às variações dos atrasos de comunicação. Além disto, ao invés de somar 4 vezes o desvio médio alisado no cômputo do *timeout*, soma somente o dobro do desvio médio. Isto reduz de maneira linear o tamanho da margem de segurança, o que diminui o tempo para detecção de um defeito.

Keshav et al. (1995) avaliaram estratégias de adaptação do *timeout* utilizado no controle de fechamento de circuitos virtuais de redes ATM (*Asynchronous Transfer Mode*) que transportam tráfego IP. Em sua avaliação, duas estratégias de adaptação se destacam: a baseada na média e na variância, e a chamada adaptativa. A primeira consiste na aplicação da estratégia sugerida por Jacobson (seção 2.3.6), ou seja, cômputo da média e do desvio médio exponencialmente ponderados. A segunda usa um *timeout* máximo admitido pelo usuário e um histograma dos intervalos de tempo entre recebimentos de pacotes. Do histograma é extraído o menor intervalo de tempo (custo) estimado para receber um novo pacote. Caso este custo (agregado a uma margem em função do custo de reabertura do circuito virtual) seja menor do que o admitido pelo usuário, o *timeout* é ajustado para este valor. Caso contrário, o *timeout* é ajustado para zero e o circuito é fechado.

2.4 Conclusões Parciais

Há dois estilos básicos de detectores de defeitos se considerado o modelo de fluxo de dados: *push* e *pull*, sendo que ambos são baseados no envio periódico de mensagens de controle. A diferença básica é que no *push* o emissor de mensagens e o controlador do *timeout* são componentes distintos e no *pull* não. No *pull*, o monitor envia periodicamente requisições de estado aos monitorados e controla o *timeout* para recebimento da mensagem de *ack*. Esta característica do estilo *pull* favorece a representação dos atrasos de comunicação como uma série temporal e faz deste detector o objeto de avaliação nesta tese.

Das características dos algoritmos de detecção percebe-se que há pelo menos dois instantes distintos onde estratégias de adaptação do *timeout* são aplicadas: quando o componente monitorado não é um suspeito e quando uma falsa suspeita é percebida.

Quando o componente monitorado não é um suspeito, a estratégia de adaptação pode variar o *timeout* considerando os valores prévios do atraso de comunicação, mantidos num preditor. Neste caso, pode-se dissociar o algoritmo de detecção dos algoritmos de previsão de valores, tal como foi realizado com a utilização do módulo CTI no trabalho de Macêdo (seção 2.3.5). Esta dissociação habilita a proposição de arquiteturas flexíveis que permitem a escolha em separado da estratégia de detecção de defeitos e da estratégia de previsão do *timeout*. Na seção 7.2, é apresentada uma arquitetura deste tipo para um serviço de detecção de defeitos.

A previsão do atraso de comunicação tem sido realizada considerando um histórico de observações (com uma ou mais observações). Nos sistemas OGS (seção 2.3.3) e Jgroup (seção 2.3.2) e no algoritmo de Macêdo, a estratégia de adaptação utiliza um histórico com apenas o último atraso de comunicação mensurado. Computado como um múltiplo deste, o *timeout* neste caso tende a ser um valor aleatório,

tal como é o atraso de comunicação. No protocolo TCP (seção 2.3.6) e no trabalho de Chen e Toueg (seção 2.3.4), um histórico com mais de um valor é utilizado. No TCP, o *timeout* é ajustado para um valor igual ao *rtt* médio adicionado de quatro vezes sua derivação média, ambos ponderados, mas com constantes de ponderação diferentes para permitir uma resposta mais rápida a flutuações no *rtt*. No trabalho de Chen e Toueg, o *timeout* é ajustado considerando a média aritmética dos atrasos de comunicação adicionada ao período de envio de mensagens de *heartbeat* e a uma margem de segurança fixa. Mesmo considerando relógios não sincronizados, atrasos de comunicação desconhecidos e possibilidade de perda de mensagens, tais estratégias consideram que os valores futuros da série temporal seguem um padrão comportamental constante ao redor de uma média, ou seja, não consideram que o histórico possa descrever uma tendência crescente ou decrescente. Bertier et al. (seção 2.3.7) utilizam a mesma estratégia de Chen e Toueg, mas com uma margem variável similar a do TCP (duas vezes a derivação média ponderada).

Quando uma falsa suspeita é percebida, para evitar novas suspeitas pode-se incrementar o *timeout* com um valor constante. A estratégia de incrementar o *timeout* a cada percepção de falsa suspeita é a maneira originalmente proposta por Chandra e Toueg para assegurar as propriedades axiomáticas numa implementação de detector. Neste sentido, com o intuito de garantir o cumprimento das propriedades axiomáticas, esta estratégia foi utilizada no Jgroup (seção 2.3.2) e no algoritmo de detecção do Macêdo (seção 2.3.5).

Quando um *timeout* expira, há duas estratégias que podem ser adotadas: assumir por hipótese que de fato o componente monitorado é um suspeito ou tentar refutar a suspeita. A refutação é a estratégia utilizada no estilo *dual* de detecção, no algoritmo de *heartbeat* acelerado e no algoritmo do TCP. Em qualquer dos casos, o uso de refutação aumenta o número de mensagens de controle, mas reduz a probabilidade de uma falsa suspeita, sendo de especial importância para modelos que consideram a possibilidade de defeitos de omissão (perda de mensagens) e que não desejam *timeouts* de valores muito elevados. Num detector do estilo *dual*, uma única retransmissão é realizada, enquanto que, nos algoritmos de *heartbeat* e do TCP, várias podem ser realizadas, dependendo dos parâmetros previamente ajustados. Entretanto, com hipóteses diferentes, os mecanismos *exponential back-off* (seção 2.3.6) e *exponential speed-up* (seção 2.3.1) operam diferentemente. Nenhum visa alterar diretamente o *timeout*, mas sim a periodicidade do envio de mensagens. O TCP deseja enviar muitas mensagens de dados no menor tempo possível, logo inicializa a periodicidade de envio em t_{min} e incrementa o período a cada retransmissão. Por outro lado, o protocolo de *heartbeat* deseja enviar poucas mensagens de controle, por isso o período entre elas é inicialmente t_{max} e não t_{min}, diminuindo o período a cada nova retransmissão. Do ponto de vista de detecção de defeitos, o mecanismo *exponencial speed-up* é eficiente, pois mesmo com o uso repetido de refutação não aumenta o tempo máximo para detecção de um defeito (terminará dentro de um período de interrogação), não comprometendo a qualidade de serviço do protocolo.

Em síntese, a conclusão é que a adaptação dinâmica do *timeout* de detectores de defeitos em períodos onde o componente monitorado não é suspeito depende da habilidade de um preditor de valores. Além disto, embora existam diferentes estratégias de previsão do atraso de comunicação futuro, a implementação de um preditor pode ser realizada de maneira independente do detector de defeitos que o

utiliza, pois para o detector interessa somente a estimativa do próximo atraso. Por outro lado, as estratégias para ajuste, em casos de estouro do *timeout* ou em casos de percepção de falsas suspeitas, dependem das propriedades do detector e devem ser implementadas junto aos algoritmos de detecção de defeitos.

3 SÉRIES TEMPORAIS

Uma modelagem que está sendo explorada, em diferentes domínios na comunidade científica de computação (BASU; MUKHERJEE; KLIVANSKY, 1996; NETO; MADEIRA, 1998; DINDA; O'HALLARON, 1999a; SCHULZ; HOCHBERGER; TAVANGARIAN, 2001), é a baseada na teoria de séries temporais (MORETTIN; CASTRO TOLOI, 1981; HAMILTON, 1994; BOX; JENKINS; REINSEL, 1994). Esta modelagem pode ser feita com objetivos de descrever, explicar, investigar, prever ou controlar alguns fenômenos. De interesse especial, tem-se que os métodos de previsão baseados nesta teoria fornecem estimativas relativamente confiáveis e sofisticadas do comportamento futuro do sistema (MORETTIN; CASTRO TOLOI, 1981).

Como, neste trabalho, aplicam-se os conceitos de séries temporais para modelar e prever valores futuros do atraso de comunicação e conseqüentemente determinar dinamicamente o *timeout* de detectores de defeitos, neste capítulo, apresentam-se as principais características de uma série temporal (seção 3.1), os principais modelos matemáticos utilizados para analisá-la (seção 3.2), e como se pode fazer previsão de valores com base nesta modelagem (seção 3.3).

3.1 Conceitos básicos

3.1.1 Definição

Genericamente, uma série temporal é definida como um conjunto de observações de uma dada variável, ordenadas no tempo, geralmente em intervalos eqüidistantes (SOUZA; CAMARGO, 1996). Quando a variável é aleatória, como o atraso de comunicação, a série temporal pode ser definida como a realização de um processo estocástico (BOX; JENKINS; REINSEL, 1994), um fenômeno que varia de acordo com leis probabilísticas à medida que o tempo passa (OCHI, 1990). Uma realização Z_t de um processo estocástico Z corresponde a uma única amostragem de n observações da variável.

Sendo a série temporal uma realização de um processo estocástico, pode-se descrevê-la por seus parâmetros estatísticos tais como média, variância, autocovariância e autocorrelação.

3.1.2 Estacionariedade e ergodicidade

Para conhecer os parâmetros estatísticos de um dado processo estocástico, é necessário conhecer todas as funções de distribuição que descrevem o comportamento do processo. Entretanto, na prática, é comum que se conheca apenas uma, ou algumas, das possíveis realizações. Portanto, para inferir o comportamento do processo a partir de uma única realização, a série temporal amostrada, pode-se assumir como hipótese que o processo em questão é estacionário e ergódico (BOX; JENKINS;

REINSEL, 1994), ou seja, que o processo permanece num estado de equilíbrio sobre um nível médio constante (estacionário) e que seu comportamento estacionário pode ser inferido de uma única realização (ergódico).

Como a maioria das séries práticas não é estacionária, ou seja, apresenta variações de nível e tendência, Box, Jenkins e Reinsel simplificam a solução assumindo como hipótese que a série temporal pertence à classe de séries não estacionárias homogêneas (não apresentam dados evolucionários nem explosivos). Séries desta classe podem ser transformadas em estacionárias aplicando um operador de diferença ∇, definido como $\nabla z_t = z_t - z_{t-1}$, onde z_t é o valor observado no instante t. Assim, por exemplo, se a série Z_t é uma série não estacionária homogênea de primeira ordem, então $w_t = Z_t - Z_{t-1} = \nabla.Z_t$ é estacionária. Na pratica isto significa que uma série não estacionária pode ser convertida numa série estacionária se realizado d diferenças sucessivas da série original ($w_t = \nabla^d.Z_t$). O número de vezes que o operador ∇ deve ser aplicado para que a série se torne estacionária é denominado ordem de integração. Neste trabalho, a ordem de integração é computada (seção 4.2.2) supondo uma classe de séries não estacionárias homogêneas.

3.2 Modelos Matemáticos para Séries Temporais

De acordo com Morettin e Toloi (1981), uma série temporal pode ser representada na forma de um sinal $f(t)$ adicionado de um ruído a_t, ou seja,

$$Z_t = f(t) + a_t$$

onde a_t tem média zero e variância constante (*ruído branco*).

Dependendo das hipóteses feitas sobre $f(t)$ e a_t, pode-se enquadrar os modelos em duas classes: modelos de erro e modelos auto-regressivos integrados de médias móveis - ARIMA.

3.2.1 Modelos de erro ou de regressão

Esta classe de modelos é usual quando as hipóteses são: (i) as observações que compõem $f(t)$ são independentes; (ii) a_t e $f(t)$ são independentes; e (iii) as observações que compõem a_t não são correlacionadas. Das hipóteses ii e iii, tem-se que qualquer efeito do tempo influencia somente a parte determinística $f(t)$.

De acordo com a forma do sinal $f(t)$, também referenciado como série $f(t)$, a série temporal Z_t pode ser representada por um modelo constante ($Z_t = \kappa + a_t$, onde κ representa o nível), um modelo de tendência linear ($Z_t = \kappa + \psi.t + a_t$, onde o termo $\psi.t$ determina a inclinação no tempo t), um modelo de regressão ($Z_t = \kappa + \psi.x_t + a_t$, onde x_t é uma quantidade observável), ou um modelo de curva de crescimento ($Z_t = \kappa.e^{(\psi.t+a_t)}$, onde, em geral, $f(t)$ é representado por um polinômio em t (tendência) e um polinômio harmônico (sazonalidade)).

3.2.2 Modelos ARIMA

Propostos por Box, Jenkins e Reinsel (1994), a classe de modelos ARIMA supõe que as observações que compõem $f(t)$ são correlacionadas. Entretanto, como $f(t)$ muitas vezes não pode ser representada como um polinômio simples somente dependente do tempo (MORETTIN; CASTRO TOLOI, 1981), a classe de modelos ARIMA considera que a série Z_t é resultante de um filtro linear, ao invés de uma

fonte de sinal mais ruído. Deste modo, o processo Z_t de saída do filtro é dado por:

$$Z_t = \mu + a_t + \psi_1.a_{t-1} + \psi_2.a_{t-2} + ...$$

onde μ é a média do processo (nível) e ψ_i são coeficientes da função de transferência. Teoricamente, se $\sum_{i=0}^{\infty} |\psi_i| < \infty$, o filtro linear é estável e o processo Z_t é estacionário.

Considerando que um processo ruído branco a_t, com média zero e variância finita, passa pela função de transferência do filtro $\psi(B) = 1 + \psi_1.B + \psi_2.B^2 + ...$, num dado instante t, o valor z_t do processo Z_t, é dado por

$$z_t = \psi(B).a_t$$

Na função de transferência $\psi(B)$, B é o operador de defasagem (*backward shift operator*) definido por $B.z_t = z_{t-1}$.

A classe de modelos proposta por Box, Jenkins e Reinsel pode descrever dois tipos de processos: os processos lineares estacionários (através dos modelos AR(p), MA(q) e ARMA(p, q) - casos particulares do modelo ARIMA) e os processos lineares não-estacionários homogêneos (através do modelo ARIMA(p, d, q)). Estes modelos são caracterizados a seguir.

3.2.2.1 Modelo auto-regressivo - AR

Este modelo é adequado para representar processos puramente auto-regressivos de ordem p (AR(p)), onde o valor corrente z_t do processo estocástico é função dos p valores prévios do processo mais um choque aleatório a_t, ou seja,

$$z_t = \phi_1.z_{t-1} + ... + \phi_p.z_{t-p} + a_t$$

A ordem p do processo indica o número de coeficientes auto-regressivos; logo, o operador auto-regressivo $\phi(B)$ é dado por $\phi(B) = 1 - \phi_1.B - \phi_2.B^2 - ... - \phi_p.B^p$. De uma forma sintética, um modelo AR(p) é definida por

$$z_t = \frac{1}{\phi(B)}.a_t$$

3.2.2.2 Modelo de médias móveis - MA

Este modelo é adequado para representar processos puramente de médias móveis[1] de ordem q (MA(q)), onde o valor corrente z_t do processo estocástico é função dos q valores prévios do choque aleatório a_t mais o choque aleatório corrente, ou seja,

$$z_t = a_t - \theta_1.a_{t-1} - ... - \theta_q.a_{t-q}$$

ou

$$z_t = \theta(B).a_t$$

A ordem q indica o número de coeficientes do operador de médias móveis $\theta(B)$, ou seja, $\theta(B) = 1 - \theta_1.B - \theta_2.B^2 - ... - \theta_q.B^q$.

[1]Como z_t é uma função algébrica ponderada dos a_t que se move no tempo, Box, Jenkins e Reinsel escolheram o nome *médias móveis*. Mas como a soma dos parâmetros do polinômio não resulta no valor de uma unidade, o nome não é exato.

3.2.2.3 Modelo auto-regressivo e de médias móveis - ARMA

Este modelo agrega tanto termos auto-regressivos $\phi(B)$ como termos de médias móveis $\theta(B)$, e é adequado para representar processos auto-regressivos e de médias móveis de ordem p e q (ARMA(p,q)), onde

$$z_t = \frac{\theta(B)}{\phi(B)} a_t$$

Como este modelo combina os modelos AR(p) e o MA(q), na prática são necessários poucos coeficientes, tanto para $\theta(B)$ quanto para $\phi(B)$, para descrever a série. Normalmente restringem-se a dois (BOX; JENKINS; REINSEL, 1994, p. 11).

3.2.2.4 Modelo auto-regressivo integrado de médias móveis - ARIMA

Este modelo é adequado para representar séries não estacionárias homogêneas através de processos auto-regressivos integrados de médias móveis de ordem p, d e q (ARIMA(p,d,q)), onde

$$z_t = \frac{\theta(B)}{\phi(B).\nabla^d} . a_t$$

A ordem d representa o número de diferenças necessárias para transformar uma série não estacionária em estacionária. Na prática, d é usualmente 0, 1 ou no máximo 2.

Uma característica importante deste modelo é que, quando $d = 0$, o modelo ARIMA inclui o modelo ARMA(p,q), como um caso especial, e também os modelos AR(p) e MA(q).

3.2.3 Classificação de acordo com o número de parâmetros envolvidos

De acordo com o número de parâmetros envolvidos nos polinômios dos modelos de série temporal, há duas classes de modelos:

- paramétricos, quando o número de parâmetros é finito; e
- não-paramétricos, quando o número de parâmetros é infinito.

Os paramétricos são analisados no domínio do tempo, enquanto os não-paramétricos são analisados no domínio da freqüência.

Neste trabalho são utilizados somente modelos paramétricos. Na análise de séries temporais que utilizam modelos paramétricos, a função de autocorrelação desempenha um papel importante, principalmente no modelo ARIMA.

3.3 Metodologia de previsão

Um modelo matemático que descreve uma série temporal não conduz, necessariamente, a um procedimento (ou método) de previsão. Será necessário especificar uma função de perda, além do modelo, para se chegar ao método. Uma função de perda que é utilizada freqüentemente é o **erro quadrático médio** (MORETTIN; CASTRO TOLOI, 1981).

Deste modo, o procedimento básico para fazer previsão consiste em modelar a série, estimar seus parâmetros de acordo com a função de perda, para então poder fazer a previsão de z_t, h passos à frente.

Os métodos de previsão são classificados como (MORETTIN; CASTRO TOLOI, 1981): métodos automáticos de previsão, normalmente de baixo custo e qualidade razoável; e métodos interativos de previsão, que oferecem melhor qualidade na previsão, mas maior custo de processamento e interação do usuário.

3.3.1 Método automático

Nesta categoria, os métodos, em geral, realizam previsões passando pelos seguintes estágios:

1. primeiro, estima-se os parâmetros do modelo com base nos valores passados da série (regressão); e

2. segundo, conhecida a função, a previsão é obtida por extrapolação.

Fazem parte desta categoria métodos como o de médias móveis simples (MMS) e dupla (MMD, ou *double moving average* - DMA), alisamento exponencial simples (AES, ou *low pass filter* - LPF), alisamento exponencial linear de Brown (AELB, ou BROWN) e alisamento exponencial quadrático de Brown (AEQB), dentre outros. Maiores detalhes sobre os métodos podem ser obtidos através de Morettin e Toloi (1981) ou Montgomery e Johnson (1976).

3.3.2 Método interativo

Segundo Box, Jenkins e Reinsel (1994), a modelagem dos dados, via uma série temporal, deve ser realizada por uma abordagem interativa composta de três estágios:

1. **identificação do modelo** - governado por análises empíricas e tentativas, este estágio procura identificar, através do emprego da função de autocorrelação da amostra e da função de autocorrelação parcial da amostra, o modelo de série temporal que melhor representa a série sob análise;

2. **estimativa dos parâmetros** do modelo identificado - neste estágio, os dados previamente amostrados são utilizados para inferir estimativas consistentes dos parâmetros, através da resolução de um sistema de equações lineares;

3. **verificação da adequação dos parâmetros estimados** (*re-fit*) - se o modelo se mostrar inadequado, o que pode ocorrer devido a mudanças no comportamento do sistema ao longo do tempo, ele deve ser modificado e aprimorado.

Somente após os três estágios da modelagem dos dados é que a previsão pode ser feita. Além disto, de tempos em tempos, os ítens 2 e 3 devem ser refeitos para re-adequar o modelo aos dados correntes.

3.3.3 Algoritmo geral de previsão

No geral, um algoritmo típico de previsão de valores tem a seguinte forma:

```
sempre que chegar uma nova medida faça {
   atualize o histórico;
```

```
   reajuste o modelo ao histórico;
   construa um novo preditor;
   execute o preditor;
}
```

onde o histórico é uma janela de valores observados.

Este algoritmo é o que se encontra nos métodos automáticos de previsão, onde o custo para re-ajustar (*re-fit*) os parâmetros da função de previsão costuma ser pequeno. Em modelos onde este custo não deve ser despresível, tal como nos modelos ARIMA, é necessário diminuir a periodicidade do reajuste segundo algum critério. O resultado é um algoritmo com a seguinte forma (DINDA; O'HALLARON, 1999a):

```
sempre que chegar uma nova medida faça {
   atualize o histórico;
   se (critério para reajuste) {
      reajuste o modelo ao histórico;
      construa um novo preditor;
   }
   execute o preditor;
}
```

3.4 Conclusões Parciais

Uma série temporal é um modelo matemático projetado para representar uma amostragem ordenada no tempo, a qual se desenvolve em períodos eqüidistantes, e que pode ser utilizado para realizar previsões.

Os atrasos de comunicação amostrados por um detector de defeitos, ao serem medidos/observados em intervalos de tempo adjacentes de tamanho t, podem ser modelados através de uma série temporal Z_t composta pelas observações $z_1, z_2, ..., z_n$. Este é o princípio de amostragem que norteia o uso desta teoria neste trabalho.

Com um número de parâmetros finito, geralmente 1 ou 2, equações de previsão podem ser derivadas da classe de modelos paramétricos, os quais têm sua análise realizada no domínio do tempo.

O modelo ARIMA(p, d, q), por ser capaz de descrever de maneira satisfatória tanto séries estacionárias como não estacionárias homogêneas (que não apresentam comportamento explosivo), bem como descrever processos estocásticos de qualquer ordem (o número de parâmetros do modelo ajusta-se às características do processo) permite construir um método de previsão adaptativo.

Esta possibilidade revela-se interessante, do ponto de vista do presente trabalho, visto que habilita um detector de defeitos a mudar o modelo de previsão do atraso de comunicação quando necessário.

4 REPRESENTAÇÃO E CARACTERÍSTICAS DO ATRASO DE COMUNICAÇÃO

No capítulo 2 descrevem-se várias estratégias utilizadas para ajustar dinamicamente o *timeout* nos períodos em que o componente monitorado não está sob suspeita. Dentre as estratégias, as que mais se destacam são as que fazem uso de um histórico de observações do atraso (seções 2.3.4, 2.3.6 e 2.3.7). Entretanto, constata-se que, em todos os casos, é assumida a hipótese de que o comportamento do atraso de comunicação varia em torno de uma média populacional sem apresentar tendência, ou seja, que o comportamento é estacionário. Além disto, pelos modelos de previsão adotados (média aritmética mais variância ou média ponderada mais desvio médio) as observações do atraso de comunicação são consideradas observações independentes e não correlacionadas no tempo, ou seja, considera-se que entre amostras do atraso de comunicação não há autocorrelação[1]. Neste capítulo, avalia-se o uso destas hipóteses (estacionariedade[2] e independência das amostras) no contexto de uma modelagem via séries temporais (BOWERMAN; O'CONNEL, 1993; BOX; JENKINS; REINSEL, 1994).

Como uma série temporal é um modelo matemático para descrever uma amostragem periódica, este capítulo esta organizado em duas seções. Na seção 4.1, demonstra-se que o atraso de comunicação, mensurado em função do tempo de ida e volta de uma mensagem (*round-trip time* - *rtt*) sobre uma rede de longa distância (*wide-area network* - WAN), embora seja amostrado de maneira aperiódica, pode ser modelado como uma série temporal se for aplicada uma interface que converta a amostragem aperiódica numa amostragem periódica. A seção 4.1 também discute e especifica uma interface para modelar o atraso de comunicação como uma série temporal. Na seção 4.2, aplicando-se a interface especificada para modelar um conjunto de aproximadamente 2 milhões de amostras[3] do *rtt*, avalia-se o comportamento da série temporal do *rtt* e verifica-se que a série temporal apresenta um comportamento predominantemente estacionário somente no período noturno, das 20:00h às 9:00h. No restante do dia, das 9:00h às 20:00h, o comportamento é predominantemente não estacionário. Um comportamento não estacionário é caracterizado por apresentar uma autocorrelação significativa ($|r_k| > 0,7$) entre as amostras do *rtt* por vários passos (VANDAELE apud VASCONCELLOS; ALVES, 2000, p. 216).

[1] Autocorrelação é a correlação serial entre amostras da mesma variável aleatória.

[2] Estacionariedade é definida como uma qualidade de um processo estocástico na qual os parâmetros estatísticos (média e variância) não mudam com o tempo.

[3] As amostras correspondem a 23 amostragens de 24 horas cada. Detalhes sobre a coleta são apresentados na seção 4.2.1.

4.1 Modelagem do atraso de comunicação de ida e volta como uma série temporal

Embora o atraso de comunicação percebido sobre redes de longa distância seja um sinal variante no tempo (BAUER; SICHITIU; PREMARATNE, 2001), o que resulta numa amostragem aperiódica do atraso de comunicação de ida e volta (rtt), nesta seção, mostra-se que as observações do rtt mensuradas por um detector de defeitos podem ser modeladas como uma série temporal, um modelo de previsão para amostragens periódicas.

A demonstração aqui apresentada está organizada em três partes: primeiro, na subseção 4.1.1, descreve-se a natureza do rtt sobre uma WAN e constrói-se um modelo de tempo discreto para representá-lo; segundo, na subseção 4.1.2, discute-se através de uma abordagem intuitiva como se pode incluir as observações do rtt numa seqüência do tipo série de tempo; e terceiro, na subseção 4.1.3, especifica-se formalmente uma interface que permite representar a amostragem aperiódica do rtt como uma série temporal.

4.1.1 Modelo discreto do rtt

Seja um serviço de monitoramento de defeitos do estilo *pull* operando sobre uma WAN, ou sobre a Internet. De maneira abstrata, este serviço pode ser visto como um sistema de controle, tal como o ilustrado na figura 4.1. No serviço/sistema, o processo monitor p envia, a cada intervalo de tempo t_i, uma mensagem req (*request*) para um processo monitorável q, localizado num outro computador, e aguarda a mensagem de ack (*acknowledgement*) correspondente. Neste contexto, as características do atraso de comunicação observado por p são definidas pelas características dos atrasos das mensagens req e ack, ou seja, por d_s e d_r respectivamente. Como os atrasos d_s e d_r sobre WANs são por natureza indeterminados e variantes no tempo (BAUER; SICHITIU; PREMARATNE, 2001), pode-se deduzir que a amostragem do rtt observado pelo monitor de defeitos p será aperiódica, independente do comportamento dos processos p e q.

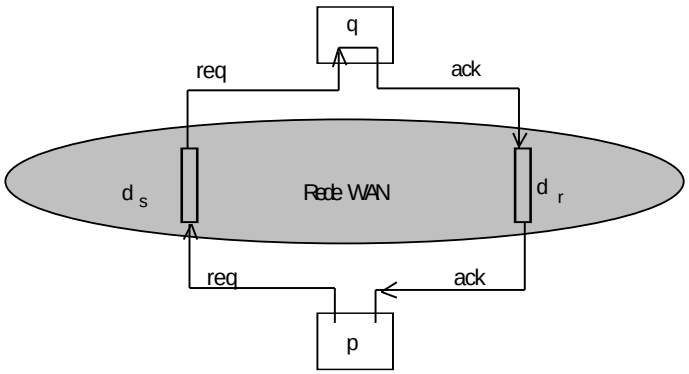

Figura 4.1: Modelo de comunicação do detector de defeitos.

A figura 4.2 ilustra a amostragem aperiódica do rtt ao longo de uma linha temporal, quando o processo monitor p envia uma mensagem req ao processo monitorável q a cada 1000 ms, iniciando no instante de tempo t. A parte (a) da figura, ilustra a magnitude de cada rtt observado pelo processo monitor, representada na figura por barras. A parte (b), ilustra a relação entre os instantes de envio das men-

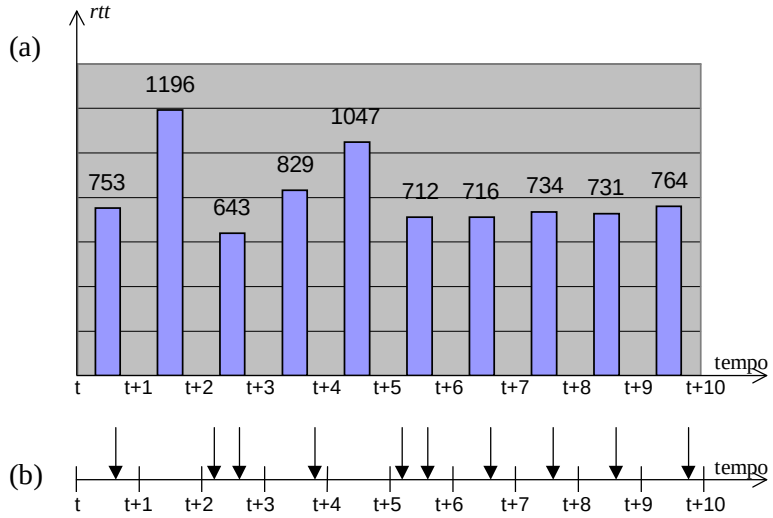

Figura 4.2: Amostragem dos atrasos de comunicação.

sagens req (instantes $t, t+1, t+2, ...$) e os instantes de recebimento das mensagens ack (setas). Observa-se que, embora o envio das mensagens req ocorra com periodicidade constante, os recebimentos das mensagens de ack não são regulares e geram intervalos de tempo nos quais nenhuma, uma ou múltiplas, mensagens são recebidas, resultando em nenhuma, uma ou múltiplas medidas do rtt. Como resultado, tem-se uma seqüência aperiódica de medidas do rtt.

Como conseqüência, para poder representar uma amostragem aperiódica como uma série temporal, um modelo para amostragens periódicas, necessita-se transformar a amostragem aperiódica em periódica.

Para formalizar esta transformação, faz-se uma separação entre o atraso de comunicação, representado pelo rtt, e a interface utilizada para tratá-lo como uma série temporal, a qual serve de conversor para uma amostragem aperiódica.

Abstraindo a interface, a seguir, define-se, um modelo discreto para descrever o atraso de comunicação representado pelo rtt. A definição da interface é apresentada nas próximas duas subseções, 4.1.2 e 4.1.3.

Seja um modelo de comunicação de uma entrada e uma saída, tal como o da figura 4.3, onde $u(t)$ representa uma seqüência periódica de instantes de envio de mensagens req, definida como

$$\forall t \geq 0, u(t) = \bigcup t | t = \quad \text{instante de envio de} \quad req_t \tag{4.1}$$

$d(t)$ representa o atraso de comunicação, definido como

$$\forall t \geq 0, d(t) = rtt_t = d_s + d_r | d_s > 0 e d_r > 0, \quad \text{mas ambos desconhecidos} \tag{4.2}$$

e $v(t)$ representa uma seqüência aperiódica de instantes de recebimento de mensagens ack, definida como

$$\forall t > 0, v(t) = \bigcup t | t = \quad \text{instante de recebimento de} \quad ack_t \tag{4.3}$$

Neste modelo, t representa o instante de tempo associado à mensagem req_t, e $t + d(t)$ define o instante de tempo em que a mensagem ack_t é percebida por p. Adicionalmente, seja uma janela de tempo $\mathcal{T}$, tal que $V(\mathcal{T})$ representa o conjunto de atrasos de ida e volta rtt amostrados na janela de tempo $\mathcal{T}$.

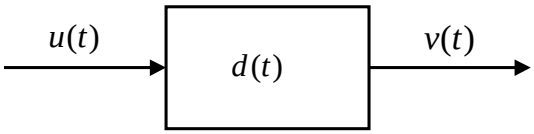

Figura 4.3: Modelo teórico para um atraso de comunicação.

Considerando um sistema de controle tal como o ilustrado na figura 4.1, onde o atraso é indeterminado e variante no tempo, tem-se que, em alguns momentos, o conjunto $V(\mathcal{T})$ pode conter mais do que um valor do rtt, enquanto em outros, pode não conter nenhuma amostra (situação ilustrada na parte (b) da figura 4.2). O normal seria conter uma amostra. Tais situações de anormalidade decorrem da natureza do atraso de comunicação $d(t)$ e da janela de tempo $\mathcal{T}$. A janela tem tamanho fixo, mas o atraso é variante no tempo.

Do ponto de vista de um detector de defeitos, $V(\mathcal{T})$ é a seqüência mais importante, pois ela contém os atrasos de comunicação percebidos pelo processo monitor p quando do monitoramento de um processo monitorável q. Assim, numa execução estável do detector de defeitos, se $\mathcal{T} = t_i$ e $\mathcal{T}$ começa no instante t, espera-se que $V(\mathcal{T}) = rtt_t$, onde rtt_t representa o tempo de ida e volta de um par de mensagens de controle iniciado no instante de tempo t. Por outro lado, toda mensagem req_t com $d(t) > \mathcal{T}$ irá gerar medidas do rtt_t numa janela de tempo futura $\mathcal{T} + m.\mathcal{T}$, onde $m = \lceil d_t/\mathcal{T} \rceil$. Logo, considerando o modelo de atraso de comunicação, define-se $V(\mathcal{T})$ como sendo:

$$V(\mathcal{T}) = \bigcup_{t \in U(\mathcal{T})} v(t) - u(t) \tag{4.4}$$

onde $U(\mathcal{T})$ representa o conjunto de instantes de tempo que produzem saídas na janela de tempo $\mathcal{T}$, a qual pode ser definida como:

$$U(\mathcal{T}) = \{t \mid t + d(t) \in \mathcal{T}, t \geq 0\} \tag{4.5}$$

Deseja-se ressaltar que as definições recém apresentadas são independentes do modelo de sistema distribuído adotado (assíncrono, síncrono, ou suas variações). Se os limites de $d(t)$ não forem conhecidos, tem-se um sistema assíncrono e $d(t)$ é definido como sendo:

$$0 < d_{min} \leq d(t) \leq \infty \tag{4.6}$$

onde d_{min} não é conhecido mas indica que os sistemas assíncronos não são infitamente rápidos. A inclusão da ocorrência $d(t) = \infty$ permite considerar perdas de mensagens (mensagens que nunca irão chegar). Logo, para desconsiderar a perda de mensagens basta fazer $d(t) < \infty$ na equação 4.6.

Por outro lado, se os limites de $d(t)$ forem conhecidos, tem-se um sistema síncrono e $d(t)$ é definido como sendo:

$$0 < d_{min} \leq d(t) \leq d_{max} < \infty \tag{4.7}$$

onde d_{min} e d_{max} são conhecidos.

As equações 4.3 a 4.7 descrevem o modelo discreto proposto para a amostragem aperiódica do rtt. De uma maneira similar à abordagem de Bauer, Sichitiu e Premaratne (2001), o modelo proposto não faz qualquer hipótese sobre como os atrasos de comunicação devem ser interfaceados com o detector de defeitos, logo, descreve somente a natureza do próprio atraso.

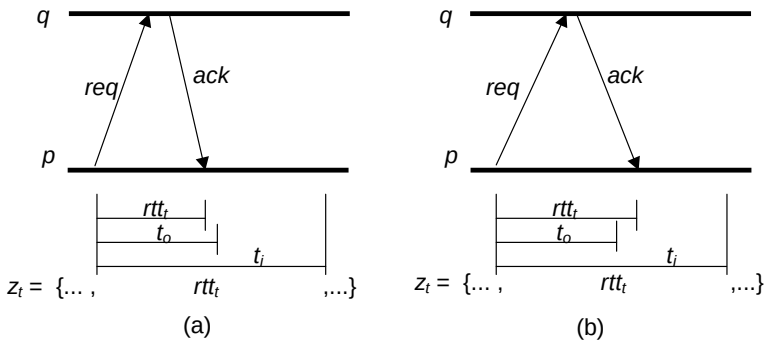

Figura 4.4: Possíveis cenários num período estável.

4.1.2 A interface para construir uma série de tempo

Na subseção anterior, definiu-se um modelo discreto para descrever a natureza do *rtt* observado por um detector de defeitos do estilo *pull*. Nesta seção, discute-se, de maneira intuitiva, como deve ser a interface que transforma uma amostragem aperiódica do *rtt* numa amostragem periódica, tendo como objetivo poder representá-la como uma série temporal. A definição formal da interface é apresentada na próxima subseção, 4.1.3.

Considerando as características de um detector de defeitos do estilo *pull*, onde há um envio de uma mensagem *req* a cada t_i unidades de tempo e o uso de um *timeout* t_o no aguardo pela mensagem *ack*, esta subseção inicia discutindo as características que uma interface deve ter quando o detector de defeitos opera num *período estável*, para então dedicar-se às características num *período instável*. No contexto aqui proposto, um período estável corresponde a um período no qual somente a mensagem *ack* esperada[4] chega ao processo monitor; ao contrário, um período instável corresponde a um período no qual mensagens podem ser perdidas ou chegar atrasadas, em relação ao tamanho do período.

A. Período estável

Seja $\mathcal{T} = t_i$ o tamanho do passo da série temporal e $t_o < t_i$. Assim, a cada passo $\mathcal{T}$, o processo monitor p envia uma mensagem req_t e espera pela mensagem ack_t correspondente. Se p recebe somente ack_t no passo $\mathcal{T}$, então $rtt_t < t_i$ e este passo corresponde a um período estável do detector de defeitos. Num período estável, dois possíveis cenários podem acontecer (figura 4.4):

(a) a medida do rtt_t é menor do que o *timeout* t_o, ou

(b) a medida do rtt_t é maior do que t_o mas menor do que o período de interrogação t_i.

Sabendo que uma série de tempo é uma seqüência cronológica de amostras, num período estável, pode-se fazer a medida do rtt_t corresponder a uma amostra da série temporal, independentemente do valor do *timeout*. Como resultado, a observação de n períodos estáveis adjacentes, gera uma série temporal $Z_t = \{rtt_1, rtt_2, ..., rtt_n\}$ das n medidas prévias do tempo de ida e volta.

[4] A mensagem esperada é a que está marcada com o número de seqüência correspondente ao último *req* sendo monitorado.

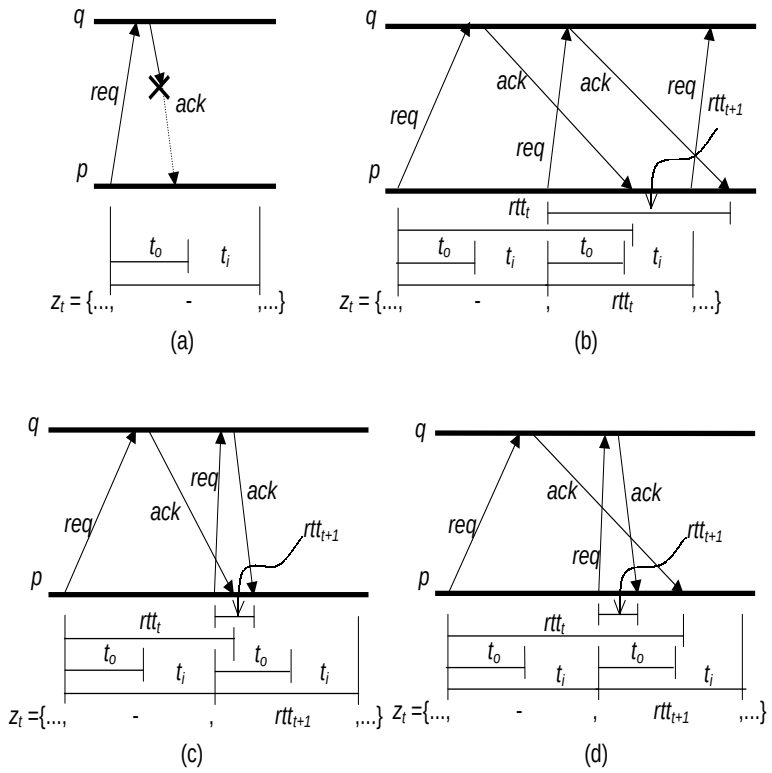

Figura 4.5: Possíveis cenários de um período instável.

B. Período instável

Um período instável de um detector de defeitos do estilo *pull* corresponde a um dos quatro cenários ilustrados na figura 4.5.

A figura 4.5a mostra um cenário onde uma mensagem de *ack* é perdida e, por conseqüência, o processo monitor não amostra o rtt no passo $\mathcal{T}$, ou seja, ocorre uma *lacuna* (*gap*) nas observações que deveriam ser representadas pela série temporal Z_t.

Para tratar a ocorrência de lacunas, ou seja, para eliminá-las da série temporal, é necessário adotar alguma *política de preenchimento*. De acordo com Ryan and Giles (1998), há pelo menos três diferentes políticas de preenchimento que podem ser aplicadas:

(*i*) *ignorar as lacunas* - esta política constrói a série temporal simplesmente suprimindo os valores não amostrados, ou seja, considerando a não amostragem do rtt_t, a série temporal resultante seria algo do tipo $..., rtt_{t-2}, rtt_{t-1}, rtt_{t+1}, ...$;

(*ii*) *preencher as lacunas com a última observação disponível* - neste caso, considerando a não amostragem do rtt_t, a série temporal resultante teria a seguinte forma $..., rtt_{t-1}, rtt_{t-1}, rtt_{t+1}, ...$; e

(*iii*) *preencher as lacunas de acordo com um método de interpolação linear.*

Uma quarta política também foi proposta por Nunes e Jansch-Pôrto (2002a):

(*iv*) *preencher as lacunas com um valor computado por* $z_t = (t_i * \overline{nag}) + z_{t-1}$, onde t_i é o período de interrogação do detector de defeitos e $\overline{nag}$ é a média do número de lacunas adjacentes observadas.

A primeira política é simples de implementar, pois não inclui processamento extra, e é a política que, segundo Ryan and Giles (1998), produz melhores resultados na análise da estacionariedade da série temporal. Introduzir valores não amostrados, tal como sugerido na segunda e terceira política, pode confundir o identificador de modelos ARIMA, dado que este algoritmo trabalha computando a estacionariedade da série e analisando suas funções de autocorrelação (vide seção 4.2 para mais detalhes sobre como é realizada a análise da estacionariedade de uma série temporal). A segunda política, embora simples, ao repetir a última amostra gera um traço horizontal (nível) constante de tamanho igual ao número de lacunas adjacentes. Similarmente, a terceira política, ao interpolar dois pontos gera uma inclinação linear de tamanho igual ao número de lacunas adjacentes. Como conseqüência, a inserção de valores de maneira forçada pode resultar em modelos preditivos equivocados que degradam a eficiência dos preditores baseados em séries temporais, pois estes consideram a dependência entre amostras adjacentes. Além disto, para fazer a interpolação linear, a terceira política necessita conhecer o próximo valor antes mesmo dele ter sido amostrado. A quarta política procura solucionar a falta de um valor futuro realizando uma interpolação não linear em função do número de lacunas adjacentes, mas recai no mesmo problema em relação a possibilidade de modelos preditivos equivocados.

Julgando que o tratamento das mensagens perdidas não deve ser realizado pelo preditor e sim pelo próprio algoritmo de detecção de defeitos, neste trabalho adota-se a primeira política, a qual é simples e eficiente.

Um cenário diferente é ilustrado na figura 4.5b. Neste cenário, observa-se a ocorrência de uma lacuna, causada por uma mensagem atrasada, e a chegada de somente uma mensagem na próxima janela de tempo $\mathcal{T}$ (a mensagem atrasada). Neste cenário, a lacuna pode ser resolvida como descrito acima, ou seja, suprimindo-a; já a recepção da mensagem atrasada pode ser resolvida inserindo na série temporal Z_t a medida do rtt correspondente ao ack, pois é o único observado no período. Se a única mensagem do período for uma mensagem antiga, ou seja, uma mensagem fora de ordem (com número de seqüência menor do que outra já recebida), a mensagem deve ser descartada, reportando este cenário ao de ocorrência de uma lacuna.

Olhando para os cenários (c) e (d) da figura 4.5, vê-se também que em alguns períodos podem ser observadas medidas múltiplas do rtt, decorrentes da recepção de diversas mensagens de ack. Nestes casos, há duas políticas que podem ser adotadas:

(i) inserir na série temporal somente uma observação do rtt; ou

(ii) inserir na série temporal, de maneira ordenada, todas as observações válidas, ou seja, todas as observações que correspondem aos acks que chegaram.

Considerando o recebimento de m mensagens, onde $m > 1$, a adoção da primeira política implica em descartar amostras válidas do rtt. Este descarte pode tornar a série temporal menos representativa, uma vez que nela, para um dado tempo de observação, haverá um número menor de amostras de rtt. Além disto, a adoção desta política implica em decidir qual amostra deve ser considerada e quais devem ser descartadas. Uma abordagem é manter somente o primeiro valor amostrado. Assim, ganha-se em desempenho pois, tão logo chegue uma mensagem, a interface pode inserir o respectivo rtt na série temporal. Entretanto, sempre que houver atrasos em bloco sem prejuízo da ordem das mensagens, situação freqüentemente

observada no tráfego sobre a Internet, esta abordagem tende a inserir somente o maior rtt na série temporal, descartando amostras que possivelmente seriam mais efetivas[5]. Uma abordagem complementar à anterior é manter o rtt correspondente ao mais recente req enviado. Entretanto, no caso de perdas de mensagens, deve-se esperar até o final da janela de tempo $\mathcal{T}$ para poder inserir algum valor na série, o que significa que o detector sempre deveria esperar por um tempo aproximadamente igual a t_i para poder processar a informação de rtt, o que é inaceitável. Logo, a adoção da primeira política não é uma boa escolha.

A segunda política difere da primeira por inserir na série o rtt correspondente a todas as mensagens de ack válidas (com número de seqüência maior do que a última recebida). Desta forma, valores válidos do rtt que extrapolam a janela de tempo $\mathcal{T}$ não são descartados, mas sim alocados às suas respectivas posições futuramente, aumentando a representatividade da série temporal. Logo, com esta política, as lacunas ignoradas corresponderão somente a mensagens perdidas e a mensagens que chegam fora de ordem.

Resumindo, para ilustrar o comportamento da interface adotada, a figura 4.6 sintetiza a atuação da interface num cenário onde 6 mensagens req são enviadas, uma mensagem ack é perdida e duas outras demasiadamente atrasadas. Na primeira janela de tempo, que começa no instante t e termina no instante $t+1$, p envia uma mensagem req a q e recebe a respectiva mensagem de ack, logo a interface insere na série temporal Z_t a medição do rtt_t. Em seguida, mantendo-se a periodicidade do envio de mensagens req, ocorrem três períodos instáveis no detector de defeitos. A mensagem ack_{t+1} é perdida e as mensagens ack_{t+2} e ack_{t+3} são atrasadas, logo nenhuma medida do rtt é inicialmente realizada entre os instantes de tempo $t+1$ e $t+4$. Entretanto, ao receber a mensagem ack_{t+2}, a interface no monitor p identifica que ack_{t+1} foi perdida ou está demasiadamente atrasada. Em ambos os casos, a política de preenchimento de lacunas deve ser aplicada; no caso, a lacuna é ignorada e o rtt_{t+2} é inserido na posição z_{t+1} da série temporal Z_t. Atuação similar faz $z_{t+2} = rtt_{t+4}$. Por outro lado, quando ack_{t+3} é recebida, a interface no processo monitor identifica, através da comparação do seu número de seqüência com o da última mensagem recebida, que ela é uma mensagem atrasada (fora de ordem). Logo, a amostra é simplesmente descartada. Como resultado, tem-se uma série temporal com a melhor representatividade possível. O algoritmo que implementa esta interface é apresentado na figura 4.7, onde π representa o conjunto de processos monitoráveis pelo processo monitor p, e $Z_{p,q}$ e $expectedTimestamp$ representam a série temporal Z_t e o número de seqüência esperado para a próxima mensagem, respectivamente, ambos mantidos por p para a amostragem dos rtt com q.

4.1.3 Descrição formal da interface

Num detector de defeitos do estilo *pull*, um processo monitor p envia a um processo monitorável q uma mensagem req, a cada t_i unidades de tempo. Para cada mensagem req o detector espera receber uma mensagem ack num tempo $t_o < t_i$. Para cada mensagem ack recebida, uma medida do rtt é realizada e pode servir para inferir futuros atrasos. Neste contexto, define-se a seguir, a interface em função do modelo discreto do atraso proposto na seção 4.1.1.

[5]Por hipótese, $t_o < t_i$; logo, supõe-se que $rtt < t_i$. Como conseqüência, no atraso em bloco, a primeira mensagem resulta num rtt bem maior do que a segunda, que por sua vez resulta num rtt bem maior do que a terceira e assim por diante.

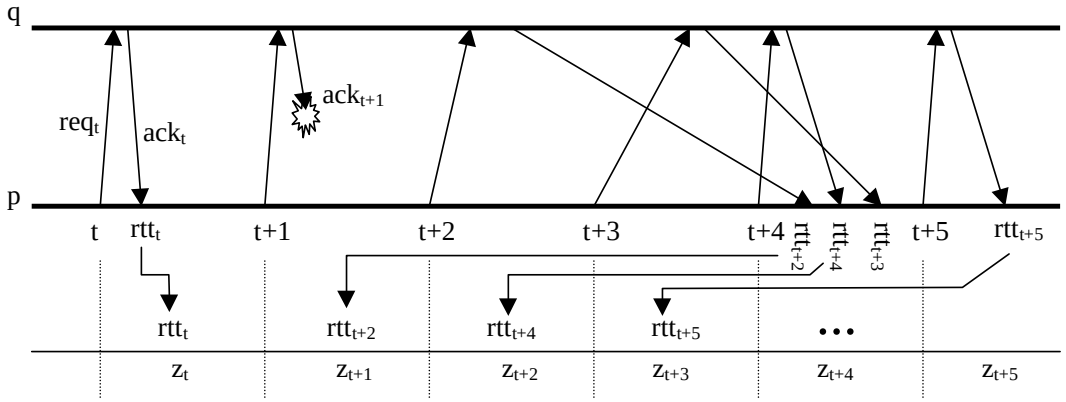

Figura 4.6: Construção da série temporal através de uma interface que converte uma amostragem aperiódica em periódica.

Todo processo p executa:

$\forall q \in \Pi : Z_{p,q} \leftarrow \{\}$
$\forall q \in \Pi : expectedTimestamp \leftarrow 0$
$i = 0$

loop
  espera por uma mensagem ack_t de um processo q
  if $timestamp_{ack_t} \geq expectedTimestamp$ **then**
    $Z_{p,q}[i] \leftarrow rtt_t$ computado a partir da mensagem ack_t
    $expectedTimestamp \leftarrow timestamp_{ack_t} + 1$
    $i \leftarrow i + 1$
  end if
end loop

Figura 4.7: Algoritmo que implementa a interface.

Seja $card(V(\mathcal{T}))$ a cardinalidade do conjunto de saída $V(\mathcal{T})$ (definido pela equação 4.4), isto é, o número de atrasos de comunicação rtt coletados no período $\mathcal{T} = t_i$, o qual começa no instante de tempo t.

Seja $B(\mathcal{T})$ o conjunto de instantes de tempo referentes às mensagens inválidas recebidas no período $\mathcal{T}$, onde mensagens inválidas são mensagens duplicadas ou mensagens com *timestamp* inferior ao do ack esperado. Sendo $U(\mathcal{T})$ o conjunto de instantes de tempo que produzem saídas na janela de tempo $\mathcal{T}$ (seção 4.1.1), tem-se que $B(\mathcal{T}) \in U(\mathcal{T})$.

Como mensagens inválidas não devem ser inseridas na série temporal, todas as mensagens inválidas devem ser descartadas pela interface, logo

$$\forall t \in B(\mathcal{T}), \quad \text{desconsidere} \quad ack_t \tag{4.8}$$

Se após enviar req_t o processo p receber somente ack_t no período $\mathcal{T}$ (cenário correspondente ao período estável ilustrado na figura 4.4), então t não pertence a $B(\mathcal{T})$ e $card(V(\mathcal{T})) = 1$. Em outras palavras existe somente um t tal que $t \in U(\mathcal{T}) - B(\mathcal{T})$. Como resultado, tem-se que $V(\mathcal{T}) = \{rtt_t\}$. Similarmente, se ao aguardar um ack_t, o processo p receber somente um ack_i, onde $i \geq t$, então $card(V(\mathcal{T})) = 1$ e $V(\mathcal{T}) = \{rtt_i\}$. Este cenário corresponde ao cenário instável ilustrado na figura 4.5(b). Logo, após aplicar 4.8,

$$\forall \mathcal{T} \mid card(V(\mathcal{T})) = 1, \quad \text{faz-se} \quad z_t = rtt_i \quad \text{sempre que} \quad i \geq t \tag{4.9}$$

Por outro lado, numa situação onde nenhum ack for recebido no período $\mathcal{T}$, situação de ocorrência de uma lacuna (figura 4.5(a)), $card(V(\mathcal{T})) = 0$. Como conseqüência, $V(\mathcal{T}) = \emptyset$ e nenhum valor é inserido na série temporal Z_t no período $\mathcal{T}$. Logo

$$\forall \mathcal{T} \mid card(V(\mathcal{T})) = 0, \quad \text{nenhuma inserção é feita em} \quad z_t \tag{4.10}$$

Numa situação onde somente mensagens inválidas são recebidas, todas as mensagens são descartadas ao aplicar 4.8 e $card(V(\mathcal{T})) = 0$. Salienta-se que, neste caso, a interface recai no caso da equação 4.10.

Numa situação onde mensagens válidas e inválidas são recebidas no período $\mathcal{T}$, se após descartar todas as mensagens inválidas, restar somente uma mensagem válida ($card(V(\mathcal{T})) = 1$), o comportamento da interface segue a equação 4.9. Por outro lado, se após descartar todas as mensagens inválidas $card(V(\mathcal{T})) > 1$, tal como ocorre nos cenários instáveis (c) e (d) da figura 4.5, então todos os rtts correspondentes aos acks válidos devem ser inseridos na série temporal (figura 4.6). Logo

$$\forall \mathcal{T} \mid card(V(\mathcal{T})) > 1, \quad \text{faz-se} \quad z_i = rtt_t \forall t \in U(\mathcal{T}) - B(\mathcal{T}) \tag{4.11}$$

onde, para cada t, o índice i é incrementado de uma unidade.

As equações 4.8 a 4.11 definem formalmente o comportamento da interface necessária para converter uma amostragem aperiódica numa amostragem periódica. Tal interface pode ser utilizada na construção de uma série temporal que representa a amostragem do rtt num serviço de monitoramento com tempo de interrogação fixado.

4.2 Análise da estacionariedade do rtt

Conforme discutido no capítulo 2, os modelos preditivos utilizados na adaptação do *timeout* de detectores de defeitos consideram que a previsão dos atrasos de

comunicação pode ser realizada com base no último valor amostrado ou em estatísticas tradicionais como média e variância de amostras independentes entre si.

Com o propósito de utilizar modelos preditivos baseados na teoria de séries temporais, nesta seção, analisa-se o comportamento das séries temporais que representam o atraso de comunicação sobre redes do tipo WAN. Como resultado, observa-se que os atrasos de comunicação numa WAN, ao invés de serem independentes uns dos outros, freqüentemente apresentam correlação significativa ($| r_k | > 0,7$) nos primeiros 5 passos, indicando uma parcela de comportamentos não estacionários e não apenas estacionários. Tal resultado motivou a exploração de modelos preditivos de segunda ordem (utilizados quando a amostragem apresenta tendência determinística) para séries não estacionárias, conforme será discutido no próximo capítulo.

Para analisar a estacionariedade dos atrasos de comunicação, foram coletadas amostragens diárias do *rtt* observado por um detector de defeitos do estilo *pull*, executando numa rede de longa distância, entre a UFRGS e três nós externos instalados nas universidades federais de Santa Maria, da Paraíba (atualmente Campina Grande) e do Pará, a partir daqui referidos como UFSM, UFCG e POP/PA, respectivamente. Os dados foram então modelados como séries temporais aplicando a interface definida na seção anterior, e analisados de acordo com a metodologia para análise de séries temporais definida por Box, Jenkins e Reinsel (1994) e Bowerman e O'Connell (1993). No primeiro, é definida a metodologia básica, enquanto, no último, são apresentadas algumas regras experimentais que permitem a análise automática. O resultado mostra que, na maioria dos casos, o comportamento do atraso de comunicação é fortemente correlacionado e não estacionário. O resultado também mostra que as séries não estacionárias podem ser transformadas em estacionárias diferenciando a série original apenas uma vez. O número de vezes corresponde ao grau de não estacionariedade da série (ordem de integração).

A seção 4.2.1 descreve como foi realizada a coleta de dados, a seção 4.2.2 apresenta os conceitos e metodologia utilizados na determinação da ordem de integração da série temporal, ou seja, na determinação do grau de estacionariedade da série, e a seção 4.2.3 apresenta os resultados da análise da estacionariedade.

4.2.1 Coleta de dados

As amostragens utilizadas neste trabalho foram geradas por um programa distribuído de monitoramento de acessibilidade executado sobre o *backbone* da Rede Nacional de Pesquisa (RNP) e da rede de pesquisa gaúcha (rede Tchê), conforme ilustra a figura 4.8.

O programa de monitoramento implementa um detector de defeitos do estilo *pull* operando com um único processo monitor, localizado sobre o nó UFRGS, e com três processos monitoráveis, localizados nos nós UFCG, UFSM e POP/PA. Periodicamente, a cada segundo, o processo monitor envia uma mensagem de solicitação de estado, mensagem *req*, para cada um dos três processos monitoráveis e aguarda o recebimento de uma mensagem de reconhecimento, mensagem *ack*. A cada recebimento de um *ack*, uma amostra do tempo de ida e volta (*rtt* da mensagem de monitoração) é computada e inserida na série temporal que representa a amostragem em questão. A conversão da amostragem aperiódica do *rtt* numa amostragem periódica é realizada pela interface implementada pelo algoritmo 4.7.

Ao todo, foram coletadas 23 amostragens, de 24 horas cada, assim distri-

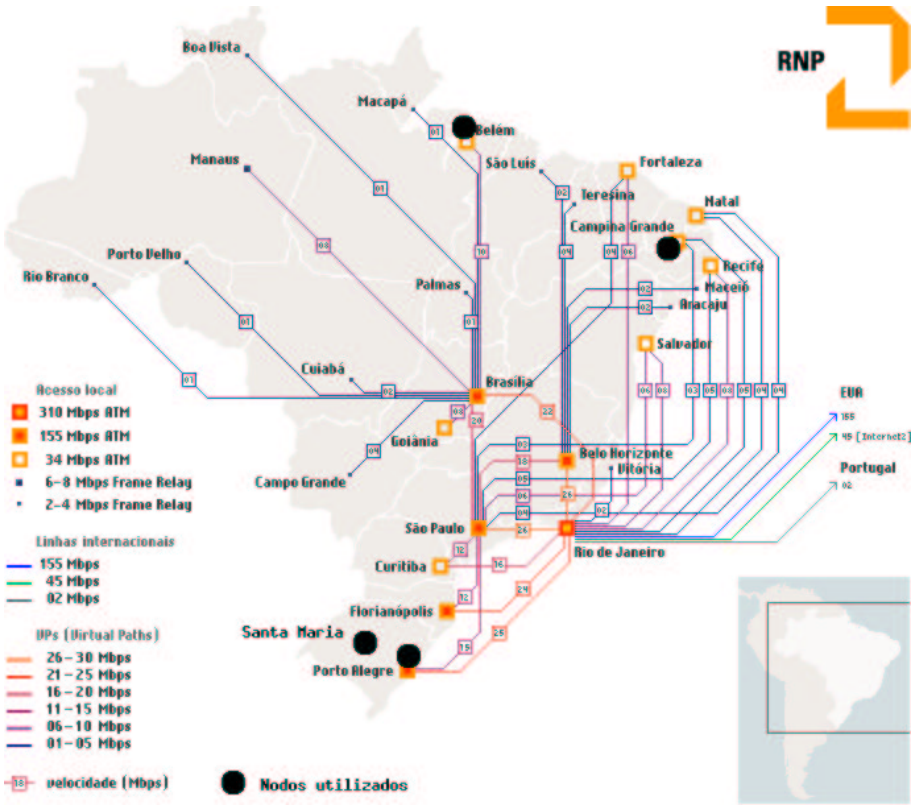

Figura 4.8: Distribuição geográfica dos nós utilizados na coleta de dados.

buídas: 9 na comunicação UFRGS-POP/PA, 8 na comunicação UFRGS-UFCG e 6 na comunicação UFRGS-UFSM. A tabela 4.1 sintetiza as amostragens, indicando a quantidade de amostras e a porcentagem de perdas de mensagens de cada uma. Das 23 amostragens, 6 correspondem a finais de semana (dias 16, 17, 23 e 24) e 17, a dias úteis. Algumas amostragens tiveram de ser descartadas devido a interrupções causadas por defeitos diversos (reinicialização de alguma das máquinas envolvidas ou indisponibilidade prolongada dos serviços de rede).

4.2.2 Metodologia da análise

Para verificar a estacionariedade do rtt segue-se a metodologia de Box, Jenkins e Reinsel (1994), a qual consiste na análise empírica da função de autocorrelação amostral (acf). A automatização do processo de verificação usa as regras experimentais propostas por Bowerman e O'Connell (1993) e por Vandaele (apud VASCONCELLOS; ALVES, 2000, p. 216). Bowerman e O'Connell indicam regras para determinar a significância estatística da acf e Vandaele indica uma regra para terminar o processo de identificação mais rapidamente.

De acordo com Bowerman e Connell (1993, p. 444) o comportamento da acf pode ser determinado usando o teste t, ou estatística t, da autocorrelação amostral r_k. A estatística t é definida como sendo a relação entre r_k e seu erro padrão S_{r_k}, ou seja,

$$t_{r_k} = r_k / S_{r_k} \tag{4.12}$$

Tabela 4.1: Amostragens diárias coletadas.

comunicação	dia	amostras	perdas (%)
UFRGS-POP/PA	18/06	83425	3,44
	19/06	83434	3,43
	20/06	81713	5,42
	21/06	82263	4,79
	22/06	81571	5,59
	23/06	86288	0,13
	24/06	85792	0,78
	26/06	83488	3,37
	27/06	83410	3,46
UFRGS-UFCG	16/06	85866	0,62
	17/06	85418	1,14
	18/06	85054	1,56
	19/06	85553	0,98
	20/06	85975	0,49
	21/06	84980	1,87
	22/06	85573	0,96
	27/06	85495	1,05
UFRGS-UFSM	16/06	86193	0,24
	17/06	86336	0,07
	18/06	75567	12,54
	21/06	78999	8,57
	26/06	78478	9,17
	27/06	78277	9,40

onde S_{r_k} é definido por

$$S_{r_k} = \sqrt{1 + 2\sum_{j=1}^{k-1} r_j^2/n} \tag{4.13}$$

Conhecida a estatística t, a regra básica utilizada para determinar o comportamento da acf consiste em verificar a ocorrência ou não de significância estatística em cada passo k. Por experiência (BOWERMAN; O'CONNEL, 1993), se o valor absoluto da estatística t for maior ou igual a $1,6$ para passos curtos ($k \leq 3$), então r_k pode ser considerada estatisticamente significante, mas se $k > 3$, então o valor absoluto da estatística t deve ser maior do que 2 para r_k ser considerada significante.

Conhecendo a significância em cada passo da acf, pode-se verificar se ao longo de alguns passos a significância decai rapidamente ou lentamente. Bowerman e O'Connell ((1993) pg. 450) indicam que uma série temporal pode ser considerada não estacionária se o nível de significância estatística das autocorrelações r_k apresentar um decaimento extremamente lento (*dies down extremely slowly*) conforme aumentar o passo k, ou seja, se a acf **declinar muito lentamente**. Ao contrário, se o nível de significância estatística das autocorrelações **declinar lentamente** (*dies down slowly*) ou apresentar um *corte*[6] nos primeiros passos (para $k < 3$), a série temporal pode ser considerada estacionária.

Entretanto, é difícil especificar a taxa com que uma dada função de autocorrelação deva declinar para ser enquadrada num dos comportamentos recém expostos. Para resolver este problema, conjuntamente com as regras experimentais de Bowerman e O'Connell faz-se uso da regra experimental de Vandaele (apud VASCONCELLOS; ALVES, 2000, p. 216), a qual diz que se $\mid r_k \mid > 0,7$ para $k > 5$, a série pode ser considerada não estacionária.

Após classificar uma dada série temporal como não estacionária, a dificuldade

[6]Um corte significa apresentar autocorrelações muito significativas no(s) primeiro(s) passo(s) e, repentinamente, apresentar autocorrelação nula em todos os demais passos. Em outras palavras, indica a ocorrência de uma descontinuidade na acf.

é que não há uma solução exata para determinar a ordem de integração (GOURIEROUX; MONFORT, 1990; BOX; JENKINS; REINSEL, 1994). O método não determinístico de Box, Jenkins e Reinsel para determinar a ordem de integração d de uma série temporal não estacionária consiste em assumir como hipótese que é possível transformar a série não estacionária numa série estacionária diferenciando-a d vezes; ou seja, constrói-se uma nova série temporal fazendo toda amostra z_t da nova série ser igual a $z_t - z_{t-1}$ da série não estacionária d vezes, até que a nova série seja estacionária. Entretanto, mesmo não determinístico, o método de Box, Jenkins e Reinsel permite uma boa solução aproximada (BOWERMAN; O'CONNEL, 1993), sendo assim adotado neste trabalho.

Em síntese, a verificação do grau de estacionariedade da série temporal é realizada em duas etapas: verificação da estacionariedade com base na função de autocorrelação da série amostrada; e, caso a série seja não estacionária, determinação da ordem de integração realizada com base no número de diferenciações necessárias para tornar a série estacionária. A seguir são apresentados os algoritmos utilizados para implementar este método.

Os algoritmos

De acordo com a metodologia descrita acima e considerando uma série temporal $Z_t = rtt_1, rtt_2, ..., rtt_n$, a estacionariedade e a ordem de integração das séries coletadas foram estimadas pelo algoritmo empírico apresentado na figura 4.9. O algoritmo recebe uma série temporal, computa a sua função de autocorrelação amostral e verifica o seu comportamento, para determinar se declina muito lentamente ou não. Enquanto o comportamento for de uma série não estacionária (acf declina muito lentamente), a série temporal é diferenciada e o procedimento é repetido para a nova série. A ordem de integração corresponde ao número de interações realizadas, desconsiderando a primeira.

No algoritmo, o procedimento $computaAcf()$ computa as autocorrelações amostrais em cada passo $k \in [1..n-1]$, de acordo com a equação 4.14:

$$r_k = \frac{\sum_{t=k+1}^{n}(rtt_t - \overline{rtt})(rtt_{t-k} - \overline{rtt})}{\sum_{t=1}^{n}(rtt_t - \overline{rtt})^2} \tag{4.14}$$

O procedimento $diferencia$ realiza a primeira diferença da série temporal fazendo $z_t^1 = z_t - z_{t-1}$, onde z_t e z_t^1 correspondem às observações, no instante t, da série original e diferenciada, respectivamente, e z_{t-1} corresponde à observação da série original no instante $t-1$. Caso seja necessária uma segunda diferenciação da série, ela é obtida fazendo $z_t^2 = z_t^1 - z_{t-1}^1$, e assim por diante. Considerando uma série temporal com n observações, ao aplicar a diferenciação para todo $t = 1, 2, ..., n$, obtém-se uma série diferenciada de tamanho $n - d$.

A característica empírica do algoritmo 4.9 vem da identificação do comportamento da acf, se declina muito lentamente ou não, realizada pelo procedimento $verificaDecaimento()$. Este procedimento, implementado pelo algoritmo 4.10, utiliza-se das regras experimentais de Bowerman e Connell, para determinar se há ou não significância estatística num dado passo, e da regra experimental de Vandaele, para considerar se a série é estacionária ou não. Em síntese, ele testa se r_k é significante para todos os primeiros 5 passos e se $|r_6| > 0,7$. Se as duas condições forem verdadeiras, então o algoritmo classifica o comportamento da acf como $declinaMuitoLentamente$.

```
inicialização:
    declinaMuitoLentamente ← 1
    comportamento ← 0
    ordem ← 0
    serieTemporal ← valores coletados
    acf ← ∅

acf ← computaAcf(serieTemporal)
comportamento ← verificaDecaimento(acf)
while comportamento == declinaMuitoLentamente do
  serieTemporal ← diferencia(serieTemporal)
  ordem ← ordem + 1
  acf ← computaAcf(serieTemporal)
  comportamento ← verificaDecaimento(acf)
end while
if ordem == 0 then
  série temporal é estacionária
else
  série temporal é não estacionária e ordem de integração = ordem
end if
```

Figura 4.9: Algoritmo para determinar a estacionariedade e a ordem de integração.

```
proc verificaDecaimento(acf):

inicialização:
    n ← número de elementos da acf

for all k ∈ {1, 2, 3, 4, 5} do
  S_{r_k} ← (1 + 2 Σ_{j=1}^{k-1} r_j^2)^{1/2} / n^{1/2}
  t_{r_k} ← r_k / S_{r_k}
  if k ≤ 3 then
    if t_{r_k} < 1.6 then
      return 0
    end if
  else if t_{r_k} < 2 then
    return 0
  end if
end for
if |r_6| < 0.7 then
  return 0
end if
return 1
```

Figura 4.10: Algoritmo para determinar se acf declina muito lentamente.

Os conjuntos de dados

Para analisar a estacionariedade do *rtt*, derivou-se dos dados originalmente coletados, 4 conjuntos de dados:

c1 um conjunto com 23 séries temporais, composto pelas 23 amostragens diárias;

c2 um conjunto com 17 séries temporais, um subconjunto contido em c1, onde são considerados somente os 17 dias úteis, no intervalo das 6h às 24h;

c3 um conjunto com 408 séries temporais, formado a partir de c1, mas onde são considerados somente os 17 dias de semana e segmentos com uma hora de duração;

c4 um conjunto com 8160 séries temporais, geradas aleatoriamente do conjunto c2, onde são considerados segmentos com 12 diferentes tamanhos (variam de 300 a 3600 amostras). Para cada tamanho são consideradas 680 séries.

Para classificar o comportamento do *rtt* como estacionário ou não, bastaria analisar as 23 grandes amostragens diárias. Entretanto, como a maioria das computações é realizada no período diurno dos dias úteis, um conjunto de dados formado somente pelas observações durante os dias de semana das 6:00h às 24h é mais representativo. De maneira similar, uma análise por hora nos revela em que períodos o *rtt* tende a ser mais estacionário. Adicionalmente, ao considerar o histórico de amostras mantidas por um detector de defeitos, sobre o qual ele realizará previsões, torna-se importante avaliar a estacionariedade considerando conjuntos de tamanhos variados. Logo, a divisão do conjunto de dados original em 4 subconjuntos é importante e necessária para efetivar os resultados.

Graficamente, pode-se perceber que diferentes subconjuntos de um mesmo conjunto podem apresentar comportamentos distintos, demonstrando que uma série estacionária pode conter trechos não estacionários e uma série não estacionária pode conter trechos estacionários. Por exemplo, considere-se as amostragens ilustradas na figura 4.11. Pode-se observar que a série do *rtt* possivelmente[7] apresenta um comportamento estacionário durante a noite, mas não estacionário durante o dia, nos três canais de comunicação; mas ao analisar visualmente o comportamento entre as 12:00h e as 13:00h, a partir das mesmas séries, conforme ilustra a figura 4.12, percebe-se que embora a amostragem diária seja possivelmente não estacionária, um subconjunto dela possivelmente apresenta um comportamento estacionário. A análise dos 4 conjuntos de dados auxilia na verificação do comportamento sob diferentes circunstâncias.

4.2.3 Resultados da análise

Resultados para séries diárias - conjunto c1

[7]Embora a análise visual de uma série temporal auxilie na identificação do seu comportamento, a análise dificilmente pode ser conclusiva (BOX; JENKINS; REINSEL, 1994).

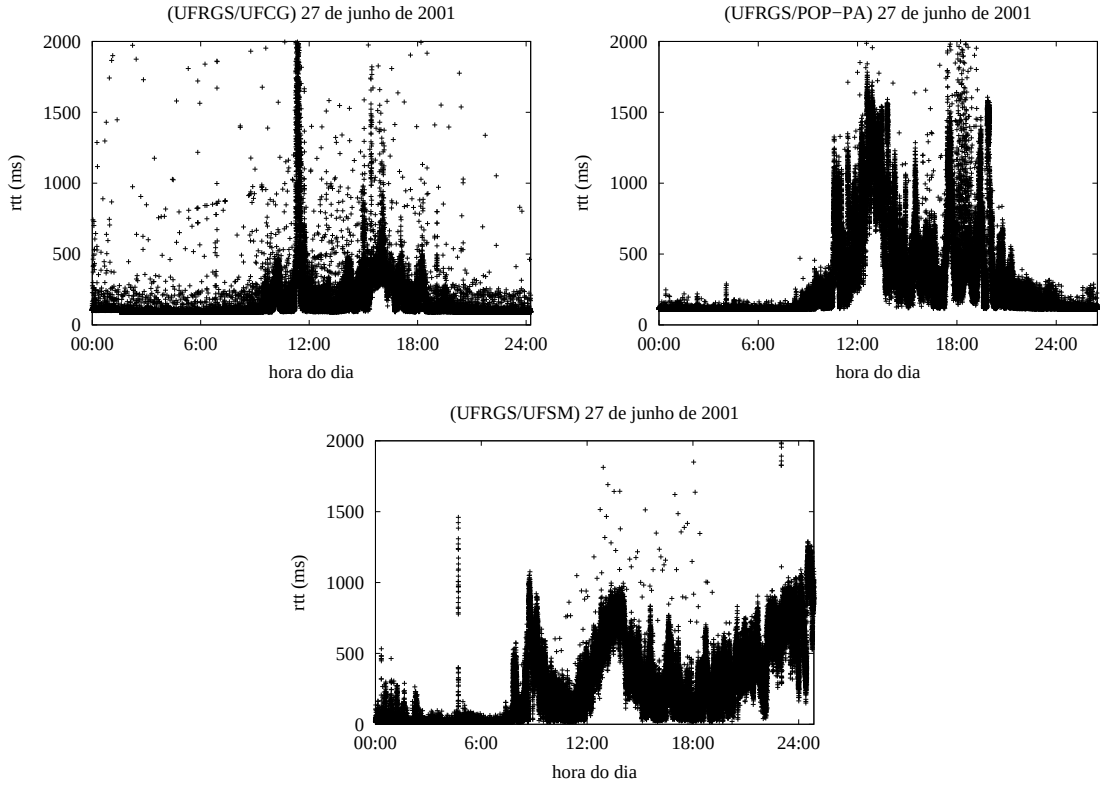

Figura 4.11: Séries temporais para amostragens do *rtt* durante 24 horas.

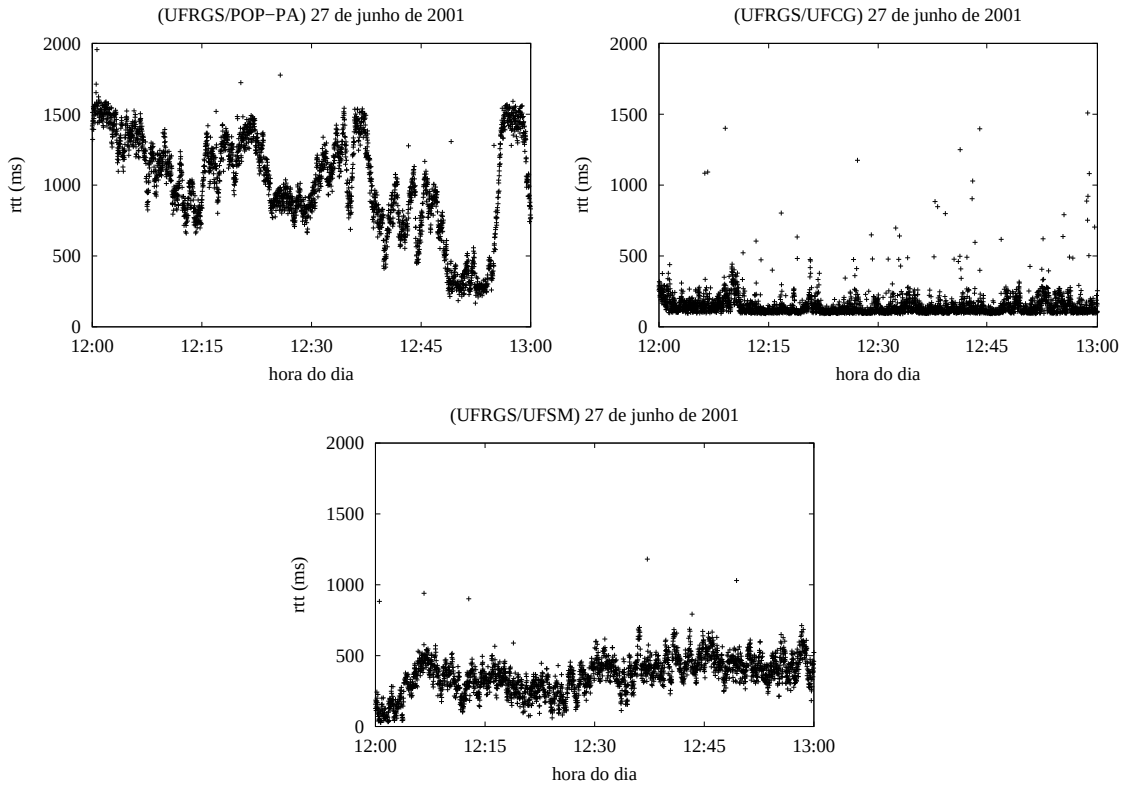

Figura 4.12: Séries temporais para amostragens do *rtt* durante uma hora (12-13h).

Considerando as 23 séries temporais correspondentes às amostragens listadas pela tabela 4.1, observou-se que em 21 delas, a função de autocorrelação amostral declina muito lentamente, conforme ilustra a figura 4.13, o que identifica um comportamento não estacionário em 91,3% das séries diárias do conjunto c1.

A figura 4.13 ilustra a acf da série temporal correspondente à amostragem realizada no dia 18 sobre o canal de comunicação UFRGS-POP/PA. O formato da acf é similar para as outras séries, exceto para a do dia 24 (domingo) sobre a mesma comunicação, e para a do dia 18 (segunda-feira) sobre a comunicação UFRGS-UFSM, conforme pode-se observar na figura 4.14, onde a acf demonstra um comportamento estacionário, pois ela apresenta valores inferiores a $0,7$ já nos primeiros passos.

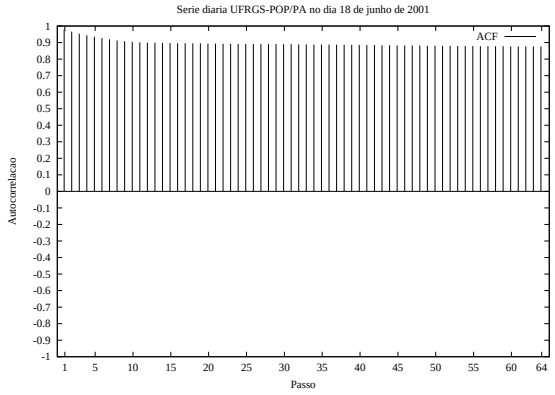

Figura 4.13: Função de autocorrelação amostral típica para séries não estacionárias.

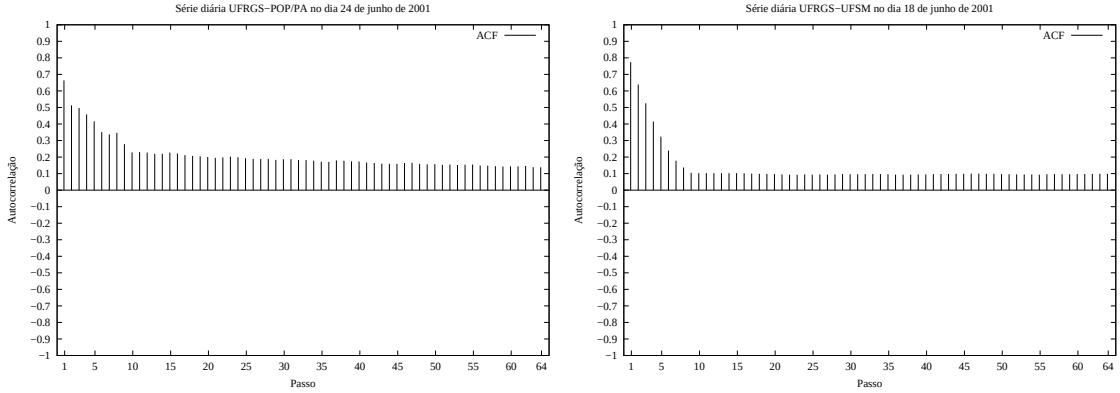

Figura 4.14: Funções de autocorrelação amostral para séries diárias estacionárias.

Todas as 21 séries não estacionárias tornaram-se estacionárias após seus elementos serem diferenciados uma única vez, indicando que as séries diárias do rtt podem ser tratadas por modelos estacionários desde que sejam convertidas para estacionárias através da técnica de diferenciação de seus elementos.

Em síntese, os resultados mostram que as séries diárias do rtt são, na maioria dos casos, não estacionárias, mas que basta uma diferenciação da série para torná-la estacionária; logo, isto indica que a metodologia de Box, Jenkins e Reinsel é suficiente para realizar previsões em séries diárias do rtt.

Resultados para séries de dias úteis - conjunto c2

Considerando o conjunto de dados c2, no qual as séries temporais correspondem às amostragens realizadas nos dias de semana no período das 6:00h às 24:00h, observou-se que das 17 séries temporais consideradas somente 1 apresenta comportamento estacionário, a série do dia 18 de junho na comunicação UFRGS-UFSM, a qual já apresentava este comportamento quando considerado o período da madrugada. Tal resultado mostra que a não estacionariedade observada nas séries do conjunto c1 é dominada pela não estacionariedade no período da 6:00h às 24h.

Em termos da ordem de integração da série, era esperado que fosse necessário aplicar mais diferenciações nas séries para torná-las estacionárias, pois no período considerado no conjunto c2 a variabilidade dos parâmetros estatísticos é maior. Entretanto, das 16 séries não estacionárias novamente 100% tornaram-se estacionárias após a primeira diferenciação da série, o que facilita o processo de previsão via modelos para séries estacionárias.

A similaridade dos resultados obtidos com os conjuntos c1 e c2 mostra que o comportamento não estacionário do *rtt* é de fato determinado pela variabilidade diurna, o que é perceptível na análise visual das séries (figuras 4.11 e 4.12).

Resultados para séries de uma hora - conjunto c3

A análise das séries temporais que representam o *rtt* ao longo de um dia (conjuntos de séries c1 e c2) mostram que a amostragem diária do *rtt* apresenta um comportamento não estacionário. Nesta seção, as séries diárias foram divididas em séries por hora e a análise da estacionariedade mostra que a não estacionariedade do *rtt* ocorre ao longo das horas úteis de um dia.

Considerando 408 séries temporais com 3600 amostras cada (conjunto c3), correspondentes aos intervalos de amostragem entre horas cheias no período das 24h, verificou-se que 254 séries são estacionárias e 154 séries são não estacionárias, ou seja, 63,5% das séries são estacionárias e 36,5% são não estacionárias, o que indica um comportamento predominantemente estacionário para o *rtt*.

Entretanto, ao dividir o dia em dois períodos, das 8:00h às 20:00h (horas úteis) e das 20:00h às 8:00h (horas noturnas), verifica-se que das 204 séries temporais de cada período (17 dias e 12 séries por dia), 80 séries temporais são estacionárias (39,2%) e 124 são não estacionárias (60,8%), se consideradas as horas úteis, e 174 séries são estacionárias (85,3%) e 30 são não estacionárias (14,7%), se consideradas as horas noturnas. A figura 4.15 ilustra a distribuição percentual dos períodos de amostragens de acordo com o critério estacionário `x` não estacionário por hora do dia, e torna evidente que o *rtt* apresenta um comportamento predominantemente estacionário no período das 20:00h às 9:00h e predominantemente não estacionário no período das 9:00h às 20:00h.

De maneira similar aos conjuntos de dados c1 e c2, as séries não estacionárias do conjunto c3 também apresentaram ordem de integração 1, o que significa que podem ser convertidas para estacionárias se aplicada uma única diferenciação.

Resultados para séries variadas - conjunto c4

Nos conjuntos de dados c1, c2 e c3, a menor série temporal corresponde a uma hora de amostragem do *rtt* (3600 amostras). Nesta seção, analisa-se a estacionariedade de séries temporais com tamanhos variando de 300 a 3600 amostras, com passo de 300 amostras, o que corresponde a um histórico cujos intervalos de

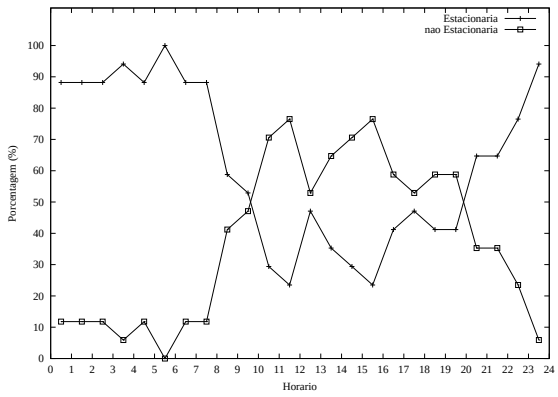

Figura 4.15: Distribuição percentual estacionário `x` não-estacionário por hora do dia considerando séries de uma hora.

amostragem vão desde 5 a 60 minutos (12 tamanhos). Para cada tamanho foram consideradas 680 séries retiradas aleatoriamente do conjunto c2, resultando 8160 séries.

O resultado da análise de estacionariedade das 8160 séries temporais está ilustrado no gráfico da figura 4.16, o qual apresenta a distribuição percentual entre séries estacionárias e não estacionárias para diferentes tamanhos de série temporal. O gráfico mostra que o tamanho da série temporal influencia na classificação comportamental do *rtt* (estacionário ou não estacionário). Séries pequenas tendem a ser estacionárias enquanto séries grandes tendem a ser não estacionárias (considera-se para séries grandes inclusive os resultados obtidos nas análises com os conjuntos c1 e c2).

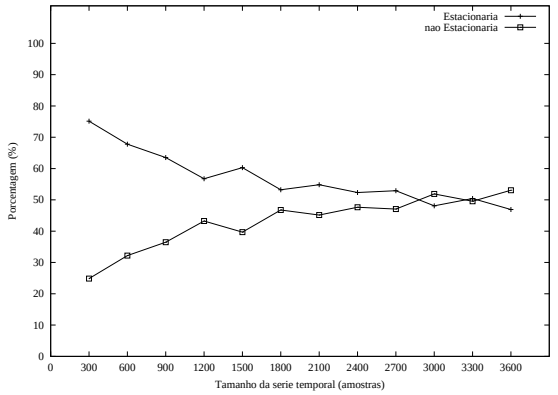

Figura 4.16: Relação estacionário `x` não-estacionário considerando para diferentes tamanhos de série temporal.

Por outro lado, é importante salientar que o resultado obtido para o conjunto de dados c4, no que diz respeito aos diferentes tamanhos de séries temporais, não indica o melhor tamanho para uma série temporal. Ele simplesmente serve para demonstrar que existe uma dependência entre o tamanho da série e a sua função de autocorrelação amostral, ou seja, indica que séries pequenas tendem a apresentar uma autocorrelação amostral menor. A seção 5.2.2 discute sobre o melhor tamanho de uma série temporal para *rtt*.

Uma análise por canal de comunicação mostra que o ponto de equilíbrio entre

a quantidade de séries estacionárias e não estacionárias pode variar de acordo com o canal de comunicação. A figura 4.17 ilustra as relações de estacionariedade por canal de comunicação. Deve ser observado que, sobre os canais UFRGS-POP/PA e UFRGS/UFSM, séries acima de 900 amostras tendem a ser não estacionárias, enquanto que na comunicação UFRGS-UFCG o mesmo não acontece.

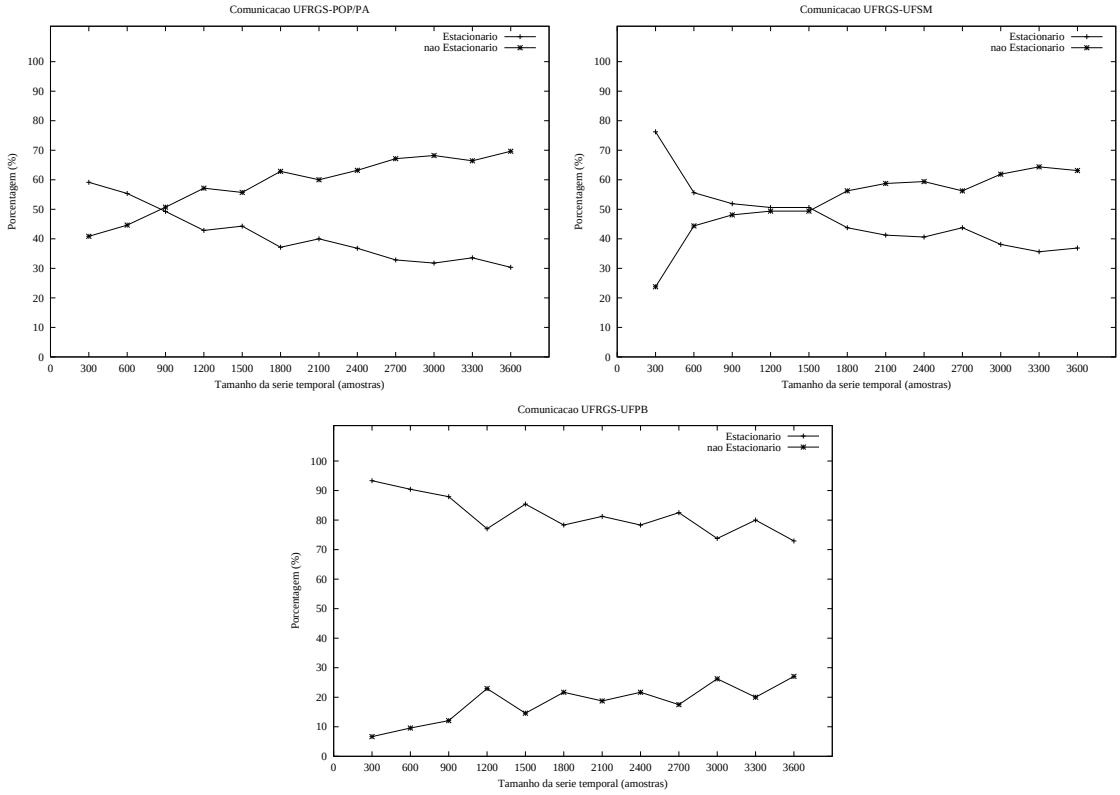

Figura 4.17: Relação estacionário `x` não-estacionário por canal de comunicação.

No que diz respeito à ordem de integração, ao ser realizada a diferenciação nas séries não estacionárias, observou-se que em 100% dos casos a série é transformada em estacionária, indicando que o *rtt* pode perfeitamente ser descrito por modelos ARIMA.

4.3 Conclusões Parciais

Este capítulo demonstrou que, embora o *rtt* não seja amostrado de maneira periódica, pois a característica do atraso de comunicação não é determinística, ele pode ser modelado como sendo uma série temporal, um modelo para amostragens periódicas, se aplicada uma interface que transforme a amostragem aperiódica em periódica.

Num detector de defeitos do estilo *pull*, onde a mensagem de interrogação de estado é enviada periodicamente, a interface de transformação de uma amostragem aperiódica em periódica é simples e pode ser facilmente integrada num detector de defeitos pelo algoritmo da figura 4.7.

A análise das séries temporais do *rtt* mostra que o comportamento do *rtt* sobre uma WAN é predominantemente não estacionário no período diurno (das 9:00h às 20:00h). Este resultado sugere que detectores de defeitos auto-ajustáveis

executando numa WAN, ao invés de utilizarem os tradicionais preditores baseados na média de amostras independentes na determinação dinâmica do *timeout*, utilizem modelos preditivos capazes de operar considerando a não estacionariedade do *rtt*.

5 MODELOS DE PREDIÇÃO DO *RTT*

Este capítulo especifica e avalia sete preditores, ou modelos preditivos, diferentes: LAST, MEAN, WINMEAN, DMA, BROWN, LPF e ARIMA. Como cada um deles apresenta características próprias, no que diz respeito às suas hipóteses sobre o comportamento da população amostral, eles são avaliados quanto a capacidade para prever a observação futura do *rtt* um passo a frente[1].

As especificações são realizadas em função do tempo de ida e volta de uma mensagem (*rtt*) e são apresentadas na seção 5.1. Como o uso do preditor ARIMA depende de parâmetros de configuração específicos, a definição destes parâmetros é apresentada na seção 5.2.

A avaliação da precisão foi realizada considerando dois conjuntos de dados: um conjunto de padrões de amostragem pré-especificados e um conjunto de amostragens reais. As seções 5.5 e 5.6 apresentam os resultados da avaliação com padrões e com amostragens reais, respectivamente. A metodologia adotada na análise da precisão é apresentada na seção 5.4. A seção 5.3 discute a escolha do *erro quadrático médio* como métrica para avaliar comparativamente a qualidade dos estimadores fornecidos pelos preditores.

5.1 Especificação dos Preditores

Sabendo que a variável aleatória de um detector de defeitos do estilo *pull* é o tempo de ida e volta de uma mensagem (*round-trip time* - *rtt*), esta seção especifica sete preditores diferentes para o *rtt*: LAST, que utiliza como parâmetro de inferência somente o último valor amostrado; MEAN, WINMEAN e LPF que, sob hipótese de estacionariedade, baseiam-se na média aritmética, ou ponderada, das amostras (ou de um subconjunto delas); DMA e BROWN, que supõem que o *rtt* comporta-se como uma série temporal não estacionária onde os erros não são correlacionados no tempo; e o ARIMA, que considera a correlação no erro e que permite o ajuste dinâmico do modelo dentre um conjunto de modelos autoregressivos e de médias móveis, estacionário ou não.

A margem de segurança, valor adicional (fixo ou variável) que compõe o próximo *timeout* a ser adotado pelo detector, não é incluída nem na especificação nem na análise dos preditores para não afetar a avaliação da precisão. Tal margem, normalmente definida em função da média e do desvio médio (seção 6.4), poderia minimizar as diferenças entre os valores preditos, por isto só é incorporada na avaliação das métricas de qualidade de serviço das diferentes implementações (uma para cada preditor) do detector de defeitos (capítulo 6).

[1]Embora o modelo de predição ARIMA tenha capacidade para prever vários passos a frente, para fins de comparação com os demais modelos de predição a avaliação é realizada para a previsão um passo a frente.

5.1.1 Preditor LAST

O preditor LAST implementa o método de previsão univariada mais simples. Utiliza o último valor amostrado do rtt como estimativa para o próximo valor. Sua equação de previsão é

$$\widehat{rtt}_{t+1} = rtt_t \tag{5.1}$$

onde rtt_t representa o tempo de ida e volta computado quando da chegada do último ack.

5.1.2 Preditor MEAN

Diferente do preditor LAST, o preditor MEAN trabalha computando o próximo valor baseado na média de toda a população de amostras do rtt, ou seja, adota por hipótese que o rtt é altamente aleatório e conseqüentemente segue um nível constante. Logo, a estimativa do rtt é computada por

$$\widehat{rtt}_{t+1} = \frac{\sum_{i=1}^{i=t} rtt_i}{t} \tag{5.2}$$

onde t representa o número de amostras do rtt e rtt_t representa o valor da amostra mais recente.

5.1.3 Preditor WINMEAN

O preditor WINMEAN (média de uma janela), assim como o preditor MEAN, computa sua estimativa baseado na média dos valores passados. Entretanto, supõe que o próximo valor segue somente os últimos $n = 30$ valores (uma janela de tamanho 30), ou seja, implementa uma média móvel simples com as últimas 30 observações. Formalmente,

$$\widehat{rtt}_{t+1} = \frac{\sum_{i=t-n}^{i=t} rtt_i}{n} \tag{5.3}$$

Por praticidade, este preditor considera que com $n \geq 30$ qualquer distribuição de probabilidade pode ser aproximada à uma distribuição Normal. Entretanto, salienta-se que, esta aproximação, embora amplamente utilizada (ALLEN, 1990), não vale para qualquer caso. Teoricamente, a aproximação só pode ser considerada válida de aplicado algum teste de Normalidade, tal como o de Shapiro Wilk (GONÇALVES, 2002).

5.1.4 Preditor LPF

O preditor LPF (*Low Pass Filter*) segue o método de previsão linear de Jacobson usado no protocolo de congestionamento do TCP (*Transport Control Protocol*) (JACOBSON, 1988). Assim, ele emprega a técnica de um filtro passa baixa para estimar valores futuros do rtt, ou seja, filtra o comportamento transiente dando um peso maior ao valor médio dos dados considerados não transientes ao computar a estimativa para o próximo valor. Sua equação de previsão tem a seguinte forma:

$$\widehat{rtt}_{t+1} = \alpha * rtt_t + (1 - \alpha) * \widehat{rtt}_t \tag{5.4}$$

onde α é a constante de alisamento que pode variar de 0 a 1 (valores baixos de *alpha* reduzem o impacto de valores transientes sobre o valor estimado), e $\widehat{rtt}_t$ é o último valor estimado por este preditor.

Conforme sugerido por Jacobson (1988), a implementação do preditor LPF ajusta o valor de α para $0,125$, o que resulta num filtro passa baixa. Na área de séries temporais, esta técnica é referida como alisamento exponencial simples (MORETTIN; CASTRO TOLOI, 1985).

5.1.5 Preditor DMA

Diferentemente dos preditores anteriores, o preditor DMA (*Double Moving Average*) estima o próximo valor do *rtt* considerando que os valores amostrados formam uma série temporal não estacionária $RTT = \{rtt_t\ , rtt_{t-1}\ , ..., rtt_{t-n-1}\}$, onde as observações que compõem a série são independentes entre si e do erro a_t.

O preditor DMA supõe que a não estacionariedade do *rtt* pode ser descrita por uma função linear dada por:

$$rtt_t = \phi + \psi.t + a_t \tag{5.5}$$

onde, a_t é o erro aleatório, ou ruído branco[2], e ϕ e ψ são os parâmetros desconhecidos da função que representam o intercepto e a declividade, respectivamente.

Assumindo que o *rtt* segue uma função linear (equação 5.5) e que o comportamento da série pode ser descrito pela média móvel das últimas n médias móveis simples, a equação de previsão do preditor DMA é (MONTGOMERY; JOHNSON, 1976):

$$\widehat{rtt}_{t+1} = 2.M_t - M_t^2 + \frac{2}{n-1}.(M_t - M_t^2) \tag{5.6}$$

onde M_t é o estimador da média móvel de primeira ordem de e M_t^2 é o estimador da média móvel de segunda ordem. A demonstração de obtenção da equação de previsão 5.6 é apresentada a seguir.

Demonstração:

Por definição, no instante t, a média móvel de primeira ordem com tamanho n corresponde à média aritmética das últimas n observações da variável aleatória *rtt*, ou seja:

$$M_t = \frac{rtt_{t-n+1} + ... + rtt_{t-1} + rtt_t}{n}$$

ou ainda,

$$M_t = M_{t-1} + \frac{rtt_t - rtt_{t-n}}{n}$$

onde M_{t-1} é a média móvel calculada no instante $t-1$.

Do mesmo modo, a média móvel de segunda ordem de tamanho n é, por definição, a média aritmética das últimas n médias móveis de primeira ordem, ou seja:

$$M_t^2 = \frac{M_{t-n+1} + ... + M_{t-1} + M_t}{n}$$

[2]Um ruído branco é um processo estocástico com média zero, variância constante e não correlacionado no tempo.

Considerando que o rtt segue um comportamento linear descrito pela equação 5.5, pode-se estimar os parâmetros ϕ e ψ a partir da esperança de M_t e M_t^2.

A esperança de M_t, $E(M_t)$, para um rtt_t linear (equação 5.5) é dada por

$$\begin{aligned} E(M_t) &= \frac{E(rtt_{t-n+1} + ... + rtt_{t-1} + rtt_t)}{n} \\ &= \frac{E(rtt_{t-n+1}) + ... + E(rtt_{t-1}) + E(rtt_t)}{n} \end{aligned} \tag{5.7}$$

como $E(rtt_t) = \phi + \psi.t$ tem-se que

$$\begin{aligned} E(M_t) &= \frac{\phi + \psi.(t-n+1) + ... + \phi + \psi.(t-1) + \phi + \psi.t}{n} \\ &= \frac{n.\phi + n.\psi.t - \frac{n.(n-1)}{2}.\psi}{n} \\ &= \phi + \psi.t - \frac{n-1}{2}.\psi \\ &= E(rtt_t) - \frac{n-1}{2}.\psi \end{aligned}$$

similarmente a esperança de M_t^2 é dada por

$$\begin{aligned} E(M_t^2) &= E(rtt_t) - (n-1).\psi \\ &= \phi + \psi.t - (n-1).\psi \end{aligned}$$

Das equações acima deriva-se o intercepto e a declividade:

$$\begin{aligned} \phi &= 2.E(M_t) - E(M_t^2) - \psi.t \\ \psi &= \tfrac{2}{n-1}[E(M_t) - E(M_t^2)] \end{aligned}$$

Substituindo $E(M_t)$ e $E(M_t^2)$ pelos seus respectivos estimadores M_t e M_t^2 tem-se que

$$\begin{aligned} \hat{\phi} &= 2.M_t - M_t^2 - \hat{\psi}.t \\ \hat{\psi} &= \tfrac{2}{n-1}.(M_t - M_t^2) \end{aligned}$$

Assim, considerando a existência de tendência linear (equação 5.5), as estimativas das observações no instante t são dadas por:

$$\widehat{rtt}_t = \hat{\phi} + \hat{\psi}.t = 2.M_t - M_t^2$$

Logo, a previsão j passos a frente pode ser obtida pela extrapolação da tendência j períodos a frente, ou seja, obtida fazendo

$$\widehat{rtt}_{t+j} = \widehat{rtt}_t + \hat{\psi}.j$$

Como conseqüência, a estimativa do rtt no período $t+1$ é dada por:

$$\widehat{rtt}_{t+1} = 2.M_t - M_t^2 + \frac{2}{n-1}.(M_t - M_t^2)$$

Fim da demonstração!

5.1.6 Preditor BROWN

Similarmente ao preditor DMA, o preditor BROWN considera que os valores amostrados do rtt formam uma série temporal $RTT = \{rtt_t\ , rtt_{t-1}\ , ..., rtt_{t-n-1}\}$ com amostras não correlacionadas no tempo. Entretanto, o preditor BROWN implementa o método de alisamento exponencial duplo, usualmente chamado Método de Brown (MORETTIN; CASTRO TOLOI, 1985), para realizar previsões.

O método de Brown é similar ao método de alisamento exponencial simples (AES) utilizado no preditor LPF, no que diz respeito à sua maneira de funcionamento (filtro de alisamento), mas difere por considerar que os dados podem apresentar não estacionariedade na forma de uma tendência linear (positiva ou negativa), tal como expresso pela equação 5.5. Logo, no instante t, o método de Brown calcula um segundo valor exponencialmente alisado como sendo

$$\widehat{rtt}_t^2 = \alpha.\widehat{rtt}_t + (1-\alpha).\widehat{rtt}_{t-1}^2 \tag{5.8}$$

onde, α é a constante de alisamento, $\widehat{rtt}_t$ é o último valor computado de acordo com o método AES (equação 5.4) e $\widehat{rtt}_{t-1}^2$ é o último valor exponencialmente alisado através do uso de 5.8.

A versão implementada do preditor BROWN, assim como no preditor LPF, minimiza o impacto das flutuações recentes no valor estimado, fazendo $\alpha = 0,125$ tanto no alisamento de primeira ordem como no de segunda ordem.

De maneira similar ao apresentado para a equação de previsão do preditor DMA, considerando o modelo de tendência linear ($rtt_t = \phi + \psi.t + a_t$), a previsão j períodos à frente é obtida pela extrapolação da tendência. Deste modo, estima-se os parâmetros ϕ e ψ do modelo linear a partir da esperança de rtt_t e rtt_t^2 como sendo:

$$\begin{aligned} \hat{\phi} &= 2.\overline{rtt}_t - rtt_t^2 \\ \hat{\psi} &= \tfrac{\alpha}{1-\alpha}.(\overline{rtt}_t - rtt_t^2) \end{aligned}$$

Sabendo que a estimativa para o valor do rtt no instante t é $\widehat{rtt}_t = E(rtt_t) = \hat{\phi}$, tem-se que a estimativa do rtt no instante t é dada por

$$\widehat{rtt}_t = 2.\overline{rtt}_t - rtt_t^2$$

Logo, a previsão para o período $t+j$ é obtida por

$$\widehat{rtt}_{t+1} = \widehat{rtt}_t + \hat{\psi}.j$$

onde,

$$\hat{\psi} = \frac{\alpha}{1-\alpha}.(\overline{rtt}_t - rtt_t^2)$$

Portanto, a equação de previsão do preditor BROWN um passo a frente é dada por:

$$\widehat{rtt}_{t+1} = 2.rtt_t - rtt_t^2 + \frac{\alpha}{1-\alpha}.(rtt_t - rtt_t^2) \tag{5.9}$$

5.1.7 Preditor ARIMA

Diferentemente dos preditores anteriores, os quais são projetados para um único tipo de processo estocástico (estacionário ou não estacionário, com nível constante ou com tendência linear), o preditor ARIMA pode ser dinamicamente ajustado para realizar previsões de acordo com vários modelos de séries temporais.

Assumindo que o processo estocástico representado pode ser uma das variantes de um processo autoregressivo integrado de médias móveis (ARIMA - *Autoregressive Integrated Moving Average process*), o preditor ARIMA incorpora a possibilidade de ser configurado para processos autoregressivos de ordem p (AR(p)), processos de médias móveis de ordem q (MA(q)), processos autoregressivos e de médias móveis de ordem p e q (ARMA(p,q)), todos nas versões estacionárias ou não (integrados de ordem d). Sinteticamente, o preditor ARIMA serve a qualquer subgrupo de processos autoregressivos integrados e de médias móveis de ordem p, d e q (ARIMA(p,d,q)), onde a ordem de integração d indica o número de diferenças necessárias para transformar a série temporal não estacionária em estacionária, conforme discutido na seção 4.2.

Além da possibilidade de servir a diferentes modelos preditivos, o preditor ARIMA considera que o erro de previsão pode estar correlacionado, distingüindo-se dos demais preditores, os quais não consideram a possibilidade de correlação entre observações adjacentes do rtt. Segundo Box, Jenkins e Reinsel (1994), não considerar a correlação pode introduzir sérias limitações na validade dos modelos.

A equação de previsão deste preditor é baseada na noção de um filtro linear, onde o filtro é alimentado com uma seqüência não previsível do rtt, chamada ruído a, e deve resultar numa seqüência parcialmente previsível do rtt. Em outras palavras, a equação de previsão corresponde a função de transferência do filtro, determinada com base nos valores passados do rtt. O filtro será de boa qualidade quando a componente aleatória da saída for mínima, ou seja, quando o modelo ARIMA adotado conseguir representar a parte previsível dos dados. A equação geral de previsão do preditor ARIMA é

$$\widehat{rtt}_t = \bar{rtt} + \frac{\theta(B)}{\phi(B).\nabla^d}.a_t \tag{5.10}$$

onde:

> B é o operador de defasagem (*backward shift operator*), definido por $B.rtt_t = rtt_{t-1}$;
>
> ∇ é o operador de diferença (*backward difference operator*), definido por $\nabla rtt_t = rtt_t - rtt_{t-1} = (1 - B).rtt_t$;
>
> a_t é o ruído no instante t;
>
> d é a ordem de não estacionariedade, ou ordem de integração; e
>
> $\phi(B)$ e $\theta(B)$ são polinomiais de B com coeficientes p e q respectivamente, onde p é o número de termos autoregressivos e q é o número de termos de médias móveis.

Pelo exposto, percebe-se que, antes de realizar previsões com o preditor ARIMA, é necessário determinar qual dos modelos ARIMA é o mais adequado para ser aplicado no filtro (estrutura do modelo) e quais os parâmetros que devem ser utilizados no modelo. A estrutura de um modelo ARIMA é definida pelos coeficientes p, q e d do modelo, ou seja, pelos termos autoregressivos (p) e de médias móveis (q) que são necessários, e pelo número de diferenças (d) necessárias para transformar a série não estacionária em estacionária. Neste trabalho, o procedimento de determinação da estrutura do modelo com base nos dados previamente conhecidos

```
inicialização:
    numNovasAmostras ← 0
while numNovasAmostras < 360 do
      espera por uma nova amostra do rtt
      numNovasAmostras ← numNovasAmostras + 1
      atualiza a série temporal com a nova amostra
      realiza previsão segundo o preditor LFP
end while

loop
  espera por uma nova amostra do rtt
  numNovasAmostras ← numNovasAmostras + 1
  atualiza a série temporal com a nova amostra
  if erro de previsão for maior do que 5% AND numNovasAmostras ≥ 120
  then
    identifica o modelo ARIMA;
    ajusta os parâmetros do modelo;
    numNovasAmostras ← 0
  end if
  realiza previsão segundo o preditor ARIMA;
end loop
```

Figura 5.1: Algoritmo de previsão embasado no preditor ARIMA.

(série temporal do rtt) é referido como **identificação** do modelo, enquanto que o procedimento para ajuste (fit), ou reajuste ($refit$), dos parâmetros do modelo é referido simplesmente como **ajuste** do modelo.

Segundo a metodologia de Box, Jenkins e Reinsel, os coeficientes p, q e d do modelo podem ser determinados através da avaliação das funções de autocorrelação e autocorrelação parcial da série e, segundo Bowerman e O'Connell (1993), a determinação pode ser realizada automaticamente usando regras experimentais consolidadas na literatura. Assim, neste trabalho, a identificação do modelo ARIMA segue os procedimentos teóricos da metodologia de Box, Jenkins e Reinsel, mas é resolvida automaticamente fazendo uso de regras experimentais. A seção 5.2.3 detalha o procedimento de identificação automática adotado.

Com o processo de identificação automatizado, adota-se um algoritmo de previsão similar ao utilizado no sistema de previsão de carga de Dinda e O'Hallaron (1999a), no qual o modelo ARIMA empregado nas previsões é periodicamente ajustado. Naquele caso, é realizado somente o ajuste periódico dos parâmetros do modelo, mas não a identificação periódica de sua estrutura. O algoritmo de previsão projetado é apresentado na figura 5.1.

O algoritmo 5.1 está configurado de tal maneira que os procedimentos de identificação e ajuste do modelo ARIMA usam como histórico as últimas 360 amostras do rtt e são disparados numa periodicidade não inferior à amostragem de 120 valores do rtt, sempre que o erro quadrático médio de previsão para as últimas 10 amostras for superior a 5% do erro quadrático médio de previsão para os valores de toda a série considerada. A próxima seção trata em detalhes o porquê desta configuração. Identificado o modelo, tanto o ajuste dos parâmetros como a previsão de

valores são tarefas realizadas pelo pacote *TimeSeries* do *Resource Prediction System - RPS* (DINDA; O'HALLARON, 1999b; DINDA, 2000).

5.2 Configuração do preditor ARIMA

Considerando um sistema de previsão dinâmico, como o especificado pelo algoritmo 5.1, a precisão do preditor ARIMA depende de fatores tais como:

- a periodicidade em que os termos polinomiais necessários para uma boa previsão são identificados (estrutura do modelo ARIMA);
- a periodicidade em que os parâmetros do modelo identificado são ajustados; e
- o tamanho da série temporal utilizada nos ítens acima.

A abordagem adotada para configurar cada um destes fatores é detalhada nas próximas duas seções.

Adicionalmente, esta seção também apresenta o método adotado para realizar a identificação automática da estrutura de um modelo ARIMA. Identificar automaticamente um modelo ARIMA não é uma tarefa simples, pois mesmo análises interativas realizadas por especialistas independentes podem resultar em modelos diferentes (BOX; JENKINS; REINSEL, 1994). A abordagem adotada segue regras experimentais coletadas na literatura especializada.

Salienta-se que qualquer alteração na configuração estabelecida nesta seção pode influenciar diretamente nos resultados obtidos neste trabalho, pois a precisão do preditor ARIMA pode ser afetada.

5.2.1 Período de identificação/ajuste do modelo ARIMA

Para poder oferecer uma boa qualidade na previsão quando as características estatísticas da variável de interesse se alteram ao longo do tempo, um sistema de previsão de valores baseado em modelos ARIMA necessita, de tempos em tempos, reajustar o modelo (parâmetros do polinômio) ou mesmo re-identificar a própria estrutura do modelo. A re-identificação trata da re-definição do número de termos autoregressivos e de médias móveis do modelo, bem como da sua ordem de integração, de acordo com os últimos valores amostrados.

Para estabelecer os instantes de identificação/ajuste do modelo, há pelo menos duas abordagens que podem ser utilizadas:

1. baseado nas características do *rtt*, estabelecer uma periodicidade pré-determinada; ou
2. baseado na qualidade da previsão, disparar os processos de identificação e ajuste sempre que o erro de previsão extrapolar um determinado limite.

A primeira abordagem tem a vantagem de ser simples de implementar e de impedir que a freqüência de atualização (identificação/ajuste) do modelo torne o preditor inviável, no que diz respeito ao seu tempo de resposta. O tempo gasto para ajustar um modelo ARIMA de baixa ordem pode chegar a 100ms no RPS (DINDA, 2000, p. 31), o que é significativo em relação ao *rtt* médio observado (seção 5.6). Por

outro lado, esta abordagem tem a desvantagem de que a periodicidade a ser adotada é de difícil determinação, pois exige uma análise prévia rigorosa das características comportamentais do *rtt*. Além disto, em sistemas dinâmicos, esta abordagem não permite que se dê garantias de precisão, pois o comportamento do tráfego pode alterar a qualquer momento, influenciando na qualidade da previsão.

Já a segunda abordagem tem a vantagem de permitir que se garanta uma dada precisão, pois pode assegurar que os erros fiquem limitados aos níveis pré-estabelecidos. Sua desvantagem é que tal garantia pode gerar uma freqüência tão grande de ajustes que pode inviabilizar o uso do preditor. Um número elevado de ajustes no modelo pode aumentar excessivamente o tempo de resposta do preditor.

Pelo exposto, neste trabalho adota-se uma terceira abordagem, a híbrida:

3. disparar os processos de identificação e ajuste com base na qualidade da previsão, mas estabelecer um período mínimo de operação para repetir a ação.

A abordagem híbrida agrega as vantagens das duas primeiras abordagens: permite fixar uma periodicidade conveniente; e permite disparar o procedimento de atualização do modelo, caso o erro de previsão extrapole um dado limite. Entretanto, necessita que seja definido o período mínimo para disparar os processos de identificação e ajuste, o qual pode ser estipulado em função do número de observações.

De acordo com Montgomery e Johnson (1976, p. 236), são necessárias pelo menos 100 observações para poder identificar um modelo ARIMA. Neste trabalho é fixado que o preditor ARIMA somente fica apto a realizar a atualização do respectivo modelo após computar pelo menos 120 previsões com um dado modelo.

Após a atualização ser habilitada, o preditor monitora o erro quadrático médio de previsão. Se o erro quadrático médio dos últimos 10 valores amostrados for maior do que 5% do erro quadrático médio da série temporal considerada, então o modelo será automaticamente atualizado, pois o preditor aumentou significativamente (mais de 5%) o seu erro de previsão.

Embora o ajuste possa ser feito de maneira independente da identificação, neste trabalho, antes de realizar um ajuste, sempre é realizada a identificação do modelo, para garantir que a atualização do modelo corresponda mais efetivamente aos últimos valores amostrados. Esta decisão leva em conta o tempo gasto por cada uma destas tarefas.

O custo do processo de identificação automática proposto, que é diretamente decorrente do consumo de tempo, é muito inferior ao custo do processo de ajuste do modelo (resolvido pelo sistema de predição de recursos RPS (DINDA; O'HALLARON, 1999b)). Considerando um conjunto de 17 execuções coletadas no período das 6h às 24h, cada uma com 64800 amostras do *rtt*, computou-se, em dois computadores com características distintas, o custo para realizar 5000 atualizações do modelo ARIMA. Como resultado, num computador com processador K6II/450MHz, o tempo para identificação de um modelo ARIMA(p,d,q) foi de 415ms (com um desvio padrão igual a 0,35ms), enquanto o tempo para ajustar os parâmetros do modelo foi de 76,94ms (com um desvio padrão igual a 28,4ms). Num computador com processador PentiumIII/800MHz, o tempo de identificação foi de 1.38ms, com um desvio padrão igual a 0,37ms, enquanto que o tempo para ajustar os parâmetros foi de 38,06ms, com um desvio padrão igual a 13,62ms. Em síntese,

tem-se que o processo de identificação é cerca de 20 vezes mais barato do que o processo de ajuste. Dinda e O'Hallaron (1999b, p. 31) realizam um estudo mais detalhado do desempenho do processo de ajuste de um modelo ARIMA.

5.2.2 Tamanho da série temporal

Como o custo computacional da etapa de ajuste do modelo ARIMA depende do tamanho da série temporal (quanto maior a série maior o custo (DINDA; O'HALLARON, 1999b)), e a representatividade da série depende da quantidade de amostras nela contida, é importante definir o melhor compromisso entre o tamanho da série e sua representatividade (aqui mensurada pelo *eqm* produzido pelo preditor ARIMA que a utiliza). Neste sentido, a metodologia aqui definida para determinar o tamanho da série temporal teve como objetivo identificar que tamanho de série fornece o menor erro quadrático médio, se considerada a abordagem híbrida para identificação e ajuste adotada (seção 5.2.1).

Para a determinação do tamanho adequado para a série temporal, das 552 horas de amostragens do *rtt* coletadas (na seção 4.2.1 foram apresentados detalhes sobre a coleta de dados), foram excluídos os finais de semana e os períodos da madrugada (das 0h às 6h), o que resultou em 306 horas de amostragens. A decisão de excluir os períodos da madrugada vem da constatação de que durante estes períodos o comportamento do *rtt* tende a ser estacionário (NUNES; JANSCH-PÔRTO, 2002b) e da hipótese de que, neste caso, preditores mais simples devem ser preferidos. A decisão para exclusão dos finais de semana seguiu os mesmos princípios.

Do conjunto de 306 horas de amostragens, foram pinçados aleatoriamente 680 casos de teste de tamanho $n + 1200$ para cada valor de n, onde n representa o tamanho da série temporal e varia de 120 a 2400 valores com passo de 120 valores. Para cada caso de teste, avaliou-se a qualidade do preditor ARIMA (em termos do *eqm*), considerando um conjunto de 1200 previsões em cada caso de teste.

Usando a metodologia descrita acima, computou-se o *mse* do preditor ARIMA e seu respectivo desvio médio para cada tamanho de série temporal considerado. A figura 5.2 sintetiza os resultados obtidos. O ponto médio de cada linha vertical representa a média dos *eqm* obtidos nos 680 casos de teste quando adotado o respectivo tamanho de série temporal. Os pontos extremos da linha indicam o desvio médio do *eqm*.

Da figura 5.2 percebe-se claramente que séries com 120 e 240 amostras são pequenas demais para servir como amostras para o procedimento de identificação do modelo ARIMA, indicando que estes tamanhos devem ser descartados. Por outro lado, os outros 18 tamanhos de série temporal avaliados apresentaram resultados muito semelhantes, tanto no que diz respeito ao *eqm* como ao seu desvio médio, indicando que qualquer um destes tamanhos de série pode servir para o processo de identificação. Logo, como o custo computacional da fase de identificação/ajuste do modelo ARIMA depende do tamanho da série temporal, o preditor ARIMA utilizado neste trabalho adota o menor dentre aqueles 18. Como resultado, a série temporal do preditor ARIMA contém 360 amostras do *rtt*.

5.2.3 Identificação automática do modelo ARIMA

Esta seção explica qual o procedimento adotado para identificar automaticamente a estrutura de um modelo ARIMA, ou seja, identificar quais os parâmetros p, d e q do modelo. Tais parâmetros indicam quantos termos autoregressivos e de

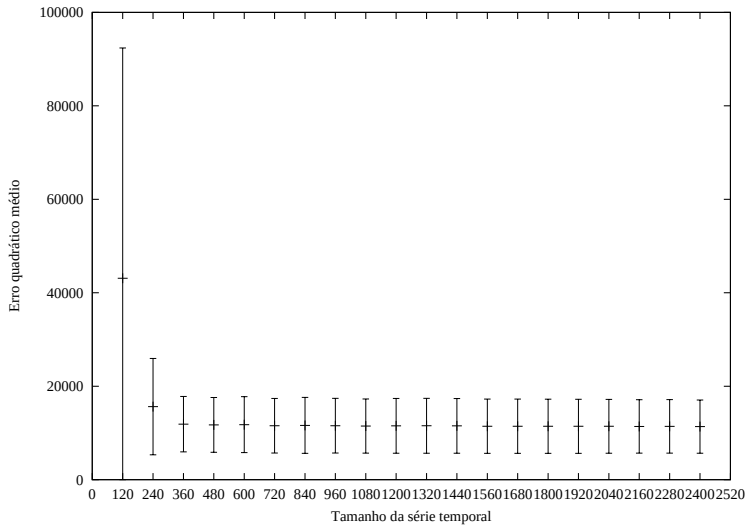

Figura 5.2: Erro quadrático médio de previsão para diferentes tamanhos de série temporal.

médias móveis devem compor o modelo, bem como qual o grau de diferenciação que deve ser aplicado na série temporal original para que ela possa ser tratada como estacionária.

De acordo com a metologia de Box, Jenkins e Reinsel (1994), a identificação automática do modelo ARIMA pode ser realizada em dois estágios:

> **primeiro -** reduzir um processo ARIMA(p,d,q) a um processo ARMA(p,q) identificando a ordem de integração d do processo, onde d representa o número de diferenciações necessárias para transformar a série não estacionária em estacionária;
>
> **segundo -** identificar os parâmetros p e q da série estacionária.

No primeiro estágio, a identificação da ordem de integração é realizada por um procedimento iterativo que analisa o comportamento da acf e diferencia a série temporal, caso necessário. Tal procedimento foi utilizado na análise da estacionariedade do rtt e está descrito na seção 4.2.2. Na prática, após aplicar o primeiro estágio da identificação, tem-se a ordem de integração d e a série temporal devidamente diferenciada (série estacionária).

No segundo estágio, a identificação automática é feita via análise do comportamento das funções de autocorrelação amostral (acf) e autocorrelação parcial amostral ($pacf$), que segundo Box, Jenkins e Reinsel apontam as seguintes regras teóricas:

- se o nível de significância estatística das autocorrelações que compõem a acf de uma série temporal *declina regularmente* com o passar do tempo, pode-se concluir que a série apresenta componentes *autoregressivos*;
- se o nível de significância estatística das autocorrelações parciais que compõem a $pacf$ de uma série temporal *declina regularmente* com o passar do tempo, pode-se concluir que a série apresenta componentes de *médias móveis*;

- se a *pacf* apresentar um *corte* no passo p, ou seja, se houver uma descontinuidade em seu declínio no passo p, e a *acf declinar regularmente*, então pode-se concluir que a série temporal é um processo puramente autoregressivo de ordem p, um AR(p);

- se a *acf* apresentar um *corte* no passo q e a *pacf declinar regularmente*, então pode-se concluir que a série temporal é um processo puramente de médias móveis de ordem q, um MA(q);

- se tanto a *acf* quanto a *pacf* declinarem regularmente, então pode-se concluir que a série temporal é um processo autoregressivo e de médias móveis, um ARMA(p,q).

Na prática, no segundo estágio, dois comportamentos são de interesse: o corte e o declínio lento, onde o corte indica a descontinuidade da *acf*, ou da *pacf*. Para identificar tais comportamentos, usando as autocorrelações amostrais, utiliza-se a regra experimental apresentada por Bowerman e O'Connell (1993), a qual diz que:

- o corte pode ser identificado testando se a autocorrelação no passo k, ou a autocorrelação parcial no passo k, abruptamente deixa de ser estatisticamente significante, ou seja, cai abruptamente para valores próximos a zero, num passo $k \leq 3$.

Se não for identificado um corte, então supõe-se que o comportamento é um declínio regular. Como resultado, o segundo estágio identifica facilmente modelos AR(p) e MA(q), baseando-se nas regras teóricas apresentadas por Box, Jenkins e Reinsel (1994) e na regra experimental apresentada por Bowerman e O'Connel (1993).

Embora a *acf* e a *pacf* sejam extremamente usuais na identificação dos modelos AR(p) e MA(q), a identificação dos modelos ARMA(p,q) nem sempre pode ser realizada com base na *acf* e *pacf*, pois estas funções não permitem uma correta identificação ARMA em todos os casos (BOX; JENKINS; REINSEL, 1994). Tanto a *acf* como a *pacf* declinam regularmente, dificultando a identificação. Por outro lado, o interesse no desenvolvimento de ferramentas que identifiquem o melhor modelo ARMA culminou em alguns métodos teóricos diferenciados (BOX; JENKINS; REINSEL, 1994, p. 197). O problema é que, em geral, a solução computacional destes métodos teóricos exige muitos recursos computacionais. Até o presente, só se tem conhecimento de dois sistemas comerciais que fazem a identificação automática de modelos ARIMA, o SCA (SCIENTIFIC COMPUTING ASSOCIATES, 2001), o qual faz uso não só da *acf* e *pacf* como também da correlação canônica, e o Auto-Box (2002), o qual não utiliza a *acf* e a *pacf* mas sim os coeficientes de regressão condicional e não condicional. De qualquer modo, o processo de identificação não é determinístico em nenhum dos casos, exigindo o uso de técnicas heurísticas de identificação.

Como a pesquisa na área de identificação automática de modelos ARMA está fora do escopo deste trabalho, sempre que se detecta a existência deste, utiliza-se o modelo ARMA(1,1), ou seja, supõe-se que os dados contenham componentes autoregressivos e de médias móveis de primeira ordem. Tal escolha não compromete a qualidade da previsão em relação aos outros preditores, pois embora não se conheça o modelo exato, exclui-se as hipóteses de um modelo puramente autoregressivo ou puramente de médias móveis.

5.3 Métrica de comparação

Dada uma variável aleatória, o rtt, e sete preditores distintos que tentam descrever o comportamento da variável e prever seus valores futuros, precisa-se de uma métrica que indique a precisão do valor previsto por cada um deles e que possa ser utilizada numa análise comparativa de precisão.

Segundo Vasconcellos e Alves (2000), estatisticamente um estimador deve ser avaliado de acordo com suas propriedades, a saber:

- **não tendenciosidade** - segundo esta propriedade, o estimador do rtt, $\widehat{rtt}$, é um estimador não tendencioso se a esperança do estimador for igual à esperança do rtt, ou seja, $E(\widehat{rtt}) = E(rtt)$;

- **eficiência** - segundo esta propriedade, o estimador mais eficiente é o que apresenta a menor variância, ou seja, se as variâncias de dois estimadores do rtt apresentarem uma relação do tipo $var[\widehat{rtt}_{e1}] < var[\widehat{rtt}_{e2}]$, então $\widehat{rtt}_{e1}$ é o estimador mais eficiente. Entretanto, esta propriedade só se aplica se os estimadores forem não tendenciosos;

- **mínimo erro quadrático médio** - de acordo com esta propriedade, o melhor estimador é o que apresenta o menor erro quadrático médio (eqm), onde

  $$eqm[\widehat{rtt}] = E[(\widehat{rtt} - rtt)^2] = var[\widehat{rtt}] + (viés[\widehat{rtt}])^2$$

  Essa propriedade pode ser utilizada em comparações entre estimadores quando não se adota, como pré-requisito, a propriedade de não-tendenciosidade. Por isso, além da variância, esta propriedade também considera a relevância do viés, ou seja, a diferença entre $E(\widehat{rtt})$ e o rtt amostrado. Deve ser observado que, segundo esta propriedade, um estimador tendencioso pode ser preferível a um não tendencioso, desde que a menor variância compense o quadrado do viés.

Quando se considera que a variável aleatória tem uma distribuição de probabilidades Normal, costuma-se usar como métricas a média e a variância amostral, pois são estimadores não tendenciosos dos parâmetros populacionais de uma curva Normal. Pelo mesmo motivo, se o rtt apresentasse um comportamento composto por um nível constante mais um ruído branco, o melhor estimador seria a esperança do rtt. Entretanto, conforme o capítulo anterior, o rtt amostrado num detector de defeitos do estilo *pull* freqüentemente apresenta comportamento não-estacionário numa WAN (seção 4.2), ou seja, sua média não é constante. Logo, neste trabalho, os estimadores fornecidos pelos preditores especificados na seção 5.1 são avaliados pelo erro quadrático médio mensurado.

5.4 Metodologia de análise

Definida a métrica para comparação entre os diferentes preditores, a análise comparativa (seção 5.6) foi realizada considerando três fatores:

1. as características teóricas dos preditores, conforme abordado na seção 5.1;

2. a análise do erro quadrático médio (análise estatística); e

3. a interpretação das curvas de previsões (análise gráfica).

As análises estatística e gráfica foram realizadas em conjunto, mas considerando dois conjuntos de dados: padrões de amostragem e amostragens reais.

A análise com padrões teve como objetivo refinar o conhecimento sobre o comportamento de cada preditor. Por isto, esta análise considera 8 padrões de amostragem distintos, cada um com 10360 amostras do *rtt*, e consiste em realizar uma análise simples da magnitude do erro quadrático médio produzido por cada preditor e uma análise gráfica das previsões realizadas. A definição dos padrões de amostragem utilizados, bem como a análise dos resultados produzidos por cada preditor são apresentados na seção 5.5.

Para amostragens reais, a análise comparativa que será apresentada na seção 5.6, foi realizada de acordo com a metodologia descrita a seguir.

- **Análise Estatística**

  1. Tendo por base 23 amostragens com aproximadamente 86400 amostras cada, relativas a 24 horas de coleta de dados, computar o erro quadrático médio dos preditores para cada amostragem e examinar os resultados através da análise da variância (ANOVA)[3]. Havendo diferenças estatisticamente significativas entre os modelos, verificar as diferenças aplicando o teste de Duncan[4] a 5% de probabilidade.
  2. Repetir o item anterior para 17 amostragens com aproximadamente 64800 amostras, relativas aos dias da semana, de segunda a sexta-feira, das 6 às 24h.
  3. Considerando 8160 amostragens, com tamanho variando de 300 a 3600 amostras, com incrementos de 300 em 300 (680 amostragens para cada tamanho), pinçados randomicamente das 17 amostragens do item anterior, computar o erro quadrático médio dos preditores para cada caso e analisar os resultados por tamanho, através da análise da variância e do teste de Duncan.

- **Análise Gráfica**

  1. Empregando as 23 amostragens, com aproximadamente 86400 amostras cada, as quais incluem comportamentos diversos e não determinístico, plotar suas curvas de previsões para cada preditor (juntamente com a curva do *rtt*) e procurar identificar situações que esclareçam o seu comportamento geral.
  2. Usando amostragens de tamanhos variados, com comportamentos específicos (alta e baixa variância, com tendência crescente e decrescente), plotar suas curvas de previsões para cada preditor e procurar identificar situações que esclareçam o seu comportamento geral.

[3] A análise da variância, também conhecida como ANOVA, é um método baseado na análise de variações amostrais que serve para testar a igualdade de três ou mais médias populacionais (WINTER, 2002).

[4] O teste de Duncan é um método para comparar a igualdade entre múltiplas médias. Como resultado o teste classifica as médias em grupos. Se duas médias forem diferentes pertencerão a grupos diferentes.

A abordagem metodológica descrita acima avalia, de maneira prática, a qualidade da previsão fornecida por cada um dos preditores em termos do erro quadrático médio (magnitude do erro) e das curvas de previsões (capacidade de se adaptar à mudanças nos dados).

No caso específico do preditor ARIMA, outras abordagens estatísticas poderiam ter sido adotadas como, por exemplo, avaliar como o modelo de previsão representa os dados já amostrados (se o modelo escolhido é de fato o melhor) ou avaliar como o modelo gera novas amostragens com as mesmas propriedades estatísticas do *rtt* (usual em geração de padrões para simulação). Entretanto, estas abordagens não poderiam ser aplicadas na avaliação dos outros preditores, pois naqueles não existe um processo de identificação/ajuste do modelo. Assim, como o uso destas outras abordagens não contribui com os objetivos deste trabalho, elas não foram realizadas.

Além da análise através de duas abordagens (estatística e gráfica), para evitar uma avaliação viciada, exploram-se diferentes parâmetros de configuração tais como:

- o comportamento da amostragem (conhecido e desconhecido);
- o tipo de preditor (LAST, MEAN, WINMEAN, DMA, BROWN, LPF e ARIMA);
- o período de teste (avaliado no intervalo [10..60] minutos);
- o número de parâmetros adotado pelo preditor ARIMA (avaliado p=[0..2], d=[0..2], q=[0..2])[5].

Na análise estatística, as estimativas dos erros quadráticos médios dos preditores foram realizadas com base no ajuste dos modelos preditivos para os primeiros 360 valores da amostragem utilizada. Ajustados os modelos preditivos, o *eqm* computado representa o erro quadrático médio obtido nas previsões até o final da amostragem em avaliação. Por exemplo, se considerada uma amostragem de 600 valores, o *eqm* corresponderá aos últimos 240 valores da amostragem.

A utilização dos primeiros 360 valores para ajuste permite que todos os preditores sejam avaliados em uma situação normal de funcionamento, ou seja, após terem passado pela fase de inicialização. Foi tomado a fase de inicialização mais longa, a do preditor ARIMA, o qual precisa de no mínimo 360 amostras para poder identificar/ajustar o modelo ARIMA e iniciar as previsões com base no modelo identificado (na seção 5.2.2, foram apresentados mais detalhes). Se a quantidade de amostras for inferior a 360, o preditor ARIMA opera de maneira idêntica ao preditor LPF.

Finalmente, embora toda mensagem gerada pelo programa de monitoramento seja marcada com um *timestamp*, a análise dos preditores foi realizada ignorando os *gaps* existentes nas amostragens do *rtt*, causados por mensagens perdidas. Esta decisão é resultado da interface utilizada para representar como invariante no tempo, um sinal que é variante no tempo, conforme foi descrito na seção 4.1.

[5] Se o identificador de modelo ARIMA identificar $p = d = q$=0, ou seja, identificar que o *rtt* é um sinal não tendencioso e não correlacionado ao longo do tempo, o preditor adotado no período será o MEAN, teoricamente o melhor preditor neste caso (vide seção 5.3).

5.5 Análise dos preditores com padrões de amostragens

Antes de avaliar a precisão e o comportamento de cada preditor sobre dados reais amostrados, nesta seção avalia-se a precisão e o comportamento dos preditores quando estimulados por padrões fictícios de amostragem do *rtt*. O uso inicial de padrões fictícios para análise auxilia na compreensão do comportamento dos preditores.

5.5.1 Definição dos padrões

Considerando que 360 amostras devem ser utilizadas para ajustar os modelos preditivos, foram gerados sete padrões distintos com 10360 amostras cada. Assim, 10000 amostras, de cada padrão, podem ser utilizadas na avaliação comparativa entre os modelos. A figura 5.3 ilustra o comportamento de cada padrão para um conjunto das primeiras 120 amostras e amplitude de 600 milisegundos (de 200ms a 800ms). Os detalhes comportamentais de cada padrão são descritos a seguir:

- **padrão 1** - representa um *rtt* aleatório com nível constante e baixa variância. Com um nível fixado em 725ms, o *rtt* varia de maneira aleatória (não correlacionado no tempo) entre 700ms e 750ms (variação máxima de 50ms);
- **padrão 2** - representa um *rtt* que se alterna a cada amostra (tipo sanfona) com nível constante e baixa variância. Com um nível fixado em 725ms, o *rtt* se alterna entre valores de 700ms e 750ms (variação fixa de 50ms a cada amostra);
- **padrão 3** - representa um *rtt* aleatório com nível constante e alta variância. Fixada uma variação máxima de 500ms, este padrão corresponde a um *rtt* aleatório entre 200ms e 700ms. A faixa de variação e os valores de 200ms e 700ms foram escolhidos de modo que estivessem aproximadamente de acordo com os níveis médios observados na coleta de dados realizada. A faixa de variação de 500ms é um pouco maior do que o desvio médio dos dados coletados, mas facilita a avaliação do comportamento dos preditores sob períodos de forte variância;
- **padrão 4** - representa um *rtt* que se alterna a cada amostra com nível constante e alta variância. Mantendo uma faixa de variação de 500ms, este padrão alterna o valor do *rtt* a cada amostra entre 200ms e 700ms;
- **padrão 5** - representa um *rtt* que alterna sua tendência linear a cada 15 amostras. Partindo de um valor mínimo de 200ms, faz o *rtt* crescer numa taxa de aproximadamente 35ms a cada amostra, durante um período de 15 observações, quando então inverte a tendência do *rtt*. A tendência decrescente, com a mesma taxa da crescente, também tem duração de 15 observações, quando então é retomada uma tendência crescente e o ciclo recomeçado;
- **padrão 6** - representa um *rtt* que alterna seu nível (tipo onda quadrada) a cada 15 amostras. Fixados dois níveis, 225ms e 725ms, este padrão considera um *rtt* aleatório com desvio máximo de 50ms, tal como no padrão 1, mas que a cada 15 amostras sofre uma mudança brusca de nível;

- **padrão 7** - representa um *rtt* senoidal com período igual a 120 amostras. O nível foi fixado em 450ms e a faixa de variação da senóide, em 500ms.

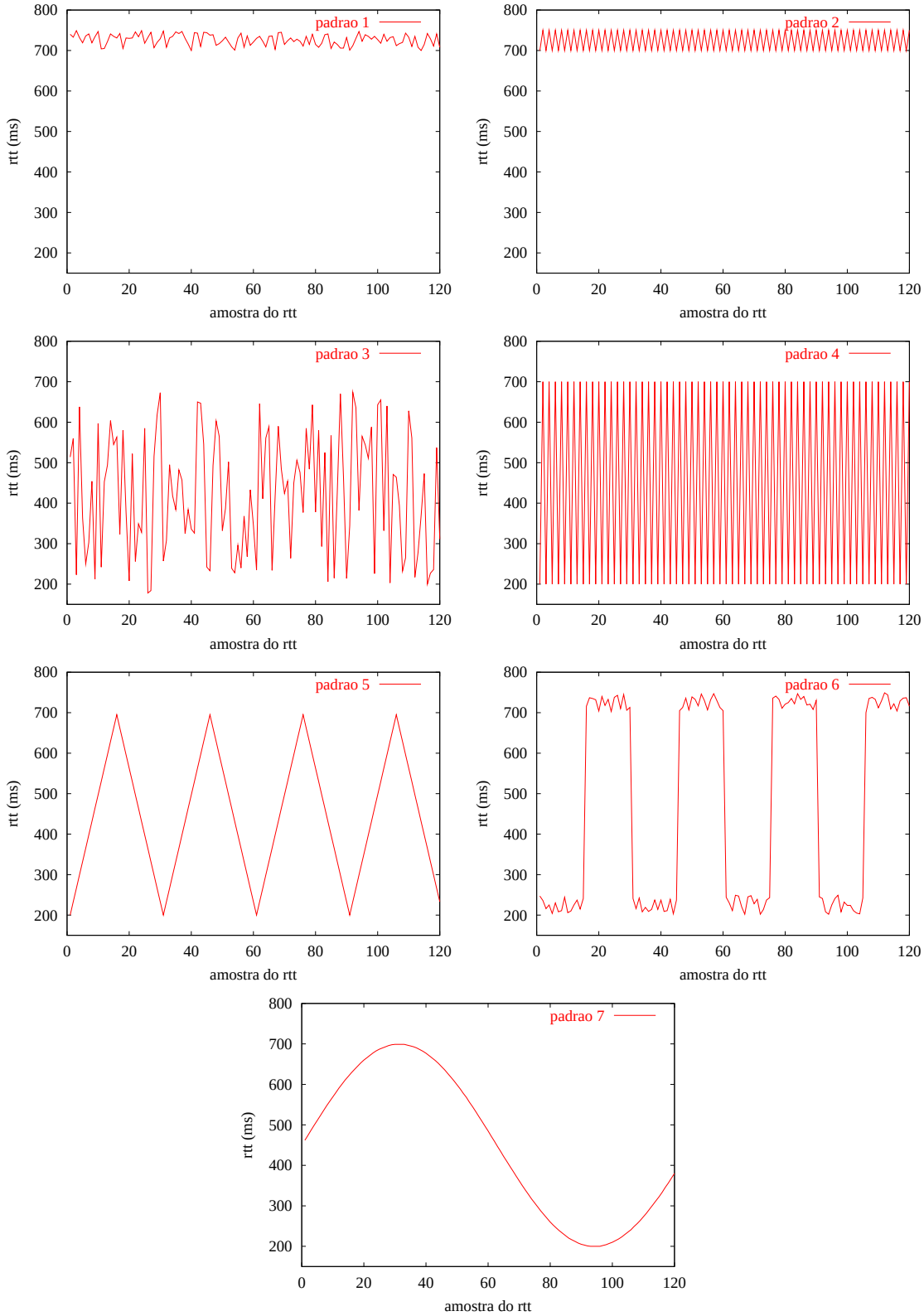

Figura 5.3: Padrões de amostragem fictícios.

Tabela 5.1: Erro quadrático médio dos preditores por padrão do *rtt*.

preditor	padrão 1	padrão 2	padrão 3	padrão 4	padrão 5	padrão 6	padrão 7
LAST	400,36	2497,50	45009,65	249750,00	1087,92	16648,99	78,36
MEAN	193,04	649,88	21408,39	62688,00	20567,69	62607,65	31185,92
WINMEAN	199,75	624,38	22165,23	62437,50	20539,55	62528,37	16560,32
DMA	204,23	624,38	22895,43	62437,50	20539,55	62514,46	7167,15
BROWN	228,48	783,22	25572,69	80575,34	13736,18	47656,80	611,26
ARIMA	194,17	0,00	21603,52	1,00	519,29	15493,44	35,91
LPF	206,70	728,20	23082,08	71217,71	16746,67	54111,23	4402,61

5.5.2 Análise da precisão dos preditores

Para analisar a precisão dos preditores, quando nas situações propostas pelos sete padrões descritos na seção anterior, computou-se o erro quadrático médio das previsões realizadas pelos preditores para cada um dos padrões. Os resultados obtidos estão listados na tabela 5.1.

Os resultados tabelados mostram que, independente da variância do *rtt*, se os valores do *rtt* seguirem um padrão aleatório sem tendência (nível constante), tal como nos padrões 1 e 3, o melhor preditor é o MEAN. Tal resultado está em sintonia com a teoria estatística (vide seção 5.3). Entretanto, nestes dois casos, o preditor ARIMA também consegue resultados muito próximos dos resultados obtidos pelo preditor MEAN, o que indica que a modelagem do *rtt* via séries temporais e um modelo preditivo ARIMA pode oferecer um estimador preciso para o *rtt* em situações de aleatoriedade. O contrário pode ser dito para o preditor LAST, o qual não se mostra adequado para um padrão comportamental aleatório. Os modelos de alisamento exponencial, tal como BROWN e LPF, também apresentaram uma pior precisão se relacionados aos modelos baseados na média amostral, tal como os modelos WINMEAN e DMA.

Por outro lado, se o padrão apresentar uma regularidade simples, tal como uma alternância a cada amostra conforme os padrões 2 e 4, o preditor ARIMA é capaz de identificar o comportamento do *rtt*, independentemente da variança da amostra, e realizar as melhores previsões, destacando-se significativamente em relação aos demais. Dentre eles, como era de se esperar, o padrão LAST demonstra ser o de menor precisão, e os preditores baseados na média apresentaram resultados satisfatórios, dado que a alternância ocorre simetricamente em relação a um nível médio. Os preditores baseados em técnicas de alisamento exponencial (BROWN e LPF) novamente não apresentaram bons resultados, ou seja, mostram-se inadequados para padrões de alta variabilidade entre amostras adjacentes.

Diferentemente dos primeiros 4 padrões, os padrões 5, 6 e 7 não apresentam um desvio muito grande entre duas amostras consecutivas. Esta característica faz o preditor LAST tornar-se um bom preditor em termos do erro quadrático médio, perdendo somente para o ARIMA nos três casos. Do resultado, percebe-se que se o *rtt* variar de maneira suave, o simples preditor LAST pode ser uma boa opção. Ao contrário, se esta variação suave ocorrer numa faixa de variação larga, tal como a de 500ms, os preditores baseados na média não são uma boa escolha. Salienta-se que este resultado pode ser extrapolado para o caso de um padrão onde ocorrem muitas variações com pequenos desvios entre amostras adjacentes, mas que é governado por variações suaves de tendência ou nível.

Para padrões que apresentam muitas variações num curto prazo, tais como os padrões de 1 a 4, exceto os preditores LAST e ARIMA, os outros preditores não apresentam diferenças significativas; entretanto, ao considerarmos variações de

tendências, tais como as dos padrões 5 e 7, ou de nível, tal como a do padrão 6, o preditor BROWN mostra-se atrativo. Tal resultado é consequência da sua excelente capacidade para se adaptar a mudanças de tendência ou nível (comportamento característico em sinais não estacionários). O preditor LPF, por considerar que os valores do *rtt* formam um sinal estacionário, é mais conservativo às mudanças de tendência ou nível, e conseqüentemente não consegue obter uma precisão melhor.

5.5.3 Análise comportamental das curvas de previsão

Para um entendimento mais completo dos resultados obtidos em função do erro quadrático médio, analisam-se também as curvas de previsão de cada preditor em cada um dos sete padrões.

A análise considera somente o período pós fase de inicialização dos preditores, ou seja, as previsões são realizadas após a série temporal do preditor ARIMA ser preenchida (após as primeiras 360 amostras). Nas figuras, a amplitude das curvas e a quantidade de amostras são escolhidas de modo a permitir uma melhor visualização do comportamento.

5.5.3.1 Quando o rtt varia de maneira aleatória a cada nova amostra (Padrões 1 e 3)

Quando o *rtt* corresponde a um ruído branco, tanto em situações de baixa variância (figura 5.4 ilustra comportamento no padrão 1) como em situações de alta variância (figura 5.5 ilustra comportamento no padrão 3), percebe-se que os preditores comportam-se de maneira previsível. O preditor MEAN realiza previsões praticamente no centro da faixa de variação permitida (de 700ms a 750ms no padrão 1 e de 200ms a 700ms no padrão 3), o que é esperado para um preditor que segue a média. Similarmente, a natureza aleatória do *rtt* faz com que o preditor ARIMA não identifique nenhum modelo autoregressivo e de médias móveis que represente o *rtt*, pois um ruído branco não é autocorrelacionado no tempo, e também realize previsões de acordo com a média das amostras. Por outro lado, os outros preditores tentam, a cada nova amostra, se ajustar a uma possível nova forma da curva do *rtt*, mas sem sucesso, acabam realizando previsões na maioria das vezes com um erro maior. Percebe-se também que o preditor LAST é o preditor que mais rápido tenta se ajustar a nova variação do *rtt*, seguido pelos preditores BROWN, LPF, DMA e WINMEAN, nesta ordem.

5.5.3.2 Quando o rtt se alterna a cada nova amostra (Padrões 2 e 4)

Quando o *rtt* apresenta um padrão conhecido, tal como o de alternância a cada nova amostra, independente da variância, se baixa ou alta, o preditor ARIMA tem boas chances de identificar o modelo autoregressivo e de médias móveis que descrevem o *rtt*. Conseqüentemente, este preditor tem boas chances de realizar previsões muito precisas do *rtt*. Esta eficácia pode ser observada nas figuras 5.6 e 5.7, onde a previsão do ARIMA é extremamente precisa, sobrepondo a curva do *rtt*. Complementarmente, as figuras também mostram que o preditor LAST é o pior deles em situações onde o *rtt* muda o sentido em relação a última amostra realizada.

Já os demais preditores, apresentam previsões próximas à média populacional, pois a alternância a cada amostra impede que haja variações significativas entre eles. Mesmo assim, percebe-se que a velocidade de ajuste ainda é maior no preditor

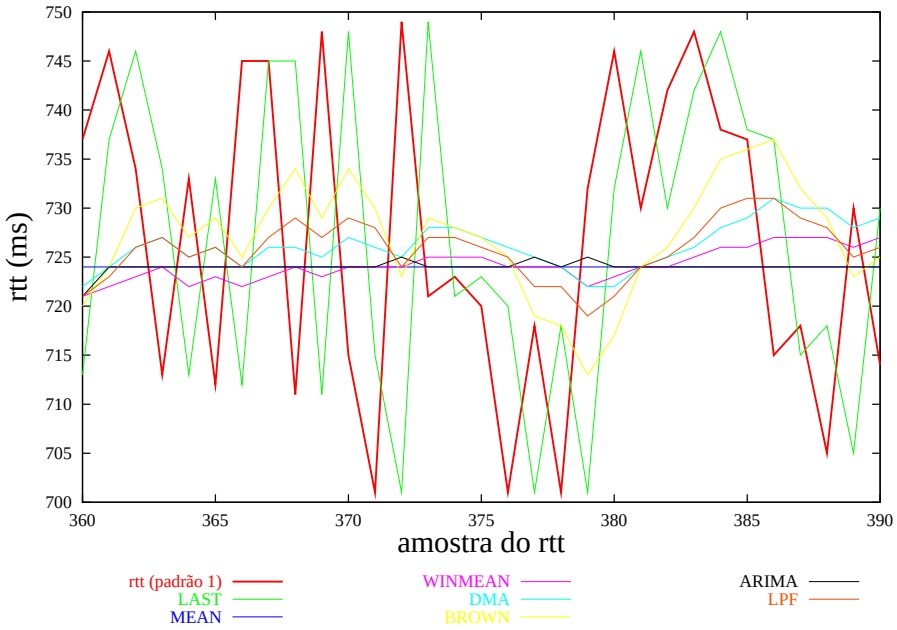

Figura 5.4: Comportamento dos preditores num período aleatório de baixa variância (Padrão 1).

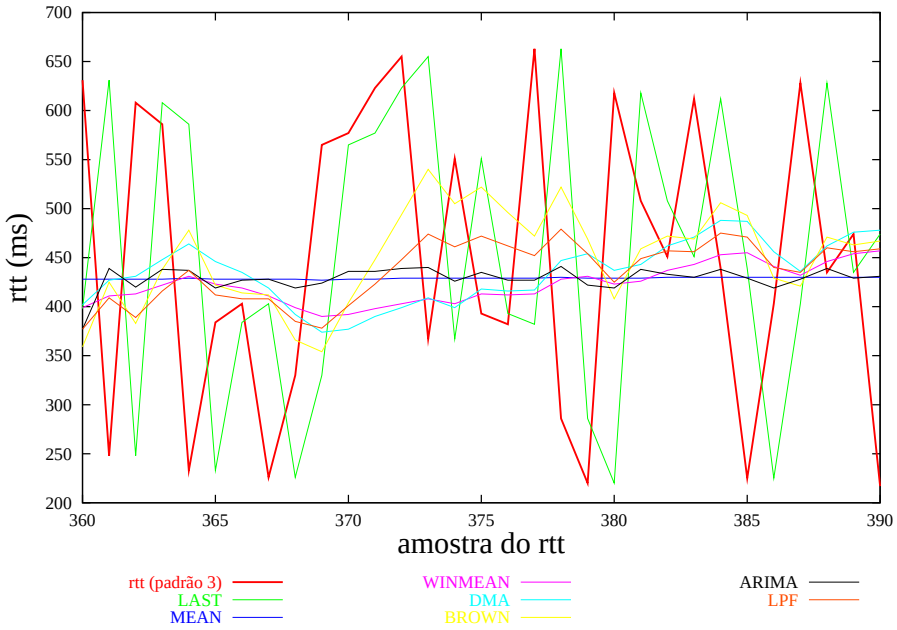

Figura 5.5: Comportamento dos preditores num período aleatório de alta variância (Padrão 3).

BROWN do que nos demais: LPF, DMA, WINMEAN e MEAN.

Destas figuras 5.4, 5.6, 5.5 e 5.7, também percebe-se que o preditor ARIMA pode ser conservador ou não em relação as mudanças nos valores das amostras. Seu conservadorismo dependerá da identificação do modelo a ser utilizado para realizar previsões, tornando importante a fase de identificação do modelo ARIMA.

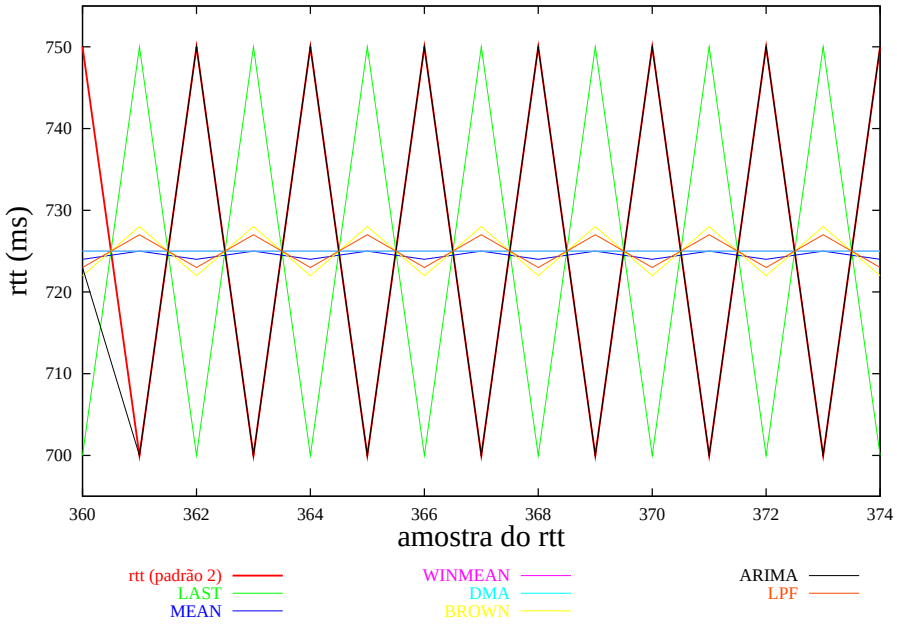

Figura 5.6: Comportamento dos preditores num período com alternância de baixa variância (Padrão 2).

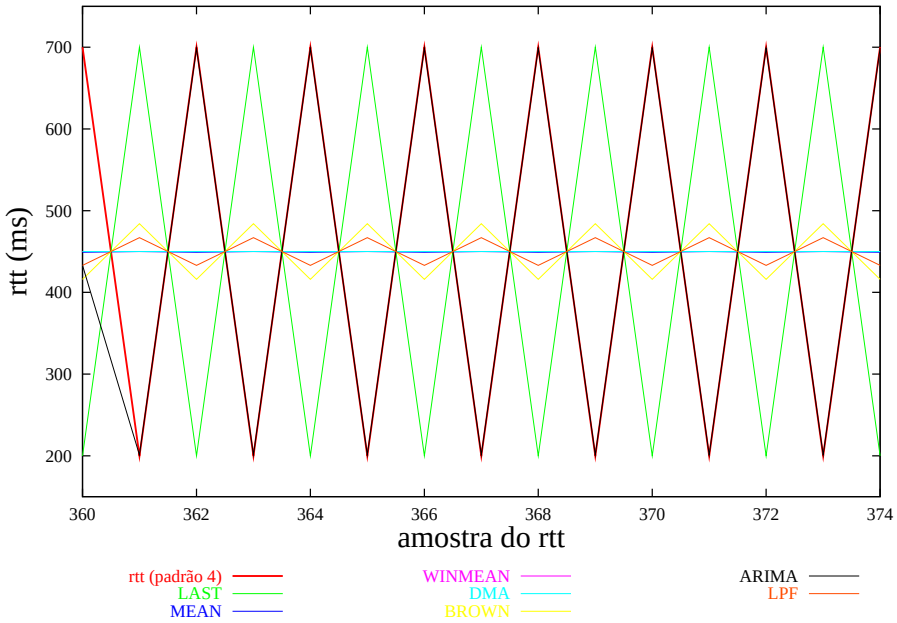

Figura 5.7: Comportamento dos preditores num período com alternância de alta variância (Padrão 4).

5.5.3.3 Quando o rtt muda sua tendência ou nível (Padrão 5 e 6)

Em situações onde um conjunto de amostras do *rtt* apresenta uma tendência linear (crescente ou decrescente), tal como ilustrado na figura 5.8, o preditor LAST realiza previsões relativamente precisas. A precisão deste preditor depende diretamente do desvio médio entre duas amostras consecutivas. Se esta tendência alterar seu sentido, rapidamente este preditor se ajustará a esta alteração, apresentando um erro grande somente no ponto da alteração de sentido. Similarmente, o preditor

ARIMA também apresenta um erro de previsão baixo, mas por outro motivo, por ele ter identificado um modelo ARMA(1,1) para representar o *rtt*.

Como a tendência se alterna a cada 15 amostras, ou seja, num período curto, os preditores basedados na média (MEAN, WINMEAN e DMA) não conseguem realizar boas previsões, ou seja, sempre predizem em torno do valor médio das amostras. Entretanto, observa-se que tanto o preditor LPF como o preditor BROWN, ambos baseados na técnica de alisamento exponencial, tentam acompanhar a variação de tendência, mas suas respostas são bem mais lentas do que as dos preditores ARIMA e LAST.

Embora todos os preditores apresentem estimativas defasadas no tempo, uma defasagem menor implica numa qualidade melhor do estimador fornecido, logo, justificando o menor erro quadrático médio do preditor ARIMA. A defasagem temporal também faz todos os preditores subestimarem todos os valores durante uma tendência positiva e superestimarem todos os valores numa tendência negativa.

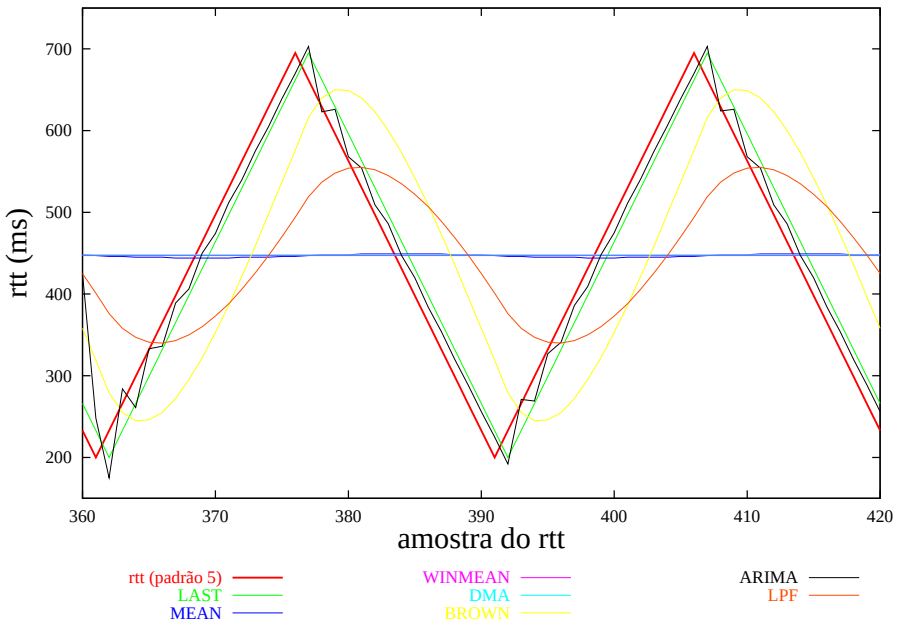

Figura 5.8: Comportamento dos preditores num período que alterna a tendência a cada 15 amostras (Padrão 5).

Em situações onde há mudanças bruscas de nível, conforme ilustra a figura 5.9, observa-se novamente que os únicos preditores capazes de se ajustarem rapidamente ao novo nível são os preditores ARIMA e LAST. Entretanto, observa-se que o preditor ARIMA, diferentemente do preditor LAST, sofre influência do valor médio do *rtt*, pois a estimativa no nível inferior é maior do que a média dos valores neste nível, enquanto a estimativa no nível superior é inferior aos valores medidos neste nível. Isto faz com que este preditor superestime o nível inferior e subestime o nível superior.

O comportamento dos outros preditores é similar ao já discutido. Dada à simetria de variação, os preditores baseados na média realizam suas previsões em torno do valor médio das amostras. Já os baseados em alisamento exponencial, quando conseguem evoluir ao ponto de prever valores próximos ao novo nível, a troca de nível faz com que eles apresentem um erro grande. Desta situação de

troca de nível também fica claro que os preditores BROWN e LPF precisam de pelos menos umas 20 amostras para conseguirem fornecer previsões relativamente precisas, quando há troca brusca de nível. Já o ARIMA, se conseguir identificar um padrão comportamental no conjunto de 360 amostras que utiliza para modelar a amostragem, ele consegue obter bons resultados.

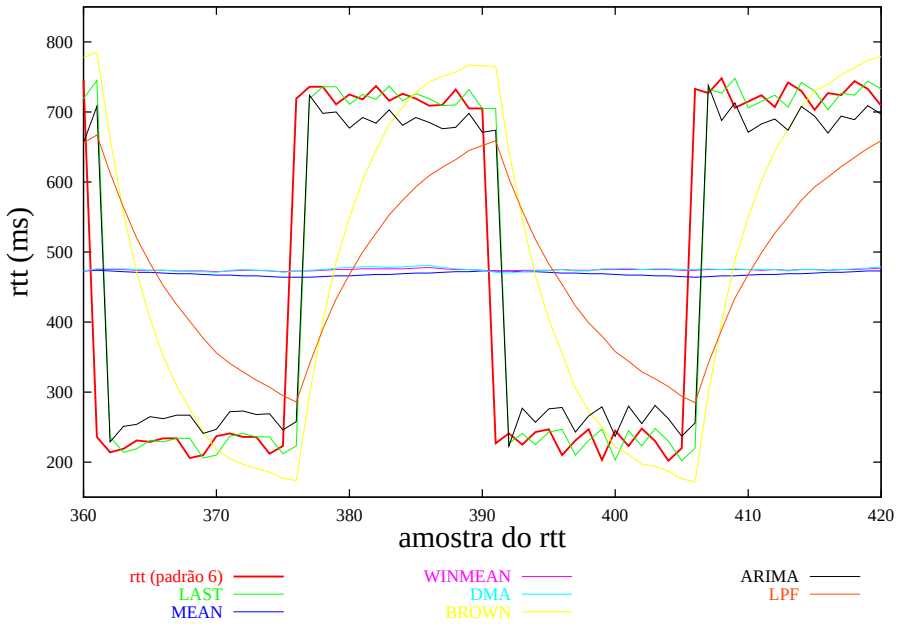

Figura 5.9: Comportamento dos preditores num período que alterna o nível a cada 15 amostras (Padrão 6).

5.5.3.4 Quando o rtt alterna sua tendência e nível de acordo com um comportamento senoidal (Padrão 7)

O sétimo padrão, representa uma situação de variação a mais longo prazo do que as anteriores. Nesta situação, o *rtt* alterna sua tendência e nível de acordo com um comportamento senoidal num período de 120 amostras. As curvas de previsão para cada preditor estão ilustradas na figura 5.10. Da figura, percebe-se que, se a variação se dá a longo prazo, somente o preditor MEAN apresenta um erro grande de previsão.

Dentre os outros preditores, novamente percebe-se que os erros menores são produzidos pelos preditores ARIMA e LAST, respectivamente. Percebe-se também que o preditor BROWN alcança rapidamente os novos valores, mas acaba atravessando a curva de amostragem e gerando erros de predição maiores nas trocas de tendência. O mesmo pode ser afirmado com relação a trocas de nível, conforme ilustra a figura 5.9.

5.6 Análise dos preditores com amostragens reais

Seguindo a metodologia de avaliação descrita na seção 5.4, cada análise realizada nas próximas subseções primeiro avalia estatisticamente os dados para posteriormente avaliar graficamente as previsões.

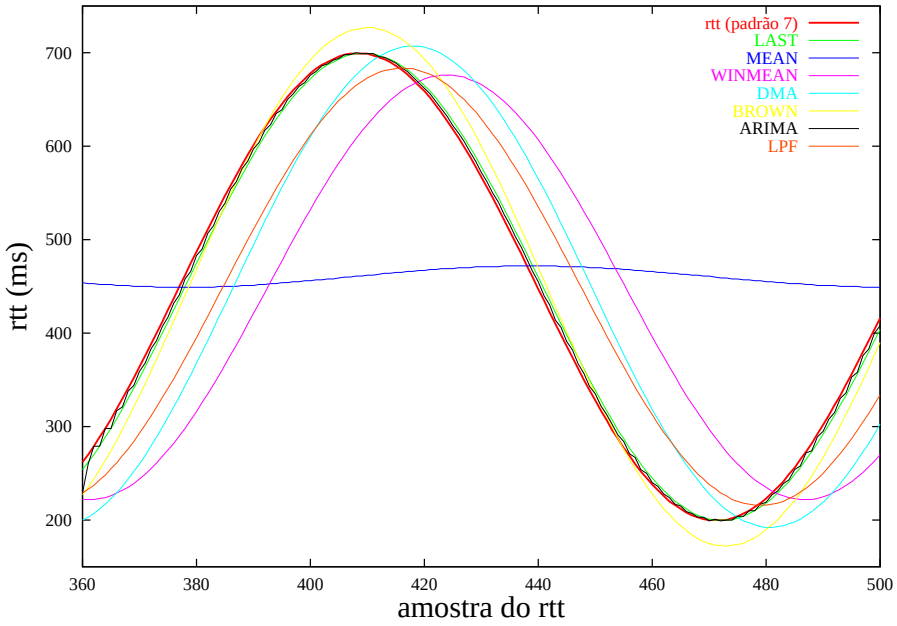

Figura 5.10: Comportamento dos preditores num período que alterna a tendência e nível de acordo com um padrão senoidal (Padrão 7).

5.6.1 Análise das amostragens com 86400 amostras

Teoricamente, de um ponto de vista estatístico, se a amostragem não apresentar correlação serial no tempo, ou seja, se a amostragem for composta por um nível mais um ruído branco, o preditor MEAN é o que deve apresentar o menor erro quadrático médio (seção 5.3). Entretanto, em geral valores adjacentes do *rtt* apresentam correlação serial significativa (seção 4.2). Logo, considerando a variação típica do *rtt* ao longo de um dia, para uma amostragem com aproximadamente 86400 amostras, que apresenta estabilidade e baixa variância durante a madrugada e instabilidade e alta variância durante o dia, espera-se que um preditor baseado na média aritmética de todos os valores passados, tal como o preditor MEAN, não consiga obter bons resultados.

Tomando como exemplo a curva de previsão do preditor MEAN frente aos valores do *rtt* amostrados na comunicação UFRGS-UFCG e UFRGS-POP/PA na mesma data, correspondente a uma quarta-feira, conforme ilustra a figura 5.11, claramente pode-se perceber que este preditor é muito conservador em relação as flutuações e conseqüentemente não consegue acompanhar as variações diárias do *rtt*.

A baixa precisão do preditor MEAN também pode ser comprovada pelo seu alto erro quadrático médio. Na tabela 5.2, são apresentados os erros quadráticos médios de todos os preditores para a quarta-feira ilustrada na figura 5.11, donde fica evidente a baixa precisão do preditor MEAN. A quarta coluna da tabela apresenta a média dos erros quadráticos nos 23 dias amostrados e mostra que a baixa precisão do MEAN também é observada nos demais dias amostrados. O resultado da análise da variância ($Pr < 0,05$) indica que há pelo menos um dos modelos que difere estatisticamente dos demais, e o teste de Duncan (tabela 5.3) mostra que a desigualdade é em relação ao preditor MEAN, o qual está isolado dos demais no grupo A, grupo de mais alto erro quadrático médio. Em síntese, tais resultados indicam que o preditor MEAN não é adequado para realizar previsões do *rtt* de uma mensagem.

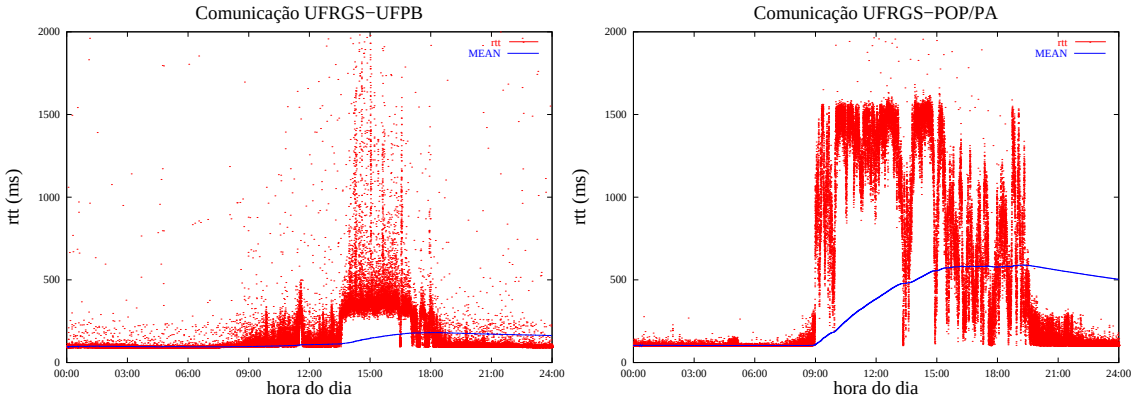

Figura 5.11: Curva de previsão do preditor MEAN numa quarta-feira.

Tabela 5.2: Erro quadrático médio por preditor.

Preditor	EQM		
	comunicação UFRGS-UFCG dia 20/06/2001	comunicação UFRGS-POP/PA dia 20/06/2001	média para os 23 dias amostrados
LAST	11836,79	4473,54	7980,70
MEAN	29163,05	273776,19	78639,44
WINMEAN	13018,96	6458,71	10020,80
DMA	12577,32	5299,08	9240,07
BROWN	10981,41	3662,95	7279,54
ARIMA	9762,74	3436,59	6406,88
LPF	10871,38	4115,28	7569,49

No lado oposto a um estimador conservador, tem-se o estimador que procura se adaptar a grandes flutuações, baseando-se somente no último valor amostrado, implementado pelo preditor LAST. Embora o estimador LAST não use estatísticas dos valores passados, se considerado o erro quadrático médio, ele demonstrou ser melhor do que os estimadores baseados na média de um histórico, tal como os preditores: MEAN, que segue a média de toda a população amostral; WINMEAN, que segue a média de uma janela contendo somente as últimas 30 amostras; e DMA, que segue a média das últimas 30 médias de uma janela de 30 amostras. Tal resultado pode ser claramente observado na tabela 5.3, onde o teste de Duncan indica: que não há diferença estatística entre os preditores, com exceção do preditor MEAN; que o erro quadrático médio do preditor LAST é menor do que os preditores baseados na média, mas maior do que o dos preditores baseados em alisamento exponencial (LPF

Tabela 5.3: Resultado do teste de Duncan a 5% de significância para amostragens com 86400 amostras.

Grupo	Média	N	MODEL
A	78639	23	MEAN
B	10021	23	WINMEAN
B	9240	23	DMA
B	7981	23	LAST
B	7569	23	LPF
B	7280	23	BROWN
B	6407	23	ARIMA

e BROWN) ou em termos autoregressivos e de médias móveis (modelo ARIMA).

Por outro lado, o fato do preditor LAST ser estatisticamente equivalente aos demais preditores, excetuando o MEAN, e mais preciso do que os preditores baseados na média, não garante que ele seja adequado para um detector de defeitos. Por exemplo, tomando por base a amostragem da conexão UFRGS-UFCG no dia 20/06/2001 (apresentada na figura 5.11), as previsões realizadas pelos preditores LAST, MEAN, WINMEAN e DMA em situações de alta e baixa variância do *rtt*, conforme ilustrado no período das 11:43 às 11:46h (figura 5.12) tem-se que quando os valores do *rtt* variam moderadamente de maneira crescente ou decrescente, num conjunto de 10 a 20 amostras, a estimativa do preditor LAST normalmente está mais próxima do valor amostrado do que a estimativa realizada pelos preditores baseados na média (MEAN, WINMEAN e DMA), os quais são mais conservadores. Por outro lado, se o valor do *rtt* apresentar uma alta variância, tal como o comportamento do *rtt* por volta das 11:45h, seguir o último valor não resulta numa boa estimativa, principalmente quando se deseja evitar falsas suspeitas, pois a instabilidade do *rtt* irá refletir-se numa instabilidade no valor do *timeout*, o que pode aumentar a probabilidade de falsas suspeitas.

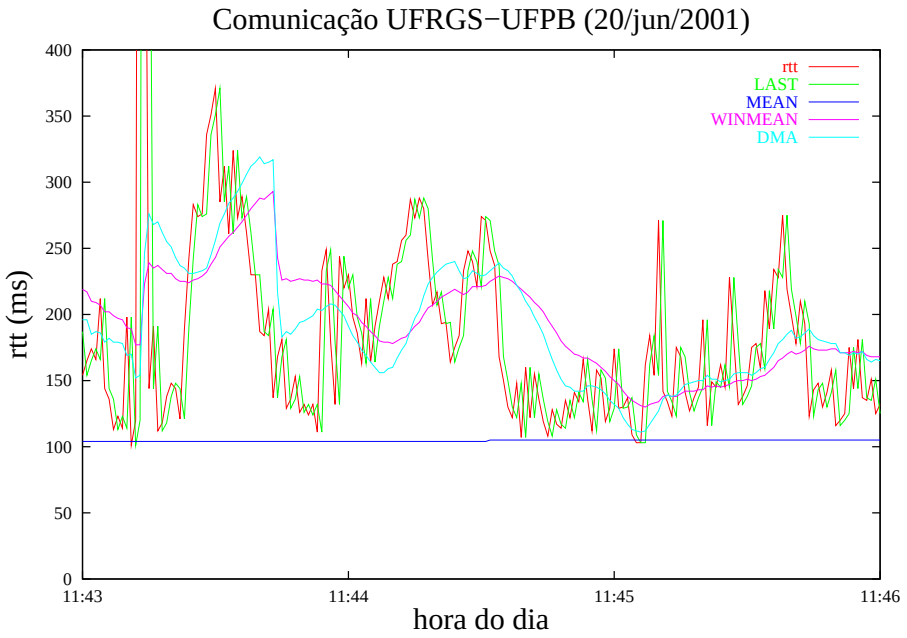

Figura 5.12: Curva de previsão dos preditores LAST, MEAN, WINMEAN e DMA em situações de alta e baixa variância do *rtt*.

Pela figura 5.12, também percebe-se que o preditor DMA, o qual adota a hipótese que a série do *rtt* segue um comportamento não estacionário, responde mais rapidamente a mudanças de tendência do *rtt* do que o preditor WINMEAN. Tal velocidade de resposta é importante para reduzir a latência para detecção de um defeito, conforme será discutido no capítulo 6. Por isto, dentre os preditores do grupo B (tabela 5.3) que se baseam na média aritmética dos valores passados (preditores WINMEAN e DMA), o DMA é melhor em termos do erro quadrático médio.

Diferentemente dos preditores baseados na média aritmética, os preditores LPF e BROWN baseiam-se na técnica de alisamento exponencial simples e duplo,

respectivamente, o que significa computar a previsão atribuindo pesos diferenciados aos valores do histórico de primeira e segunda ordem (amostras recentes pesam mais do que amostras remotas). Esta característica resulta em preditores mais eficientes, se comparado aos preditores baseados na média aritmética, pois: apresentam menores erros quadráticos médios (tabela 5.3); e são mais velozes para se adaptar às mudanças de tendência (figura 5.13). Da figura 5.13, a qual mostra o comportamento dos preditores DMA, BROWN, ARIMA e LPF no mesmo período ilustrado pela figura 5.12, observa-se que embora estatisticamente equivalente, ao considerar um comportamento não estacionário, o preditor BROWN obtém previsões com menor *eqm* do que o preditor LPF, conforme pode ser confirmado pelo erro quadrático médio indicado na tabela 5.3, e responde mais rapidamente às variações de tendência.

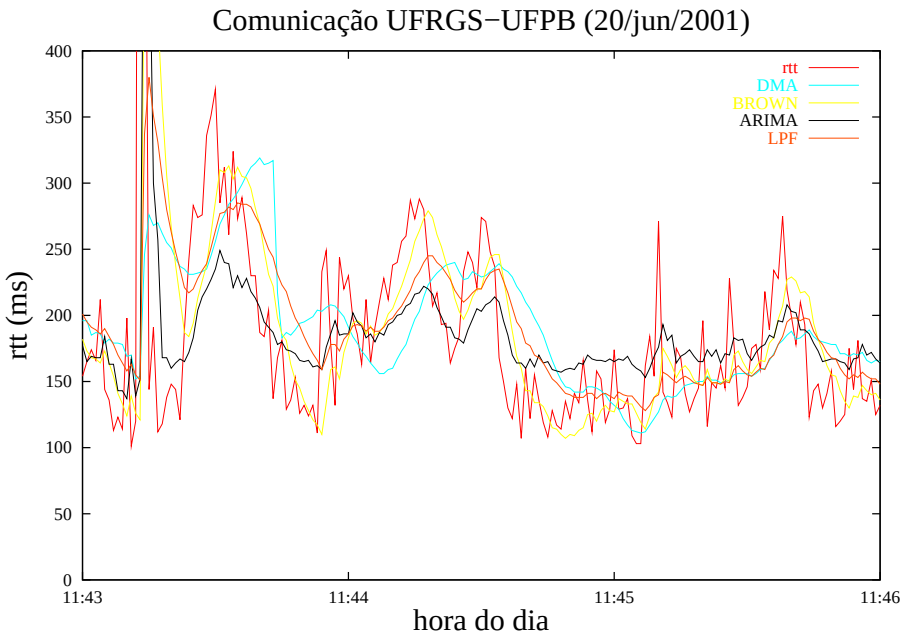

Figura 5.13: Curva de previsão dos preditores DMA, BROWN, ARIMA e LPF em amostragens de 86400 amostras sob situações de alta e baixa variância.

Da figura 5.13, também observa-se que o preditor ARIMA é mais conservador do que os preditores baseados em alisamento exponencial, no que diz respeito à velocidade de adaptação às variações na tendência. Entretanto, em execuções de 24 horas, este preditor mostra-se mais eficiente do que os demais, em termos do erro quadrático médio, conforme indicado nas tabelas 5.2 e 5.3.

Dos resultados apresentados nessa seção, nota-se que para amostragens de 24 horas:

- o preditor MEAN é o pior dos preditores;
- excetuando-se o preditor MEAN, estatisticamente não há diferença entre os estimadores fornecidos pelos preditores implementados;
- o estimador fornecido pelo preditor BROWN é o que mais rapidamente consegue responder a variações de tendência; e

Tabela 5.4: Resultado do teste de Duncan a 5% de significância para amostragens com 64800 amostras.

Grupo	Média	N	MODEL
A	120196	17	MEAN
B	17217	17	WINMEAN
B	15866	17	DMA
B	13386	17	LAST
B	12928	17	LPF
B	12389	17	BROWN
B	10737	17	ARIMA

- o estimador fornecido pelo preditor ARIMA é o que apresentou a melhor qualidade em termos do erro quadrático médio.

5.6.2 Análise das amostragens com 64800 amostras

Nesta seção, discute-se a análise da qualidade dos preditores quando aplicados às amostragens coletadas durante os dias de semana, no período das 6:00h às 24:00h, período em que ocorrem mais variações no rtt.

Em termos estatísticos, a análise da variância indica que pelo menos um dos modelos difere significativamente dos demais ($Pr < 0,05$). De acordo com o resultado do teste de Duncan (tabela 5.4), percebe-se que somente o preditor MEAN difere dos demais. Além disto, observa-se que o resultado se equivale ao obtido para amostragens de 24 horas, ou seja: o preditor MEAN é o pior dos preditores; os preditores LAST, WINMEAN, DMA, BROWN, ARIMA e LPF são estatisticamente equivalentes; e o preditor ARIMA é o que oferece o melhor estimador em termos do erro quadrático médio.

Para a análise gráfica, novamente avaliou-se a capacidade dos preditores em fornecer estimadores que respondam rapidamente a variações na tendência. Para fins de comparação, a figura 5.14 ilustra o comportamento dos preditores DMA, BROWN, ARIMA e LPF, no mesmo período apresentado na figura 5.13. Das figuras, pode-se observar que o comportamento dos preditores é idêntico ao obtido para amostragens de 24 horas. Este resultado era esperado, uma vez que os preditores tendem a esquecer rapidamente do seu histórico remoto, conforme pode ser comprovado pelas equações 5.6, 5.9, 5.10 e 5.4 (respectivamente correspondentes aos preditores citados).

Para ilustrar que os comportamentos independem do tamanho da amostragem (em períodos pós inicialização), na figura 5.15, mostra-se o comportamento dos preditores DMA, BROWN, ARIMA e LPF no período das 8:41h às 8:44h sob a comunicação UFRGS-POP/PA, ilustrado originalmente na figura 5.11. O gráfico (a) mostra o comportamento dos preditores quando eles começam a executar às 6:00h e o gráfico (b) quando eles começam a executar às 8:35h.

Do gráfico (a), observa-se que os preditores apresentam comportamentos próprios bem definidos, ou seja:

- o preditor BROWN é o que se ajusta mais rapidamente a variações de tendência;
- o preditor ARIMA é o que realiza previsões mais precisas, se considerado o seu erro médio; e
- o preditor DMA é o que responde mais lentamente a variações na tendência

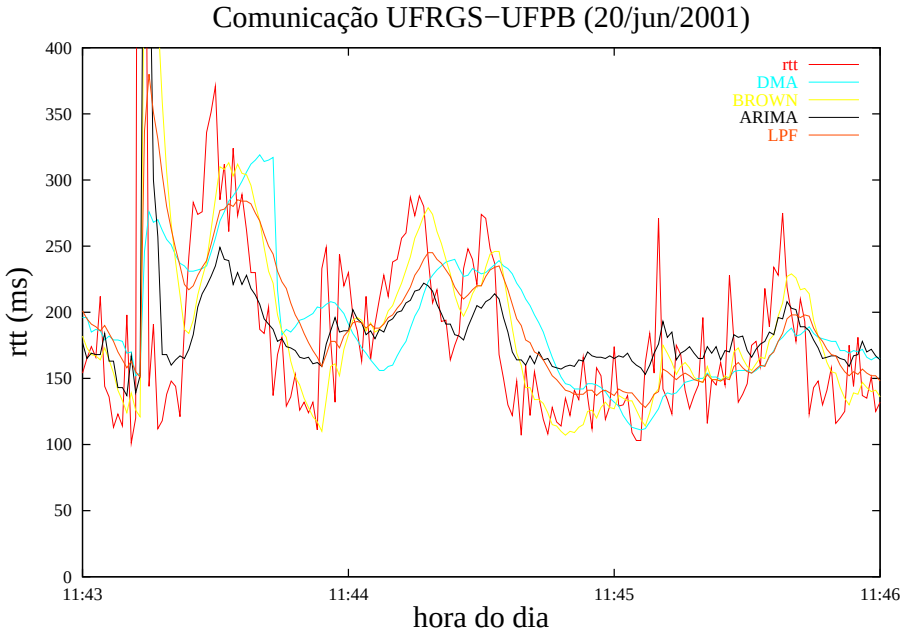

Figura 5.14: Curva de previsão dos preditores DMA, BROWN, ARIMA e LPF em amostragens de 64800 amostras sob situações de alta e baixa variância.

do *rtt*.

Do gráfico (b), observa-se que os comportamentos citados acima começam a ocorrer tão logo acabe a fase de inicialização, às 8:41:17h, ou seja, independem do padrão descrito pela amostragem. Cabe salientar que a perda de 17 mensagens na fase de inicialização do preditor ARIMA, faz com que o seu estado interno só esteja pronto para realizar previsões segundo modelos ARIMA às 08:41:17h, ao invés das 08:41:00h, como poderia se esperar. O tamanho da série temporal foi ajustado para ter 360 amostras.

5.6.3 Análise das amostragens com tamanhos variados

Considerando somente o período pós-inicialização dos preditores, esta seção analisa estatisticamente o comportamento de cada preditor considerando execuções de tamanhos variados (com 300, 600, 900, 1200, 1500, 1800, 2100, 2400, 2700, 3000, 3300 e 3600 amostras). O objetivo é identificar se o comportamento com períodos de execução de tamanhos diferentes pode mesmo ser considerado semelhante aos já obtidos para amostragens de 24 e 18 horas, o que indicaria que as características dos preditores também valem para amostragens de qualquer tamanho (desde que maiores do que o mínimo necessário para preencher a fase de inicialização).

Pela análise estatística das 8160 amostragens, separadas em conjuntos de 680 amostragens para cada um dos 12 tamanhos de amostragens indicadas, pode-se verificar que pelo menos um modelo sempre difere estatisticamente dos demais (vide tabela 5.5): o estimador fornecido pelo preditor MEAN, considerado o pior deles.

Dos 12 tamanhos de série avaliados, somente na análise das séries com 300 e 600 amostras o preditor MEAN não está sozinho num grupo, estando junto com o preditor DMA (tabela 5.5). Este resultado é equivalente ao já obtido para execuções de longa duração (86400 e 68400 amostras), onde este preditor mostrou-se pouco eficaz na predição do *rtt*. No lado oposto ao preditor MEAN, o estimador fornecido

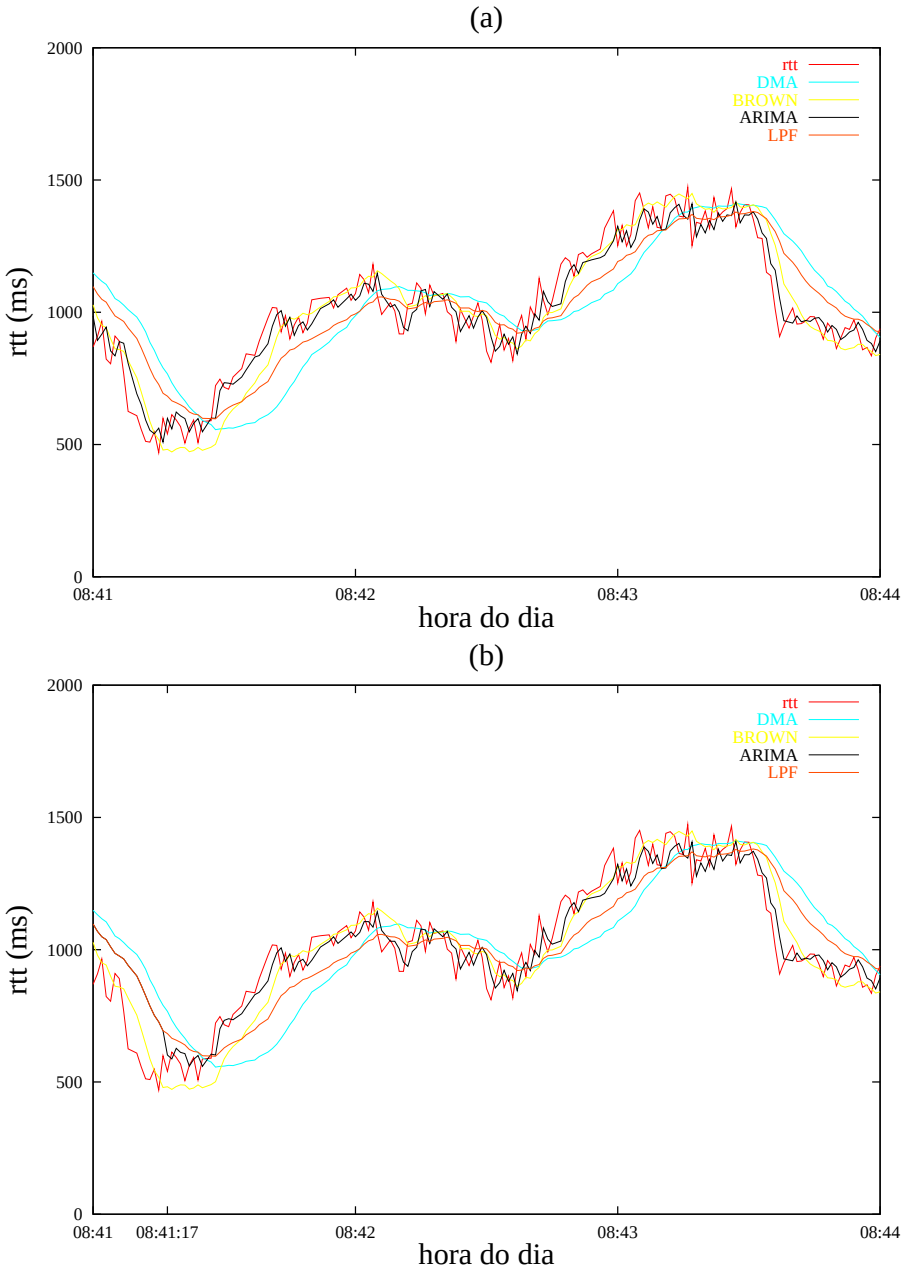

Figura 5.15: Curvas de previsão dos preditores DMA, BROWN, ARIMA e LPF na comunicação UFRGS-POP/PA (20/jun/2001) sob dois instantes de inicialização.

pelo preditor ARIMA mostrou-se o mais preciso em termos do erro quadrático médio (obteve o menor *eqm* em todos os tamanhos de execuções).

Embora os grupos resultantes do teste de Duncan não tenham indicado que o preditor ARIMA seja estatisticamente diferentes dos demais, percebe-se que ele sempre pertence ao grupo dos melhores, juntamente com os preditores LAST, LPF e BROWN. Além disto, percebe-se que dentro do melhor grupo, o estimador fornecido pelo preditor ARIMA é sempre o mais preciso, seguido por BROWN, LPF e LAST, nesta ordem. Tal ordem indica que, embora não se possa distingüí-los em termos estatísticos, há uma ordem de precisão, em relação ao erro quadrático médio, que independe do tamanho do período de execução. A figura 5.16 ilustra claramente esta ordem ao apresentar uma relação dos erros quadráticos médios obtidos por cada um,

Tabela 5.5: Resumo dos resultados do teste de Duncan a 5% de significância.

	300 amostras		600 amostras		900 amostras		1200 amostras	
Preditor	Grupo	eqm	Grupo	eqm	Grupo	eqm	Grupo	eqm
MEAN	A	25443	A	28022	A	32206	A	34955
DMA	A	24333	A	25805	B	17910	B	15978
WINMEAN	B	12576	B	12724	C	12786	B e C	13150
LAST	B	11400	B	11426	C e D	11435	D e C	11766
LPF	B	9602	B	9565	C e D	9560	D	9812
BROWN	B	9230	B	9235	C e D	9241	D	9485
ARIMA	B	8595	B	8634	D	8582	D	8792

	1500 amostras		1800 amostras		2100 amostras		2400 amostras	
Preditor	Grupo	eqm	Grupo	eqm	Grupo	eqm	Grupo	eqm
MEAN	A	37030	A	38636	A	40712	A	43341
DMA	B	15208	B	14667	B	14566	B	15060
WINMEAN	B e C	13426	B	13481	B e C	13858	B e C	14707
LAST	B, C e D	12383	B, C e D	12846	B, C e D	12902	B, C e D	13318
LPF	C e D	10136	C e D	10301	C e D	10487	B, C e D	11088
BROWN	C e D	9852	C e D	10069	D	10204	C e D	10749
ARIMA	D	9135	D	9389	D	9512	D	9910

	2700 amostras		3000 amostras		3300 amostras		3600 amostras	
Preditor	Grupo	eqm	Grupo	eqm	Grupo	eqm	Grupo	eqm
MEAN	A	46722	A	48162	A	49776	A	52019
DMA	B	15940	B	15992	B	16232	B	16577
WINMEAN	B	15910	B	16125	B	16515	B	17002
LAST	B e C	13682	B e C	13908	B e C	14227	B e C	14487
LPF	B e C	11831	B e C	12028	B e C	12338	B e C	12676
BROWN	B e C	11380	B e C	11578	B e C	11877	B e C	12186
ARIMA	C	10294	C	10415	C	10644	C	10815

em diferentes tamanhos de execução.

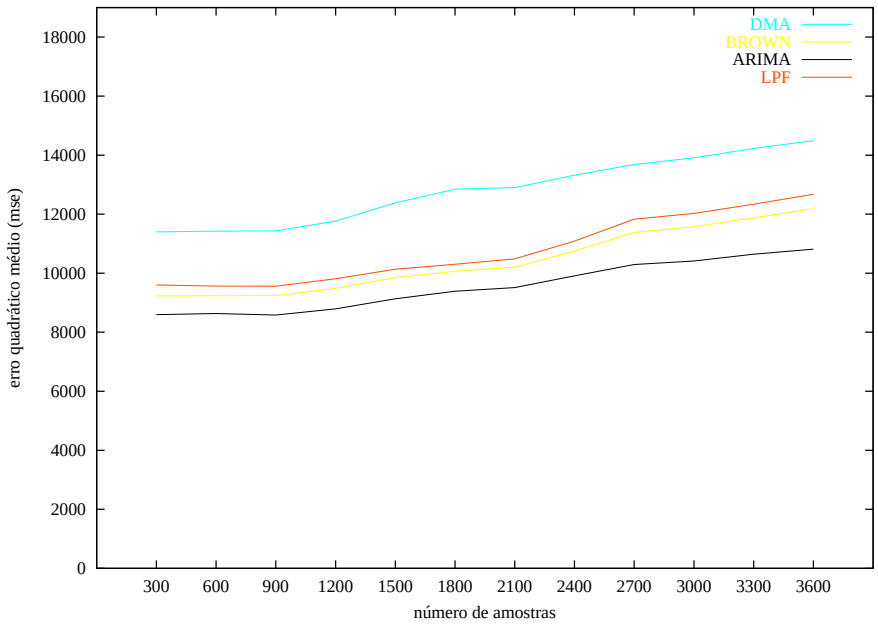

Figura 5.16: Erro quadrático médio para amostragens de tamanhos diferentes.

5.7 Conclusões Parciais

Neste capítulo, foram especificados e avaliados sete tipos diferentes de preditores (LAST, MEAN, WINMEAN, DMA, BROWN, LPF e ARIMA), cada um explorando métodos estatísticos com características distintas. O mais simples deles,

o preditor LAST, usa como parâmetro de inferência somente o valor absoluto do último valor amostrado. Sob a hipótese de não tendenciosidade, o preditor MEAN baseia-se na média aritmética do histórico de toda a população, o preditor WINMEAN baseia-se na média móvel simples dos últimos 30 valores amostrados e o preditor LPF baseia-se na média exponencialmente ponderada dos valores passados (alisamento exponencial). Sob a hipótese de que os valores do rtt formam uma série temporal não estacionária, os preditores DMA e BROWN baseiam-se na média móvel dupla e no alisamento exponencial duplo, respectivamente. Similarmente, sob a hipótese de que os valores passados formam uma série temporal, mas podendo ser estacionária ou não, o preditor ARIMA dinamicamente identifica e ajusta um modelo ARIMA para representar as últimas 360 amostras do rtt, e então se baseia neste modelo para realizar previsões.

Utilizando um conjunto de padrões de amostragem, observou-se que preditores baseados na média só conseguem bons resultados em amostragens que tendem a valores aleatórios. Quando o padrão de amostragem apresenta um comportamento possível de ser identificado (não é aleatório), o preditor ARIMA é o que tem melhores condições para realizar previsões precisas, pois pode: *i*) responder rápido a mudanças bruscas, como no caso do padrão 6; *ii*) ser conservador em relação a mudanças, como no caso do padrão 1 e 3; ou mesmo, *iii*) prever exatamente o próximo valor, como no caso dos padrões 2 e 4. Preditor LAST, embora simples, também oferece bons resultados, mas somente nos casos onde não há muita variação entre amostras adjacentes.

As análises gráfica e estatística dos diversos preditores, sob amostragens reais, mostram que o preditor MEAN não é adequado para representar os dados, pois não consegue acompanhar as variações da curva do *rtt*, resultando num erro quadrático médio extremamente alto em relação aos demais preditores.

Embora estatisticamente a diferença entre os outros preditores (excetuando o MEAN) não seja tão evidente, em termos do erro quadrático médio, os preditores DMA e WINMEAN são menos precisos do que os preditores LAST, LPF, BROWN e ARIMA. Já dentre os quatro preditores mais precisos, em termos do erro quadrático médio, o preditor ARIMA é o mais preciso deles em praticamente todos os casos, seguido pelos preditores BROWN, LPF e LAST, nesta ordem.

Em termos comportamentais, em situações de alta variância, o preditor LAST não é adequado porque reflete os ruídos da variável de interesse na estimação do próximo valor, tornando a previsão também aleatória e com alta variabilidade. Já em situações de mudança de tendência, o preditor BROWN demonstrou ser o mais eficiente, pois seu comportamento é o que responde mais rapidamente às mudanças de tendência. Os preditores LPF e ARIMA também respondem bem a variações rapidamente, mas de forma mais suave do que com BROWN.

Finalmente, conclui-se que preditores baseados em modelos de séries temporais oferecem bons resultados tanto em termos de precisão como em termos de velocidade de resposta a mudanças de tendência. Além disto, também conclui-se que os preditores baseados em termos autoregressivos são os mais precisos (ARIMA, BROWN e LPF), principalmente quando consideram a não estacionariedade do *rtt* (ARIMA e BROWN).

6 INFLUÊNCIA DO PREDITOR NA QUALIDADE DO DETECTOR DE DEFEITOS

Neste capítulo, mostra-se que a utilização de um preditor baseado em séries temporais, tal como o preditor ARIMA, quando associado a uma margem de segurança compatível com os requisitos que se deseja, auxilia na obtenção de uma qualidade de serviço (QoS) melhor de um detector de defeitos. Para demonstrar este resultado, utilizam-se sete implementações de um detector de defeitos do estilo *pull*, cada uma implementando um dos preditores especificados no capítulo anterior, e, com o auxílio de um simulador, medem-se diferentes parâmetros que são reconhecidos como métricas adequadas à avaliação de detectores.

O capítulo está organizado como segue: inicialmente, apresentam-se as métricas de qualidade de serviço utilizadas (seção 6.1), descreve-se o ambiente de simulação utilizado para a computação das métricas (seção 6.2), descreve-se a metodologia para computação das métricas no ambiente simulado (seção 6.3), e definem-se as margens de segurança adicionadas aos valores estimados pelos preditores (seção 6.4). Em seguida, faz-se um relacionamento entre métricas de qualidade de serviço e o *timeout* (seção 6.5) e realiza-se a análise da QoS das diferentes implementações do detector de defeitos do estilo *pull* (seção 6.6). As conclusões parciais são apresentadas na seção 6.7.

6.1 Métricas de QoS

Até pouco tempo atrás, a precisão de detectores de defeitos baseados em *timeout*s vinha sendo avaliada através da métrica *probabilidade de* timeout*s prematuros* (RAYNAL; TRONEL, 1999), ou seja, pela probabilidade de um temporizador exceder seu limite para um componente que não está defeituoso. Entretanto, para comparar diferentes implementações de detectores de defeitos, esta métrica não é apropriada porque:

(a) depende da forma que a implementação foi realizada; e

(b) é pouco útil para as aplicações, a menos que seja apresentada junto com outras medidas específicas relativas à implementação como, por exemplo, com que freqüência os temporizadores são inicializados, se a inicialização dos temporizadores ocorre em intervalos regulares ou variáveis, se os *timeout*s são fixos ou variáveis, etc.

Por outro lado, métricas gerais para avaliação de algoritmos distribuídos, tal como o *número de mensagens trocadas* e o *grau de latência* (SCHIPER, 1997), são importantes para avaliar a escalabilidade e o desempenho do algoritmo utilizado por um detector de defeitos, mas não permitem avaliar a qualidade de detectores que usam o mesmo algoritmo distribuído como, por exemplo, detectores de defeitos do

estilo *pull* que diferem por implementar métodos diversos de previsão de valores. Portanto, o ideal é que diferentes implementações de detectores de defeitos sejam comparadas utilizando métricas de avaliação que independem da implementação do detector.

Um conjunto de sete métricas que independem da implementação do detector foi proposto por Chen, Toueg e Aguilera (2002). Tais métricas, por avaliarem probabilisticamente o comportamento de um detector de defeitos sem fazer referência a *timeout*s, foram adotadas neste trabalho para servirem de base à análise comparativa da influência de diferentes tipos de preditores na qualidade de serviço de um detector de defeitos.

As novas métricas são apresentadas em dois grupos: as primárias e as secundárias. As métricas secundárias podem ser derivadas das métricas primárias. Considerando um sistema com dois componentes p e q, onde o módulo de detecção de defeitos de p monitora q, estas métricas são assim definidas (CHEN; TOUEG; AGUILERA, 2002):

Métricas primárias

1. **Tempo para detecção de um defeito** (*Detection time* - T_D) - mede a velocidade de detecção de defeitos, ou seja, T_D é o tempo decorrido desde o instante em que q falha (colapso) até o instante em que p suspeita permanentemente de q. Probabilisticamente, T_D é a variável aleatória associada ao tempo decorrido do instante em que q falha (colapso) até o instante em que ocorre a última transição do estado Confiável (*Trust*) para o estado Suspeito (*Suspect*), S-transição, sem haver mais transições.

2. **Tempo para recorrência ao erro** (*Mistake recurrence time* - T_{MR}) - mede o tempo entre dois erros consecutivos cometidos pelo detector (suspeitas incorretas). Probabilisticamente, T_{MR} é a variável aleatória que representa o tempo entre duas S-transições consecutivas.

3. **Duração de um erro** (*Mistake duration* - T_M) - mede o tempo que o detector de defeitos leva para corrigir um erro. Probabilisticamente, T_M é a variável aleatória correspondente ao tempo decorrido entre uma S-transição e a próxima T-transição (passagem do estado Suspeito para o estado Confiável).

Métricas derivadas

4. **Taxa de erros média** (*Average mistake rate* - λ_M) - mede a taxa com que o detector de defeitos erra, isto é, representa o número médio de S-transições por unidade de tempo.

5. **Probabilidade de uma resposta precisa** (*Query accuracy probability* - P_A) - é a probabilidade de que a saída do detector de defeitos esteja correta num dado tempo aleatório.

6. **Duração de um período bom** (*Good period duration* - T_G) - mede a duração de um período bom, ou seja, o tempo decorrido entre uma T-transição e a próxima S-transição.

7. **Duração de um período bom à frente** (*Forward good period duration* - T_{FG}) - é uma variável aleatória que representa o tempo restante do período bom, ou seja, dado um instante aleatório, representa o tempo restante até o momento da próxima S-transição.

Salienta-se que, dentre as métricas, somente a primeira relaciona-se ao desempenho do detector de defeitos; as outras avaliam a sua precisão. Das seis métricas de precisão, as duas primeiras são as mais importantes, pois delas pode-se derivar as quatro seguintes, conforme define o teorema 1 apresentado em (CHEN; TOUEG; AGUILERA, 2002), reproduzido a seguir. No teorema, $E(X)$ e $V(X)$ representam a esperança e a variância da variável aleatória X, respectivamente, e $Pr(A)$ representa a probabilidade do evento A.

Teorema 1 *Para um detector de defeitos ergódico*[1]*, valem os seguintes resultados:*
(1) $T_G = T_{MR} - T_M$
(2) Se $0 < E(T_{MR}) < \infty$*, então*

$$\lambda_M = \frac{1}{E(T_{MR})}$$

$$P_A = \frac{E(T_G)}{E(T_{MR})} = \frac{E(T_{MR}) - E(T_M)}{E(T_{MR})}$$

(3) Se $0 < E(T_{MR}) < \infty$ *e* $E(T_G) = 0$*, então* T_{FG} *é sempre* 0.
Se $0 < E(T_{MR}) < \infty$ *e* $E(T_G) \neq 0$*, então para todo* $x \in [0, \infty)$*,*

$$Pr(T_{FG} \leq x) = \frac{1}{E(T_G)} \int_0^x Pr(T_G > y)dy$$

$$E(T_{FG}^k) = \frac{E(T_G^{K+1})}{(k+1)E(T_G)}$$

Em particular,

$$E(T_{FG}) = \frac{E(T_G^2)}{2E(T_G)} = \frac{E(T_G)}{2}\left(1 + \frac{V(T_G)}{E(T_G)^2}\right)$$

Neste trabalho, consideram-se as três métricas primárias e apenas duas dentre as quatro métricas secundárias: a taxa de erros média (λ_M) e a probabilidade de uma resposta precisa (P_A). As duas métricas secundárias escolhidas são importantes porque refletem a precisão de um detector de defeitos em duas situações muito utilizadas na prática: (1) quando o detector informa assincronamente uma suspeita à aplicação, situação onde a menor taxa de erros é importante; e (2) quando o detector é interrogado pela aplicação, situação onde a probabilidade de uma resposta precisa é importante.

[1]Ergodicidade é um conceito da teoria dos processos estocásticos que, aplicado ao caso em estudo, indica que, em execuções livres de defeitos, o detector de defeitos lentamente esquece seu histórico passado, ou seja, seu comportamento futuro depende somente do seu comportamento recente (CHEN, 2000, p. 25).

6.2 Ambiente de simulação

Há pelo menos duas abordagens para realizar a computação das métricas de qualidade de serviço de um detector de defeitos:

- instrumentar cada implementação do detector de defeitos e executá-las num ambiente distribuído real; ou
- construir um simulador capaz de computar as métricas de interesse com base em dados previamente coletados.

A primeira abordagem tem a vantagem de tratar com situações reais de funcionamento, mas tem a desvantagem de não permitir que duas implementações distintas sejam avaliadas sob um mesmo conjunto de dados, pois cada execução (amostragem) resulta numa série distinta do *rtt* (variável estocástica), ou seja, os dados amostrados em cada execução serão diferentes se as execuções forem realizadas em tempos diferentes ou mesmo em execuções concorrentes.

A segunda abordagem, embora não corresponda a uma execução real, permite repetir uma dada execução tantas vezes quantas foram necessárias. Além disto, também permite que as execuções sejam realizadas com base em amostragens reais. Estas características são importantes numa análise comparativa, onde se deseja avaliar diferentes implementações do detector de defeitos frente ao mesmo conjunto de dados.

Adotando a segunda abordagem, neste capítulo, avalia-se a qualidade de serviço das sete implementações do detector de defeitos do estilo *pull* computando as métricas num simulador alimentado com os dados reais coletados numa rede *wide area* (na seção 4.2.1, foram apresentados os detalhes sobre a realização da coleta dos dados e sobre as amostragens coletadas).

Implementado na linguagem de programação Java (JDK 1.3 da Sun Microsystems (2003)), executando sobre um kernel Linux 2.4 (Distribuição RedHat 7.2 (2002)), o simulador foi projetado para realizar previsões do *timeout* considerando a política de *supressão de lacunas* (explicada na seção 4.1).

6.3 Computação das métricas no simulador

Nesta seção, explica-se como foram computados os instantes de transição de estado mantidos pelo detector de defeitos e as métricas de qualidade de serviço.

6.3.1 Computação dos instantes de transição

De acordo com a especificação das métricas de qualidade de serviço (seção 6.1), as métricas devem ser computadas com base no histórico de transições de estado de um detector de defeitos, onde o histórico indica precisamente todos os instantes de transição de estado confiável para suspeito ($t_{Stransicao}$) e todos os instantes de transição de estado suspeito para confiável ($t_{Ttransicao}$).

No simulador, o histórico ordenado de transições é representado por $H_{transicao}$ e é definido como

$$H_{transicao} = \cup_{k=1}^{n} t_{Stransicao}^{k} \bigcup \cup_{k=1}^{n} t_{Ttransicao}^{k}$$

onde n indica tanto o número de S-transições quanto o de T-transições da execução.

No ambiente real, a computação dos instantes de transição corresponde à leitura do relógio local sempre que uma troca de estado for percebida. No ambiente simulado, para ficar independente do relógio local, a computação dos instantes de transição é realizada somente em função das referências temporais de um detector do estilo *pull*, ou seja: do tempo de ida e volta de uma mensagem (rtt), do intervalo de interrogação (t_i), do *timeout* (t_o), do número de seqüência da mensagem enviada na qual esta sendo contabilizado o *timeout* ($nseq$) e do número de seqüência da mensagem recebida ($mseq$).

Como tais referências são previamente conhecidas (t_i, rtt, $nseq$ e $mseq$), ou computadas (t_o), pelo simulador, ao invés do componente monitor do simulador enviar uma mensagem req a cada t_i unidades de tempo para o componente monitorável, busca-se na base de dados a tupla de informações $\langle mseq, rtt \rangle$ que o detector obteria ao processar o recebimento de uma mensagem de ack.

Da tupla $\langle mseq, rtt \rangle$ sabe-se que a mensagem req_{mseq} foi enviada rtt instantes de tempo antes da mensagem ack_{mseq} ser recebida. Logo, adotando como referência o instante de tempo do envio da primeira mensagem req, ou seja, o envio de req_0 (início da execução do módulo de monitoração de defeitos), sabe-se que o instante de recebimento de uma mensagem ack_{mseq} é dado por $mseq.t_i + rtt$, e que o instante de estouro do *timeout* associado a esta mensagem é dado por $mseq.t_i + t_o$.

Deste modo, num primeiro instante, considerando uma tupla $\langle mseq, rtt \rangle$ o simulador poderia verificar se o t_o expirou, testando se $mseq.t_i + rtt > mseq.t_i + t_o$. Entretanto, na prática, o controle do intervalo de tempo cujo limite corresponde ao *timeout* é disparado no instante do envio das mensagens req e é interrompido no recebimento das mensagens ack. Assim, se o emissor enviar uma nova mensagem antes do controle do *timeout* ter sido interrompido e antes dele ter expirado, o emissor não reinicializa t_o, pois nenhuma mensagem ack foi recebida desde que o controle do *timeout* foi disparado. Como conseqüência, para verificar se um dado *timeout* expirou, o simulador utiliza o $nseq$ correspondente ao número de seqüência da mensagem req que disparou a última contagem do *timeout*, e compara se $(mseq.t_i + rtt) > (nseq.t_i + t_o)$.

O número de seqüência $nseq$ é ajustado a cada passo para um valor igual a $mseq + 1$, caso $mseq \geq nseq$. Caso $mseq < nseq$, a mensagem é considerada antiga (inválida) e é descartada, não gerando alteração no $nseq$.

Sempre que uma mensagem válida é observada, o *timeout* é recomputado considerando um histórico de amostras do rtt, sobre o qual é aplicado a política de supressão de lacunas.

Agora, seja $H_{transicao}[t-1]$ o instante da última transição do estado *suspeito* para *confiável*. Para uma tupla $\langle mseq, rtt \rangle$, uma nova suspeita só poderá ser gerada se $rtt > t_o$ para a mensagem de número $mseq$ e se o componente monitorável não estiver sob suspeita, ou seja, $mseq.t_i + rtt > mseq.t_i + t_o > H_{transicao}[t-1]$.

Se uma nova suspeita é verificada, então duas transições de estado são inseridas no histórico de transições $H_{transicao}$: uma transição do estado *confiável* para o estado *suspeito*, $t_{Stransicao} = nseq.t_i + t_o$, e uma transição do estado *suspeito* para o estado *confiável*, $t_{Ttransicao} = mseq.t_i + rtt$, nesta ordem.

Em síntese, cada vez que uma suspeita é percebida no simulador, duas transições são computadas: $H_{transicao}[t] \leftarrow t_{Stransicao}$ e $H_{transicao}[t+1] \leftarrow t_{Ttransicao}$.

A lógica descrita para computação dos instantes de transição foi implementada usando o algoritmo da figura 6.1, onde t_o corresponde ao mais recente *timeout*

utilizado, t_i corresponde ao intervalo de interrogação utilizado pelo módulo monitor e o índice t indica a ordem das transições no histórico de transições.

inicialização:
  $t \leftarrow 1$;
  $seq \leftarrow$ série coletada;
  $H_{transicao}[0..2*\text{tamanho da série}] \leftarrow 0$;
  $nseq \leftarrow 1$;
  $t_i \leftarrow$ período de interrogação;
  $t_o \leftarrow$ *timeout* inicial;

for all $\langle mseq, rtt \rangle \in seq$ **do**
  busca uma tupla $\langle mseq, rtt \rangle$ em seq;
  if $mseq \geq nseq$ **then**
    if $(mseq.t_i + rtt) > (nseq.t_i + t_o) > H_{transicao}[t-1]$ **then**
      $H_{transicao}[t] \leftarrow nseq.t_i + t_o$;
      $H_{transicao}[t+1] \leftarrow mseq.t_i + rtt$;
      $t \leftarrow t + 2$;
    end if
    $nseq \leftarrow mseq + 1$;
    computa novo t_o;
  end if
end for

Figura 6.1: Algoritmo que computa o histórico de transições de estado.

6.3.2 Computação das métricas de QoS

Considerando que, na prática, os atrasos de comunicação nos dois sentidos da comunicação são relativamente semelhantes, a computação das métricas de qualidade de serviço de um dado detector de defeitos adota a seguinte hipótese: o atraso de comunicação de uma mensagem *req* é idêntico ao atraso de comunicação da mensagem *ack* correspondente. Com esta hipótese, o cômputo da métrica de desempenho T_D pode ser simplificado, pois no pior caso, quando a falha no componente monitorável ocorre logo após o envio de um *ack*, o cômputo do T_D pode considerar que a falha ocorreu a $rtt/2$ unidades de tempo do último envio de uma mensagem *req*. O uso desta hipótese não traz prejuízos para a análise comparativa, pois diferenças significativas (quando o atraso num sentido é freqüentemente maior do que no sentido inverso) seriam percebidas por todos os preditores e desapareceriam ao serem realizadas as análises estatísticas.

Cômputo da métrica de desempenho T_D - para computar esta métrica o simulador age como se a falha no componente monitorável ocorresse logo após este componente responder a uma mensagem de *ack*. Na prática, este procedimento poderia ser implementado através de um injetor de falhas que impedisse a geração de novas mensagens de *ack*. Entretanto, como escolher o momento adequado para injetar uma falha? O simulador resolve esta questão computando T_D toda vez que uma mensagem *ack* for recebida, ou seja, o T_D relatado pelo simulador corresponde a média de todas as possíveis medidas numa dada execução sob simulação.

Para atender ao pior caso de falha, caso que deve ser observado pela métrica T_D, o simulador vale-se da hipótese de que o atraso de comunicação de uma mensagem req é idêntico ao atraso de comunicação da mensagem de ack correspondente, ou seja, o instante de ocorrência de uma falha t_{falha} é definido como sendo

$$t_{falha} = nseq.t_i + \frac{rtt_{nseq}}{2}$$

Assim, considerando t_{falha}, o tempo para percepção de um defeito gerado por esta falha (tempo de detecção de um defeito) é definido como

$$t_d = (nseq + 1).t_i + t_o - t_{falha} = t_i + t_o - \frac{rtt_{nseq}}{2} \tag{6.1}$$

onde t_o corresponde ao *timeout* computado quando do recebimento do ack_{nseq}, o qual tem a contagem disparada quando do envio da mensagem req de número $nseq + 1$, e rtt_{nseq} corresponde ao atraso de comunicação de ida e volta correspondente ao par de mensagens $\langle req_{nseq}, ack_{nseq} \rangle$. Logo, o tempo médio para detecção de um defeito é dado por

$$T_D = \frac{\sum_{k=1}^{n} t_d^k}{n} \tag{6.2}$$

onde n é o número de vezes que t_d foi cômputado numa dada execução.

Adicionalmente, como o requisito de qualidade de serviço associado ao T_D pode ser o seu limite superior T_D^u (pior caso), o simulador também computa este limite pesquisando pelo maior T_D computado.

Cômputo das métricas de precisão primárias - diferentemente do T_D, as métricas de precisão são definidas com base em execuções livres de falhas e em função dos instantes de transição de estado $t_{Stransicao}$ e $t_{Ttransicao}$.

Seguindo a definição das métricas de precisão (seção 6.1), o tempo de duração de um erro (duração de uma falsa suspeita) é computado por

$$t_m = t_{Ttransicao} - t_{Stransicao}$$

onde $t_{Stransicao}$ e $t_{Ttransicao}$ são adjacentes e $t_{Stransicao}$ precede $t_{Ttransicao}$. Ciente que as transições foram inseridas no histórico $H_{transicao}$ de maneira ordenada e em pares, conforme o algoritmo 6.1, tem-se que

$$t_m = H_{transicao}[t+1] - H_{transicao}[t] \text{ para todo } t < n * 2 \mid t = 1, 3, 5, ..$$

onde $H_{transicao}[t+1] = t_{Ttransicao}$ e $H_{transicao}[t] = t_{Stransicao}$. Logo, o tempo médio de duração de um erro é dado por

$$T_M = \frac{\sum_{k=1}^{n} t_m^k}{n} \tag{6.3}$$

onde n é o número de falsas suspeitas observadas.

De maneira similar, o tempo de recorrência ao erro é computado por

$$t_{mr} = H_{transicao}[t+2] - H_{transicao}[t] \text{ para todo } t < (n-1) * 2 \ , \ t = 2, 4, 6, ..$$

onde $H_{transicao}[t+2]$ e $H_{transicao}[t]$ são dois instantes $t_{Stransicao}$ adjacentes. Logo, o tempo médio de recorrência ao erro é dado por

$$T_{MR} = \frac{\sum_{k=1}^{n-1} t_{mr}^{k}}{n-1} \tag{6.4}$$

onde n é o número de falsas suspeitas observadas.

Cômputo das métricas de precisão derivadas - conhecidas as métricas primárias, as métricas derivadas podem ser computadas de acordo com os relacionamentos definidos pelo teorema descrito na seção 6.1.

6.4 Margem de segurança

Como a previsão fornecida por um preditor pode ser um valor menor do que o valor real amostrado, para estabelecer um *timeout* em função de um valor previsto é necessário fazer uso de uma *margem de segurança*. Em termos probabilísticos, a margem de segurança é um fator de ajuste que, quando aplicado a um valor previsto de uma variável aleatória, deve garantir, com algum grau de confiança, que o valor a ser observado esteja dentro de um determinado intervalo de previsão (BOX; JENKINS; REINSEL, 1994, p. 2). Em outras palavras, o uso de uma margem de segurança na determinação do *timeout* deve garantir que o valor previsto sobreponha o valor real a ser amostrado, com uma probabilidade P (previamente determinada).

Conforme abordado no capítulo 2, na área de *timeout*s adaptativos, os valores previstos pelos preditores são ajustados de acordo com três tipos de margem de segurança:

1. uma margem *empírica* - no OGS (*Object Group Service* - seção 2.3.3), é considerado que o próximo *rtt* possa sofrer uma variação na ordem de duas vezes o último *rtt*, o que remete a uma margem de segurança igual ao dobro do último *rtt* amostrado. Salienta-se que esta abordagem não permite que se definam intervalos de confiança adequadamente, e que o *timeout* resultante é tão aleatório quanto o *rtt*;

2. uma margem fixa *definida em função dos requisitos de QoS* - esta abordagem é adotada no algoritmo NFD-E (CHEN; TOUEG; AGUILERA, 2002), um algoritmo que supõe relógios não sincronizados e um comportamento desconhecido para o atraso de comunicação. Um procedimento de configuração prévio determina a periodicidade de envio de mensagens t_i em função dos requisitos de qualidade de serviço e ajusta a margem de segurança Δ para $\Delta = T_D^u - t_i$, onde T_D^u corresponde à qualidade de serviço requerida pelo usuário para a métrica de desempenho. Numa dada configuração, onde t_i é constante, a margem de segurança também será um valor constante, pois T_D^u também é constante;

3. uma margem *variável de acordo com o erro médio da previsão* - definida no protocolo TCP (seção 2.3.6), esta margem de segurança corresponde à média exponencialmente ponderada dos desvios (erros) do *rtt*. Recentemente, esta abordagem também foi utilizada no detector de defeitos adaptativo de Bertier, Marin e Sens (2002). Esta margem varia constantemente.

Como nenhuma destas margens se preocupa em atender requisitos de confiabilidade, ou seja, dar garantias de que o valor estimado sobreponha o valor real com uma probabilidade previamente definida, uma quarta margem é proposta:

4. uma margem variável em *função do intervalo de confiança da previsão* - baseado na hipótese de que os valores previstos seguem um modelo estocástico de forma conhecida, ou seja, que tanto o estimador a um passo como o erro de previsão ε são normalmente distribuídos, esta margem corresponde à amplitude do intervalo de confiança de uma dada previsão, para um nível de confiança previamente especificado.

Neste capítulo, a análise da qualidade de serviço dos detectores considera três cenários: sem margem de segurança (margem *nula*), com uma margem variável de acordo com o erro médio de previsão (margem *ep*) e com uma margem variável de acordo com o intervalo de confiança da previsão (*ic*). Estas margens foram escolhidas porque a margem *ep* realiza um alisamento exponencial do erro de previsão ε, enquanto que a margem *ic* garante uma margem pequena, mas suficiente para sobrepor o *timeout* com 99% de confiança. As margens *ep* e *ic*, utilizadas na análise deste capítulo, são especificadas nas seções 6.4.2 e 6.4.3, respectivamente.

6.4.1 Separando a margem em dois fatores

Assumindo que o *timeout* pode ser definido somente em função do atraso de comunicação e de uma margem de segurança Δ, a expressão geral para determinação dinâmica de um *timeout* t_o pode ser expressa como:

$$t_o = \widehat{rtt} + \Delta \tag{6.5}$$

onde $\widehat{rtt}$ é o valor previsto para a próxima observação do tempo de ida e volta de uma mensagem de controle. Em tal expressão, o valor de Δ deve ser suficiente para sobrepor as variações temporais do atraso de comunicação. Adicionalmente, Δ também pode refletir incrementos necessários para implementar propriedades axiomáticas que impliquem em terminação num tempo finito como, por exemplo, incrementos gerados de acordo com a percepção de suspeitas incorretas. Logo, a margem de segurança Δ pode ser dividida em dois fatores, tal como:

$$\Delta = \beta + \delta \tag{6.6}$$

onde β é o fator de ajuste responsável exclusivamente por sobrepor variações nos atrasos de comunicação e δ é o fator de ajuste responsável por sobrepor qualquer outro tipo de influência decorrente de observações do sistema (suspeitas incorretas) ou decorrente das hipóteses adotadas (perda de mensagens).

Com divisão da margem de segurança em dois fatores, os mecanismos de adaptação, como o incremento do período de envio de mensagens de controle (*exponential speed-up* utilizado no protocolo de *heartbeat* acelerado) ou o mecanismo para incremento do *timeout* no caso de falsas suspeitas (incremento k utilizado no Jgroup), refletem-se na expressão do *timeout* exclusivamente através do fator δ.

A divisão de Δ em dois fatores possibilita implementar uma abordagem de análise onde um dos fatores é determinado de maneira dinâmica e outro de maneira estática.

Fixando o fator δ, da equação 6.6 tem-se $\delta = 0$ e conseqüentemente pode-se avaliar comparativamente a influência dos diferentes preditores na qualidade de serviço do detector de defeitos, ou seja, as variações na margem de segurança decorrem somente das variações na qualidade da previsão (especificamente através do fator β, que é uma função do $\widehat{rtt}$).

6.4.2 Margem variável de acordo com o erro médio da previsão

Baseando-se na abordagem dinâmica de ajuste proposta por Jacobson para o protocolo TCP (seção 2.3.6), o fator de ajuste β da margem de segurança Δ corresponde ao erro médio exponencialmente alisado, o qual é computado como

$$\beta_{t+1} = \beta_t + \alpha.(\mid rtt_t - \widehat{rtt}_t \mid -\beta_t) \tag{6.7}$$

onde α é a constante de alisamento. Conforme sugerido por Jacobson, faz-se $\alpha = 0,25$, o que resulta numa rápida resposta do fator β às variações no erro.

Por ser uma função do erro, esta margem tende a ser maior quando a variabilidade do rtt é grande. Isto faz o *timeout* aumentar e conseqüentemente o número de falsas suspeitas diminuir. Por outro lado, nenhum nível de confiança pode ser garantido, ou seja, não se pode afirmar que com esta margem 99% das amostras do rtt serão menores do que o *timeout*.

6.4.3 Margem variável em função do intervalo de confiança da previsão

Como a previsão do *timeout* corresponde a previsão do rtt, realizada em função da amostragem observada, o grau de confiança de um dado *timeout* será uma função do seu intervalo de confiança. A determinação da amplitude do intervalo de confiança da previsão depende da distribuição amostral dos erros de previsão e do grau de confiança desejado.

Fazendo a margem de segurança corresponder à amplitude do intervalo de confiança da previsão, pode-se garantir, com uma dada probabilidade, que o valor do *timeout* será maior do que o próximo rtt que será amostrado. Para tanto, é necessário especificar a margem de segurança de acordo com a função de distribuição do estimador.

Logo, a seguir primeiro define-se a distribuição de probabilidades adotada para o erro de previsão, para então definir a equação para computar o fator de ajuste β da margem de segurança Δ, em função do intervalo de confiança da previsão.

Por hipótese, considera-se que o valor do intervalo de interrogação é grande o suficiente para que cada observação do tempo de ida e volta de uma mensagem, rtt_i, seja independente das observações anteriores, a partir das quais se obteve $\widehat{rtt}_i$. Com esta hipótese faz-se o rtt_i ser independente do seu estimador ($\widehat{rtt}_i$), e conseqüentemente faz-se o $\widehat{rtt}_i$ ser independente dos estimadores passados.

Para amostras independentes, supondo que o estimador a um passo apresenta um comportamento linear[2] e segue uma distribuição Normal, tem-se que a distribuição de probabilidade do estimador $\widehat{rtt}_i$ é dada por (WERKEMA, 1996)

$$\widehat{rtt}_i \approx N\left(a + b.rtt_i \ , \ \sigma^2\left[\frac{1}{n} + \frac{(rtt_i - \overline{rtt}_i)^2}{\sum_{i=1}^{n}(rtt_i - \overline{rtt})^2}\right]\right) \tag{6.8}$$

[2] O adjetivo linear é usado aqui para indicar que o modelo é linear nos parâmetros e não que seja uma função linear dos valores estimados no passado.

onde σ^2 e $\widehat{rtt}$ correspondem à variância e à média da população, respectivamente.

Considerando que o preditor modela adequadamente o comportamento dos dados, tem-se que o erro de previsão ε, definido como $\varepsilon_i = rtt_i - \widehat{rtt}_i$, é um ruído branco, ou seja, a esperança do erro é zero e sua variância é dada por

$$\begin{aligned} V(\varepsilon_i) &= V(rtt_i) + V(\widehat{rtt}_i) \\ &= \sigma^2 + \sigma^2 \left[\frac{1}{n} + \frac{(rtt_i - \overline{rtt}_i)^2}{\sum_{i=1}^{n} (rtt_i - \overline{rtt})^2} \right] \\ &= \sigma^2 \left[1 + \frac{1}{n} + \frac{(rtt_i - \overline{rtt}_i)^2}{\sum_{i=1}^{n} (rtt_i - \overline{rtt})^2} \right] \end{aligned} \tag{6.9}$$

Com estas características, têm-se que a distribuição de probabilidades do erro de previsão ε_i é

$$\varepsilon_i \approx N\left(0\ ,\ \sigma^2 \left[1 + \frac{1}{n} + \frac{(rtt_i - \overline{rtt}_i)^2}{\sum_{i=1}^{n} (rtt_i - \overline{rtt})^2} \right]\right) \tag{6.10}$$

onde σ^2 pode ser estimada por

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} (rtt - \widehat{rtt}_i)^2}{n-2} \tag{6.11}$$

Conhecidas as distribuições de probabilidades do estimador e do erro de previsão e considerando um histórico com mais de 30 amostras ($n > 30$), pode-se definir uma margem de segurança capaz de fazer o valor do *timeout* sobrepor o valor amostrado do rtt com 99% de confiança (outro nível de confiança também pode ser escolhido). O intervalo de confiança da previsão, que é adicionado ao valor previsto do $\widehat{rtt}_{i+1}$, ou seja, o fator de ajuste β da margem de segurança Δ, é computado por:

$$\beta = 2,58\ \widehat{\sigma}\ \sqrt{1 + \frac{1}{n} + \frac{(rtt_i - \overline{rtt})^2}{\sum_{i=1}^{n} (rtt_i - \overline{rtt})^2}} \tag{6.12}$$

onde $2,58$ é o valor correspondente ao nível de confiança de 99% numa distribuição normal padronizada $z \sim N(0,1)$, e σ é o desvio padrão do rtt. Se for desejado um nível de confiança diferente, basta substituir $2,58$ pelo valor tabelado correspondente. Para 95%, por exemplo, o valor tabelado é $1,96$.

6.5 Relação das métricas de QoS com o *timeout*

Conforme descrito na seção 6.1, um detector de defeitos deve ser avaliado tanto em relação ao seu desempenho quanto em relação a sua precisão.

Do ponto de vista do desempenho, o melhor detector pode ser:

- o que apresenta o menor tempo médio para detecção de um defeito (T_D); ou
- o que fornece o menor limite superior (*upper bound*) do tempo para detecção de um defeito (T_D^u).

O primeiro enfoque corresponde ao desejo de aplicações que executam em sistemas onde não há requisitos temporais rígidos, enquanto o segundo enfoque corresponde ao desejo de aplicações onde há tais requisitos, tal como em sistemas de tempo real.

Em relação ao *timeout*, a comparação do T_D indica qual das implementações oferece o menor *timeout* em média, não importando se ele costuma subestimar o valor real ou não. Por outro lado, T_D^u auxilia na identificação de qual das implementações gera a maior estimativa do *timeout* em relação ao *rtt*. Para atender aos dois enfoques, a análise realizada neste capítulo considera tanto T_D quanto T_D^u.

Do ponto de vista da precisão, segundo Chen, Toueg e Aguilera (2002), o detector de defeitos mais preciso é o que oferece conjuntamente:

- o *menor* tempo médio de duração de um erro (T_M); e
- o *maior* tempo médio de recorrência ao erro (T_{MR}).

Tais métricas correspondem às duas métricas primárias definidas por Chen e Toueg para medir a precisão de um detector de defeitos. De acordo com a definição (seção 6.1), uma independe da outra, mas juntas definem se um detector é melhor do que outro em termos de precisão.

Supondo uma execução livre de falhas, como as utilizadas nas simulações, sabe-se que uma medida de T_M só ocorre quando uma falsa suspeita é descoberta, ou seja, o *timeout* deve ter expirado e posteriormente uma mensagem válida do componente monitorável deve ter sido recebida. Logo, tal métrica auxilia na verificação do quanto o valor previsto para o *timeout* é *menor* do que o que deveria ter sido adotado para evitar uma falsa suspeita.

Já a métrica de precisão T_{MR}, que descreve o tempo transcorrido entre dois erros consecutivos do detector, permite verificar a capacidade que uma dada implementação do detector tem para conseguir estabelecer um valor de *timeout* capaz de sobrepor o valor do *rtt* amostrado e evitar uma falsa suspeita.

Em síntese, em termos do *timeout*, se uma dada implementação do detector de defeitos apresentar o menor T_M e o maior T_{MR}, em relação às outras implementações, pode-se dizer que: o *timeout* utilizado é o mais próximo do que seria o *timeout ideal*, do ponto de vista da precisão, ou seja, o *timeout* utilizado confere àquela implementação a maior rapidez na recuperação de um erro, e é suficientemente grande para conferir-lhe a menor taxa de recorrência ao erro e a melhor probabilidade de uma resposta correta ($P_A = (T_{MR} - T_M)/T_{MR}$).

6.6 Análise da qualidade de serviço

Conforme discutido na seção 6.1, a análise da qualidade de serviço considera cinco métricas: uma de desempenho, o tempo médio para detecção de um defeito (T_D) e seu limite superior (T_D^u); e quatro de precisão, o tempo médio de duração de um erro (T_M), o tempo médio de recorrência ao erro (T_{MR}), a taxa de erros (λ) e a probabilidade de uma resposta precisa (P_A).

Em termos da precisão, quando uma dada implementação apresenta os melhores resultados nas duas métricas primárias T_M e T_{MR}, ela é considerada a mais precisa e dispensa a análise das métricas secundárias λ e P_A. Em alguns casos, para tornar a análise mais clara, analisa-se também a variabilidade das medidas realizadas (ANOVA e Duncan).

A metodologia geral utilizada para análise consiste em:

- avaliar as métricas de QoS para cada um dos 7 padrões de amostragem definidos na seção 5.5.1; e
- avaliar as métricas de QoS sob as amostragens reais de tempos de ida e volta coletados sobre uma WAN (vide seção 4.2.1 para mais detalhes sobre a coleta de dados).

Para cada caso, são analisadas as métricas obtidas quando a margem de segurança:

(*i*) é nula;

(*ii*) é variável em função da média exponencialmente alisada do erro de previsão; e

(*iii*) é variável em função do intervalo de confiança da previsão.

A análise das métricas de precisão, quando a margem é nula, salienta as diferenças entre os modelos preditivos puros, permitindo uma melhor comparação entre as implementações, enquanto que, com uma margem de segurança, salienta-se a influência da margem adotada.

A análise da qualidade de serviço usando padrões de amostragens fictícios serve para melhor entender o comportamento das métricas de qualidade de serviço utilizadas frente a traços de amostragem, pois cada padrão de amostragem tem um comportamento conhecido, enquanto que um traço real não. Desta forma, as relações teóricas entre qualidade de serviço e *timeout*, discutidas na seção 6.5, são abordadas sob casos controlados.

As avaliações com os padrões de amostragem e com as amostragens reais são detalhadas nas respectivas seções subseqüentes.

6.6.1 QoS em padrões de amostragens

Esta seção avalia a qualidade de serviço das implementações do detector de defeitos usando padrões de amostragem fictícios.

6.6.1.1 Metodologia

Na análise, foram utilizados 7 padrões, os mesmos utilizados na análise da precisão dos preditores e que estão definidos na seção 5.5.1. Cada padrão tem 10360 amostras, sendo que as primeiras 360 servem exclusivamente para ajustar os modelos preditivos.

Considerando todos os padrões, a análise da qualidade de serviço inicia pela avaliação do desempenho (métricas T_D e T_D^u), realiza a avaliação da precisão (métricas T_{MR}, λ e P_A) e é concluída pela avaliação conjunta da precisão e desempenho.

Em termos da margem de segurança, a análise sempre inicia com a análise sem margem de segurança, passa para uma margem de segurança fixa, onde verifica-se vários tamanhos de margem fixa, e termina com a análise sob as duas margens de segurança variáveis já especificadas (*ep* e *ic*).

6.6.1.2 Análise do desempenho

Esta seção descreve a análise do desempenho sem e com margem de segurança usando 7 padrões fictícios de amostragens.

Análise do desempenho quando a margem de segurança é nula

De acordo com a tabela 6.1, a qual apresenta os tempos médios para detecção de um defeito do tipo colapso para cada par detector/preditor (daqui para a frente indicado pelo nome do preditor), percebe-se que o T_D para um dado padrão é praticamente invariável para implementações com diferentes preditores.

O fato de, num dado padrão, praticamente todos as implementações convergirem para um mesmo T_D decorre de que, em todos os padrões, os erros tendem a se anular. Os preditores tentam corrigir seus erros a cada passo, o que faz o *timeout* estar superdimensionado em determinados instantes e subdimensionado em outros. Ao computar o valor médio do *timeout*, as dispersões para cima e para baixo tendem a se anular (vide figura 5.6) e o $\widehat{t_o} = \widehat{rtt}$. Assim, de acordo com a equação 6.2, o T_D resulta numa função do valor médio do último rtt amostrado, dado que o tempo de interrogação t_i é fixo. Especificamente, $T_D = t_i + \widehat{rtt}/2$. Logo, se a amostragem for estacionária (não apresentar tendência), o tempo médio para detecção de um defeito T_D independe do tipo de preditor adotado pelo detector.

Tabela 6.1: Tempo médio (em segundos) para detecção de um defeito do tipo colapso sem o uso de margem de segurança.

Padrão	LAST	MEAN	WINMEAN	DMA	BROWN	ARIMA	LPF
1	1,362	1,361	1,362	1,362	1,362	1,362	1,362
2	1,362	1,362	1,362	1,362	1,362	1,362	1,362
3	1,212	1,213	1,212	1,212	1,212	1,213	1,212
4	1,224	1,224	1,225	1,225	1,225	1,225	1,225
5	1,223	1,223	1,223	1,224	1,223	1,223	1,223
6	1,237	1,236	1,237	1,237	1,237	1,237	1,237
7	1,225	1,225	1,224	1,225	1,225	1,225	1,225

Para interpretar mais claramente a semelhança dos resultados, considera-se as medidas de T_D, obtidas das implementações que usam os preditores LAST e ARIMA, quando computadas sobre o padrão 2 com uma margem de segurança nula (figura 6.2). Neste caso, de acordo com a curva de previsão, claramente o preditor LAST é menos preciso do que o preditor ARIMA. Observa-se que a curva de previsão do ARIMA está exatamente em cima da curva do rtt, enquanto que a curva de previsão LAST é complementar, gerando uma previsão errada a cada nova amostra.

Considerando as características dos preditores LAST e ARIMA, tem-se que quando o instante da falha segue um rtt igual a 750ms, o preditor LAST prevê que o próximo rtt vai ser $750ms$ e faz o *timeout* ser $750ms$. Já a implementação com o preditor ARIMA, que captura exatamente o comportamento deste padrão, prevê que o *timeout* deve ser $700ms$, o valor que realmente é amostrado.

Logo, fazendo $ti = 1000ms$, se a falha ocorrer logo após o envio de uma mensagem req cujo o respectivo ack chega $750ms$ após ($rtt = 750ms$), da equação 6.1 tem-se que o t_d para a implementação que usa um preditor LAST será $1375ms$ ($t_d = 1000 + 750 - 750/2$) e para a implementação que usa um preditor ARIMA será $1325ms$ ($t_d = 1000 + 700 - 750/2$). Por outro lado, se a falha ocorrer logo após o envio de um req cujo o respectivo ack chega $700ms$ após ($rtt = 700ms$), a situação se inverte e o *timeout* resultante na implementação ARIMA é maior. Especificamente, $t_d = 1000 + 700 - 700/2 = 1350ms$ para a implementação LAST e $t_d = 1000 + 750 - 700/2 = 1400ms$ para a implementação ARIMA. Em síntese, ao computar a métrica T_D, a qual corresponde à média das medidas de tempo máximo

para detecção (equação 6.2), para uma dada execução do padrão 2, os dois preditores apresentarão os mesmos valores para T_D, mas a derivação do preditor ARIMA será maior.

A derivação dos tempos para detecção de um defeito é percebida na medição do limite superior da métrica T_D, na medição de T_D^u, conforme mostra a tabela 6.2. Nota-se que o T_D^u para a implementação ARIMA no padrão 2 é $1400ms$ e para a implementação LAST é $1375ms$. A mesma situação acontece no caso do padrão 4, onde o preditor ARIMA apresenta um $T_D^u = 1599ms$. Este resultado mostra que se o limite superior do tempo para detecção de um defeito for o mais importante para o usuário, o uso de um preditor mais preciso pode não ser a melhor escolha.

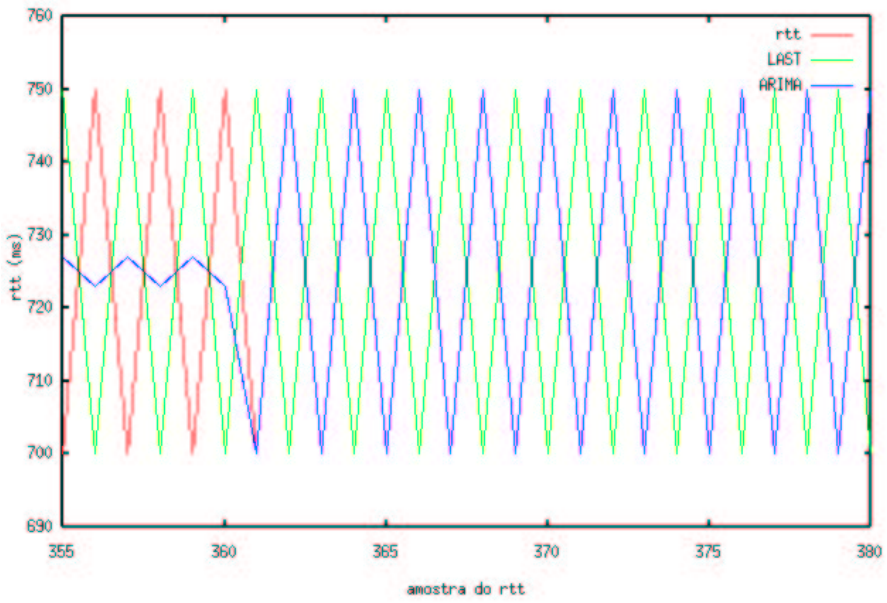

Figura 6.2: Comportamento dos detectores LAST e ARIMA num período de alternância com baixa variância (padrão 2).

Tabela 6.2: Limite superior do tempo (em segundos) para detecção de um defeito do tipo colapso sem o uso de margem de segurança.

Padrão	LAST	MEAN	WINMEAN	DMA	BROWN	ARIMA	LPF
1	1,375	1,375	1,383	1,386	1,383	1,381	1,382
2	1,375	1,374	1,375	1,375	1,372	1,400	1,373
3	1,337	1,342	1,410	1,415	1,424	1,392	1,405
4	1,350	1,349	1,350	1,350	1,316	1,599	1,333
5	1,348	1,347	1,347	1,348	1,359	1,358	1,308
6	1,375	1,374	1,380	1,383	1,559	1,379	1,514
7	1,350	1,361	1,382	1,378	1,378	1,358	1,348

Considerando o comportamento do *timeout* num padrão com alternância de nível, observa-se (figura 6.3) claramente que os preditores divergem em duas propriedades:

1. capacidade para se ajustar rapidamente a uma mudança de nível; e

2. influência dos valores passados nas novas previsões.

Por exemplo, o preditor BROWN, que considera que o rtt pode variar de acordo com uma tendência linear, ajusta-se rapidamente ao novo nível, mas realiza previsões que extrapolam o novo nível. A figura 6.3 ilustra previsões do preditor BROWN na ordem de $790ms$, quando o novo nível do rtt variava pouco em torno de $700ms$. De maneira similar, o retorno ao nível inferior, quanto as previsões ainda estão superdimensionadas, gera picos no T_D^u. A tabela 6.2 registra $T_D^u = 1559ms$ para o BROWN no padrão 6. Por outro lado, preditores baseados na média, por serem conservadores, não apresentam estes picos, fazendo com que o T_D^u seja mais baixo. Logo, para que um detector não degrade o seu T_D^u, é desejável que o preditor utilizado seja conservador em relação a mudanças que elevam as previsões, ou seja, não resultem em picos positivos no *timeout*. Um T_D^u grande pode comprometer a aplicabilidade do detector de defeitos.

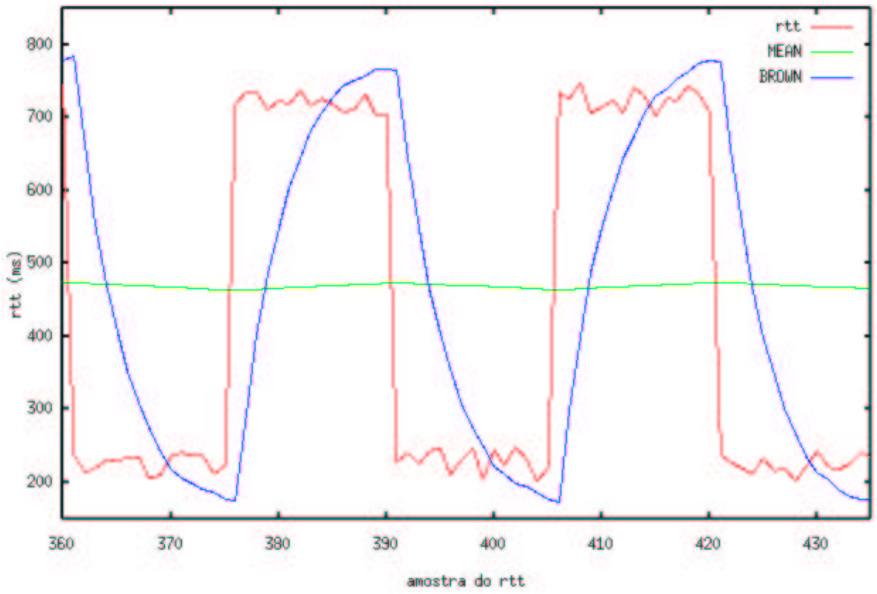

Figura 6.3: Comportamento dos detectores BROWN e MEAN num período com alternância de nível (padrão 6).

Em síntese, com margem nula, percebe-se que:

1. se o padrão de amostragem for estacionário, o T_D será uma função do rtt médio e não do tipo de preditor adotado; e

2. do ponto de vista do T_D^u, preditores conservadores são preferíveis.

Margem de segurança e o desempenho

O uso de uma margem de segurança tem impacto direto no desempenho do detector de defeitos. Para verificar esta afirmação, utiliza-se o padrão de amostragem senoidal (padrão 7) e três margens de segurança fixas: 0ms, 50ms e 100ms. A tabela 6.3 apresenta os tempos médios e os tempos limites para detecção de um defeito (T_D e T_D^u) obtidos em cada detector sob análise. Os resultados mostram que o valor da margem de segurança é diretamente refletido nos valores do T_D e T_D^u. Este comportamento é explicável pela equação 6.5, onde o *timeout* é composto pela adição

do valor previsto para o próximo rtt e da margem de segurança adotada. Como o T_D^u é o limite superior do T_D, a mesma influência ocorre nesta métrica. Logo, pode-se dizer que o tempo para detecção de um defeito, bem como o seu limite superior, refletem diretamente o valor da margem de segurança adotada. Para margens que variam em função do valor previsto, as características do preditor determinam a amplitude da margem.

Tabela 6.3: Tempos para detecção de um defeito no padrão 7 para diferentes margens de segurança fixa.

Margem	LAST	MEAN	WINMEAN	DMA	BROWN	ARIMA	LPF
	T_D						
0	1,225	1,225	1,224	1,225	1,225	1,225	1,225
50	1,275	1,275	1,274	1,275	1,275	1,275	1,275
100	1,325	1,325	1,324	1,325	1,325	1,325	1,325
	T_D^u						
0	1,350	1,361	1,382	1,378	1,378	1,358	1,348
50	1,400	1,411	1,432	1,428	1,428	1,408	1,398
100	1,450	1,461	1,482	1,478	1,478	1,458	1,448

Análise do desempenho quando a margem de segurança não é nula

Conforme discutido na seção 6.4, duas abordagens de geração da margem de segurança foram avaliadas:

1. uma margem gerada em função da média exponencialmente alisada dos erros de previsão (conforme definido na seção 6.4.2), daqui para frente chamada margem ep; e

2. uma margem gerada em função do intervalo de confiança da previsão, com nível de confiança de 99% (conforme definido na seção 6.4.3), daqui para frente chamada margem ic.

Sabendo que o valor da margem de segurança influencia diretamente no tempo para detecção de um defeito, a análise com margem de segurança é dirigida para mostrar que a escolha desta margem depende do tipo de preditor adotado. Salienta-se que embora uma margem menor possa diminuir a precisão do detector de defeitos, em termos do desempenho ela é preferível.

A tabela 6.4 apresenta os tempos médios para detecção de um defeito, bem como seu limite superior, para todas as implementações em cada padrão de amostragem, quando margens de segurança variáveis são utilizadas.

Da tabela, tem-se que, para amostragens aleatórias tais como as representadas pelo padrão 1, tanto T_D como T_D^u são menores em todas as implementações, quando é aplicada a margem ic. Considerando o T_D, se a variação for baixa (padrão 1), a margem ic é em média 23% menor do que a margem ep e, se a variação for alta (padrão 3), a margem ic é em média 28% menor. Salienta-se que as porcentagens foram computadas em função do tamanho das margens em relação ao desempenho médio obtido com uma margem zero (tabelas 6.1 e 6.2). Ao considerar o T_D^u, a diferença é ainda maior: a margem ic é em média 43% e 42% menor para os padrões 1 e 3, respectivamente, se desconsiderado o caso da implementação LAST onde, para ambos os padrões ela é 67% menor do que a margem ep. Estes resultados indicam que o uso da margem ic oferece ao detector um desempenho melhor se o padrão de

Tabela 6.4: Tempos para detecção de um defeito para margens de segurança variáveis.

Padrão	Margem	LAST	MEAN	WINMEAN	DMA	BROWN	ARIMA	LPF
		T_D						
1	ep	1,430	1,412	1,413	1,413	1,415	1,413	1,414
	ic	1,401	1,399	1,401	1,401	1,401	1,401	1,401
2	ep	1,561	1,463	1,462	1,462	1,474	1,362	1,469
	ic	1,429	1,429	1,428	1,428	1,429	1,429	1,429
3	ep	1,880	1,712	1,717	1,722	1,740	1,713	1,723
	ic	1,596	1,593	1,600	1,599	1,597	1,597	1,598
4	ep	3,223	2,226	2,225	2,225	2,361	1,229	2,292
	ic	1,902	1,902	1,892	1,892	1,903	1,903	1,903
5	ep	1,354	1,720	1,720	1,721	1,663	1,307	1,683
	ic	1,613	1,607	1,606	1,607	1,613	1,613	1,613
6	ep	1,433	2,236	2,237	2,237	1,823	1,493	2,051
	ic	1,917	1,910	1,924	1,923	1,917	1,917	1,917
7	ep	1,256	1,861	1,685	1,529	1,314	1,245	1,463
	ic	1,415	1,413	1,415	1,416	1,415	1,415	1,416
		T_D^u						
1	ep	1,521	1,456	1,459	1,463	1,465	1,462	1,461
	ic	1,423	1,422	1,425	1,428	1,429	1,431	1,427
2	ep	1,574	1,475	1,475	1,475	1,484	1,480	1,480
	ic	1,442	1,441	1,441	1,441	1,439	1,489	1,440
3	ep	2,755	2,203	2,257	2,265	2,255	2,205	2,234
	ic	1,804	1,884	1,855	1,873	1,894	1,928	1,870
4	ep	3,349	2,350	2,350	2,350	2,452	2,400	2,400
	ic	2,028	2,027	2,017	2,017	1,994	2,506	2,011
5	ep	1,479	2,052	2,051	2,055	1,802	1,611	1,943
	ic	1,745	1,752	1,730	1,731	1,755	1,757	1,698
6	ep	2,029	2,439	2,442	2,446	2,463	2,008	2,525
	ic	2,073	2,179	2,163	2,171	2,260	2,066	2,222
7	ep	1,368	2,385	2,042	1,781	1,494	1,397	1,634
	ic	1,604	1,703	1,627	1,563	1,622	1,614	1,543

amostragem for aleatório, pois tanto o T_D como o T_D^u são os menores em todas as implementações.

Para padrões com alternância regular (padrões 2 e 4), em todas as implementações, exceto a baseada no preditor ARIMA, tanto o T_D como o T_D^u são menores com o uso da margem *ic*, pois esta margem é menor do que a margem computada em função do erro de previsão. Por outro lado, na implementação baseada no preditor ARIMA, a situação se inverte, pois a melhor margem é a *ep*. Este resultado decorre do fato de que o preditor ARIMA consegue capturar o comportamento dos padrões 2 e 4, o que faz o erro médio cair para zero. Logo, a margem baseada no erro médio também cai para zero, enquanto que a margem baseada no intervalo de confiança da previsão permanece relativamente estável, num valor diferente de zero. A figura 6.4 ilustra o comportamento do *timeout* em cada implementação, quando aplicada cada uma das margens variáveis.

Para um padrão de amostragem com variação de tendência a cada 15 amostras, tal como o representado pelo padrão 5, percebe-se na tabela 6.4 que a margem *ic* resulta em melhores desempenhos, tanto em termos do T_D com em termos do T_D^u, nas implementações MEAN, WINMEAN, DMA, BROWN e LPF. Por outro lado, nas implementações ARIMA e LAST, a margem *ep* é mais adequada. Estes resultados decorrem de que, em média, o erro nas implementações ARIMA e LAST é mais baixo do que nas outras implementações.

À parte dos valores médios, é interessante salientar que a característica da curva do *timeout* muda de acordo com a margem. As figuras 6.5, 6.6 e 6.7 mostram o comportamento do *timeout* para ambas as margens nos padrões 5, 6 e 7, respectivamente. Das figuras, nota-se que, com a margem *ep*, qualquer variação (para cima

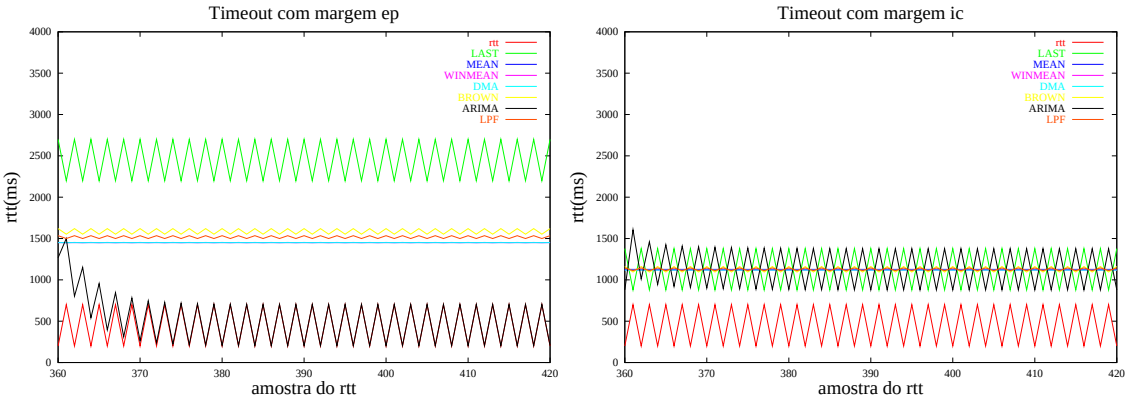

Figura 6.4: Comportamento do *timeout* no padrão 4.

ou para baixo) na curva de amostragem reflete-se em picos na curva do *timeout*. Por isto, preditores de baixa precisão, tais como os baseados na média, resultam nos piores desempenhos com esta margem. Entretanto, se o preditor consegue capturar o comportamento da curva de amostragem, o erro será pequeno e a margem *ep* torna-se a mais adequada.

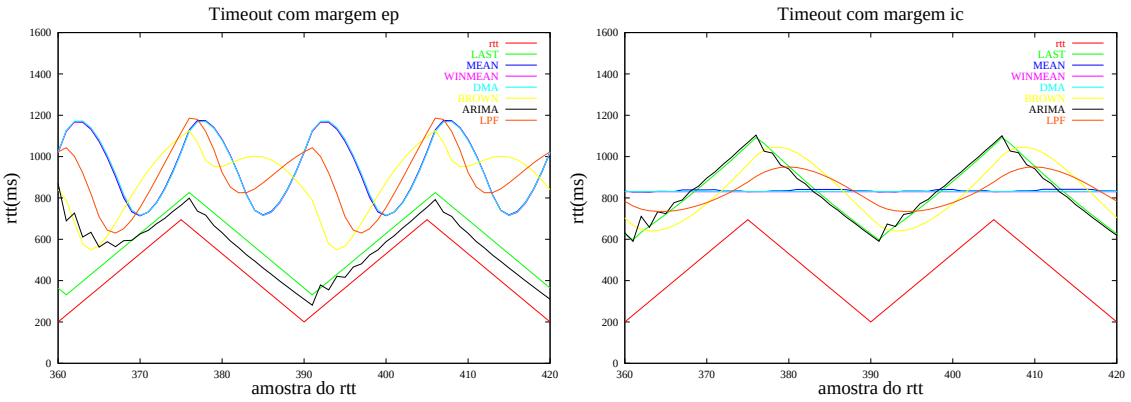

Figura 6.5: Comportamento do *timeout* no padrão 5.

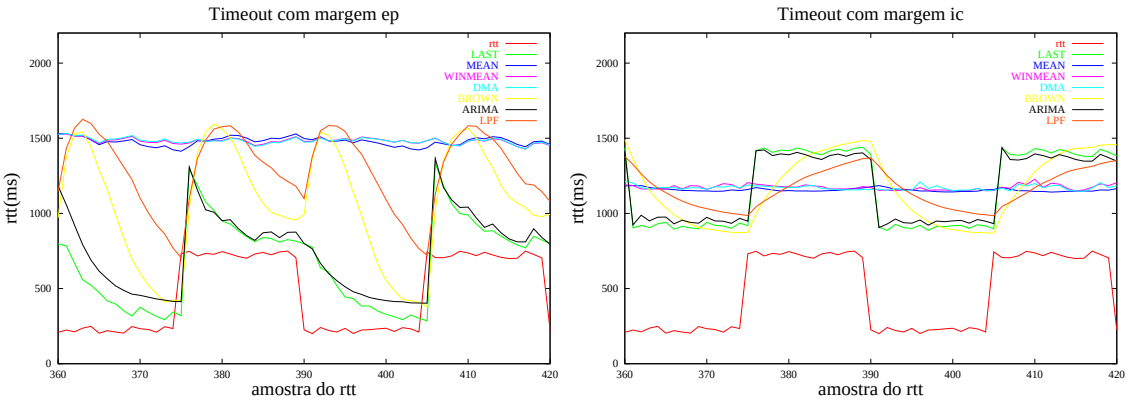

Figura 6.6: Comportamento do *timeout* no padrão 6.

Em síntese, da análise do desempenho com margem de segurança percebe-se que:

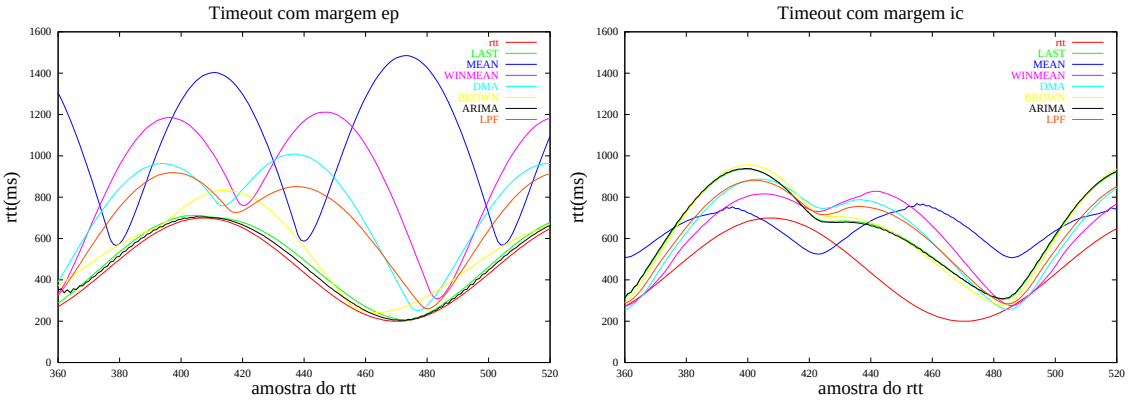

Figura 6.7: Comportamento do *timeout* no padrão 7.

1. quanto maior a margem de segurança, menor o desempenho;
2. em termos do desempenho do detector, a escolha do melhor preditor depende da margem adotada;
3. são preferíveis as combinações de preditores precisos com margens baseadas no erro de previsão, e preditores baseados na média (menos precisos) com margens que independem do erro (mais estáveis).

6.6.1.3 Análise da precisão

Esta seção analisa a precisão dos detectores por tipo de padrão de amostragem. A separação por tipo auxilia na interpretação das relações entre o comportamento do *rtt*, o *timeout* e as métricas de precisão.

Amostragens com alternância adjacente regular

Para padrões de amostragem com alternância adjacente regular (padrões 2 e 4) e margem de segurança nula, a precisão (métricas T_M, T_{MR}, λ e P_A) de cada implementação do detector de defeitos é listada na tabela 6.5. Dos resultados tabelados, percebe-se que:

(i) com um detector que determina um *timeout* de forma integralmente correta, tal com o preditor ARIMA, não há suspeitas incorretas e a precisão é ótima ($T_M = 0$ e $T_{MR} = 10000$ segundos);

(ii) excetuando o ARIMA, nenhum detector consegue estabelecer um *timeout* capaz de sobrepor o pico superior da sanfona dos padrões, pois erram uma vez a cada duas mensagens enviadas ($\lambda = 0,5$ para um período de interrogação de 1 segundo); e

(iii) a métrica de precisão T_M reflete a amplitude do erro de previsão que gera a expiração do *timeout*, ou seja, o quão cedo o *timeout* expirou. O erro do detector LAST é a largura da sanfona (50ms para o padrão 2 e 500ms para o padrão 4) e define seu T_M. O erro dos detectores baseados na média é praticamente a metade da amplitude da sanfona e define seus T_M.

Considerando a aplicação de uma margem de segurança variável (margens *ep* e *ic*), obteve-se para todas as implementações $T_M = 0$s, $T_{MR} = 10000$s, $\lambda = 0$ por

Tabela 6.5: Precisão dos detectores para amostragens com alternância adjacente regular quando a margem de segurança é nula.

Padrão	LAST	MEAN	WINMEAN	DMA	BROWN	ARIMA	LPF
				T_M			
2	0,050	0,026	0,025	0,025	0,028	0,0	0,027
4	0,500	0,251	0,250	0,250	0,284	0,0	0,267
				T_{MR}			
2	2,0	2,0	2,0	2,0	2,0	10000,0	2,0
4	2,0	2,0	2,0	2,0	2,0	10000,0	2,0
				λ			
2	0,500	0,500	0,500	0,500	0,500	0,0	0,500
4	0,500	0,500	0,500	0,500	0,500	0,0	0,500
				P_A			
2	0,974	0,986	0,987	0,987	0,985	1,0	0,986
4	0,750	0,874	0,875	0,875	0,857	1,0	0,866

segundo e $P_A = 1$. Tal resultado mostra que: ambas margens resultam num *timeout* suficientemente grande para fazer com que todas as implementações do detector de defeitos não cometam nenhum erro, pois o valor do *timeout* é maior do que o maior pico do *rtt* (vide curva do *timeout* para o padrão 4 na figura 6.4).

A partir da existência de uma margem de segurança fixa, recomputa-se as métricas de precisão T_M e T_{MR} para diversos tamanhos da margem fixa. As figuras 6.8 e 6.9 apresentam as novas precisões para os padrões 2 e 4, respectivamente, conforme a margem fixa muda. Os resultados mostram que o tempo médio de uma suspeita incorreta (T_M) depende do tamanho da margem de segurança. Nas figuras, um aumento linear da margem de segurança faz o T_M diminuir também de maneira linear, pois as falsas suspeitas são sempre correspondentes aos picos positivos da sanfona. Quando $T_M = 0$ a margem de segurança é suficiente para *sempre* compensar a previsão de um *timeout* subdimensionado. Entretanto, na prática, a escolha da margem fixa ideal não é possível, pois o atraso de comunicação não é determinístico.

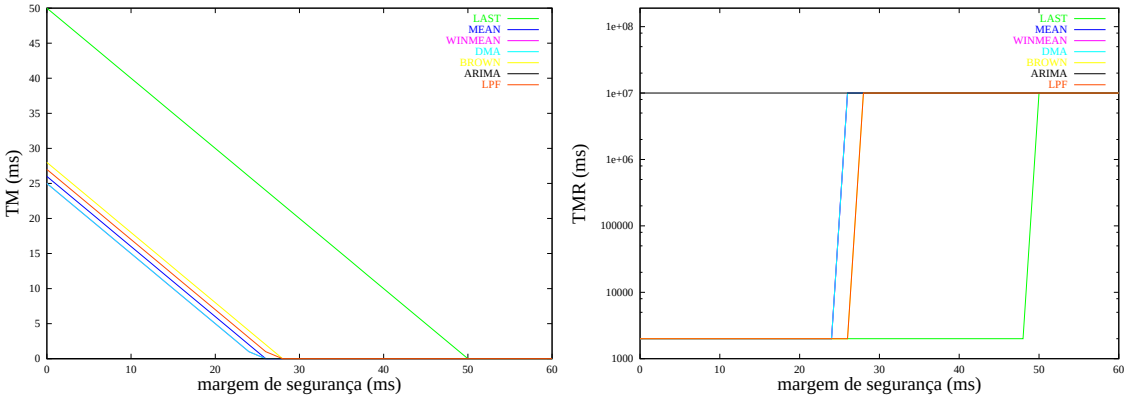

Figura 6.8: Métricas T_M e T_{MR} para o padrão 2 com diferentes margens de segurança.

Em síntese, a análise de amostragens com alternância adjacente regular mostra que:

1. preditores 100% precisos fazem um detector ser 100% preciso; e

2. a métrica de precisão T_M reflete o quão cedo, em média, o *timeout* expirou.

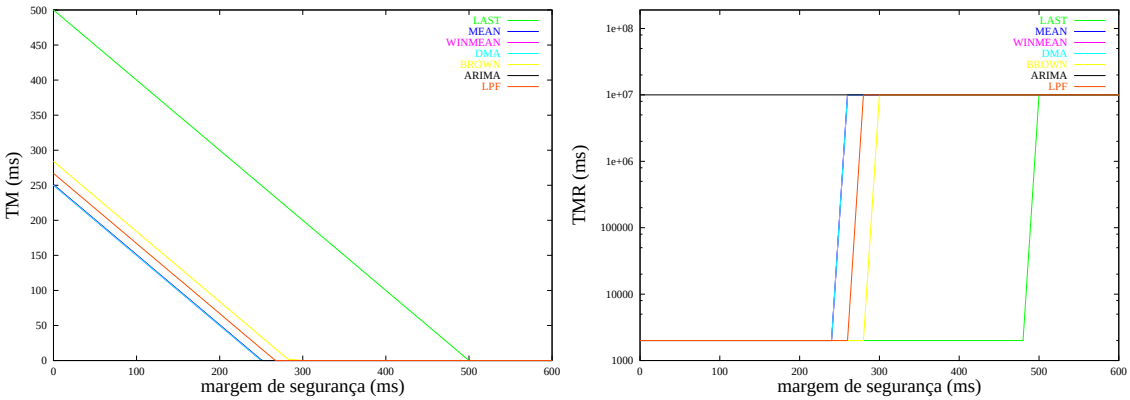

Figura 6.9: Métricas T_M e T_{MR} para o padrão 4 com diferentes margens de segurança.

Amostragens que resultam num sinal aleatório

As medidas de precisão de cada detector, considerando as margens nula, *ep* e *ic*, para as amostragens aleatórias dos padrões 1 e 3, são listadas nas tabelas 6.6 e 6.7, respectivamente. Os resultados mostram que a variabilidade entre amostras adjacentes dos padrões aleatórios diminuem a precisão do LAST em termos da duração de uma falsa suspeita (T_M). Com ou sem margem, o maior T_M é sempre do LAST ($167ms$ com margem zero). Com margem *ic*, excetuando o LAST, todos alcançam a melhor precisão em termos da taxa de erros e da probabilidade de uma resposta precisa ($\lambda \leq 0,001$ e $P_A = 1$).

Mesmo as margens de segurança *ep* e *ic* sendo eficientes em termos de precisão, para preditores de baixa qualidade, o maior custo da sua eficiência é percebido no desempenho do detector, conforme discutido na seção 6.6.1.2.

Tabela 6.6: Precisão dos detectores para a amostragem aleatória do padrão 1.

Margem	LAST	MEAN	WINMEAN	DMA	BROWN	ARIMA	LPF
				T_M			
nula	0,017	0,013	0,013	0,013	0,013	0,013	0,013
ep	0,006	0,002	0,002	0,004	0,006	0,002	0,004
ic	0,004	0,000	0,000	0,001	0,001	0,000	0,001
				T_{MR}			
nula	2,042	1,988	2,009	2,051	2,042	2,048	2,045
ep	272,111	1103,999	781,272	781,272	345,370	1174,999	803,999
ic	40,032	10000,0	10000,0	10000,0	675,091	10000,0	10000,0
				λ			
nula	0,490	0,503	0,498	0,488	0,490	0,488	0,489
ep	0,004	0,001	0,001	0,001	0,003	0,001	0,001
ic	0,025	0,000	0,000	0,000	0,001	0,000	0,000
				P_A			
nula	0,992	0,993	0,994	0,994	0,994	0,994	0,994
ep	1,000	1,000	1,000	1,000	1,000	1,000	1,000
ic	1,000	1,000	1,000	1,000	1,000	1,000	1,000

Adotando diferentes tamanhos de margem fixa, conforme ilustram as figuras 6.10 e 6.11, percebe-se que, para um sinal aleatório o melhor preditor é o que segue a média da população, pois a amplitude máxima de variabilidade dos padrões é fixada e conseqüentemente detectores que seguem a média sobrepõem toda a curva com uma margem um pouco maior do que 50% do intervalo de variabilidade da amostragem. As implementações MEAN e ARIMA são as que apresentam os melhores resultados em ambos padrões, pois são as que necessitam da menor margem de segurança para evitar todas as suspeitas. A MEAN necessita de uma margem maior ou igual a 26ms

Tabela 6.7: Precisão dos detectores para a amostragem aleatória do padrão 3.

Margem	LAST	MEAN	WINMEAN	DMA	BROWN	ARIMA	LPF
				T_M			
nula	0,167	0,124	0,126	0,127	0,132	0,125	0,128
ep	0,044	0,012	0,020	0,028	0,036	0,011	0,043
ic	0,044	0,000	0,000	0,026	0,015	0,000	0,010
				T_{MR}			
nula	1,999	2,004	1,997	1,999	2,008	2,006	2,005
ep	385,911	1092,008	1092,006	1075,261	443,612	1092,012	842,807
ic	36,611	10000,0	10000,0	10000,0	2543,680	10000,0	10000,0
				λ			
nula	0,500	0,499	0,501	0,500	0,498	0,499	0,499
ep	0,003	0,001	0,001	0,001	0,002	0,001	0,001
ic	0,027	0,000	0,000	0,000	0,000	0,000	0,000
				P_A			
nula	0,916	0,938	0,937	0,936	0,934	0,938	0,936
ep	1,000	1,000	1,000	1,000	1,000	1,000	1,000
ic	0,999	1,000	1,000	1,000	1,000	1,000	1,000

no padrão 1 e 260ms no padrão 3, enquanto que a ARIMA necessita de 30ms no padrão 1 e 300ms no padrão 3. A boa precisão do ARIMA indica que o preditor se ajusta adequadamente aos dados, conseguindo obter resultados similares ao MEAN.

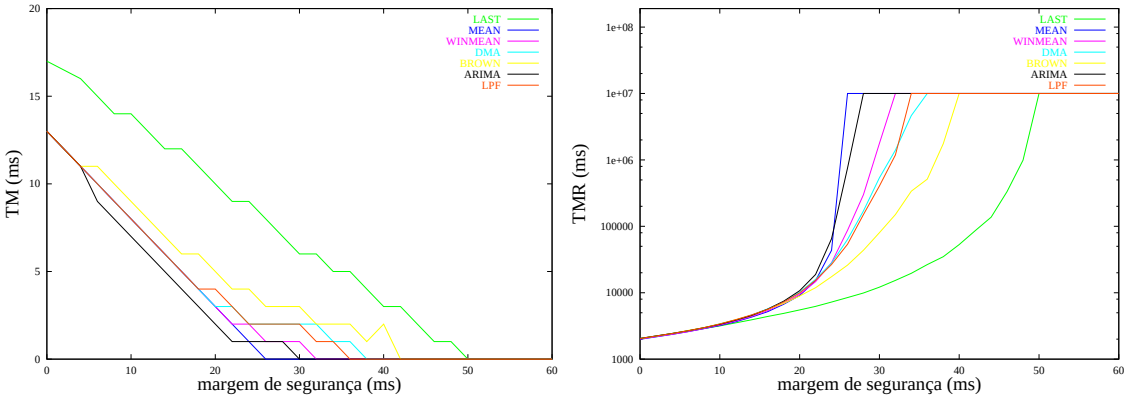

Figura 6.10: Métricas T_M e T_{MR} para o padrão 1 com diferentes margens de segurança.

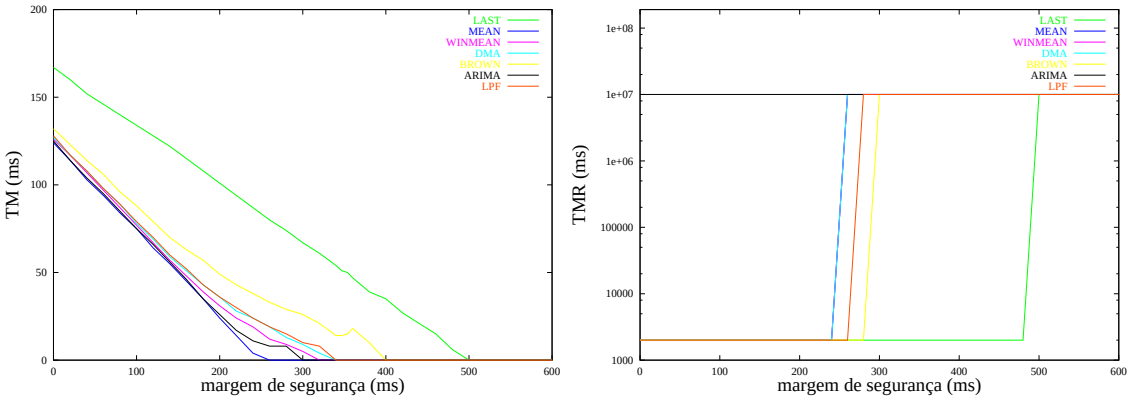

Figura 6.11: Métricas T_M e T_{MR} para o padrão 3 com diferentes margens de segurança.

Em síntese, se a amostragem for um sinal aleatório, tem-se que:

1. a variabilidade entre amostras adjacentes dos padrões aleatórios dimini a precisão do LAST em termos da duração de uma falsa suspeita; e

2. o detector mais preciso é o que ajusta o seu *timeout* de acordo com a média da população.

Amostragem com variação de tendência a cada 15 amostras

Considerando uma amostragem com variação de tendência linear a cada 15 amostras (padrão 5) e uma margem de segurança nula, a precisão dos detectores é apresentada na tabela 6.8. Dos resultados percebe-se que a variação relativa do T_{MR} entre as sete implementações do detector de defeitos não é significativa e que de acordo com o T_M, as implementações ARIMA e LAST são as mais precisas.

Sem margem de segurança, o T_{MR} é similar para todos devido à regularidade do padrão. Detectores que seguem à média subestimam o *timeout* sempre que o *rtt* é superior a média (50% das amostras) e detectores que conseguem se ajustar mais rapidamente subestimam o *timeout* quando a tendência é crescente (50% das amostras). Por outro lado, o ARIMA e o LAST capturam o comportamento do *rtt*, resultando nos menores T_M ($T_M = 0,021$s para a ARIMA e $TM = 0,033$s para a LAST contra $T_M > 0,110$s para as demais).

Tabela 6.8: Precisão dos detectores para o padrão 5 quando a margem de segurança é nula.

	LAST	MEAN	WINMEAN	DMA	BROWN	ARIMA	LPF
T_M	0,033	0,124	0,124	0,123	0,110	0,021	0,114
T_{MR}	1,998	2,0	2,0	2,0	1,998	1,999	1,999
λ	0,500	0,500	0,500	0,500	0,500	0,500	0,500
P_A	0,983	0,937	0,937	0,938	0,944	0,989	0,942

Considerando diferentes margens fixas, conforme ilustram os gráficos da figura 6.12, observa-se que: primeiro, as implementações baseadas na média não são adequadas para padrões de amostragens com variações de tendência, pois elas necessitam da maior margem de segurança para conseguir sobrepor todos os casos de falsas suspeitas (as implementações MEAN, WINMEAN e DMA apresentam exatamente as mesmas medidas para T_M e T_{MR} conforme a margem fixa é trocada); segundo, as implementações baseadas na técnica de alisamento exponencial são melhores do que as implementações baseadas na média, mas necessitam de uma margem de segurança de cerca de 3 vezes a margem necessária para as implementações ARIMA e LAST; e terceiro, as duas implementações mais precisas (ARIMA e LAST), embora necessitem de uma pequena margem ($\sim$50ms) para se tornarem 100% precisas, apresentam um cruzamento em suas curvas.

O cruzamento das curvas das implementações ARIMA e LAST deve-se a uma instabilidade na curva de previsão do preditor ARIMA, quando ocorre a troca de uma tendência decrescente para uma tendência crescente. A instabilidade também ocorre na troca de uma tendência crescente para uma decrescente, mas, neste último caso, a instabilidade fica acima da curva de amostragem e não interfere na precisão do detector, só no desempenho. Na troca de uma tendência decrescente para crescente, a instabilidade faz com que, em alguns pontos, o *timeout* utilizado na implementação ARIMA seja menor do que o utilizado na implementação LAST, aumentando a probabilidade de falsas suspeitas nestes pontos. Uma suspeita nestes pontos faz

com que a medida parcial do T_M da implementação ARIMA seja maior do que o da implementação LAST. Como estas falsas suspeitas são as últimas a serem eliminadas, conforme aumenta a margem de segurança, elas geram a inversão nas curvas do T_M e T_{MR}.

Dos gráficos da figura 6.12, também observa-se que, excetuando as implementações ARIMA e LAST, as implementações que usam um preditor com melhor capacidade de se ajustar rapidamente a novas tendências (o preditor BROWN se ajusta mais rápido do que o preditor LPF, que se ajusta mais rápido do que os preditores baseados na média) produzem um T_M menor e um T_{MR} maior para a mesma magnitude da margem de segurança, indicando que esta característica é importante para obter uma melhor precisão num detector de defeitos.

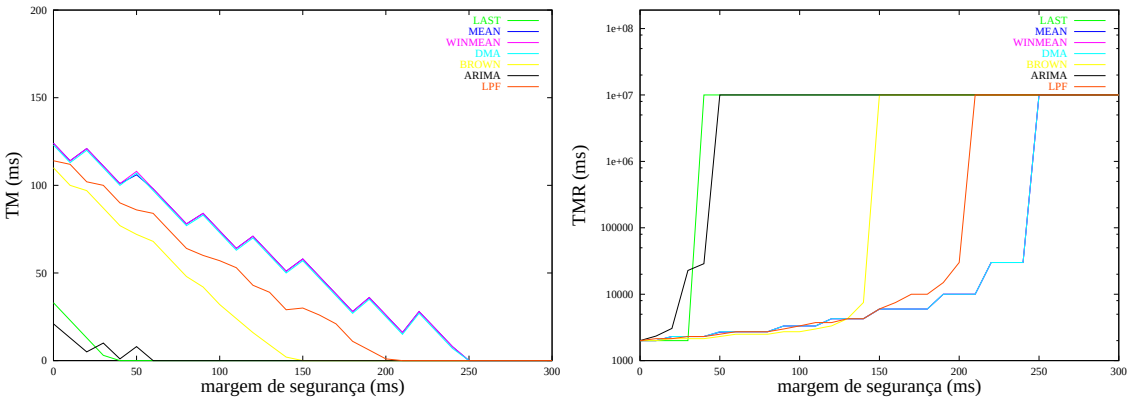

Figura 6.12: Métricas T_M e T_{MR} para o padrão 5 com diferentes margens de segurança.

Em síntese, para padrões com variação de tendência a cada 15 amostras, onde o *rtt* ora cresce de maneira linear ora decresce da mesma maneira, tem-se que:

1. a precisão é melhor nos detectores que conseguem ajustar rapidamente seu *timeout* à mudança de tendência e conseguem manter o crescimento, ou diminuição, linear do *rtt*;

2. o relacionamento entre amostras de *rtt* adjacentes é capturado na modelagem ARIMA, resultando em uma boa precisão para o detector ARIMA com a menor margem de segurança; e

3. a regularidade na ascendência e descendência faz o detector LAST ter uma boa precisão com uma pequena margem de segurança.

Amostragem com variação brusca de nível a cada 15 amostras

A tabela 6.9 apresenta a precisão dos detectores, sob margem nula, *ep* e *ic*, para um padrão de amostragem com variação brusca de nível (variação de 500ms) a cada 15 amostras, onde num dado nível a amostragem varia de maneira aleatória, mas com amplitude máxima fixada em 50ms.

Sem margem de segurança, as métricas de precisão indicam que: primeiro, em termos do T_M, as implementações LAST e ARIMA fornecem resultados bem mais precisos do que as outras implementações (o T_M das implementações LAST e

ARIMA é menos de 50% do T_M para as outras implementações); e, segundo, que em termos da métrica T_{MR}, a implementação LAST é novamente a mais precisa, pois apresenta os maiores T_{MR}s em todos os três tamanhos de execução, seguida da implementação ARIMA, que apresenta o segundo maior T_{MR} nos três tamanhos de execução, mas com diferenças pouco significativas quando comparado com a implementação BROWN.

O fato da implementação LAST ser mais precisa do que a ARIMA parece contradizer os resultados da análise para os padrões 1 e 3, onde indicou-se que o LAST é a pior abordagem se o sinal for aleatório. Entretanto, ao analisar a curva de previsão dos preditores LAST e ARIMA (figura 5.9 do capítulo anterior), percebe-se que o preditor ARIMA sofre influência da média populacional pois, embora o preditor responda rapidamente à troca de nível, ele tende a se aproximar da média (realiza previsões levemente acima do nível inferior e levemente abaixo do nível superior do padrão). Já a previsão do LAST não sofre tal influência; logo, a baixa variância no nível (50ms) faz suas previsões serem mais próximas dos valores amostrados do que as do ARIMA. A alteração no nível é 10 vezes maior (500ms). Como conseqüência, as sucessivas subestimações do *timeout* aumentam o T_M e diminuem o T_{MR} da implementação ARIMA.

Considerando-se os casos de margens variáveis, nota-se que a margem *ic* faz com que todas as implementações tenham precisão ótima. Por outro lado, a margem *ep* só consegue precisão ótima nas implementações baseadas na média. Tal resultado reflete o fato de que o grande erro de previsão dos preditores baseados na média geram margens grandes, enquanto que, para preditores mais precisos, ao realizar a troca de nível, a margem não é suficiente para sobrepor a diferença de nível. Por outro lado, a curva do *timeout* com o padrão 6 (figura 6.6) mostra que preditores precisos associados a uma margem *ep*, embora gerem alguns erros, têm uma boa relação *custo* `x` *benefício*, pois a combinação faz o detector errar somente quando as mudanças ocorrem de nível baixo para alto, mantendo um *timeout* bem menor do que com a margem *ic* neste padrão.

Tabela 6.9: Precisão dos detectores para o padrão 6.

Margem	LAST	MEAN	WINMEAN	DMA	BROWN	ARIMA	LPF
				T_M			
nula	0,050	0,251	0,250	0,250	0,146	0,064	0,203
ep	0,423	0,0	0,0	0,0	0,343	0,314	0,019
ic	0,0	0,0	0,0	0,0	0,0	0,0	0,0
				T_{MR}			
nula	2,046	1,997	1,997	1,997	2,004	2,005	1,997
ep	29,909	10000,0	10000,0	10000,0	29,999	29,999	50,357
ic	10000,0	10000,0	10000,0	10000,0	10000,0	10000,0	10000,0
				λ			
nula	0,489	0,501	0,501	0,501	0,499	0,499	0,501
ep	0,033	0,000	0,000	0,000	0,033	0,033	0,020
ic	0,000	0,000	0,000	0,000	0,000	0,000	0,000
				P_A			
nula	0,976	0,874	0,875	0,875	0,927	0,968	0,898
ep	0,986	1,000	1,000	1,000	0,989	0,990	1,000
ic	1,000	1,000	1,000	1,000	1,000	1,000	1,000

Considerando diversos tamanhos de margem de segurança fixa, as métricas T_M e T_{MR} para o padrão 6 comportam-se como ilustrado na figura 6.13. Dos gráficos, constata-se que os preditores com menor erro quadrático médio (LAST e ARIMA) partem de baixo T_M e de um baixo T_{MR}, mas, com uma pequena margem (em torno de 50ms), o T_M sobe para algo em torno de 400ms e o T_{MR} sobe para aproximada-

mente 30 segundos. Este cenário mostra que o T_M e o T_{MR} podem sofrer grandes variações de acordo com as medidas que eles representam, ou seja, após uma pequena margem, o LAST e o ARIMA erram pouco e, quando erram, é na troca de um nível baixo para um alto, ponto onde eles subestimam o *timeout* com um valor próximo da diferença de nível. Já detectores baseados na média, com uma margem de segurança de aproximadamente 50% do desvio entre níveis (50%=250ms no padrão 6) alcançam a precisão ótima (a curva do DMA sobrepõe as curvas MEAN e WINMEAN em ambos gráficos). Estes resultados mostram que, se o preditor for preciso, pequenas margens podem ser suficientes para obter uma boa precisão (T_{MR} cresce muito mais do que o T_M).

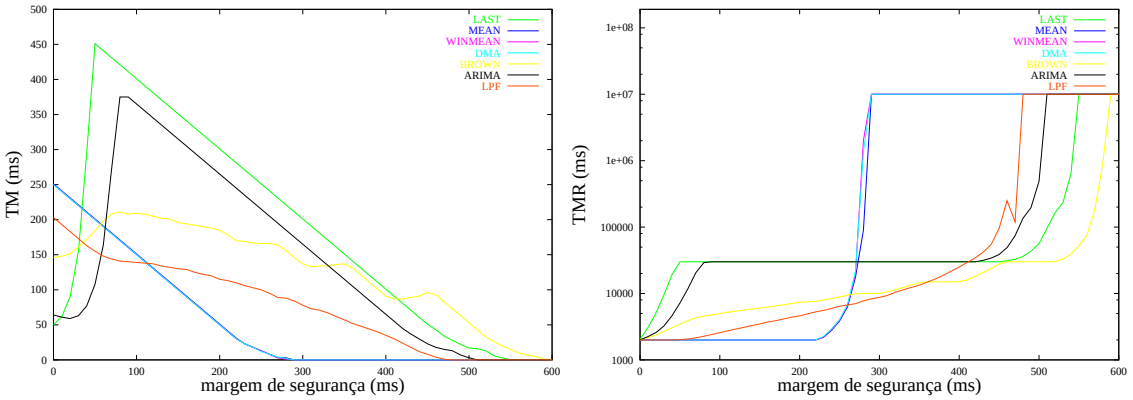

Figura 6.13: Métricas T_M e T_{MR} para o padrão 6 com diferentes margens de segurança.

Em síntese, a análise da precisão com um padrão com variação brusca de nível mostra que:

1. preditores precisos podem contribuir para obtenção de uma boa qualidade de serviço (precisão e desempenho) com margens pequenas; e

2. margens baseadas no erro de previsão, se usadas com preditores precisos, podem resultar numa boa relação *custo* `x` *benefício* nos casos de alterações freqüentes de níveis.

Amostragem com variação senoidal com período de 120 amostras

Para um padrão de amostragem senoidal, a precisão dos detectores, sem margem e com margens *ep* e *ic*, é apresentada na tabela 6.10.

Com margem nula, observa-se o preditor ARIMA confere ao detector de defeitos a melhor precisão, pois resulta no menor T_M ($T_M = 5$ms) e no maior T_{MR} ($T_{MR} = 2100$ms).

No caso de uma margem variável, a margem *ep* é a que apresenta o melhor resultado pois, com ela, todas as implementações, exceto a ARIMA, resultam num $T_M = 0$s e num $T_{MR} = 10000$s. Como conseqüência, a taxa de erros é zero e a probabilidade de uma resposta precisa é 1 em todas as implementações. Na tabela, embora o T_M e o T_{MR} do ARIMA com margem *ep* sejam diferentes de 0,0 e 1000,0, respectivamente, foi inserido $\lambda = 0,000$ e $P_A = 1,000$ por motivos de arredondamento (os valores absolutos são $\lambda = 0,000489355$ e $P_A = 0,999999511$). O baixo

erro do preditor ARIMA resulta numa pequena margem *ep* e, conseqüentemente, o *timeout* resultante não consegue sobrepor pequenas variações geradas nos momentos de reajuste do preditor ARIMA. A figura 6.14 ilustra uma pequena oscilação na curva do *timeout* gerado pelo preditor ARIMA, onde em alguns instantes o *timeout* torna-se menor do que o *rtt* amostrado. Por outro lado, neste padrão, a margem *ic* é melhor para a implementação ARIMA, pois faz com que este detector não gere nenhuma falsa suspeita, mas não é suficiente para as implementações que utilizam preditores baseados na média amostral.

Tabela 6.10: Precisão dos detectores para o padrão 7.

Margem	LAST	MEAN	WINMEAN	DMA	BROWN	ARIMA	LPF
				T_M			
nula	0,008	0,158	0,116	0,076	0,022	0,005	0,059
ep	0,0	0,0	0,0	0,0	0,0	0,001	0,0
ic	0,0	0,080	0,004	0,011	0,0	0,0	0,0
				T_{MR}			
nula	2,074	1,997	1,987	1,991	2,011	2,100	1,991
ep	10000,0	10000,0	10000,0	10000,0	10000,0	2043,506	10000,0
ic	10000,0	4,688	20,824	16,835	10000,0	10000,0	10000,0
				λ			
nula	0,482	0,501	0,503	0,502	0,497	0,476	0,502
ep	0,000	0,000	0,000	0,000	0,000	0,000	0,000
ic	0,000	0,213	0,048	0,059	0,000	0,000	0,000
				P_A			
nula	0,996	0,921	0,942	0,962	0,989	0,998	0,970
ep	1,000	1,000	1,000	1,000	1,000	1,000	1,000
ic	1,000	0,983	1,000	0,999	1,000	1,000	1,000

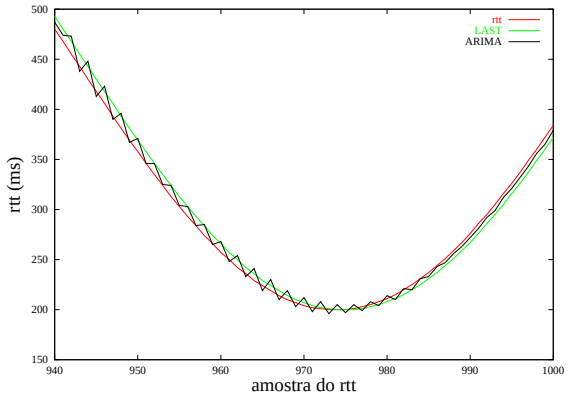

Figura 6.14: Curva de previsão para o ARIMA e LAST no padrão 7.

Ao considerar o T_M e T_{MR} sob diversas margens de segurança fixas, conforme ilustrado na figura 6.15, observa-se que as implementações que necessitam das menores margens de segurança para eliminar todas as falsas suspeitas são as implementações LAST e ARIMA, ou seja, as que utilizam os preditores mais precisos para este padrão. Os gráficos da figura 6.15 também mostram que, para padrões amostrais com variação senoidal com período de 120 amostras, os preditores baseados na média não conseguem acompanhar as variações e acabam necessitando de margens de segurança grandes. Já os preditores baseados em mecanismos de alisamento exponencial necessitam de margens menores do que os baseados na média, mas há uma diferença significativa entre eles. A implementação BROWN necessita de uma margem 50% menor do que a implementação LPF. O BROWN reajusta-se mais rapidamente às alterações.

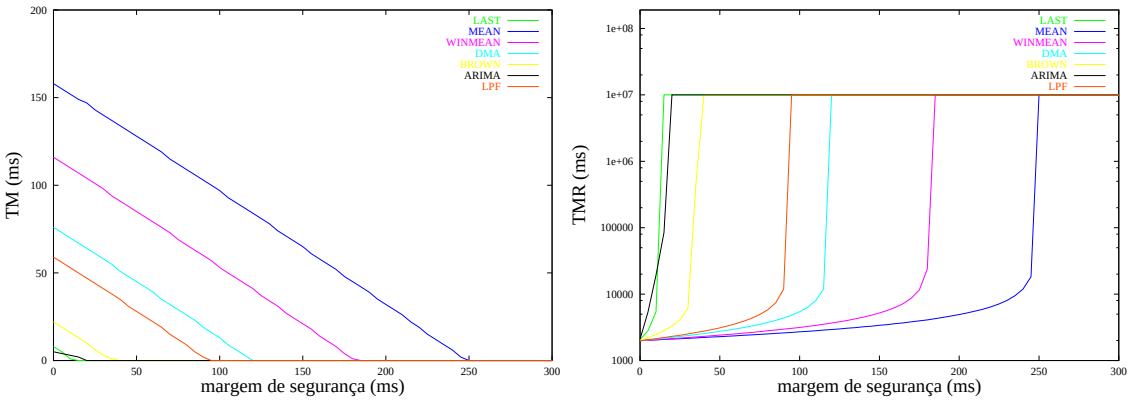

Figura 6.15: Métricas T_M e T_{MR} para o padrão 7 com diferentes margens de segurança fixas.

Em síntese, a análise de um padrão senoidal mostra que com margem variável, a escolha da melhor abordagem de previsão depende da escolha da margem. Margens baseados no erro de previsão favorecem detectores baseados na média, enquanto que margens baseadas no intervalo de confiança da previsão favorecem preditores mais precisos.

6.6.1.4 Síntese da análise de QoS com padrões de amostragem

A análise da qualidade de serviço dos detectores, utilizando padrões de amostragens com comportamento conhecido, mostra que:

- quanto mais preciso for o preditor, menor a margem de segurança necessária para obter o nível de qualidade de serviço desejado;
- a escolha do melhor preditor depende da característica da margem de segurança escolhida;
- com margens baseadas no erro de previsão (ep), preditores precisos produzem margens menores, favorecendo o desempenho (T_D) do detector, mas preditores baseados na média (menos precisos) favorecem a precisão dos detectores, pois produzem margens maiores;
- com uma margem baseada no intervalo de confiança da previsão (ic), a margem praticamente não muda com a mudança do preditor, fazendo com que preditores que mais subestimem o rtt forneçam um melhor desempenho, enquanto preditores precisos fornecem $timeouts$ um pouco maiores, mas que são mais eficazes em relação à precisão do detector.

A análise também mostra que:

- a métrica de precisão T_M reflete o quão cedo, em média, o $timeout$ expirou;
- um detector baseado no preditor ARIMA captura o relacionamento entre amostras adjacentes, possibilitando uma boa precisão com uma margem de segurança pequena;

- em termos de desempenho, preditores conservadores minimizam os grandes T_D^u produzidos pelos grandes desvios gerados por preditores que se ajustam rápido;

- quanto maior a variabilidade entre amostras adjacentes, menor a precisão do LAST em termos da duração de uma falsa suspeita;

- se o *rtt* apresentar alto grau de aleatoriedade, o detector mais preciso é o que ajusta o seu *timeout* de acordo com a média da população;

- para uma boa precisão, os detectores devem conseguir ajustar rapidamente seu *timeout* às mudanças de tendência ou nível.

6.6.2 QoS com amostragens reais

Esta seção apresenta e discute o desempenho e a precisão das implementações do detector de defeitos, quando aplicados a amostragens reais.

6.6.2.1 Metodologia

Considerando a existência das várias implementações do detector de defeitos, cada uma utilizando um dos tipos de preditores especificados no capítulo 5, a avaliação da qualidade de serviço dos detectores com amostragens reais foi realizada considerando três conjuntos de dados:

- um com 23 traços, com duração de um dia cada, os quais incluem finais de semana e períodos da madrugada (amostragens com até 86400 amostras);

- outro com 17 traços, cada um correspondendo a uma amostragem realizada num dia de semana no período das 6 as 24 horas (amostragens com até 64800 amostras); e

- outro com 680 traços, cada um com aproximadamente 3130 amostras. Estes correspondem a 40 traços randomicamente escolhidos do segundo conjunto de dados.

Num primeiro momento, para verificar a influência dos preditores em execuções longas, com 24 e 18 horas, realizou-se a análise da qualidade de serviço utilizando os dois primeiros conjuntos de traços. A seção 6.6.2.2 apresenta e discute os resultados.

Num segundo momento, para poder verificar o impacto dos preditores em execuções de menor duração realizadas em períodos diurnos, faz-se a análise da qualidade de serviço para o terceiro conjunto de dados. Os resultados são apresentados e discutidos na seção 6.6.2.3.

Em ambas análises, os resultados são apresentados na forma de valores médios, obtidos na avaliação dos n traços que compõem o experimento. A avaliação do impacto de uma margem de segurança também foi feita nas duas análises.

6.6.2.2 Análise das amostragens diárias

Análise do desempenho para o conjunto de amostragens de 24h

Considerando os 23 traços do primeiro conjunto de dados, os desempenhos obtidos pelos detectores são apresentados na tabela 6.11. A tabela lista os desempenhos quando a margem é nula, quando é baseada no erro de previsão (*ep*) e quando é baseada no intervalo de confiança da previsão (*ic*).

Tabela 6.11: T_D e T_D^u em amostragens de 24 horas.

Margem nula								
	T_D	Grupo		T_D^u	Grupo		
MEAN	1,045	B		MEAN	1,206		D	
WINMEAN	1,109		A	WINMEAN	2,162			C
DMA	1,109		A	LPF	2,207			C
LPF	1,109		A	DMA	2,245			C
ARIMA	1,109		A	BROWN	2,496	B		C
BROWN	1,109		A	ARIMA	3,034	B	A	
LAST	1,110		A	LAST	3,336		A	
Margem *ep*								
	T_D	Grupo		T_D^u	Grupo		
ARIMA	1,208	B		ARIMA	7,832		A	
BROWN	1,211	B		LAST	8,137		A	
LAST	1,212	B		BROWN	8,185		A	
LPF	1,214	B		LPF	8,216		A	
DMA	1,228	B		DMA	8,922		A	
WINMEAN	1,233	B		WINMEAN	9,225		A	
MEAN	1,559		A	MEAN	10,823		A	
Margem *ic*								
	T_D	Grupo		T_D^u	Grupo		
MEAN	1,143	B		MEAN	3,860	B		
ARIMA	1,210		A	WINMEAN	4,857	B	A	
LAST	1,210		A	DMA	5,052	B	A	
BROWN	1,211		A	LPF	5,172	B	A	
LPF	1,211		A	ARIMA	5,676	B	A	
WINMEAN	1,211		A	BROWN	5,933	B	A	
DMA	1,211		A	LAST	6,608		A	

O resultado mostra que, sem margem de segurança, o tempo médio para detecção de um defeito (T_D), nas sete implementações, praticamente permanece inalterado se o detector de defeitos substituir um dado preditor por outro. A exceção é se o preditor for o MEAN. De acordo com o teste de múltiplas médias de Duncan, pode-se afirmar que o detector de defeitos MEAN é mais rápido do que os demais em sua detecção, pois com este preditor o detector forma o grupo B, o melhor grupo em termos de T_D.

O baixo T_D do MEAN com margem nula deriva do fato de que, ao longo de um dia, o período das 0h às 8h alimenta um grande histórico de pequenos atrasos de comunicação no preditor e, conseqüentemente, quando o atraso aumenta (após as 9h), o preditor MEAN subestima o valor do *rtt* por um período muito longo. Com um *timeout* menor, naturalmente o T_D também será menor. Este efeito não foi percebido com os padrões de amostragem da seção anterior porque os padrões são simétricos, mas pode ser visualizado pelo comportamento do preditor MEAN ilustrado na figura 5.11.

Considerando o T_D^u, ainda com margem nula, o resultado obtido é equivalente ao obtido com padrões de amostragem, ou seja, com preditores baseados na média dos valores amostrados (normalmente conservadores) obtém-se os menores tempos máximos para detecção de um defeito. Este resultado decorre de que um *timeout* sem picos não causa um tempo grande para detecção de um defeito.

Ao aplicar uma margem *ep*, a situação se inverte completamente: o T_D da implementação MEAN é significativamente maior do que o dos demais (está isolado no grupo A de Duncan). O motivo é que o erro de previsão da implementação

MEAN é significativamente o maior (vide resultados do capítulo anterior), o que gera a maior margem de segurança e conseqüentemente o maior T_D. O T_D reflete integralmente o tamanho da margem de segurança. Por outro lado, a implementação ARIMA, a que utiliza o preditor mais preciso, apresenta o melhor desempenho tanto em termos do T_D quanto em termos do T_D^u, embora não haja diferença estatística no T_D^u. Todos os preditores pertencem ao grupo A.

A influência do erro de previsão também pode ser percebida em outras implementações. Nota-se que as implementações baseadas na média aritmética (WINMEAN e DMA) apresentam um T_D levemente superior ao das implementações com preditores não baseados na média, mas o tempo máximo para detecção (T_D^u) tem uma diferença mais acentuada.

Por outro lado, com uma margem ic, as implementações que usam preditores baseados na média são as que apresentam os melhores desempenhos, sendo que a MEAN se destaca como a melhor, em termos do T_D. Em termos do T_D^u, os preditores baseados na média auxiliam na obtenção dos melhores desempenhos. O motivo é o mesmo que para margem nula: os preditores baseados na média são os que mais subestimam o atraso. A magnitude da margem ic é similar para todos os preditores.

Análise do desempenho para o conjunto de amostragens de 18h

Considerando 17 execuções com aproximadamente 64800 amostras do rtt em cada uma, correspondentes a 18 horas de amostragem durante dias úteis, os desempenhos das sete implementações do detector de defeitos são apresentados na tabela 6.12. A amostragem ainda corresponde a execuções longas, mas restringe os dados ao período diurno, onde a rede é mais utilizada, e conseqüentemente a variabilidade do rtt é significativamente maior.

Tabela 6.12: T_D e T_D^u em amostragens de 18 horas.

Margem nula							
	T_D	Grupo		T_D^u	Grupo		
MEAN	1,121	A		MEAN	1,372	D	
WINMEAN	1,165	A		WINMEAN	2,379		C
ARIMA	1,166	A		LPF	2,435		C
BROWN	1,166	A		DMA	2,482		C
LPF	1,166	A		BROWN	2,814	B	C
DMA	1,166	A		ARIMA	3,464	B	A
LAST	1,166	A		LAST	3,936		A
Margem ep							
	T_D	Grupo		T_D^u	Grupo		
ARIMA	1,321	B		ARIMA	9,513		A
BROWN	1,326	B		LAST	9,690		A
LAST	1,327	B		BROWN	9,971		A
LPF	1,332	B		LPF	9,982		A
DMA	1,355	B		DMA	10,868		A
WINMEAN	1,365	B		WINMEAN	11,256		A
MEAN	1,998		A	MEAN	13,153		A
Margem ic							
	T_D	Grupo		T_D^u	Grupo		
MEAN	1,275		A	MEAN	4,821	B	
LAST	1,323		A	WINMEAN	5,736	B	A
ARIMA	1,324		A	DMA	6,010	B	A
BROWN	1,324		A	LPF	6,181	B	A
LPF	1,324		A	ARIMA	6,733	B	A
WINMEAN	1,324		A	BROWN	7,165	B	A
DMA	1,325		A	LAST	8,055		A

Considerando as três margens (nula, ep e ic), também com traços de 18 horas, percebe-se que o tempo médio para detecção de um defeito (T_D) não se

altera significativamente com a alteração do preditor utilizado. O único preditor que provoca uma alteração significativa continua sendo o MEAN quando a margem *ep* é aplicada, caso em que ele é a pior escolha.

A implementação ARIMA, assim como para traços de 24 horas, continua fornecendo o melhor desempenho (T_D e T_D^u) com uma margem de segurança *ep*, confirmando que, para este tipo de margem, preditores precisos (com menores erros quadráticos médios) são os mais indicados.

Por outro lado, quando a margem de segurança é computada em função do intervalo de confiança da previsão, as implementações baseadas na média aritmética oferecem os melhores desempenhos (T_D^u), mesmo quando há maior variabilidade no *rtt* (traços de períodos diurnos). As diferenças entre os detectores aponta que somente o MEAN e o LAST diferem em relação ao T_D^u.

No geral, percebe-se que a situação não se altera, se comparada a traços de 24 horas: com margem *ep*, o melhor preditor é o mais preciso, ou seja, o ARIMA, pois é o que gera a menor margem de segurança e, com margem *ic*, preditores conservadores (baseados na média) são preferidos, pois a margem não muda significativamente entre o preditores e estes são os que mais subestimam o *rtt*.

Em síntese, a análise dos desempenhos com traços de 24 e 18 horas mostra que:

1. de um ponto de vista estatístico, o tempo médio para detecção de um defeito (T_D) é similar para todas as implementações, exceto para a MEAN, a qual resulta num tempo médio para detecção de um defeito (T_D) significativamente melhor em alguns casos. Entretanto, o resultado decorre da sua incapacidade para gerar uma estimativa maior do que a do valor amostrado;

2. com margens que sofrem pouca influência da precisão do preditor, tal como a *ic*, preditores baseados na média (conservadores) resultam em menores tempos máximos para detectar um defeito (T_D^u). Preditores que reagem mais rapidamente às mudanças no *rtt* (menos conservadores), embora mais precisos, geram grandes desvios; logo, um maior T_D^u; e

3. com margens de segurança variáveis em função do erro de previsão, tal com a *ep*, preditores precisos resultam em melhores desempenhos (T_D e T_D^u) em termos absolutos.

Análise da precisão

Como definido na seção 6.5, o detector de defeitos mais preciso é o que fornece conjuntamente o menor tempo médio de duração de um erro (T_M) e o maior tempo médio de recorrência ao erro (T_{MR}). Entretanto, de acordo com as tabelas 6.13 e 6.14, as quais apresentam as medidas de T_M e T_{MR} obtidas com amostragens reais diárias de 24 e 18 horas, respectivamente, com base somente nas duas métricas primárias, nenhuma das implementações do detector de defeitos pode ser considerada como sendo a mais precisa, pois nenhuma oferece o menor T_M e o maior T_{MR} simultaneamente. Assim, para tais amostragens, faz-se necessário avaliar as métricas de precisão secundárias λ e P_A.

Considerando amostragens com 24 horas (tabela 6.13) e uma margem de segurança nula, observa-se que um preditor ARIMA faz com que o detector de

Tabela 6.13: Métricas de precisão para amostragens com 24 horas.

Margem	Métrica	LAST	MEAN	WINMEAN	DMA	BROWN	ARIMA	LPF
nula	T_M	0,029	0,211	0,042	0,040	0,032	0,040	0,037
	T_{MR}	2,424	3,431	3,049	3,106	2,813	3,665	3,276
	λ	0,412541	0,291460	0,327976	0,321958	0,355492	0,272851	0,305344
	P_A	0,988036	0,938502	0,986225	0,987444	0,988624	0,989359	0,988702
ep	T_M	0,057	0,093	0,052	0,058	0,051	0,077	0,056
	T_{MR}	30,084	74,769	29,112	32,013	26,517	40,388	29,933
	λ	0,033241	0,013375	0,034350	0,031237	0,037712	0,024760	0,033409
	P_A	0,998138	0,998756	0,998214	0,998219	0,998077	0,998118	0,998163
ic	T_M	0,055	0,209	0,053	0,052	0,056	0,067	0,058
	T_{MR}	30,188	10,831	26,163	26,495	31,089	36,076	31,116
	λ	0,033126	0,092328	0,038222	0,03744	0,032166	0,027719	0,032138
	P_A	0,998178	0,980796	0,997974	0,998075	0,998199	0,998143	0,998136

Tabela 6.14: Métricas de precisão para amostragens com 18 horas.

Margem	Métrica	LAST	MEAN	WINMEAN	DMA	BROWN	ARIMA	LPF
nula	T_M	0,042	0,269	0,060	0,056	0,045	0,053	0,051
	T_{MR}	2,257	2,184	2,683	2,659	2,478	2,967	2,749
	λ	0,443066	0,457875	0,372717	0,376081	0,403551	0,337041	0,363769
	P_A	0,981391	0,876832	0,977637	0,978939	0,981840	0,982137	0,981448
ep	T_M	0,100	0,122	0,095	0,101	0,090	0,127	0,099
	T_{MR}	45,605	80,475	47,331	49,410	40,446	57,107	46,043
	λ	0,021927	0,012426	0,021128	0,020239	0,024724	0,017511	0,021719
	P_A	0,997807	0,998484	0,997993	0,997956	0,997775	0,997776	0,997850
ic	T_M	0,093	0,249	0,086	0,085	0,097	0,111	0,099
	T_{MR}	42,424	5,673	34,856	35,953	45,442	51,464	44,495
	λ	0,023572	0,176274	0,028689	0,027814	0,022006	0,019431	0,022474
	P_A	0,997808	0,956108	0,997533	0,997636	0,997865	0,997843	0,997775

defeitos correspondente seja o mais preciso em termos das duas métricas secundárias avaliadas. A sua taxa de erros λ é $0,272821$ e a sua probabilidade de uma resposta precisa P_A é $0,989359$. Logo, mesmo que o detector ARIMA não tenha o menor T_M, o alto T_{MR} lhe confere a melhor precisão em termo das métricas secundárias λ e P_A. O detector MEAN também tem um bom T_{MR}, o que lhe deixa com o segundo menor λ, mas seu alto T_M resulta na menor probabilidade de uma resposta precisa ($P_A = 0,938502$).

Ao considerar amostragens com 18 horas, ainda com uma margem de segurança nula, observa-se (tabela 6.14) que a implementação ARIMA oferece a menor taxa de erros ($\lambda = 0,337041$ por segundo), sendo seguida pela implementação LPF. Neste conjunto de dados, onde a variabilidade do *rtt* é mais significativa, a implementação MEAN é a que resulta na menor precisão, tanto em termos da taxa de erros ($\lambda = 0,457875$ por segundo) como em termos da probabilidade de uma resposta precisa ($P_A = 0,876832$). A baixa precisão da implementação MEAN deve-se à baixa qualidade da previsão MEAN e da não aplicação de qualquer margem de segurança.

O uso da margem *ep* faz com que a implementação MEAN seja a mais precisa das implementações, tanto em termos de λ como em termos do P_A, quando consideradas amostragens de 24 e 18 horas. Tal resultado reflete a grande margem de segurança gerada em função do significativo erro de previsão apresentado pelo preditor MEAN, o que também gera o baixo desempenho desta implementação (vide tabelas 6.11 e 6.12).

Do ponto de vista da taxa de erros, com margem *ep*, o detector ARIMA é o que tem a segunda menor taxa de erros ($\lambda = 0,017511$ por segundo). Por outro lado, diferente da implementação MEAN, a baixa taxa de erros deve-se à boa precisão do preditor ARIMA e não à sua alta margem de segurança. Com uma boa precisão, a

implementação ARIMA oferece o melhor desempenho (o desempenho T_D do MEAN é 51% pior do que o do ARIMA com a margem *ep*, enquanto que o T_D^u é 38% pior). Tal constatação mostra que o detector baseado em séries temporais ARIMA pode obter uma boa precisão com um bom desempenho.

Em termos da probabilidade de uma resposta precisa, o uso da margem em função do erro faz com que todas as implementações apresentem praticamente a mesma precisão, $P_A \cong 0,998$. A diferença entre o maior e o menor P_A é de 0,000679 ou 0,0679%. Deste resultado, tem-se que, com uma margem em função do erro médio, o uso de um ou outro preditor não afeta a probabilidade de uma resposta precisa, quando o detector é consultado pela aplicação.

Com uma margem *ic*, a probabilidade de uma resposta precisa também fica praticamente igual para todos os detectores. Exceto o MEAN, que oferece um P_A significativamente menor do que os demais, os outros oferecem um $P_A \cong 0,9978$. Entretanto, do ponto de vista da taxa de erros, com a margem *ic*, a implementação ARIMA é a mais eficiente, tanto para as amostragens de 24 horas como para as de 18 horas. A taxa de erros do segundo mais preciso, o LPF, é 15% superior a do ARIMA. Mais uma vez, o resultado é reflexo da boa precisão do preditor ARIMA. Como seu desempenho com esta margem é compatível com o oferecido pelos demais detectores, com esta margem de segurança, o preditor ARIMA oferece a melhor relação *custo* `x` *benefício*.

Da análise da precisão, com traços de 24 e 18 horas, tem-se que:

1. a margem de segurança interfere diretamente na precisão do detector de defeitos, mascarando a baixa qualidade de alguns preditores como, por exemplo, o MEAN;

2. a relação *custo* `x` *benefício* revela o preditor ARIMA como sendo o melhor; e

3. o preditor ARIMA demonstrou que, independente da margem de segurança, sua boa precisão auxilia na melhoria da qualidade de serviço de um detector de defeitos.

6.6.2.3 Análise das amostragens de pequena duração

Similarmente à seção anterior, esta seção apresenta e discute as métricas de QoS (desempenho e precisão) das implementações do detector de defeitos quando as implementações recebem traços reais de pequena duração (3130 amostras).

Análise do desempenho

Considerando o novo conjunto de dados, as métricas de desempenho T_D e T_D^u são apresentadas na tabela 6.15. Tanto os números absolutos do T_D quanto a classificação resultante do teste de múltiplas médias de Duncan, indicam que, para uma margem nula e uma margem *ic*, não há diferença significativa entre os detectores em termos do T_D, mas somente em termos do T_D^u.

A similaridade da qualidade de serviço em termos do T_D decorre da característica global dos preditores, ou seja, tentar corrigir seus erros a cada passo. Tal tentativa de correção faz com que o *timeout* oscile em torno da curva do *rtt*, resultando em momentos onde o *rtt* é superestimado e momentos em que ele é subestimado. Assim, com uma margem nula, ou uma margem similar para todos os preditores (*ic*),

ao fazer a média do T_D, a oscilação positiva anula a negativa, independentemente da amplitude da oscilação. Como conseqüência, o tempo médio para detecção de um defeito T_D esconde as características individuais dos preditores. Este resultado também foi observado na análise com padrões de amostragem (seção 6.6.1.2) e com traços reais diários, e mostra que, sem margem de segurança ou com margem *ic*, a troca de um preditor não altera significativamente a qualidade de serviço T_D de um detector.

Tabela 6.15: T_D e T_D^u em amostragens de pequena duração.

Margem nula						
	T_D	Grupo		T_D^u	Grupo	
MEAN	1,242	A		MEAN	1,439	D	
WINMEAN	1,252	A		WINMEAN	1,605		C
DMA	1,253	A		LPF	1,636		C
LPF	1,253	A		DMA	1,640		C
ARIMA	1,253	A		BROWN	1,764	B	
BROWN	1,253	A		ARIMA	1,815	B	
LAST	1,253	A		LAST	2,132		A
Margem *ep*						
	T_D	Grupo		T_D^u	Grupo	
ARIMA	1,462		C	LPF	3,481	B	
BROWN	1,471		C	DMA	3,611	B	
LAST	1,473		C	WINMEAN	3,656	B	
LPF	1,481		C	BROWN	3,676	B	
DMA	1,514	B		ARIMA	3,715	B	
WINMEAN	1,530	B		LAST	4,121		A
MEAN	1,821		A	MEAN	4,323		A
Margem *ic*						
	T_D	Grupo		T_D^u	Grupo	
MEAN	1,481	A		MEAN	2,284		E
LPF	1,491	A		WINMEAN	2,451	D	
BROWN	1,491	A		DMA	2,507	D	
WINMEAN	1,492	A		LPF	2,549	D	C
DMA	1,492	A		ARIMA	2,675	B	C
LAST	1,492	A		BROWN	2,737	B	
ARIMA	1,492	A		LAST	2,976		A

A amplitude da oscilação não aparece no T_D mas aparece no T_D^u. Detectores que seguem a média das amostras resultam num T_D^u menor, pois a margem não difere muito entre as implementações e o preditor não gera picos na curva de previsão.

No caso da margem *ep*, as implementações LAST e MEAN resultam nos piores T_D^u. Dentre os demais, não há diferença significativa (todos pertencem ao grupo B). Em termos do T_D, o detector MEAN é o pior (é o único no grupo A de Duncan). Os outros dois detectores baseados na média (WINMEAN e DMA) também se destacam formando o segundo pior grupo (grupo B). Já os demais detectores (mais precisos), por apresentarem um baixo erro quadrático médio, geram as menores margens de segurança e conseqüentemente os menores T_D. O LAST tem um pior T_D^u devido ao seu comportamento nada conservador, o que resulta em picos na curva de previsão, os quais se refletem na curva do *timeout*.

Em síntese, os resultados obtidos com traços reais de pequena duração mostram que:

1. sem margem de segurança ou com margem *ic*, a troca de um preditor não altera significativamente a qualidade de serviço T_D de um detector, mas preditores conservadores resultam em melhores T_D^u; e

2. com uma margem *ep*, preditores precisos como o ARIMA, BROWN e LPF resultam em detectores com bons desempenhos, tanto em termos do T_D

como do T_D^u. Preditores impulsivos, como o LAST, ou muito conservadores, como o MEAN, são indesejáveis com esta margem.

Análise da precisão

Para o conjunto de 680 traços (todos em dias úteis das 6 às 24h), a precisão oferecida por cada implementação do detector de defeitos é descrita na tabela 6.16. Dos resultados, tem-se que, independente da margem de segurança, nenhum preditor torna o detector de defeitos o mais preciso, pois nenhuma implementação oferece o menor T_M e o maior T_{MR} simultaneamente.

Tabela 6.16: Métricas de precisão para amostragens de pequena duração.

Margem nula										
	T_M	Grupo			T_{MR}	Grupo		λ		P_A
LAST	0,054			F	MEAN	3,196		A	0,312891	ARIMA	0,978780
ARIMA	0,056		E	F	ARIMA	2,639	B		0,378931	BROWN	0,975703
BROWN	0,057	D	E		LPF	2,523		C	0,396354	LAST	0,975543
LPF	0,062	D			WINMEAN	2,521		C	0,396668	LPF	0,975426
DMA	0,069		C		DMA	2,476		C	0,403877	DMA	0,972132
WINMEAN	0,075	B			BROWN	2,346	D		0,426257	WINMEAN	0,970250
MEAN	0,153		A		LAST	2,208		E	0,452899	MEAN	0,952128
Margem *ep*										
	T_M	Grupo			T_{MR}	Grupo		λ		P_A
BROWN	0,249			D	MEAN	368,543		A	0,002713	MEAN	0,998993
LAST	0,253		C	D	WINMEAN	139,798	B		0,007153	WINMEAN	0,998011
LPF	0,270	B	C	D	DMA	131,029	B	C	0,007632	DMA	0,997932
DMA	0,271	B	C	D	LPF	124,453	B	C	0,008035	LPF	0,997831
WINMEAN	0,278	B	C		ARIMA	120,312	B	C	0,008312	ARIMA	0,997648
ARIMA	0,283	B			LAST	112,459		C	0,008892	BROWN	0,997773
MEAN	0,371		A		BROWN	111,789		C	0,008945	LAST	0,997750
Margem *ic*										
	T_M	Grupo			T_{MR}	Grupo		λ		P_A
MEAN	0,178	F			ARIMA	169,314		A	0,005906	BROWN	0,998028
WINMEAN	0,208		E		BROWN	168,825		A	0,005923	ARIMA	0,997962
DMA	0,238	D			LPF	145,658	B		0,006865	LPF	0,997899
LAST	0,269		C		LAST	123,029		C	0,008128	LAST	0,997814
LPF	0,306	B			DMA	102,329	D		0,009772	DMA	0,997674
BROWN	0,333		A		WINMEAN	81,730		E	0,012235	WINMEAN	0,997455
ARIMA	0,345		A		MEAN	36,718	F		0,027235	MEAN	0,995152

Para uma margem de segurança nula, as métricas secundárias (λ e P_A) indicam que: o detector MEAN é melhor em termos da taxa de erros, mas tem a mais baixa probabilidade de uma resposta precisa (95% contra 97% dos demais), e que o detector ARIMA é o melhor em termos da probabilidade de uma resposta precisa e tem a segunda melhor taxa de erros. Em termos estatísticos, o ARIMA distingüe-se dos demais detectores por oferecer uma boa regularidade em termos de T_M e T_{MR}. Em ambas as métricas, ele pertence ao segundo melhor grupo[3].

Com uma margem *ep*, estatisticamente há menos diferenças entre os detectores do que com margem nula (vide formação dos grupos na tabela 6.16). Ao invés de sete grupos para o T_M, agora há somente quatro grupos, e ao invés de cinco grupos para o T_{MR}, agora há somente três grupos. Em termos práticos, isto significa que a precisão oferecida pelos detectores mostra-se mais equilibrada, pois a troca entre os preditores BROWN, LAST, LPF e DMA não altera significativamente o T_M oferecido, embora haja diferença absoluta não significativa entre suas médias. Do mesmo

[3] A análise da variância das medidas indica que há diferenças significativas entre as precisões oferecidas pelos diferentes detectores ($\Pr < 0{,}001$ tanto para o T_M como para o T_{MR}) e as diferenças apontadas pelo teste de Duncan são descritas nas colunas *Grupo* da tabela 6.16.

modo, a troca entre os preditores WINMEAN, DMA, LPF e ARIMA, também não altera significativamente a qualidade de serviço T_{MR} oferecida pelo detector. De qualquer modo, com a margem *ep*, os preditores menos precisos baseados na média, especialmente o MEAN, são os que tornam o detector mais preciso, tanto em termos do λ quanto em termos do P_A.

A boa precisão dos detectores baseados na média deriva do grande erro de previsão dos seus preditores, principalmente do MEAN, o que resulta em grandes *timeouts* (grandes erros geram grandes margens de segurança). Um grande *timeout* resulta num grande T_{MR}, o que compensa o baixo T_M a ponto de não interferir em P_A. O problema é que um grande *timeout* resulta em longo tempo para detecção de um defeito, conforme discutido na análise de desempenho. Na tabela 6.15 mostrou-se que, considerando a margem *ep*, os tempos máximos para detecção de defeitos (T_D) são maiores nos detectores baseados na média.

Por outro lado, em termos de uma relação *custo* `x` *benefício*, a taxa de erros dos detectores LPF e ARIMA compete com as da média (pertencem ao segundo grupo mais preciso, juntamente com WINMEAN e DMA), mas oferecem os melhores desempenhos (T_D^u e T_D, respectivamente). Assim, tais detectores obtêm um bom desempenho, sem comprometer a precisão, pois a taxa de erros fica entre as do grupo B, perdendo somente para o MEAN. A probabilidade de uma resposta precisa fica apenas 0,11% menor do que o MEAN.

Com uma margem de segurança *ic*, a precisão oferecida define um novo cenário, conforme apresentado na tabela 6.16. Neste cenário, dois detectores se destacam em função da taxa de erros e da probabilidade de uma resposta precisa: os detectores ARIMA e BROWN. Ambos são estatisticamente melhores do que os demais quanto à taxa de erros, e um pouco melhores do que os demais em termos absolutos da probabilidade de uma resposta precisa. De maneira similar ao MEAN com margem *ep*, o bom T_{MR} destes detectores compensa o baixo T_M.

É importante salientar que a taxa de erros dos detectores é dependente da qualidade do preditor, pois há uma relação estrita entre a classificação dos preditores mais precisos (tabela 5.5) com a classificação dos detectores com menor taxa de erros (tabela 6.16). Resultado similar também é percebido para a P_A. Além disto, é importante salientar que, embora o desempenho medido no pior caso do detector MEAN seja 17,1% melhor do que o do detector ARIMA (T_D^u do MEAN é 2,284 segundos e do ARIMA é 2,675 segundos, conforme tabela 6.15), e 19,8% melhor do que o do BROWN ($T_D^u = 2,737$), a taxa de erros do ARIMA e do BROWN é 78,3% melhor do que a do MEAN. Em outras palavras, o ganho em relação à precisão é muito superior à perda em relação ao desempenho, o que faz dos preditores baseados em séries temporais, ARIMA e BROWN, os mais indicados com margem *ic* em termos da relação *custo* `x` *benefício*.

Em síntese, da análise da precisão com amostragens de pequena duração, tem-se que:

1. em termos das métricas primárias, nenhum preditor torna o detector o mais preciso;

2. a determinação do melhor preditor varia de acordo com a margem de segurança adotada;

3. com uma margem em função do erro de previsão, detectores que ajustam seu *timeout* baseando-se nos valores médios dos atrasos de comunicação,

obtêm melhor precisão do que os que baseiam-se em preditores mais precisos, mas seu desempenho é menor;

4. com margem de segurança *ic*, os preditores ARIMA e BROWN resultam nas mais baixas taxas de erros, enquanto que o desempenho é um pouco inferior aos demais. Logo, resultam numa boa relação *custo* `x` *benefício*.

6.7 Conclusões Parciais

Neste capítulo, avaliou-se a qualidade de serviço (QoS) de sete versões de um detector de defeitos do estilo *pull*, onde cada versão implementa um mecanismo diferente de ajuste dinâmico do *timeout*. Para a avaliação, foram utilizadas métricas de qualidade de serviço independentes do *timeout* (métricas de Chen, Toueg e Aguilera), logo independentes dos detalhes de implementação ou de parâmetros de configuração dos algoritmos. Para favorecer a comparação de cada versão dos algoritmos de previsão, utilizou-se um simulador que computa as métricas de QoS com base nos mesmos conjuntos de dados previamente coletados.

Considerando que o valor retornado por um mecanismo de previsão é uma tentativa para acertar o próximo valor amostrado, todo detector de defeitos auto-ajustável deve adicionar ao valor predito uma margem de segurança. Dentre as margens de segurança propostas na literatura (capítulo 2), a margem variável em função do erro de previsão (margem *ep*) foi utilizada nas comparações. Entretanto, como esta margem não define intervalos de confiança, uma nova margem de segurança foi proposta e especificada, a margem variável em função do intervalo de confiança da previsão (margem *ic*).

Como resultado da avaliação da QoS das sete versões, usando padrões de amostragens fictícios e reais (longos e curtos), conclui-se que:

1. a simples escolha de um preditor não garante que o detector de defeitos auto-ajustável seja o mais rápido ou o mais preciso, pois nenhum preditor garante o melhor desempenho e a melhor precisão independentemente da margem de segurança;

2. a escolha de um preditor não deve ser realizada de forma dissociada da escolha de uma margem de segurança, pois o tipo de margem depende do tipo de preditor escolhido e dos requisitos da aplicação;

3. preditores precisos fazem um detector de defeitos tornar-se mais preciso com o auxílio de uma margem de segurança menor;

4. do ponto de vista do desempenho (T_D), preditores precisos devem ser utilizados com margens baseadas no erro de previsão, enquanto preditores não muito precisos como, por exemplo, os baseados na média, devem ser utilizados com margens que sofrem pouca influência do erro de previsão;

5. do ponto de vista da precisão, preditores conservadores como, por exemplo, os baseados na média, devem ser utilizados com uma margem baseada no erro de previsão, enquanto preditores que se ajustam rápido às variações da amostragem (mais precisos) devem ser utilizados com margens de segurança fixas ou variáveis em função do intervalo de confiança da previsão;

6. a relação *custo* x *benefício* indica que, para traços de longa duração, preditores tais como o ARIMA, que conseguem capturar dinamicamente o comportamento do rtt, são adequados com os dois tipos de margens avaliadas (ep e ic). Para traços de curta duração, com margem ep, o detector MEAN oferece a melhor relação, mas usando margem ic, os detectores ARIMA e BROWN oferecem as mais baixas taxas de erros, enquanto que o desempenho é um pouco inferior aos demais;

7. uma margem em função do intervalo de confiança da previsão é mais eficiente do que uma margem em função do erro médio, pois a primeira resulta num T_D^u menor em todas as implementações.

7 UM SERVIÇO DE DETECÇÃO DE DEFEITOS ADAPTATIVO

No capítulo 6, mostrou-se que, para obter uma boa qualidade de serviço num detector de defeitos auto-ajustável (adaptativo), é importante combinar adequadamente o tipo de preditor e o tipo de margem de segurança. Entretanto, esta combinação depende dos requisitos da aplicação, ou seja, se ela deseja priorizar o desempenho, a precisão, ou a melhor relação *custo* `x` *benefício*. Neste sentido, um detector de defeitos deveria possibilitar que a aplicação pudesse configurar a combinação *margem* `x` *preditor* desejada, ou mesmo que pudesse experimentar diferentes combinações, a fim de verificar qual delas oferece os melhores resultados no ambiente específico utilizado pela aplicação. Similarmente, configurar e experimentar algoritmos de detecção distintos pode também ser desejável.

Este capítulo discute as vantagens de se ter detectores de defeitos na forma de um serviço (seção 7.1) e apresenta uma arquitetura configurável e flexível para um serviço de detecção de defeitos (seção 7.2), na qual os algoritmos de detecção, predição e de cômputo da margem de segurança podem ser facilmente escolhidos, dentre várias opções disponíveis, e configurados. O capítulo também apresenta o AFDService (seção 7.3), um serviço de detecção de defeitos que implementa a arquitetura proposta para permitir que o cliente possa escolher dentre um conjunto de preditores e margens. O AFDService implementa um detector de defeitos configurável do estilo *pull*, sete estratégias para adaptação dinâmica do *timeout* e três tipos de margem de segurança.

7.1 Detector de defeitos como um serviço

Usuais em muitos domínios de aplicação como, por exemplo, em sistemas de controle e supervisão e na resolução de acordo em sistemas distribuídos, mecanismos de detecção de defeitos podem ser especificados de várias maneiras, sendo encontradas as seguintes variações:

- no estilo do fluxo de informações - estilo *push*, *pull* ou *dual* (FELBER et al., 1999);
- na arquitetura lógica de comunicação - a comunicação pode ser por difusão (*broadcast*) (CHANDRA; TOUEG, 1996), de maneira hierárquica (BRASILEIRO; FIGUEIREDO; SAMPAIO, 2002; FELBER et al., 1999), ou em anel (LARREA; ARÉVALO; FERNÁNDEZ, 1999);
- no estilo de ajuste do *timeout* - o *timeout* pode ser fixo ou ser ajustado para acompanhar o comportamento do atraso de comunicação (adaptativo);

- quanto à estratégia de ajuste, se o *timeout* for variável - pode ser baseada no último valor amostrado (FELBER, 2000), na média (CHEN; TOUEG; AGUILERA, 2002), ou em séries temporais (NUNES; JANSCH-PÔRTO, 2003); ou

- quanto às propriedades asseguradas pelo detector de defeitos - variações dentre garantias resultantes das propriedades de abrangência e precisão (CHANDRA; TOUEG, 1996; LARREA, 2002) ou a obediência a requisitos de qualidade de serviço (CHEN; TOUEG; AGUILERA, 2002).

Como conseqüência das diferentes especificações, encontram-se diversas implementações para detectores de defeitos, as quais costumam estar fundidas aos protocolos clientes. Esta abordagem melhora o desempenho das aplicações (SERGENT; DÉFAGO; SCHIPER, 1999), mas aumenta a complexidade do projeto e da implementação do protocolo, pois o projetista também deve se preocupar em resolver problemas temporais, o que aumenta o tempo de desenvolvimento e, conseqüentemente, o custo do projeto.

Para reduzir a complexidade do projeto da aplicação, um detector de defeitos pode ser encapsulado num serviço oferecido no *middleware* (RENESSE; MINSKY; HAYDEN, 1998). Conceitualmente, um detector pode ser enquadrado como um *serviço* que provê informações sobre componentes defeituosos. Adicionalmente, numa modelagem orientada a objetos, detectores de defeitos podem ser classificados como objetos primitivos (FELBER et al., 1999), o que facilita a proposição de uma arquitetura genérica para o serviço.

Para possibilitar a aplicabilidade de um serviço de detecção em qualquer contexto, Felber et al. (1999) utilizaram um conjunto genérico de interfaces de objetos para um serviço deste tipo. A figura 7.1 ilustra o diagrama de classes das interfaces do serviço de monitoramento de objetos, onde a linha pontilhada salienta as que são orientadas à aplicação.

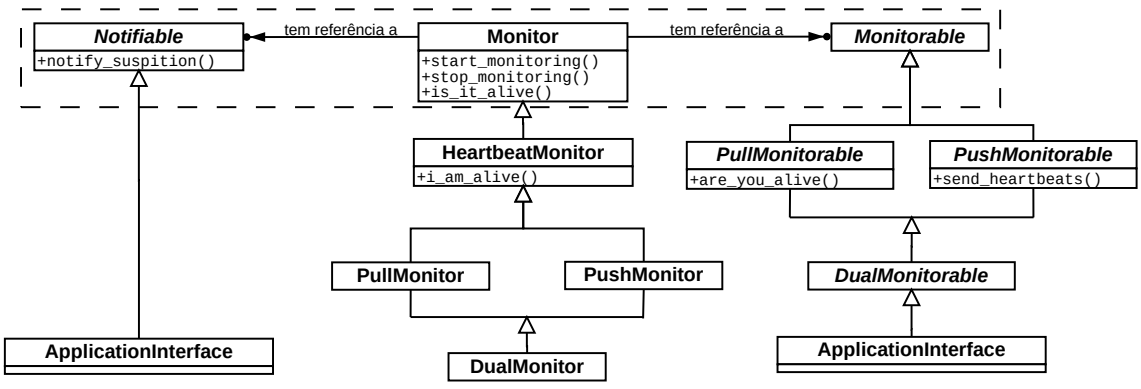

Figura 7.1: Diagrama de classes das interfaces do serviço de monitoramento de objetos utilizado por Felber et al. (1999).

Da figura, percebe-se que Felber et al. definem três componentes básicos para um detector de defeitos: um monitor (`Monitor`), um monitorável (`Monitorable`) e um notificável (`Notifiable`). Componentes monitoráveis são componentes da aplicação que devem ser observados pelos componentes monitores. Componentes notificáveis são componentes da aplicação que desejam ser notificados de maneira assíncrona sobre as alterações de estado (normalmente percepção de acessibilidade)

dos componentes monitoráveis. Componentes monitores encapsulam o algoritmo de detecção de defeitos; logo colecionam informações sobre o estado dos componentes monitoráveis da aplicação. Um componente monitor interage com a aplicação cliente, respondendo às suas solicitações de estado ou notificando-a quando ocorrerem mudanças de estado.

A vantagem de trabalhar-se com três componentes orientados à aplicação é que o serviço de detecção pode ser adaptado a diferentes topologias de redes (hierárquicas ou não), sem que a aplicação ou os protocolos distribuídos envolvidos necessitem ser modificados (FELBER et al., 1999). Para isto, basta a aplicação indicar, dentre seus componentes, quais são monitores, quais são monitorados e quais devem ser notificados sobre mudanças de estado (acessibilidade) dos componentes monitorados. Observa-se que as interfaces `Notifiable` e `Monitorable` devem ser implementadas pela aplicação, enquanto que a classe `Monitor` é implementada pelos algoritmos de detecção.

Em termos de configurabilidade dos parâmetros de um detector, do ponto de vista da aplicação, as interfaces de objeto de Felber et al. dão uma *visão limitada* ao cliente. O esforço foi na definição de um conjunto mínimo de métodos para a interação cliente-serviço, pois a finalidade principal foi a de isolar da aplicação detalhes funcionais dos detectores providos pelo serviço.

Do ponto de vista do serviço, distintos algoritmos de detecção de defeitos podem ser implementados especializando a classe `Monitor`, o que melhora a reutilização de código. Por motivo similar, a arquitetura de Felber et al. define um conjunto de interfaces derivadas da interface `Monitorable`, conforme ilustra a figura 7.1, as quais devem ser implementadas de acordo com o estilo de detecção pretendido. Entretanto, um projeto baseado na especialização de classes e na implementação de interfaces específicas, por estilo de detecção, dificulta a substituição de um algoritmo de um estilo por outro. Para tanto, o código da aplicação deveria ser alterado, pois o detector deveria trocar de tipo.

Quando se pretende dar ao cliente a liberdade de poder configurar parâmetros de grão fino no detector como, por exemplo, sua margem de segurança ou o tipo de preditor utilizado no cômputo do *timeout*, um conjunto mais amplo de métodos é necessário. Similarmente, para dar flexibilidade de escolha do algoritmo, uma interface comum a diversos algoritmos de detecção é importante.

7.2 Arquitetura configurável e flexível para um serviço de detecção

Esta seção apresenta uma arquitetura configurável e flexível para um serviço de detecção de defeitos. Ela é configurável no sentido de que permite a configuração de parâmetros de grão fino no detector. É flexível no sentido de que permite a troca, estática ou dinâmica, de algoritmos de detecção, de predição ou de margem de segurança, bem como permite a rápida inclusão de novos algoritmos.

Inicialmente, mostra-se que os algoritmos de detecção e de previsão podem ser dissociados um do outro e que a arquitetura básica de um serviço de detecção, por sua vez, pode separar algoritmos de detecção e de previsão (seção 7.2.1). Em seguida, considerando as interfaces de objeto orientadas à aplicação sugeridas por Felber et al., definem-se novos métodos voltados à aplicação, bem como define-se uma nova arquitetura de interfaces voltada ao serviço (seção 7.2.2). Estas definições

habilitam a configurabilidade do serviço de detecção. Finalmente, propõe-se o uso do padrão de projeto *Strategy* (GAMMA et al., 1994) para obter flexibilidade na arquitetura do serviço (seção 7.2.3).

7.2.1 Dissociando algoritmos de detecção e de previsão

Algoritmos de detecção de defeitos são algoritmos distribuídos que têm como objetivo principal monitorar o estado de componentes distribuídos. Conceitualmente, as versões básicas de tais algoritmos não necessitam *prever* valores, pois seu *timeout* costuma ser fixo[1], o que sugere que um algoritmo de detecção independe de algoritmos de previsão. Por outro lado, algumas versões que ajustam dinamicamente o seu *timeout*, utilizam algoritmos de previsão de valores, conforme discutido na seção 2.3.

Algoritmos de previsão, tais como os de previsão dos tempos de comunicação, costumam ser algoritmos centralizados que têm como objetivo prever valores futuros com base em alguma abordagem de previsão (o capítulo 5 apresentou algumas abordagens).

Em síntese, a previsão do tempo de comunicação não deveria ser uma tarefa de um algoritmo de detecção de defeitos, mas sim de um algoritmo de previsão específico associado a ele, pois conceitualmente pertencem a categorias distintas.

Considerando que algoritmos de detecção e de previsão pertencem a categorias distintas, propõe-se um serviço de detecção de defeitos adaptativo constituído, num dado nó, por dois módulos: um módulo de detecção e um módulo de previsão. A figura 7.2 ilustra a arquitetura básica do serviço de detecção composto por estes dois módulos, bem como as relações entre os módulos locais do serviço. Num dado nó i, o módulo local de detecção será referido a seguir como módulo fd_i.

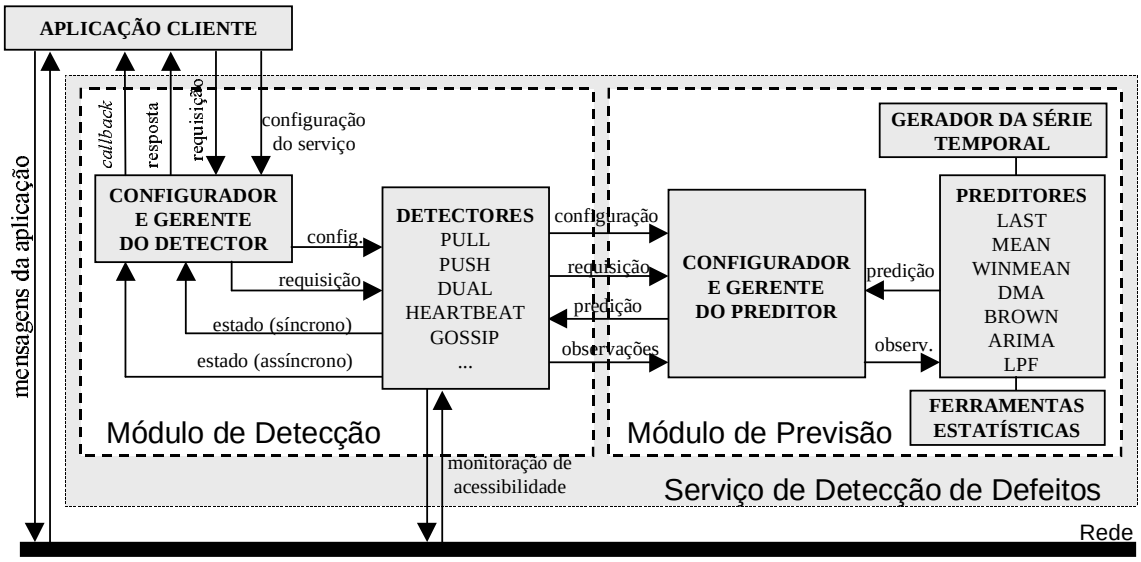

Figura 7.2: Arquitetura dos módulos de detecção e previsão.

Os módulos de detecção fd_i que compõem o algoritmo de detecção FD interagem entre si através da rede de comunicação e são responsáveis pela interação com a aplicação cliente. A interação entre módulos fd_i normalmente restringe-se à

[1] O *timeout* fixo pode também ser acrescentado de uma constante k a cada percepção de uma falsa suspeita, resultando num tempo máximo de espera incremental.

monitoração de acessibilidade, enquanto a interação com os clientes serve para ajustar a *configuração do serviço* e prover a adequada troca de informações de estado entre serviço e aplicação. Especificamente, a configuração compreende a definição da topologia (quais módulos fd_i devem ser monitorados por cada fd_j), do estilo de detecção, da abordagem de adaptação do *timeout* e do tipo de margem de segurança.

Em tempo de execução, a aplicação cliente também interage com os módulos fd_i, recebendo informações de estado sobre os componentes monitorados. As informações podem ser prestadas na forma síncrona (*requisição/resposta*) ou assíncrona (*callback*).

Cada módulo fd_i mantém um repositório de diferentes algoritmos de detecção de defeitos e cada módulo de previsão mantém um repositório de distintos algoritmos de previsão. Os repositórios são controlados por um configurador e gerente, local a cada módulo, cujo papel é coordenar a interação com o cliente e gerar flexibilidade na escolha de algoritmos de detecção e de previsão.

Responsável por realizar previsões do *timeout* com base num fluxo de amostras (*observações* do atraso de comunicação), um módulo de previsão é anexado a cada fd_i, sendo sua configuração realizada pelo respectivo fd_i. Alimentado com as observações do atraso, o módulo de previsão constrói um histórico, ou série temporal, dos atrasos de comunicação e computa, de acordo com a abordagem de previsão previamente definida, o próximo *timeout* do detector de defeitos. A *previsão* é informada ao detector sempre que uma nova observação é fornecida ou após uma *requisição* explícita.

Salienta-se que a arquitetura proposta oferece independência entre os algoritmos de detecção de defeitos e de previsão de valores. Esta independência favorece a agregação e a troca de algoritmos de detecção ou de ajuste do *timeout* separadamente, tornando a arquitetura mais flexível.

7.2.2 Novas interfaces orientadas à aplicação

Com as interfaces de objeto definidas por Felber et al., a aplicação cliente indica quais dos seus componentes devem ser monitorados, quais devem ser notificados e quais devem conter componentes ativos de detecção de defeitos (monitores), ou seja, define topologias variadas, hierárquicas ou não, para o serviço de detecção. Por outro lado, as interfaces propostas por Felber et al. restringem as possibilidades de configuração do serviço, pois definem um conjunto mínimo de métodos. A intenção foi fornecer uma *visão limitada* do serviço ao cliente de modo a esconder da aplicação detalhes funcionais dos detectores.

Quando se pretende oferecer ao cliente a liberdade de poder configurar parâmetros de grão fino no detector como, por exemplo, sua margem de segurança ou o tipo de preditor utilizado no cômputo do *timeout*, um conjunto mais amplo de métodos deve ser definido.

Para fornecer uma *visão ampla* ao serviço de detecção, ou seja, possibilitar que ele seja configurável, as interfaces de objeto orientadas à aplicação devem ser alteradas, exceto a interface `Notifiable`, a qual se restringe ao suporte do *callback*. A figura 7.3 apresenta uma arquitetura de interfaces alternativa voltada a um serviço configurável.

Na arquitetura, a interface `Monitor` continua tendo o mesmo papel: especificar a interface de um detector de defeitos para a aplicação. Entretanto, o seu conjunto de métodos foi ampliado. A nova interface `Monitor` provê à aplicação cli-

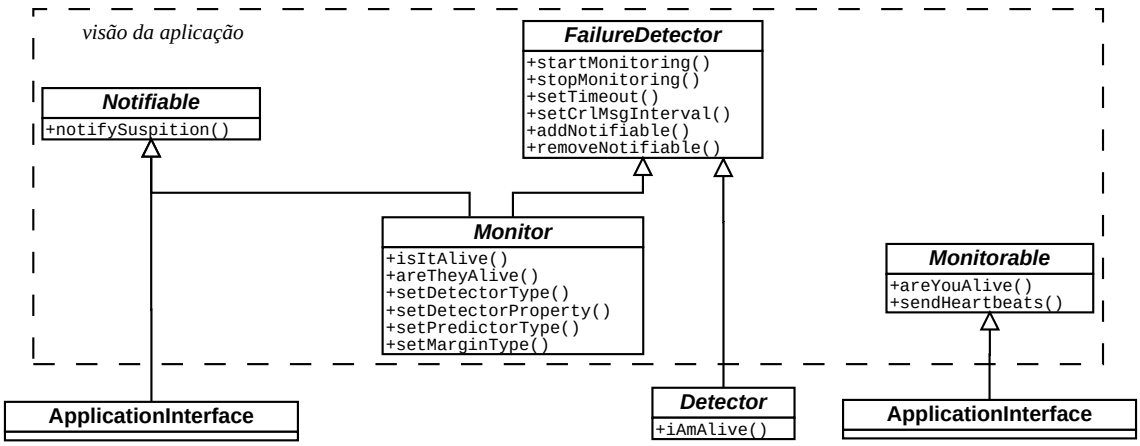

Figura 7.3: Arquitetura de interfaces para um serviço de detecção configurável.

ente um amplo conjunto de métodos que possibilitam tanto a consulta de estados quanto a configuração de parâmetros funcionais do detector. Os métodos necessários às consultas de estado são especificados na própria interface `Monitor`, enquanto que a configuração de parâmetros funcionais são herdados da interface `FailureDetector`.

Orientada às implementações dos algoritmos de detecção de defeitos que devem estar disponíveis no repositório de detectores (figura 7.2), a interface `FailureDetector` possibilita que cada detector de defeitos ofereça à aplicação a possibilidade de:

- ajustar o *timeout* para um valor fixo (método `setTimeout()`);
- ajustar o intervalo de interrogação ou o intervalo de envio de mensagens de *heartbeat* (método `setCrlMsgInterval()`);
- adicionar ou remover referências a objetos notificáveis (métodos `addNotifiable()` e `removeNotifiable()`, respectivamente); e
- indicar quais componentes da aplicação devem começar a ser monitorados e que componentes devem parar de ser monitorados (métodos `startMonitoring()` e `stopMonitoring()`, respectivamente).

Orientada à implementaçao do configurador e gerente do módulo de detecção (figura 7.2), a interface `Monitor` especializa a interface `FailureDetector` e possibilita que a aplicação cliente possa também escolher:

- o estilo de detecção (`setDetectorType()`);
- as propriedades do detector de defeitos (`setDetectorProperty()`);
- o tipo de preditor a ser utilizado (`setPredictorType()`); e
- o tipo de margem de segurança (`setMarginType()`).

Adicionalmente, a interface `Monitor` possibilita que a aplicação consulte:

- o estado de um componente específico (`isItAlive()`). Os estados possíveis de um componente são *suspeito*, *confiável* ou *desconhecido*, sendo que o estado *desconhecido* corresponde a um componente não monitorado.

- o estado de um conjunto de componentes (`areTheyAlive()`).

Observa-se que a escolha de um algoritmo de detecção dentre vários disponíveis em um repositório de algoritmos só foi possível eliminando o conjunto de diversas especializações das interfaces `Monitor` e `Monitorable` da arquitetura original proposta por Felber et al. Salienta-se que o uso de diversas especializações em nível de interface é importante quando se pensa em implementações específicas de um dado algoritmo, mas esta política dificulta a troca dinâmica, e até mesmo estática, de algoritmos contidos num repositório de algoritmos de um serviço. As interfaces utilizadas em implementações diversas seriam diferentes. Por este motivo é necessário definir interfaces genéricas que sirvam a qualquer algoritmo de detecção.

Neste sentido, a interface genérica `Monitorable` é mantida, mas os métodos das especializações (métodos voltados a componentes monitoráveis específicos) são nela agrupados. Com este agrupamento, uma implementação desta interface pode operar tanto como um monitorável *pull* quanto como um monitorável *push*, facilitando a troca do tipo de detector no serviço. Como os métodos `areYouAlive()` e `sendHeartbeats()` são dependentes do serviço, a implementação de tais métodos pode ser fornecida pelo serviço de detecção de defeitos que adotar esta arquitetura, tal como realizado na implementação de Felber et al.

Do mesmo modo, para facilitar a troca de um detector, as quatro especializações da interface `Monitor` (`HeartbeatMonitor`, `PullMonitor`, `PushMonitor` e `DualMonitor`, conforme ilustra a figura 7.1) são eliminadas. Ao invés de uma especialização para cada tipo de detector, tem-se agora uma única interface `Detector` orientada aos detectores. Com visibilidade restrita ao serviço de detecção, tal interface especifica o método `iAmAlive()`, necessário nos estilos de detecção que envolvem a informação de estado.

Fazendo a interface `Monitor` ser uma especialização das interfaces `FailureDetector` e `Notifiable`, qualquer objeto de uma classe que implemente a interface `Monitor` atua como um detector de defeitos e como um objeto notificável. Atuar como um detector de defeitos é o que a aplicação espera do gerente do módulo de detecção. Atuar como um objeto notificável é o que o serviço necessita para poder manter no gerente as informações de estado dos objetos monitorados, independente do tipo de detector selecionado, possibilitando assim a troca de detectores sem perder a informação de estado dos componentes monitorados.

7.2.3 Flexibilizando a arquitetura do serviço

Da seção anterior, tem-se que a adoção de três interfaces orientadas à aplicação torna o serviço de detecção capaz de suportar a configuração de topologias hierárquicas, ou não, entre objetos monitores, monitoráveis e notificáveis. Tem-se também que a incorporação, na interface `Monitor`, de métodos específicos para configuração do serviço habilita que a aplicação configure parâmetros de funcionamento do serviço, possibilitando uma *visão ampla* do serviço de detecção ao invés de uma *visão limitada*.

Nesta seção, mostra-se que a adoção de uma única interface de classe, orientada aos algoritmos específicos de detecção ou predição, possibilita flexibilizar a arquitetura do serviço de detecção.

Inicialmente, da figura 7.2, observa-se que tanto o módulo de detecção quanto o de previsão mantêm um repositório de algoritmos de detecção e de predição, respectivamente. Em outras palavras, cada módulo provê diferentes implementações

de uma dada classe de algoritmos, as quais devem poder ser facilmente escolhidas ou substituídas.

Do ponto de vista dos algoritmos de detecção, a interface `Detector` (figura 7.3) é quem define as interações básicas que um algoritmo de detecção deve suportar; logo, é a interface que todos os algoritmos de detecção devem implementar.

De maneira similar, do ponto de vista dos algoritmos de predição, pode-se especificar uma interface abstrata `Predictor` (figura 7.4), a qual define os métodos que todo algoritmo de previsão deve suportar: o método `Step()`, um método que serve para passar ao preditor uma nova observação do atraso de comunicação; e `getPrediction()`, um método que possibilita que o preditor seja consultado sobre sua última previsão.

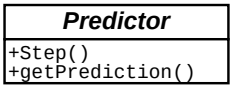

Figura 7.4: Interface abstrata `Predictor`.

O uso de apenas uma interface voltada à classe de algoritmos de um dado repositório como ocorre, por exemplo, com as interfaces `Detector` e `Predictor`, permite que o objeto gerente de um dado módulo, conhecendo-a, possa interagir com qualquer algoritmo particular do seu repositório, independentemente dos detalhes de implementação do algoritmo.

Esta flexibilização arquitetural assemelha-se ao padrão de projeto *Strategy* (GAMMA et al., 1994). Observa-se que a estrutura do padrão, apresentada na figura 7.5, é composta de diversas estratégias concretas que implementam uma única interface `Strategy`. Em outras palavras, `Strategy` é uma interface abstrata que deve ser implementada por toda e qualquer estratégia, chamada estratégia concreta (implementação de um algoritmo específico), que deseje prover a funcionalidade especificada na interface.

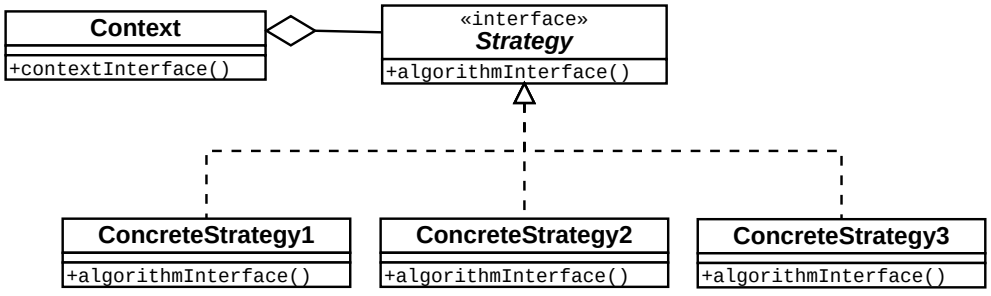

Figura 7.5: Estrutura do padrão de projeto *Strategy*.

No módulo de detecção, a interface `Detector` corresponde à interface abstrata `Strategy` e cada implementação de um detector de defeitos (algoritmo do repositório de detectores) corresponde a uma estratégia concreta. Similarmente, no módulo de previsão, a interface `Predictor` corresponde à interface `Strategy` e cada implementação do preditor corresponde a uma classe concreta. Estas correspondências possibilitam que novos algoritmos de detecção ou predição possam ser facilmente agregados ao serviço, tal como comprovado no padrão de projeto *Strategy*.

Para flexibilizar o projeto e permitir a escolha ou substituição dinâmica de qualquer das estratégias concretas, o padrão *Strategy* se utiliza de uma classe con-

texto (`Context`). Sendo a única classe que interage diretamente com o cliente, um objeto da classe contexto é responsável por determinar (a partir da solicitação do cliente, ou baseado em algum requisito) a escolha de uma dada estratégia, delegando àquela estratégia a responsabilidade pelo provimento da funcionalidade requerida pelo cliente.

Fazendo os gerentes e configuradores dos módulos de detecção e previsão serem representados pelas classes concretas `FDManager` e `PredManager`, respectivamente, e inserindo-as na arquitetura dos módulos como mostram as figuras 7.6 e 7.7, tem-se, em ambos os casos, uma correspondência com a classe `Context` do padrão *Strategy*. Deste modo, a flexibilidade para trocar algoritmos do padrão *Strategy* também é válida na arquitetura de serviço de detecção proposta.

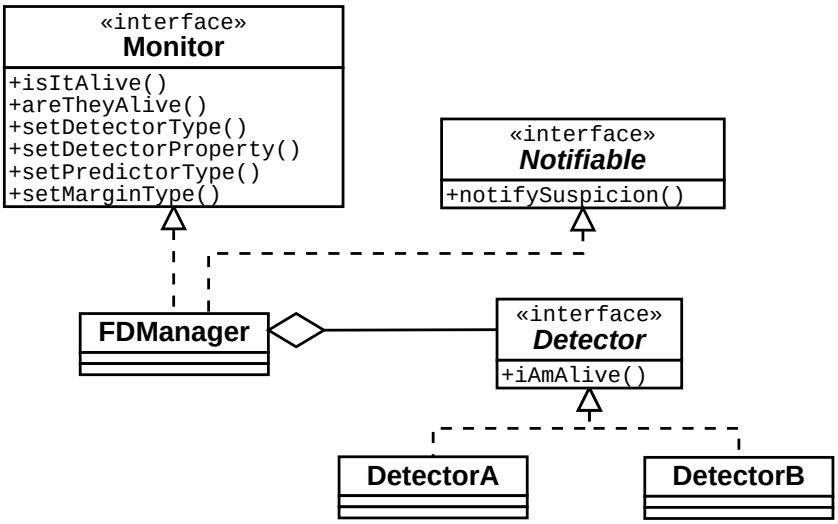

Figura 7.6: Arquitetura versátil para o módulo de detecção.

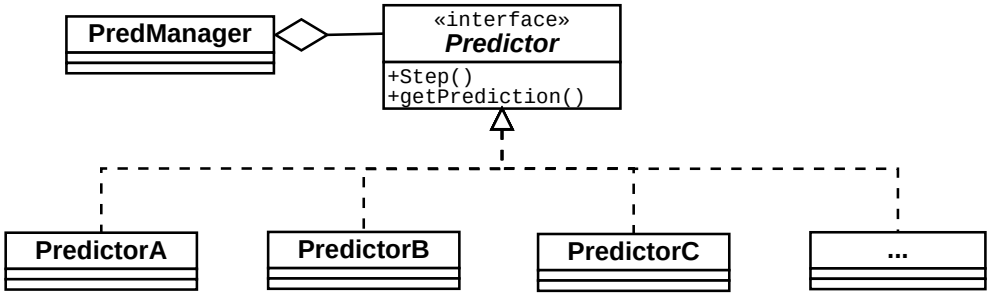

Figura 7.7: Arquitetura versátil para o módulo de previsão.

Em síntese, o uso do padrão de projeto *Strategy* permite: a inserção facilitada de novos algoritmos de detecção ou de predição; e a troca estática ou dinâmica de tais algoritmos. Para possibilitar a troca dinâmica, o estado do detector e do preditor são mantidos nos gerentes dos respectivos módulos.

7.3 AFDService - um serviço de detecção de defeitos adaptativo

O AFDService é um serviço de detecção de defeitos que monitora nós de um sistema distribuído, e que pode ser configurado para adaptar o seu *timeout* automaticamente, de acordo com o comportamento do atraso de comunicação. De acordo com o modelo de sistema distribuído definido na seção 2.1, o AFDService foi projetado para suportar defeitos do tipo colapso, omissão e de temporização. Implementado na linguagem de programação Java (SUN MICROSYSTEMS, 2003), o paradigma de comunicação é por troca de mensagens (através de *sockets* UDP).

Com o objetivo de disponibilizar os diversos mecanismos de adaptação de *timeout* abordados neste texto e possibilitar que o serviço seja configurável, o AFDService implementa a arquitetura de interfaces definida na seção anterior. Tal arquitetura permite que várias implementações de preditores possam ser facilmente incorporadas ao serviço, bem como permite que a aplicação possa escolher o preditor a ser utilizado pelo detector, tal como sugerido na arquitetura da figura 7.2. A definição de interfaces separadas, para semânticas de detecção de defeitos e de previsão de valores, também possibilita que os modelos de previsão possam ser combinados com margens de segurança distintas.

A implementação atual do AFDService disponibiliza: um detector de defeitos do estilo *pull*, sete opções de modelos preditivos (LAST, MEAN, WINMEAN, DMA, BROWN, ARIMA e LPF), e três opções de margem de segurança (fixa, variável em função do erro de previsão (ep), ou variável em função do intervalo de confiança da previsão (ic)). Salienta-se que outros estilos de detecção, bem como outras abordagens de previsão, podem ser incluídos, bastando para isto implementar as respectivas interfaces abstratas e incluir as referências na classe gerente de cada módulo.

A seguir, descreve-se a estratégia global de adaptação do *timeout* (seção 7.3.1), implementada no módulo de detecção, e descreve-se como os módulos de detecção (seção 7.3.2) e previsão (seção 7.3.3) estão organizados no AFDService.

7.3.1 Estratégia global de adaptação do *timeout*

Partindo das hipóteses de que o atraso de comunicação entre instâncias (módulos de detecção fd_i) de um algoritmo de detecção FD varia de maneira desconhecida, podendo gerar defeitos de temporização, e que mensagens podem ser perdidas, gerando defeitos de omissão, a estratégia global de adaptação do *timeout* no AFDService está estruturada em três fases: Normal, de Refutação e de Recuperação, conforme descrito a seguir.

Fase Normal

Na fase Normal de operação, caracterizada por ausência de suspeitas, o detector de defeitos pode operar com um *timeout* fixo ou variável. No caso variável, o detector ajusta o seu *timeout* dinamicamente de acordo com uma previsão do próximo *timeout*, realizada de acordo com parâmetros de configuração previamente definidos (modelo preditivo e margem de segurança). Quando o *timeout* desta fase expira, o detector de defeitos considera que pode ter havido uma perda de mensagem (defeito de omissão) ou um atraso expressivo (defeito de temporização) e, por isto,

deixa a fase Normal e vai para a fase de Refutação. Salienta-se que o objetivo desta fase é tentar evitar possíveis suspeitas decorrente de variações no atraso, e lançar suspeitas sobre componentes que provavelmente estejam em colapso.

Para manter um alto nível de qualidade de serviço, na fase Normal, o ajuste dinâmico do *timeout* utiliza o par preditor-margem de segurança configurado pelo usuário através dos métodos `setPredictorType()` e `setMarginType()`. Deste modo, nesta fase, o cômputo do *timeout* considera dois fatores: 1 - a previsão do atraso de comunicação ($\widehat{rtt}$); e 2 - uma margem de segurança (Δ).

Para prever o atraso de comunicação, o AFDService utiliza um dentre sete modelos preditivos (LAST, MEAN, WINMEAN, DMA, BROWN, ARIMA e LPF) oferecidos pelo módulo de previsão. A opção ARIMA implica que o modelo matemático utilizado pelo preditor (um polinômio) será adaptado periodicamente de acordo com o erro de previsão observado. Nas seções 5.2.1 e 5.2.3, foram apresentados, respectivamente, os detalhes sobre a periodicidade do ajuste e sobre o método de identificação automática do modelo ARIMA. As outras opções implicam que o modelo matemático de previsão escolhido não será alterado ao longo de uma dada execução.

Já a margem de segurança Δ é composta por (seção 6.4.1): 1 - uma margem β, para sobrepor variações no atraso de comunicação; e 2 - uma margem δ, para sobrepor suspeitas incorretas ou perda de mensagens.

A margem β é configurada pela aplicação através do método `setMarginType()`, definido na interface `Monitor`, e é implementada no módulo de previsão. Assim, a previsão fornecida pelo módulo correspondente resulta da soma do atraso esperado $\widehat{rtt}$ com a margem β. No AFDService, os tipos de margem β oferecidos são: um valor fixo, um valor variável de acordo com o erro de previsão (definido pela equação 6.7), ou um valor variável de acordo com o intervalo de confiança da previsão (definido pela equação 6.12). Deste modo, na fase Normal, sempre que o *timeout* for configurado para se ajustar de acordo com uma previsão do atraso, seu ajuste será dado por $t_o = \widehat{rtt} + \beta + \delta$, onde β é variável.

Por outro lado, a margem de segurança δ é fixa nesta fase. Seu valor só pode ser alterado na fase de Recuperação e depende da classe do detector de defeitos, configurada através do método `setDetectorProperty()`.

Em síntese, na fase Normal, o AFDService tenta evitar falsas suspeitas decorrentes de defeitos de temporização, ajustando dinamicamente o *timeout* de acordo com previsões realizadas pelo par preditor-margem de segurança configurado. Como o modelo de sistema distribuído admite defeitos de omissão, uma suspeita indicada pela fase Normal deve ser reconsiderada na fase de Refutação antes de ser efetivada.

Fase de Refutação

Na fase de Refutação, o detector reenvia uma mensagem de controle ao componente monitorado com a intenção de tentar refutar uma possível falsa suspeita, o que contribui para melhorar a precisão do detector. Por isto, durante esta fase, se uma mensagem de controle válida for recebida, o detector volta imediatamente à fase Normal e não gera nenhuma alteração de estado do componente monitorado. Salienta-se que o estouro do *timeout* na fase Normal pode ter acontecido por dois motivos: um defeito de omissão ou um de temporização (inclui-se neste o defeito gerado por um erro de previsão).

A tentativa de refutação se justifica porque:

- nos casos de defeitos de omissão, o *timeout* da fase Normal possivelmente será insuficiente, fazendo com que a probabilidade de perda de mensagens (cerca de 10% numa WAN (CHEN, 2000)) possa se refletir inteiramente na probabilidade de falsas suspeitas; e

- como numa WAN a probabilidade de altas variações no atraso de comunicação não é desprezível, e grandes variações dificilmente são capturadas por modelos preditivos, defeitos de temporização podem ocorrer com freqüência.

No AFDService, a tentativa de refutação é feita através de uma única verificação adicional de acessibilidade (envio de uma mensagem *Are you alive?*), tal como sugerido no estilo *dual* de detectores de defeitos (FELBER et al., 1999). Entretanto, caso o serviço seja estendido para suportar requisitos de qualidade de serviço passados pela aplicação, tal como sugerido por Chen, Toueg e Aguilera (2002), a refutação pode ser também estendida para suportar um mecanismo baseado na técnica *exponential speed-up* (seção 2.3.1).

A mensagem retransmitida nesta fase é marcada com o mesmo número de seqüência da mensagem esperada, mas contém um novo tempo de envio. Assim, a primeira mensagem que for recebida (esperada ou retransmitida) exclui o processamento da segunda e terá seu *rtt* incluído no conjunto de amostras do *rtt*.

Se o *timeout* da retransmissão também expirar, então o componente monitorado é indicado como suspeito, ou seja, inserido na lista de suspeitos (nomeada *suspectList*).

Fase de Recuperação

Para garantir a recuperação de um componente erroneamente inserido na lista de suspeitos, o componente monitor periodicamente requisita o estado de todos os componentes monitoráveis, incluindo os que pertencem a lista de suspeitos. Deste modo, caso o componente suspeito não esteja realmente defeituoso, ele irá enviar um reconhecimento positivo ao monitor num tempo finito (por hipótese, o atraso de comunicação é desconhecido mas finito), o que possibilita a recuperação do componente.

Logo, na fase de recuperação, fase em que o componente monitor recebe uma mensagem de um componente monitorável considerado suspeito, o ajuste do *timeout* corresponde ao ajuste no fator δ da margem de segurança e depende das propriedades do detector configuradas através do método `setDetectorProperty()`. Na versão atual, o AFDService implementa duas opções: classe $\diamond\mathcal{Q}$ (CHANDRA; TOUEG, 1996) e a política do *melhor esforço*. Na segunda opção, δ é previamente fixado em um valor constante e não é alterado, o que faz a margem de segurança Δ variar somente em função da componente β.

Caso o detector tenha sido configurado para operar de acordo com a classe $\diamond\mathcal{Q}$, onde é necessário garantir propriedades axiomáticas, o valor da margem de segurança δ é incrementado com um valor constante k sempre que for observada a ocorrência de uma falsa suspeita. Ou seja, a magnitude da margem δ é proporcional ao número de falsas suspeitas observadas ou ao número de execuções da fase de Recuperação.

Validação da estratégia global de adaptação

Teorema 7.1 *Seja um sistema com sincronismo parcial fraco S, onde, em toda execução, existem limites finitos sobre a velocidade relativa dos processos e sobre o atraso máximo de comunicação, mas estes limites são desconhecidos e só valem após um instante de tempo finito, também desconhecido (modelo $M3$, conforme descrito na seção 2.1). O detector* pull *implementado pelo AFDService, o qual adota a estratégia global de adaptação do* timeout *descrita acima, pertence a classe $\diamond\mathcal{Q}$ em S, ou seja, num tempo finito, ele atende às seguintes propriedades:*

Propriedade 1 - *todo componente que falha por colapso é permanentemente suspeito por pelo menos um componente correto (*weak completeness*); e*

Propriedade 2 - *haverá um momento após o qual todos os componentes corretos não serão mais erroneamente suspeitos por outros componentes corretos (*eventually strong accuracy*).*

PROVA Para demonstrar que a Propriedade 1 é válida, supõe-se que um componente q encontra-se falho, com o comportamento correspondente ao colapso. Num tempo finito, mesmo que um componente correto p envie periodicamente mensagens de requisição de estado (req) a q, o componente q não irá mais enviar mensagens de reconhecimento positivo (ack). Assim, haverá um instante de tempo finito t após o qual: a) o componente correto p irá suspeitar de q, pois o *timeout* para receber uma resposta de q é finito, ou seja, num tempo finito o componente será indicado com suspeito na fase de Refutação; e b) p não receberá mais mensagens de q após este *timeout*. Logo, a Propriedade 1 é válida.

Para demonstrar a validade da Propriedade 2, considera-se dois componentes corretos p e q e duas situações: $S1$) o *timeout* de p para q expira um número finito de vezes; e $S2$) o *timeout* de p para q expira infinitas vezes.

Na situação $S1$, o envio periódico de mensagens req faz com que haja um instante t, após a última expiração do *timeout* (o *timeout* expira um número finito de vezes), no qual q responde a p, fazendo p confiar permanentemente em q. Como resultado a situação $S1$ é válida.

Na situação $S2$, um dado componente correto alterna entre intervalos de confiança e de suspeita com relação a outro componente correto infinitas vezes. Incrementando o *timeout* por uma constante k toda vez que uma falsa suspeita for percebida (fase de Recuperação), faz com que o *timeout* cresça até um valor que sobreponha os limites de tempo finitos, mas previamente desconhecidos, do modelo de sistema distribuído. Após este momento, o *timeout* não irá mais expirar, o que implica que a situação $S2$ nunca irá ocorrer.

Deste modo, haverá um instante após o qual p não suspeitará mais de q. Logo a Propriedade 2 é válida. □

7.3.2 O módulo de detecção de defeitos

No ADFService, o módulo de detecção de defeitos é implementado como um pacote em Java nomeado `FD`. O diagrama de classes do pacote é apresentado na figura 7.8.

O cerne da arquitetura do módulo de detecção de defeitos é formado pelas interfaces `Monitor` e `Detector`. Assim, o detector de defeitos da classe `PullDetector`

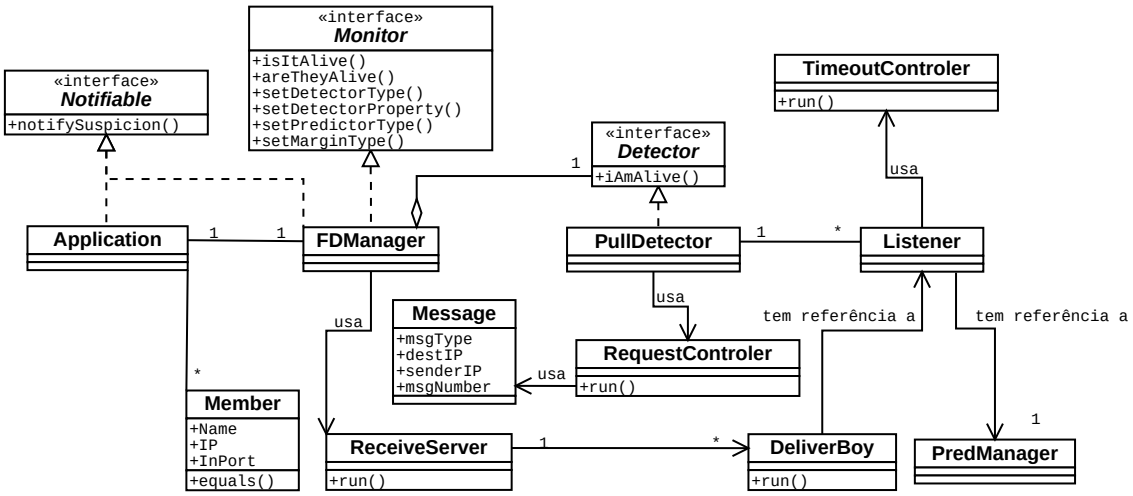

Figura 7.8: Diagrama de classes do módulo de detecção.

implementa a interface `Detector`, enquanto o gerente da estratégia de detecção, da classe `FDManager`, implementa a interface `Monitor`. Com esta estrutura, novos algoritmos de detecção podem ser adicionados ao módulo, desde que implementem a interface `Detector` e suas identificações sejam incluídas na classe `FDManager`. Para que a aplicação possa escolher (através do método `setDetector()`) uma das estratégias de detecção do repositório, a classe `FDManager` deve ter ciência de quais estratégias de detecção estão implementadas. A classe `FDManager` corresponde à classe `Context` do padrão *Strategy*.

No módulo de detecção, a identificação de objetos monitoráveis (nós do sistema distribuído) é realizada por um objeto do tipo `Member`, o qual contém os seguintes atributos: um nome, um endereço IP e uma porta de recepção de mensagens. Estes objetos devem ser instanciados pela aplicação e passados ao `FDManager` através do método `startMonitoring()`. A manutenção do estado destes objetos é realizada pelo `FDManager`, o qual mantém uma lista de membros suspeitos (*suspectList*).

Mensagens de monitoramento são encapsuladas num objeto serializável do tipo `Message`, o qual tem como atributos: o tipo e o número da mensagem de monitoramento; o endereço IP do emissor e do destinatário; e dois campos para informações específicas do protocolo de detecção. Como exemplo de uso destes, no detector do estilo *pull*, um campo é utilizado para carregar o instante de tempo do envio da mensagem de requisição de estado e o outro campo é utilizado para informar ao objeto monitorado a porta para onde o reconhecimento deve ser direcionado.

Para comunicação com os módulos de detecção remotos, o AFDService implementa uma classe servidora de recepção (`ReceiveServer`), a qual é responsável por receber as mensagens provenientes de todos os monitores e monitoráveis associados ao módulo local. Esta arquitetura favorece a troca dinâmica da estratégia de detecção pois, ao destruir um dado detector, o ponto de entrada do nó não é destruído. O problema da concorrência de acesso ao objeto `ReceiveServer` foi resolvido criando uma *thread* `DeliverBoy` (ESTEFANEL, 2000) para tratar cada mensagem que é recebida. No AFDService, a *thread* `DeliverBoy` tem duas funções: servir como objeto monitorável para um detector do estilo *pull* e servir como roteador de mensagens para as *threads* de escuta (`Listener`) do detector de defeitos.

O detector `PullDetector` implementa o algoritmo de detecção de defeitos

similar ao apresentado na seção 2.2.2, o qual adapta o *timeout* de acordo com as fases apresentadas na seção anterior. Para enviar periodicamente mensagens de solicitação de estado aos nós monitorados (regra R1), o `PullDetector` escalona um objeto do tipo `RequestControler`, a cada t_i unidades de tempo. Para controlar se o *timeout* de cada nó monitorado estoura (regra R3), o `PullDetector` instancia um objeto escuta (tipo `Listener`) para cada nó monitorado. Um dado `Listener` controla o *timeout* com o auxílio de um objeto da classe `TimeoutControler`, o qual é escalonado para atuar *timeout* instantes de tempo à frente. Caso não seja desescalonado a tempo, o `TimeoutControler` coloca o nó monitorado na lista de suspeitos realizando um *callback* ao método `notifySuspicion` da classe `FDManager`.

O comportamento do detector, quando é realizada a percepção de uma falsa suspeita (regra R5), depende da classe de detecção escolhida pela aplicação (através do método `setDetectorProperty()`). Foi pré-especificado (por *default*) que a escolha é pela classe do melhor esforço, o que faz com que nenhum valor k seja adicionado ao *timeout* quando uma falsa suspeita for percebida. Quando uma mensagem for recebida dentro do *timeout* associado (regra R4), o ajuste do *timeout* depende do tipo de preditor escolhido pela aplicação (através do método `setPredictorType()`). O `Listener` mantém uma referência para o gerente do módulo de previsão `PredManager` e transmite-lhe observações do *rtt*. Como retorno, o `PredManager` fornece previsões equivalentes ao *rtt* previsto adicionado de uma margem β.

Ao ser instanciado, um objeto da classe `FDManager` pode receber uma lista de membros a serem monitorados, ou então esta lista pode ser informada através da invocação do método `startMonitoring()`. Caso a lista esteja vazia, ou contenha apenas a identificação do nó local, o AFDService faz do nó local um objeto monitorado, ou seja, instancia somente o servidor de recepção `ReceiveServer`. Neste caso, mensagens que não forem de solicitação de estado são simplesmente ignoradas. Caso a lista de membros contenha algum nó remoto, o AFDService opera localmente como um detector de defeitos sobre aqueles membros monitoráveis, podendo também ser monitorado.

Para ser notificado de uma suspeita de maneira assíncrona, a aplicação cliente deve implementar a interface `Notifiable`. Por outro lado, a qualquer instante, a aplicação pode consultar, de maneira síncrona, o estado de algum ou de alguns membros monitorados invocando o método `isItAlive()` ou `areTheyAlive()` do objeto da classe `FDManager`.

7.3.3 O módulo de previsão

Independente do módulo de detecção, implementado como um `package` Java, o módulo de previsão, implementado como outro `package`, serve para fornecer previsões de acordo com um dentre sete possíveis modelos preditivos: ARIMA, BROWN, LAST, MEAN, WINMEAN, DMA e LPF. O módulo também possibilita que a previsão do *timeout* seja fornecida com ou sem margem de segurança. Os três tipos de margem de segurança β oferecidas são: uma margem fixa (do tipo `FixMargin`), uma margem variável em função do erro de previsão (do tipo `TCPMargin`), e uma margem variável em função do intervalo de confiança da previsão (do tipo `ICMargin`). A figura 7.9 apresenta diagrama de classes do módulo de previsão.

As classes correspondentes às implementações dos preditores são identificadas pelo nome de cada preditor e implementam a interface `Predictor`. Observa-se que

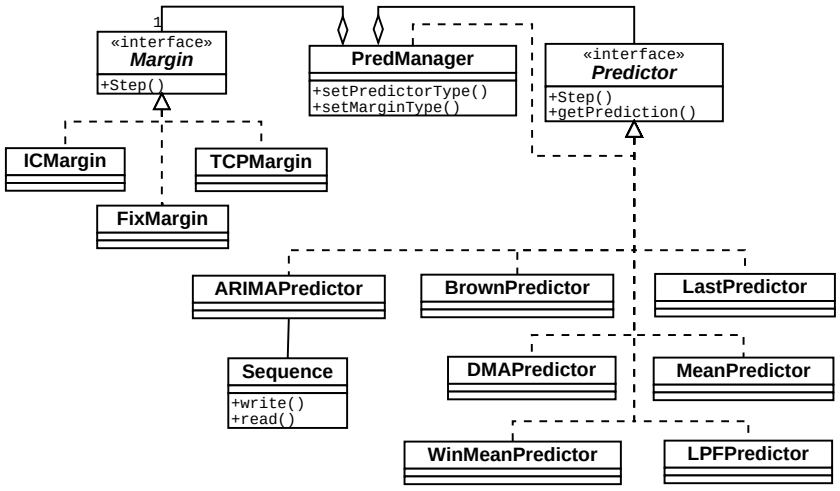

Figura 7.9: Diagrama de classes do módulo de predição.

o padrão de projeto `Strategy` é aplicado tanto na organização das classes de predição como na organização das classes de margens de segurança, sendo `PredManager` a classe responsável pela função de contexto das duas implementações do padrão *Strategy* (uma para os preditores e outra para as margens).

Para possibilitar a escolha de um preditor ou margem de segurança, a classe `PredManager` define dois métodos de configuração: o `setPredictorType` e o `setMarginType`, os quais possibilitam a configuração do tipo de preditor e de margem de segurança, respectivamente.

Mesmo que não contenha nenhum algoritmo de predição, para comportar-se como um preditor a todos os clientes do módulo de predição, a classe `PredManager` implementa a interface `Predictor`. De acordo com a implementação do padrão `Strategy`, um objeto da classe `PredManager` delega a funcionalidade de predição a um objeto concreto `Predictor`. De maneira similar, objetos da classe `PredManager` delegam a um objeto `Margin` a tarefa de computar a margem de segurança.

No AFDService, num dado nó monitor, para cada nó monitorado haverá um objeto `PredManager`, instanciado por um objeto da classe `FDManager`, o qual é alimentado com observações do atraso de comunicação com o nó monitorado. Em outras palavras, haverá uma instância do preditor por nó monitorado.

No preditor `ARIMAPredictor`, a série temporal *seq* utilizada nas previsões é mantida por um objeto `Sequence`, o qual implementa a interface de conversão de uma amostragem aperiódica numa série temporal (seção 4.1). O objeto `Sequence` mantém uma série temporal de tamanho $n = 360$, pois 360 é o melhor tamanho de série para ajustar um modelo ARIMA (seção 5.2.2). Após inserir com sucesso uma nova amostra na série temporal *seq*, o preditor ARIMA realiza uma previsão invocando o método nativo `Step()`, um método de uma biblioteca compartilhada implementada em C++ e que contém um procurador para acessar métodos do RPS. A identificação do modelo ARIMA informado ao RPS está implementada neste procurador, mas a lógica de decisão do instante de re-identificação ou reajuste do modelo está implementada no método `Step()` da classe `ARIMAPredictor`. Caso o valor não possa ser inserido na série, o preditor repete o último valor previsto.

7.4 Conclusões Parciais

Como diferentes combinações de *preditores* `x` *margem de segurança* oferecem diferentes níveis de qualidade de serviço (capítulo 6), uma aplicação cliente pode desejar utilizar um serviço de detecção de defeitos que possa ser configurado, ou estendido com novos algoritmos especializados. Neste sentido, uma arquitetura de interfaces genérica e versátil para um serviço de detecção de defeitos é proposta.

Para ser genérica, a arquitetura de interfaces é composta por três interfaces orientadas ao cliente (`Monitor`, `Notifiable` e `Monitorable`), tal como apresentado por Felber et al. (1999). Do ponto de vista do cliente, as três interfaces servem a qualquer usuário, pois detectores de defeitos (monitores) auxiliam componentes da aplicação (notificáveis) provendo informações de estado sobre outros componentes (monitoráveis).

Para ser versátil, a arquitetura de interfaces contém uma única interface orientada ao serviço (`Detector`), uma generalização da interface `Monitor`, e define métodos para uma interação de grão fino entre serviço e aplicação. Com uma única interface orientada ao serviço, todo detector deve implementar esta interface, permitindo o uso do padrão de projeto `Strategy`. O uso do padrão `Strategy` possibilita não só a incorporação fácil de novos algoritmos à arquitetura, mas também o intercâmbio dinâmico entre eles, desde que o gerente do módulo de detecção implemente a interface voltada a aplicação, a interface `Monitor`. Com métodos voltados à configurabilidade do serviço, a aplicação cliente pode ajustar a operação do detector para cenários específicos usando somente um objeto da classe `FDManager`.

A implementação do AFDService comprovou que a arquitetura para um serviço de detecção proposta neste capítulo pode ser utilizada para implementar um serviço que pode ser configurado para experimentar diversos algoritmos de previsão e de geração de margem de segurança. O AFDService permite a alteração fácil da estratégia de previsão de valores, possibilitando o uso de diferentes hipóteses a respeito da adaptação do *timeout* (hipóteses sobre o comportamento do atraso de comunicação assumidas por cada preditor). Atualmente, sete estratégias estão implementadas.

Finalmente, a arquitetura abre uma porta para a investigação de serviços de detecção que possam trocar automaticamente seus algoritmos internos, pois possibilita a manutenção de informações essenciais em gerentes de módulos. O gerente do módulo de detecção mantém o estado dos objetos monitorados e os parâmetros de configuração do serviço, enquanto o gerente de predição mantém o histórico dos atrasos e os parâmetros de configuração do preditor.

8 CONCLUSÃO

Detectores de defeitos fornecem uma importante abstração para a construção de sistemas tolerantes a falhas em ambientes assíncronos, pois encapsulam o problema da impossibilidade de distinguir entre um defeito e um expressivo atraso, gerado por um defeito de temporização num nó ou canal de comunicação. Especificados através de duas propriedades axiomáticas (abrangência e precisão), ou através de requisitos de qualidade de serviço, detectores possibilitam a validação de um acordo distribuído em ambientes sujeitos a falhas. Entretanto, para garantir a terminação do protocolo de detecção, é necessário assumir como hipótese um sistema parcialmente síncrono. Na prática, isto se reflete em implementações que costumam fazer uso de *timeouts*. No intuito de ajustar adequadamente o *timeout*, o que pode minimizar os erros dos detectores, alguns algoritmos de detecção tentam adaptar o *timeout* ao comportamento do atraso de comunicação. Para tanto, ajustam o *timeout* com base na previsão do atraso de comunicação. Neste trabalho, dissertou-se sobre o uso de modelos preditivos baseados em séries temporais no contexto de detectores de defeitos.

Considerando os dois estilos básicos de comunicação adotados em detectores de defeitos (estilos *push* e *pull*), a análise da manipulação do *timeout* (capítulo 2) mostrou que o ajuste dinâmico do *timeout* ao comportamento do atraso de comunicação é uma estratégia de ajuste aplicada somente em períodos onde o componente monitorado não está sob suspeita. Nos instantes de estouro do *timeout* (instante inicial da percepção de uma possível falsa suspeita), outras políticas de adaptação costumam ser exploradas como, por exemplo, a tentativa de refutação ou o incremento permanente do *timeout*. A análise também mostrou que o algoritmo que implementa o preditor do *timeout* pode ser dissociado do algoritmo que implementa o detector pois, quando informado de uma nova observação do atraso, o preditor não depende do detector para gerar sua previsão. Logo, em períodos de monitoramento de um componente, a capacidade de um detector auto-ajustável para alcançar uma boa qualidade de serviço depende da habilidade do módulo de previsão do *timeout*, a qual depende da habilidade do modelo preditivo utilizado (que tenta representar o comportamento do atraso de comunicação) e da margem de segurança (que tenta sobrepor adequadamente as variações no atraso).

Na busca por preditores mais eficientes do que os tradicionalmente utilizados em detectores, explorou-se o uso de modelos preditivos baseados na teoria de séries temporais, em detectores do estilo *pull*. Tal teoria necessita que a amostragem aperiódica do atraso de comunicação de ida e volta (*rtt*) seja transformada em periódica, a fim de adequar-se as suas restrições. Este trabalho mostrou (seção 4.1) que tal transformação pode ser realizada através de uma interface simples e de baixo custo computacional, possibilitando a modelagem do *rtt* como uma série temporal.

A representação de diversos traços do *rtt* como séries temporais, e a conseqüente análise da autocorrelação das séries (seção 4.2), mostra que as séries tem-

porais do *rtt* apresentam autocorrelação significativa e comportamento não estacionário na maior parte do período diurno (das 9:00h às 20:00h). Este resultado demonstra que um modelo preditivo usado para prever o comportamento do atraso de comunicação deveria considerar a hipótese de não estacionariedade e de correlação entre amostras adjacentes.

Sob hipóteses diversas para o comportamento da variável aleatória *rtt*, a análise do poder preditivo de sete preditores distintos (capítulo 5) mostrou que preditores baseados em modelos de séries temporais oferecem bons resultados, tanto em termos de precisão como em termos de velocidade de resposta à mudanças de tendência. Além disto, a análise também mostrou que os preditores baseados em termos autoregressivos (ARIMA, BROWN e LPF) são os mais precisos, principalmente quando consideram a não estacionariedade do *rtt* (ARIMA e BROWN).

Quando aplicados na determinação dinâmica do *timeout* (capítulo 6), percebe-se que a simples escolha de um preditor não garante que o detector de defeitos auto-ajustável ofereça a melhor qualidade de serviço. Para tal, a escolha do preditor deve ser realizada em conjunto com o tipo de margem de segurança. A avaliação de dois tipos de margens de segurança variáveis (em função do erro de predição - *ep*, e em função do intervalo de confiança da previsão - *ic*) mostrou que o uso de um preditor ARIMA em traços de longa duração resulta num detector de defeitos com a melhor relação *custo* `x` *benefício*. Em traços de curta duração, um preditor conservador, tal como o MEAN, quando usado em conjunto com uma margem *ep*, faz o detector oferecer relações melhores. Entretanto, mesmo em traços de curta duração, as combinações ARIMA-*ic* e BROWN-*ic* resultam em taxas de erros mais baixas a um desempenho competitivo (pouco inferior ao obtido com a combinação MEAN-*ep*). Em síntese, o uso de preditores baseados em séries temporais, em especial o preditor ARIMA, pode resultar num detector auto-ajustável com melhor qualidade de serviço.

Em termos práticos, a implementação do AFDService, um serviço que segue uma arquitetura flexível, mostrou que os preditores e margens de segurança discutidos nesta tese podem ser facilmente aglutinados, escolhidos, ou ter o seu conjunto estendido. A independência dos algoritmos de previsão de valores e de detecção de defeitos é uma característica chave na especificação de um serviço deste tipo. A troca de algoritmos e a escolha da melhor combinação preditor-margem de segurança foi possível aglutinando algoritmos de previsão na forma especificada pelo padrão de projeto *Strategy*.

Trabalhos futuros

Embora este trabalho tenha se concentrado na avaliação do uso de séries temporais para o estilo *pull* de detecção, a amostragem aperiódica do estilo de detecção *push* também pode ser modelada como uma série temporal. Como sugestão, a interface pode ser construída para uma granulosidade maior em termos de amostras, ou seja, a periodicidade da série temporal pode corresponder a um múltiplo n, maior do que dois, do período de envio de mensagens de *heartbeat* (período t_i) e as amostras da série podem corresponder a médias dos atrasos observados no período. Um mecanismo de normalização como o proposto por Chen, Toueg e Aguilera (2002) pode também ser utilizado para contabilizar somente o atraso de transmissão D. A figura 8.1 ilustra uma interface deste tipo. Salienta-se que, com esta interface, o ajuste

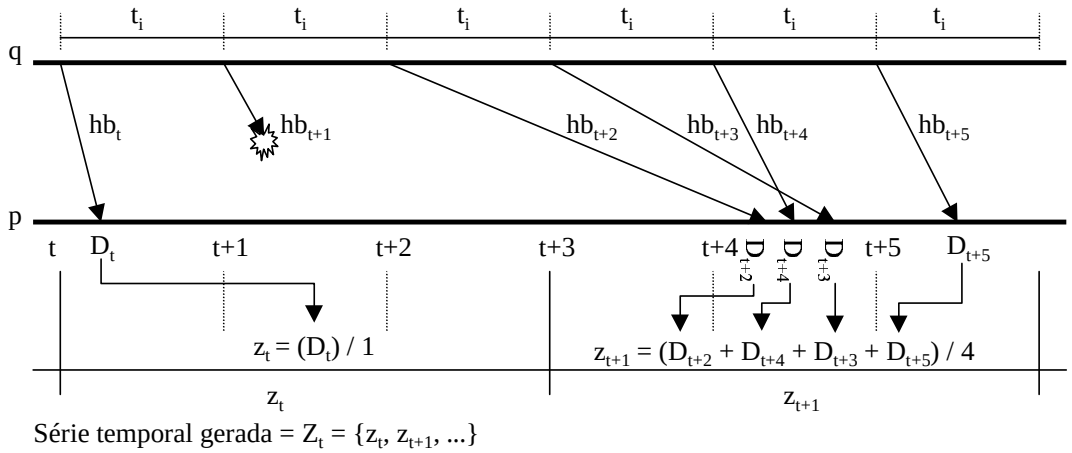

Figura 8.1: Sugestão de interface para construção de uma série temporal dos atrasos de comunicação de mensagens de *heartbeat*.

do *timeout* não deve ser realizado a cada nova amostra, mas sim a cada período de tempo $n.t_i$. A princípio, a diminuição na freqüência de ajuste do *timeout* pode até parecer estranha, mas pode trazer benefícios em relação ao custo computacional da solução. Uma solução deste tipo foi utilizada por Schultz, Hochberger e Tavangarian (2001) para previsão do desempenho de comunicação e pode ser explorada no contexto de detecção de defeitos.

Caso um detector de defeitos realize amostragens periódicas do atraso de comunicação, tal como a amostragem resultante de consultas periódicas a um agente SNMP, a série temporal pode ser diretamente construída, sem necessidade de interface de transformação junto ao detector. Neste caso, a interface de transformação de um sinal variante no tempo para um invariante no tempo está inclusa no próprio SNMP. Entretanto, neste caso, salienta-se que a taxa de atualização da informação no agente SNMP deve ser superior à taxa de solicitação da informação por parte do detector. Alguns trabalhos, tal como o discutido na seção 2.3.5, baseiam-se no SNMP e podem ser adaptados para usarem séries temporais.

Neste trabalho, a estrutura do modelo de previsão ARIMA (polinômio de previsão) foi identificado por um algoritmo baseado em regras experimentais (seção 5.2.3). Como a precisão do preditor ARIMA depende muito do modelo escolhido, outros métodos de identificação automática do modelo podem ser explorados em trabalhos futuros. Neste sentido, podem ser explorados métodos mais eficientes baseados em testes de aceitação do modelo selecionado como, por exemplo, os que consideram a análise das estatísticas de Box-Pierce e Ljung-Box (BOX; JENKINS; REINSEL, 1994). Para melhorar a precisão das previsões do atraso de comunicação também pode ser explorado o uso de métodos de combinação de previsões, tal como o método que combina previsões do modelo de alisamento exponencial com o modelo ARIMA (FERNANDES et al., 2002), entretanto, é importante salientar que a combinação de modelos preditivos tem o custo da realização de dois cálculos de previsão simultaneamente.

REFERÊNCIAS

AGUILERA, M. K.; CHEN, W.; TOUEG, S. Heartbeat: A Timeout-Free Failure Detector for Quiescent Reliable Communication. In: INTERNATIONAL WORKSHOP ON DISTRIBUTED ALGORITHMS, 11., 1997, Saarbrücken, Germany. **Proceedings...** Berlin: Springer-Verlag, 1997. p.126–140. (Lecture Notes in Computer Science, v.1320).

AGUILERA, M. K.; CHEN, W.; TOUEG, S. Failure Detector and Consensus in the Crash-Recover Model. In: INTERNATIONAL SYMPOSIUM ON DISTRIBUTED COMPUTING, DISC, 12., 1998. **Proceedings...** Berlin: Springer-Verlag, 1998. p.231–245.

ALLEN, A. O. **Probability, Statistics, and Queueing Theory with Computer Science Applications**. 2nd ed. Boston: Academic Press, 1990.

AUTOBOX. **Automatic Forecasting Systems (Autobox)**. Disponível em: <http://www.autobox.com>. Acesso em: jul. 2002.

BASU, S.; MUKHERJEE, A.; KLIVANSKY, S. Time Series Models for Internet Traffic. In: IEEE INFOCOM CONFERENCE, 1996, San Francisco. **Proceedings...** [S.l.]: IEEE Computer Society, 1996.

BAUER, P.; SICHITIU, M.; PREMARATNE, K. On the Nature of the Time-Variant Communication Delays. In: IASTED CONFERENCE MODELING, IDENTIFICATION AND CONTROL, MIC, 2001, Innsbruck, Austria. **Proceedings...** [S.l.: s.n.], 2001. p.792–797.

BERTIER, M.; MARIN, O.; SENS, P. Implementation and performance evaluation of an adaptable failure detector. In: INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS, DSN, 2002, Washington-USA. **Proceedings...** Los Alamitos: IEEE Press, 2002. p.354–363.

BOWERMAN, B. L.; O'CONNEL, R. T. **Forecasting and Time Series**: an Applied Approach. 3rd ed. Belmont, CA: Duxbury Press, 1993.

BOX, G. E. P.; JENKINS, G. M.; REINSEL, G. C. **Time Series Analysis**: Forecasting and Control. 3rd ed. New Jersey: Prentice Hall, 1994.

BRASILEIRO, F. V.; FIGUEIREDO, J. C. A. de; SAMPAIO, L. M. R. A Hierarchical Failure Detection Service with Perfect Semantics. In: WORKSHOP DE TESTES E TOLERÂNCIA A FALHAS, WTF, 3., 2002, Buzios-Brasil. **Anais...** Rio de Janeiro: UFRJ, 2002. p.25–32.

CASE, J.; FEDOR, M.; SCHOFFSTALL, M.; DAVIN, J. **A Simple Network Management Protocol (SNMP)**: RFC 1157. [S.l.]: Network Working Group, 1990.

CHANDRA, T. D.; TOUEG, S. Unreliable Failure Detectors for Reliable Distributed Systems. **Journal of the ACM**, New York, v.43, n.2, p.225–267, Mar. 1996.

CHEN, W. **On the Quality of Service of Failure Detectors**. 2000. Tese (Doutorado em Ciência da Computação) — Dep. of Computer Science, Cornell Univ., Ithaca.

CHEN, W.; TOUEG, S.; AGUILERA, M. K. On the Quality of Service of Failure Detectors. **IEEE Transactions on Computers**, Los Alamitos, v.51, n.5, p.561–580, May 2002.

CRISTIAN, F.; FETZER, C. **The timed Asynchronous System Model**. San Diego: Dept. of Computer Science and Engineering - University of California, 1997. (CSE97-519).

DINDA, P. A. **Resource Signal Prediction and Its Application to Real-time Scheduling Advisors**. 2000. Tese (Doutorado em Ciência da Computação) — School of Computer Science - Carnegie Mellon University, Pittsburgh.

DINDA, P. A.; O'HALLARON, D. R. An Evaluation of Linear Models for Host Load Prediction. In: IEEE INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING, HPDC, 8., 1999, Redondo Beach, California. **Proceedings...** [S.l.]: IEEE Computer Society, 1999.

DINDA, P. A.; O'HALLARON, D. R. **An Extensible Toolkit For Resource Prediction In Distributed Systems**. Pittsburgh: Computer Science Department - Carnegie Mellon University, 1999. (MU-CS-99-13).

DOLEV, D.; DWORK, C.; STOCKMEYER, L. On the Minimal Synchronism Needed for Distributed Consensus. **Journal of the ACM**, New York, v.34, n.1, p.77–97, Jan. 1987.

DWORK, C.; LYNCH, N.; STOCKMEYER, L. Consensus in the Presence of Partial Synchrony. **Journal of the ACM**, New York, v.35, n.2, p.288–323, Apr. 1988.

ESTEFANEL, L. A. B. **Detectores de Defeitos Não Confiáveis**. 2000. Trabalho Individual (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

FELBER, P. **The CORBA Object Group Service - A Service Approach to Object Groups in CORBA**. 1998. Tese (Doutorado em Ciência da Computação) — Départment d'Informatique, EPFL, Lausanne.

FELBER, P. **OGS Timeout**. (sobre o ajuste do timeout no OGS) [mensagem pessoal]. Mensagem recebida por <ceretta@inf.ufrgs.br> em nov. 2000.

FELBER, P.; DÉFAGO, X.; GUERRAOUI, R.; OSER, P. Failure Detectors as First Class Objects. In: INTL. SYMP. ON DISTRIBUTED OBJECTS AND APPLICATIONS, DOA, 1999. **Proceedings...** [S.l.]: IEEE Computer Society, 1999.

FERNANDES, S.; TEICHRIEB, V.; SADOK, D.; KELNER, J. Time Series Applied to Network Traffic Prediction: A Revised Approach. In: IASTED INTERNATIONAL CONFERENCE ON APPLIED MODELLING AND SIMULATION, 2002, Cambridge. **Proceedings...** Anaheim: ACTA Press, 2002.

FISCHER, M.; LYNCH, N.; PATERSON, M. Impossibility of Distributed Consensus with One Faulty Process. **Journal of the ACM**, New York, v.32, p.374–382, Apr. 1985.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns, Elements of Reusable Object-Oriented Software**. Massachusetts: Addison Wesley, 1994.

GARTNER, F. C. Fundamentals of Fault-Tolerant Distributed Computing in Asynchronous Environments. **ACM Computing Surveys**, [S.l.], v.31, n.1, Mar. 1999.

GONÇALVES, C. F. F. **Estatística**. Londrina: UEL, 2002.

GOUDA, M. G.; MCGUIRE, T. M. Accelerated Heartbeat Protocols. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 18., 1998, Amsterdam, The Netherlands. **Proceedings...** [S.l.]: IEEE Computer Society, 1998. p.202–209.

GOURIEROUX, C.; MONFORT, A. **Time Series and Dynamic Models**. [S.l.]: Cambridge University Press, 1990.

GUERRAOUI, R.; SCHIPER, A. Consensus Service: a modular approach for building agreement protocols in distributed systems. In: IEEE INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT COMPUTING, 26., 1996. **Proceedings...** [S.l.]: IEEE Computer Society, 1996. p.168–177.

GUERRAOUI, R.; SCHIPER, A. Software-Based Replication for Fault Tolerance. **IEEE Computer**, Los Alamitos, v.30, n.4, p.68–74, Apr. 1997.

HAMILTON, J. D. **Time Series Analysis**. 3rd ed. New Jersey: Princeton University Press, 1994.

JACOBSON, V. Congestion Avoidance and Control. In: ACM SYMPOSIUM ON COMMUNICATIONS, ARCHITECTURES AND PROTOCOLS, ACM SIGCOMM, 1988, Palo Alto. **Proceedings...** New York: ACM Press, 1988. p.314–329. Slightly-revised 1992 version of the 1988 paper.

JALOTE, P. **Fault-Tolerance in Distributed Systems**. New Jersey: Prentice Hall, 1994.

KESHAV, S.; LUND, C.; PHILLIPS, S.; REINGOLD, N.; SAHAN, H. An Empirical Evaluation of Virtual Circuit Holding Time Polices in IP-over-ATM Networks. **IEEE Journal of Selected Areas in Communication**, [S.l.], Oct. 1995.

LAMPORT, L. **Distribution**. (sobre sistemas distribuídos) [mensagem enviada ao DEC SRC bulletin board em 12:23:29 PDT on 28 May 1987]. Disponível em <http://research.microsoft.com/users/lamport/pubs/pubs.html>. Acesso em: jan. 2003.

LARREA, M. **Efficient Algorithms to Implement Failure Detectors and Solve Consensus in Distributed Systems**. 2000. Tese (Doutorado em Ciência da Computação) — Universidad del País Vasco, San Sebastian.

LARREA, M. Understanding Perfect Failure Detectors. In: ACM SYMPOSIUM ON PRINCIPLES OF DISTRIBUTED COMPUTING, 21., 2002, Monterey-California. **Proceedings...** New York: ACM Press, 2002. p.257–257.

LARREA, M.; ARÉVALO, S.; FERNÁNDEZ, A. Efficient Algorithms to Implement Unreliable Failure Detectors in Partially Synchonous Systems. In: INTERNATIONAL SYMPOSIUM ON DISTRIBUTED COMPUTING, DISC, 13., 1999, Bratislava, Slovak Rep. **Proceedings...** Berlin: Springer-Verlag, 1999. p.34–48. (Lecture Notes in Computer Science, v.1693).

MACÊDO, R. **Implementing Failure Detection through the use of a Self-Tuned Time Connectivity Indicator**. Salvador-Brazil: Laboratório de Sistemas Distribuídos - LaSiD, 1998. (RT008/98).

MACÊDO, R. Failure Detection in Asynchronous Distributed Systems. In: WORKSHOP DE TESTES E TOLERÂNCIA A FALHAS, WTF, 2., 2000, Curitiba-Brazil. **Anais...** Curitiba: UFPR, 2000. p.76–81.

MONTGOMERY, D. C.; JOHNSON, L. A. **Forecasting and time series analysis**. New York: McGraw-Hill, 1976.

MONTRESOR, A. **System Support for Programming Object-Oriented Dependable Applications in Partitionable Systems**. 2000. Tese (Doutorado em Ciência da Computação) — Department of Computer Science, University of Bologna, Bologna.

MORETTIN, P. A.; CASTRO TOLOI, C. M. de. **Modelos para Previsão de Séries Temporais**. Rio de Janeiro, Brasil: Instituto de Matemática Pura e Aplicada, 1981.

MORETTIN, P. A.; CASTRO TOLOI, C. M. de. **Previsão de Séries Temporais**. São Paulo, Brasil: Atual, 1985.

NETO, F. W.; MADEIRA, E. R. M. Aplicando a Técnica de Gerenciamento Pró-Ativo de Redes de Computadores. In: SIMPÓSIO DE REDES DE COMPUTADORES, 16., 1998. **Anais...** Rio de Janeiro: UFF, 1998. p.71–90.

NUNES, R. C.; JANSCH-PÔRTO, I. Modelling Communication Delays in Distributed Systems using Time Series. In: IEEE SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS, SRDS, 21., 2002, Osaka-Japan. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.268–273.

NUNES, R. C.; JANSCH-PÔRTO, I. Non-stationary Communication Delays in Failure Detectors. In: IEEE LATIN-AMERICAN TEST WORKSHOP, LATW, 3., 2002, Montevideo-Uruguay. **Proceedings...** Amissville: IEEE Computer Society, 2002. p.16–21.

NUNES, R. C.; JANSCH-PÔRTO, I. A Lightweight Interface to Predict Communication Delays using Time Series. In: LATIN-AMERICAN SYMPOSIUM ON DEPENDABLE COMPUTING, LADC, 1., 2003, São Paulo. **Proceedings...** Berlin: Springer-Verlag, 2003.

OCHI, M. K. **Applied Probability and Stochastic Processes in Engineering and Physical Sciences**. New York: John Wiley & Sons, 1990.

POWELL, D. Distributed Fault-Tolerance: A Short Tutorial. In: INTERNATIONAL WORKSHOP ON DEPENDABLE COMPUTING AND ITS APPLICATIONS, 1998, Johannesburg - South Africa. **Proceedings...** [S.l.: s.n.], 1998.

PRADHAN, D. K. **Fault-Tolerant Computer System Design**. New Jersey: Prentice Hall, 1996.

RAYNAL, M.; TRONEL, F. Group Membership Failure Detecton: a Simple Protocol and its Probabilistic Analysis. **Distributed Systems Engineering Journal**, [S.l.], v.6, n.3, p.95–102, 1999.

REDHAT. **Linux, Embedded Linux and Open Source Solutions**. Disponível em: <http://www.redhat.com>. Acesso em: jul. 2002.

RENESSE, R. van; MINSKY, Y.; HAYDEN, M. A Gossip-Style Failure Detection Service. In: MIDDLEWARE CONFERENCE: IFIP INTERNATIONAL CONFERENCE ON DISTRIBUTED SYSTEMS PLATFORMS AND OPEN DISTRIBUTED PROCESSING, 1998. **Proceedings...** Berlin: Springer Verlag, 1998. p.55–70.

SCHIPER, A. Early Consensus in an Asynchronous System with a Weak Failure Detector. **Distributed Computing**, Berlin, v.10, n.3, p.149–157, Apr. 1997.

MULLENDER, S. (Ed.). **What Good are Models and What Models are Good?** 2nd ed. [S.l.]: Addison-Wesley, 1993. p.17–26.

SCHULZ, J.; HOCHBERGER, C.; TAVANGARIAN, D. Prediction of Communication Performance for Wide Area Computing Systems. In: EUROMICRO WORKSHOP ON PARALLEL AND DISTRIBUTED PROCESSING, 9., 2001, Mantova-Italy. **Proceedings...** [S.l.]: IEEE Computer Society, 2001. p.480–486.

SCIENTIFIC COMPUTING ASSOCIATES. **SCA System**. Disponível em: <http://www.scausa.com>. Acesso em: jun. 2001.

SERGENT, N.; DÉFAGO, X.; SCHIPER, A. **Failure Detectors**: implementation issues and impact on consensus performance. Lausanne: École Polytechnique Fédérale de Lausanne - EPFL, 1999. (Technical Report SSC/1999/019).

SOUZA, R. C.; CAMARGO, M. E. **Análise e Previsão de Séries Temporais**: os Modelos ARIMA. Ijuí: Sedigraf, 1996.

SUN MICROSYSTEMS. **The Source for Java Technology**. Disponível em: <http://java.sun.com>. Acesso em: jul. 2003.

RYAN, K. F.; GILES, D. E. A. **Testing for Unit Roots With Missing Observations**. [S.l.]: JAI Press, 1998. p.203–242.

VASCONCELLOS, M. A. S.; ALVES, D. **Manual de Econometria**. São Paulo: Atlas, 2000.

WERKEMA, M. C. C. **Análise de Regressão**: como Entender o Relacionamento entre as Variáveis de um Processo. Belo Horizonte: Fundação Christiano Ottoni, 1996.

WINTER, E. M. W. **Probabilidade e Inferência para Bioestatística**. Curitiba: Departamento de Estatística, UFPR, 2002. Disponível em: <http://www.est.ufpr.br/ mario/material>. Acesso em: dez. 2002.