Qualidade de Serviço de Detectores de Defeitos na Presença de Rajadas de Perdas de Mensagens

Irineu Sotoma

Tese de Doutorado

Instituto de Computação
Universidade Estadual de Campinas

Qualidade de Serviço de Detectores de Defeitos na Presença de Rajadas de Perdas de Mensagens

Irineu Sotoma

29 de setembro de 2006

Banca Examinadora:

- Prof. Dr. Edmundo Roberto Mauro Madeira (Orientador)
- Prof. Dr. Joni da Silva Fraga
  Departamento de Automação e Sistemas (DAS-UFSC)
- Prof. Dr. Elias Procópio Duarte Júnior
  Departamento de Informática (DI-UFPR)
- Prof. Dr. Luiz Eduardo Buzato
  Instituto de Computação (IC-UNICAMP)
- Profa. Dra. Islene Calciolari Garcia
  Instituto de Computação (IC-UNICAMP)
- Profa. Dra. Cecília Mary Fischer Rubira (Suplente)
  Instituto de Computação (IC-UNICAMP)
- Prof. Dr. Maurício Ferreira Magalhães (Suplente)
  Faculdade de Engenharia Elétrica e de Computação (FEEC-UNICAMP)

E BC
MADA IT-UNICAMP
So78q
EX
BC/ 70923
16.123-06
D X
11,00
12/12/06
393721

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Bibliotecária: Miriam Cristina Alves – CRB8a / 5094

Sotoma, Irineu.

So78q Qualidade de serviço de detectores de defeitos na presença de rajadas de perdas de mensagens / Irineu Sotoma -- Campinas, [S.P.: s.n.], 2006.

Orientador: Edmundo Roberto Mauro Madeira.

Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Processamento distribuído. 2. Tolerância à falha (Computação). 3. Markov, Processos de. I. Madeira, Edmundo Roberto Mauro. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Quality of service of failure detectors in the presence of message loss bursts

Palavras-chave em inglês (Keywords): 1. Distributed processing. 2. Fault tolerance. 3. Markov processes.

Área de concentração: Sistemas distribuídos.

Titulação: Doutor em Ciência da Computação

Banca examinadora: Prof. Dr. Edmundo Roberto Mauro Madeira (IC-UNICAMP)
Prof. Dr. Joni da Silva Fraga (DAS-UFSC)
Prof. Dr. Elias Procópio Duarte Júnior (DI-UFPR)
Prof. Dr. Luiz Eduardo Buzato (IC-UNICAMP)
Profa. Dra. Islene Calciolari Garcia (IC-UNICAMP)

Data da defesa: 29/09/2006

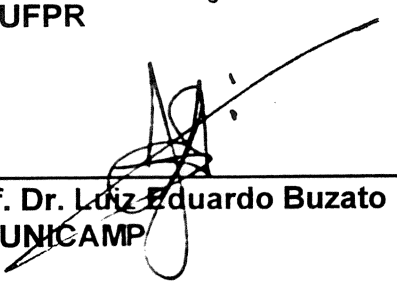
Programa de Pós-Graduação: Doutorado em Ciência da Computação

TERMO DE APROVAÇÃO

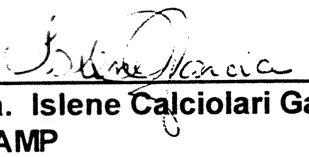
Tese defendida e aprovada em 29 de setembro de 2006, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Joni da Silva Fraga
DAS - UFSC

Prof. Dr. Elias Procópio Duarte Júnior
DI - UFPR



Prof. Dr. Luiz Eduardo Buzato
IC - UNICAMP



Profa. Dra. Islene Calciolari Garcia
IC - UNICAMP

Prof. Dr. Edmundo Roberto Mauro Madeira
IC - UNICAMP

Qualidade de Serviço de Detectores de Defeitos na Presença de Rajadas de Perdas de Mensagens

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Irineu Sotoma e aprovada pela Banca Examinadora.

Campinas, 29 de setembro de 2006.

Prof. Dr. Edmundo Roberto Mauro Madeira
(Orientador)

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

© Irineu Sotoma, 2006.
Todos os direitos reservados.

À Rosimeire, por seu amor, apoio e paciência que me ajudam a progredir.
À Lara e ao Pedro, pelas alegrias que me fazem esquecer as dificuldades.

Agradecimentos

A Deus, Criador, Redentor, Inspirador, que sempre me acompanha tanto nos momentos alegres, quanto nos mais difíceis, me acolhendo e dando forças e inspiração para superar os obstáculos.

Aos meus pais José e Amélia, que me deram a vida e a oportunidade de uma boa educação.

À minha esposa Rosimeire, que sempre esteve do meu lado, com muito amor.

Aos meus filhos, Lara e Pedro, que compreenderam os meus períodos de ausência.

Aos meus familiares de coração e de sangue, pelo apoio em toda a minha caminhada.

Ao meu orientador, Prof. Dr. Edmundo Roberto Mauro Madeira, pela paciência, disponibilidade, orientação e apoio nas decisões.

Aos membros da banda examinadora, Prof. Dr. Edmundo Roberto Mauro Madeira, Prof. Dr. Joni da Silva Fraga, Prof. Dr. Elias Procópio Duarte Júnior, Prof. Dr. Luiz Eduardo Buzato, Profa. Dra. Islene Calciolari Garcia, pela leitura do texto da tese, pelas sugestões, e pelas críticas construtivas, que possibilitaram melhorar bastante o texto final da tese.

Aos administradores da rede do IC-2, Simone, Camila, Carlos e Roland, aos analistas de suporte Furlan e Éverton, pelo suporte ágil na resolução de problemas, e aos demais funcionários do IC-UNICAMP, pelo apoio técnico e administrativo.

Ao Prof. Dr. Jorge Stolfi pelas valiosas discussões sobre modelagem probabilística, à Profa. Dra. Nancy Lopes Garcia, pelo apoio técnico em Cadeias de Markov e outras questões de Probabilidade, e ao Prof. Dr. Raul Ceretta Nunes, pela disponibilização dos seus traces sobre alguns links nacionais.

Ao Nielsen e ao Gilberto, pela devida configuração dos computadores do LIS usados nas primeiras simulações.

Ao Carlos Augusto Froldi, que me sugeriu o uso do rdate, no lugar de NTP, na coleta de dados de Campo Grande a Campinas.

Aos membros da banca examinadora do meu exame de qualificação específico, Prof. Dr. Luiz Eduardo Buzato e Prof. Dr. Joni da Silva Fraga, pelas valiosas sugestões que me fizeram mudar o encaminhamento da tese.

Aos professores e colegas Edson Cáceres, Marcelo Henriques Carvalho, Nalvo, Marcelo Turine, Beto, e Edna, pelo apoio nas etapas finais deste doutorado.

Aos colegas do IC, à turma do bandejão, e em particular aos meus colegas da sala 92 de doutorado: Borin, Adílson, Cláudio, Fernando Castor, Fernando Menezes, Gregório, Daniel, pelo companheirismo e pelos momentos de descontração.

Ao Adílson, Fábio Verdi, Borin, Evandro, e Daniel por me oferecerem pousada nas minhas vindas ao IC.

À FAPESP (processo número 00/05369-2) e ao CNPq (processo número 62.0123/04-4), pelo apoio financeiro.

A todos aqueles que não cito nestes agradecimentos, mas que de alguma forma me ajudaram a concluir este doutorado.

Resumo

A Qualidade de Serviço (QoS) de detectores de defeitos determina a rapidez que um detector de defeitos q detecta a quebra de um processo p, e a precisão que q informa essa quebra. Em redes de longa distância e em redes sem fio, a ocorrência de quebras de processo, altas variações de atraso e perdas de pacotes em rajadas são comuns. Nestas condições, uma escolha adequada de parâmetros, por um configurador de detectores de defeitos, para manter o detector de defeitos satisfazendo os requisitos de QoS, é requerida.

Por isso, este trabalho propõe um configurador de detector de defeitos que leva em conta a distribuição de probabilidade de comprimento de rajadas de perdas de pacotes de mensagem, através do uso de um modelo de Markov. Os resultados da simulação mostram que os parâmetros fornecidos pelo configurador proposto tendem a levar o detector de defeitos a satisfazer os requisitos de QoS em redes sujeitas a rajadas de perdas. Adicionalmente, a pesquisa mostra que é possível melhorar a precisão do detector de defeitos usando uma combinação de estimadores simples de atrasos de mensagens.

Abstract

The quality of service (QoS) of failure detectors determines how fast a failure detector q detects the crash of a process p, and how accurate q informs the p crash. In wide area networks and wireless networks, the occurrence of process crashes, high delay variations and burst losses in message packets are common. In these conditions, an adequate choice in the failure detector parameters, by a failure detector configurator, to keep the failure detector satisfying the QoS requirements, is required.

Therefore, this work proposes a failure detector configurator which takes into account the probability distribution of loss burst lengths of message packets, by using a Markov model. The simulation results show that the parameters provided by the proposed configurator tend to lead the failure detector to satisfy the QoS requirements in networks subject to message loss bursts. Additionally, the work shows that is possible improve the accuracy of the failure detector by using a simple combination of simple message delay estimators.

Sumário

Agradecimentos **viii**

Resumo **x**

Abstract **xi**

1 Introdução **1**
1.1 Contexto do Problema 1
1.2 Descrição do Problema 3
1.3 Descrição Sucinta da Solução 3
1.4 Contribuições do Trabalho 4
1.5 Organização do Texto 5

2 Conceitos Fundamentais **7**
2.1 Dependabilidade de Sistemas 7
2.1.1 Atributos de Dependabilidade 8
2.1.2 Ameaças à Dependabilidade 8
2.1.3 Meios para a Obtenção de Dependabilidade 9
2.2 Sistemas Distribuídos 10
2.2.1 Propriedades de Sistemas Distribuídos 11
2.2.2 Modelos de Sistemas Distribuídos 12
2.2.3 Componentes de Sistemas Distribuídos 13
2.3 Canais de Comunicação e Comunicação Confiável e Não Confiável 14
2.3.1 Modelos Básicos para a Medição de Atrasos e Perdas de Pacotes . . 15
2.3.2 Modelagem de Atrasos 15
2.3.3 Modelagem de Perdas 15
2.4 Consenso Distribuído 17
2.4.1 Definição de Consenso 17
2.4.2 Resultados de Impossibilidade de Consenso Distribuído 18
2.5 Detectores de Defeitos 19

2.5.1 Definição de Detectores de Defeitos não Confiáveis 19
2.5.2 Propriedades de Detectores de Defeitos não Confiáveis 20
2.5.3 Detectores de Defeitos e Consenso Distribuído 21
2.5.4 Detectores de Defeitos Adaptativos 25
2.5.5 Qualidade de Serviço de Detectores de Defeitos 26
2.6 Conclusões 27

3 Qualidade de Serviço de Detectores de Defeitos 29
3.1 O Trabalho de Chen et al 29
3.1.1 Modelo de Sistema 29
3.1.2 Modelo Probabilístico da Rede 30
3.1.3 Saída do Detector de Defeitos e Transições 30
3.1.4 Métricas para a Qualidade de Serviço 30
3.1.5 Relacionamento entre as Métricas de QoS 31
3.1.6 Algoritmo NFD-S 32
3.1.7 Análise Básica de QoS de NFD-S 32
3.1.8 Configurador para NFD-S 35
3.1.9 Lidando com Comportamento de Mensagens Desconhecido 36
3.1.10 Lidando com Comportamento de Mensagens Desconhecido e Relógios não Sincronizados 38
3.1.11 Configurador para NFD-U 39
3.2 Trabalhos Relacionados 41
3.2.1 Trabalhos de Bertier 41
3.2.2 Trabalhos de Nunes e Jansch-Pôrto e de Falai e Bondavalli 42
3.2.3 Trabalhos de Hayashibara 44
3.3 Conclusões 44

4 QoS de Detectores de Defeitos na Presença de Rajadas de Perdas 47
4.1 Hipóteses da Proposta 47
4.1.1 Modelo de Sistema 47
4.1.2 Modelo Probabilístico Modificado para a Rede 48
4.1.3 Métricas, Relacionamento entre as Métricas, Algoritmos NFD-S e NFD-U 48
4.2 Modelo de Markov para as Rajadas de Perdas 48
4.3 Modelo Alterado para a Análise de QoS de NFD-S 51
4.4 Modelo do Configurador para NFD-S 58
4.5 Otimização do Configurador para NFD-S 63
4.6 Distribuição de Atrasos Desconhecida e NFD-S 64
4.7 Modelo do Configurador para NFD-U 69

4.8 Conclusões 73

5 Avaliação da Proposta **75**
5.1 Visão Geral do Simulador 75
5.1.1 Geração dos Dados de Atrasos e Perdas 76
5.1.2 Geração dos Valores Analíticos 76
5.1.3 Simulação Usando os Arquivos de Dados 76
5.1.4 Geração dos Gráficos dos Resultados 77
5.2 Simulação com Dados Gerados 77
5.2.1 Configuração da Simulação com Dados Gerados 77
5.2.2 Cálculo das Probabilidades de Rajadas de Perdas 78
5.2.3 Análise dos Intervalos de Recorrência de Erros 79
5.2.4 Análise de Intervalos de Duração de Erros 82
5.2.5 Discussão Geral sobre a Simulação com Dados Gerados 85
5.2.6 Comparação entre o Modelo com Espaço de Estados Limitado e o Não Limitado 86
5.3 Simulação com Dados Coletados 87
5.3.1 Configuração da Simulação com Dados Coletados 88
5.3.2 Análise do Tempo de Detecção 88
5.3.3 Análise dos Intervalos de Recorrência de Erros 90
5.3.4 Análise de Intervalos de Duração de Erros 94
5.3.5 Discussão Geral sobre a Simulação com Dados Coletados 101
5.4 Conclusões 102

6 Melhorando a Precisão **105**
6.1 Um Método Simples para Melhorar a Precisão 105
6.2 Avaliação do Estimador Composto 106
6.2.1 Análise do Tempo de Detecção 106
6.2.2 Análise dos Intervalos de Recorrência de Erros 107
6.2.3 Análise de Intervalos de Duração de Erros 109
6.2.4 Discussão Geral sobre MEAN-WINMEAN4 113
6.3 Conclusões 113

7 Conclusão **115**
7.1 Contribuições da Tese 116
7.2 Dificuldades Encontradas 117
7.3 Trabalhos Futuros 117

Referências Bibliográficas **119**

Lista de Figuras

2.1 Modelo de Sanneck com espaço de estados ilimitado. 17

3.1 Algoritmo NFD-S com parâmetros η e δ (relógios sincronizados). 33
3.2 Configuração de NFD-S. 36
3.3 Algoritmo NFD-U com parâmetros η e α (relógios não sincronizados, mas EA_is são conhecidos). 38
3.4 Configuração de NFD-U. 40
3.5 Configuração de NFD-E. 41

4.1 Ilustração para a Definição 5 quando $k = 2$. 52
4.2 Um exemplo para a Proposição 6.3 quando $k = 3$. 54
4.3 Configuração de NFD-S por NSM-NFD-S. 62
4.4 Configuração de NFD-U por NSM-NFD-U. 72
4.5 Configuração de NFD-E por NSM-NFD-E. 73

5.1 Verificação de $E(T_{MR}) \geq T_{MR}^L$ para distribuição geométrica e $p_L = 0,01$. . 80
5.2 Verificação de $E(T_{MR}) \geq T_{MR}^L$ para distribuição geométrica e $p_L = 0,03$. . 80
5.3 Verificação de $E(T_{MR}) \geq T_{MR}^L$ para distribuição Pareto e $p_L = 0,01$. 81
5.4 Verificação de $E(T_{MR}) \geq T_{MR}^L$ para distribuição Pareto e $p_L = 0,03$. 82
5.5 Verificação de $E(T_M) \leq T_M^U$ para distribuição geométrica e $p_L = 0,01$. . . . 83
5.6 Verificação de $E(T_M) \leq T_M^U$ para distribuição geométrica e $p_L = 0,03$. . . . 83
5.7 Verificação de $E(T_M) \leq T_M^U$ para distribuição Pareto e $p_L = 0,01$. 84
5.8 Verificação de $E(T_M) \leq T_M^U$ para distribuição Pareto e $p_L = 0,03$. 85
5.9 Comparação de T_{MR} com $p_L = 0,01$. 87
5.10 Comparação de T_M com $p_L = 0,01$. 87
5.11 Tempo de detecção atendido por NFD-E ao usar o estimador LAST. 89
5.12 Tempo de detecção atendido por NFD-E com o estimador MEAN. 90
5.13 Tempo de detecção não atendido por NFD-E com WINMEAN32. 90
5.14 T_{MR}^L não atendido pelo configurador Chen-NFD-E usando NFD-E. 91
5.15 Comportamento de NFD-E em forma de escada. 92
5.16 Comportamento de NFD-E em forma de escada. 92

5.17 Comportamento de NFD-E mais próximo aos valores analíticos. 93
5.18 Comportamento de NFD-E mais próximo aos valores analíticos. 93
5.19 Comportamento de NFD-E mais próximo aos valores analíticos. 94
5.20 T_M^U não é atendido para valores pequenos de T_D^U. 95
5.21 T_M^U é melhor atendido para valores pequenos de T_D^U. 95
5.22 T_M^U não é atendido para valores pequenos de T_D^U. 96
5.23 T_M^U não é atendido para valores pequenos de T_D^U. 96
5.24 T_M^U é melhor atendido para valores pequenos de T_D^U. 97
5.25 T_M^U é melhor atendido para valores pequenos de T_D^U. 97
5.26 T_M^U não é atendido principalmente para valores pequenos de T_D^U. 98
5.27 T_M^U não é atendido principalmente para valores pequenos de T_D^U. 98
5.28 T_M^U é melhor atendido para valores pequenos de T_D^U. 99
5.29 T_M^U é melhor atendido para valores pequenos de T_D^U. 99
5.30 T_M é ruim com WINMEAN256. 100
5.31 T_M é ruim com WINMEAN256. 100
5.32 T_M é influenciado pela variância e por h. 101
5.33 T_M é influenciado pela variância e por h. 101

6.1 O tempo de detecção máximo atendido por NFD-E, para o dia 08 de dezembro ($p_L = 0,030864$ e $h = 468$). 107
6.2 T_{MR}^L atendido melhor por NFD-E usando MEAN-WINMEAN4, para o dia 09 de dezembro ($p_L = 0,00072$ e $h = 1$). 108
6.3 T_{MR}^L atendido melhor por NFD-E usando MEAN-WINMEAN4, para o dia 04 de dezembro ($p_L = 0,007164$ e $h = 12$). 108
6.4 T_{MR}^L atendido melhor por NFD-E usando MEAN-WINMEAN4, para o dia 08 de dezembro ($p_L = 0,030864$ e $h = 468$). 109
6.5 T_M fornecido por NFD-E é similar com WINMEAN4 e MEAN-WINMEAN4, para o dia 09 de dezembro ($p_L = 0,00072$ e $h = 1$). 110
6.6 T_M fornecido por NFD-E é melhor com MEAN-WINMEAN4, para o dia 09 de dezembro ($p_L = 0,00072$ e $h = 1$). 110
6.7 T_M fornecido por NFD-E é similar com WINMEAN4 e MEAN-WINMEAN4, para o dia 04 de dezembro ($p_L = 0,007164$ e $h = 12$). 111
6.8 T_M fornecido por NFD-E é melhor com MEAN-WINMEAN4, para o dia 04 de dezembro ($p_L = 0,007164$ e $h = 12$). 111

6.9 T_M fornecido por NFD-E é similar com WINMEAN4 e MEAN-WINMEAN4, para o dia 08 de dezembro ($p_L = 0,030864$ e $h = 468$). 112
6.10 T_M fornecido por NFD-E é melhor com MEAN-WINMEAN4, para o dia 08 de dezembro ($p_L = 0,030864$ e $h = 468$). 112

Lista de Tabelas

4.1 Cálculo das probabilidades da cadeia de Markov. 50

Lista de Acrônimos

NFD-S	*New Failure Detector with Synchronized Clocks*
NFD-U	*New Failure Detector with Unsynchronized Clocks and known expected arrival times of heartbeats*
NFD-E	*New Failure Detector with Unsynchronized Clocks and Estimates of expected arrival times of heartbeats*
NTP	*Network Time Protocol*
FLP	Resultado de impossibilidade de Fischer, Lynch e Paterson [50]
Chen-NFD-S	Configurador de Chen et al para NFD-S
Chen-NFD-U	Configurador de Chen et al para NFD-U
Chen-NFD-E	Configurador de Chen et al para NFD-E
NSM-NFD-S	Novo configurador de Sotoma e Madeira para NFD-S
NSM-NFD-U	Novo configurador de Sotoma e Madeira para NFD-U
NSM-NFD-E	Novo configurador de Sotoma e Madeira para NFD-E
LAST	Estimador de atraso que usa apenas o último *heartbeat* recebido
MEAN	Estimador de atraso que usa todas os *heartbeats* recebidos
WINMEANn	Estimador de atraso que usa apenas os últimos n*heartbeats* recebidos

Capítulo 1

Introdução

Este capítulo contextualiza o problema abordado nesta pesquisa e o apresenta, fornece uma visão sucinta da solução proposta e das contribuições do trabalho, e mostra a organização do texto. Atualmente, na literatura em língua portuguesa, não há consenso sobre a terminologia associada à dependabilidade[1]. Por isso, colocamos os termos da língua inglesa em notas de rodapé, pelo menos na primeira vez que ocorrem no texto.

1.1 Contexto do Problema

Em redes de longa distância e em redes sem fio, a quebra de processos e a perda de pacotes em rajadas ocorrem com freqüência. Neste trabalho, o termo perdas em rajadas refere-se a perdas consecutivas de pacotes. Um processo pode quebrar, por exemplo, devido a ataques de negação de serviço, quedas de energia, problemas com o hardware do computador em que o processo está executando, e até mesmo problemas devido a falhas de projeto e implementação do sistema operacional ou do programa aplicativo. A perda de pacotes em rajadas é comum, no caso de redes de longa distância, devido a problemas de congestionamento, que levam os roteadores a descartarem pacotes, e no caso de redes sem fio, devido a ruídos no ambiente que interconecta os processos.

Na área de tolerância a falhas, os conceitos de falha, erro e defeito são bem distintos[2]. *Falha* é uma característica que pode levar um sistema a não fornecer serviço de acordo com a funcionalidade especificada. A falha pode causar um *erro*, ou seja, um desvio em algum estado interno de algum componente do sistema. Se o erro for suficiente para

[1] Do inglês: *dependability*.

[2] Traduzimos *fault*, *error*, e *failure*, respectivamente como falha, erro e defeito.

que um componente leve o sistema a fornecer um serviço fora da sua especificação, então temos um *defeito*. Contudo, em um sistema com vários componentes, um defeito em um componente A pode ser considerado uma falha para um componente B que usa o serviço falho de A [16]. Portanto, neste trabalho os termos falha e defeito serão usados de forma intercambiável, exceto quando for conveniente diferenciá-los.

Uma forma bastante comum de detectar se um processo quebrou é o uso de detectores de defeitos, que são componentes de sistema que frequentemente monitoram o estado de outros componentes. Os componentes monitoráveis podem pertencer ao mesmo sistema ou a outros. O objetivo dos detectores de defeitos geralmente é informar rapidamente e com precisão sobre a ocorrência de defeitos em processos (processadores), sob canais de comunicação confiáveis e não confiáveis. As áreas de redes de computadores e sistemas distribuídos possuem vários exemplos que usam detectores de defeitos.

A área de redes de computadores usa temporizadores, como uma espécie de detectores de defeitos, e a utilização básica de temporizadores está presente na retransmissão de mensagens TCP (Protocolo de Controle de Transmissão) [112, 93]. Alguns outros exemplos são controle de congestão [51], *rate-based clocking* [13], e *soft-timer-based network polling* [13].

A área de sistemas distribuídos tem amplamente usado detectores de defeitos como um bloco básico de construção para resolver vários problemas, tais como consenso distribuído e *group membership*. No problema de consenso [40, 44, 50, 27], um conjunto de processos propõe valores a cada um dos outros e eles devem concordar sobre algum valor dentre aqueles propostos. No problema de *group membership* [17, 31, 95], um conjunto de processos deve manter uma lista atualizada dos processos atualmente bons e conectados. Há também outros problemas que podem ser resolvidos em sistemas assíncronos estendidos com detectores de defeitos, tais como *non-blocking atomic commitment*, difusão confiável com terminação (*terminating reliable broadcast*), e registrador atômico [71, 39].

Além disso, detectores de defeitos podem ter uma qualidade de serviço (QoS[3]), que é indicada por alguns requisitos, como tempo máximo de indicação de uma quebra de processo e a precisão com que um detector de defeitos indica que um processo realmente quebrou. O trabalho de Chen, Toueg e Aguilera [30], deste ponto em diante referido simplificadamente como Chen et al, formaliza a QoS de detectores de defeitos através da definição de uma métrica para a velocidade de detecção de um processo quebrado e métricas para a precisão da suspeita de quebras de processo. Eles desenvolvem um

[3]Do inglês: *Quality of Service*.

novo algoritmo detector de defeitos para relógios sincronizados chamado de NFD-S (novo detector de defeitos para relógios sincronizados[4]), e outros para relógios não sincronizados, chamados de NFD-U (novo detector de defeitos para relógios não sincronizados[5]) e NFD-E (novo detector de defeitos para relógios não sincronizados com estimativa[6]). Para cada um dos algoritmos apresentados eles fornecem um procedimento de configuração, que nesta tese chamaremos de configurador.

Os configuradores recebem como parâmetros as condições de atraso e de perda de mensagens e os requisitos de QoS, e fornecem como saída os parâmetros do detector de defeitos correspondente. Esses parâmetros de saída vão configurar o quanto o detector de defeitos deve esperar pela chegada de uma mensagem *heartbeat* do processo monitorado, e de quanto em quanto tempo o processo monitorado deve enviar mensagens *heartbeat* ao detector de defeitos. Para a perda de mensagens, os configuradores de Chen et al assumiram apenas a probabilidade de perda média de mensagens.

1.2 Descrição do Problema

Dentro deste contexto apresentado, o problema abordado nesta pesquisa é descrito da seguinte forma: *Considere um cenário em que há a ocorrência de perdas de mensagens em rajadas, por exemplo, em redes de longa distância* [11, 111, 114] *e em redes sem fio* [12, 93]. *Neste cenário, os algoritmos NFD-S, NFD-U e NFD-E e os configuradores de Chen et al ainda garantem os requisitos de QoS? Caso os algoritmos de detectores de defeitos e/ou os configuradores não sejam adequados para este cenário, então como desenvolver outros algoritmos e/ou configuradores que garantam os requisitos de QoS?*

1.3 Descrição Sucinta da Solução

Este texto apresenta, como uma solução para este problema, um modelo de QoS para detectores de defeitos na presença de rajadas de perdas que se baseia no arcabouço teórico de Chen et al, em relação a QoS, e no modelo de Markov de Sanneck [92], para

[4]Do inglês: *New Failure Detector with Synchronized clocks.*

[5]Do inglês: *New Failure Detector with Unsynchronized clocks and known expected arrival times of heartbeats.*

[6]Do inglês: *New Failure Detector with Unsynchronized clocks and estimates of expected arrival times of heartbeats.*

capturar a ocorrência de perdas em rajadas. Para avaliar os configuradores propostos foram realizadas simulações destes com os algoritmos e os configuradores de Chen et al.

Para a construção dos configuradores, o modelo probabilístico de rede de Chen et al foi mudado para levar em conta as probabilidades de ocorrência de rajadas de perdas de tamanhos diferentes. Um modelo básico de Markov foi construído para determinar a ocorrência de chegadas de mensagens ou a perda em rajadas. Usando o mesmo algoritmo NFD-S de Chen et al, demos uma nova interpretação para a probabilidade de ocorrência de chegadas e perdas de mensagens, usando o modelo básico de Markov. A seguir desenvolvemos novos configuradores, para os algoritmos NFD-S, NFD-U e NFD-E, de Chen et al, que levam em conta os modelos propostos.

Foram realizados dois tipos de simulações, com dados gerados e com dados coletados, para avaliar a solução proposta. Para a simulação com dados gerados foi desenvolvido um gerador de mensagens que gera a seqüência de atrasos e perdas, seguindo a distribuição exponencial para os atrasos, e as distribuições de Pareto e geométrica para as rajadas de perdas. Para a segunda simulação, os dados coletados foram coletados de um link entre a UNICAMP e a UFMS (Universidade Federal de Mato Grosso do Sul). Os dados gerados e os dados coletados foram usados, respectivamente, em execuções dos algoritmos NFD-S e NFD-E, e alguns gráficos foram obtidos e os resultados analíticos gerados pelos configuradores propostos foram comparados com o comportamento dos algoritmos NFD-S e NFD-E.

1.4 Contribuições do Trabalho

As principais contribuições deste trabalho, sobre qualidade de serviço de detectores de defeitos na presença de rajadas de perdas, são algumas respostas ao problema mencionado na seção 1.2:

1) Os algoritmos de detectores de defeitos de Chen et al [30], NFD-S, NFD-U e NFD-E são robustos o suficiente para serem utilizados mesmo quando há probabilidade alta de ocorrência de rajadas longas de perdas de mensagens *heartbeat*.

2) Os configuradores propostos e apresentados nesta pesquisa são adequados para a configuração dos detectores de defeitos mesmo quando rajadas longas de perdas ocorrem com alta freqüência. Por exemplo, quando a distribuição de probabilidade

das rajadas de perdas é similar a Pareto. Os configuradores propostos também comportam-se bem quando a distribuição é geométrica.

3) Os experimentos confirmam resultados anteriores de que é difícil obter estimadores/preditores de atrasos que levem os detectores de defeitos a atenderem as principais métricas de QoS ao mesmo tempo.

4) A partir de experimentos adicionais verificou-se que é possível, em vários casos, melhorar a precisão de detectores de defeitos através de estimadores compostos resultantes da combinação de estimadores de atrasos simples.

Estas contribuições estão descritas no Capítulo 4, sobre o modelo proposto, no Capítulo 5 sobre a avaliação do modelo proposto via simulação e no Capítulo 6, sobre um estimador composto.

1.5 Organização do Texto

O restante dos capítulos do texto está organizado da seguinte maneira. O Capítulo 2 expõe alguns conceitos fundamentais usados no texto, tais como confiabilidade de sistemas, sistemas distribuídos, modelagem de canais de comunicação, detectores de defeitos e consenso distribuído. O Capítulo 3 discute sucintamente o arcabouço teórico de Chen et al [30], sobre qualidade de serviço de detectores de defeitos, e discute outros trabalhos relacionados. O Capítulo 4 apresenta o modelo proposto para a qualidade de serviço de detectores de defeitos na presença de rajadas de perdas. O Capítulo 5 mostra alguns detalhes do simulador usado para avaliar o modelo proposto, descreve como foram realizadas as simulações, e discute os resultados da simulação. O capítulo 6 apresenta uma pequena modificação no estimador do algoritmo NFD-E que possibilita uma melhoria da precisão de detectores de defeitos. Finalmente, o Capítulo 7 discute as contribuições da pesquisa, as dificuldades encontradas no seu desenvolvimento, e sugere alguns trabalhos futuros.

Capítulo 2

Conceitos Fundamentais

Este capítulo apresenta os conceitos fundamentais utilizados no restante do texto. Inicia com dependabilidade de sistemas e apresenta uma visão geral dos componentes de sistemas distribuídos. O enfoque é então direcionado aos componentes mais básicos de sistemas distribuídos tolerantes a falhas, que são canais de comunicação, comunicação confiável e não confiável, consenso distribuído e detectores de defeitos. Finalmente, este capítulo termina com algumas conclusões.

2.1 Dependabilidade de Sistemas

Esta seção baseia-se nas definições de Azivienis et al [16]. Neste trabalho, estamos interessados na dependabilidade[7], ou seja, na confiança justificada dada a um sistema em relação ao serviço correto que ele fornece. Um *sistema* é uma entidade que interage com outras entidades. Estas outras entidades formam o *ambiente* do sistema. Um sistema é composto por vários *componentes*. A *função* de um sistema é o que se pretende que o sistema realize, e está descrita na sua *especificação funcional*. O *comportamento* de um sistema é o que o sistema faz para implementar a sua função. O *serviço* oferecido por um sistema é o comportamento do sistema que é percebido pelos usuários do sistema. Serviço *correto* é aquele que implementa a função do sistema.

A dependabilidade agrega vários atributos, que podem ser ameaçados. Contudo, os atributos podem ser alcançados através de alguns meios. Os atributos de dependabilidade são discutidos na seção 2.1.1, as ameaças na seção 2.1.2, e os meios na seção 2.1.3.

[7]Do inglês: *dependability*.

2.1.1 Atributos de Dependabilidade

Os atributos de dependabilidade de um sistema são:

- *Disponibilidade*[8]: capacidade de estar pronto para fornecer um serviço correto.
- *Confiabilidade*[9]: capacidade de fornecer um serviço correto de forma contínua.
- *Segurança*[10]: capacidade de não provocar conseqüências catastróficas para os usuários e o ambiente que interagem com o sistema.
- *Integridade*[11]: capacidade de não sofrer alterações impróprias ou indesejáveis.
- *Manutenibilidade*[12]: capacidade de permitir reparos e modificações.

Alguns exemplos de sistemas e um dos atributos (dentre todos) que cada um deles deveria ter são: caixas eletrônicos (*disponibilidade* para realizar operações bancárias quando for necessário), naves espaciais (*confiabilidade* para que a nave não pife e a tripulação perca o controle), usinas nucleares (*segurança* para evitar que algum material radioativo vaze para fora da usina de forma descontrolada), servidores de arquivos (*integridade* dos dados para que possam ser recuperados corretamente com o mesmo conteúdo que foram gravados), e sistemas comerciais de grande escala (*manutenibilidade* para permitir rápida correção de falhas de projeto e implementação, e para atualizar rapidamente o sistema com novos recursos).

2.1.2 Ameaças à Dependabilidade

A dependabilidade de um sistema pode ser prejudicada pelas seguintes ameaças: falha, erro e defeito. Um *defeito* de serviço ocorre quando o serviço não está de acordo com a sua especificação funcional, ou porque a sua especificação não descreve adequadamente a função do sistema. *Modos de defeito de serviço*[13] são os diferentes modos em que um defeito se manifesta no sistema. Um *erro* é um estado do sistema em que pelo menos um

[8] Do inglês: *Availability*.
[9] Do inglês: *Reliability*.
[10] Do inglês: *Safety*.
[11] Do inglês: *Integrity*.
[12] Do inglês: *Maintainability*.
[13] Detalhes podem ser consultados na seção 3.3 de Azivienis et al [16].

dos seus componentes oferece um serviço incorreto que pode levar ou não à ocorrência de defeito no sistema. A causa de um erro é chamada de *falha*.

Uma falha é *ativa* quando ela produz um erro, caso contrário, ela é *latente*. A *ativação de falhas* é a aplicação de uma entrada a um componente que leva a falha latente a tornar-se ativa. A relação causa-efeito é: falha pode levar a um erro que pode levar a um defeito. Considere o exemplo a seguir:

- Um sistema possui um processo monitor chamado q e um processo monitorado chamado p, que estão em redes distintas na Internet. O processo p envia periodicamente uma mensagem *p-vivo* para indicar ao processo q que ele está ativo (não está quebrado). O processo q possui um estimador para o valor de *timeout*, que é o período de tempo que ele deve esperar para receber uma mensagem *p-vivo*. Se alguma dessas mensagens não chegar antes de terminar o período determinado por *timeout*, então q indica que p está quebrado, caso contrário, ele indica que p está ativo. Caso a estimativa gerada por q não seja boa então ele irá supor erroneamente que p está quebrado. Seja recuperação o período decorrido entre o instante que q indica erroneamente que p está quebrado até o momento em que q indica que p está ativo. Uma parte da especificação exige que a média diária dos períodos de recuperação fornecida por q seja no máximo o valor *recuperação-solicitada* fornecido pelo usuário. E outra parte da especificação indica, baseada em estudos, que o melhor estimador para atender o requisito *recuperação-solicitada* é aquele que usa a média de atraso das 32 mensagens *p-vivo* mais recentes, e a perda média das mensagens *p-vivo*.

- Suponha que a implementação do estimador de *timeout* possua a seguinte falha: uso da perda média das mensagens *p-vivo* e da média de atraso das 10 mensagens *p-vivo* mais recentes. Neste caso, o valor gerado pelo estimador eventualmente será um *erro* quando levar q a indicar que p está quebrado mas ele estiver ativo. A estimativa errada não levará a um defeito enquanto o valor médio diário dos períodos de recuperação for menor ou igual à *recuperação-solicitada*. Caso contrário, ocorre um *defeito* no sistema.

2.1.3 Meios para a Obtenção de Dependabilidade

Os vários meios, para se obter os atributos de dependabilidade de um sistema, podem ser agrupados nas quatro categorias seguintes:

- *Prevenção de Falhas*[14]: corresponde aos meios para prevenir que falhas ocorram ou sejam introduzidas.
- *Tolerância a Falhas*[15]: corresponde aos meios para evitar defeitos de serviços mesmo que falhas ocorram.
- *Remoção de Falhas*[16]: corresponde aos meios para reduzir o número e a severidade das falhas.
- *Previsão de Falhas*[17]: corresponde aos meios para estimar o número de falhas presentes, a incidência futura, e as prováveis conseqüências delas.

A prevenção e a tolerância a falhas visam capacitar o sistema para alcançar a dependabilidade. A remoção e a previsão de falhas visam obter confiança na dependabilidade do sistema justificando que as especificações de dependabilidade são adequadas e que o sistema provavelmente as garante.

2.2 Sistemas Distribuídos

Não há consenso na literatura sobre uma definição de sistemas distribuídos. Uma definição geral para sistemas distribuídos, que é adaptada da definição de Tel [102] é: Sistema Distribuído é uma coleção de computadores, processadores ou processos que são capazes de trocar informações entre si. Contudo, esta definição confunde-se com a definição de redes de computadores de Tanenbaum [101].

Uma definição similar, dada por Coulouris et al [34], é: Sistema Distribuído é um sistema em que o hardware e o software localizado em computadores em rede comunicam-se e coordenam suas ações somente pela passagem de mensagens. Coulouris et al fornece a Internet como um exemplo de sistema distribuído e a Web como um serviço fornecido pela Internet. Uma definição mais adequada, fornecida por Veríssimo e Rodrigues [108], é a mesma de Tel, mais o aspecto de compartilhamento de estado entre os processos a fim de resolver algum problema.

Uma das definições mais restritivas é dada por Tanenbaum [101], onde podemos considerar a definição de Veríssimo e Rodrigues [108] como o aspecto de coesão usado

[14]Do inglês: *Fault prevention.*
[15]Do inglês: *Fault tolerance.*
[16]Do inglês: *Fault removal.*
[17]Do inglês: *Fault forecasting.*

por ele, mais o aspecto de transparência em que um usuário vê todo o sistema como uma única máquina virtual. Para Tanenbaum, um exemplo de sistema distribuído é a Teia Mundial (*World Wide Web*), que apresenta tudo ao usuário na forma de uma página *Web*. Ou seja, para Tanenbaum, a distinção entre uma rede e um sistema distribuído está no software (middleware ou sistema operacional), em vez do hardware.

Neste trabalho, pode-se definir sistema distribuído como um conjunto de computadores interligados por canais de comunicação e que trocam informações entre si com objetivo de resolver um problema em comum.

Nas subseções seguintes apresentamos algumas propriedades de sistemas distribuídos, modelos de sistemas distribuídos e alguns componentes de sistemas distribuídos.

2.2.1 Propriedades de Sistemas Distribuídos

Em sistemas distribuídos, a corretude de um algoritmo distribuído geralmente é determinada pelas propriedades de *safety* e *liveness* [69, 54, 102, 15]. O termo *safety* neste contexto, não possui o mesmo significado que o termo *segurança* definido na seção 2.1.1. Uma *configuração* de um sistema é o conjunto de estados de todos os componentes de um sistema [102]. *Safety* indica que, em qualquer execução de um sistema, uma certa propriedade deve permanecer válida *durante cada configuração* alcançada naquela execução [102], ou seja, algo de incorreto *nunca* ocorrerá dentro de um sistema [54]. *Liveness* indica que, em qualquer execução de um sistema, uma certa propriedade deve valer para *alguma configuração* alcançada naquela execução [102], ou seja, algo correto irá acontecer em algum momento durante a execução do sistema [54].

Considere o sistema com os processos monitor q e monitorado p da seção 2.1.2. Naquele exemplo, uma propriedade de *safety* poderia ser: a média diária dos períodos de recuperação, fornecida por q deve ser no máximo o valor *recuperação-solicitada* fornecido pelo usuário. Uma propriedade de *liveness* poderia ser: o processo q indica o estado (ativo ou quebrado) de p.

Se a implementação do sistema seguisse rigorosamente a especificação, e a especificação estiver correta, q sempre iria garantir as propriedades de *safety* e *liveness* descritas anteriormente. Caso contrário, se o estimador implementado fosse aquele incorreto descrito na seção 2.1.2, então a propriedade de *liveness* continuaria a ser garantida, mas não a propriedade de *safety*, porque o estimador em algum momento poderia levar q a fornecer uma média diária dos períodos de recuperação maior que *recuperação-solicitada*.

2.2.2 Modelos de Sistemas Distribuídos

Na *World Wide Web* pode-se considerar que grande parte dos computadores deste sistema não possui limites bem definidos de atraso nem de perda de mensagens e cada computador tem o seu próprio relógio local. Para ser possível a resolução de algum problema distribuído dentro deste sistema, ou em outros sistemas distribuídos, é necessário supor um modelo do sistema. Modelos de sistemas distribuídos usualmente definem dois aspectos: temporização e hipóteses de defeitos.

Em relação à temporização, o modelo síncrono é o mais restritivo porque supõe que há limites no tempo de processamento e no atraso das mensagens [41]. O modelo de sistema distribuído que mais se aproximaria da Internet seria o modelo de um *sistema distribuído assíncrono* [50], onde não há restrições de temporização [27, 77, 54, 102]: limites no tempo de atraso de mensagens, desvio de relógios ou no tempo necessário para executar um passo do sistema. Há também os modelos de sistema distribuído *parcialmente síncrono* [40, 44, 72], *quase-síncrono* [106, 7, 8], *assíncrono temporizado* [35, 78, 109, 49], e *Timely Computing Base* [24, 107, 25], que ficam entre os extremos síncrono e assíncrono. Recentemente, Veríssimo [105] definiu o modelo híbrido de sistema distribuído chamado Wormholes. Neste modelo, partes do sistema, com propriedades diferentes, podem coexistir, tais como processos e canais de comunicação assíncronos com defeitos arbitrários (quaisquer tipos de defeitos), e processos e canais de comunicação síncronos com defeitos de quebra.

Os modelos *quase-síncrono*, *assíncrono temporizado*, e *Timely Computing Base* têm sido usados para sistemas de tempo real, onde restrições de tempo são essenciais. Nesses sistemas há a preocupação de que o sistema também tenha a propriedade de *timeliness*, que informalmente indica a capacidade do sistema de iniciar certas execuções no tempo programado e gastar um tempo limitado para terminá-las [108, 107].

Em relação às hipóteses de defeitos, os modelos de sistemas distribuídos basicamente fazem suposições sobre a perda e/ou atraso de mensagens nos canais de comunicação que interligam os processos, e sobre a quebra de processos. A seguir, nas hipóteses de defeitos, supõem-se que *canal de comunicação confiável* é aquele que sempre entrega uma mensagem, ou seja, não há perda de mensagens nem duplicação de mensagens, e as mensagens são entregues não corrompidas. Um *canal de comunicação não confiável* é aquele que possui as mesmas propriedades de um canal confiável, com a exceção de que pode perder mensagens.

As hipóteses de defeitos mais usadas em modelos de sistemas distribuídos são: *os processos podem quebrar e parar*[18], e os canais de comunicação são confiáveis [27], ou não confiáveis [3]; e *os processos podem quebrar e se recuperar*[19], e os canais de comunicação são confiáveis [61], ou não confiáveis [4].

Em modelos de sistemas distribuídos de tempo real, a hipótese de defeito mais usada é a que segue. Os processos podem quebrar e parar, e os canais de comunicação são não confiáveis, e os processos e os canais de comunicação podem sofrer defeitos de desempenho. Defeitos de desempenho são aqueles que ocorrem quando um processo não reage a um evento dentro de um intervalo de tempo máximo pré-definido, ou quando os canais de comunicação entregam mensagens com atraso [35, 107].

2.2.3 Componentes de Sistemas Distribuídos

Há vários trabalhos [22, 9, 104, 36, 75, 76, 83, 33, 48, 67, 55, 64] que indicam quais são os serviços ou componentes que um sistema distribuído tolerante a falhas deveria fornecer. Alguns desses componentes são:

- *Canais de comunicação*: Os canais de comunicação usualmente são confíaveis ou não confiáveis, da forma definida na seção 2.2.2.

- *Comunicação confiável e não confiável*: A comunicação pode ser confiável ou não, com garantia de entrega ordenada ou não de mensagens, com envio a todos os processos do sistema (difusão), ou dirigida a um único processo ou a um grupo deles.

- *Detectores de defeitos*: Os detectores de defeitos são usados para detectar se um processo quebrou ou não [27]. Os detectores de defeitos podem usar o componente comunicação confiável ou não confiável, dependendo do modelo de sistema.

- *Gerenciamento de grupos*: O gerenciamento de grupos cuida da criação, remoção, e estabelecimento de uma visão consistente do conjunto de processos que pertencem a determinado grupo [17, 31, 95, 96]. Este componente geralmente usa os componentes detectores de defeito e comunicação confiável em grupo.

[18] Do inglês: *Crash-stop*.
[19] Do inglês: *Crash-recovery*.

- *Consenso distribuído*: No problema de consenso [50, 40, 27], um conjunto de processos propõe valores a cada um dos outros e eles devem concordar sobre algum valor dentre aqueles propostos. Este componente pode usar o componente comunicação confíavel para comunicação em grupo ou difusão confiável.

- *Replicação de processos e dados*: Processos e dados podem ser replicados para permitir uma maior disponibilidade de processos e/ou de dados [97, 59, 18, 45, 110]. Geralmente a replicação usa todos os componentes anteriores com comunicação confiável.

- *Recuperação por retrocesso de estado*[20]: Requer que os processos armazenem o seu estado periodicamente e de forma consistente, a cada ponto de verificação, para que eles possam continuar o processamento após a ocorrência de defeitos em sua execução [46]. A recuperação por retorno pode usar o componente comunicação confiável para estabelecer os pontos de verificação.

- *Transações atômicas*: Uma transação atômica é aquela composta de várias operações, onde todas são realizadas ou nenhuma é realizada. As transações atômicas permitem manter a consistência dos dados manipulados nas transações, mesmo se algum defeito ocorrer nos processos envolvidos [56, 28, 89, 113]. Este componente pode usar todos os componentes anteriores com comunicação confiável.

Contudo, devido ao enfoque desta pesquisa, as próximas seções detalham somente alguns dos componentes que estão diretamente relacionados com este trabalho: canais de comunicação, comunicação confiável e não confiável, consenso distribuído, e detectores de defeitos.

2.3 Canais de Comunicação e Comunicação Confiável e Não Confiável

A maioria dos trabalhos na literatura que afirmam fornecer comunicação confiável refere-se à abstração de canal confiável no sentido prático. Ou seja, através de retransmissões de pacotes e envio de pacotes de confirmação é possível implementar um canal "confiável" sobre um canal não confiável (veja o capítulo 22 de [77]). O problema

[20]Do inglês: *Rollback-recovery*.

é que essas soluções não são quiescentes. Ou seja, se um processo receptor p quebra e não envia alguma mensagem de confirmação a um processo q, então o processo q não irá parar de retransmitir a mensagem que não foi confirmada pelo processo p. Aguilera et al [5] apresentam uma forma de fornecer comunicação confiável quiescente, estendendo o sistema distribuído com um detector de defeitos chamado Heartbeat.

2.3.1 Modelos Básicos para a Medição de Atrasos e Perdas de Pacotes

Os canais de comunicação, particularmente se consideramos redes de longa distância, possuem grande variação no atraso e na perda de pacotes [52]. Portanto, é importante considerar bons estimadores para o atraso e para a perda de pacotes. Dois modelos amplamente usados para o envio de pacotes, para a medição de atrasos e perdas de pacotes pelos canais de comunicação, são:

- *Modelo Pull*: um processo E envia pacotes a um processo R, que a cada pacote recebido de E retorna um pacote para E. A cada pacote recebido, E calcula o tempo de ida e volta (*round trip time*).

- *Modelo Push*: um processo E envia pacotes com o tempo de envio para um processo R, que a cada pacote recebido calcula o atraso usando o tempo de envio (que está em cada pacote recebido) e o tempo de recebimento. Para obter uma boa precisão no cálculo do atraso é conveniente que os relógios de E e R estejam sincronizados, p. ex. usando NTP (*Network Time Protocol* - Protocolo de Tempo de Rede)[21].

2.3.2 Modelagem de Atrasos

Na Internet, alguns trabalhos procuram usar a estimativa de atrasos para uso em *timers* para a retransmissão de pacotes [65, 62, 6], e em detectores de defeitos [20, 87].

A modelagem tem usado tanto métodos simples para estimação, tais como apenas o valor do último atraso [99], média simples [98], média exponencialmente ponderada [62], quanto métodos mais avançados como séries temporais [85, 86, 87].

[21] Um sítio na Internet com várias informações sobre NTP é `http://www.ntp.org/`

2.3.3 Modelagem de Perdas

A modelagem de perdas tem sido usada para modelar perdas de pacotes em redes de longa distância e em redes sem fio porque a ocorrência de perda de pacotes em rajadas é comum.

Bolot [23] mediu a perda de pacotes usando o modelo *pull*: a) Probabilidade de perda incondicional (*ulp*), definida como $Pr(rtt_n = 0)$, onde *rtt* é o tempo de *round trip time* do pacote n; b) Probabilidade de perda condicional (*clp*), definida como $Pr(rtt_{n+1} = 0 \mid rtt_n = 0)$; e Comprimento de perda de pacotes (*packet loss gap*), *plg*, definido como $1/(1-clp)$. *ulp* é a média de perda de mensagens. *clp* caracteriza a tendência para perdas em rajadas de pacotes. Ele usou os intervalos entre envios η = (8, 20, 50, 100, 200, 500), em ms, e encontrou, respectivamente, *ulp* = (0,23, 0,16, 0,12, 0,10, 0,11, 0,07), *clp* = (0,60, 0,42, 0,27, 0,18, 0,18, 0,09), e *plg* = (2,5, 1,7, 1,3, 1,2, 1,2, 1,1). Fica claro que, à medida que η aumenta: a) *ulp* diminui, estabilizando por volta de 10%; b) *clp* também diminui, mas tem valores sempre maiores que *ulp*; e c) *plg* é diretamente proporcional a *clp*, por definição, e estaciona mais próximo de 1. A definição de *clp* indica intuitivamente a probabilidade que o próximo pacote $n+1$ seja perdido dado que o pacote n foi perdido. Quanto maior for o valor de *clp* maior será o tamanho de rajada de perda esperado. É por isso que *clp* é diretamente proporcional à *plg*.

Modelos de cadeias de Markov têm sido adequados para modelar rajadas de perdas de pacotes em redes de longa distância [111, 114] e em redes sem fio [74, 68]. Os modelos de Markov possuem geralmente 2^m estados, onde m é a ordem da cadeia de Markov, e representa as últimas perdas consecutivas que são consideradas pela cadeia de Markov.

Yajnik et al [111] observaram que entre 33 segmentos de conjuntos de dados com η igual a 160 ms, 7 são bem modelados pelo modelo de Bernoulli, 10 são bem modelados pela Cadeia de Markov de ordem 1 (2 estados), e 16 são bem modelados por modelos de Cadeia de Markov com ordem 2 ou maior. Entretanto, entre 33 segmentos de conjuntos de dados com η igual a 20 ms e 40 ms, os modelos de Bernoulli e de Cadeia de Markov de ordem 1 não são precisos, e a ordem estimada para a Cadeia de Markov variou de 10 a 42.

Zhang [114] verificou que em traços com perdas (que possuem taxa de perdas acima de 1%), de conjuntos de dados com média η igual a 100 ms (10 Hz rate) e 200 ms (20 Hz rate), que somente 6% são bem modelados como Bernoulli, 68% com o modelo de Cadeia de Markov de ordem 1 (2 estados), e 96% com o modelo de Cadeia de Markov de ordem

2 (4 estados).

Sanneck [92] propõe um modelo de Markov para rajadas de perdas que usa apenas $m+1$ estados. A abordagem dele usa a distribuição de probabilidade do tamanho das rajadas de perdas para aproximar tanto as probabilidades de estado quanto as probabilidades de transição de estado. O modelo considerado nesta pesquisa é o modelo de Markov para rajadas de perdas com espaço de estados ilimitado ($m + 1$ estados) (veja a Figura 2.1). O modelo é chamado ilimitado porque m tende a infinito e considera sempre m como o tamanho máximo de rajada de perdas ocorrido. A variável aleatória X é definida como segue: $X = 0$ significa *nenhuma* perda de pacotes, $X = z$ $(0 < z < m)$ significa *exatamente* z pacotes consecutivos perdidos, $X \geq z$ significa *pelo menos* z pacotes consecutivos perdidos. Uma transição de estado ocorre dependendo das probabilidades de transição p_{ij}, com $i < j$ (para comprimentos de rajadas de perdas menores ou iguais a m) ou $i \geq j = 0$ (para uma chegada de pacote). A probabilidade de estado do sistema para $0 < z \leq m$ é $Pr(X \geq z)$, e para $z = 0$ é $Pr(X = 0)$.

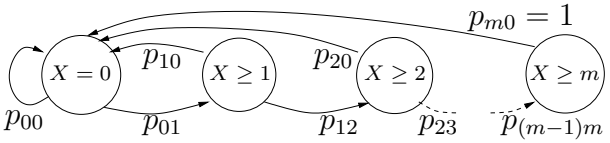


Figura 2.1: Modelo de Sanneck com espaço de estados ilimitado.

2.4 Consenso Distribuído

Nas subseções seguintes apresentamos o problema de consenso distribuído e discutimos brevemente quando este problema pode ser resolvido em um sistema distribuído.

2.4.1 Definição de Consenso

Consenso distribuído se refere ao problema em que um grupo de n processos em um sistema distribuído, onde todos os processos iniciam com um mesmo valor (p. ex. *confirmar* ou *abortar*) ou um conjunto de valores (p. ex. lista de processos de um grupo), e precisam concordar sobre algum valor (ou conjunto de valores) proposto por algum dos n processos [50, 40, 44, 26, 70, 3, 4, 58]. É importante ressaltar que vários problemas,

tais como transação atômica não bloqueante[22], difusão atômica confiável[23], e comunicação em grupo com ordenação total[24], podem ser resolvidos através do uso de um serviço de consenso.

Uma definição correta e formal de um problema de consenso depende de três variáveis: tipo de sistema distribuído (síncrono, assíncrono, sincronia parcial, etc), hipóteses de defeitos (processos podem quebrar e parar, processos podem quebrar e recuperar, canais de comunicação confiáveis ou não confiáveis), e quais são as propriedades. O que é exatamente um processo correto ou falho depende das hipóteses de defeitos [57]. Há basicamente três propriedades ou condições que vão depender do tipo de sistema distribuído e das hipóteses de defeitos supostas [57]:

- *Terminação*[25]: indica quais processos irão, em algum momento, decidir.
- *Acordo*[26]: indica quais processos irão decidir sobre o mesmo valor.
- *Validade*[27]: indica quais são os valores de decisão possíveis para cada tipo de processo, de acordo com os valores iniciais.

Quando ambos os processos (corretos e falhos) devem atender a alguma propriedade, essa propriedade é chamada uniforme. A propriedade de terminação define a propriedade de *liveness* do problema de consenso, enquanto as propriedades de acordo e validade definem a propriedade de *safety* [57]. Outras definições de consenso, como a apresentada por Chandra e Toueg [27], possuem a propriedade de *integridade*[28], que indica que cada processo (correto ou falho) decide no máximo uma vez.

2.4.2 Resultados de Impossibilidade de Consenso Distribuído

Consenso pode ser resolvido deterministicamente em sistemas distribuídos síncronos onde processos e canais de comunicação são confiáveis. Neste cenário, os processos não falham e as mensagens são entregues sem corrupção e exatamente uma única vez. Quando os processos são confiáveis mas os canais de comunicação não são, o consenso não pode ser

[22]Do inglês: *non-blocking atomic commitment.*
[23]Do inglês: *atomic broadcast.*
[24]Do inglês: *total order multicast.*
[25]Do inglês: *Termination.*
[26]Do inglês: *Agreement.*
[27]Do inglês: *Validity.*
[28]Do inglês: *Integrity.*

resolvido deterministicamente. E quando os canais de comunicação são confiáveis mas os processos podem quebrar ou ter comportamento bizantino, é possível resolver o problema de consenso deterministicamente desde que algumas restrições sejam obedecidas [77].

Entretanto, Fischer, Lynch e Paterson [50] provaram o seguinte resultado de impossibilidade: não existe algoritmo determinístico que solucione o problema de consenso em sistemas distribuídos assíncronos onde pelo menos um processo pode quebrar, e os canais de comunicação são confiáveis. Deste ponto em diante esse resultado de impossibilidade será referido apenas como FLP.

Várias alternativas têm sido propostas para contornar o FLP, entre elas destacam-se o uso de algoritmos aleatorizados e a definição de sistemas distribuídos mais restritos. Os algoritmos aleatorizados conseguem contornar o FLP com probabilidade 1 porque levam o conjunto de execuções sem terminação a terem probabilidade 0 [19, 14]. Aspnes [14] apresenta uma visão geral de protocolos de consenso aleatorizados para contornar FLP em sistemas distribuídos assíncronos e apresenta protocolos aleatorizados que resolvem consenso em sistemas que possuem memória compartilhada.

Há várias definições de sistemas distribuídos mais restritos, tais como os modelos *parcialmente síncronos* [40, 44, 72], *quase síncrono* [106, 7, 8], *assíncrono temporizado* [35, 78, 109, 49], o modelo assíncrono estendido com detectores de defeitos não confiáveis [27, 3, 4, 57], e o modelo de oráculos [90, 79, 58]. Há ainda abordagens híbridas que usam algoritmos aleatorizados com detectores de defeitos não confiáveis [2, 82]. O modelo Wormholes [105] permite contornar FLP e é uma opção para resolver consenso (em vetor) até mesmo em sistemas com defeitos bizantinos em processos e canais de comunicação assíncronos [84]. A implementação de wormhole usada para resolver consenso em vetor é o *Trusted Timely Computing Base* (TTCB). TTCB é síncrono e falha somente por quebra.

2.5 Detectores de Defeitos

Esta seção discute brevemente sobre detectores de defeitos em sistemas distribuídos. Raynal [91] escreveu um tutorial recente sobre detectores de defeitos para sistemas distribuídos assíncronos.

2.5.1 Definição de Detectores de Defeitos não Confiáveis

Chandra e Toueg [27] definem detectores de defeitos não confiáveis no contexto de sistemas distribuídos assíncronos com canais de comunicação confiáveis, onde os processos podem "quebrar" mas não se recuperam após a "quebra". Cada processo possui um módulo detector de defeitos e uma lista com os processos suspeitos de terem "quebrado".

Esses detectores de defeitos não são confiáveis porque podem fazer suspeitas incorretas sobre quais processos estão ativos e quais "quebraram". Isto pode acontecer devido às características próprias que o sistema distribuído assíncrono possui de não ter limites superiores para os tempos de processamento e de transmissão de mensagens.

Quando o detector de defeitos funciona mal, porque coloca e tira indefinidamente processos da lista de suspeitos, haverá prejuízo para *liveness* enquanto o detector de defeitos não agir corretamente por um "tempo suficiente" para que o algoritmo de consenso termine. Entretanto, mesmo com o detector de defeitos funcionando mal, não haverá prejuízo para *safety*, ou seja, o algoritmo de consenso não decidirá sobre valores diferentes (seria uma violação da propriedade de acordo) nem sobre valores não permitidos (seria uma violação da propriedade de validade).

2.5.2 Propriedades de Detectores de Defeitos não Confiáveis

Para evitar a definição de detectores de defeitos baseada em particularidades de implementação, Chandra e Toueg [27] definem detectores de defeitos de acordo com as propriedades abstratas de completude e precisão. A propriedade de completude indica se algum ou todos os processos devem suspeitar, de forma permanente, dos processos que "quebraram". A propriedade de precisão indica quando algum ou todos os processos ativos podem ser suspeitos de terem "quebrado". Há duas propriedades de completude e quatro propriedades de precisão:

- *Completude forte*: há um tempo futuro a partir do qual todo processo que "quebra" é suspeito de forma permanente por todos os processos corretos.
- *Completude fraca*: há um tempo futuro a partir do qual todo processo que "quebra" é suspeito de forma permanente por algum processo correto.
- *Precisão forte*: nenhum processo é suspeito antes que ele "quebre".
- *Precisão fraca*: algum processo correto nunca é suspeito antes de ter "quebrado".

- *Precisão terminal forte*: há um tempo futuro a partir do qual todos os processos corretos não são suspeitos por algum processo correto.
- *Precisão terminal fraca*: há um tempo futuro a partir do qual algum processo correto nunca é suspeito por algum processo correto.

Cada combinação de uma propriedade de completude com uma propriedade de precisão produz uma classe diferente de detectores de defeitos. Por exemplo, a classe P possui precisão forte e completude forte, a classe $\Diamond P$ possui precisão terminal forte e completude forte, a classe S possui precisão fraca e completude forte, a classe $\Diamond S$ possui precisão terminal fraca e completude forte, e a classe $\Diamond W$ possui precisão terminal fraca e completude fraca.

Há um algoritmo de redução que transforma qualquer detector de defeitos D com completude fraca em outro detector de defeitos D' com completude forte. Diz-se então que D' é redutível a D ($D \succeq D'$) e que D' é mais fraco que D, ou seja, o que pode ser feito por D' também pode ser feito por D. Se $D \succeq D'$ e $D' \succeq D$ então D e D' são equivalentes ($D \cong D'$). Como também pode-se reduzir trivialmente um detector de defeitos D com completude forte em outro detector de defeitos D' com completude fraca, Chandra e Toueg provam que $P \cong Q$, $S \cong W$, $\Diamond P \cong \Diamond Q$, e $\Diamond S \cong \Diamond W$.

2.5.3 Detectores de Defeitos e Consenso Distribuído

A definição de um problema de consenso para um sistema distribuído assíncrono com processos que podem quebrar e canais de comunicação confiáveis supõe que um processo correto é aquele que nunca quebra e um processo falho é aquele que quebra, durante a execução do algoritmo de consenso [57]. Além disso, as propriedades deste tipo de problema de consenso são [27]:

- *Terminação*: Todo processo correto decide sobre algum valor em algum tempo futuro.
- *Integridade Uniforme*: Todo processo decide no máximo uma vez.
- *Acordo Uniforme*: Nenhum processo (correto ou falho) decide sobre valores diferentes.
- *Validade Uniforme*: Todo processo que decide, decide sobre um valor proposto por algum processo.

Esse tipo de problema de consenso é chamado consenso uniforme, porque a propriedade de acordo força a mesma decisão para todos os processos (corretos ou falhos) que decidem.

Chandra e Toueg possibilitam consenso em um sistema distribuído assíncrono ao estendê-lo com detectores de defeitos não confiáveis, contornando FLP. Eles mostram dois algoritmos de consenso uniforme, um supondo detectores de defeitos da classe S e que pelo menos um processo não "quebre", e o outro supondo detectores de defeitos da classe $\Diamond S$ e que mais da metade dos processos não "quebre".

Eles também sumarizam o fato, provado por Chandra et al [26], de que um detector de defeitos da classe $\Diamond W$ é o mais fraco para que consenso seja solucionado em sistemas distribuídos assincronos. Neste caso, um detector de defeitos $\Diamond W_0$ que seja o mais fraco detector de defeitos da classe $\Diamond W$ exibe todos os comportamentos de detectores de defeitos permitidos pelas propriedades de completude fraca e precisão terminal fraca que definem $\Diamond W$. Chandra et al [26] mostram que $\Diamond W_0$ é o mais fraco detector de defeitos que pode ser usado para resolver consenso com $n > 2f$, onde f é o número máximo de processos que podem "quebrar" e n é o número total de processos. Mais ainda, eles provam que se $n \leq 2f$, qualquer detector de defeitos que pode ser usado para resolver consenso deve ser estritamente mais forte que $\Diamond W_0$.

Friedman et al [53] definem uma nova primitiva $fd_end()$, do detector de defeitos, que determina que as saídas seguintes do detector de defeitos devem ser ignoradas. Desta forma, detectores de defeitos não precisam estar sincronizados "para sempre" e nem precisam executar "para sempre", mas apenas o tempo suficiente para que um protocolo de nível superior, como consenso, termine. Esta primitiva define uma nova classe de detectores de defeitos chamada $\Diamond S_{bl}$, que possui *completude forte* e *precisão terminal fraca com tempo de vida limitado*. t_e denota o instante de tempo da primeira invocação de $fd_end()$ que um processo realiza. A *precisão terminal fraca com tempo de vida limitado* estabelece que há um tempo t tal que, a partir de t até t_e, há um processo ativo que não é suspeito por todos os processos que não quebraram até o instante t_e. Após t_e, qualquer processo pode suspeitar de qualquer processo. Eles demonstraram a aplicabilidade do detector de defeitos $\Diamond S_{bl}$ em um algoritmo de consenso e em um algoritmo de registrador atômico pré-existentes que usam um detector de defeitos $\Diamond S$.

Várias extensões ao modelo de detectores de defeitos não confiáveis de Chandra e Toueg têm sido feitas para outras hipóteses de defeitos: processos podem quebrar (mas não se recuperam) e os canais de comunicação não são confiáveis e redes não são particionáveis (cada par de processos é alcançável a partir de outros a partir de determinados tipos

de caminhos) [1], os processos podem quebrar (mas não se recuperam) e os canais de comunicação não são confiáveis e a rede é particionável [3], os processos podem quebrar e se recuperar e os canais de comunicação não são confiáveis [4], os processos podem quebrar e não se recuperar (ou os processos podem parar de enviar apenas alguns tipos de mensagens, sem quebrar: uma espécie de comportamento bizantino) e os canais de comunicação são confiáveis [43, 42].

É importante destacar que o uso de detectores de defeitos não confiáveis possibilita a resolução não somente do problema de consenso distribuído, mas de outros problemas em sistemas distribuídos: *Atomic Broadcast* [27], *terminating reliable broadcast* [38], *non-blocking atomic commitment* [28], *group membership* [94, 95].

Oráculos são uma abstração para sistemas distribuídos que usam a idéia básica de detectores de defeitos, que é a de estender o sistema distribuído com um módulo que fornece informações a partir da manipulação das características do sistema subjacente. Mostefaoui et al. [79] definem oráculos como um dispositivo com propriedades abstratas e com uma interface (conjunto de operações), que permitem resolver problemas em sistemas assíncronos sujeitos a defeitos. Mostefaoui et al apresentam quatro oráculos: um detector de defeitos adivinho (Θ), um detector de defeitos escondido (Heartbeat), um detector de defeitos ($\Diamond S$), e um oráculo aleatório. Estes oráculos resolvem alguns problemas em sistemas distribuídos assíncronos sujeitos a quebras de processos e canais de comunicação com perdas justas (*fair lossy*). Canais de comunicação com perdas justas são aqueles que perdem mensagens de forma intermitente, mas como o processo emissor retransmite mensagens, a mensagem será recebida em algum momento pelo processo receptor caso este não quebre [3, 5].

Os dois primeiros oráculos são usados para resolver *quiescent URB (uniform realible broadcast)*, e os outros dois para resolver consenso. O oráculo $\Diamond S$ é o detector de defeitos não confiável $\Diamond S$ definido por Chandra and Toueg [27]. Um outro oráculo, definido para resolver consenso em sistemas distribuídos assíncronos sujeitos a quebras de processos e canais de comunicação confiáveis, é o oráculo líder (Ω). Esse oráculo satisfaz a seguinte propriedade: em algum momento futuro, todos os processos corretos escolhem um mesmo processo correto como o líder [26, 70, 58].

Jiménez et al [63] apresentam um novo detector de defeitos da classe Ω que não requer o conhecimento antecipado da identidade dos demais processos do sistema (*membership*). A restrição imposta pelo algoritmo é que cada processo correto possa ser alcançado a partir de qualquer outro processo correto no sistema, os processos podem quebrar mas

não se recuperar, e os canais de comunicação podem ser de dois tipos: i) *eventually timely*: há um limite desconhecido Δ nos atrasos de mensagens e um tempo T de estabilização global desconhecido (de todo o sistema), tal que se uma mensagem é enviada através de qualquer um desses canais em um tempo $t \geq T$, então esta mensagem é recebida até o tempo $t + \Delta$; e ii) *lossy asynchronous*: as mensagens podem ser perdidas ou atrasadas arbitrariamente. Além disso, eles mostram que é impossível implementar um detector de defeitos com *completude fraca* (por exemplo, $\Diamond W$), sem o conhecimento de *membership*, até mesmo em sistemas síncronos com canais confiáveis e sujeitos a quebra de um único processo.

Larrea et al [73] descrevem uma classe de detectores de defeitos chamada terminalmente consistente[29] e denotada por $\Diamond C$. Um detector dessa classe fornece uma lista de processos suspeitos, como detectores da classe $\Diamond S$, e um processo correto, como detectores da classe Ω. A grande vantagem de detectores de defeitos $\Diamond C$ é que permitem que algoritmos de consenso usem uma lista de suspeitos da forma convencional ($\Diamond S$), mas sem precisar usar o paradigma de rotação de coordenadores porque usa-se como coordenador o processo líder retornado por Ω. Com $\Diamond C$ pode-se reduzir o número de rodadas usuais ($\Omega(n)$) de algoritmos de consenso que usam $\Diamond S$, para uma única rodada, quando há pelo menos um processo correto que nunca é suspeito por qualquer processo. O algoritmo proposto por eles, para consenso com $\Diamond C$, também tem a característica importante de esperar por uma maioria de respostas positivas para decidir, mesmo que muitas outras respostas sejam negativas. Isto permite decidir em rodadas que muitos algoritmos de consenso existentes podem não decidir após receber uma única resposta negativa.

Mostefaoui e Raynal [81] introduzem a idéia de escopo da propriedade de precisão de detectores de defeitos. O escopo é um número $k \leq n$ de processos que não podem suspeitar erroneamente de um processo correto. Eles definem as classes S_k e $\Diamond S_k$ de detectores de defeitos, e apresentam um algoritmo de consenso que usa $\Diamond S_k$. Anceaume et al [10] apresentam um protocolo que transforma S_k em S, e $\Diamond S_k$ em $\Diamond S$. Eles também determinam que a condição necessária e suficiente para essas transformações serem possíveis é $f < k$, onde f é o número máximo de processos que podem quebrar.

Mostefaoui et al [80] usam o protocolo de Anceaume et al para transformar $\Diamond S_k$ em $\Diamond S$, e descrevem um protocolo que transforma $\Diamond S$ em Ω. Esse protocolo que transforma $\Diamond S$ em Ω usa mensagens com tamanho limitado a apenas um número de rodada mais uma

[29] Do inglês: *eventually consistent*.

única identidade de processo, diferente dos algoritmos de Chu [32] que usam mensagens que podem ter tamanhos não limitados. Mostefaoui et al também sugerem uma forma de garantir as propriedades de um detector $\Diamond S_k$ através de um cluster de $k = f + 1$ processos com canais de comunicação temporizados, e o restante do sistema permanecendo assíncrono.

2.5.4 Detectores de Defeitos Adaptativos

A maior parte das redes reais possui alguma variação no tempo de transmissão de mensagens e na perda de mensagens. Existem detectores de defeitos adaptativos que procuram variar os valores de *timeout* e/ou intervalo de envio de mensagens de monitoração de acordo com os valores de atraso e de perda recentemente percebidos.

Dois trabalhos desenvolvidos, no início desta pesquisa, supõem o modelo de Chandra e Toueg [27] (processos podem quebrar mas não se recuperar e os canais de comunicação são confiáveis): ADAPTATION [98] e DPCP [99]. Esses trabalhos usam o modelo *pull* e supõem dois eventos: `TIMEOUT` (uma mensagem de detecção de defeitos `I_AM_ALIVE` não é recebida dentro do período de *timeout* esperado) e `RESPONSE` (uma mensagem de detecção de defeitos `I_AM_ALIVE` é recebida dentro do período de *timeout* esperado). Além disso, para todos esses trabalhos, a carga de trabalho do sistema (`CTS`) corresponde à carga de trabalho do processamento nos nós mais os atrasos dos canais de comunicação.

O artigo sobre ADAPTATION descreve novos algoritmos adaptativos para detecção de defeitos em sistemas distribuídos assíncronos. Tais algoritmos aumentam ou diminuem os valores de *timeout* esperado (variável `timeout`) e o intervalo de envio de mensagens (variável `monitoring_interval`) de detecção de defeitos de acordo com os tempos decorridos por um conjunto de mensagens `I_AM_ALIVE` recentemente recebidas. O parâmetro `between_factor` relaciona o valor de `timeout` recentemente calculado com o valor da variável `monitoring_interval`. O parâmetro `timeout_factor` é usado após cada evento `TIMEOUT` para ajustar rapidamente os valores das variáveis `timeout` e `monitoring_interval` de forma a evitar que os próximos eventos sejam do tipo `TIMEOUT`, quando a causa das falsas suspeitas dos objetos monitorados é provocada por aumentos na `CTS` além do suportado pela variável `timeout`. Além disso, o artigo apresenta a influência do uso de ACE+TAO na implementação dos algoritmos em CORBA, alguns resultados experimentais e como FT-CORBA poderia ser estendido com os algoritmos adaptativos.

O artigo sobre DPCP apresenta uma abordagem diferente de ADAPTATION que

ignora o tempo decorrido por várias mensagens `I_AM_ALIVE` recentemente recebidas e usa apenas o tempo decorrido pela última mensagem `I_AM_ALIVE` recebida. Desta forma, DPCP apresenta uma precisão maior na detecção de defeitos do que ADAPTATION. Além disso, DPCP também usa os parâmetros `between_factor` e `timeout_factor` da mesma forma que ADAPTATION. Contudo, DPCP usa, para suportar pequenos aumentos na `CTS`, o parâmetro `timeout_delta` que é agregado ao valor da variável `timeout`. Adicionalmente, DPCP aborda a escalabilidade do número de mensagens de detecção de defeitos transmitidas através do parâmetro `MinimumTimeUnit`, que especifica um valor mínimo para a variável `monitoring_interval`. Este artigo também mostra que DPCP é melhor que ADAPTATION porque apresenta um número muito menor de falsas suspeitas (provocadas por aumentos na `CTS` além do suportado pela variável `timeout`).

Chen et al [30], Bertier et al [20, 21], Nunes e Jansch-Pôrto [85, 86, 87], Hayashibara et al [60] apresentam alguns dos detectores de defeitos adaptativos mais recentes. Chen et al apresentam um algoritmo do tipo *push* que usa a média dos atrasos recentes e uma margem de segurança (α) para estimar o próximo valor de timeout. Além disso, α poderia ser calculado periodicamente, para ser mais ajustado a mudanças nos atrasos e perdas da rede. Bertier et al descrevem um algoritmo do tipo *pull* que usa a média dos atrasos mais recentes e o estimador de Jacobson (para a margem de segurança). Nunes e Jansch-Pôrto usam um algoritmo do tipo *pull* e realizam vários experimentos que mostram que o uso de preditores de séries temporais é adequado quando o cálculo da margem de segurança (constante, estimador de Jacobson, ou estimador baseado em intervalo de confiança) também é bem escolhido. Hayashibara et al usam um algoritmo do tipo *push* e apresentam uma nova abordagem em que os detectores de defeitos podem fornecer como saída não somente valores binários (processo vivo ou processo suspeito), mas também um intervalo de valores. O detector de defeitos proposto por Hayashibara et al é chamado de detector de defeitos acumulador[30], e é representado por φ. Além disso, Hayashibara et al usam uma estimativa que supõe que o tempo entre chegadas das mensagens de monitoração seguem a distribuição normal. Um pouco mais de detalhes desses trabalhos serão apresentados no Capítulo 3.

[30]Do inglês: *accrual*.

2.5.5 Qualidade de Serviço de Detectores de Defeitos

A principal limitação de ADAPTATION e DPCP é a suposição de canais confiáveis, que geralmente não ocorrem na prática. Por outro lado, esta tese supõe que os processos podem quebrar mas os canais de comunicação não são confiáveis. Além disso, usamos as definições de qualidade de serviço de detectores de defeitos definidos por Chen et al [30]. Através do uso das métricas de QoS e do modelo de Chen et al, que serão descritos no Capítulo 3, apresentamos, no Capítulo 4, uma extensão ao modelo de Chen et al que aborda rajadas de perdas de mensagens.

2.6 Conclusões

Sistemas distribuídos estão espalhados pelo mundo todo devido à Internet. Contudo, para que cada sistema distribuído funcione adequadamente, as suas propriedades de *safety* e *liveness* precisam ser garantidas. Desta forma, o sistema iria sempre produzir o resultado de acordo com a sua especificação. Contudo, sistemas podem falhar devido a influências externas ou internas, ameaçando a sua dependabilidade.

Um sistema distribuído tolerante a falhas é construído com vários componentes, desde canais de comunicação confiáveis, comunicação confiável, detectores de defeitos, até replicação de processos e transações atômicas. Esta pesquisa atua particularmente no impacto da modelagem de perdas em detectores de defeitos. Pretende-se então que os detectores de defeitos tenham uma qualidade de serviço previsível, mesmo na ocorrência de rajadas de perdas de pacotes.

Capítulo 3

Qualidade de Serviço de Detectores de Defeitos

Este capítulo descreve sucintamente o trabalho de Chen et al [30] sobre a qualidade de serviço (QoS) de detectores de defeitos em sistemas distribuídos. Após detalhar alguns aspectos do referido trabalho, apresentamos alguns trabalhos relacionados e algumas conclusões. O trabalho de Chen et al é a base para o trabalho desta tese, e por isso, para mantê-la auto-contida, iremos mostrar a maior parte das definições e teoremas sobre QoS de detectores de defeitos realizadas naquele trabalho.

3.1 O Trabalho de Chen et al

Esta seção mostra sucintamente qual o modelo de sistema, o modelo probabilístico da rede, as métricas de QoS, os algoritmos de detectores de defeitos, e os configuradores propostos por Chen et al [30].

3.1.1 Modelo de Sistema

O modelo de sistema considerado possui dois processos p e q e supõe o modelo probabilístico de rede definido na Seção 3.1.2 a seguir. O processo monitorado p envia periodicamente mensagens *heartbeat* ao processo q. O processo detector de defeitos q verifica periodicamente, através do recebimento ou não de mensagens *heartbeat*, se o processo monitorado p quebrou.

3.1.2 Modelo Probabilístico da Rede

O modelo de rede é probabilístico e possui as seguintes características:

1) p e q são conectados por um canal de comunicação que não cria ou duplica mensagens, mas pode atrasar e perder mensagens;

2) *probabilidade de perda de mensagens* p_L é a probabilidade de uma mensagem ser perdida pelo canal de comunicação, e *atraso de mensagem* D é o atraso a partir do momento em que uma mensagem é enviada até o momento em que ela é recebida, desde que a mensagem não seja perdida pelo canal de comunicação;

3) o valor esperado $E(D)$ e a variância $V(D)$ de D são finitos;

4) p e q têm acesso a seus próprios relógios locais, que não possuem desvio;

5) o comportamento probabilístico da rede não muda ao passar do tempo;

6) as quebras não podem ser previstas;

7) os comportamentos de atraso e perda das mensagens que um processo envia são independentes se o processo quebrar e quando ele quebra;

8) o canal de comunicação entre p e q possui a propriedade de *independência de mensagens*: os comportamentos de quaisquer duas mensagens *heartbeat* enviadas por p são independentes.

3.1.3 Saída do Detector de Defeitos e Transições

A partir do monitoramento, q pode fornecer como saída em um instante t ou S (suspeita da quebra de p) ou T (confiança de que p está vivo). Uma mudança alternada entre estas saídas é chamada de *transição*.

Uma *transição-S* ocorre quando a saída de q muda de T para S; e uma *transição-T* ocorre quando a saída de q muda de S para T. As definições de *transição-S* e *transição-T* são usadas para definir as métricas de QoS que caracterizam a velocidade e a precisão de detectores de defeitos.

3.1.4 Métricas para a Qualidade de Serviço

As métricas de QoS são definidas como variáveis aleatórias.

Tempo de Detecção[31] (T_D) é a métrica de QoS para a velocidade do detector de defeitos, e descreve quão rápido q detecta a quebra de p. T_D representa o tempo que decorre desde

[31] Do inglês: *detection time*.

a quebra de p até que a última *transição-S* (de q) ocorra.

Há métricas de QoS para *precisão* que descrevem quão bem q evita erros. Um erro ocorre quando q fornece S, mas p ainda está vivo.

As duas métricas de precisão primárias são:

1) *Tempo de recorrência de erros*[32] (T_{MR}): o tempo que decorre entre duas *transições-S* consecutivas. Isto indica quanto tempo o detector de defeitos q gasta entre dois erros sobre a quebra de p; e

2) *Duração de erros*[33] (T_M): o tempo que decorre desde uma *transição-S* até a próxima *transição-T*. Isto indica quanto tempo o detector de defeitos q gasta para mudar a informação incorreta sobre a quebra de p.

Junto a estas duas métricas primárias, há quatro métricas de precisão que são derivadas de T_{MR} e T_M:

1) *Taxa de erros média*[34] (λ_M): a taxa na qual um detector de defeitos erra;

2) *Probabilidade de consulta com precisão*[35] (P_A): a probabilidade de que a saída do detector de defeitos está correta em um instante de tempo aleatório;

3) *Duração de um período bom*[36] (T_G): o tempo que decorre desde uma *transição-T* até a próxima *transição-S*; e

4) *Duração de um período bom adiante*[37] (T_{FG}): o tempo que decorre desde um instante de tempo aleatório em que q confia que p está vivo até o instante de tempo da próxima *transição-S*.

3.1.5 Relacionamento entre as Métricas de QoS

O Teorema 1 de Chen et al, a seguir, relaciona as seis métricas de precisão. $Pr(A)$ denota a probabilidade do evento A; $E(X)$, $E(X^k)$, e $V(X)$ denotam o valor esperado (ou a média), o k-ésimo momento, e a variância da variável aleatória X, respectivamente. Uma história de detector de defeitos é uma seqüência de saídas (S ou T) que o detector de defeitos p fornece. Em execuções sem defeito, um detector de defeitos ergódico é aquele que fornece histórias que seguem uma distribuição probabilística ergódica. Isto significa que, em execuções sem defeito, o detector de defeitos lentamente "*esquece*" da sua história

[32] Do inglês: *mistake recurrence time.*
[33] Do inglês: *mistake duration.*
[34] Do inglês: *average mistake rate.*
[35] Do inglês: *query accuracy probability.*
[36] Do inglês: *good period duration.*
[37] Do inglês: *forward good period duration.*

passada: a partir de qualquer tempo adiante, seu comportamento futuro pode depender apenas do seu comportamento recente.

Teorema 1. *Para qualquer detector de defeitos ergódico, os seguintes resultados valem:*

1. $T_G = T_{MR} - T_M$.
2. *Se* $0 < E(T_{MR}) < \infty$, *então* $\lambda_M = 1/E(T_{MR})$ *e* $P_A = E(T_G)/E(T_{MR})$.
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*

   3a) Para todo $x \in [0, \infty)$, $Pr(T_{FG} \leq x) = \int_0^x Pr(T_G > y)dy/E(T_G)$.

   3b) $E(T_{FG}^k = E(T_G^{k+1})/[(k+1)E(T_G)]$.

   3c) Em particular, $E(T_{FG}) = [1 + V(T_G)/E(T_G)^2]E(T_G)/2$.

3.1.6 Algoritmo NFD-S

A Figura 3.1 mostra o algoritmo NFD-S (novo detector de defeitos para relógios sincronizados) de Chen et al que supõe relógios sincronizados. NFD-S possui dois parâmetros: η e δ. p envia mensagens *heartbeat* $m_1, m_2, \ldots$ para q a cada η unidades de tempo. σ_i denota o instante de envio da mensagem m_i. q desloca os σ_i's adiante por δ para obter a seqüência de tempos $\tau_1 < \tau_2 < \ldots$, onde $\tau_i = \sigma_i + \delta$, para $i \geq 1$. Para $i = 0$, $\tau_0 = 0$. q usa os τ_i's e os tempos que ele recebe as mensagens *heartbeat* para determinar se ele confia ou suspeita da quebra de p, a cada período de tempo $[\tau_i, \tau_{i+1})$. A partir de τ_i até τ_{i+1}, somente as mensagens m_j com $j \geq i$ podem afetar a saída de q.

3.1.7 Análise Básica de QoS de NFD-S

A análise básica de QoS de NFD-S supõe o Lema 2 e a Proposição 13, que estão descritos a seguir. O Lema 2 determina quais mensagens podem alterar a saída do detector de defeitos. A Proposição 13 descreve quais mensagens levam o detector de defeitos a produzir uma *transição-S*.

Lema 2. *Para todo* $i \geq 0$ *e todo tempo* $t \in [\tau_i, \tau_{i+1})$, q *confia em* p *no tempo* t *se e somente se* q *recebeu alguma mensagem* m_j *com* $j \geq i$ *até o tempo* t.

Process p:
1 for all $i \geq 1$, at time $\sigma_i = i\eta$, send heartbeat m_i to q;
Process q:
2 Initialization: $output = S$; {suspeita de p inicialmente}
3 for all $i \geq 1$, at time $\tau_i = \sigma_i + \delta$:
4 **if** did not receive m_j with $j \geq i$ **then** $output \leftarrow S$; {suspeita de p se nenhuma mensagem válida é recebida}
5 upon receive message m_j at time $t \in [\tau_i, \tau_{i+1})$:
6 **if** $j \geq i$ **then** $output \leftarrow T$; {confia em p quando alguma mensagem válida é recebida}

Figura 3.1: Algoritmo NFD-S com parâmetros η e δ (relógios sincronizados).

Proposição 13. *1) Uma transição-S pode ocorrer somente no tempo τ_i para algum $i \geq 2$ e ela ocorre em τ_i se e somente se a mensagem m_{i-1} é recebida por q antes do tempo τ_i e nenhuma mensagem m_j com $j \geq i$ é recebida por q até o tempo τ_i; 2) Lema 2 permanece verdadeiro se $j \geq i$ na sentença for substituído por $i \leq j \leq i + k$; 3) parte 1) acima permanece verdadeira se $j \geq i$ na sentença for substituído por $i \leq j < i + k$.*

A análise básica de QoS de Chen et al está descrita pela Definição 1, Proposição 3, e Teorema 5, que estão a seguir.

Definição 1.

1. *Para qualquer $i \geq 1$, seja k o menor inteiro tal que, para todo $j \geq i + k$, m_j é enviada no ou após o tempo τ_i.*

2. *Para qualquer $i \geq 1$, seja $p_j(x)$ a probabilidade que q não receba a mensagem m_{i+j} no tempo $\tau_i + x$, para cada $j \geq 0$ e cada $x \geq 0$; seja $p_0 = p_0(0)$.*

3. *Para qualquer $i \geq 2$, seja q_0 a probabilidade que q receba a mensagem m_{i-1} antes do tempo τ_i.*

4. *Para qualquer $i \geq 1$, seja $u(x)$ a probabilidade que q suspeita de p no tempo $\tau_i + x$, para cada $x \in [0, \eta)$.*

5. *Para qualquer $i \geq 2$, seja p_s a probabilidade que uma transição-S ocorra no tempo τ_i.*

A Proposição 3 abaixo mostra que a Definição 1 pode ser expressa de forma independente de i. p_L é a probabilidade de perda de heartbeats. $Pr(D > y)$ e $Pr(D < y)$ são respectivamente, a probabilidade que uma mensagem atrase mais que y, e a probabilidade que uma mensagem atrase menos que y.

Proposição 3.

1. $k = \lceil \delta/\eta \rceil$.
2. *Para todo* $j \geq 0$ *e para todo* $x \geq 0$,
   $p_j(x) = p_L + (1 - p_L)Pr(D > \delta + x - j\eta)$.
3. $q_0 = (1 - p_L)Pr(D < \delta + \eta)$.
4. *Para todo* $x \in [0, \eta)$, $u(x) = \prod_{j=0}^{k} p_j(x)$.
5. $p_s = q_0 u(0)$.

Deste ponto em diante são considerados os casos não degenerados, ou seja, quando $p_0 > 0$ e $q_0 > 0$. A Proposição 14 a seguir apresenta a relação entre $u(0)$ e $u(x)$, que é usada na prova da Proposição 21 da seção 3.1.8.

Proposição 14. $u(0) \geq p_0^k$ *e, para todo* $x \in [0, \eta)$, $u(0) \geq u(x)$.

O Lema 15 a seguir, que mostra o cálculo da probabilidade de consulta com precisão, é usada para encontrar o valor de $E(T_M)$.

Lema 15. $P_A = 1 - \frac{1}{\eta} \int_0^\eta u(x)dx$.

Teorema 5. *Considere um sistema com relógios sincronizados, onde a probabilidade de perda de mensagens é* p_L *e a distribuição de atrasos de mensagens é* $Pr(D \leq x)$. *O detector de defeitos NFD-S com parâmetros* η *e* δ *possui as seguintes propriedades:*

1. *O tempo de detecção é limitado como segue e o limite é estrito:*

$$T_D \leq \delta + \eta. \tag{3.1}$$

2. *O tempo de recorrência de erros médio é:*

$$E(T_{MR}) = \frac{\eta}{p_s}. \tag{3.2}$$

3. *A duração de erros média é:*

$$E(T_M) = \frac{\int_0^\eta u(x)d(x)}{p_s}. \tag{3.3}$$

3.1.8 Configurador para NFD-S

Esta seção é um resumo da seção 4 de Chen et al. Eles supõem que: 1) os relógios locais dos processos estão sincronizados e 2) sabe-se o comportamento probabilístico das mensagens, ou seja, a probabilidade de perda de mensagens p_L e a distribuição dos atrasos de mensagens $Pr(D \leq x)$.

O objetivo é encontrar um procedimento de configuração, daqui em diante chamado de configurador, que possui como entrada essas premissas e os requisitos de QoS (T_D^U, T_{MR}^L, T_M^U). T_D^U é um limite superior sobre o tempo de detecção, T_{MR}^L é um limite inferior sobre o tempo de recorrência de erros médio, e T_M^U é um limite superior sobre a duração de erros média. Em outras palavras, os requisitos de QoS são:

$$T_D \leq T_D^U, E(T_{MR}) \geq T_{MR}^L, E(T_M) \leq T_M^U. \tag{4.1}$$

A partir dessa entrada, o configurador fornece como saída *QoS não pode ser alcançada*, ou os parâmetros η e δ satisfazem os requisitos de QoS. Para minimizar a largura de banda de rede usada pelo detector de defeitos, o configurador pretende encontrar o maior intervalo entre envios de *heartbeats* η que satisfaça estes requisitos de QoS.

A partir do Teorema 5, o objetivo pode ser reescrito como um problema de programação matemática:

$$maximizar\ \eta$$

$$sujeito\ a\ \delta + \eta \leq T_D^U \tag{4.2}$$

$$\frac{\eta}{p_s} \geq T_{MR}^L \tag{4.3}$$

$$\frac{\int_0^\eta u(x)dx}{p_s} \leq T_M^U, \tag{4.4}$$

onde os valores de $u(x)$ e p_s são dados pela Proposição 3. Chen et al substituíram a asserção (4.4) pela restrição mais simples e mais forte a seguir.

Proposição 21. *Se $p_0 > 0$ e $q_0 > 0$ (o caso não degenerado), então $E(T_M) \leq \eta/q_0$.*

Configurador Chen-NFD-S. Então o configurador para encontrar η e δ resulta em:

- *Passo 1*: Computar $q_0' = (1 - p_L)Pr(D < T_D^U)$ e seja $\eta_{max} = q_0' T_M^U$. Se $\eta_{max} = 0$, então fornecer como saída *"QoS não pode ser alcançada"* e parar; senão continuar.

- *Passo 2*: Seja $f(\eta) = \eta/(q'_0 \prod_{j=1}^{\lceil T_D^U/\eta \rceil - 1}[p_L + (1-p_L)Pr(D > T_D^U - j\eta)])$ (4.5)
  Encontrar o maior $\eta \leq \eta_{max}$ tal que $f(\eta) \geq T_{MR}^L$. Tal η sempre existe.
- *Passo 3*: Atribuir $\delta = T_D^U - \eta$ e fornecer como saída η e δ.

A Figura 3.2 apresenta um esquema da configuração de NFD-S. O estimador de comportamento probabilístico de *heartbeats* fornece estimativas para p_L, $E(D)$ e $V(D)$. Essas estimativas, juntamente com os requisitos de QoS, são fornecidos ao configurador, que fornece então os parâmetros η e δ para NFD-S.

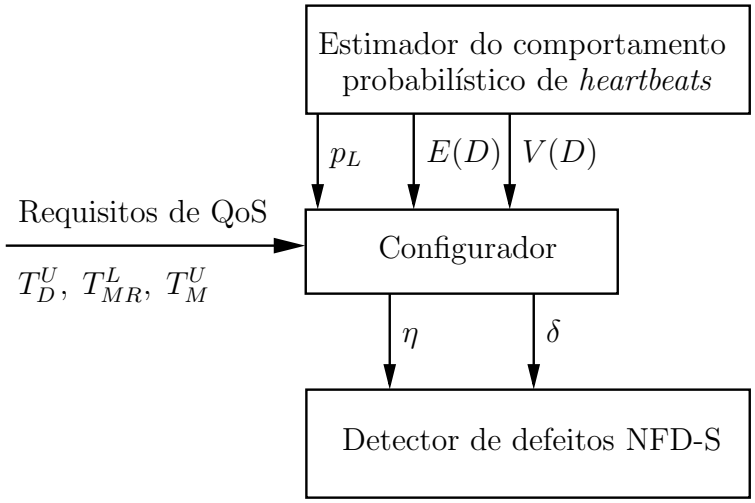


Figura 3.2: Configuração de NFD-S.

Teorema 7. *Considere um sistema em que os relógios são sincronizados e o comportamento probabilístico das mensagens é conhecido. Suponha que seja dado um conjunto de requisitos de QoS como em (4.1). O configurador acima possui dois resultados possíveis: 1) Ele fornece como saída η e δ. Neste caso, com parâmetros η e δ, o detector de defeitos NFD-S satisfaz os requisitos de QoS fornecidos. 2) Ele fornece como saída "QoS não pode ser alcançada". Neste caso,* nenhum *detector de defeitos pode alcançar os requisitos de QoS solicitados.*

3.1.9 Lidando com Comportamento de Mensagens Desconhecido

Esta seção é um resumo da seção 5 de Chen et al. Eles supõem que: 1) os relógios locais dos processos são sincronizados e 2) o comportamento probabilístico das mensagens

é desconhecido. Neste caso ainda é possível computar η e δ usando apenas p_L, $E(D)$, e $V(D)$.

Para supor essas premissas, é usada a seguinte *Desigualdade Unilateral* da teoria das probabilidades: para qualquer variável aleatória D com um valor esperado finito e uma variância finita,

$$Pr(D > t) \leq \frac{V(D)}{V(D)+(t-E(D))^2}, \text{ para todo } t > E(D). \tag{5.1}$$

Com isso, alguns limites sobre as métricas de QoS do algoritmo NFD-S podem ser derivados, da forma descrita no Teorema 9 a seguir.

Teorema 9. *Considere um sistema com relógios sincronizados e suponha que* $\delta > E(D)$. *Para o algoritmo NFD-S, nós temos* $E(T_{MR}) \geq \eta/\beta$ *e* $E(T_M) \leq \eta/\gamma$, *onde*

$$\beta = \prod_{j=0}^{k_0} \frac{V(D) + p_L(\delta - E(D) - j\eta)^2}{V(D) + (\delta - E(D) - j\eta)^2}, k_0 = \lceil (\delta - E(D))/\eta \rceil - 1,$$

e

$$\gamma = \frac{(1-p_L)(\delta - E(D) + \eta)^2}{V(D) + (\delta - E(D) + \eta)^2}.$$

O seguinte configurador supõe que $T_D^U > E(D)$:

- *Passo 1*: Computar $\gamma' = (1-p_L)(T_D^U - E(D))^2/(V(D) + (T_D^U - E(D))^2)$ e seja $\eta_{max} = min(\gamma' T_M^U, T_D^U - E(D))$. Se $\eta_{max} = 0$, então fornecer como saída *"QoS não pode ser alcançada"* e parar; senão continuar.
- *Passo 2*: Seja $f(\eta) = \eta \cdot \prod_{j=1}^{\lceil (T_D^U - E(D))/\eta \rceil - 1} \frac{V(D)+(T_D^U - E(D) - j\eta)^2}{V(D)+p_L(T_D^U - E(D) - j\eta)^2}$. (5.2)
  Encontrar o maior $\eta \leq \eta_{max}$ tal que $f(\eta) \geq T_{MR}^L$. Tal η sempre existe.
- *Passo 3*: Atribuir $\delta = T_D^U - \eta$ e fornecer como saída η e δ.

Teorema 10. *Considere um sistema em que os relógios são sincronizados e o comportamento probabilístico das mensagens não é conhecido. Suponha que seja fornecido um conjunto de requisitos de QoS como em (4.1). O configurador acima possui dois possíveis resultados: 1) Ele fornece como saída* η *e* δ. *Neste caso, com parâmetros* η *e* δ, *o detector de defeitos NFD-S satisfaz os requisitos de QoS solicitados. 2) Ele fornece como saída "QoS não pode ser alcançada". Neste caso,* nenhum *detector de defeitos pode alcançar os requisitos de QoS solicitados.*

3.1.10 Lidando com Comportamento de Mensagens Desconhecido e Relógios não Sincronizados

Esta seção é um resumo da seção 6 de Chen et al. O detector de defeitos NFD-S supõe relógios sincronizados (por exemplo, via NTP). Desta forma, esta seção mostra como construir um configurador para o detector de defeitos NFD-U, modificado a partir de NFD-S. O NFD-U (veja a Figura 3.3), possui dois parâmetros: η e α. A única diferença entre os algoritmos NFD-U e o NFD-S (veja a seção 3.1.6) é a forma que eles definem os τ_is: no NFD-S, $\tau_i = \sigma_i + \delta$, enquanto, no NFD-U, $\tau_i = EA_i + \alpha = \sigma_i + E(D) + \alpha$, onde $EA_i = \sigma_i + E(D)$).

```
Process p:  {usando o relógio local de p}
1   for all i ≥ 1, at time iη, send heartbeat m_i to q;
Process q:  {usando o relógio local de q}
2   Initialization:
3      τ_0 = 0;
4      l = −1;                          {l guarda o maior número de seqüência de todas
                                         as mensagens recebidas de q até o momento}
5   upon τ_{l+1} = current time:        {se o tempo alcança τ_{l+1}, então nenhuma das
                                         mensagens recebidas ainda é válida}
6      output ← S;                      {suspeita de p já que nenhuma mensagem
                                         recebida ainda é válida neste momento}
7   upon receive message m_j at time t:
8      if j > l then {uma mensagem com um número de seqüência maior foi recebida}
9         l ← j;
10        τ_{l+1} ← EA_{l+1} + α;       {define o próximo ponto τ_{l+1} usando o
                                         tempo de chegada esperado de m_{l+1}}
11        if t < τ_{l+1} then output ← T;  {confia em p já que m_l ainda é válida no tempo t}
```

Figura 3.3: Algoritmo NFD-U com parâmetros η e α (relógios não sincronizados, mas EA_is são conhecidos).

Substituindo δ por $E(D) + \alpha$ no Teorema 9, os seguintes limites nas métricas de precisão de NFD-U podem ser obtidas:

Teorema 11. *Considere um sistema com relógios não sincronizados e suponha que*

$\alpha > 0$. *Para o algoritmo NFD-U, nós temos* $E(T_{MR}) \geq \eta/\beta$ *e* $E(T_M) \leq \eta/\gamma$, *onde*

$$\beta = \prod_{j=0}^{k_0} \frac{V(D) + p_L(\alpha - j\eta)^2}{V(D) + (\alpha - j\eta)^2},\ k_0 = \lceil (\alpha/\eta \rceil - 1,\ \text{e}\ \gamma = \frac{(1-p_L)(\alpha+\eta)^2}{V(D) + (\alpha+\eta)^2}.$$

Teorema 11 pode ser usado para computar os parâmetros η e α do detector de defeitos NFD-U de modo que ele satisfaça os seguintes requisitos de QoS:

$$T_D \leq T_D^u + E(D), E(T_{MR}) \geq T_{MR}^L, E(T_M) \leq T_M^U. \tag{6.1}$$

Perceba que o limite superior do tempo de detecção T_D *não é* T_D^u, mas T_D^u mais o atraso médio de mensagens *desconhecido* $E(D)$. Logo, o limite superior real T_D^U no tempo de detecção é $T_D^u + E(D)$.

3.1.11 Configurador para NFD-U

Configurador Chen-NFD-U. O configurador para o NFD-U torna-se então:

- *Passo 1*: Computar $\gamma' = (1-p_L)(T_D^u)^2/(V(D)+(T_D^u)^2)$ e seja $\eta_{max} = min(\gamma' T_M^U, T_D^u)$. Se $\eta_{max} = 0$, então forneça como saída *"QoS não pode ser alcançada"* e parar; senão continuar.

- *Passo 2*: Seja $f(\eta) = \eta \cdot \prod_{j=1}^{\lceil T_D^u/\eta \rceil - 1} \frac{V(D)+(T_D^u - j\eta)^2}{V(D)+p_L(T_D^u - j\eta)^2}$. (6.2)
  Encontrar o maior $\eta \leq \eta_{max}$ tal que $f(\eta) \geq T_{MR}^L$. Tal η sempre existe.

- *Passo 3*: Atribuir $\alpha = T_D^u - \eta$ e fornecer como saída η e α.

Teorema 12. *Considere um sistema com relógios não sincronizados, onde o comportamento probabilístico das mensagens não é conhecido. Suponha que seja fornecido um conjunto de requisitos de QoS como em (6.1). O configurador acima possui dois resultados possíveis: 1) Ele fornece como saída* η *e* α. *Neste caso, com parâmetros* η *e* α, *o detector de defeitos NFD-U satisfaz os requisitos de QoS solicitados. 2) Ele fornece como saída "QoS não pode ser alcançada". Neste caso,* nenhum *detector de defeitos pode alcançar os requisitos de QoS solicitados.*

A Figura 3.4 apresenta um esquema da configuração de NFD-U. O estimador de comportamento probabilístico de *heartbeats* fornece apenas estimativas para p_L e $V(D)$, visto que o configurador não usa $E(D)$. Diferente do configurador para NFD-S, o configurador para NFD-U usa T_D^u no lugar de T_D^U. As estimativas e os requisitos de QoS

são fornecidos ao configurador, que fornece os parâmetros η e α para NFD-U. NFD-U usa esses parâmetros mais os EA_{l+1}'s, que são conhecidos.

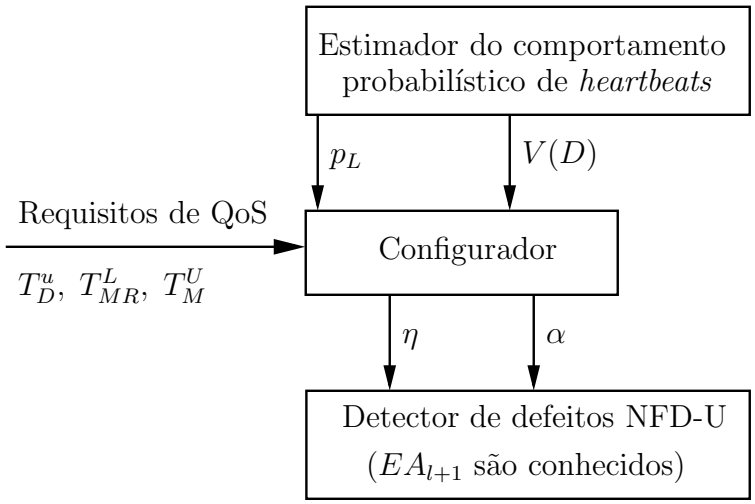


Figura 3.4: Configuração de NFD-U.

A seção 6.2.2 de Chen et al também mostra como estimar p_L, e $V(D)$. A seção 6.3 de Chen et al mostra o algoritmo NFD-E, que modifica o algoritmo NFD-U somente sobre os tempos de chegada esperados. Enquanto NFD-U supõe que q sabe o valor exato de todos os EA_is, NFD-E os estima.

Desta forma, cada vez que q executa a linha 10 do algoritmo NFD-U, q considera as n mais recentes mensagens *heartbeat*, denotadas por $m'_1, \ldots, m'_n$. Seja $s_1, \ldots, s_n$ os números de seqüência de tais mensagens e $A'_1, \ldots, A'_n$ os seus instantes de recebimento de acordo com o relógio local de q. Logo, EA_{l+1} é estimado por:

$$EA_{l+1} \approx \frac{1}{n}\left(\sum_{i=1}^{n} A'_i - \eta s_i\right) + (l+1)\eta.$$

Eles mostraram, a partir de simulações, que NFD-E e NFD-U são praticamente indistinguíveis para valores de n tão baixos quanto 30.

Configurador Chen-NFD-E. A Figura 3.5 apresenta um esquema da configuração de NFD-E. A única diferença com a configuração de NFD-U (Figura 3.4) é que o estimador do comportamento probabilístico de *heartbeats* também fornece estimativas de EA_{l+1}'s para NFD-E.

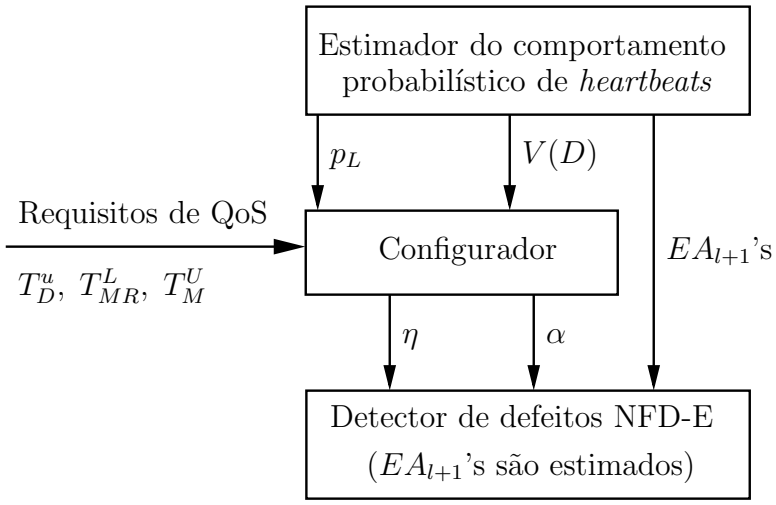


Figura 3.5: Configuração de NFD-E.

3.2 Trabalhos Relacionados

Pesquisas sobre a aplicação de métricas de QoS de detectores de defeitos estão apenas iniciando. Os trabalhos de Bertier et al [20, 21], Nunes e Jansch-Pôrto [85, 86, 87], Hayashibara et al [60], que foram discutidos na seção 2.5.4 no contexto de detectores de defeitos adaptativos, também usam as métricas de QoS de Chen et al e são discutidos nas subseções a seguir.

3.2.1 Trabalhos de Bertier

Bertier et al. [20, 21] usam a duração de erros média $E(T_M)$ para avaliar o seu detector de defeitos. Em Bertier et al. [20], a avaliação, usando dados reais, mostra que a estimação dinâmica proposta leva a um tempo de detecção menor que o de Chen et al, a custas de um pequeno aumento no número de falsas suspeitas e na duração de erros média em relação a Chen et al.

O estimador de Jacobson foi melhor que a deles e a de Chen et al na duração de erros média e no tempo de detecção médio, mas produziu mais do que o dobro de falsas suspeitas do que o estimador dinâmico proposto.

Em Bertier et al. [21], a avaliação, usando dados reais, mostra que o uso de adaptadores da QoS para atender a QoS de vários processos pode levar a um tempo de detecção maior, mas com muito menos falsas suspeitas e uma duração de erros média bem menor. Um

desses adaptadores propostos usa uma modificação do configurador para NFD-U de Chen et al.

3.2.2 Trabalhos de Nunes e Jansch-Pôrto e de Falai e Bondavalli

Nunes e Jansch-Pôrto [87] usam as métricas (T_D, T_M, T_{MR}, λ_M, P_A) de QoS para avaliar alguns preditores para atrasos de mensagens e a combinação destes com o cálculo de margens de segurança. Eles usaram os preditores MEAN (o menos preciso entre estes três), WINMEAN e ARIMA (o mais preciso entre estes três) para o cálculo do atraso de mensagens, e as margens de segurança variáveis *peb* e *cib*.

MEAN usa a média de todos os atrasos já percebidos, WINMEAN usa a média dos últimos 30 atrasos percebidos, e ARIMA usa um modelo de séries temporais com média móvel, integrada e auto-regressiva. *peb*[38] é recalculado para acomodar cada erro do preditor, e *cib*[39] supõe que o preditor modela adequadamente os atrasos.

Um resultado importante que eles mostram, supondo o modelo *pull*, é que num cenário de margem de segurança constante e idêntica para os três preditores, a melhora da precisão do preditor de atrasos: a) não implica necessariamente em menor tempo de detecção, e b) não implica necessariamente na melhoria da precisão do detector de defeitos. ARIMA+*cib* levou o detector de defeitos a apresentar um tempo de detecção, no pior caso, maior que MEAN+*cib*.

Contudo, ARIMA+*peb* levou o detector de defeitos a apresentar um tempo de detecção, no pior caso, menor que MEAN+*peb*. MEAN+*cib* levou o detector de defeitos a ter o menor T_M, mas com o menor T_{MR} entre os três preditores. ARIMA+*peb* e WINMEAN+*peb* levaram o detector de defeitos a ter o menor T_M, mas com o menor T_{MR} em relação a MEAN. MEAN+*peb* ou WINMEAN+*cib* ou ARIMA+*cib* são as melhores combinações para P_A.

Outro resultado importante é que a combinação de preditor MEAN com margem de segurança *peb* leva o detector de defeitos a ser mais preciso, enquanto a combinação preditor ARIMA com margem de segurança *cib* resulta em uma melhor relação precisão/desempenho para o detector de defeitos. Eles mostram que a combinação preditor-margem de segurança é importante para obter uma boa QoS e que a escolha dessa combinação depende dos requisitos da aplicação.

[38] Acrônimo para o termo *prediction error based margin*.
[39] Acrônimo para o termo *confidence interval based margin*.

Falai e Bondavalli [47] fizeram uma análise similar à realizada por Nunes e Jansch-Pôrto [87]. Contudo, Falai e Bondavalli usaram apenas as métricas (T_D, T_M, T_{MR}, e P_A) de QoS para avaliar alguns preditores para atrasos de mensagens e a combinação destes com o cálculo de margens de segurança. Eles usaram os preditores MEAN, WINMEAN, ARIMA, LAST, e LPF para o cálculo do atraso de mensagens, e as margens de segurança variáveis SM_{CI} e SM_{JAC}.

MEAN usa a média de todos os atrasos já percebidos, WINMEAN usa a média dos últimos 10 atrasos percebidos, e ARIMA usa um modelo de séries temporais com média móvel, integrada e auto-regressiva, LAST usa o último atraso percebido, e LPF usa a suavização exponencial da última previsão com os últimos atrasos. As margens de segurança SM_{CI} e SM_{JAC} são similares, respectivamente, às margens de segurança *cib*, e *peb* de Nunes e Jansch-Pôrto [87]. *cib* usa apenas valor constante de 2,58, e SM_{CI} pode assumir os valores constantes 1,648, 2,586 e 3,31. *peb* usa apenas o valor constante 1, e SM_{JAC} pode assumir os valores constantes 1, 2 e 4.

Ele confirmam, supondo o modelo *push* (ao contrário de Nunes e Jansch-Pôrto que usam o modelo *pull*), no cenário de margem de segurança constante e idêntica para os três preditores, de forma similar a Nunes e Jansch-Pôrto, que a combinação preditor-margem de segurança é importante para obter uma boa QoS e que a escolha dessa combinação depende dos requisitos da aplicação, e que a melhora da precisão do preditor de atrasos: a) não implica necessariamente em menor tempo de detecção, e b) não implica necessariamente na melhoria da precisão do detector de defeitos.

MEAN levou o detector de defeitos a apresentar um tempo de detecção, médio e no pior caso, maior que os demais preditores. LAST+SM_{JAC} é a melhor combinação para menor tempo de detecção, médio e no pior caso, que os demais preditores. LAST+SM_{JAC} é a melhor combinação para um baixo T_M, mas oferece o pior T_{MR}. LAST+SM_{CI} é a melhor combinação para um alto T_{MR}, mas oferece o pior T_M. ARIMA+SM_{CI} é a melhor combinação para um alto P_A, mas ARIMA+SM_{JAC} fornece o pior P_A. LAST+SM_{CI} é a segunda melhor combinação para um alto P_A, e LAST+SM_{JAC} fornece o melhor P_A (na nossa avaliação pois com SM_{JAC}, LAST fornece os dois melhores valores dentre as demais). Para uma comparação mais precisa entre os dois trabalhos, pode-se usar, respectivamente, os resultados com *cib* e *peb* de Nunes e Jansch-Pôrto apenas com os resultados obtidos com SM_{CI} e valor constante 2,586 (referida como CI_{med} nas figuras), e com SM_{JAC} e valor constante 1 (referida como JAC_{low} nas figuras) de Falai e Bondavalli.

3.2.3 Trabalhos de Hayashibara

Hayashibara et al. [60] usam a taxa de erros média (λ_M) para avaliar o seu detector de defeitos acumulador φ em relação aos detectores de defeitos de Chen et al [30] (NFD-E) e de Bertier et al [20]. A avaliação foi realizada por simulação usando um traço[40] real em uma WAN e um traço real em uma LAN.

Na WAN, o desempenho dos detectores de defeitos φ e NFD-E são similares, enquanto o de Bertier et al não funciona bem (porque ele foi projetado para uma LAN). Na LAN, o desempenho de φ supera o detector de defeitos NFD-E e é similar ao de Bertier et al quando o tempo de detecção é bem agressivo (abaixo de 1s). Contudo, com valores de tempo de detecção acima de 1,4 s, φ e o detector de defeitos NFD-E são similares.

Défago et al [37] apresentam a definição formal de detectores de defeitos *accrual* através das propriedades de *accruement* (para processos falhos) e *upper bound* (para processos corretos). Eles também relacionam detectores de defeitos *accrual*, com diferentes funções limite[41], através das métricas T_D, T_{MR}, λ_M e T_G. Défago et al apresentam várias implementações de detectores de defeitos *accrual*, como φ e κ. O detector de defeitos κ muda os níveis de suspeita de acordo com perdas de mensagens e a variabilidade de tempos de chegada de mensagens.

3.3 Conclusões

Neste capítulo descrevemos sucintamente o trabalho de Chen et al [30], sobre QoS de detectores de defeitos. Apresentamos trabalhos relacionados que usam as métricas propostas. A partir dos trabalhos relacionados é possível perceber que as métricas estão sendo cada vez mais usadas para a avaliação de detectores de defeitos, tanto para verificar estimadores/preditores quanto algoritmos de detecção diferentes. Percebe-se, a partir dos trabalhos de Nunes e Jansch-Pôrto [87] e de Falai e Bondavalli [47], que a relação entre T_{MR} e T_M age como uma balança: quando um melhora (T_{MR} aumenta) o outro piora (T_M aumenta). Além desse problema nas métricas de precisão, nem sempre uma combinação preditor-margem de segurança, que é boa para precisão, é boa para tempo de detecção.

Considerando o melhor conhecimento nosso, o único trabalho, após o trabalho de Chen et al, que configura o detector de defeitos a partir de requisitos de QoS e então verifica

[40] Do inglês: *trace*.
[41] Do inglês: *threshold*.

se o detector de defeitos realmente satisfaz os requisitos, considerando explicitamente a ocorrência de perdas consecutivas de *heartbeats*, é esta pesquisa. O capítulo 4 a seguir apresenta o modelo proposto.

O trabalho anterior que mais se aproximaria a esta pesquisa seria o trabalho de Bertier et al [21]. Mas as rajadas de perdas não são consideradas explicitamente visto que o objetivo deles é o uso do detector de defeitos adaptativo em uma LAN, que possui pouca ocorrência de rajadas de perdas de pacotes.

Capítulo 4

QoS de Detectores de Defeitos na Presença de Rajadas de Perdas

Este Capítulo apresenta o modelo proposto para QoS de detectores de defeitos na presença de rajadas de perdas de mensagens. A modularidade fornecida pelo trabalho de Chen et al [30] facilitou o desenvolvimento desta proposta. Como poderá ser observado, essa característica foi aproveitada para "trocar" os módulos teóricos para estender os resultados do trabalho deles para sistemas em que ocorrem rajadas de perdas com as mais diversas distribuições de probabilidade.

4.1 Hipóteses da Proposta

Esta seção apresenta as hipóteses do modelo proposto, que basicamente estende o modelo de Chen et al com as informações sobre as rajadas de perdas.

4.1.1 Modelo de Sistema

O modelo de sistema é o mesmo daquele apresentado na seção 3.1.1 e supõe o modelo probabilístico de rede da Seção 4.1.2 a seguir. O processo monitorado p enviando mensagens *heartbeat* ao detector de defeitos q, através de um canal de comunicação não confiável.

4.1.2 Modelo Probabilístico Modificado para a Rede

O modelo probabilístico de rede considerado no modelo proposto é o mesmo de Chen et al (veja a seção 3.1.2), exceto pelas seguinte mudanças:

1) Além da probabilidade de perda de mensagens p_L e o atraso de mensagens D, o canal de comunicação entre p e q também possui uma distribuição de probabilidades de comprimentos de rajadas de perdas, dada por todos os $p_{L,z}$'s (de acordo com a Tabela 4.1 da seção 4.2). z é o comprimento de uma rajada de perdas.

2) A propriedade de independência de mensagens não é requerida. Pode haver um comportamento independente de quaisquer duas mensagens, ou um comportamento dependente de cada mensagem somente com a sua predecessora.

4.1.3 Métricas, Relacionamento entre as Métricas, Algoritmos NFD-S e NFD-U

Este Capítulo supõe as mesmas métricas de QoS, relacionamento entre as métricas de acordo com o Teorema 1, e os algoritmos NFD-S, NFD-U e NFD-E já vistos no Capítulo 3.

4.2 Modelo de Markov para as Rajadas de Perdas

O modelo de Markov de Sanneck [92] (veja a seção 2.3.3) é a base para a Definição 2 a seguir. O modelo de Markov proposto possui h estados (1 a h) para perdas de mensagens, e o estado 0 para chegada de mensagens. Uma seqüência de chegadas de mensagens mantém a cadeia de Markov no estado 0. A cada mensagem consecutiva que é perdida, há uma nova mudança de estado na cadeia de Markov, até o comprimento máximo de rajadas percebido (estado h). Devido a esta definição de h, a probabilidade de uma transição de estado para um estado maior que h é zero. Em qualquer estado, se uma mensagem é recebida, uma transição para o estado 0 ocorre. O configurador NSM-NFD-S proposto, a ser apresentado na seção 4.4, supõe que toda a informação na Definição 2 já está disponível quando o configurador é ativado. Isto é possível devido às aproximações das probabilidades de estado e de transição de estado, de acordo com a Tabela 4.1 descrita

ao final desta seção.

Definição 2.

1. *Z_n é uma seqüência de variáveis aleatórias com valores dentro do espaço $F = \{0, 1\}$. $Z_n = 0$ significa que uma mensagem foi recebida por q, e $Z_n = 1$ significa que uma mensagem foi perdida.*

2. *h é o maior comprimento de rajada de perdas que tem sido percebido por q até o instante de tempo em que o configurador é ativado.*

3. *$S = [0, h]$, com $S \subseteq \mathbf{N}$, é o conjunto de estados possíveis na cadeia de Markov.*

4. *$X_{n+1} = f(X_n, Z_{n+1})$ é a variável aleatória que define uma cadeia de Markov, com $X_n \in S$, e X_0 é o primeiro estado observado. Se $X_n < h$, então $X_{n+1} = Z_{n+1}X_n + Z_{n+1}$; senão se $X_n = h$, então $X_{n+1} = 0$, para $Z_{n+1} = 0$. Se $X_n = h$ e $Z_{n+1} = 1$ a cadeia de Markov não está definida, de acordo com o item 2.*

5. *O significado da variável aleatória X_{n+1}, definida no item 4, é como segue: $X_{n+1} = 0$ significa nenhuma mensagem perdida, $X_{n+1} = z$ $(0 < z \leq h)$ significa exatamente z mensagens consecutivas perdidas, $X_{n+1} \geq z$ significa pelo menos z mensagens consecutivas perdidas.*

6. *O significado das probabilidades de estado e de transição de estado, definidas no item 4, é como segue. As transições de estado ocorrem dependendo das probabilidades de transição p_{ij}, com $i < j$ (para comprimentos de rajadas de perdas menores ou iguais a h) ou $i \geq j = 0$ (para uma chegada de mensagem). A probabilidade de estado do sistema para $0 < z \leq h$ é $Pr(X_{n+1} \geq z)$, e para $z = 0$ é $Pr(X_{n+1} = 0)$.*

As Definições 3 e 4, a seguir, simplificam a notação para as probabilidades de transição de estado da cadeia de Markov da Definição 2. A Definição 3 mostra a probabilidade de realizar transições de estado correspondentes a uma rajada de perdas de comprimento $es - bs + 1$.

Definição 3. *A probabilidade de transições de estado adiante, de um estado bs para um estado $es \geq bs$, é definida como $forw(bs, es) = \prod_{n=bs}^{es-1} p_{n(n+1)}$. A partir da* Definição 2, *quando $n \geq h$, $forw(bs, es) = 0$.*

A Definição 4 mostra a probabilidade de realizar uma transição para o estado 0 quando uma mensagem é recebida após uma rajada de perdas de comprimento i, ou quando a mensagem recebida segue uma outra mensagem previamente recebida.

Definição 4. *A probabilidade de uma transição de estado retroativa, a partir de um*

estado i para o estado 0 *é definida como* $to0(i) = p_{i0}$*, a partir da* Definição 2. $p_{i0} = 1 - p_{i(i+1)}$*, para* $i < h$*, e* $p_{h0} = 1$*, para* $i = h$.

A Tabela 4.1, baseada em Sanneck [92], mostra como as probabilidades usadas pela cadeia de Markov da Definição 2 poderiam ser aproximadas, através do uso das probabilidades de rajadas de perdas, chamadas $p_{L,z}$, para cada rajada de perdas de comprimento z. Se esta informação não estiver disponível antecipadamente e p_L estiver disponível, pode-se usar a fórmula de perda média, na Tabela 4.1, para gerar alguma distribuição de probabilidade (por exemplo, uniforme) dos $p_{L,z}$ antes do uso do modelo.

Tabela 4.1: Cálculo das probabilidades da cadeia de Markov.

Modelo de Markov com $h+1$ *estados*	*a é o maior número de seqüência de heartbeat válido que foi recebido*	$a \to \infty$
Rajada de perdas $(0 < z \leq h)$	$p_{L,z} = \frac{o_z}{a}$ (o_z é o número de rajadas de perdas de comprimento z)	$Pr(X_{n+1} = z)$
Perda média	$p_L = \sum_{z=1}^{h} \frac{zo_z}{a} = \sum_{z=1}^{h} zp_{L,z}$	$E[X_{n+1}]$
Perda acumulada $(0 < z \leq h)$	$p_{L,cum}(z) = \sum_{n=z}^{h} \frac{o_n}{a} = \sum_{n=z}^{h} p_{L,n}$	$Pr(X_{n+1} \geq z)$ (probabilidade de estado)
Perda acumulada $(z = 0)$	$p_{L,cum}(0) = 1 - p_L$	$Pr(X_{n+1} = 0)$ (nenhuma perda)
Perda condicional $(0 < z \leq h)$	$p_{L,cond}(z-1, z) = \frac{p_{L,cum}(z)}{p_{L,cum}(z-1)}$	$Pr(X_{n+1} \geq z \mid X_n \geq z-1)$ (probabilidade de transição de estado $p_{(z-1)z}$)

A probabilidade de perda média (p_L) do modelo proposto possui o mesmo significado de Chen et al (veja a seção 3.1, pág. 566, de [30]), ou seja, é a probabilidade de uma mensagem ser perdida pelo canal de comunicação. Na seção 5.2, pág. 570, de [30], Chen et al fornecem um exemplo de como estimar p_L da seguinte forma: pode-se usar os números de seqüência das mensagens *heartbeat* para contar o número de *heartbeats* perdidos e então dividir este número pelo maior número de seqüência recebido até então.

No nosso modelo esse maior número de seqüência de *heartbeats* é o a na Tabela 4.1. Quando a tende ao infinito, p_L corresponde a $E[X_{n+1}]$. Usando a definição de esperança matemática, e sabendo que X_{n+1} toma os valores discretos de 0 a h (Definição 2), temos $E[X_{n+1}] = \sum_{z=0}^{h} zPr(X_{n+1} = z) = 0Pr(X_{n+1} = 0) + \sum_{z=1}^{h} zPr(X_{n+1} = z) = \sum_{z=1}^{h} zp_{L,z}$.

4.3 Modelo Alterado para a Análise de QoS de NFD-S

A Definição 5, a seguir, usa a cadeia de Markov da Definição 2 para a QoS de detectores de defeitos na presença de rajadas de perdas. A Definição 5 caracteriza a probabilidade de que uma *transição-S* ocorra no instante τ_i, e o que é necessário para que ela ocorra. k define quantas mensagens poderiam ser recebidas dentro de η, supondo um atraso de δ. Uma *transição-S* corresponde a uma recepção da mensagem m_{i-1} antes de τ_i seguida por nenhuma recepção de uma mensagem m_j, $j \geq i$, antes de τ_i. A recepção é determinada pela probabilidade q_0, e a posterior não recepção por $u(x)$, que é a probabilidade de nenhuma das k mensagens chegar antes de τ_i devido a perdas de mensagens ou a mensagens atrasadas.

Definição 5.

1. *Para qualquer $i \geq 1$, seja k o menor inteiro tal que, para todo $j \geq i + k$, m_j é enviada no ou após o instante τ_i.*

2. *Para qualquer $i \geq 2$, seja q_0 a probabilidade de q receber a mensagem m_{i-1} antes do instante τ_i. Neste caso, a cadeia de Markov está no estado 0.*

3. *Para qualquer $i \geq 1$, seja $u(x)$ a probabilidade de q suspeitar de p, através do não recebimento das mensagens m_{i+j}, para cada $0 \leq j \leq k-1$, no instante $\tau_i + x$, para todo $x \in [0, \eta)$. Esta definição supõe que a cadeia de Markov já está no estado 0 (definições 5.2 e 5.4). Por isso, a partir do estado 0, a cadeia de Markov realiza transições.*

4. *Para qualquer $i \geq 2$, seja p_s a probabilidade de uma transição-S ocorrer no instante τ_i. Isto caracteriza toda a cadeia de Markov.*

A Figura 4.1 ilustra a Definição 5. Nesta figura $k = 2$, $i = 2$, $j = i + k = 4$, $\delta = 2$ e $\eta = 1$. $m_{j=4}$ não aparece na figura mas seria enviada em $\sigma_{j=4}$, que corresponde ao instante τ_i. q_0 é a probabilidade da mensagem $m_{i-1=2-1=1}$ atrasar menos que $\delta + \eta = 3$ e chegar em q antes do instante $\tau_{i=2}$. $u(0)$ é a probabilidade que $m_{i=2}$ e $m_{i+1=3}$ cheguem antes de $\tau_{i=2}$. Ou seja, $u(0)$ é a probabilidade que $m_{i=2}$ se perca ou se atrase mais que $\delta = 2$, mais a probabilidade que $m_{i+1=3}$ se perca ou se atrase mais que $\delta - \eta = 1$.

q_0 define que a cadeia de Markov fica no estado 0 e o detector q fornece como saída T, e então haverá uma seqüência de possíveis transições pela cadeia de Markov, indicada por todas as $2^{k=2}$ possibilidades de perdas e atrasos de m_2 e m_3, levando o detector q a fornecer como saída S, o que caracteriza a *transição-S*.

A Proposição 6, a seguir, descreve matematicamente a Definição 5 de uma forma

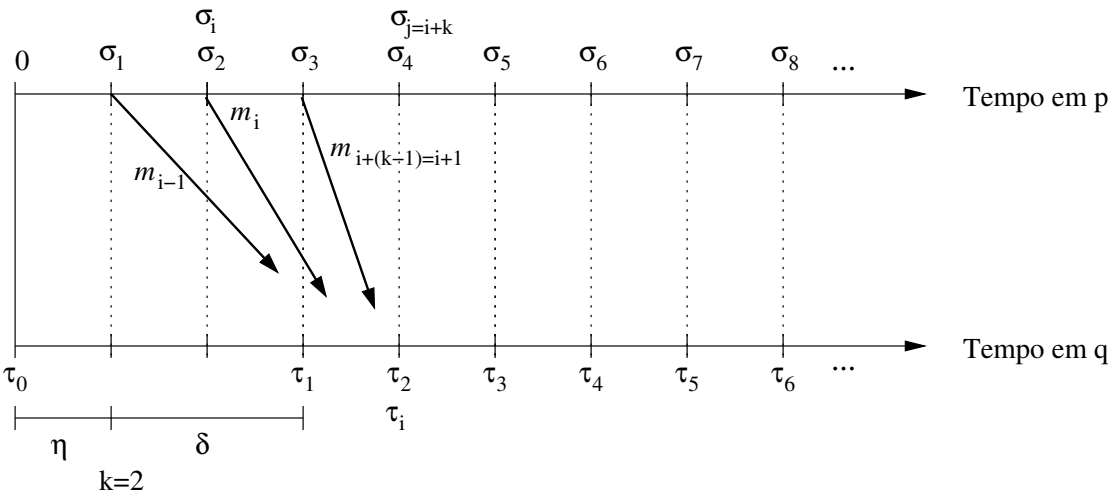


Figura 4.1: Ilustração para a Definição 5 quando $k = 2$.

independente de i.

Proposição 6.

1. $k = \lceil \delta/\eta \rceil$.
2. $q_0 = Pr(X_{n+1} = 0)Pr(D < \delta + \eta)$.
3. *Para todo* $x \in [0, \eta)$*, e* w *inicialmente igual a* k*,* $u(x) = u_w(x)$*.* $u_w(x)$ *é definido como segue:*

$$
\begin{aligned}
u_1(x) &= forw(0,1) + to0(0)Pr(D > \delta + x - (k-1)\eta), \text{ para } w = 1;\\
u_w(x) &= to0(0)Pr(D > \delta + x - (k-w)\eta)u_{w-1}(x)\\
&\quad + \sum_{a=1}^{w-2} forw(0,a)to0(a)Pr(D > \delta + x - (a+k-w)\eta)u_{w-(a+1)}(x)\\
&\quad + forw(0,w-1)to0(w-1)Pr(D > \delta + x - (k-1)\eta)\\
&\quad + forw(0,w), \text{ para } w > 1.
\end{aligned}
$$

4. $p_s = q_0 u(0)$.

Na Proposição 6.3: i) k mensagens poderiam ser recebidas dentro de $[\tau_{i-1}, \tau_i + x)$; ii) o número total de permutações de perdas (representadas por bits 1) e atrasos (representados por bits 0) é 2^k; e iii) o índice w indica quantas mensagens estão sendo consideradas, e a partir de qual entre $k - w$ a $k - 1$. Por exemplo, $w = k$ considera as mensagens 0 a $k - 1$, e $w = k - 1$ considera as mensagens 1 a $k - 1$. A seguir está uma descrição detalhada da Proposição 6.3.

$u_1(x) = forw(0,1) + to0(0)Pr(D > \delta + x - (k-1)\eta)$ representa a probabilidade da $(k-1)$-ésima mensagem ser perdida, ou ser atrasada mais do que $\delta + x - (k-1)\eta$ unidades de tempo, de forma que ela não seja recebida dentro de $[\tau_{i-1}, \tau_i + x)$.

$forw(0, w-1)to0(w-1)Pr(D > \delta + x - (k-1)\eta)$ em $u_w(x)$ considera a probabilidade de padrões de mensagens com sufixo $1^{w-1}0$. $forw(0, w-1)$ fornece a probabilidade da seqüência de $w-1$ perdas de mensagens $k-w$ a $k-2$. Logo, $to0(w-1)Pr(D > \delta + x - (k-1)\eta)$ fornece a probabilidade da $(k-1)$-ésima mensagem ser recebida com atraso maior que $\delta + x - (k-1)\eta$.

$forw(0, w)$ em $u_w(x)$ considera a probabilidade de padrões de mensagens com sufixo 1^w, ou seja, a probabilidade da seqüência de w perdas de mensagens $k-w$ a $k-1$.

$to0(0)Pr(D > \delta + x - (k-w)\eta)u_{w-1}(x)$ em $u_w(x)$ considera a probabilidade de padrões com prefixo 0. $to0(0)Pr(D > \delta + x - (k-w)\eta)$ significa a probabilidade do atraso da $(k-w)$-ésima mensagem ser maior que $\delta + x - (k-w)\eta$ unidades de tempo. $u_{w-1}(x)$ representa a recorrência que calcula as probabilidades das 2^{w-1} combinações das próximas $w-1$ mensagens. Esta recorrência termina quando $w = 2$, através da chamada a $u_1(x)$.

$\sum_{a=1}^{w-2} forw(0,a)to0(a)Pr(D > \delta + x - (a+k-w)\eta)u_{w-(a+1)}$ em $u_w(x)$ considera a probabilidade de padrões que possuem prefixo 1^a0, onde a, $1 \leq a \leq w-2$, é o número de perdas consecutivas de mensagens. A probabilidade resultante das perdas é descrita por $forw(0,a)$. $to0(a)Pr(D > \delta + x - (a+k-w)\eta)$ representa a probabilidade da $(a+k-w)$-ésima mensagem ser atrasada mais que $\delta + x - (a+k-w)\eta$. $u_{w-(a+1)}(x)$ representa a probabilidade de ocorrer as $2^{w-(a+1)}$ combinações das próximas $w-(a+1)$ mensagens. Esta recorrência termina quando $a = w-2$, através da chamada a $u_1(x)$.

Prova da Proposição 6.

1) A prova da Proposição 6.1 é a mesma da tese de Chen [29]: é imediata do fato que m_j é enviada no instante $\tau_i - \delta + (j-i)\eta$ para todo $i \geq 1$.

2) A prova da Proposição 6.2 segue diretamente do fato que q_0 é a probabilidade de m_{i-1} não ser perdida e ser recebida com um atraso menor que $\delta + \eta$ unidades de tempo, que leva ao estado 0 ($Pr(X_{n+1} = 0)$).

3) Uma prova por indução forte segue para verificar se a recorrência funciona. Na base da indução, quando $w = k = 1$, $u_1(x)$ representa a probabilidade da $(k-1)$-ésima mensagem ser perdida, ou não ser recebida dentro de $[\tau_{i-1}, \tau_i + x)$ porque seu atraso é maior que $\delta + x - (k-1)\eta$ unidades de tempo. Quando $w = k = 2$, $u_2(x)$ claramente usa o primeiro, terceiro, e quarto termos da definição de $u_w(x)$, e somente $u_1(x)$ no primeiro termo. Na hipótese de indução, nós consideramos que $u_w(x)$ funciona quando

$2 \leq w \leq k-1$. No passo de indução, nós vericamos se $u_w(x)$ funciona quando $2 \leq w \leq k$. Fica claro que quando $w = k$, o primeiro termo de $u_w(x)$ usa $u_{w-1}(x)$, que pela hipótese de indução é calculada por $u_{k-1}(x)$. Quando $w = k$, o segundo termo de $u_w(x)$ usa $u_{w-(a+1)}(x)$, onde $w-(a+1)$ varia de $k-2$ para 1. Por isso, pela hipótese de indução, $u_{w-(a+1)}(x)$ é corretamente calculado. As provas do terceiro e quarto termos seguem diretamente. Logo, $u_w(x)$ funciona quando $2 \leq w \leq k$.

4) A partir da Proposição 6.2, q fornece como saída T, e a partir da Proposição 6.3, q fornece como saída S, levando a uma *transição-S*. Logo, p_s realmente é a probabilidade que uma *transição-S* ocorra no instante τ_i. □

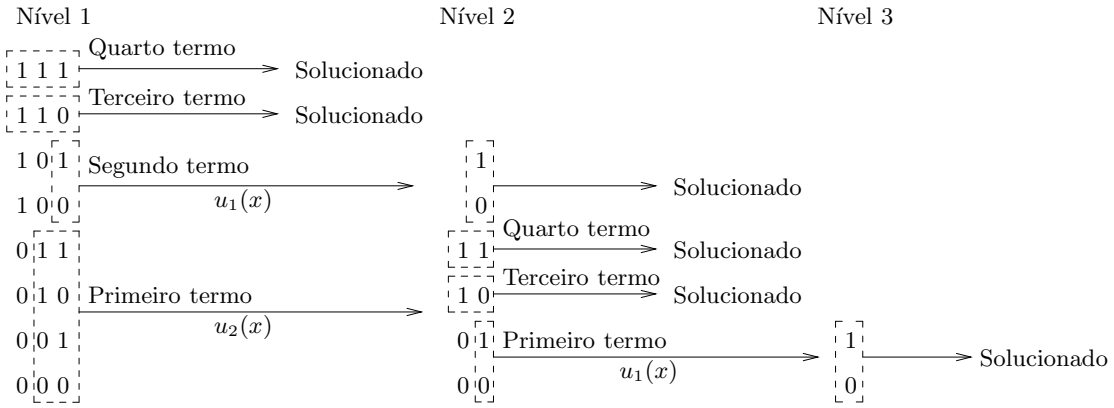


Figura 4.2: Um exemplo para a Proposição 6.3 quando $k = 3$.

A Figura 4.2 apresenta um exemplo que usa os termos da Proposição 6.3. $k = 3$, que leva a 3 níveis de recursão e 2^3 permutações, cada uma com comprimento k. Este exemplo mostra a recursão na construção dessas permutações (padrões de bits):

No nível 1: a) o primeiro termo trata dos padrões que iniciam com 0 (011, 010, 001, e 000), através da chamada a $u_2(x)$; b) o segundo termo lida com os padrões que iniciam com seqüência de 1's e são seguidos por uma seqüência de 0's (101 e 100), exceto aqueles padrões que possuem seqüências de 1's e somente o último bit é 0; os padrões 1 e 0 são tratados pela chamada a $u_1(x)$; c) o terceiro termo trata dos padrões em que todos os bits são 1, exceto pelo último bit que é 0 (110); e d) o quarto termo lida com padrões onde há somente 1's (111).

No nível 2: Os padrões (11, 10, 01, e 00) são tratados por $u_2(x)$, onde: a) o primeiro termo trata dos padrões 01 e 00, e chama $u_1(x)$; b) o terceiro termo gera o padrão 10; e c) o quarto termo lida com o padrão 11. Os padrões 1 e 0 são tratados por $u_1(x)$.

No nível 3: os padrões 1 e 0 são tratados por $u_1(x)$.

A partir da Figura 4.2 e a Proposição 6.3, vemos que a idéia é gerar todas as permutações de 0's e 1's, para um dado k, e então realizar transições na cadeia de Markov de acordo com perdas (1's), e recepções de mensagens (0's). A probabilidade de uma transição adiante é $forw$, e de uma transição retroativa é $to0$. Entretanto, para uma recepção de mensagem após $\tau_i + x$, para todo $x \in [0, \eta)$, a transição retroativa torna-se $to0Pr(D > y)$, onde $y \in [0, \delta)$. Cada permutação de comprimento k possui uma probabilidade de ocorrer, e $u_k(x)$ é a soma de todas as 2^k probabilidades.

A Proposição 7, a seguir, descreve os casos não degenerados onde q eventualmente recebe uma mensagem válida de p antes do instante τ_i ($q_0 > 0$), e eventualmente q suspeita de p ($u(0) > 0$). Isto elimina os casos quando q suspeita de p permanentemente com probabilidade 1 ($q_0 = 0$), e quando q confia em p permanentemente com probabilidade 1 ($u(0) = 0$). Esta Proposição é usada posteriormente pela Proposição 17.

Proposição 7. *Os casos não degenerados $q_0 > 0$ e $u(0) > 0$, no modelo proposto, ocorrem quando $Pr(D > \delta) > 0$, $Pr(D < \delta + \eta) > 0$, $0 < Pr(X_{n+1} = 0) < 1$, e $0 < Pr(X_{n+1} \geq 1) < 1$.*

Prova. Se $q_0 = 0$ ou $u(0) = 0$ então nenhuma *transição-S* ocorreria porque $p_s = q_0 u(0)$ seria zero. Então eliminamos estes casos degenerados e supomos que $q_0 > 0$ e $u(0) > 0$. A partir da Proposição 6, $q_0 = Pr(X_{n+1} = 0)Pr(D < \delta + \eta)$. Visto que supomos que $Pr(D < \delta + \eta) > 0$ e $Pr(X_{n+1} = 0) > 0$, e $Pr(X_0 = 0) = Pr(X_{n+1} = 0) > 0$ então $q_0 > 0$. A partir da definição de $u(x)$ na Proposição 6.4, $u(0) = u_k(0)$. Neste caso nós também consideramos as mensagens 0 a $k - 1$. $u_w(0)$, com w inicialmente igual a k, é então:

$$
\begin{aligned}
u_1(0) &= forw(0,1) + to0(0)Pr(D > \delta - (k-1)\eta), \text{ para } w = 1; \\
u_w(0) &= to0(0)Pr(D > \delta - (k-w)\eta)u_{w-1}(0) \\
&\quad + \sum_{a=1}^{w-2} forw(0,a)to0(a)Pr(D > \delta - (a+k-w)\eta)u_{w-(a+1)}(0) \\
&\quad + forw(0, w-1)to0(w-1)Pr(D > \delta - (k-1)\eta) \\
&\quad + forw(0,w), \text{ para } w > 1.
\end{aligned}
$$

Quando $k = 1$, o cálculo de $forw(0,1)$ na Tabela 4.1 usando $0 < Pr(X_{n+1} = 0) < 1$ e $0 < Pr(X_{n+1} \geq 1) < 1$, leva a $forw(0,1)$ ser maior que zero. Então, $u(0) = u_1(0) > 0$. Quando $k > 1$, é suficiente provar que um dos termos de $u_w(0)$ é maior que 0. Então, usando a Tabela 4.1 e somente o primeiro termo de $u_w(0)$, quando nós geramos o padrão 0^k, supondo que $Pr(D > \delta) > 0$, $to0(0)Pr(D > \delta - (k-w)\eta)u_{w-1}(0) > 0$. Então,

$u(0) > 0$. □

A seguir, a Definição 8 caracteriza a probabilidade para q suspeitar de p, e a Proposição 9 descreve matematicamente esta probabilidade. A principal diferença entre a Definição 5.3 e a Definição 8 é que na primeira a cadeia de Markov inicia no estado 0 devido às Definições 5.2 e 5.4, e na Definição 8 a cadeia de Markov pode estar em qualquer estado $s \in S$ (veja a Definição 2).

Definição 8. *Para qualquer $i \geq 1$, seja $v(x)$ a probabilidade que q suspeita de p no instante $\tau_i + x$, para cada $x \in [0, \eta)$. Esta suspeita ocorre quando nenhuma das mensagens m_{i+j} é recebida até o instante $\tau_i + x$, para cada $0 \leq j \leq k-1$. $v(x)$ supõe que a cadeia de Markov pode iniciar em qualquer estado $s \in S$.*

A Proposição 9 é similar à Proposição 6 exceto que a cadeia de Markov pode iniciar em qualquer estado $s \in S$.

Proposição 9. *$v(x) = Pr(X_{n+1} = 0)v_{0,k}(x) + \sum_{s=1}^{h} Pr(X_{n+1} \geq s)v_{s,k}(x)$. $u_{w-1}(x)$ e $u_{w-(a+1)}(x)$ usam a definição de $u(x)$ na Proposição 6 e a Definição 8. $v_{s,w}(x)$, com w inicialmente igual a k, é definido como segue:*

$$
\begin{aligned}
v_{s,1}(x) &= forw(s, s+1) + to0(s)Pr(D > \delta + x - (k-1)\eta), \text{ para } w = 1; \\
v_{s,w}(x) &= to0(s)Pr(D > \delta + x - (k-w)\eta)u_{w-1}(x) \\
&\quad + \sum_{a=1}^{w-2} forw(s, s+a)to0(s+a)Pr(D > \delta + x - (a+k-w)\eta)u_{w-(a+1)}(x) \\
&\quad + forw(s, s+w-1)to0(s+w-1)Pr(D > \delta + x - (k-1)\eta) \\
&\quad + forw(s, s+w), \text{ para } w > 1.
\end{aligned}
$$

Prova. A prova de $v(x)$ é imediata a partir do fato que a partir de qualquer estado da cadeia de Markov é possível perder uma mensagem (transição adiante) ou receber uma mensagem (transição ao estado 0). Adicionalmente, a definição de $u(x)$ na Proposição 6 pode ser usada porque a cadeia de Markov está sempre no estado 0 quando $u_{w-1}(x)$ e $u_{w-(a+1)}(x)$ são chamados na definição de $v_{s,w}(x)$. □

A Proposição 10 é usada posteriormente na prova da Proposição 17. A intuição aqui é que com mais tempo para esperar ($x > 0$), a probabilidade de termos uma suspeita ($u(x)$) é menor.

Proposição 10. *Para todo $x \in [0, \eta)$, $u(0) \geq u(x)$.*

Prova. A partir da definição de $u(x)$ na Proposição 6.4, $u(0) = u_k(0)$. Neste caso nós também consideramos as mensagens 0 a $k-1$. $u_w(0)$, com w inicialmente igual a k,

é então:

$$
\begin{aligned}
u_1(0) &= forw(0,1) + to0(0)Pr(D > \delta - (k-1)\eta), \text{ para } w = 1;\\
u_w(0) &= to0(0)Pr(D > \delta - (k-w)\eta)u_{w-1}(0)\\
&\quad + \sum_{a=1}^{w-2} forw(0,a)to0(a)Pr(D > \delta - (a+k-w)\eta)u_{w-(a+1)}(0)\\
&\quad + forw(0,w-1)to0(w-1)Pr(D > \delta - (k-1)\eta)\\
&\quad + forw(0,w), \text{ para } w > 1.
\end{aligned}
$$

Para provar que para todo $x \in [0,\eta)$, $u(0) \geq u(x)$, é suficiente perceber que $u(x)$ é influenciado por $Pr(D > \delta + x - y\eta)$, onde y pode ser $k-w$, $a+k-w$, ou $k-1$. Visto que $0 \leq y \leq \delta$, $\delta + x - y\eta \geq 0$ para todo $x \in [0,\eta)$. Quanto maior é o valor de $\delta + x - y\eta$, menor ou igual é o valor de $Pr(D > \delta + x - y\eta)$. Mais ainda, $Pr(D > \delta + x - y\eta)$ é sempre usado em multiplicações seguidas por somas de números positivos. Por isso, se $x_1 = 0$ e $x_2 \in [0,\eta)$, $Pr(D > \delta + x_1 - y\eta) \geq Pr(D > \delta + x_2 - y\eta)$, que implica em $u(x_1) \geq u(x_2)$, levando a $u(0) \geq u(x)$. □

O Lema 11, a seguir, é usado posteriormente pelo Teorema 16.3. Sua prova é a mesma daquela do Lema 15 de Chen et al [30], exceto pelo uso de $v(x)$ de nossa Proposição 9 no lugar de $u(x)$. Este Lema fornece a probabilidade de consulta com precisão, que é a probabilidade de q estar confiando que p esteja ativo em um instante de tempo aleatório.

Lema 11. $P_A = 1 - \frac{1}{\eta}\int_0^\eta v(x)dx$.

O Lema 12 a seguir, que é o Lema 17 de Chen et al [30], ainda é válido em nosso modelo. Este Lema, que é usado posteriormente pelo Teorema 14, apresenta um processo estocástico definido pelas variáveis aleatórias $T_{MR,n}$ e $T_{M,n}$. Estas variáveis representam, respectivamente, o período entre duas *transições-S* consecutivas e o período entre uma *transição-S* até a próxima *transição-T*. Desta forma $T_{M,n} \leq T_{MR,n}$ para todo $n \geq 1$, e nós obtemos que para todo $n \geq 2$, $T_{MR} = T_{MR,n}$, $T_M = T_{M,n}$, e $T_G = T_{MR} - T_M$, que é a relação encontrada no Teorema 1.1 da seção 3.1.5.

Lema 12. $\{(T_{MR,n}, T_{M,n}), n = 1, 2, \ldots\}$ *é um processo regenerativo com recompensa atrasada*[42].

O Lema 13 a seguir, que é o Lema 4 de Chen et al [30], ainda é válido em nosso modelo porque o nosso modelo usa o mesmo algoritmo NFD-S (veja a Figura 3.1). Este Lema possibilita o uso do Teorema 1 (veja a seção 3.1.5) na análise de NFD-S.

[42]Do inglês: *delayed renewal reward process.*

Lema 13. *NFD-S é um detector de defeitos ergódico.*

O Lema 14 a seguir, que corresponde ao Lema 16 de Chen et al [30], ainda é válido em nosso modelo, através do uso da definição de p_s em nossa Proposição 6.

Lema 14. $E(T_{MR}) = \eta/p_s$.

A prova de Chen et al [30] para o seu Lema 18 é ainda válida para o Lema 15 a seguir, porque o mesmo algoritmo NFD-S é usado.

Lema 15. $T_D \leq \delta + \eta$ *e este limite é estrito.*

O Teorema a seguir sumariza a nossa análise de QoS de NFD-S, sob rajadas de perdas, a partir das Definições, Lemas, e Proposições que seguem a nossa Definição 2 da seção 4.2.

Teorema 16. *Considere um sistema com relógios sincronizados, onde a probabilidade de perda de mensagens p_L, a distribuição de atrasos de mensagens $Pr(D \leq x)$, e a distribuição de probabilidade dos comprimentos de rajadas de perdas são conhecidas. O detector de defeitos NFD-S com parâmetros η e δ possui as seguintes propriedades:*

1. *O tempo de detecção é limitado como segue e o limite é estrito:* $T_D \leq \delta + \eta$. (4.1)
2. *O tempo de recorrência de erros médio é:* $E(T_{MR}) = \frac{\eta}{p_s}$. (4.2)
3. *A duração de erros média é:* $E(T_M) = \frac{\int_0^\eta v(x)d(x)}{p_s}$. (4.3)

Prova. As partes 1 e 2 do Teorema seguem diretamente dos Lemas 15 e 14. A parte 3 é derivada da relação entre $E(T_M)$, P_A, e $E(T_{MR})$, a partir da parte 2 do Teorema 1 e os resultados sobre P_A e $E(T_{MR})$ a partir dos Lemas 11 e 14. □

4.4 Modelo do Configurador para NFD-S

De forma similar a Chen et al, nosso objetivo é encontrar um configurador, que toma como entrada o comportamento probabilístico de *heartbeats* e os requisitos de QoS (T_D^U, T_{MR}^L, T_M^U), e fornece como saída η e δ. T_D^U é um limite superior para o tempo de detecção, T_{MR}^L é um limite inferior sobre o tempo de recorrência de erros médio, e T_M^U é um limite superior sobre a duração de erros média. Em outras palavras, os requisitos de QoS são:

$$T_D \leq T_D^U, E(T_{MR}) \geq T_{MR}^L, E(T_M) \leq T_M^U. \quad (4.4)$$

A partir do Teorema 16, o objetivo pode ser reescrito como um problema de programação matemática:

maximizar η:

restrito a $\delta + \eta \leq T_D^U$ (4.5)

$$\frac{\eta}{p_s} \geq T_{MR}^L \tag{4.6}$$
$$\frac{\int_0^\eta v(x)dx}{p_s} \leq T_M^U \tag{4.7}$$

onde o valor de $v(x)$ é dado pela Proposição 9, e o valor de p_s é dado pela Proposição 6. Similar a Chen et al [30], a asserção (4.7) pode ser substituído por uma restrição mais simples e mais forte como segue.

Proposição 17. *Nos casos não degenerados da Proposição 7,* $E(T_M) \leq \frac{v(0)\eta}{q_0 u(0)}$.

Prova. Esta prova é a mesma do Lema 16 de Chen et al [30], exceto pelo uso de $u(0) > 0$ da Proposição 7. Para todo $i \geq 2$, Chen et al supõem que A_i é o evento em que uma *transição-S* ocorre no instante τ_i. Através do uso de $u(0) > 0$ é possível afirmar que $Pr(A_i) = p_s = q_0 u(0) > 0$ nos casos não degenerados. Pela Proposição 10, $u(0) \geq u(x)$, para todo $x \in [0, \eta)$, que também é válido para $v(0) \geq v(x)$. Desta forma, a partir da igualdade (4.3) e da Proposição 7: $E(T_M) = \frac{\int_0^\eta v(x)dx}{p_s} \leq \frac{\int_0^\eta v(0)dx}{q_0 u(0)} = \frac{v(0)\eta}{q_0 u(0)}$. □

A partir da asserção (4.5) e das Proposições 6 e 17, obtemos a Proposição 18 a seguir, que é usada posteriormente pelo configurador proposto NSM-NFD-S.

Proposição 18. *Seja* $k' = \lceil T_D^U/\eta \rceil - 1$. *Da mesma forma que Chen et al [30],* $v'(0)$ *e* $u'(0)$ *consideram a seguir, somente as mensagens* 0 *a* $k-1$. $v'(0) = Pr(X_{n+1} = 0)v_{0,k'}(0) + \sum_{s=1}^{h} Pr(X_{n+1} = s)v'_{s,k'}(0)$. $v'_{s,w}(0)$, *que é baseado na Proposição 9, é definido como segue:*

$$\begin{aligned}
v'_{s,1}(0) &= forw(s, s+1) + to0(s)Pr(D > T_D^U - k'\eta), \text{ para } w = 1;\\
v'_{s,w}(0) &= to0(s)Pr(D > T_D^U - (k'-w+1)\eta)u'_{w-1}(0)\\
&\quad + \sum_{a=1}^{w-2} forw(s, s+a)to0(s+a)Pr(D > T_D^U - (a+k'-w+1)\eta)u'_{w-(a+1)}(0)\\
&\quad + forw(s, s+w-1)to0(s+w-1)Pr(D > T_D^U - k'\eta)\\
&\quad + forw(s, s+w), \text{ para } w > 1.
\end{aligned}$$

Os termos $u'_{w-1}(0)$ *e* $u'_{w-(a+1)}(0)$ *de* $v'_{s,w}(0)$ *usam a definição de* $u'(0)$ *a seguir, que é*

baseada na Proposição 6:

$$
\begin{aligned}
u'_1(0) &= forw(0,1) + to0(0)Pr(D > T_D^U - k'\eta), \text{ para } w = 1;\\
u'_w(0) &= to(0)Pr(D > T_D^U - (k' - w + 1)\eta)u'_{w-1}(0)\\
&\quad + \sum_{a=1}^{w-2} forw(0,a)to0(a)Pr(D > T_D^U - (a + k' - w + 1)\eta)u'_{w-(a+1)}(0)\\
&\quad + forw(0,w-1)to0(w-1)Pr(D > T_D^U - k'\eta)\\
&\quad + forw(0,w), \text{ para } w > 1.
\end{aligned}
$$

Prova. Pela parte 1 do Teorema 16, usamos $T_D \leq T_D^U = \eta + \delta$ para provar $u'(0) = u(0)$ a seguir:

Quando $w = k = 1$:

$$
\begin{aligned}
u_1(0) &= forw(0,1) + to0(0)Pr(D > \delta - (k-1)\eta), \text{ para } k = \lceil \delta/\eta \rceil;\\
&= forw(0,1) + to0(0)Pr(D > \eta + \delta - k'\eta), \text{ para } k = \lceil \delta/\eta \rceil = \lceil 1 + \delta/\eta \rceil - 1 = k';\\
&= forw(0,1) + to0(0)Pr(D > \eta + \delta - k'\eta), \text{ para } k' = \lceil (\eta + \delta)/\eta \rceil - 1;\\
&= forw(0,1) + to0(0)Pr(D > T_D^U - k'\eta), \text{ para } k' = \lceil T_D^U/\eta \rceil - 1;\\
&= u'_1(0).
\end{aligned}
$$

Quando $w = k > 1$:

$$
\begin{aligned}
u_w(0) &= to0(0)Pr(D > \delta - (k-w)\eta)u_{w-1}(0)\\
&\quad + \sum_{a=1}^{w-2} forw(0,a)to0(a)Pr(D > \delta - (a + k - w)\eta)u_{w-(a+1)}(0)\\
&\quad + forw(0,w-1)to0(w-1)Pr(D > \delta - (k-1)\eta)\\
&\quad + forw(0,w), \text{ for } k = \lceil \delta/\eta \rceil;
\end{aligned}
$$

$$
\begin{aligned}
&= to0(0)Pr(D > \eta + \delta - (k' - w + 1)\eta)u_{w-1}(0) \\
&+ \sum_{a=1}^{w-2} forw(0,a)to0(a)Pr(D > \eta + \delta - (a + k' - w + 1)\eta)u_{w-(a+1)}(0) \\
&+ forw(0, w-1)to0(w-1)Pr(D > \eta + \delta - k'\eta) \\
&+ forw(0, w), \text{ for } k = \lceil \delta/\eta \rceil = \lceil (\eta + \delta)/\eta \rceil - 1 = k'; \\
&= to0(0)Pr(D > T_D^U - (k' - w + 1)\eta)u_{w-1}(0) \\
&+ \sum_{a=1}^{w-2} forw(0,a)to0(a)Pr(D > T_D^U - (a + k' - w + 1)\eta)u_{w-(a+1)}(0) \\
&+ forw(0, w-1)to0(w-1)Pr(D > T_D^U - k'\eta) \\
&+ forw(0, w), \text{ for } k' = \lceil T_D^U/\eta \rceil - 1; \\
&= u'_w(0) \\
&= to0(0)Pr(D > T_D^U - (k' - w + 1)\eta)u'_{w-1}(0) \\
&+ \sum_{a=1}^{w-2} forw(0,a)to0(a)Pr(D > T_D^U - (a + k' - w + 1)\eta)u'_{w-(a+1)}(0) \\
&+ forw(0, w-1)to0(w-1)Pr(D > T_D^U - k'\eta) \\
&+ forw(0, w), \text{ for } k' = \lceil T_D^U/\eta \rceil - 1;
\end{aligned}
$$

Logo, $u'(0) = u(0)$. A prova de $v'(0)$ segue diretamente da prova de $v(0)$ da Proposição 9 e da prova acima sobre $u'(0) = u(0)$. A única mudança é o uso do estado s nas probabilidades $to0$ e $forw$. □

Configurador NSM-NFD-S. A partir das asserções (4.4), (4.5), (4.6), (4.7) e das Proposições 17 e 18, obtemos o seguinte configurador, chamado de configurador NSM-NFD-S, para encontrar η e δ:
Passo 1: Computar $q'_0 = Pr(X_{n+1} = 0)Pr(D < T_D^U)$ e fazer $g(\eta) = v'(0)\eta/q'_0u'(0)$, onde $v'(0) = v'_{k'}(0)$. Se $q'_0u'(0) = 0$, então fornecer como saída *"QoS não pode ser alcançada"* e parar. Caso contrário, encontrar o maior $\eta_{max} \leq T_D^U$ tal que $g(\eta_{max}) \leq T_M^U$.
Passo 2: Seja $f(\eta) = \eta/q'_0u'(0)$, encontrar o maior $\eta \leq \eta_{max}$ tal que $f(\eta) \geq T_{MR}^L$.
Passo 3: Atribuir $\delta = T_D^U - \eta$ e fornecer como saída η e δ.

A Figura 4.3 apresenta um esquema da configuração de NFD-S. As diferenças com a Figura 3.2 são que o estimador de comportamento probabilístico de *heartbeats* também fornece estimativas para $p_{L,z}$'s além de p_L, $E(D)$ e $V(D)$ e o configurador usado é o NSM-NFD-S, que é diferente de Chen-NFD-S definido no capítulo 3.

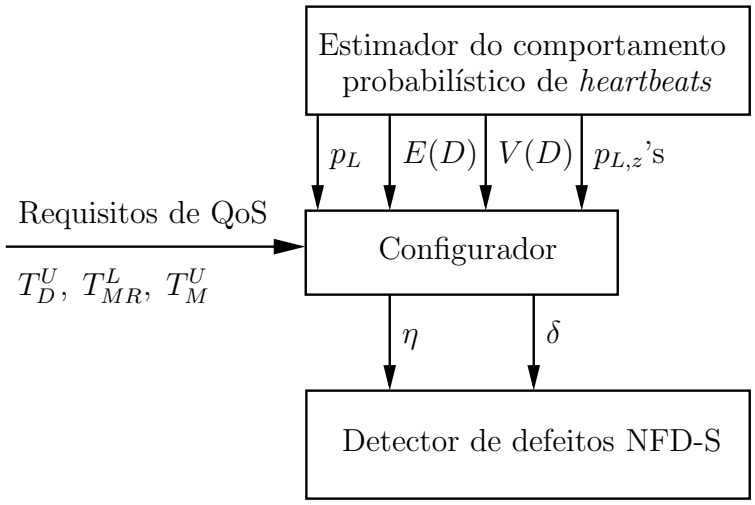


Figura 4.3: Configuração de NFD-S por NSM-NFD-S.

Teorema 19. *Considere um sistema em que os relógios estão sincronizados e a probabilidade de perda de mensagens p_L, a distribuição de atrasos de mensagens $Pr(D \leq x)$, e a distribuição de probabilidade de comprimentos de rajadas de perdas são conhecidas. Suponha que seja dado um conjunto de requisitos de QoS como em (4.4). O configurador NSM-NFD-S possui dois resultados possíveis: 1) Ele fornece como saída η e δ. Neste caso, com parâmetros η e δ, o detector de defeitos NFD-S satisfaz os requisitos de QoS solicitados. 2) Ele fornece como saída "QoS não pode ser alcançada".*

Prova. Nós provamos o Teorema, baseado em Chen et al [30], nas duas partes seguintes:

1. Suponha que o configurador NSM-NFD-S forneça como saída *"QoS não pode ser alcançada"*. Então, o configurador pára no *Passo 1* e, desta forma, $q_0'u'(0) = 0$. $q_0' = 0$ implica que $Pr(D < T_D^U) = 0$ ou $Pr(X_{n+1} = 0) = 1 - p_L = 0$. Nestas condições, para satisfazer $T_D \leq T_D^U$, q tem que suspeitar de p no instante $t > T_D^U$. Logo, para qualquer detector de defeitos, nós temos que $E(T_M) = \infty$ e desta forma, ele falha para satisfazer $E(T_M) \leq T_M^U$. $u'(0) = 0$ implica que não há suspeita alguma. Logo, $E(T_M) = g(\eta) = \infty$, que leva o detector de defeitos a falhar para satisfazer $E(T_M) \leq T_M^U$. Por isso, o detector de defeitos não pode satisfazer a QoS solicitada neste caso.

2. Suponha que o configurador NSM-NFD-S forneça como saída os parâmetros η e δ. Então, pelo *Passo 3*, temos que $T_D^U = \eta + \delta$. Pela parte 1 do Teorema 16, $T_D \leq T_D^U$ é satisfeito. Pelo *Passo 1* e a Proposição 6, $q_0' = Pr(X_{n+1} = 0)Pr(D < \eta + \delta) = q_0$, e pelo *Passo 1* e a Proposição 18, $u'(0) = u(0)$. Perceba que temos $q_0'u'(0) > 0$ visto que,

caso contrário, $g(\eta) = \infty$, e o configurador NSM-NFD-S forneceria *"QoS não pode ser alcançada"* em vez de η e δ. Pela Proposição 17 e *Passo 1*, $E(T_M) \leq g(\eta) = \frac{v'(0)\eta}{q'_0 u'(0)} = \frac{v(0)\eta}{q_0 u(0)} \leq T_M^U$. Logo, $E(T_M) \leq T_M^U$ é satisfeita. Desta forma, $f(\eta) = \eta/q'_0 u'(0) = \eta/q_0 u(0) = \eta/p_s = E(T_{MR})$ pela restrição (4.2). Pelo *Passo 2*, $f(\eta) \geq T_{MR}^L$ é satisfeita. □

4.5 Otimização do Configurador para NFD-S

$u'(0)$ na Proposição 18 leva tempo exponencial para ser executada, mesmo se $forw$, $to0$, e $Pr(D > y)$ forem previamente calculadas. Isto ocorre devido às chamadas recursivas a $u'_{w-1}(0)$ e $u'_{w-(a+1)}(0)$. Entretanto, a partir da análise da recorrência, é possível desenvolver uma solução iterativa (veja o exemplo na Figura 4.2 da seção 4.3).

Para todo $x \in [0, \eta]$, e seja a_u um vetor que armazena os valores de $u'_w(x)$ calculados para todo $w \in [1, k']$. Os termos $prefix_0$, $prefix_1^a0$, $suffix_1^{w-1}0$, e $suffix_1^w$ correspondem, respectivamente ao, primeiro, segundo, terceiro e quarto termos da definição de $u'(0)$ na Proposição 18. Mais detalhes podem ser obtidos a partir da explicação da Proposição 6 na seção 4.3. $u'(0) = calc_u'(x, k')$, que é definida como segue:

procedimento $calc_u'(x, k')$

1 $a_u[1] \leftarrow forw(0,1) + to0(0)Pr(D > T_D^U - k'\eta)$;
2 para $w \leftarrow 2$ até k'
3 $\quad prefix_0 \leftarrow to(0)Pr(D > T_D^U - (k' - w + 1)\eta)a_u[w-1]$;
4 $\quad prefix_1^a0 \leftarrow 0$;
5 $\quad$ para $a \leftarrow 1$ até $w - 2$
6 $\quad\quad prefix_1^a0 \leftarrow prefix_1^a0$
7 $\quad\quad\quad + forw(0,a)to0(a)Pr(D > T_D^U - (a + k' - w + 1)\eta)a_u[w - (a+1)]$;
8 $\quad suffix_1^{w-1}0 \leftarrow forw(0, w-1)to0(w-1)Pr(D > T_D^U - k'\eta)$;
9 $\quad suffix_1^w \leftarrow forw(0, w)$;
10 $\quad a_u[w] \leftarrow prefix_0 + prefix_1^a0 + suffix_1^{w-1}0 + suffix_1^w$;

Supor que $forw$, $to0$, e $Pr(D > y)$ são previamente calculados antes da chamada a $calc_u'(x, k')$, e os respectivos resultados colocados nos vetores adicionais, que são usados na implementação de $calc_u'(x, k')$. Desta forma, a partir dos dois laços dentro do procedimento, fica claro que este procedimento gasta tempo quadrático (polinomial) para ser executado.

4.6 Distribuição de Atrasos Desconhecida e NFD-S

Para computar η e δ quando $Pr(D \leq x)$ é desconhecida, podemos usar, de forma similar a Chen et al, a seguinte *Desigualdade Unilateral*: Para qualquer variável aleatória D com um valor esperado finito e uma variância finita, $Pr(D > t) \leq \frac{V(D)}{V(D)+(t-E(D))^2}$, para todo $t > E(D)$.

A partir da aplicação dessa *Desigualdade Unilateral* sobre as Proposições 6, 9 e 18, obtemos a Proposição 20 e a Heurística 21, a seguir.

Proposição 20. *Seja* $k_0 = \lceil(\delta - E(D))/\eta\rceil$. *A seguir,* $u^{\geq}(0)$ *e* $v^{\geq}(0)$ *consideram as mensagens* 0 *to* $k_0 - 1$, *e* w *é inicialmente* k_0. $v^{\geq}(0) = Pr(X_{n+1} = 0)v^{\geq}_{0,k_0}(0) + \sum_{s=1}^{h} Pr(X_{n+1} \geq s)v^{\geq}_{s,k_0}(0)$. $v^{\geq}_{s,w}(0)$, *que é baseada na Proposição 9, é definida como segue:*

$$
\begin{aligned}
v^{\geq}_{s,1}(0) &= forw(s, s+1) + to0(s)(V(D)/(V(D) + (\delta - E(D) - (k_0 - 1)\eta)^2)), \text{ para } w = 1;\\
v^{\geq}_{s,w}(0) &= to0(s)(V(D)/(V(D) + (\delta - E(D) - (k_0 - w)\eta)^2))u^{\geq}_{w-1}(0)\\
&\quad + \sum_{a=1}^{w-2} forw(s, s+a)to0(s+a)(V(D)/(V(D) + (\delta - E(D) - (a + k_0 - w)\eta)^2))u^{\geq}_{w-(a+1)}(0)\\
&\quad + forw(s, s+w-1)to0(s+w-1)(V(D)/(V(D) + (\delta - E(D) - (k_0 - 1)\eta)^2))\\
&\quad + forw(s, s+w), \text{ para } w > 1.
\end{aligned}
$$

Os termos $v^{\geq}_{w-1}(0)$ *e* $v^{\geq}_{w-(a+1)}(0)$ *de* $v^{\geq}_{s,w}(0)$ *usam a seguinte definição de* $u^{\geq}(0)$, *que é baseada na Proposição 6:*

$$
\begin{aligned}
u^{\geq}_{1}(0) &= forw(0, 1) + to0(0)(V(D)/(V(D) + (\delta - E(D) - (k_0 - 1)\eta)^2)), \text{ para } w = 1;\\
u^{\geq}_{w}(0) &= to(0)(V(D)/(V(D) + (\delta - E(D) - (k_0 - w)\eta)^2))u^{\geq}_{w-1}(0)\\
&\quad + \sum_{a=1}^{w-2} forw(0, a)to0(a)(V(D)/(V(D) + (\delta - E(D) - (a + k_0 - w)\eta)^2))u^{\geq}_{w-(a+1)}(0)\\
&\quad + forw(0, w-1)to0(w-1)(V(D)/(V(D) + (\delta - E(D) - (k_0 - 1)\eta)^2))\\
&\quad + forw(0, w), \text{ para } w > 1.
\end{aligned}
$$

Prova. Para $w = 1$:

$u_1(0) = forw(0,1) + to0(0)Pr(D > \delta - (k-1)\eta)$(Pela Desigualdade Unilateral temos:)

$\leq forw(0,1) + to0(0)(V(D)/(V(D) + (\delta - (k-1)\eta - E(D))^2)$, para $k = \lceil \delta/\eta \rceil$.

Pela definição da desigualdade unilateral temos que garantir que:

$$\delta - (k-1)\eta > E(D) \Leftrightarrow k < \frac{\delta - E(D)}{\eta} + 1 \Rightarrow k_0 = \left\lceil \frac{\delta - E(D)}{\eta} + 1 \right\rceil - 1$$

$$\Leftrightarrow k_0 = \left\lceil \frac{\delta - E(D)}{\eta} \right\rceil.$$

Portanto, usando k_0 temos:

$forw(0,1) + to0(0)(V(D)/(V(D) + (\delta - (k_0 - 1)\eta - E(D))^2) = u_1^{\geq}(0)$,

para $k_0 = \lceil (\delta - E(D))/\eta \rceil$.

Para $w > 1$:

$$\begin{aligned}
u_w(0) &= to0(0)Pr(D > \delta - (k-w)\eta)u_{w-1}(0) \\
&+ \sum_{a=1}^{w-2} forw(0,a)to0(a)Pr(D > \delta - (a+k-w)\eta)u_{w-(a+1)}(0) \\
&+ forw(0,w-1)to0(w-1)Pr(D > \delta - (k-1)\eta) \\
&+ forw(0,w)\text{(Pela Desigualdade Unilateral temos:)} \\
&\leq to0(0)(V(D)/(V(D) + (\delta - (k-w)\eta - E(D))^2))u_{w-1}(0) \\
&+ \sum_{a=1}^{w-2} forw(0,a)to0(a)(V(D)/(V(D) + (\delta - (a+k-w)\eta - E(D))^2))u_{w-(a+1)}(0) \\
&+ forw(0,w-1)to0(w-1)(V(D)/(V(D) + (\delta - (k-1)\eta - E(D))^2) \\
&+ forw(0,w), \text{ para } k = \lceil \delta/\eta \rceil.
\end{aligned}$$

Pela definição da desigualdade unilateral temos que garantir que:

$\delta - (k-1)\eta > E(D), \quad \delta - (k-w)\eta > E(D), \quad \delta - (a+k-w)\eta > E(D)$,

ou seja, basta garantir que $\delta - (k-1)\eta > E(D)$, que leva a

$k_0 = \left\lceil \frac{\delta - E(D)}{\eta} \right\rceil$ como demonstrado anteriormente.

Portanto, usando k_0 temos:

$$
\begin{aligned}
& to0(0)(V(D)/(V(D)+(\delta-(k_0-w)\eta-E(D))^2))u_{w-1}(0) \\
& + \sum_{a=1}^{w-2} forw(0,a)to0(a)(V(D)/(V(D)+(\delta-(a+k_0-w)\eta-E(D))^2))u_{w-(a+1)}(0) \\
& + forw(0,w-1)to0(w-1)(V(D)/(V(D)+(\delta-(k_0-1)\eta-E(D))^2) \\
& + forw(0,w), \text{ para } k_0=\lceil(\delta-E(D))/\eta\rceil; \\
& \leq to0(0)(V(D)/(V(D)+(\delta-(k_0-w)\eta-E(D))^2))u^{\geq}_{w-1}(0) \\
& + \sum_{a=1}^{w-2} forw(0,a)to0(a)(V(D)/(V(D)+(\delta-(a+k_0-w)\eta-E(D))^2))u^{\geq}_{w-(a+1)}(0) \\
& + forw(0,w-1)to0(w-1)(V(D)/(V(D)+(\delta-(k_0-1)\eta-E(D))^2) \\
& + forw(0,w), \text{ para } k_0=\lceil(\delta-E(D))/\eta\rceil; \\
& = u^{\geq}_{w}(0).
\end{aligned}
$$

Por analogia, a prova de $v^{\geq}(0)$ segue diretamente da prova de $u^{\geq}(0)$. □

A partir das proposição 20 obtemos a Heurística 21 a seguir.

Heurística 21. *Considere um sistema com relógios sincronizados e suponha que* $\delta > E(D)$. *Para o algoritmo NFD-S, temos* $E(T_{MR}) \geq \eta/\beta$ *e* $E(T_M) \approx v^{\geq}(0)\eta/\gamma$, *onde:*

$$
\begin{aligned}
\beta &= Pr(X_{n+1}=0)u^{\geq}(0), \\
\gamma &= Pr(X_{n+1}=0)\left(\frac{(\delta+\eta-E(D))^2}{(V(D)+(\delta+\eta-E(D))^2)}\right)u^{\geq}(0), \\
k_0 &= \lceil(\delta-E(D))/\eta\rceil.
\end{aligned}
$$

Comentário. A definição de k_0 vem da Proposição 20. Vamos comentar primeiro

sobre $E(T_{MR}) \geq \eta/\beta$:

$$\begin{aligned}
E(T_{MR}) &= \frac{\eta}{p_s} \\
&= \frac{\eta}{Pr(X_{n+1}=0)Pr(D<\delta+\eta)u(0)} \text{(Pelo Teorema 16.2 e a Proposição 6)} \\
&\geq \frac{\eta}{Pr(X_{n+1}=0)u(0)} \\
&\geq \frac{\eta}{Pr(X_{n+1}=0)u^{\geq}(0)} \text{(Pela Proposição 20)} \\
&= \eta/\beta.
\end{aligned}$$

Logo, se $\eta/\beta \geq T_{MR}^L$, então $E(T_{MR}) \geq T_{MR}^L$. A seguir comentamos sobre $E(T_M) \leq v^{\geq}(0)\eta/\gamma$:

$$\begin{aligned}
E(T_M) &\leq \frac{v(0)\eta}{q(0)u(0)} \text{(Pela Proposição 17)} \\
&= \frac{v(0)\eta}{Pr(X_{n+1}=0)Pr(D<\delta+\eta)u(0)} \text{(Pela Proposição 6)}
\end{aligned}$$

Então, pela Proposição 20, pela aplicação da desigualdade unilateral, e pelo uso de k_0, usamos a seguinte aproximação:

$$\begin{aligned}
&\frac{v^{\geq}(0)\eta}{Pr(X_{n+1}=0)\left(\frac{(\delta+\eta-E(D))^2}{(V(D)+(\delta+\eta-E(D))^2)}\right)u^{\geq}(0)} \\
&= v^{\geq}(0)\eta/\gamma.
\end{aligned}$$

Se $v^{\geq}(0)\eta/\gamma \approx T_M^U$, então $E(T_M) \approx T_M^U$. □

A partir da asserção (4.5) e da Proposição 20, obtemos a Proposição 22 a seguir, que será usada pelo configurador NSM-OSI-NFD-S[43].

Proposição 22. *Seja* $k_0' = \lceil (T_D^U - E(D))/\eta \rceil - 1$. *A seguir,* $u^{\geq'}(0)$ *e* $v^{\geq'}(0)$ *consideram, as mensagens* 0 *to* $k_0'-1$, *e* w *é inicialmente* k_0'. $v^{\geq'}(0) = Pr(X_{n+1}=0)v_{0,k_0}^{\geq'}(0) + \sum_{s=1}^{h} Pr(X_{n+1} \geq s)v_{s,k_0}^{\geq'}(0)$. $v_{s,w}^{\geq'}(0)$, *que é baseada na Proposição 20, é definida como*

[43]Do inglês: *New Sotoma and Madeira One-Sided Inequality for NFD-S.*

segue:

$$
\begin{aligned}
v^{\geq'}_{s,1}(0) &= forw(s,s+1) + to0(s)(V(D)/(V(D)+(T_D^U - E(D) - k_0'\eta)^2)), \text{ para } w=1;\\
v^{\geq'}_{s,w}(0) &= to0(s)(V(D)/(V(D)+(T_D^U - E(D) - (k_0' - w + 1)\eta)^2))u^{\geq'}_{w-1}(0)\\
&\quad + \sum_{a=1}^{w-2} forw(s,s+a)to0(s+a)(V(D)/(V(D)+(T_D^U - E(D) - (a + k_0' - w + 1)\eta)^2))u^{\geq'}_{w-(a+1)}(0)\\
&\quad + forw(s,s+w-1)to0(s+w-1)(V(D)/(V(D)+(T_D^U - E(D) - k_0'\eta)^2))\\
&\quad + forw(s,s+w), \text{ para } w>1.
\end{aligned}
$$

Os termos $v^{\geq'}_{w-1}(0)$ *e* $v^{\geq'}_{w-(a+1)}(0)$ *de* $v^{\geq'}_{s,w}(0)$ *usam a seguinte definição de* $u^{\geq'}(0)$*, que é baseada na Proposição 6:*

$$
\begin{aligned}
u^{\geq'}_{1}(0) &= forw(0,1) + to0(0)(V(D)/(V(D)+(T_D^U - E(D) - k_0'\eta)^2)), \text{ para } w=1;\\
u^{\geq'}_{w}(0) &= to(0)(V(D)/(V(D)+(T_D^U - E(D) - (k_0' - w + 1)\eta)^2))u^{\geq'}_{w-1}(0)\\
&\quad + \sum_{a=1}^{w-2} forw(0,a)to0(a)(V(D)/(V(D)+(T_D^U - E(D) - (a + k_0' - w + 1)\eta)^2))u^{\geq'}_{w-(a+1)}(0)\\
&\quad + forw(0,w-1)to0(w-1)(V(D)/(V(D)+(T_D^U - E(D) - k_0'\eta)^2))\\
&\quad + forw(0,w), \text{ para } w>1.
\end{aligned}
$$

Configurador NSM-OSI-NFD-S. A partir das asserções (4.4), (4.5), (4.6), (4.7), da Heurística 22 e da Proposição 23, obtemos o seguinte configurador, chamado de configurador NSM-OSI-NFD-S, para encontrar η e δ:
Passo 1: Fazer $g(\eta) = v^{\geq'}(0)\eta/\gamma'$, para $\gamma' = Pr(X_{n+1}=0)\left(\frac{(T_D^U - E(D))^2}{(V(D)+(T_D^U - E(D))^2)}\right)u^{\geq'}(0)$. Se $\gamma' = 0$, então fornecer como saída *"QoS não pode ser alcançada"* e parar. Caso contrário, encontrar o maior $\eta_{max} \leq T_D^U$ tal que $g(\eta_{max}) \leq T_M^U$.
Passo 2: Seja $f(\eta) = \eta/\beta'$, para $\beta' = Pr(X_{n+1}=0)u^{\geq'}(0)$. Encontrar o maior $\eta \leq \eta_{max}$ tal que $f(\eta) \geq T_{MR}^L$.
Passo 3: Atribuir $\delta = T_D^U - \eta$ e fornecer como saída η e δ.

Teorema 23. *Considere um sistema em que os relógios estão sincronizados e a probabilidade de perda de mensagens* p_L *e a distribuição de probabilidade de comprimentos de rajadas de perdas são conhecidas, e a distribuição de atrasos de mensagens* $Pr(D \leq x)$ *seja desconhecida. Suponha que seja dado um conjunto de requisitos de QoS como em (4.4). O configurador NSM-OSI-NFD-S possui dois resultados possíveis: 1) Ele fornece*

como saída η e δ. Neste caso, com parâmetros η e δ, o detector de defeitos NFD-S aproxima-se dos requisitos de QoS solicitados. 2) Ele fornece como saída "QoS não pode ser alcançada".

Prova. Nós provamos o Teorema 23 baseando-se na prova do Teorema 19, nas duas partes seguintes:

1. Suponha que o configurador NSM-OSI-NFD-S forneça como saída *"QoS não pode ser alcançada"*. Então, o configurador pára no *Passo 1* e, desta forma, $\gamma' = 0$. $\gamma' = 0$ implica que $Pr(X_{n+1} = 0) = 0$, $\left(\frac{(T_D^U - E(D))^2}{(V(D)+(T_D^U - E(D))^2)}\right) = 0$, ou $u^{\geq'}(0) = 0$. $Pr(X_{n+1} = 0) = 1 - p_L = 0$ implica que uma vez que q suspeitar de p, $E(T_M) = \infty$ e desta forma, q falha para satisfazer $E(T_M) \approx T_M^U$. $\left(\frac{(T_D^U - E(D))^2}{(V(D)+(T_D^U - E(D))^2)}\right) = 0$ implica que $T_D^U = E(D)$, e $u^{\geq'}(0) = 0$ implica que não há suspeita alguma. Nestes dois últimos casos, $E(T_M) = g(\eta) = \infty$, que leva o detector de defeitos a falhar para satisfazer $E(T_M) \approx T_M^U$. Por isso, o detector de defeitos não pode satisfazer a QoS solicitada neste caso.

2. Suponha que o configurador NSM-OSI-NFD-S forneça como saída os parâmetros η e δ. Então, pelo *Passo 3*, temos que $T_D^U = \eta + \delta$. Pela parte 1 do Teorema 16, $T_D \leq T_D^U$ é satisfeito. Perceba que temos $\gamma' > 0$ visto que, caso contrário, $g(\eta) = \infty$, e o configurador NSM-OSI-NFD-S forneceria *"QoS não pode ser alcançada"* em vez de η e δ. Pela Heurística 21, pela Proposição 17 e pelo *Passo 1*, $E(T_M) \approx g(\eta) = v^{\geq'}(0)\eta/\gamma' \approx \frac{v(0)\eta}{q_0 u(0)} \leq T_M^U$. Logo, $E(T_M)$ aproxima-se de T_M^U. $f(\eta) = \eta/\beta'$, pela restrição (4.2) e pela Heurística 21, $E(T_{MR}) \geq f(\eta) = \eta/\beta' = \eta/\beta$. Então pelo *Passo 2*, $E(T_{MR}) \geq T_{MR}^L$ é satisfeita. □

4.7 Modelo do Configurador para NFD-U

O detector de defeitos NFD-S e os configuradores propostos NSM-NFD-S e NSM-OSI-NFD-S supõem relógios sincronizados (por exemplo, via NTP). Desta forma, esta seção mostra como construir um configurador NSM-NFD-U para configurar o detector de defeitos NFD-U (veja a seção 3.1.10). NSM-NFD-U supõe que os relógios não estejam sincronizados.

De forma similar a Chen et al, a análise de QoS de NFD-U é obtida pela simples troca de δ por $E(D) + \alpha$ na Proposição 20 e na Heurística 21. Adicionalmente, o detector de defeitos NFD-U deve satisfazer o seguintes requisitos de QoS, os mesmos de Chen et al:

$$T_D \leq T_D^u + E(D), \quad E(T_{MR}) \geq T_{MR}^L, \quad E(T_M) \leq T_M^U. \tag{4.8}$$

O limite superior real do tempo de detecção T_D não é T_D^u, mas T_D^u mais o atraso médio

de mensagens *desconhecido* $E(D)$. Desta forma, o limite superior real T_D^U sobre o tempo de detecção é $T_D^u + E(D)$.

Aplicando $\delta = E(D) + \alpha$ na Proposição 20, obtemos a seguinte Proposição 24.

Proposição 24. *Seja* $k_0 = \lceil \alpha/\eta \rceil$. *A seguir,* $u^{\geq}(0)$ *e* $v^{\geq}(0)$ *consideram as mensagens* 0 *to* $k_0 - 1$, *e* w *é inicialmente* k_0. $v^{\geq}(0) = Pr(X_{n+1} = 0)v^{\geq}_{0,k_0}(0) + \sum_{s=1}^{h} Pr(X_{n+1} \geq s)v^{\geq}_{s,k_0}(0)$. $v^{\geq}_{s,w}(0)$, *que é baseada na Proposição 9, é definida como segue:*

$$
\begin{aligned}
v^{\geq}_{s,1}(0) &= forw(s, s+1) + to0(s)(V(D)/(V(D) + (\alpha - (k_0 - 1)\eta)^2)), \text{ para } w = 1;\\
v^{\geq}_{s,w}(0) &= to0(s)(V(D)/(V(D) + (\alpha - (k_0 - w)\eta)^2))u^{\geq}_{w-1}(0)\\
&\quad + \sum_{a=1}^{w-2} forw(s, s+a)to0(s+a)(V(D)/(V(D) + (\alpha - (a + k_0 - w)\eta)^2))u^{\geq}_{w-(a+1)}(0)\\
&\quad + forw(s, s+w-1)to0(s+w-1)(V(D)/(V(D) + (\alpha - (k_0 - 1)\eta)^2))\\
&\quad + forw(s, s+w), \text{ para } w > 1.
\end{aligned}
$$

Os termos $v^{\geq}_{w-1}(0)$ *e* $v^{\geq}_{w-(a+1)}(0)$ *de* $v^{\geq}_{s,w}(0)$ *usam a seguinte definição de* $u^{\geq}(0)$, *que é baseada na Proposição 6:*

$$
\begin{aligned}
u^{\geq}_{1}(0) &= forw(0, 1) + to0(0)(V(D)/(V(D) + (\alpha - (k_0 - 1)\eta)^2)), \text{ para } w = 1;\\
u^{\geq}_{w}(0) &= to(0)(V(D)/(V(D) + (\alpha - (k_0 - w)\eta)^2))u^{\geq}_{w-1}(0)\\
&\quad + \sum_{a=1}^{w-2} forw(0, a)to0(a)(V(D)/(V(D) + (\alpha - (a + k_0 - w)\eta)^2))u^{\geq}_{w-(a+1)}(0)\\
&\quad + forw(0, w-1)to0(w-1)(V(D)/(V(D) + (\alpha - (k_0 - 1)\eta)^2))\\
&\quad + forw(0, w), \text{ para } w > 1.
\end{aligned}
$$

Aplicando $\delta = E(D) + \alpha$ na Heurística 21, obtemos a seguinte Heurística 25.

Heurística 25. *Considere um sistema com relógios não sincronizados e suponha que* $\alpha > 0$. *Para o algoritmo NFD-U, temos* $E(T_{MR}) \geq \eta/\beta$ *e* $E(T_M) \approx v^{\geq}(0)\eta/\gamma$, *onde:*

$$
\begin{aligned}
\beta &= Pr(X_{n+1} = 0)u^{\geq}(0),\\
\gamma &= Pr(X_{n+1} = 0)\left(\frac{(\alpha + \eta)^2}{(V(D) + (\alpha + \eta)^2)}\right)u^{\geq}(0),\\
k_0 &= \lceil \alpha/\eta \rceil.
\end{aligned}
$$

Usando (4.5), $\delta + \eta \leq T_D^U$, $T_D^U = T_D^u + E(D)$, e $\delta = E(D) + \alpha$, temos que $\alpha + \eta = T_D^u$. Usando essa informação na Proposição 24 obtemos a Proposição 26 a seguir.

Proposição 26. *Seja* $k_0' = \lceil T_D^u/\eta \rceil - 1$. *A seguir,* $u^{\geq'}(0)$ *e* $v^{\geq'}(0)$ *consideram, as mensagens* 0 *to* $k_0' - 1$. $v^{\geq'}(0) = Pr(X_{n+1} = 0)v_{0,k_0'}^{\geq'}(0) + \sum_{s=1}^{h} Pr(X_{n+1} \geq s)v_{s,k_0'}^{\geq'}(0)$. $v_{s,w}^{\geq'}(0)$, *que é baseada na Proposição 9, é definida como segue:*

$$
\begin{aligned}
v_{s,1}^{\geq'}(0) &= forw(s, s+1) + to0(s)(V(D)/(V(D) + (T_D^u - k_0'\eta)^2)), \text{ para } w = 1;\\
v_{s,w}^{\geq'}(0) &= to0(s)(V(D)/(V(D) + (T_D^u - (k_0' - w + 1)\eta)^2))u_{w-1}^{\geq'}(0)\\
&\quad + \sum_{a=1}^{w-2} forw(s, s+a)to0(s+a)(V(D)/(V(D) + (T_D^u - (a + k_0' - w + 1)\eta)^2))u_{w-(a+1)}^{\geq'}(0)\\
&\quad + forw(s, s+w-1)to0(s+w-1)(V(D)/(V(D) + (T_D^u - k_0'\eta)^2))\\
&\quad + forw(s, s+w), \text{ para } w > 1.
\end{aligned}
$$

Os termos $v_{w-1}^{\geq'}(0)$ *e* $v_{w-(a+1)}^{\geq'}(0)$ *de* $v_{s,w}^{\geq'}(0)$ *usam a seguinte definição de* $u^{\geq'}(0)$, *que é baseada na Proposição 6:*

$$
\begin{aligned}
u_1^{\geq'}(0) &= forw(0, 1) + to0(0)(V(D)/(V(D) + (T_D^u - k_0'\eta)^2)), \text{ para } w = 1;\\
u_w^{\geq'}(0) &= to(0)(V(D)/(V(D) + (T_D^u - (k_0' - w + 1)\eta)^2))u_{w-1}^{\geq'}(0)\\
&\quad + \sum_{a=1}^{w-2} forw(0, a)to0(a)(V(D)/(V(D) + (T_D^u - (a + k_0' - w + 1)\eta)^2))u_{w-(a+1)}^{\geq'}(0)\\
&\quad + forw(0, w-1)to0(w-1)(V(D)/(V(D) + (T_D^u - k_0'\eta)^2))\\
&\quad + forw(0, w), \text{ para } w > 1.
\end{aligned}
$$

Configurador NSM-NFD-U. A partir das asserções (4.4), (4.5), (4.6), (4.7), da Heurística 25 e Proposição 26, obtemos o seguinte configurador, chamado de configurador NSM-NFD-U, para encontrar η e α:

Passo 1: Fazer $g(\eta) = v^{\geq'}(0)\eta/\gamma'$, para $\gamma' = Pr(X_{n+1} = 0)\left(\frac{(T_D^u)^2}{(V(D)+(T_D^u)^2)}\right)u^{\geq'}(0)$. Se $\gamma' = 0$, então fornecer como saída *"QoS não pode ser alcançada"* e parar. Caso contrário, encontrar o maior $\eta_{max} \leq T_D^u$ tal que $g(\eta_{max}) \leq T_M^U$.

Passo 2: Seja $f(\eta) = \eta/\beta'$, para $\beta' = Pr(X_{n+1} = 0)u^{\geq'}(0)$. Encontrar o maior $\eta \leq \eta_{max}$ tal que $f(\eta) \geq T_{MR}^L$.

Passo 3: Atribuir $\alpha = T_D^u - \eta$ e fornecer como saída η e α.

A Figura 4.4 apresenta um esquema da configuração de NFD-U. Neste caso os EA_{l+1}'s são conhecidos. As diferenças com a Figura 3.2 são que o estimador de comportamento probabilístico de *heartbeats* também fornece estimativas para $p_{L,z}$'s e o configurador usado é o NSM-NFD-U, que é diferente de Chen-NFD-U definido no capítulo 3.

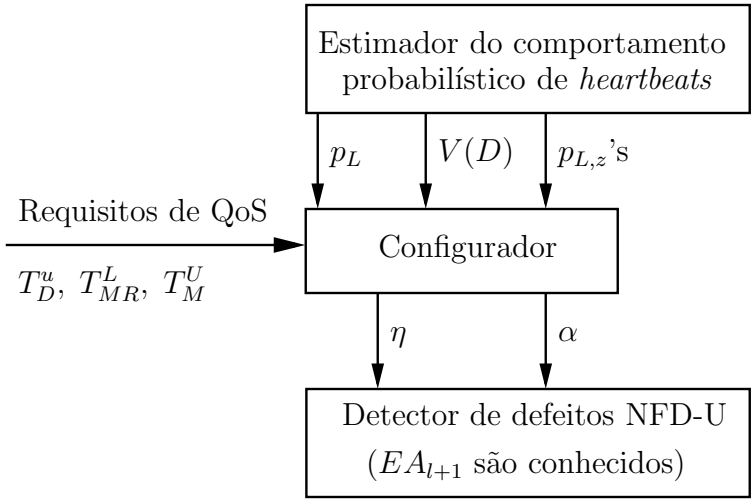


Figura 4.4: Configuração de NFD-U por NSM-NFD-U.

Teorema 27. *Considere um sistema em que os relógios não estão sincronizados e a probabilidade de perda de mensagens p_L e a distribuição de probabilidade de comprimentos de rajadas de perdas são conhecidas, e a distribuição de atrasos de mensagens $Pr(D \leq x)$ seja desconhecida. Suponha que seja dado um conjunto de requisitos de QoS como em (4.8). O configurador NSM-NFD-U possui dois resultados possíveis: 1) Ele fornece como saída η e α. Neste caso, com parâmetros η e α, o detector de defeitos NFD-U aproxima-se dos requisitos de QoS solicitados. 2) Ele fornece como saída "QoS não pode ser alcançada".*

Uma prova para o Teorema 27 pode ser derivada diretamente a partir da prova do Teorema 23.

A Figura 4.5 apresenta um esquema da configuração de NFD-E. A única diferença com a configuração de NFD-U (Figura 4.4) é que os EA_{l+1}'s são estimados. No capítulo a seguir, são apresentados vários estimadores de EA_{l+1} que podem ser usados.

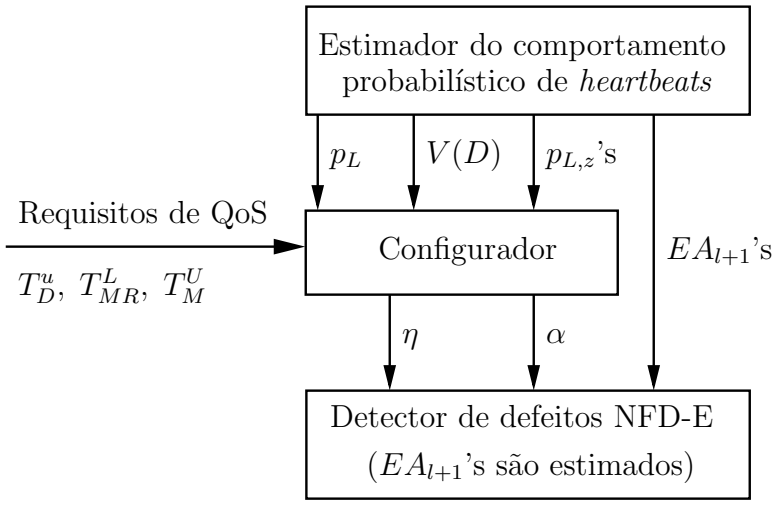


Figura 4.5: Configuração de NFD-E por NSM-NFD-E.

4.8 Conclusões

A informação adicional sobre comprimentos de rajadas de perdas pôde ser incorporada na análise de QoS dos algoritmos NFD-S, NFD-U e NFD-E de Chen et al [30]. O modelo proposto estende o modelo de Chen et al com cadeias de Markov para capturar o relacionamento entre perdas consecutivas de mensagens. Finalmente apresentamos os configuradores NSM-NFD-S, NSM-OSI-NFD-S, NSM-NFD-U e NSM-NFD-E. O Capítulo 5 apresenta a avaliação do modelo proposto.

A utilização da desigualdade unilateral, de forma similar a Chen et al, permite não supor qual é a distribuição de probabilidade dos atrasos das mensagens. Ao contrário do trabalho de Hayashibara et al [60], que supõe a distribuição normal para atrasos de mensagens.

Capítulo 5

Avaliação da Proposta

Este capítulo apresenta a avaliação do modelo para QoS de detectores de defeitos na presença de rajadas de perdas. A seção 5.1 apresenta uma visão geral do simulador desenvolvido. As seções 5.2 e 5.3 apresentam, respectivamente, a simulação com dados gerados usando algumas distribuições de probabilidades de tamanhos de rajadas de perdas de mensagens, e a simulação com dados coletados do link entre o Instituto de Computação (IC) da UNICAMP e o Departamento de Computação e Estatística (DCT) da UFMS. Em ambas as seções descreve-se a metodologia da simulação e a análise dos gráficos. A comparação é realizada com os configuradores de Chen et al, referidos como Chen-NFD-S, e Chen-NFD-E, e com os configuradores propostos NSM-NFD-S e NSM-NFD-E. Finalmente, a seção 5.4 apresenta algumas conclusões sobre o modelo proposto.

5.1 Visão Geral do Simulador

Esta seção apresenta uma descrição do simulador desenvolvido para a comparação dos dados analíticos de QoS gerados pelos configuradores e os valores reais de QoS dos detectores de defeitos. A geração dos arquivos de atrasos e perdas (seção 5.1.1) e a geração dos valores analíticos (seção 5.1.2) foram implementadas usando `C++`. A simulação usando os arquivos de dados (seção 5.1.3) é implementada usando `Python (http://www.python.org/)` e `SimPy (http://simpy.sourceforge.net/)`. A geração dos gráficos a partir dos resultados obtidos (seção 5.1.4) é realizada por `gnuplot (http://www.gnuplot.info/)` após a aplicação de scripts `bash (http://www.gnu.org/software/bash/)` e `gawk`

(http://www.gnu.org/software/gawk/gawk.html).

5.1.1 Geração dos Dados de Atrasos e Perdas

No caso dos dados gerados, desenvolveu-se um gerador de arquivos de entrada, que são usados na comparação dos configuradores. O atraso de mensagens segue a distribuição exponencial (usa como parâmetro um atraso médio de mensagens), e a distribuição do comprimento de rajadas de perdas (usa como parâmetro um comprimento máximo de rajadas de perdas e o tipo da distribuição, que pode ser uniforme, geométrica ou Pareto). O gerador constrói então um arquivo que será usado na simulação: cada linha do arquivo contém -1 (para representar uma perda de mensagem), ou um valor maior que zero (para representar um atraso de mensagem). Esse arquivo será referido posteriormente como `ARQ_DADOS_GERADOS`.

No caso dos dados coletados, usou-se a versão 0.70 da ferramenta `rude` (`http://rude.sourceforge.net/rude`) para enviar pacotes e `crude` para coletá-los. A partir do arquivo gerado pela coleta de `crude`, ordena-se o arquivo pelo número de seqüência (para descobrir as perdas) e constrói-se um arquivo que contém o mesmo formato descrito no parágrafo anterior. Esse arquivo será referido posteriormente como `ARQ_DADOS_COLETADOS`.

5.1.2 Geração dos Valores Analíticos

No caso dos dados gerados, usam-se os valores arbitrários que foram usados para gerar `ARQ_DADOS_GERADOS`. Estes valores arbitrários são então usados pelos configuradores Chen-NFD-S e NSM-NFD-S para gerar valores esperados (analíticos) para T_M e T_{MR}.

No caso dos dados coletados, após uma varredura de `ARQ_DADOS_COLETADOS` obtêm-se a média e variância de atrasos, a probabilidade de perdas, o tamanho máximo de rajadas de perdas, e a distribuição do comprimento de rajadas de perdas. Estes valores obtidos são então usados pelos configuradores Chen-NFD-E e NSM-NFD-E para gerar valores esperados (analíticos) para T_M e T_{MR}.

5.1.3 Simulação Usando os Arquivos de Dados

Os algoritmos NFD-S e NFD-E são simulados usando respectivamente, `ARQ_DADOS_GERADOS` e `ARQ_DADOS_COLETADOS`. O simulador do processo p lê esses arquivos para atrasar ou perder as mensagens a serem enviadas ao simulador do processo

q. Na verificação de precisão, ao mesmo tempo que a simulação ocorre, os dados de T_M e T_{MR} são coletados. Na verificação de tempo de detecção, é calculada a média e o valor máximo observado após quebras arbitrárias (distribuição uniforme) em um período considerado. Foram usados diversos tipos de estimadores/preditores para realizar a comparação.

5.1.4 Geração dos Gráficos dos Resultados

A partir de arquivos de saída resultantes da simulação usando os arquivos de dados, são gerados vários arquivos intermediários e, ao final, arquivos Postscript com as comparações entre os valores de QoS previstos pelos configuradores e os valores observados durante a simulação.

5.2 Simulação com Dados Gerados

Esta seção apresenta: a configuração da simulação com dados gerados usando as distribuições geométrica e Pareto nos comprimentos das rajadas de perdas; as análises dos intervalos de recorrência de erros e de duração de erros, e do tempo de detecção; uma discussão geral sobre os resultados; e uma breve comparação com um trabalho anterior que usa o modelo de espaços de estados limitado.

5.2.1 Configuração da Simulação com Dados Gerados

A simulação possui configuração similar à usada por Chen et al [30]: o atraso de mensagens D segue uma distribuição exponencial, $T_D^U \in \{1, 3, 5\}$ com passos de $0,1$, $E(D) = 0,02$, $V(D) = 0,0004$, $\eta = 1$, e $\delta = T_D^U - \eta = T_D^U - 1$. A distribuição exponencial foi usada devido à sua representação analítica simples: $Pr(D < x) = 1 - e^{-\lambda x}$, for $x \geq 0$. No nosso caso usamos $\lambda = 1/E(D)$. Mais ainda, esta distribuição é frequentemente usada para modelar atrasos em canais de comunicação [103]. Como consideramos simulação, a unidade de tempo não é usada. Assume-se que todos os valores de tempo referem-se à mesma unidade de tempo, por exemplo, segundos. O configurador NSM-NFD-S usado nesta simulação é aquele definido na seção 4.3.

$h \in \{4, 8, 12\}$ é o comprimento de rajada de perdas máximo usado para gerar as rajadas (veja Definição 2). Nas Figuras 5.1 e 5.2, e Figuras 5.3 e 5.4, para cada valor de T_D^U, respectivamente, nós plotamos $E(T_{MR})$ para considerar a média de 300 intervalos de

recorrência de erros, e $E(T_M)$ para considerar a média de 300 intervalos de duração de erro. As distribuições de comprimento de rajadas de perdas usadas são a geométrica [11] e Pareto [88]. À medida que o comprimento de rajadas aumenta, a distribuição geométrica fornece uma diminuição brusca nas probabilidades de rajadas de perdas, enquanto a distribuição de Pareto fornece uma diminuição suave. Nós denominamos rajadas longas de perdas àquelas que possuem comprimento próximo a h. Então, a distribuição geométrica modela uma baixa ocorrência de rajadas longas de perdas, e a de Pareto modela uma alta ocorrência.

Nós também realizamos simulações para verificar se o limite solicitado T_D^U é satisfeito. Para cada valor de T_D^U, h, p_L e ambas as distribuições de rajadas de perdas, em 50 execuções com tempos de quebra arbitrários, o algoritmo NFD-S sempre satisfez T_D^U.

Para cada configuração de simulação, a seguinte seqüência de passos foi realizada:

1) Gerar aleatoriamente a carga da rede (os atrasos e as rajadas de perdas);

2) Simular a execução do algoritmo NFD-S usando a carga. $E(T_{MR})$ e $E(T_M)$ são calculados;

3) Executar os configuradores (Chen-NFD-S e NSM-NFD-S) e obter os valores de QoS analíticos para T_{MR}^L e T_M^U;

4) Plotar os resultados dos passos 2 e 3;

5) Simular a execução do algoritmo NFD-S usando a carga. T_D^U é calculado durante esta execução;

6) Plotar os resultados do passo 5; e

7) Analisar os gráficos gerados nos passos 4 e 6 comparando os valores analíticos de QoS com os valores de $E(T_{MR})$, $E(T_M)$, e T_D.

5.2.2 Cálculo das Probabilidades de Rajadas de Perdas

A distribuição geométrica foi calculada através da seguinte fórmula, adaptada de Trivedi [103]: $loss_burst_probability_i = (1 - p_L)(p_L^i)/i$, para $i \in [1, h]$. A distribuição Pareto foi calculada indiretamente através da fórmula a seguir, também baseada em Trivedi [103]: $state_probability_i = \alpha\beta^\alpha i^{-\alpha-1}$, para $i \geq \beta, \alpha > 0, i \in [1, h]$. β é um parâmetro de localização que indica o menor valor possível para o tamanho de rajadas. α é um parâmetro de forma da distribuição. A partir de $state_probability_i$'s, as $loss_burst_probability_i$'s foram calculadas iniciando com $i = h$ até $i = 1$, através da fórmula de perda acumulada na Tabela 1.

A distribuição Pareto usou os valores de parâmetros $\alpha = 1,06$ e $\beta = 1$, que foram

obtidas por Paxson [88] em pacotes TCP na Internet. Ambos os cálculos tomam cuidado para que as probabilidades de rajadas de perdas se encaixem no valor de p_L, de acordo com o cálculo de perda média na Tabela 1.

5.2.3 Análise dos Intervalos de Recorrência de Erros

Nesta seção, como os resultados com $h \in \{4,\ 12\}$ são similares aqueles com $h = 8$, nós mostramos somente os resultados para $h = 8$. Além disso, as Figuras irão usar o termo CHEN-analytic para se referir ao valor analítico gerado pelo configurador Chen-NFD-S, e o termo SM-analytic para se referir ao valor analítico gerado pelo configurador NSM-NFD-S. As Figuras 5.1, 5.2, 5.3 e 5.4 apresentam as curvas sobre intervalos de recorrência de erros. A partir de (4.4) na seção 4.4, dizemos que um configurador satisfaz T_{MR}^L quando a sua saída analítica, indicada como analítico nas Figuras 5.1 a 5.4, é menor ou igual a $E(T_{MR})$ obtida a partir da execução do algoritmo NFD-S.

As Figuras 5.1 e 5.2 mostram, respectivamente, os resultados de simulação para $p_L = 0,01$ e $p_L = 0,03$, sob a distribuição geométrica nos comprimentos das rajadas de perdas. Nestes casos, ambos os configuradores Chen-NFD-S e NSM-NFD-S satisfazem o T_{MR}^L. De fato, para $T_D^U \in [1,\ 2]$, ambos os configuradores são equivalentes. Entretanto, para $T_D^U \in [2,1,\ 3,5]$, o configurador NSM-NFD-S se aproxima melhor ao comportamento de NFD-S. Neste intervalo, o configurador Chen-NFD-S subestima o comportamento de NFD-S.

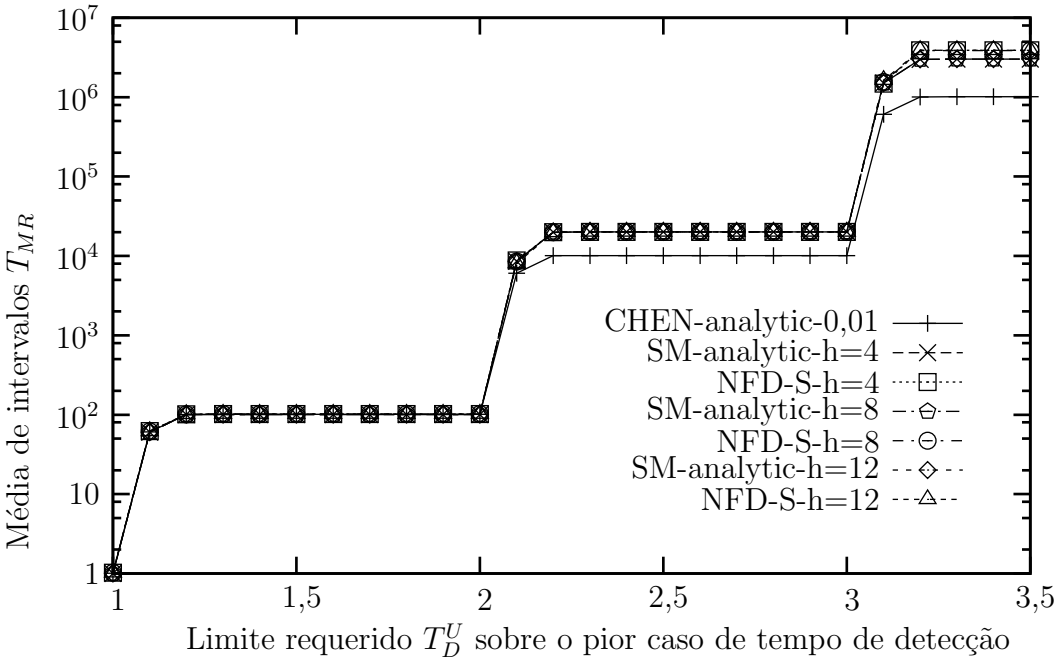


Figura 5.1: Verificação de $E(T_{MR}) \geq T_{MR}^L$ para distribuição geométrica e $p_L = 0,01$.

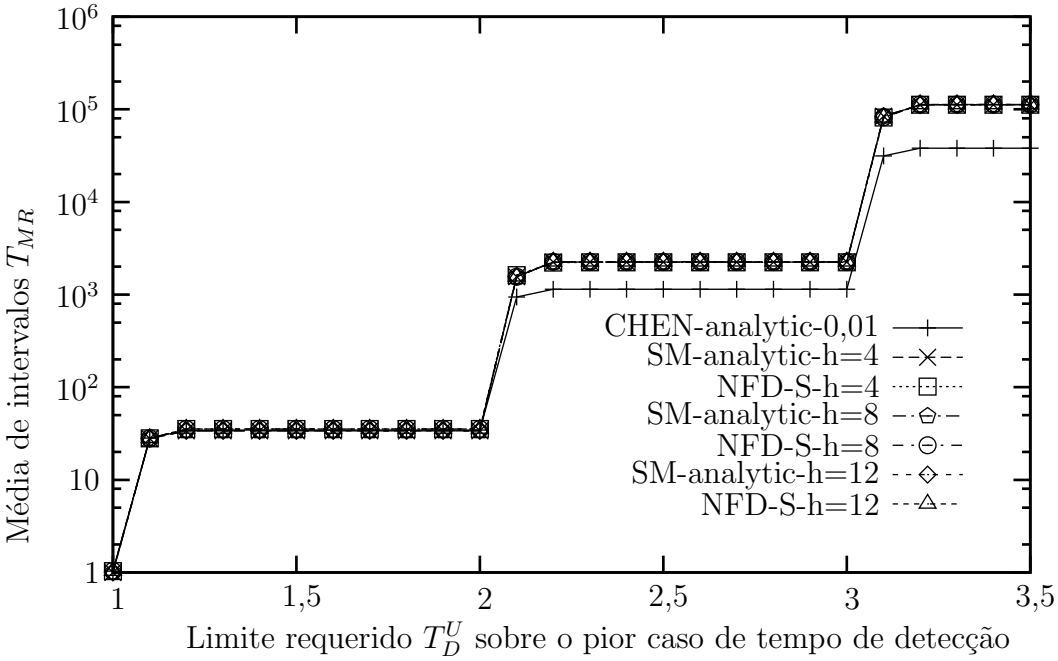


Figura 5.2: Verificação de $E(T_{MR}) \geq T_{MR}^L$ para distribuição geométrica e $p_L = 0,03$.

As Figuras 5.3, e 5.4 mostram, respectivamente, os resultados de simulação para $p_L = 0,01$ e $p_L = 0,03$, sob a distribuição Pareto nos comprimentos de rajadas de perdas. Nestes

casos, somente o configurador NSM-NFD-S satisfaz T_{MR}^L em todos os casos. De fato, para $T_D^U \in [1,\ 2]$, ambos os configuradores satisfazem. Entretanto, para $T_D^U \in [2,1,\ 3,5]$, o configurador NSM-NFD-S se aproxima melhor ao comportamento de NFD-S. Neste intervalo, o configurador Chen-NFD-S superestima o comportamento de NFD-S, levando-o a não satisfazer o T_{MR}^L.

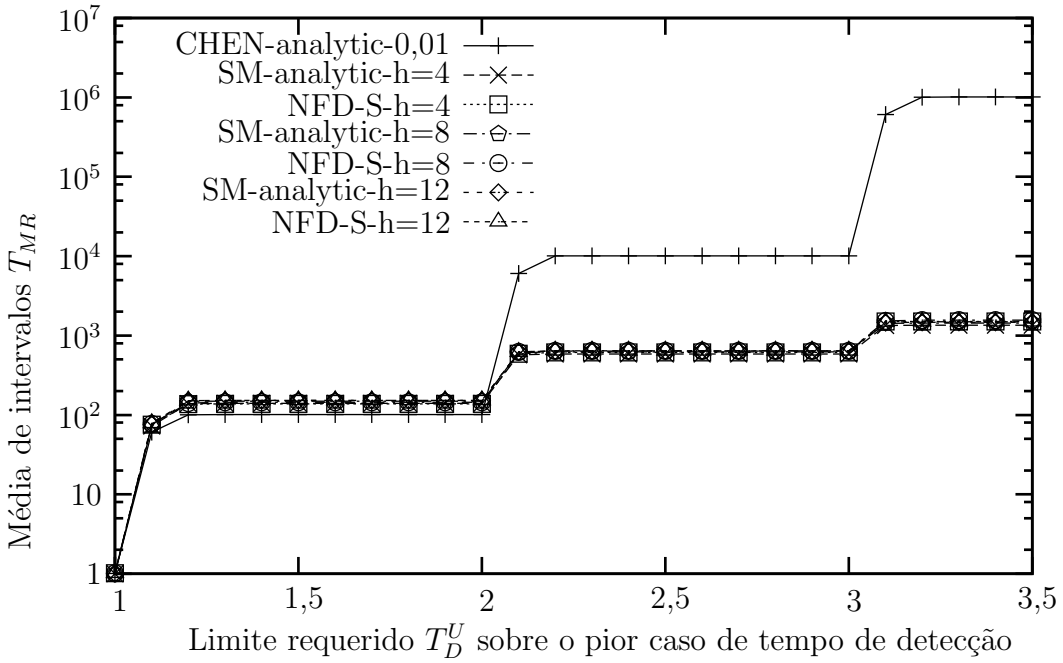


Figura 5.3: Verificação de $E(T_{MR}) \geq T_{MR}^L$ para distribuição Pareto e $p_L = 0,01$.

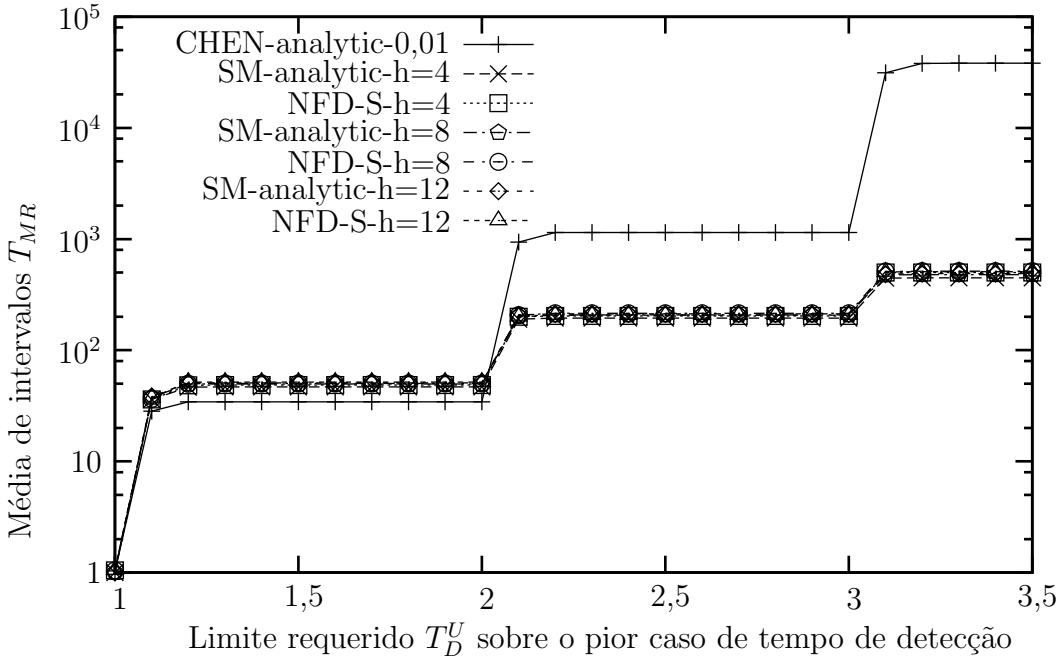


Figura 5.4: Verificação de $E(T_{MR}) \geq T_{MR}^L$ para distribuição Pareto e $p_L = 0,03$.

O configurador NSM-NFD-S satisfaz T_{MR}^L, de acordo com as Figuras 5.1 a 5.4, dentro do intervalo de confiança de 99%, o mesmo obtido por Chen et al na sua simulação.

5.2.4 Análise de Intervalos de Duração de Erros

As Figuras 5.5, 5.6, 5.7 e 5.8 apresentam as curvas sobre intervalos de duração de erros. Além disso, as Figuras irão usar o termo CHEN-analytic para se referir ao valor analítico gerado pelo configurador Chen-NFD-S, e o termo SM-analytic para se referir ao valor analítico gerado pelo configurador NSM-NFD-S. A partir da equação (4.4) na seção 4.4, dizemos que um configurador satisfaz T_M^U quando sua saída analítica, indicada como analítico nas Figuras 5.5 a 5.8, é maior ou igual a $E(T_M)$ obtido a partir da execução do algoritmo NFD-S.

As Figuras 5.5 e 5.6 mostram, respectivamente, os resultados de simulação para $p_L = 0,01$ e $p_L = 0,03$, sob a distribuição geométrica nos comprimentos de rajadas de perdas. Como neste caso os resultados com $h \in \{4,\ 12\}$ são similares aqueles com $h = 8$, nós mostramos somente os resultados para $h = 8$. As Figuras 5.5 e 5.6 indicam que ambos os configuradores satisfazem T_M^U, e que para $T_D^U \in [1,\ 3,5]$ ambos os configuradores são equivalentes.

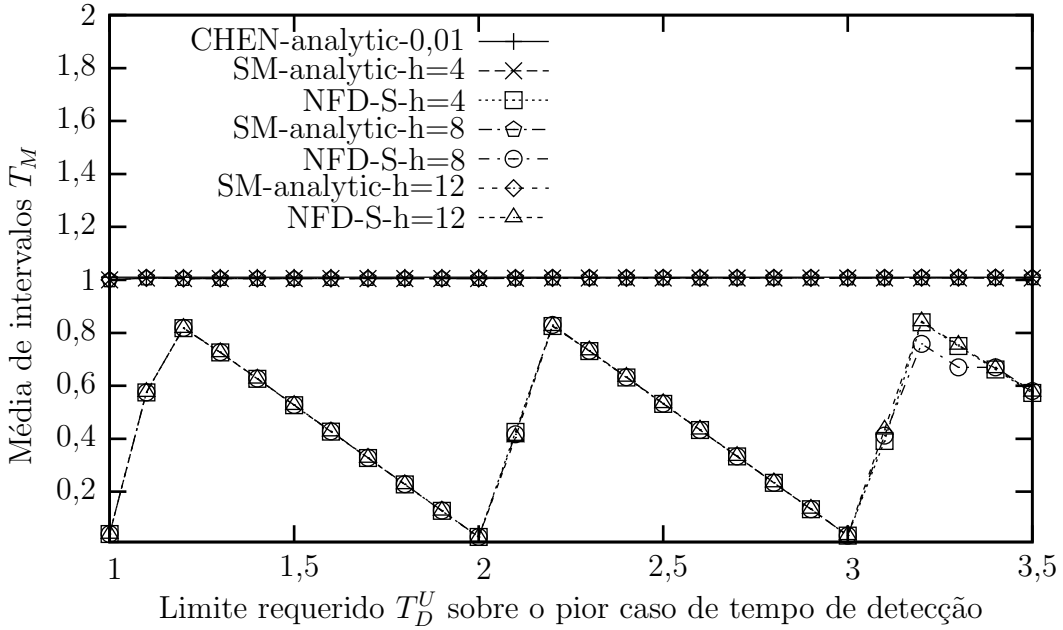


Figura 5.5: Verificação de $E(T_M) \leq T_M^U$ para distribuição geométrica e $p_L = 0,01$.

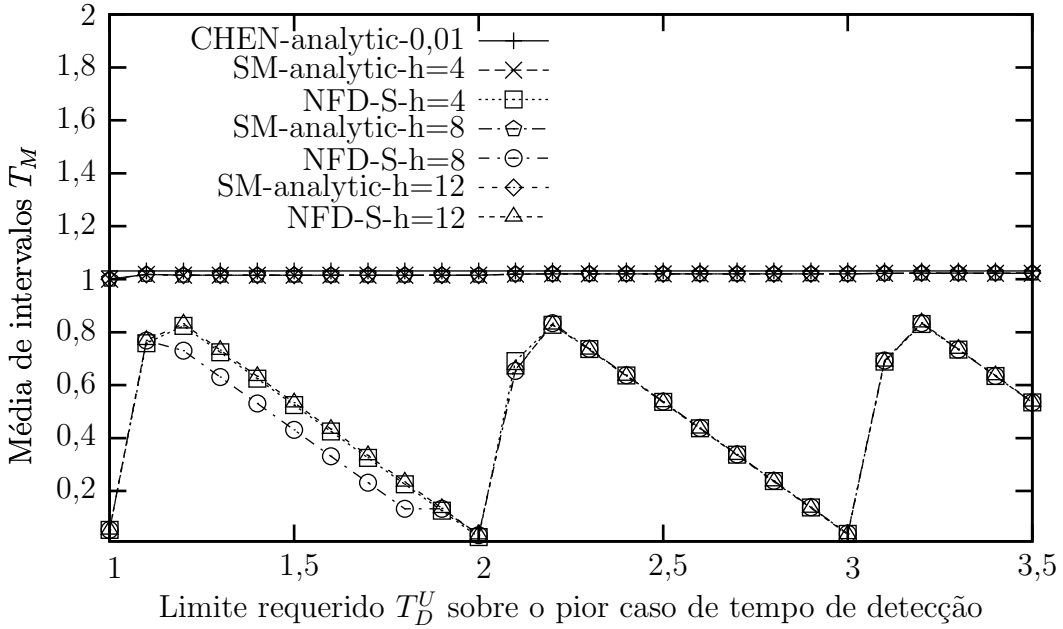


Figura 5.6: Verificação de $E(T_M) \leq T_M^U$ para distribuição geométrica e $p_L = 0,03$.

As Figuras 5.7 e 5.8 mostram, respectivamente, os resultados de simulação para $p_L = 0,01$ e $p_L = 0,03$, sob a distribuição Pareto nos comprimentos de rajadas de perdas. Como neste caso os resultados com $h \in \{4,\ 8,\ 12\}$ diferem um do outro, mostramos todos eles

nas Figuras 5.7 e 5.8. Essas Figuras indicam que somente o configurador NSM-NFD-S satisfaz T_M^U em todos os casos. De fato, em $26 \times 3 = 78$ pontos do comportamento de NFD-S, o configurador Chen-NFD-S satisfaz somente T_M^U em 33 casos para $T_D^U \in \{1,\ 1,1,\ 2,5\} \cup [1,4,\ 2] \cup [2,6,\ 3]$ na Figura 5.7, e 31 casos para os mesmos intervalos na Figura 5.8.

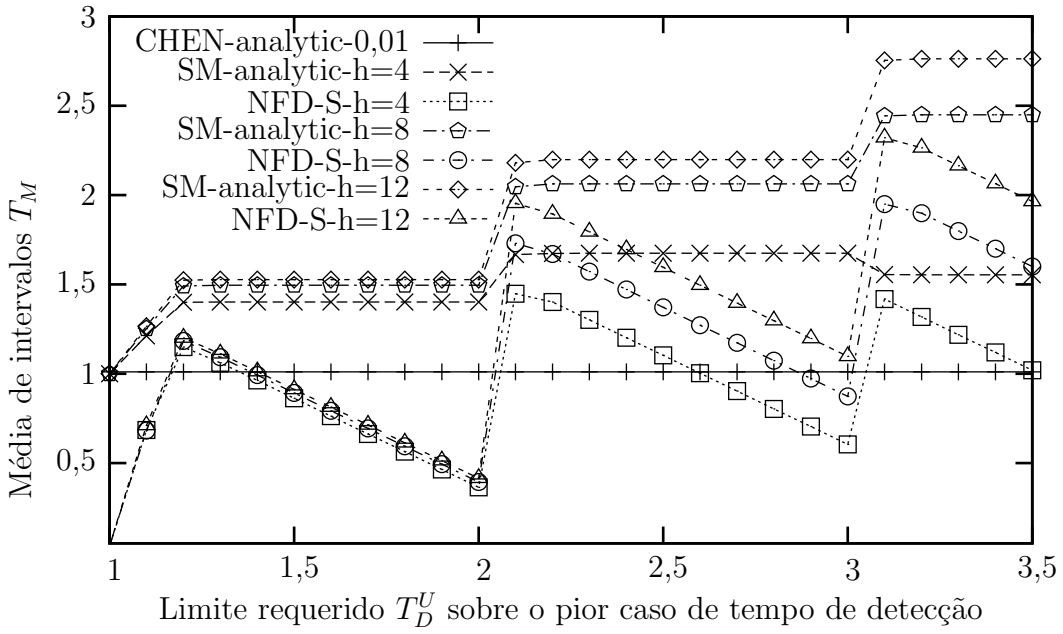


Figura 5.7: Verificação de $E(T_M) \leq T_M^U$ para distribuição Pareto e $p_L = 0,01$.

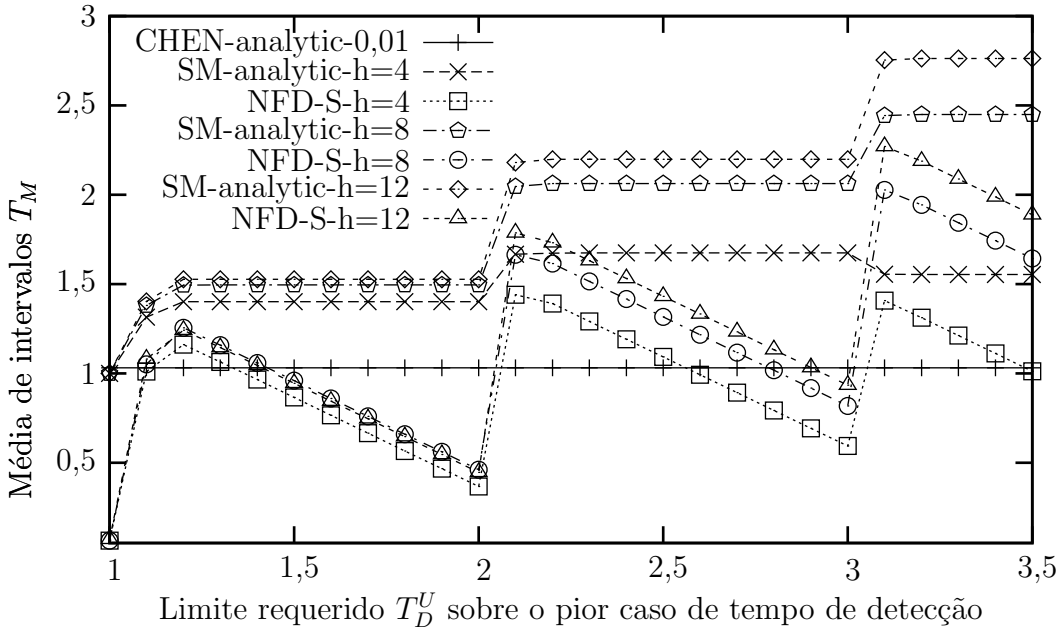


Figura 5.8: Verificação de $E(T_M) \leq T_M^U$ para distribuição Pareto e $p_L = 0,03$.

5.2.5 Discussão Geral sobre a Simulação com Dados Gerados

Os resultados apresentados na simulação com dados gerados mostra claramente que o configurador NSM-NFD-S proposto leva o algoritmo NFD-S a atender os requisitos de QoS tanto em termos de precisão, quanto no tempo de detecção. No entanto, o configurador Chen-NFD-S leva o algoritmo NFD-S a atender os requisitos de tempo de detecção, mas falha em termos de precisão (intervalo de recorrência de erros e intervalo de duração de erro).

A capacidade melhorada do configurador NSM-NFD-S deve-se claramente ao uso de comprimentos de rajadas de perdas no cálculo dos valores analíticos, o que não ocorre no configurador Chen-NFD-S. Mais ainda, quanto maior é o comprimento de rajadas de perdas, pior é o comportamento do configurador Chen-NFD-S. Ou de outro modo, quanto mais distante da distribuição geométrica, por exemplo Pareto, de comprimentos de rajadas de perdas, melhor é o comportamento do configurador NSM-NFD-S em relação ao configurador Chen-NFD-S. Além disso, o configurador NSM-NFD-S possui comportamento bem melhor em relação ao intervalo de duração de erro, o que é natural, visto que a ocorrência de rajadas de perdas leva NFD-S a demorar a se recuperar de uma transição-S.

5.2.6 Comparação entre o Modelo com Espaço de Estados Limitado e o Não Limitado

O modelo de Markov usado nesta tese foi o modelo de comprimento de perdas com espaço de estados não limitado, diferente do trabalho de Sotoma e Madeira [100], que usou o modelo limitado. Esta seção mostra que o modelo não limitado funciona melhor que o modelo limitado.

O trabalho anterior sobre o modelo limitado usou somente a distribuição uniforme nos comprimentos de rajadas de perdas, enquanto esta tese usa as distribuições geométrica e Pareto. Naquele trabalho, as simulações foram feitas com alguns valores arbitrários de T_M^U, T_{MR}^L, e T_D^U, que forneceram ambos os parâmetros η e δ diferentes para cada requisito de QoS. Diferente daquele trabalho, esta tese usa o modelo não limitado e simula usando uma abordagem similar a de Chen et al, que realiza análise de simulação a partir do uso de um valor de η fixo.

Através do uso da mesma configuração de simulação das seções anteriores, comparamos o comportamento dos dois modelos. O configurador NSM-NFD-S, para o algoritmo NFD-S, de Sotoma e Madeira [100] foi reexecutado com a configuração de simulação desta tese. A Figura 5.9 mostra as curvas de T_{MR} para a distribuição Pareto, para $h = 4$ e $p_L = 0,01$. O modelo não limitado proposto se encaixa melhor ao comportamento de NFD-S, enquanto o modelo limitado fornece valores menores em $T_D^U \in [3,1,\ 3,5]$. Isto significa que o modelo limitado fornece menores parâmetros η nestes casos, que levam a um maior custo devido a uma maior taxa de envio de heartbeats.

A Figura 5.10 mostra as curvas de T_M para a distribuição Pareto, para $h = 4$ e $p_L = 0,01$. O modelo não limitado proposto satisfaz $E(T_M) \leq T_M^U$ para todos os casos, enquanto o modelo limitado falha na garantia de QoS quando $T_D^U \in [3,1,\ 3,5]$.

Para ambos T_M e T_{MR}, o que vem a seguir é válido. A simulação com a distribuição Pareto, para $h = 4$ e $p_L = 0,03$, também age similarmente e não é mostrada. Nas outras simulações com as outras configurações, o modelo limitado age similarmente ao comportamento do modelo não limitado proposto.

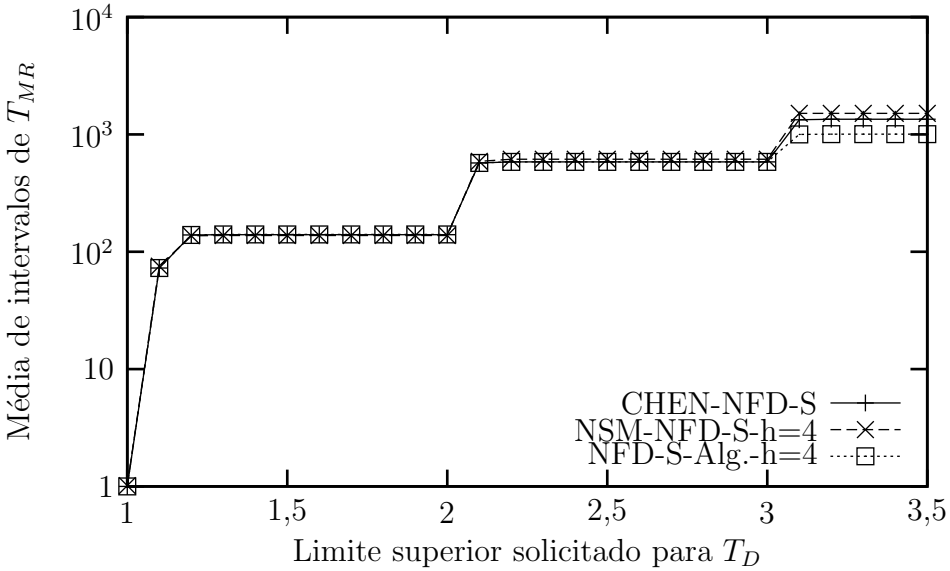


Figura 5.9: Comparação de T_{MR} com $p_L = 0,01$.

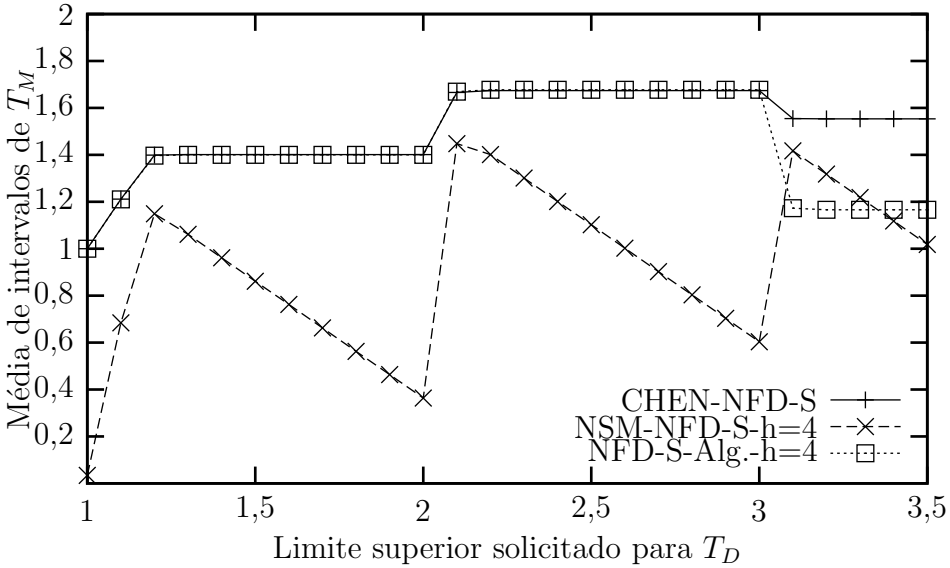


Figura 5.10: Comparação de T_M com $p_L = 0,01$.

5.3 Simulação com Dados Coletados

Esta seção apresenta: a configuração da simulação com dados coletados de uma rede real; as análises dos intervalos de recorrência de erros e de duração de erros, e do tempo de detecção; e uma discussão geral sobre os resultados.

5.3.1 Configuração da Simulação com Dados Coletados

A simulação com dados coletados usou dados do link entre o computador `prof.dct.ufms.br` do Departamento de Computação e Estatística (DCT) da UFMS e o computador `belem.ic.unicamp.br` do Instituto de Computação da UNICAMP. A coleta foi realizada entre os dias 29 de novembro e 13 de dezembro de 2005, com exceção dos dias 02 e 06 de dezembro. Os pacotes fluíram unidirecionalmente do DCT até o IC usando a versão 0.70 da ferramenta `rude (http://rude.sourceforge.net/rude)` para enviar pacotes e `crude` para coletá-los, com o computador `prof` sincronizando o seu horário com o computador `belem` com `rdate` a cada 1 minuto. O período usado em cada dia foi de 7:01 h às 24 h.

No topo de cada gráfico haverá as seguintes informações: p_L, h, E, V, $min(T_{MR})$ ou $min(T_M)$, e $max(T_{MR})$ ou $max(T_M)$. Foi usado o algoritmo NFD-E com estimação dos atrasos usando: a) a média de todos os atrasos vistos até determinado instante de tempo (MEAN), b) a média dos últimos 4 atrasos vistos até determinado instante de tempo (WINMEAN4), c) a média dos últimos 8 atrasos vistos até determinado instante de tempo (WINMEAN8), d) a média dos últimos 16 atrasos vistos até determinado instante de tempo (WINMEAN16), e) a média dos últimos 32 atrasos vistos até determinado instante de tempo (WINMEAN32), f) a média dos últimos 64 atrasos vistos até determinado instante de tempo (WINMEAN64), g) a média dos últimos 128 atrasos vistos até determinado instante de tempo (WINMEAN128), h) a média dos últimos 256 atrasos vistos até determinado instante de tempo (WINMEAN256), e i) apenas o último atraso visto em determinado instante de tempo (LAST).

Os valores de T_D^U usados variaram de 1,6 a 4, com passos de 0,1. O valor de 1,6 ocorreu porque consideramos o valor de $\eta = 1$, e a média dos atrasos ficou em torno de 0,5 segundos.

Os dados coletados na WAN mostram uma média alta nos atrasos das mensagens, cerca de 0,5 segundos, e uma variância alta, com cerca de 0,09 segundos (que fornece um desvio-padrão de cerca de 0,3 segundos). Ao contrário dos valores usados na seção anterior, que supunha uma rede bem menos carregada, com média de 0,02 e variância de 0,0004.

5.3.2 Análise do Tempo de Detecção

O tempo de detecção foi verificado através do cálculo do maior tempo de detecção observado em 200 execuções com tempos de quebra arbitrários para cada período de um

dia e para cada valor de tempo de detecção. Os dados mostram que o uso de uma janela para a estimativa do próximo atraso não leva o detector NFD-E a garantir as restrições de tempo de detecção. De fato, somente os estimadores MEAN e LAST fizeram NFD-E atender, praticamente de forma indistinguível, o tempo de detecção. Os estimadores WINMEANs sempre levaram NFD-E a transgredir o tempo de detecção.

Nas Figuras 5.11, 5.12, 5.13 a seguir, são mostrados os tempos de detecção máximo (obtido), médio (considera apenas os valores diferentes de zero) e máximo esperado (linha de referência), do NFD-E, para LAST, MEAN e WINMEAN32, respectivamente. Os demais gráficos com os outros estimadores tiveram comportamentos muito similares e não são mostrados.

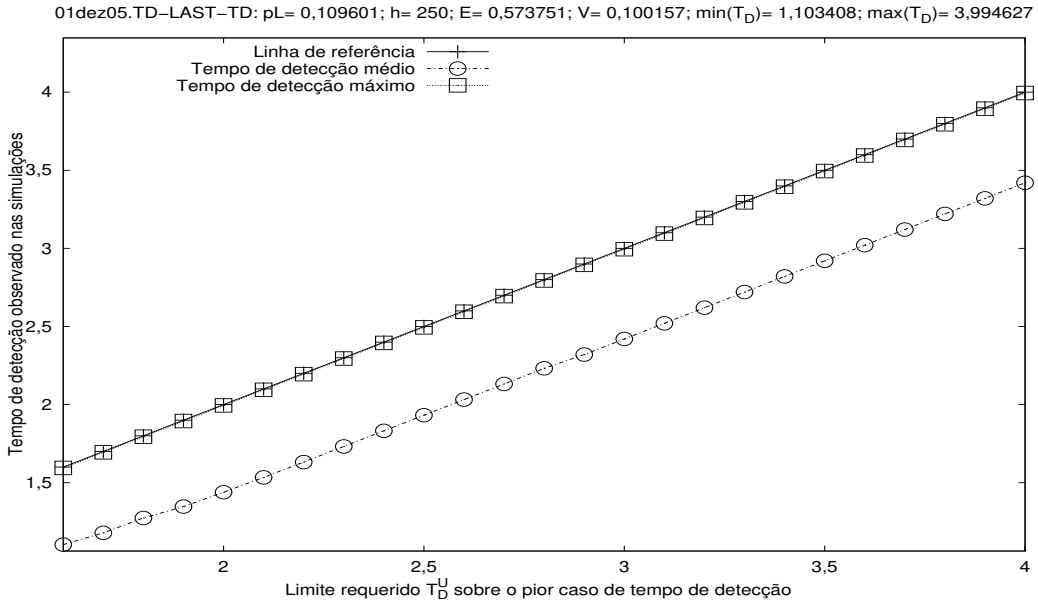


Figura 5.11: Tempo de detecção atendido por NFD-E ao usar o estimador LAST.

A Figura 5.11 mostra claramente que o tempo de detecção é atendido pelo NFD-E ao utilizar o estimador LAST. Da mesma forma, a Figura 5.12 mostra que o tempo de detecção é atendido pelo NFD-E com o estimador MEAN. A Figura 5.13 mostra que o tempo de detecção não é atendido pelo NFD-E ao utilizar o estimador WINMEAN32. Os demais estimadores WINMEAN* também apresentam um comportamento similar ao estimador WINMEAN32.

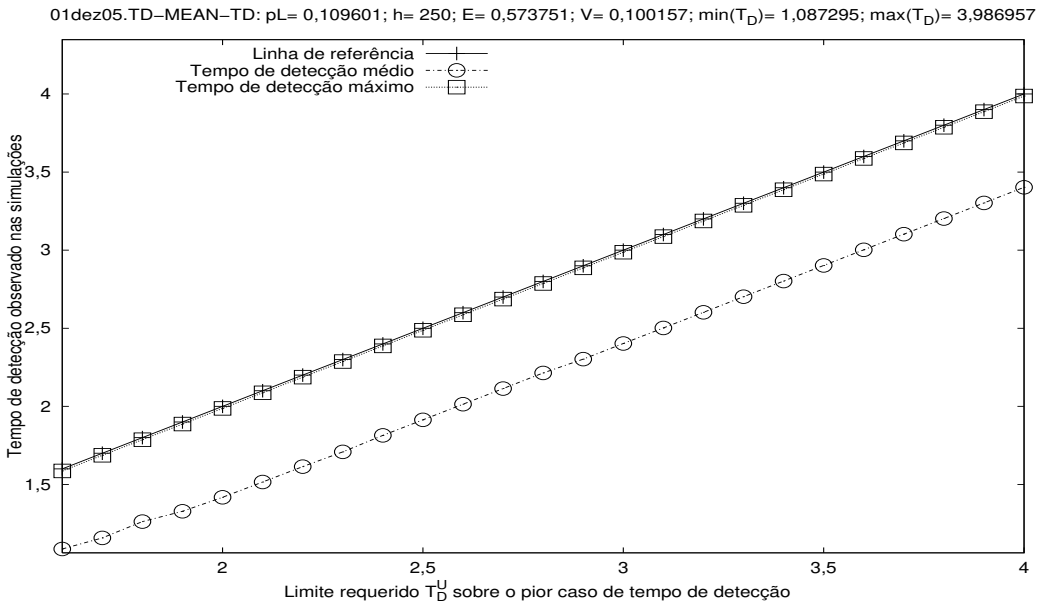


Figura 5.12: Tempo de detecção atendido por NFD-E com o estimador MEAN.

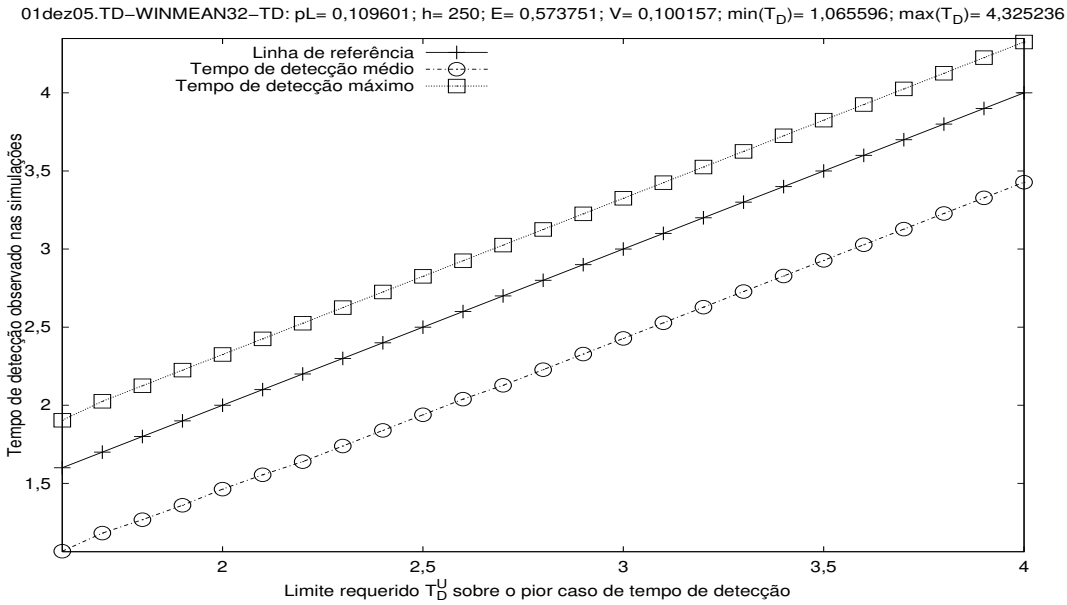


Figura 5.13: Tempo de detecção não atendido por NFD-E com WINMEAN32.

5.3.3 Análise dos Intervalos de Recorrência de Erros

O intervalo de recorrência de erros foi verificado através da média de intervalos de recorrência de erros observado no período de um dia para cada valor de tempo de detecção. As Figuras irão usar o termo CHEN-analytic para se referir ao valor analítico gerado pelo

configurador Chen-NFD-E, e o termo NSM-analytic para se referir ao valor analítico gerado pelo configurador NSM-NFD-E.

Os dados mostram que o tipo de cálculo de estimativa não influencia muito no atendimento do intervalo de recorrência de erros solicitado (representado pelo valor analítico nas Figuras a seguir). Contudo, o configurador NSM-NFD-E possui comportamento similar ao configurador Chen-NFD-E, e atende o requisito de T_{MR}^{L} em todos os casos. Mas o configurador Chen-NFD-E falha no dia 04 de dezembro, como mostra a Figura 5.14 a seguir. A Figura mostra o comportamento usando o estimador WINMEAN32 mas o comportamento é similar para os demais estimadores usados. No caso da Figura 5.14, $p_L = 0,007164$, mas $h = 12$, o que explica o fato do configurador Chen-NFD-E falhar, pois ele considera apenas o p_L baixo. O configurador NSM-NFD-E considera tanto p_L baixo quanto o comprimento das rajadas de perdas ($h = 12$), e por isso não superestima o valor de T_{MR}.

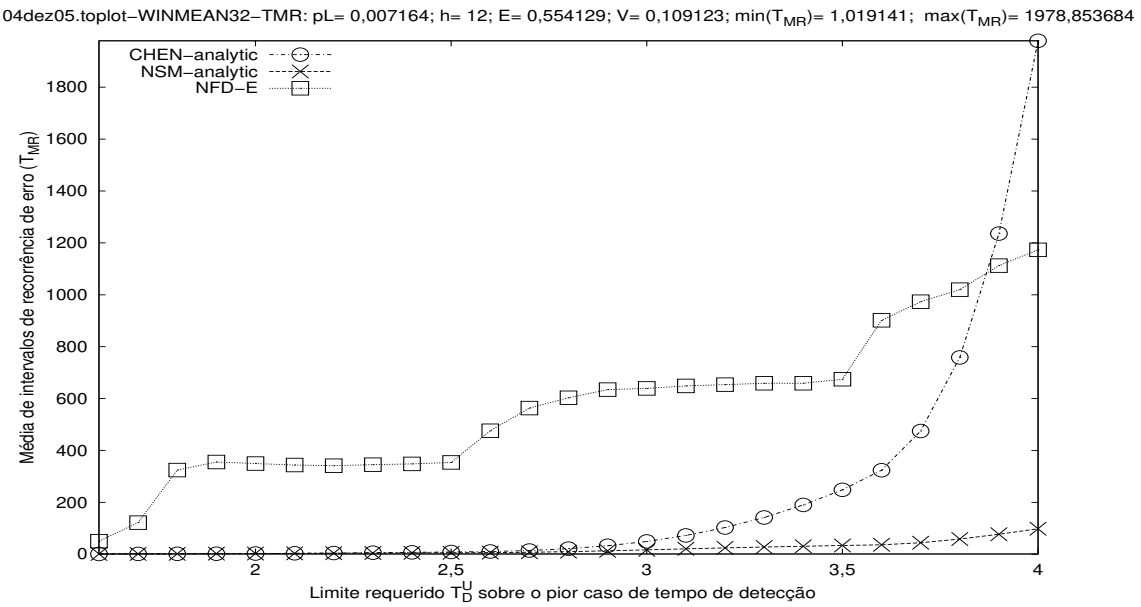


Figura 5.14: T_{MR}^{L} não atendido pelo configurador Chen-NFD-E usando NFD-E.

Os estimadores que usam janela (WINMEAN*), com tamanho não muito grande, possuem comportamento similar e geram gráficos em forma de escadas bem definidas (Figuras 5.15, e 5.16). Contudo, estimadores com janelas grandes, por exemplo 256 (Figura 5.17), e os estimadores LAST e MEAN possuem comportamento similar e geram gráficos mais suaves e com comportamento do NFD-E mais próximo aos valores analíticos (Figuras 5.18, 5.19).

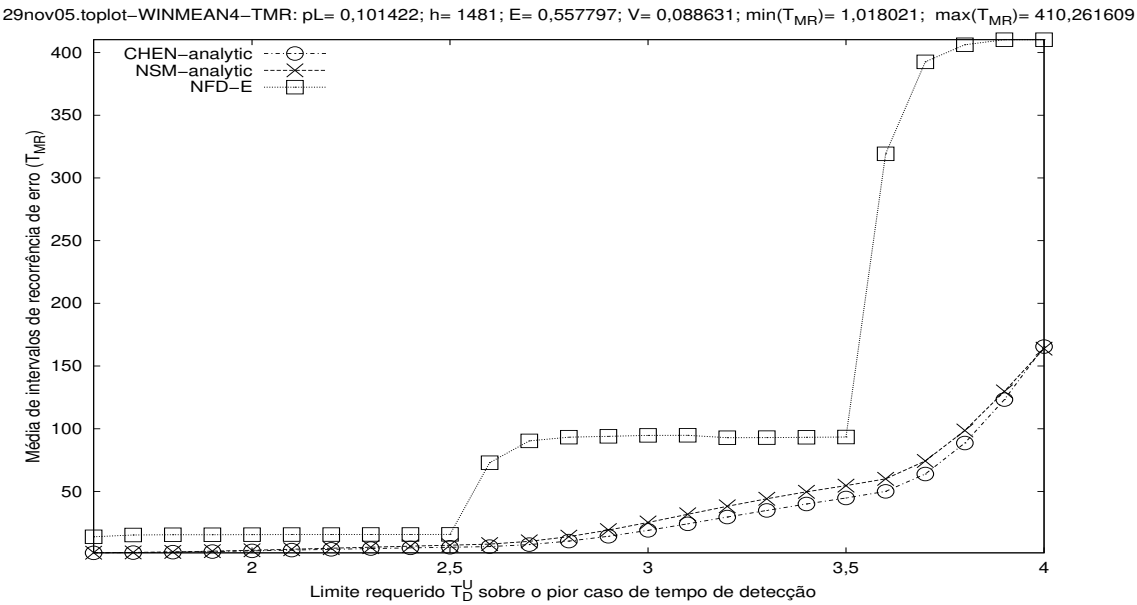


Figura 5.15: Comportamento de NFD-E em forma de escada.

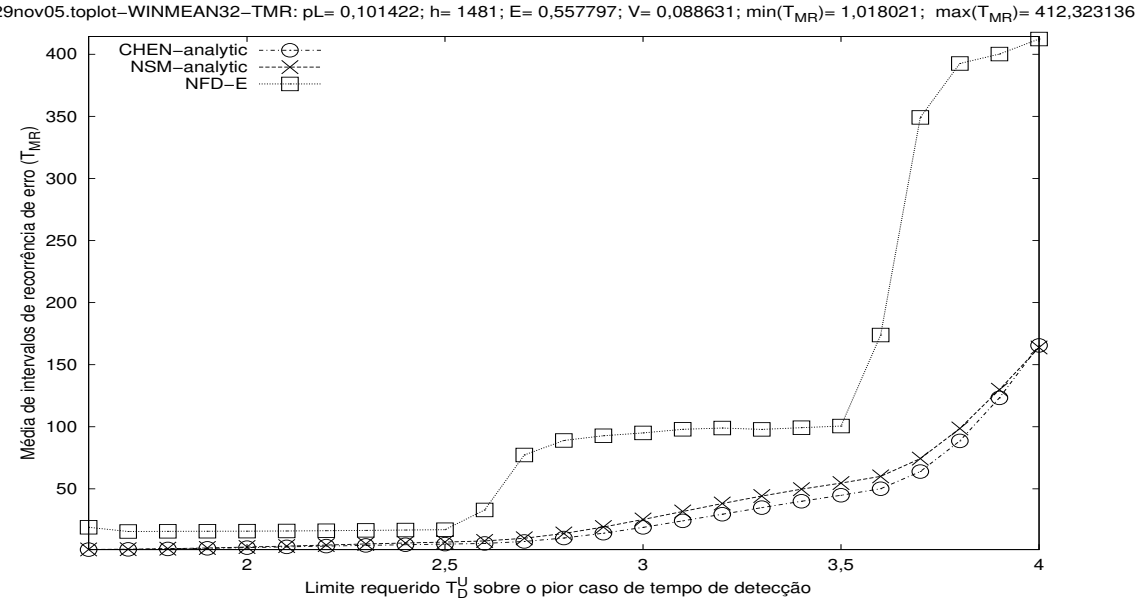


Figura 5.16: Comportamento de NFD-E em forma de escada.

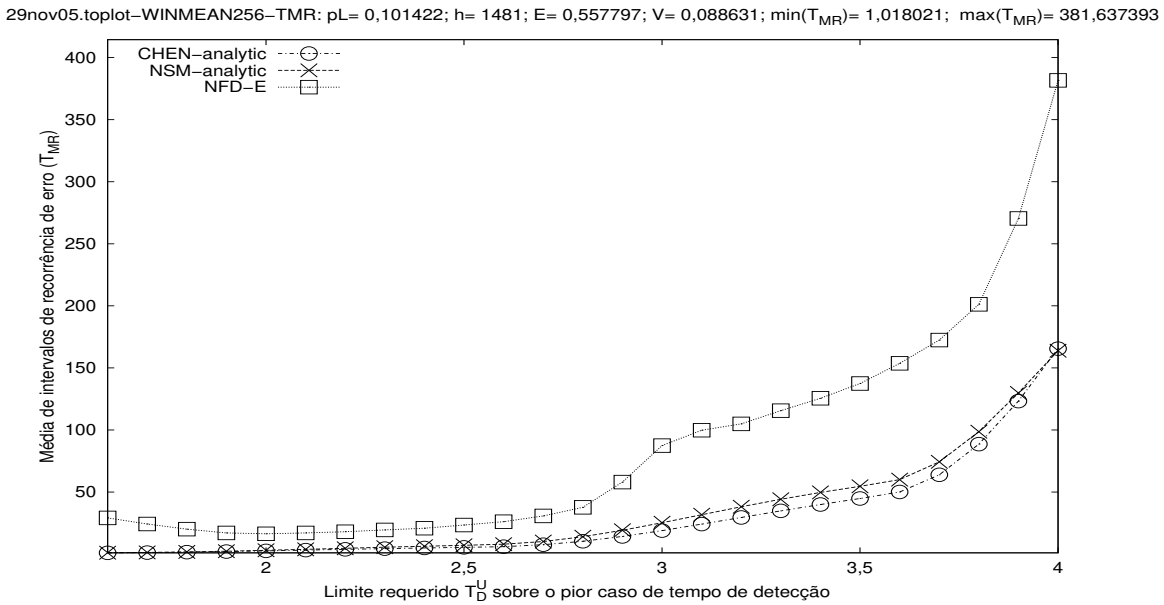


Figura 5.17: Comportamento de NFD-E mais próximo aos valores analíticos.

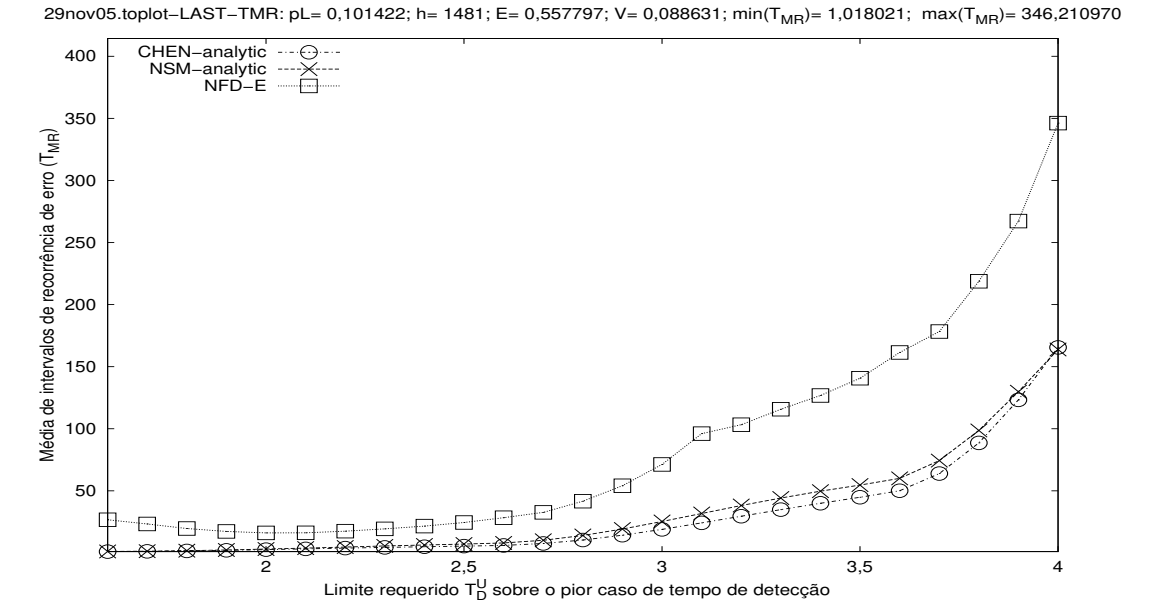


Figura 5.18: Comportamento de NFD-E mais próximo aos valores analíticos.

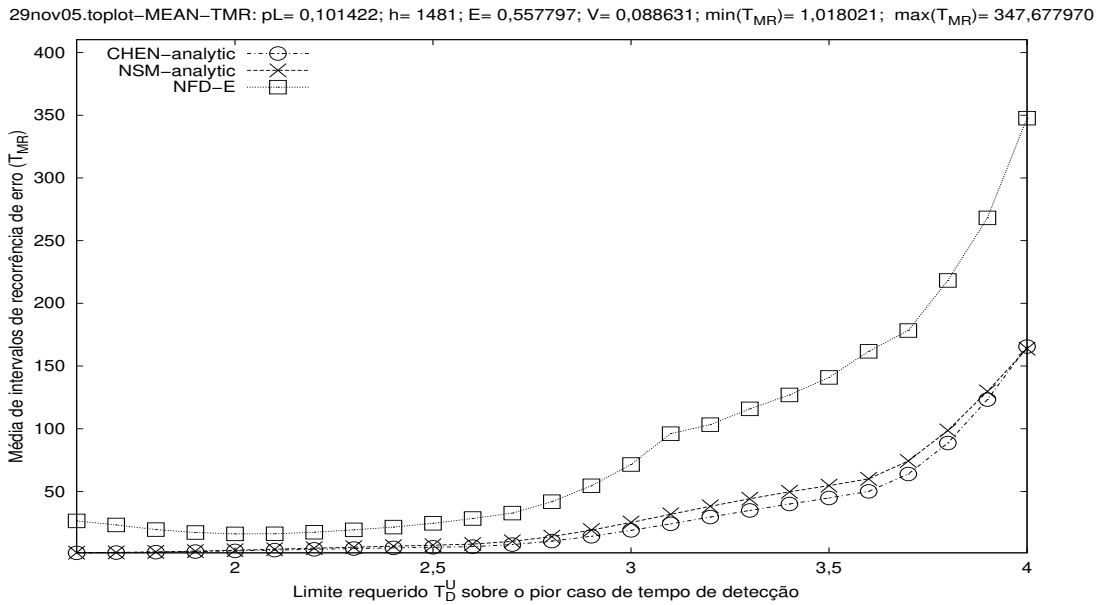


Figura 5.19: Comportamento de NFD-E mais próximo aos valores analíticos.

5.3.4 Análise de Intervalos de Duração de Erros

O intervalo de duração de erros foi verificado através da média de intervalos de duração de erros observado no período de um dia para cada valor de tempo de detecção. As Figuras irão usar o termo CHEN-analytic para se referir ao valor analítico gerado pelo configurador Chen-NFD-E, e o termo NSM-analytic para se referir ao valor analítico gerado pelo configurador NSM-NFD-E.

Os dados mostram que o tipo de cálculo de estimativa influencia bastante no atendimento do intervalo de duração de erros solicitado (representado pelo valor analítico nas Figuras a seguir). Além disso, a quantidade de intervalos de duração de erros relacionada com o tamanho máximo de rajadas e/ou variância também influencia bastante. Tanto o configurador NSM-NFD-E quanto o configurador Chen-NFD-E não conseguem atender o requisito de T_M^U em muitos casos. Mas o configurador NSM-NFD-E é claramente melhor que Chen-NFD-E quando há rajadas de perdas grandes.

Na Figura 5.20 a seguir, WINMEAN256 representa o comportamento de WINMEAN128, WINMEAN64, LAST e MEAN. WINMEAN32, WINMEAN16 e WINMEAN8 se aproximam de WINMEAN4 na Figura 5.21, que oferece a melhor estimativa em relação ao comportamento de NFD-E para valores de T_D^U pequenos. Neste caso, em que $p_L = 0,125$ é altíssima, e o tamanho máximo de rajadas de perdas não é muito grande $h = 25$, ambos os configuradores NSM-NFD-E e Chen-NFD-E

comportam-se de forma similar.

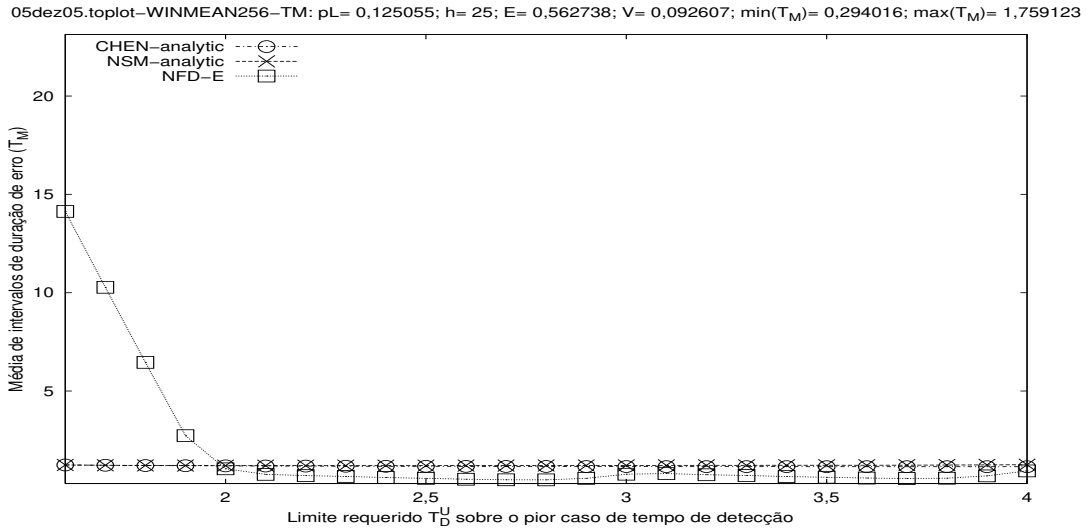


Figura 5.20: T_M^U não é atendido para valores pequenos de T_D^U.

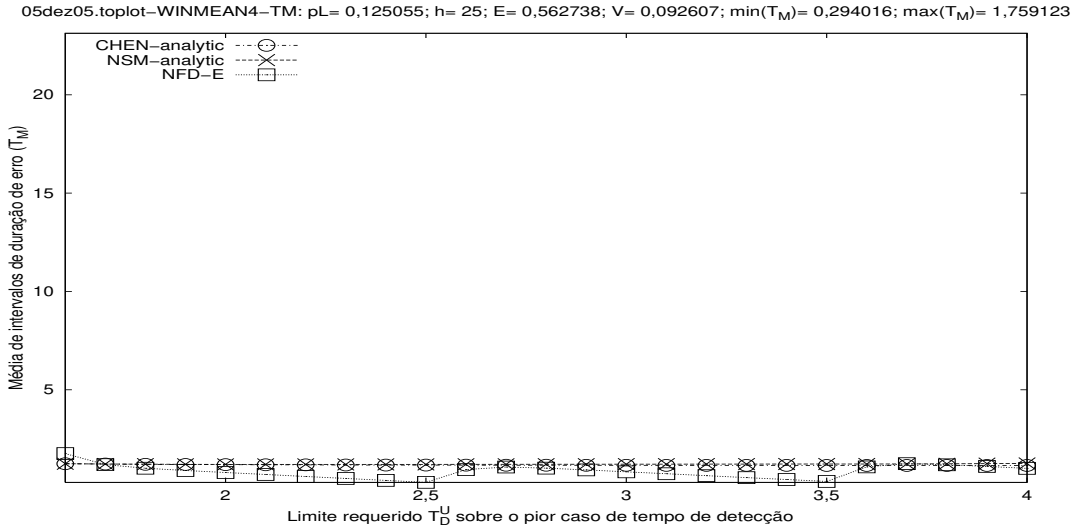


Figura 5.21: T_M^U é melhor atendido para valores pequenos de T_D^U.

Na Figura 5.22, WINMEAN256 representa o comportamento de WINMEAN128, WINMEAN64, LAST e MEAN. WINMEAN32, WINMEAN16 e WINMEAN8 se aproximam de WINMEAN4 na Figura 5.24, que oferece a melhor estimativa em relação ao comportamento de NFD-E para valores de T_D^U pequenos. Para T_D^U a partir de 2,5 e para T_D^U a partir de 3,5, WINMEAN 4 piora bastante, mas isso ocorre devido à pouca quantidade de intervalos de duração de erro (veja as Figuras 5.23 e 5.25) em relação

ao tamanho máximo de rajada ocorrida (468). Neste caso, em que $p_L = 0,03$ é alto, mas o tamanho máximo de rajadas de perdas é muito grande $h = 468$, o configurador NSM-NFD-E fornece valores analíticos melhores que o configurador Chen-NFD-E, a partir de $T_D^U = 2,5$.

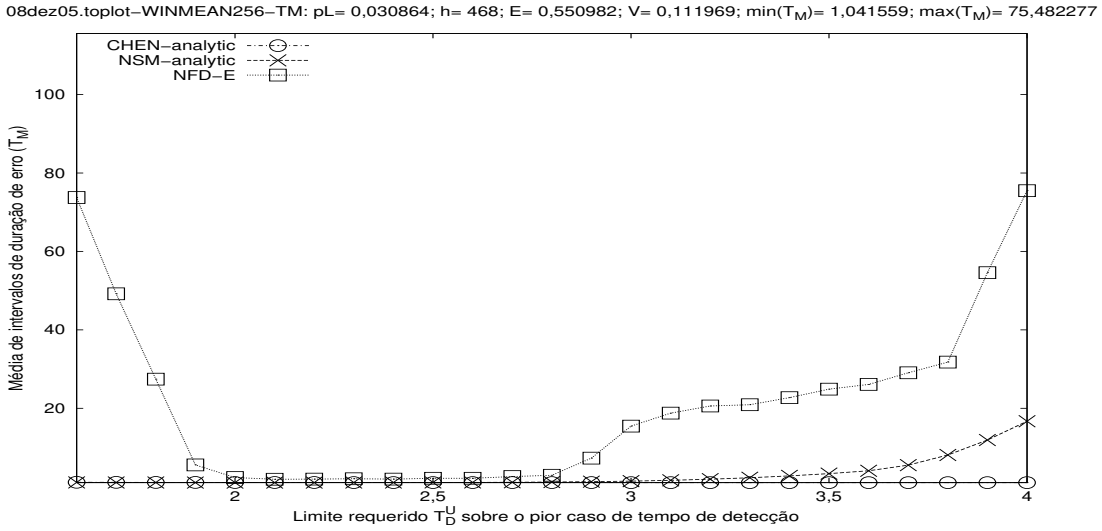


Figura 5.22: T_M^U não é atendido para valores pequenos de T_D^U.

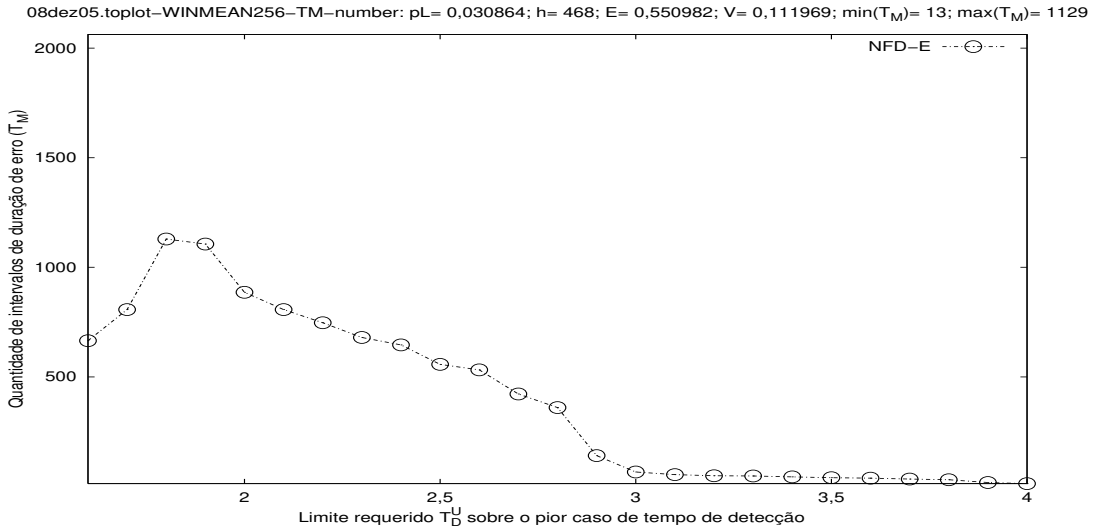


Figura 5.23: T_M^U não é atendido para valores pequenos de T_D^U.

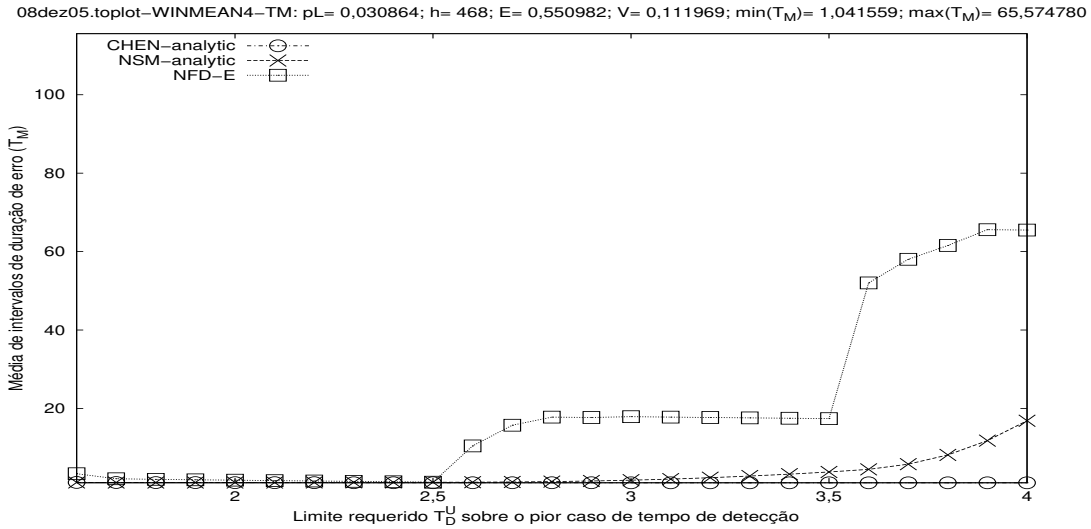


Figura 5.24: T_M^U é melhor atendido para valores pequenos de T_D^U.

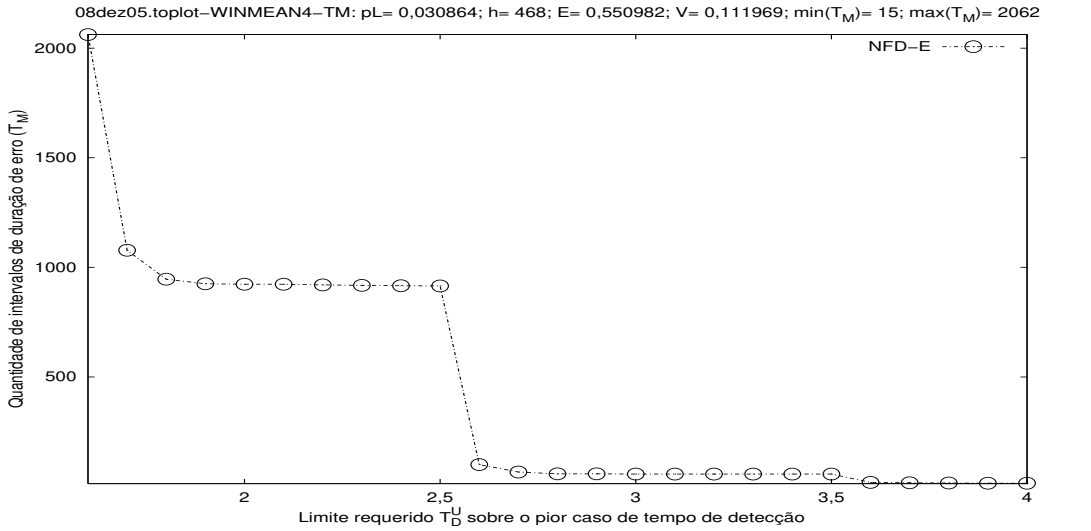


Figura 5.25: T_M^U é melhor atendido para valores pequenos de T_D^U.

Na Figura 5.26 a seguir, WINMEAN256 representa o comportamento de WINMEAN128, WINMEAN64, LAST e MEAN. WINMEAN32, WINMEAN16 e WINMEAN8 se aproximam de WINMEAN4 na Figura 5.28, que oferece a melhor estimativa em relação ao comportamento de NFD-E para valores de T_D^U pequenos. Para T_D^U a partir de 2,5, WINMEAN4 piora, mas isso ocorre devido à pouca quantidade de intervalos de duração de erro (veja as Figuras 5.27 e 5.29) que não é suficiente para fornecer uma média de T_M^U baixa devido à variância (0,114588), apesar do tamanho de rajada de perda máximo ser h=1, e da probabilidade de perda ser baixíssima (0,000720). Neste

caso, ambos os configuradores NSM-NFD-E e Chen-NFD-E fornecem valores analíticos similares, o que é natural porque a probabilidade de perdas é baixíssima ($p_L = 0,000720$) e o comprimento máximo de perdas é baixíssimo ($h = 1$).

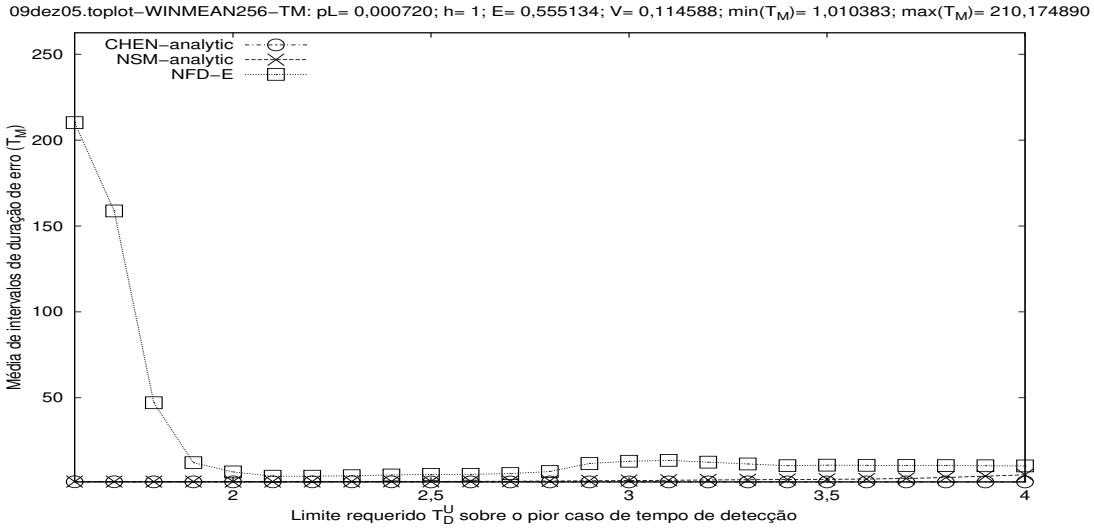


Figura 5.26: T_M^U não é atendido principalmente para valores pequenos de T_D^U.

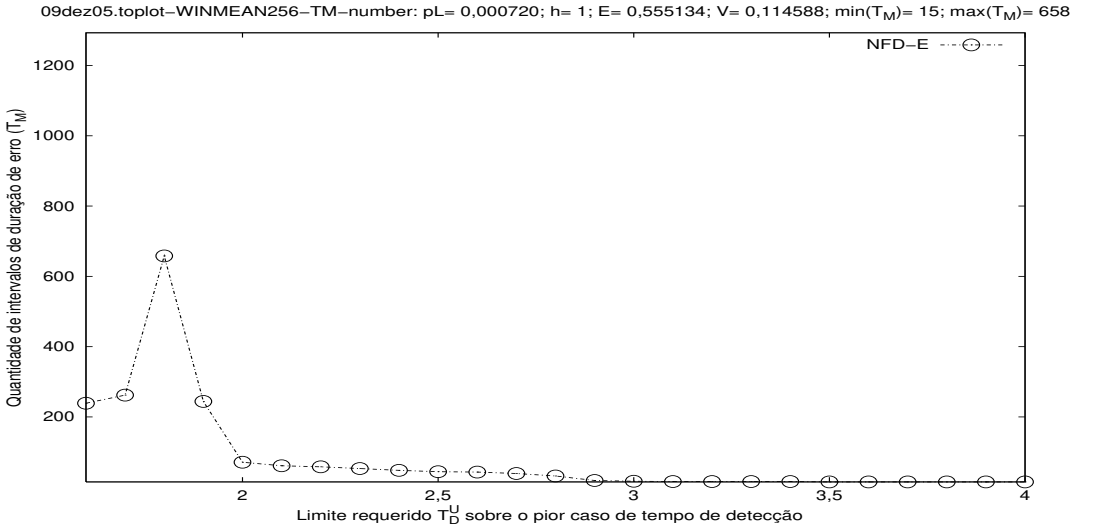


Figura 5.27: T_M^U não é atendido principalmente para valores pequenos de T_D^U.

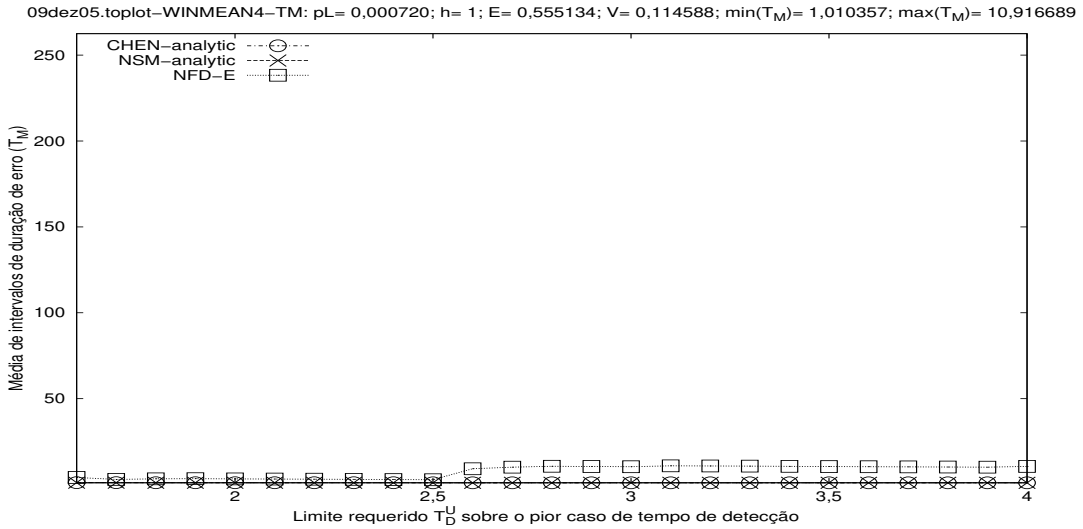


Figura 5.28: T_M^U é melhor atendido para valores pequenos de T_D^U.

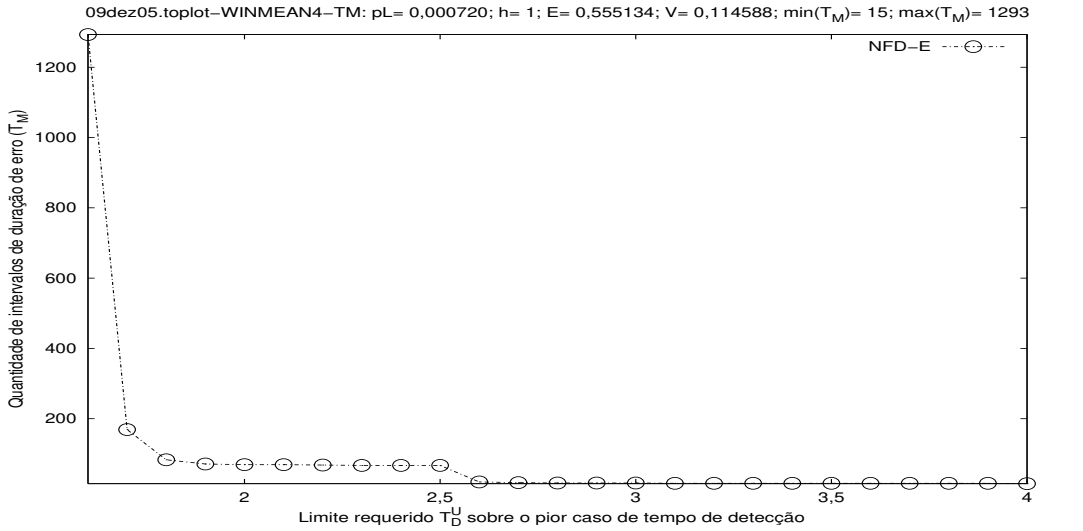


Figura 5.29: T_M^U é melhor atendido para valores pequenos de T_D^U.

Na Figura 5.30 a seguir, WINMEAN256 representa o comportamento de WINMEAN128, WINMEAN64, LAST e MEAN. WINMEAN32, WINMEAN16 e WINMEAN8 se aproximam de WINMEAN4 na Figura 5.32, que oferece a melhor estimativa em relação ao comportamento de NFD-E. A probabilidade de perda também é baixíssima (0,006510) e a variância é alta (0,095989), mas o tamanho de rajada de perda máximo é alto (115).

Com WINMEAN4, para $T_D^U = 1,6$ a variância influencia para que T_M^U não seja atendida, apesar da quantidade alta de intervalos de duração de erro (veja a Figura

5.33). Para $T_D^U = 1,7$ há uma brusca redução de intervalos de duração de erro, mas que ainda é acima de $h = 115$, o que não influencia muito no T_M. Mas a partir de $T_D^U = 1,8$, a quantidade de intervalos de duração de erro é menor que $h = 115$, o que influencia muito no T_M, e piora a partir de $T_D^U = 2,6$. Contudo, o configurador NSM-NFD-E consegue melhorar a previsão a partir de $T_D^U = 3$, em relação ao configurador Chen-NFD-E, que permanece constante. O configurador NSM-NFD-E consegue ser melhor do que Chen-NFD-E porque considera, além de p_L considerado por Chen-NFD-E, o comprimento das rajadas de perdas ($h = 115$), que Chen-NFD-E não considera.

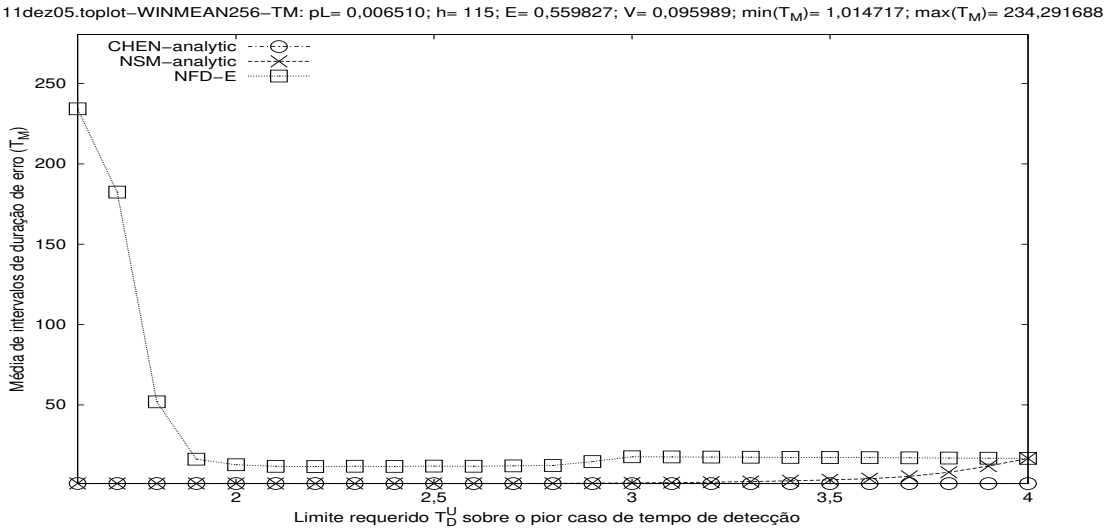


Figura 5.30: T_M é ruim com WINMEAN256.

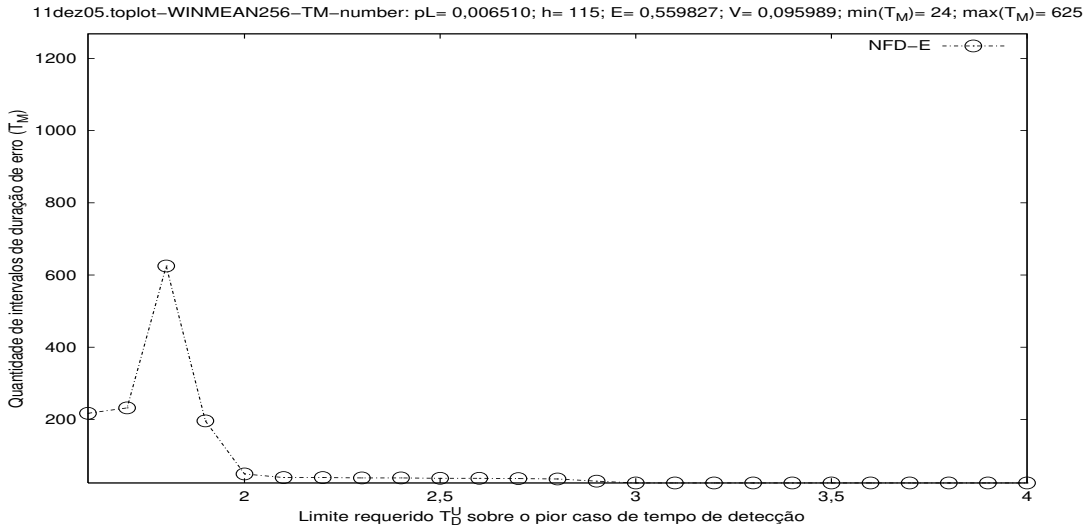


Figura 5.31: T_M é ruim com WINMEAN256.

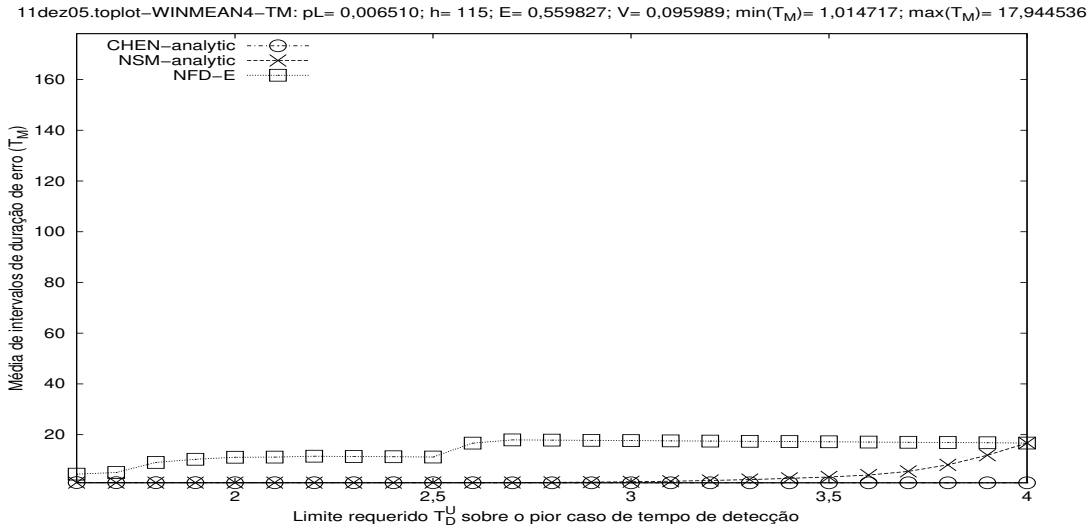


Figura 5.32: T_M é influenciado pela variância e por h.

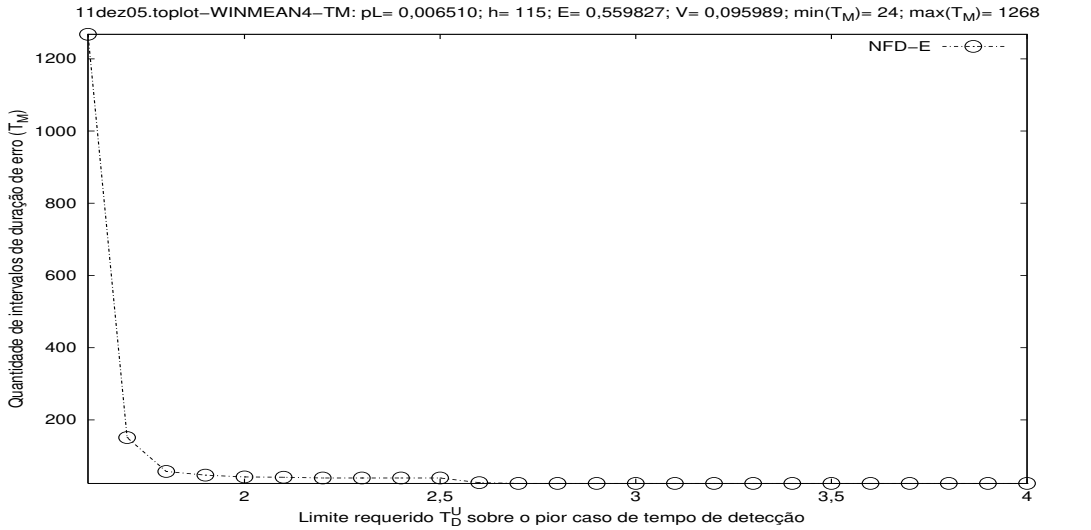


Figura 5.33: T_M é influenciado pela variância e por h.

5.3.5 Discussão Geral sobre a Simulação com Dados Coletados

Na simulação com dados coletados de redes reais, que possuem valores de atraso, variância e de comprimento de rajadas maiores que os dados gerados, o estimador de atraso utilizado interfere bastante nos resultados. No caso do tempo de detecção, apenas os estimadores MEAN e LAST levaram o algoritmo NFD-E a atenderem o tempo de detecção, enquanto todos os estimadores do tipo WINMEAN não foram bem. Isso se deve principalmente ao fato que MEAN terá valores mais estáveis que WINMEAN*, o

que é uma vantagem para o tempo de detecção pois períodos temporários com atrasos acima da média, usarão um valor de média no caso de MEAN, e um valor mais alto no caso de WINMEAN*. No caso de LAST, o uso de somente o mais recente atraso como estimativa leva a uma detecção também rápida.

No caso de intervalos de recorrência de erros, MEAN e LAST também são bons estimadores e levam o NFD-E a se comportar de maneira mais próxima aos valores analíticos dos configuradores Chen-NFD-E e NSM-NFD-E. Os estimadores WINMEAN* com janelas pequenas, como WINMEAN4, são melhores que MEAN e LAST. O configurador NSM-NFD-E gerou valores analíticos que não falharam em dia algum, mas o configurador Chen-NFD-E falhou no dia 04 de dezembro. Essa falha é decorrente justamente da não consideração do comprimento de rajada de perdas no modelo usado pelo configurador Chen-NFD-E.

No caso de intervalos de duração de erros, ao contrário do tempo de detecção e do intervalo de recorrência de erros, MEAN e LAST não são bons estimadores, principalmente para valores pequenos de T_D^U, e levam o NFD-E a se comportar de maneira bem distante dos valores analíticos dos configuradores Chen-NFD-E e NSM-NFD-E. Para os intervalos de duração de erros, WINMEAN4 foi o melhor estimador. Ambos os configuradores falham em relação aos intervalos de duração de erros. Ambos os configuradores geram valores similares quando a probabilidade de perdas é alta ou baixa, e as rajadas de perdas não são muito longas. Mas o configurador NSM-NFD-E é melhor que o configurador Chen-NFD-E quando os comprimentos de rajadas de perdas são longos.

Os resultados confirmam aqueles obtidos por Nunes e Jansch-Pôrto [87] e Falai e Bondavalli [47] sobre a dificuldade em se encontrar estimadores/preditores que sejam bons para levar os detectores de defeitos a atenderem T_D^U, T_M^U e T_{MR}^L ao mesmo tempo.

5.4 Conclusões

Uma primeira conclusão sobre o modelo proposto é que o uso de distribuições de probabilidade dos comprimentos de rajadas de perdas melhora a qualidade de um configurador quando há rajadas de perdas. No caso apresentado neste capítulo, os configuradores propostos são melhores que os de Chen. Uma segunda conclusão é que a escolha de um estimador de atrasos influencia muito a QoS de um algoritmo de detecção em redes reais. Mais ainda, os experimentos indicam que a escolha de um estimador irá melhorar os intervalos de recorrência de erros (uso de WINMEAN4, MEAN ou LAST) e/ou os intervalos de duração de erros (uso de WINMEAN4), *ou* o atendimento de tempo

de detecção (uso de MEAN e LAST). A terceira conclusão é que um modelo alternativo de configurador deve dar mais atenção à variância de atrasos.

Capítulo 6

Melhorando a Precisão

A partir dos resultados obtidos no capítulo 5, este capítulo apresenta uma pequena modificação no estimador do algoritmo NFD-E, através da combinação dos estimadores MEAN e WINMEAN4, em um estimador chamado MEAN-WINMEAN4. Essa modificação permite que uma decisão seja feita em tempo de execução sobre qual estimador dentre aqueles dois deve ser usado. Os resultados da simulação mostram que MEAN-WINMEAN4 melhora a precisão do detector de defeitos quando comparado com WINMEAN4 e MEAN separadamente, em particular com um T_{MR} melhor e um menor número de *transições-S*, ao mesmo tempo que T_M se mantém similar e o pior caso de tempo de detecção se mantém equivalente ao de WINMEAN4.

6.1 Um Método Simples para Melhorar a Precisão

A idéia usada para melhorar a precisão é simples: se a carga atual da rede está abaixo da média, use o valor de MEAN como estimativa do próximo atraso de mensagem, caso contrário, use WINMEAN4. O estimador MEAN é uma boa estimativa quando a carga da rede está abaixo da média, e é uma escolha ruim quando a carga está acima da média. O contrário ocorre com WINMEAN4, que é uma boa estimativa quando a carga da rede está acima da média, e é uma escolha ruim quando a carga está abaixo da média. Portanto, o estimador proposto, que chamamos de MEAN-WINMEAN4, usa o melhor de MEAN e de WINMEAN4 para melhorar a precisão de NFD-E.

O detector de defeitos q considera as 4 mais recentes mensagens *heartbeat*, denotadas por $m'_1, \ldots, m'_4$. Seja $s_1, \ldots, s_4$ os números de seqüência de tais mensagens e $A'_1, \ldots, A'_4$ os seus instantes de recebimento de acordo com o relógio local de q. Então WINMEAN4

é estimado por,

$$WINMEAN4_{l+1} \approx \frac{1}{4}\left(\sum_{i=1}^{4}(A'_i - \eta s_i)\right).$$

O detector de defeitos q também considera as n mensagens *heartbeat* já recebidas, denotadas por $m'_1, \ldots, m'_n$. Seja $s_1, \ldots, s_n$ os números de seqüência de tais mensagens e $A'_1, \ldots, A'_n$ os seus instantes de recebimento de acordo com o relógio local de q. Então MEAN é estimado por:

$$MEAN_{l+1} \approx \frac{1}{n}\left(\sum_{j=1}^{n}(A'_j - \eta s_j)\right).$$

Considere m'_k como a mensagem mais recentemente recebida, s_k o número de seqüência de tal mensagem e A'_k o seu instante de recebimento de acordo com o relógio local de q. EA_{l+1} (estimador MEAN-WINMEAN4 proposto) na linha 10 do algoritmo NFD-U/NFD-E (seção 3.1.10) torna-se então:

$$EA_{l+1} \approx \begin{cases} MEAN_{l+1} + (l+1)\eta, & \text{se } A'_k - \eta s_k \leq MEAN_l; \\ WINMEAN4_{l+1} + (l+1)\eta, & \text{caso contrário.} \end{cases}$$

6.2 Avaliação do Estimador Composto

A configuração da simulação é a mesma descrita na seção 5.3.1, exceto que realizamos simulações somente referentes aos dias usados nas figuras do capítulo 5 (Avaliação da Proposta). E para tornar a avaliação mais simples e direta, as figuras usadas neste capítulo são referentes aos dias 09 de dezembro ($p_L = 0,00072$; $h = 1$), 04 de dezembro ($p_L = 0,007164$; $h = 12$), e 08 de dezembro ($p_L = 0,030864$; $h = 468$). Esta seção apresenta as análises dos intervalos de recorrência de erros e de duração de erros, e do tempo de detecção, e uma discussão geral sobre os resultados.

6.2.1 Análise do Tempo de Detecção

O tempo de detecção de NFD-E com MEAN-WINMEAN4 manteve-se equivalente àquele com WIMMEAN4, mesmo o tempo de deteção máximo, como pode ser observado na Figura 6.1, para o dia 08 de dezembro. Isso se deve ao fato que MEAN-WINMEAN4 usa WINMEAN4 para valores atuais de atrasos maiores que a média. Os demais gráficos

para os dias 04 e 09 de dezembro tiveram comportamentos muito similares e não são mostrados.

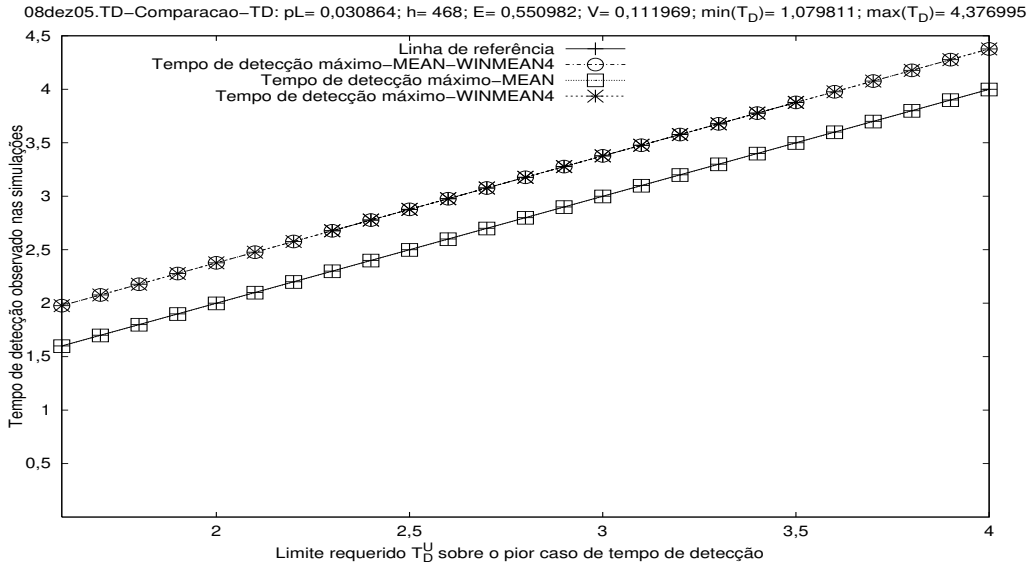


Figura 6.1: O tempo de detecção máximo atendido por NFD-E, para o dia 08 de dezembro ($p_L = 0,030864$ e $h = 468$).

6.2.2 Análise dos Intervalos de Recorrência de Erros

As Figuras 6.2, 6.3 e 6.4 mostram claramente que MEAN-WINMEAN4 melhora bastante o T_{MR} de NFD-E em relação à MEAN e a WINMEAN4. Na Figura 6.2, referente ao dia 09 de dezembro, o estimador MEAN-WINMEAN4 levou o NFD-E a fornecer um T_{MR} melhor que WINMEAN4 quando $T_D^U = [1,6,\ 2,7] \cup [3,6,\ 3,9]$, e um T_{MR} equivalente a WINMEAN4 nos demais intervalos ($T_D^U = [2,8,\ 3,5] \cup 4$). Na Figura 6.3, referente ao dia 04 de dezembro, o estimador MEAN-WINMEAN4 levou o NFD-E a fornecer um T_{MR} melhor que WINMEAN4 quando $T_D^U = [1,6,\ 1,8] \cup [2,3,\ 2,6] \cup [3,2,\ 3,7]$, e um T_{MR} equivalente a WINMEAN4 nos demais intervalos ($T_D^U = [1,9,\ 2,2] \cup [2,7,\ 3,1] \cup [3,8,\ 4,0]$). Na Figura 6.4, referente ao dia 08 de dezembro, o estimador MEAN-WINMEAN4 levou o NFD-E a fornecer um T_{MR} melhor que WINMEAN4 quando $T_D^U = [2,2,\ 2,7] \cup [3,1,\ 3,6]$, e um T_{MR} equivalente a WINMEAN4 nos demais intervalos ($T_D^U = [1,6,\ 2,1] \cup [2,8,\ 3,0] \cup [3,7,\ 4,0]$).

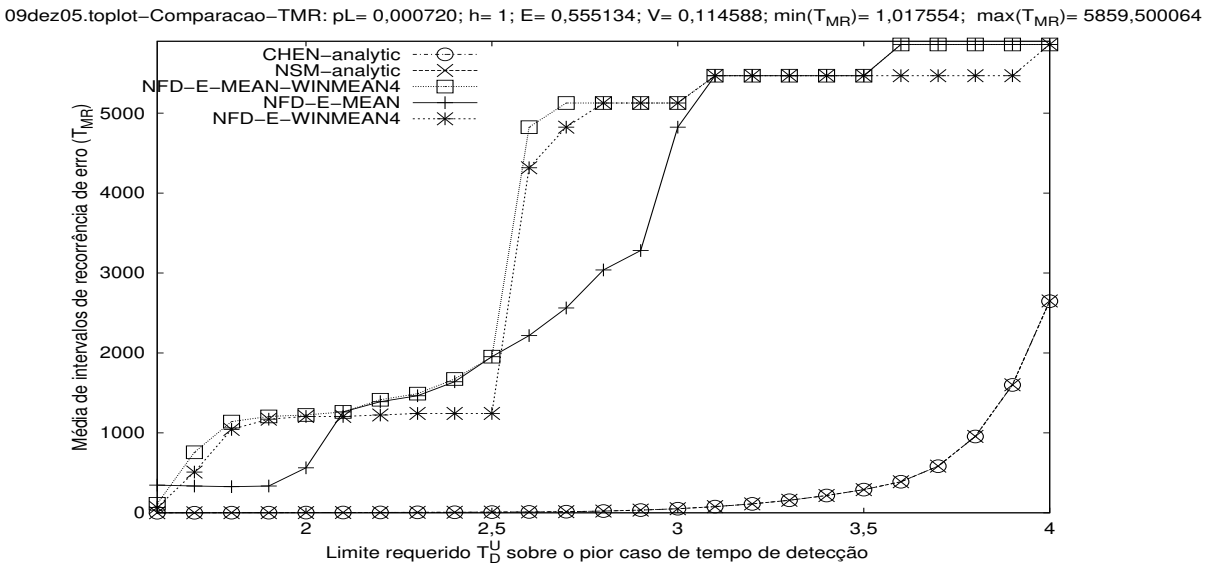


Figura 6.2: T_{MR}^{L} atendido melhor por NFD-E usando MEAN-WINMEAN4, para o dia 09 de dezembro ($p_L = 0,00072$ e $h = 1$).

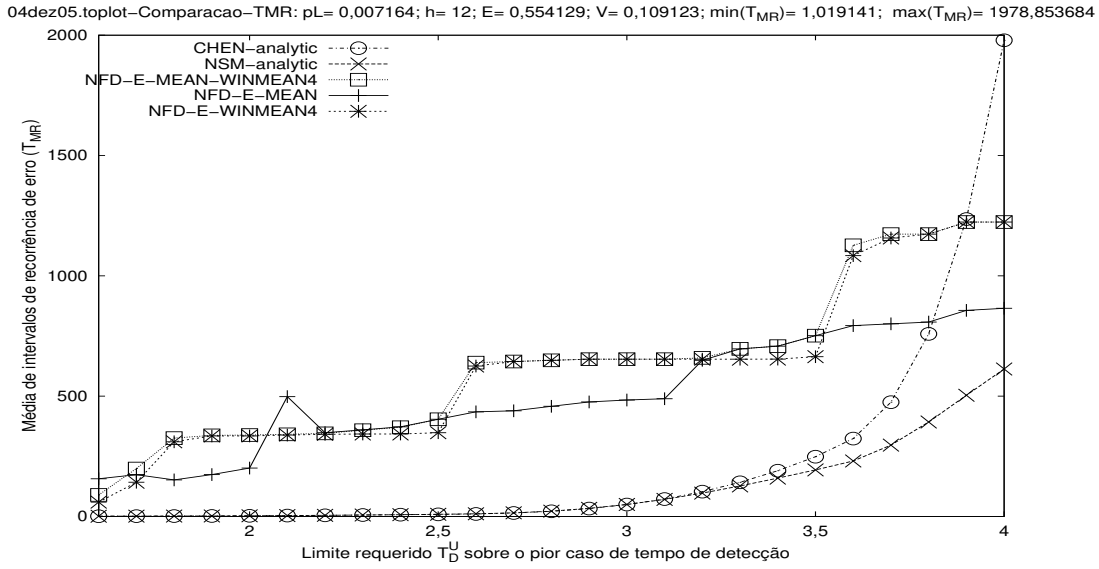


Figura 6.3: T_{MR}^{L} atendido melhor por NFD-E usando MEAN-WINMEAN4, para o dia 04 de dezembro ($p_L = 0,007164$ e $h = 12$).

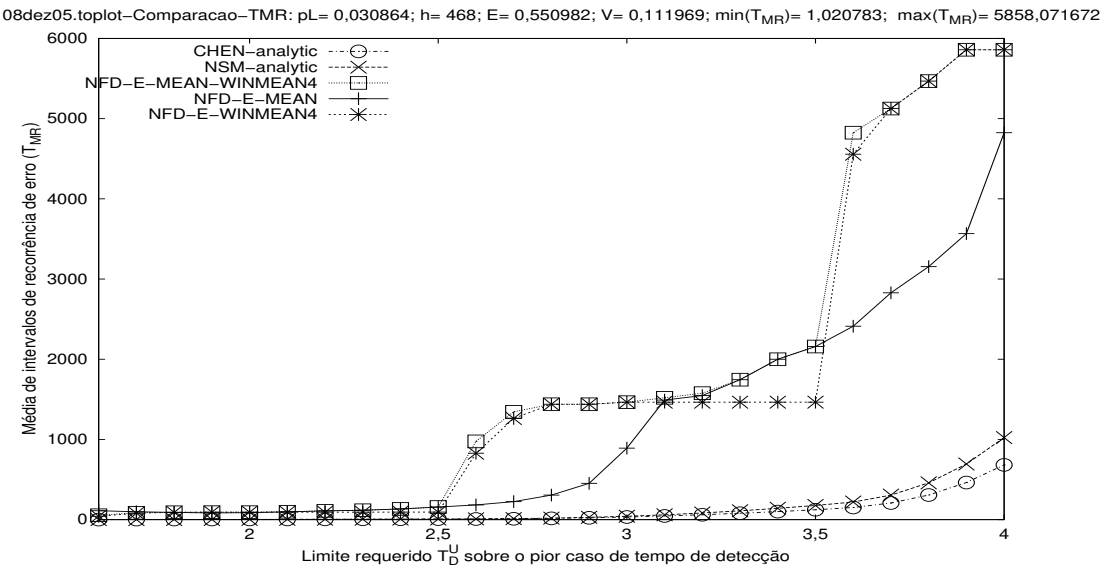


Figura 6.4: T_{MR}^{L} atendido melhor por NFD-E usando MEAN-WINMEAN4, para o dia 08 de dezembro ($p_L = 0,030864$ e $h = 468$).

Há uma observação importante a partir das Figuras 6.2, 6.3 e 6.4: há um valor de T_D^U, nestas Figuras é 2.5, que divide dois comportamentos de NFD-E. Até $T_D^U = 2,5$, à medida que aumentamos o p_L e o h, do dia 09 de dezembro ($p_L = 0,00072$ e $h = 1$) até 08 de dezembro ($p_L = 0,030864$ e $h = 468$), as Figuras mostram claramente a sensibilidade a essas valores, com o achatamento das curvas progressivamente até que haja um platô na Figura 6.4 (dia 08 de dezembro). Após $T_D^U = 2,5$ parece haver tempo de sobra para determinar os atrasos e perdas das mensagens, de forma que T_{MR} aumenta de forma crescente em todos esses dias.

6.2.3 Análise de Intervalos de Duração de Erros

As Figuras 6.5, 6.7 e 6.9 mostram que o uso de MEAN-WINMEAN4 é praticamente equivalente ao uso de WINMEAN4 em relação a T_M. T_M obtido com MEAN-WINMEAN4 é um pouco pior do que o oferecido por WINMEAN4 apenas em poucos casos. Apesar disso, NFD-E com MEAN-WINMEAN4 provoca menos transições-S do que com WINMEAN4, como pode ser visto nas Figuras 6.6, 6.8 e 6.10. Na Figura 6.5, referente ao dia 09 de dezembro, o estimador MEAN-WINMEAN4 levou o NFD-E a fornecer um T_M equivalente ao fornecido por WINMEAN4, exceto no intervalo $T_D^U = [2,2,\ 2,7]$, em que WINMEAN4 foi um pouco melhor. Na Figura 6.6, referente ao dia 09 de dezembro, o estimador MEAN-WINMEAN4 levou o NFD-E a fornecer uma menor quantidade de

transições-S, que corresponde a uma menor quantidade de intervalos de T_M, do que WINMEAN4, principalmente no intervalo $T_D^U = [1,6,\ 1,8] \cup [2,1,\ 2,6]$.

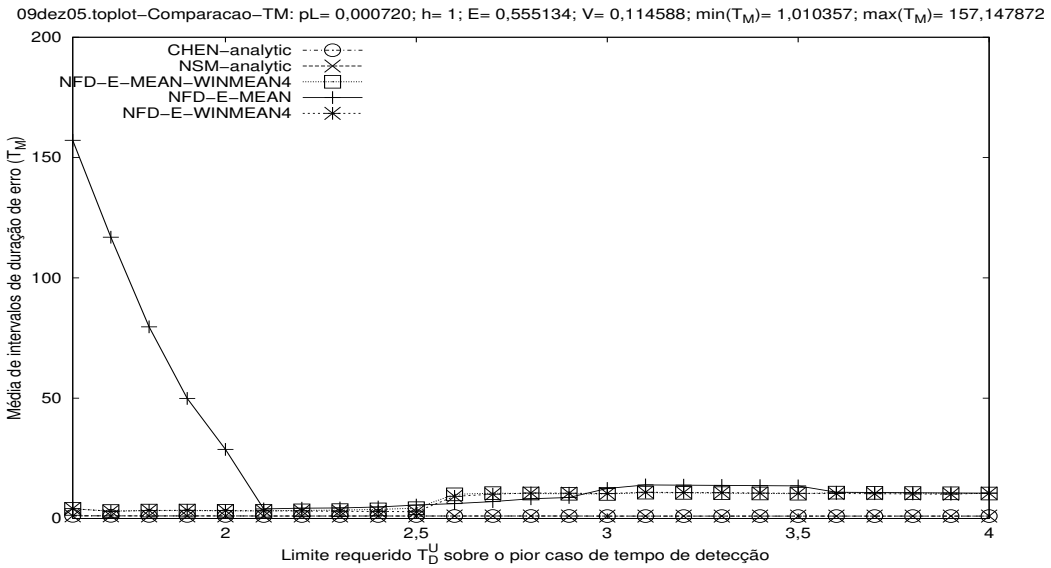


Figura 6.5: T_M fornecido por NFD-E é similar com WINMEAN4 e MEAN-WINMEAN4, para o dia 09 de dezembro ($p_L = 0,00072$ e $h = 1$).

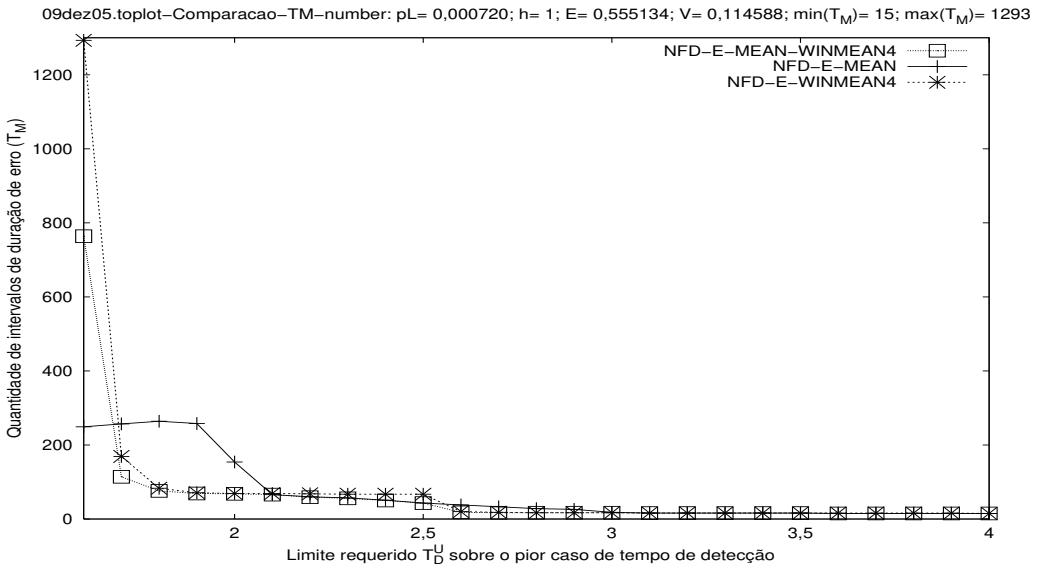


Figura 6.6: T_M fornecido por NFD-E é melhor com MEAN-WINMEAN4, para o dia 09 de dezembro ($p_L = 0,00072$ e $h = 1$).

Na Figura 6.7, referente ao dia 04 de dezembro, o estimador MEAN-WINMEAN4 levou o NFD-E a fornecer um T_M equivalente ao fornecido por WINMEAN4 no intervalo $T_D^U = [1,7,\ 4]$, sendo que MEAN-WINMEAN4 foi um pouco melhor que WINMEAN4

quando $T_D^U = 1,6$. Na Figura 6.8, referente ao dia 04 de dezembro, o estimador MEAN-WINMEAN4 levou o NFD-E a fornecer uma menor quantidade de transições-S do que WINMEAN4, principalmente no intervalo $T_D^U = [1,6,\ 1,8] \cup [2,3,\ 2,5] \cup [3,3,\ 3,6]$.

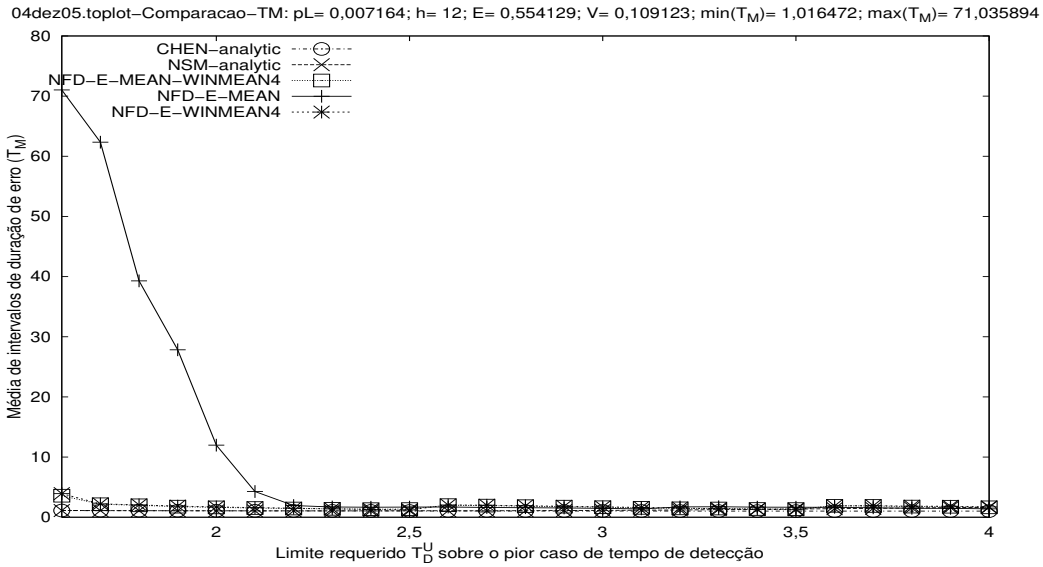


Figura 6.7: T_M fornecido por NFD-E é similar com WINMEAN4 e MEAN-WINMEAN4, para o dia 04 de dezembro ($p_L = 0,007164$ e $h = 12$).

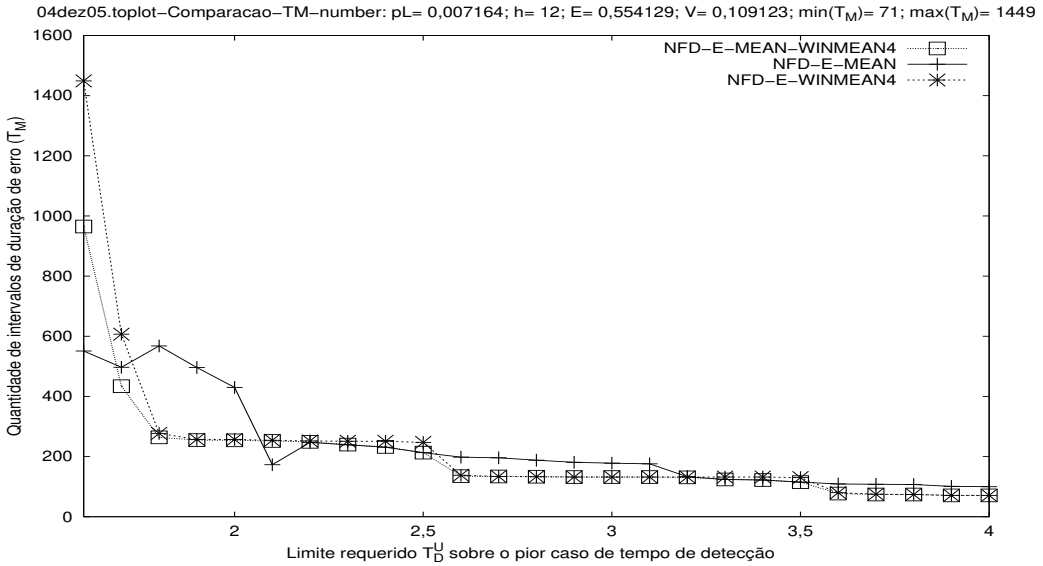


Figura 6.8: T_M fornecido por NFD-E é melhor com MEAN-WINMEAN4, para o dia 04 de dezembro ($p_L = 0,007164$ e $h = 12$).

Na Figura 6.9, referente ao dia 08 de dezembro, o estimador MEAN-WINMEAN4 levou o NFD-E a fornecer um T_M equivalente ao fornecido por WINMEAN4 no intervalo $T_D^U =$

$[1,7,\ 2,1]\cup[2,8,\ 3]\cup[3,7,\ 4]$, um pouco melhor que WINMEAN4 quando $T_D^U = 1,6$, e pior que WINMEAN4 quando $T_D^U = [2,2,\ 2,7] \cup [3,1,\ 3,6]$. Na Figura 6.10, referente ao dia 08 de dezembro, o estimador MEAN-WINMEAN4 levou o NFD-E a fornecer uma menor quantidade de transições-S, principalmente no intervalo $T_D^U = [1,6,\ 1,8] \cup [2,1,\ 2,7] \cup [3,3,\ 3,5]$.

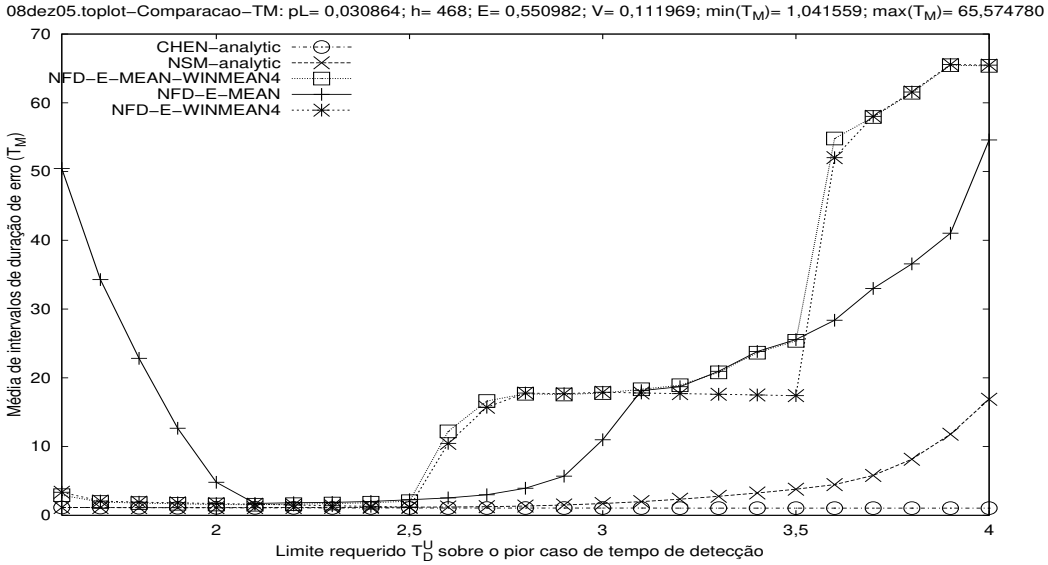


Figura 6.9: T_M fornecido por NFD-E é similar com WINMEAN4 e MEAN-WINMEAN4, para o dia 08 de dezembro ($p_L = 0,030864$ e $h = 468$).

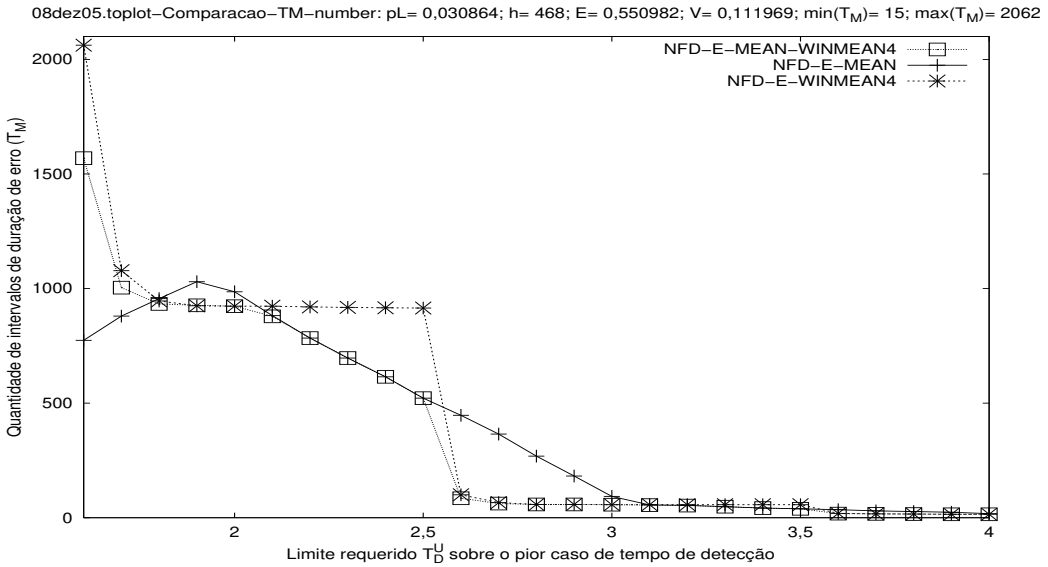


Figura 6.10: T_M fornecido por NFD-E é melhor com MEAN-WINMEAN4, para o dia 08 de dezembro ($p_L = 0,030864$ e $h = 468$).

A partir das Figuras 6.5, 6.7 e 6.9 percebe-se, de forma similar ao ocorrido com T_{MR}, que há um valor de T_D^U, nestas Figuras é 2,5, que divide dois comportamentos de NFD-E. Até $T_D^U = 2,5$, NFD-E apresenta um comportamento semelhante e mais próximo aos valores analíticos, independente se aumentamos p_L e h. Após $T_D^U = 2,5$, NFD-E sofre com variância de atrasos alta ($V = 0,114598$) no dia 09 de dezembro e probabilidade de perdas baixa e sem rajadas de perdas ($p_L = 0,00072$ e $h = 1$), e sofre com variância alta ($V = 0,111969$) no dia 08 de dezembro e probabilidade de perdas alta e rajadas longas de perdas ($p_L = 0,030864$ e $h = 468$).

6.2.4 Discussão Geral sobre MEAN-WINMEAN4

O uso de MEAN-WINMEAN4 em NFD-E manteve o tempo de detecção máximo equivalente ao de WINMEAN4, ao mesmo tempo em que melhorou consideravelmente a precisão. No caso de intervalos de recorrência de erros, MEAN-WINMEAN4 leva a valores maiores do que WINMEAN4. No caso de intervalos de duração de erros, MEAN-WINMEAN4 leva NFD-E a fornecer valores equivalentes ao de WINMEAN4. Apesar do desempenho não muito bom de T_M no dia 08 de dezembro ($p_L = 0,030864$ e $h = 468$), nos demais dias o desempenho de T_M foi bom. Além disso, no dia 08 de dezembro, a Figura 6.10 mostra claramente que para $T_D^U > 2,5$, a quantidade de intervalos de duração de erro fica menor que $h = 468$, o que prejudicou o cálculo do valor esperado de T_M. Finalmente, o uso de MEAN-WINMEAN4 diminui o número de transições-S de NFD-E, como atestam as Figuras sobre a quantidade de intervalos de duração de erros ocorridos. Contudo, MEAN-WINMEAN4 agiu bem como um estimador de atrasos, visto que para os dias 09 de dezembro ($p_L = 0,00072$; $h = 1$) e 04 de dezembro ($p_L = 0,007164$; $h = 12$), em que a probabilidade de perdas foi baixa e o comprimento de rajadas foi pequeno, MEAN-WINMEAN4 agiu bem também em relação a T_M. É importante ressaltar que pesquisa adicional deve ser realizada para verificar o valor mais adequado de T_D^U, de acordo com as condições da rede, como pôde ser observado nas discussões sobre T_M e T_{MR}, em que valores de T_D^U até 2,5 apresentaram melhores resultados.

6.3 Conclusões

Este capítulo mostrou que é possível melhorar a precisão de detectores de defeitos através da combinação de estimadores simples. Contudo, ainda é preciso um estudo mais detalhado para uma conclusão definitiva, particularmente para desenvolver estimadores

que tratam melhor as rajadas muito longas de perdas, que afetam bastante a duração de erros (T_M). Esta abordagem é diferente dos trabalhos de Nunes e Jansch-Pôrto [87] e Falai e Bondavalli [47], que usam apenas uma combinação estimador/preditor mais margens de segurança em cada execução dos detectores de defeitos. Os resultados de simulação indicam que modelos alternativos para rajadas de perdas e de variância de atrasos com algum processo de decisão podem melhorar ainda mais a QoS de detectores de defeitos.

Capítulo 7

Conclusão

Detectores de defeitos não confiáveis definidos por Chandra e Toueg [27], baseados em propriedades (independentes de implementação) de completude e precisão, têm sido usados para contornar o resultado de impossibilidade FLP [50], ao estender o modelo de sistema distribuído assíncrono com detectores de defeitos não confiáveis sujeito a pelo menos uma quebra de processo e canais de comunicação confiáveis. Esses detectores de defeitos são usados para resolver vários problemas em sistemas distribuídos assíncronos. O trabalho pioneiro de Chen et al [30] sobre QoS de detectores de defeitos definiu métricas para precisão e tempo de detecção e apresentou configuradores que não usam a informação sobre rajadas de perdas de mensagens. Estas métricas são adequadas para avaliar implementações de detectores de defeitos.

Esta tese abordou o problema da qualidade de serviço (QoS) de detectores de defeitos sob a presença de rajadas de perdas de mensagens. Esse problema é de grande interesse em sistemas distribuídos reais porque muitos são construídos sobre redes de longa distância e sobre redes sem fio, em que a ocorrência de rajadas de perdas de mensagens é comum. A tese propôs um modelo baseado em cadeias de Markov para capturar a informação sobre rajadas de perdas. Os resultados de simulação mostram que o modelo proposto leva a configuradores mais adequados do que os de Chen et al quando há rajadas longas de perdas. A partir da avaliação da proposta foi apresentado um estimador composto que melhora a precisão de detectores de defeitos, principalmente aumentando o intervalo de recorrência de erros e diminuindo a quantidade de *transições-S*, ao mesmo tempo que mantém o tempo de detecção máximo e o intervalo de duração de erro dentro de limites razoáveis. É importante destacar que a proposta apresentada melhora a precisão de detectores de defeitos não confiáveis, principalmente em um cenário com freqüência alta de

rajadas de perdas de *heartbeats*, o que pode levar consenso distribuído a ser concluído mais rapidamente. Este capítulo apresenta as contribuições da tese, as dificuldades encontradas no seu desenvolvimento, e sugere alguns trabalhos futuros.

7.1 Contribuições da Tese

As quatro principais contribuições deste trabalho, sobre qualidade de serviço de detectores de defeitos na presença de rajadas de perdas, incluem algumas respostas ao problema mencionado na seção 1.2:

1) Os algoritmos de detectores de defeitos de Chen et al [30], NFD-S, NFD-U e NFD-E são robustos o suficiente para serem utilizados mesmo quando há probabilidade alta de ocorrência de rajadas longas de perdas de mensagens *heartbeat*.

2) Os configuradores propostos e apresentados nesta pesquisa são adequados para a configuração dos detectores de defeitos mesmo quando rajadas longas de perdas ocorrem com alta freqüência. Por exemplo, quando a distribuição de probabilidade das rajadas de perdas é similar a Pareto. Os configuradores propostos também comportam-se bem quando a distribuição é geométrica. Além disso, quando não há rajadas ($h = 1$), o comportamento dos configuradores propostos são similares ao de Chen et al.

3) Os experimentos confirmam resultados anteriores de que é difícil obter estimadores/preditores de atrasos que levem os detectores de defeitos a atenderem T_D^U, T_M^U e T_{MR}^L ao mesmo tempo.

4) A partir de experimentos adicionais verificou-se que é possível, em vários casos, melhorar a precisão (T_M e T_{MR}) ao mesmo tempo em que diminui-se a ocorrência de *transições-S*, através da combinação de estimadores simples. Aqueles experimentos adicionais indicam que uma abordagem adaptativa com algum processo de decisão pode melhorar significativamente a precisão de detectores de defeitos.

As três contribuições secundárias deste trabalho são:

1) O arcabouço teórico do trabalho de Chen et al possui ampla modularidade, comprovada pela extensão modular do trabalho deles com a abordagem com cadeia de Markov.

2) Uma revisão bibliográfica extensa sobre detectores de defeitos, em particular daqueles que empregam o trabalho de Chen et al.

3) Um protótipo de um simulador que pode ser estendido com outras distribuições de probabilidade de atrasos e de rajadas de perdas e outros algoritmos para detectores de defeitos.

7.2 Dificuldades Encontradas

A primeira dificuldade foi a mudança do enfoque inicial de middleware tolerante a falhas para um estudo mais formal de um dos seus componentes, no caso, detectores de defeitos. O estudo inicial levou à implementação de dois detectores de defeitos adaptativos [98, 99] com `C++` e `CORBA`, supondo canais de comunicação confiável. Resolvemos então tirar a suposição de canais de comunicação confiável, para que os detectores de defeitos não dependessem de uma camada que tratasse das perdas de mensagens. Desta forma, após ler o trabalho de Chen et al sobre qualidade de serviço de detectores de defeitos, resolvemos estendê-lo para que considerasse informações de rajadas de perdas, que ocorrem com bastante freqüência em muitos enlaces de longa distância e em redes sem fio. A segunda e mais severa dificuldade foi a utilização da desigualdade unilateral no nosso modelo. Após muito esforço preferimos utilizá-la diretamente para o configurador para NFD-U e supor uma heurística.

7.3 Trabalhos Futuros

Alguns trabalhos futuros a serem desenvolvidos são:

1) Desenvolver um modelo que dê mais atenção à variância das mensagens e à pouca quantidade de intervalos de T_M em relação a h, para que T_M^U seja atendido de forma mais satisfatória.

2) Melhorar o ambiente de simulação: poderiam ser implementados algoritmos para o modelo *pull* e adicionadas mais métricas a serem comparadas.

3) Definir um modelo de adaptação dos detectores: períodos diferentes do dia possuem comportamento diferente de atrasos e perdas de mensagens. Seria interessante definir um modelo de adaptação que realizasse a adaptação do detector de forma dinâmica

e automática de forma a otimizar ao máximo a QoS do detector de acordo com as condições atuais da rede.

4) Aproveitar na prática, e com maior confiança, os vários resultados existentes sobre detectores de defeitos baseados em Chandra e Toueg [27]: encontrar uma relação teórico-prática em que, dado um detector de defeitos q com QoS e um certo período de tempo Δ, qual seria a probabilidade de que q iria agir dentro de cada uma das classes de detectores de defeitos de Chandra e Toueg durante Δ. Desta forma, um serviço ou aplicação poderia esperar um comportamento mais previsível de serviços baseados em detectores de defeitos. Modelos formais como *I/O automata* [77] e suas variantes (por exemplo, *timed I/O automata* [66]), poderiam ser usados para desenvolver essa relação.

5) Definir uma arquitetura para prover detectores de defeitos com QoS: tal arquitetura permitiria que um detector de defeitos fosse capaz de atender requisitos de QoS para processos diferentes. O trabalho de Hayashibara et al [60, 37] é um ponto de partida para desenvolver a arquitetura.

Referências Bibliográficas

[1] M. K. Aguilera, W. Chen, e S. Toueg. Heartbeat: a timeout-free failure detector for quiescent reliable communication. Em *Proceedings of the 11th International Workshop on Distributed Algorithms, 1997*, volume LNCS, pp. 126–140, 1997.

[2] M. K. Aguilera, W. Chen, e S. Toueg. Failure detection and randomization: a hybrid approach to solve consensus. *SIAM Journal of Computing*, 28(3):890–903, 1998.

[3] M. K. Aguilera, W. Chen, e S. Toueg. Using the heartbeat failure detector for quiescent reliable communication and consensus in partitionable networks. *Theoretical Computer Science*, 220(1):3–30, Junho de 1999.

[4] M. K. Aguilera, W. Chen, e S. Toueg. Failure detection and consensus in the crash-recovery model. *Distributed Computing*, 13(2):99–125, Abril de 2000.

[5] M. K. Aguilera, W. Chen, e S. Toueg. On quiescent reliable communication. *SIAM Journal of Computing*, 29(6):2040–2073, Abril de 2000.

[6] M. Allman e V. Paxson. On estimating end-to-end network path properties. Em *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 263–274, 1999.

[7] C. Almeida e P. Veríssimo. Timing failure detection and real-time group communication in quasi-synchronous systems. Em *Proceedings of the Eighth Euromicro Workshop on Real-Time Systems (EURWRTS'96)*, pp. 230–235, L'Aquila, Itália, Junho de 1996.

[8] C. Almeida e P. Veríssimo. Using light-weight groups to handle timing failures in quasi-synchronous systems. Em *Proceedings of 19th IEEE Real-Time Systems Symposium*, pp. 430–439, Madrid, Espanha, Dezembro de 1998.

[9] Y. Amir, L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, e P. Ciarfella. The totem single-ring ordering and membership protocol. *ACM Transactions on Computer Systems*, 13(4):311–342, Novembro de 1995.

[10] E. Anceaume, A. Fernández, A. Mostefaoui, G. Neiger, e M. Raynal. A necessary and sufficient condition for transforming limited accuracy failure detectors. *Journal of Computer and System Sciences*, 68(1):123–133, Fevereiro de 2004.

[11] J. Andrén, M. Hilding, e D. Veitch. Understanding end-to-end internet traffic dynamics. Em *Proceedings of IEEE 1998 Global Communications Conference (GLOBECOM 98)*, volume 2, pp. 1118–1122, Sidney, Austrália, Novembro de 1998.

[12] F. Anjum e L. Tassiulas. Comparative study of various tcp versions over a wireless link with correlated losses. *IEEE/ACM Transactions on Networking*, 11(3):370–383, Junho de 2003.

[13] M. Aron e P. Druschel. Soft timers: efficient microsecond software timer support for network processing. *ACM Transactions on Computer Systems*, 18(3):197–228, Agosto de 2000.

[14] J. Aspnes. Randomized protocols for asynchronous consensus. *Distributed Computing*, 16(2–3):165–175, Setembro de 2003.

[15] D. Avresky e N. Natchev. Dynamic reconfiguration in computer clusters with irregular topologies in the presence of multiple node and link failures. *IEEE Transactions on Computers*, 54(5):603–615, Maio de 2005.

[16] A. Azivienis, J.-C. Laprie, B. Randell, e C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, Janeiro-Março de 2004.

[17] Ö. Babaoglu, R. Davoli, e A. Montresor. Group communication in partitionable systems: specification and algorithms. *IEEE Transactions on Software Engineering*, 27(4):308–336, Abril de 2001.

[18] A. Bartoli. Implementing a replicated service with group communication. *Journal of Systems Architecture*, 50(8):493–519, Agosto de 2004.

[19] M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols. Em *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pp. 27–30, Agosto de 1983.

[20] M. Bertier, O. Marin, e P. Sens. Implementation and performance evaluation of an adaptable failure detector. Em *Proceedings of the 2002 International Conference on Dependable Systems and Networks (DSN'02)*, pp. 354–363, Junho de 2002.

[21] M. Bertier, O. Marin, e P. Sens. Performance analysis of a hierarchical failure detector. Em *Proceedings of the 2003 International Conference on Dependable Systems and Networks (DSN'03)*, pp. 635–644, Junho de 2003.

[22] K. P. Birman e T. A. Joseph. Reliable communication in the presence of failures. *ACM Transactions on Computer Systems*, 5(1):47–76, Fevereiro de 1987.

[23] J. Bolot. End-to-end packet delay and loss behavior in the internet. Em *Proceedings of Symposium on Communications Architectures and Protocols (SIGCOMM'93)*, pp. 289–298, San Francisco, Estados Unidos, Setembro de 1993.

[24] A. Casimiro e P. Veríssimo. Using the timely computing base for dependable qos adaptation. Em *Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems (SRDS 2001)*, pp. 208–217, New Orleans, Estados Unidos, Outubro de 2001.

[25] A. Casimiro e P. Veríssimo. Generic timing fault tolerance using a timely computing base. Em *Proceedings of the 2002 International Conference on Dependable Systems and Networks (DSN'02)*, pp. 27–36, Junho de 2002.

[26] T. D. Chandra, V. Hadzilacos, e Sam Toueg. The weakest failure detector for solving consensus. *Journal of the Association for Computing Machinery*, 43(4):685–722, Julho de 1996.

[27] T. D. Chandra e Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the Association for Computing Machinery*, 43(2):225–267, Março de 1996.

[28] B. Charron-Bost. Comparing the atomic commitment and consensus problems. Em *Future Directions in Distributed Computing: Research and Position Papers. Editors:*

A. Schiper, A.A. Shvartsman, H. Weatherspoon, B.Y. Zhao (Eds.): LNCS 2003, pp. 29–34, 2003.

[29] W. Chen. *On the quality of service of failure detectors*. Tese de Doutorado, Cornell University, Cornell, Maio de 2000.

[30] W. Chen, S. Toueg, e M. K. Aguilera. On the quality of service of failure detectors. *IEEE Transactions on Computers*, 51(5):561–580, Maio de 2002.

[31] G. V. Chockler, I. Keidar, e R. Vitenberg. Group communication specifications: a comprehensive survey. *ACM Computing Surveys*, 33(4):427–469, Dezembro de 2001.

[32] F. Chu. Reducing ω to $\diamond w$. *Information Processing Letters*, 67(6):289–293, Setembro de 1998.

[33] N. Cook, S. Shrivastava, e S. Wheater. Distributed object middleware to support dependable information sharing between organisations. Em *Proceedings of the 2002 International Conference on Dependable Systems and Networks (DSN'02)*, pp. 249–258, Junho de 2002.

[34] G. Coulouris, J. Dollimore, e T. Kindberg. *Distributed systems: concepts and design*. Addison-Wesley, Essex, Inglaterra, quarta edição, 2005.

[35] F. Cristian e C. Fetzer. The timed asynchronous distributed system model. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):642–657, Junho de 1999.

[36] M. Cukier, J. Ren, C. Sabnis, D. Henke, J. Pistole, W. H. Sanders, D. E. Bakken, M. E. Berman, D. A. Karr, e R. E. Schantz. Aqua: an adaptive architecture that provides dependable distributed objects. Em *Proceedings of Seventeenth IEEE Symposium on Reliable Distributed Systems (SRDS 1998)*, pp. 245–253, Outubro de 1998.

[37] X. Défago, P. Urbán, N. Hayashibara, e T. Katayama. Definition and specification of accrual failure detectors. Em *Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, pp. 206–215, Junho-Julho de 2005.

[38] C. Delport-Gallet, H. Fauconnier, e R. Guerraoui. A realistic look at failure detectors. Em *Proceedings of the 2002 International Conference on Dependable Systems and Networks (DSN'02)*, pp. 345–353, Junho de 2002.

[39] C. Delport-Gallet, H. Fauconnier, R. Guerraoui, V. Hadzilacos, P. Kouznetsov, e S. Toueg. The weakest failure detectors to solve certain fundamental problems in distributed computing. Em *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing(PODC'04)*, pp. 338–346, Julho de 2004.

[40] D. Dolev, C. Dwork, e L. Stockmeyer. On the minimal synchronism needed for distributed consensus. *Journal of the Association for Computing Machinery*, 34(1):77–97, Janeiro de 1987.

[41] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, e W. E. Weihl. Reaching approximate agreement in the presence of faults. *Journal of the Association for Computing Machinery*, 33(3):499–516, Maio de 1986.

[42] A. Doudou, B. Garbinato, e R. Guerraoui. Encapsulating failure detection: from crash to byzantine failures. Em *Ada-Europe 2002. LNCS 2361*, pp. 24–50, 2002.

[43] A. Doudou, B. Garbinato, R. Guerraoui, e A. Schiper. Muteness failure detectors: specification and implementation. Em *EDCC-3'99. LNCS 1667*, pp. 71–87, 1999.

[44] C. Dwork, N. Lynch, e L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the Association for Computing Machinery*, 35(2):288–323, Abril de 1988.

[45] X. Défago e A. Schiper. Semi-passive replication and lazy consensus. *Journal of Parallel and Distributed Computing*, 64(12):1380–1398, Dezembro de 2004.

[46] E. N. Elnozahy, L. Alvisi, Y.-M. Wang, e D. B. Johnson. A survey of rollback-recovery protocols in message passing systems. *ACM Computing Surveys*, 34(3):375–408, Setembro de 2002.

[47] L. Falai e A. Bondavalli. Experimental evaluation of the qos of failure detectors on wide area network. Em *Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, pp. 624–633, Junho-Julho de 2005.

[48] P. Felber e P. Narasimhan. Experiences, strategies, and challenges in building fault-tolerant corba systems. *IEEE Transactions on Computers*, 53(5):497–511, Maio de 2004.

[49] C. Fetzer. Perfect failure detection in timed asyncronous systems. *IEEE Transactions on Computers*, 52(2):99–112, Fevereiro de 2003.

[50] M. J. Fischer, N. A. Lynch, e M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the Association for Computing Machinery*, 32(2):374–382, Abril de 1985.

[51] S. Floyd. A report on recent developments in tcp congestion control. *IEEE Communications Magazine*, 39(4):84–90, Abril de 2001.

[52] S. Floyd e V. Paxson. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, Agosto de 2001.

[53] R. Friedman, A. Mostefaoui, e M. Raynal. Asynchronous bounded lifetime failure detectors. *Information Processing Letters*, 94(2):85–91, Abril de 2005.

[54] F. C. Gärtner. Fundamentals of fault-tolerant distributed computing in asynchronous environments. *ACM Computing Surveys*, 31(1):1–26, Março de 1999.

[55] A.S. Grimshaw e A. Natrajan. Legion: lessons learned building a grid operating system. *Proceedings of the IEEE*, 93(3):589–603, Março de 2005.

[56] R. Guerraoui. Non-blocking atomic commit in asynchronous distributed systems with failure detectors. *Distributed Computing*, 15(1):17–25, Janeiro de 2002.

[57] R. Guerraoui, M. Hurfin, A. Mostefaoui, R. Oliveira, M. Raynal, e A. Schiper. Consensus in asynchronous distributed systems: a concise guide tour. *Distributed Systems: Editors: S. Krakowiak and S. K. Shrivastava, LNCS 1752*, 1752:33–47, 2000.

[58] R. Guerraoui e M. Raynal. The information structure of indulgent consensus. *IEEE Transactions on Computers*, 53(4):453–466, Abril de 2004.

[59] R. Guerraoui e A. Schiper. Software-based replication for fault tolerance. *IEEE Computer*, 30(4):68–74, Abril de 1997.

[60] N. Hayashibara, X. Défago, R. Yared, e T. Katayama. The φ accrual failure detector. Em *Proceedings of IEEE 23rd Symposium on Reliable Distributed Systems (SRDS 2004)*, Outubro de 2004.

[61] M. Hurfin, A. Mostefaoui, e M. Raynal. Consensus in asynchronous systems where processes can crash and recover. Em *Proceedings of Seventeenth IEEE Symposium on Reliable Distributed Systems (SRDS 1998)*, pp. 280–286, Outubro de 1998.

[62] V. Jacobson. Congestion avoidance and control. Em *Proceedings of ACM Sugcomm '88*, pp. 314–329, 1988.

[63] E. Jiménez, S. Arévalo, e A. Fernández. Implementing unreliable failure detectors with unknown membership. *Information Processing Letters*.

[64] Z. Kalbarczyk, R. K. Iyer, e W. Long. Application fault tolerance with armor middleware. *IEEE Internet Computing*, 9(2):28–37, Março-Abril de 2005.

[65] P. Karn e C. Partridge. Improving round-trip time estimates in reliable transport protocols. *ACM Transactions on Computer Systems*, 9(4):364–373, Novembro de 1991.

[66] D. K. Kaynar, N. Lynch, R. Segala, e F. Vaandrager. The theory of timed i/o automata (versão obtida do arquivo http://theory.lcs.mit.edu/tds/papers/kirli/mainfinal.pdf). Relatório técnico, Massachusetts Institute of Technology, Novembro de 2005.

[67] G. Kola, T. Kosar, e M. Livny. Phoenix: making data-intensive grid applications fault-tolerant. Em *Proceedings of Fifth IEEE/ACM International Workshop on Grid Computing*, pp. 251–258, Novembro de 2004.

[68] A. Konrad, B. Y. Zhao, A. D. Joseph, e R. Ludwig. A Markov-channel model algorithm for wireless networks. *Wireless Networks*, 9(3):189–199, Maio de 2003.

[69] L. Lamport. A simple approach to specifying concurrent systems. *Communications of the ACM*, 32(1):32–45, Janeiro de 1989.

[70] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, Maio de 1998.

[71] M. Larrea. On the weakest failure detector for hard agreement problems. *Journal of Systems Architecture*, 49(7–9):345–353, Outubro de 2003.

[72] M. Larrea, A. Fernandéz, e S. Arévalo. On the implementation of unreliable failure detectors in partially synchronous systems. *IEEE Transactions on Computers*, 53(7):815–828, Julho de 2004.

[73] M. Larrea, A. Fernández, e S. Arévalo. Eventually consistent failure detectors. *Journal of Parallel and Distributed Computing*, 65(3):361–373, March de 2005.

[74] H. Lee e P. K. Varshney. Gap-based modeling of packet losses over the internet. Em *Proceedings of 10th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS 2002)*, pp. 507–510, Julho de 2002.

[75] X. Liu, C. Kreitz, R. van Renesse, J. Hickey, M. Hayden, K. Birman, e R. Constable. Building reliable, high-performance communication systems from components. Em *Proceedings of the seventeenth ACM symposium on Operating systems*, pp. 80–92, 1999.

[76] L. C. Lung, J. Fraga, J.-M. Farines, e J. R. S. de Oliveira. Experiências com comunicação de grupo nas especificações fault tolerant corba. Em *Anais do XVIII Simpósio Brasileiro de Redes de Computadores - SBRC'2000*, pp. 85–100, Maio de 2000.

[77] N. A. Lynch. *Distributed algorithms*. Morgan Kauffman Publishers, San Francisco, Estados Unidos, 1996.

[78] S. Mishra, C. Fetzer, e F. Cristian. The timewheel group communication system. *IEEE Transactions on Computers*, 51(8):883–899, Agosto de 2002.

[79] A. Mostefaoui, E. Mourgaya, e M. Raynal. An introduction to oracles for asynchronous distributed systems. *Future Generation Computer Systems*, 18(6):757–767, Maio de 2002.

[80] A. Mostefaoui, S. Rajsbaum, M. Raynal, e C. Travers. From failure detectors with limited scope accuracy to system-wide leadership. Em *Proceedings of 20th International Conference on Advanced Information Networking and Applications (AINA 2006)*, volume 1, pp. 81–86, Viena, Aústria, Abril de 2006.

[81] A. Mostefaoui e M. Raynal. Unreliable failure detectors with limited scope accuracy and an application to consensus. Em *Proceedings of 19th Conference on Foundations of Software Technology and Theoretical Computer Science (FST and TCS 99), LNCS 1738*, pp. 329–341, Chennai, Índia, Dezembro de 1999.

[82] A. Mostefaoui, M. Raynal, e F. Tronel. The best of both worlds: A hybrid approach to solve consensus. Em *Proceedings of the 2000 International Conference on Dependable Systems and Networks (DSN'00)*, pp. 513–522, Junho de 2000.

[83] B. Natarajan, A. Gokhale, S. Yajnik, e D. C. Schmidt. Doors: towards high-performance fault tolerant corba. Em *Proceedings of International Symposium on Distributed Objects and Applications (DOA'00)*, pp. 39–48, Setembro de 2000.

[84] N. F. Neves, M. Correia, e P. Veríssimo. Solving vector consensus with a wormhole. *IEEE Transactions on Parallel and Distributed Systems*, 16(12), Dezembro de 2005.

[85] R. C. Nunes e I. Jansch-Pôrto. Modeling communication delays in distributed systems using time series. Em *Proceedings of IEEE 21st Symposium on Reliable Distributed Systems (SRDS 2002)*, Suita, Japão, Outubro de 2002.

[86] R. C. Nunes e I. Jansch-Pôrto. Non-stationary communication delays in failure detectors. Em *Proceedings of Third IEEE Latin-American Test Workshop (LATW 2002)*, Montivideo, Argentina, Fevereiro de 2002.

[87] R. C. Nunes e I. Jansch-Pôrto. Qos of timeout-based self-tuned failure detectors: the effects of the communication delay predictor and the safety margin. Em *Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN'04)*, pp. 753–761, Junho de 2004.

[88] V. Paxson. End-to-end internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3), Junho de 1999.

[89] F. Pedone, R. Guerraoui, e A. Schiper. The database state machine approach. *Distributed and Parallel Databases*, 14(1), Julho de 2003.

[90] M. Raynal. Quiescent uniform reliable broadcast as an introduction to failure detector oracles. Em *Proceedings of the 6th International Conference on Parallel Computing Technologies (PaCT 2001), LNCS 2127*, pp. 98–111, Setembro de 2001.

[91] M. Raynal. A short introduction to failure detectors for asynchronous distributed systems. *ACM SIGACT News*, 36(1), Março de 2005.

[92] H. Sanneck. *Packet loss recovery and control for voice transmission over the Internet*. Tese de Doutorado, Technischen Universität Berlin, Berlin, Outubro de 2000.

[93] P. Sarolahti, M. Kojo, e K. Raatikainen. F-rto: An enhanced recovery algorithm for tcp retransmission timeouts. *ACM SIGCOMM Computer Communications Review*, 33(2), Abril de 2003.

[94] A. Schiper. Failure detection vs group membership in fault-tolerant distributed systems: hidden trade-offs. Em *Proceedings of Process Algebra and Probabilistic Methods. Performance Modeling and Verification: Second Joint International Workshop PAPM-PROBMIV 2002*, volume LNCS 2399, pp. 1–15, Julho de 2002.

[95] A. Schiper. Group communication: where are we today and future challenges. Em *Proceedings of the Third IEEE International Symposium on Network Computing and Applications (NCA'04)*, pp. 109–117, Agosto–Setembro de 2004.

[96] A. Schiper e S. Toueg. From set membership to group membership: a separation of concerns. *IEEE Transactions on Dependable and Secure Computing*, 3(1):2–12, Janeiro-Março de 2006.

[97] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: a tutorial. *ACM Computing Surveys*, 22(4):299–319, Dezembro de 1990.

[98] I. Sotoma e E. R. M. Madeira. Adaptation - algorithms to adaptive fault monitoring and their implementation on corba. Em *Proceedings of 3rd International Symposium on Distributed Objects and Applications (DOA'01)*, pp. 219–228, Roma, Itália, Setembro de 2001.

[99] I. Sotoma e E. R. M. Madeira. Dpcp (discard past consider present)-a novel approach to adaptive fault detection in distributed systems. Em *Proceedings of the Eighth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2001)*, pp. 76–82, Bolonha, Itália, Outubro–Novembro de 2001.

[100] I. Sotoma e E. R. M. Madeira. A markov model for quality of service of failure detectors in the presence of loss bursts. Em *Proceedings of 18th International Conference on Advanced Information Networking and Applications (AINA 2004)*, volume 2, pp. 62–67, Fukuoka, Japão, Março de 2004.

[101] A. S. Tanenbaum. *Computer networks*. Prentice Hall PTR, New Jersey, Estados Unidos, quarta edição, 2003.

[102] G. Tel. *Introduction to distributed algorithms*. Cambridge University Press, Cambridge, Inglaterra, segunda edição, 2000.

[103] K. S. Trivedi. *Probability and statistics with reliability, queuing and computer science applications*. John Wiley & Sons, New York, segunda edição, 2002.

[104] R. van Renesse, K. P. Birman, e S. Maffeis. Horus: a flexible group communication system. *Communications of the ACM*, 39(4):76–83, Abril de 1996.

[105] P. Veríssimo. Travelling through wormholes: a new look at distributed systems models. *ACM SIGACT News*, 37(1), Março de 2006.

[106] P. Veríssimo e C. Almeida. Quasi-synchronism: a step away from the traditional fault-tolerant real-time system models. *Bulletin of Technical Committee on Operating Systems and Application Environments (TCOS)*, 7(4):35–39, Dezembro de 1995.

[107] P. Veríssimo e A. Casimiro. The timely computing base model and architecture. *IEEE Transactions on Computers*, 51(8):916–930, Agosto de 2002.

[108] P. Veríssimo e L. Rodrigues. *Distributed systems for system architects*. Kluwer Academic Publishers, Norwell, Estados Unidos, 2001.

[109] Y. Wang, E. Anceaume, F. Brasileiro, F. Greve, e M. Hurfin. Solving the group priority inversion problem in a timed asynchronous system. *IEEE Transactions on Computers*, 51(8):900–915, Agosto de 2002.

[110] M. Wiesmann e A. Schiper. Comparison of database replication techniques based on total order broadcast. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):551–566, Abril de 2005.

[111] M. Yajnik, S. B. Moon, J. Kurose, e D. Towsley. Measurement and modeling of the temporal dependence in packet loss. Em *Proceedings of Eighteenth Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM'99)*, pp. 345–352, New York, Estados Unidos, Março de 1999.

[112] L. Zhang. Why tcp timers don't work well. *ACM SIGCOMM Computer Communications Review*, 16(3), Agosto de 1986.

[113] L. Zhang, L. E. Moser, e P. M. Melliar-Smith. Unification of transactions and replication in three-tier architectures based on corba. *IEEE Transactions on Dependable and Secure Computing*, 2(1), Janeiro–Março de 2005.

[114] Y. Zhang. *Characterizing end-to-end Internet performance*. Tese de Doutorado, Cornell University, Cornell, Agosto de 2001.